
MC9S08QL8 MCU Series Reference Manual

Covers: MC9S08QL8
MC9S08QL4

MC9S08QL8RM
Rev. 1
07/2018



Revision History

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

<http://nxp.com>

The following revision history table summarizes changes contained in this document.

Revision Number	Revision Date	Description of Changes
0	06/2018	Initial creation.
1	07/2018	Removed blank pages in the chapters. Updated PTA and PTC registers.

List of Chapters

Chapter Number	Title	Page
Chapter 1	Device Overview	13
Chapter 2	Pins and Connections	17
Chapter 3	Modes of Operation	25
Chapter 4	Memory	37
Chapter 5	Resets, Interrupts, and General System Control	57
Chapter 6	Parallel Input/Output Control	77
Chapter 7	Keyboard Interrupt (S08KBIV2)	89
Chapter 8	Central Processor Unit (S08CPUV5)	95
Chapter 9	Analog Comparator (S08ACMPVLPV1)	115
Chapter 10	Analog-to-Digital Converter (S08ADC12V1)	121
Chapter 11	Internal Clock Source (S08ICSV3)	149
Chapter 12	Modulo Timer (S08MTIMV1)	163
Chapter 13	Real-Time Counter (S08RTCV1)	173
Chapter 14	Serial Communications Interface (S08SCIV4).....	181
Chapter 15	Timer/Pulse-Width Modulator (S08TPMV3)	201
Chapter 16	Development Support	225



Contents

Section Number	Title	Page
Chapter 1		
Device Overview		
1.1	Devices in the MC9S08QL8 Series	13
1.2	MCU Block Diagram	14
1.3	System Clock Distribution	15
Chapter 2		
Pins and Connections		
2.1	Introduction	17
2.2	Device Pin Assignment	17
2.3	Recommended System Connections	19
2.4	Pin Detail	20
2.4.1	Power Pins	20
2.4.2	Oscillator	20
2.4.3	RESET Pin	21
2.4.4	Background / Mode Select (BKGD/MS)	21
2.4.5	General-Purpose I/O (GPIO) and Peripheral Ports	22
Chapter 3		
Modes of Operation		
3.1	Introduction	25
3.2	Features	25
3.3	Run Mode	25
3.3.1	Low Power Run Mode (LPRun)	25
3.4	Active Background Mode	26
3.5	Wait Mode	27
3.5.1	Low Power Wait Mode (LPWait)	27
3.6	Stop Modes	28
3.6.1	Stop2 Mode	29
3.6.2	Stop3 Mode	30
3.6.3	Active BDM Enabled in Stop Mode	31
3.6.4	LVD Enabled in Stop Mode	31
3.6.5	Stop Modes in Low Power Run Mode	31
3.7	Mode Selection	31
3.7.1	On-Chip Peripheral Modules in Stop and Low Power Modes	34
Chapter 4		
Memory		
4.1	MC9S08QL8 Series Memory Map	37

4.2	Reset and Interrupt Vector Assignments	37
4.3	Register Addresses and Bit Assignments	39
4.4	RAM	43
4.5	Flash	44
4.5.1	Features	44
4.5.2	Program and Erase Times	44
4.5.3	Program and Erase Command Execution	45
4.5.4	Burst Program Execution	46
4.5.5	Access Errors	48
4.5.6	Flash Block Protection	48
4.5.7	Vector Redirection	49
4.6	Security	49
4.7	Flash Registers and Control Bits	50
4.7.1	Flash Clock Divider Register (FCDIV)	51
4.7.2	Flash Options Register (FOPT and NVOPT)	52
4.7.3	Flash Configuration Register (FCNFG)	53
4.7.4	Flash Protection Register (FPROT and NVPROT)	53
4.7.5	Flash Status Register (FSTAT)	54
4.7.6	Flash Command Register (FCMD)	55

Chapter 5

Resets, Interrupts, and General System Control

5.1	Introduction	57
5.2	Features	57
5.3	MCU Reset	57
5.4	Computer Operating Properly (COP) Watchdog	58
5.5	Interrupts	59
5.5.1	Interrupt Stack Frame	60
5.5.2	External Interrupt Request (IRQ) Pin	60
5.5.3	Interrupt Vectors, Sources, and Local Masks	61
5.6	Low-Voltage Detect (LVD) System	63
5.6.1	Power-On Reset Operation	63
5.6.2	Low-Voltage Detection (LVD) Reset Operation	63
5.6.3	Low-Voltage Detection (LVD) Interrupt Operation	63
5.6.4	Low-Voltage Warning (LVW) Interrupt Operation	63
5.7	Peripheral Clock Gating	63
5.8	Reset, Interrupt, and System Control Registers and Control Bits	64
5.8.1	Interrupt Pin Request Status and Control Register (IRQSC)	64
5.8.2	System Reset Status Register (SRS)	66
5.8.3	System Background Debug Force Reset Register (SBDFFR)	67
5.8.4	System Options Register 1 (SOPT1)	68
5.8.5	System Options Register 2 (SOPT2)	69
5.8.6	System Device Identification Register (SDIDH, SDIDL)	70
5.8.7	System Power Management Status and Control 1 Register (SPMSC1)	71
5.8.8	System Power Management Status and Control 2 Register (SPMSC2)	72

5.8.9	System Power Management Status and Control 3 Register (SPMSC3)	73
5.8.10	System Clock Gating Control 1 Register (SCGC1)	74
5.8.11	System Clock Gating Control 2 Register (SCGC2)	75

Chapter 6

Parallel Input/Output Control

6.1	Port Data and Data Direction	77
6.2	Pullup, Slew Rate, and Drive Strength	78
6.2.1	Port Internal Pullup Enable	78
6.2.2	Port Slew Rate Enable	78
6.2.3	Port Drive Strength Select	78
6.3	Pin Behavior in Stop Modes	79
6.4	Parallel I/O and Pin Control Registers	79
6.4.1	Port A Registers	80
6.4.2	Port B Registers	82
6.4.3	Port C Registers	85

Chapter 7

Keyboard Interrupt (S08KBIV2)

7.1	Introduction	89
7.1.1	KBI Clock Gating	89
7.1.2	Features	90
7.1.3	Modes of Operation	90
7.1.4	Block Diagram	90
7.2	External Signal Description	91
7.3	Register Definition	92
7.3.1	KBI Interrupt Status and Control Register (KBISC)	92
7.3.2	KBI Interrupt Pin Select Register (KBIPE)	93
7.3.3	KBI Interrupt Edge Select Register (KBIES)	93
7.4	Functional Description	93
7.4.1	Edge Only Sensitivity	94
7.4.2	Edge and Level Sensitivity	94
7.4.3	Pullup/Pulldown Resistors	94
7.4.4	Keyboard Interrupt Initialization	94

Chapter 8

Central Processor Unit (S08CPUV5)

8.1	Introduction	95
8.1.1	Features	95
8.2	Programmer's Model and CPU Registers	96
8.2.1	Accumulator (A)	96
8.2.2	Index Register (H:X)	96
8.2.3	Stack Pointer (SP)	97
8.2.4	Program Counter (PC)	97
8.2.5	Condition Code Register (CCR)	97

8.3	Addressing Modes	99
8.3.1	Inherent Addressing Mode (INH)	99
8.3.2	Relative Addressing Mode (REL)	99
8.3.3	Immediate Addressing Mode (IMM)	99
8.3.4	Direct Addressing Mode (DIR)	99
8.3.5	Extended Addressing Mode (EXT)	100
8.3.6	Indexed Addressing Mode	100
8.4	Special Operations	101
8.4.1	Reset Sequence	101
8.4.2	Interrupt Sequence	101
8.4.3	Wait Mode Operation	102
8.4.4	Stop Mode Operation	102
8.4.5	BGND Instruction	103
8.5	HCS08 Instruction Set Summary	104

Chapter 9

Analog Comparator (S08ACMPVLPV1)

9.1	Introduction	115
9.1.1	ACMP Configuration Information	115
9.1.2	ACMP/TPM Configuration Information	115
9.1.3	ACMP Clock Gating	115
9.1.4	Stop1 Not Available	115
9.1.5	Features	116
9.1.6	Modes of Operation	116
9.1.7	Block Diagram	116
9.2	External Signal Description	117
9.3	Register Definition	117
9.3.1	Status and Control Register (ACMPxSC)	117
9.4	Functional Description	118
9.5	Interrupts	118

Chapter 10

Analog-to-Digital Converter (S08ADC12V1)

10.1	Introduction	121
10.1.1	ADC Clock Gating	121
10.1.2	Module Configurations	121
10.1.3	Features	124
10.1.4	ADC Module Block Diagram	124
10.2	External Signal Description	125
10.2.1	Analog Power (V_{DDA})	126
10.2.2	Analog Ground (V_{SSA})	126
10.2.3	Voltage Reference High (V_{REFH})	126
10.2.4	Voltage Reference Low (V_{REFL})	126
10.2.5	Analog Channel Inputs (ADx)	126
10.3	Register Definition	126

10.3.1	Status and Control Register 1 (ADCSC1)	126
10.3.2	Status and Control Register 2 (ADCSC2)	128
10.3.3	Data Result High Register (ADCRH)	129
10.3.4	Data Result Low Register (ADCRL)	129
10.3.5	Compare Value High Register (ADCCVH)	130
10.3.6	Compare Value Low Register (ADCCVL)	130
10.3.7	Configuration Register (ADCCFG)	130
10.3.8	Pin Control 1 Register (APCTL1)	132
10.3.9	Pin Control 2 Register (APCTL2)	133
10.3.10	Pin Control 3 Register (APCTL3)	134
10.4	Functional Description	135
10.4.1	Clock Select and Divide Control	135
10.4.2	Input Select and Pin Control	136
10.4.3	Hardware Trigger	136
10.4.4	Conversion Control	136
10.4.5	Automatic Compare Function	139
10.4.6	MCU Wait Mode Operation	140
10.4.7	MCU Stop3 Mode Operation	140
10.4.8	MCU Stop2 Mode Operation	141
10.5	Initialization Information	141
10.5.1	ADC Module Initialization Example	141
10.6	Application Information	143
10.6.1	External Pins and Routing	143
10.6.2	Sources of Error	145

Chapter 11

Internal Clock Source (S08ICSV3)

11.1	Introduction	149
11.1.1	DCO Select bits	149
11.1.2	Features	150
11.1.3	Block Diagram	150
11.1.4	Modes of Operation	151
11.2	External Signal Description	152
11.3	Register Definition	152
11.3.1	ICS Control Register 1 (ICSC1)	153
11.3.2	ICS Control Register 2 (ICSC2)	154
11.3.3	ICS Trim Register (ICSTRM)	155
11.3.4	ICS Status and Control (ICSSC)	155
11.4	Functional Description	157
11.4.1	Operational Modes	157
11.4.2	Mode Switching	159
11.4.3	Bus Frequency Divider	160
11.4.4	Low Power Bit Usage	160
11.4.5	DCO Maximum Frequency with 32.768 kHz Oscillator	160
11.4.6	Internal Reference Clock	160

11.4.7 External Reference Clock	161
11.4.8 Fixed Frequency Clock	161
11.4.9 Local Clock	161

Chapter 12

Modulo Timer (S08MTIMV1)

12.1 Introduction	163
12.1.1 MTIM/TPM Configuration Information	163
12.1.2 MTIM Clock Gating	163
12.1.3 Features	164
12.1.4 Modes of Operation	164
12.1.5 Block Diagram	165
12.2 External Signal Description	165
12.3 Register Definition	165
12.3.1 MTIM Status and Control Register (MTIMSC)	167
12.3.2 MTIM Clock Configuration Register (MTIMCLK)	168
12.3.3 MTIM Counter Register (MTIMCNT)	169
12.3.4 MTIM Modulo Register (MTIMMOD)	169
12.4 Functional Description	170
12.4.1 MTIM Operation Example	171

Chapter 13

Real-Time Counter (S08RTCV1)

13.1 Introduction	173
13.1.1 ADC Hardware Trigger	173
13.1.2 RTC Clock Sources	173
13.1.3 RTC Modes of Operation	173
13.1.4 RTC Status after Stop2 Wakeup	173
13.1.5 RTC Clock Gating	173
13.1.6 Features	174
13.1.7 Modes of Operation	174
13.1.8 Block Diagram	175
13.2 External Signal Description	175
13.3 Register Definition	175
13.3.1 RTC Status and Control Register (RTCSC)	176
13.3.2 RTC Counter Register (RTCCNT)	177
13.3.3 RTC Modulo Register (RTCMOD)	177
13.4 Functional Description	177
13.4.1 RTC Operation Example	178
13.5 Initialization/Application Information	179

Chapter 14

Serial Communications Interface (S08SCIV4)

14.1 Introduction	181
14.1.1 SCI Clock Gating	181

14.1.2	Features	183
14.1.3	Modes of Operation	183
14.1.4	Block Diagram	183
14.2	Register Definition	186
14.2.1	SCI Baud Rate Registers (SCIBDH, SCIBDL)	186
14.2.2	SCI Control Register 1 (SCIC1)	187
14.2.3	SCI Control Register 2 (SCIC2)	188
14.2.4	SCI Status Register 1 (SCIS1)	189
14.2.5	SCI Status Register 2 (SCIS2)	191
14.2.6	SCI Control Register 3 (SCIC3)	192
14.2.7	SCI Data Register (SCID)	193
14.3	Functional Description	193
14.3.1	Baud Rate Generation	193
14.3.2	Transmitter Functional Description	194
14.3.3	Receiver Functional Description	195
14.3.4	Interrupts and Status Flags	197
14.3.5	Additional SCI Functions	198

Chapter 15

Timer/Pulse-Width Modulator (S08TPMV3)

15.1	Introduction	201
15.1.1	ACMP/TPM Configuration Information	201
15.1.2	TPM External Clock	201
15.1.3	TPM Pin Repositioning	201
15.1.4	TPM Clock Gating	201
15.1.5	TPMV3 Differences from Previous Versions	201
15.1.6	Migrating from TPMV1	204
15.1.7	Features	205
15.1.8	Modes of Operation	205
15.1.9	Block Diagram	206
15.2	Signal Description	208
15.2.1	Detailed Signal Descriptions	208
15.3	Register Definition	211
15.3.1	TPM Status and Control Register (TPMxSC)	211
15.3.2	TPM-Counter Registers (TPMxCNTH:TPMxCNTL)	212
15.3.3	TPM Counter Modulo Registers (TPMxMODH:TPMxMODL)	213
15.3.4	TPM Channel n Status and Control Register (TPMxCnSC)	214
15.3.5	TPM Channel Value Registers (TPMxCnVH:TPMxCnVL)	215
15.4	Functional Description	217
15.4.1	Counter	217
15.4.2	Channel Mode Selection	218
15.5	Reset Overview	221
15.5.1	General	221
15.5.2	Description of Reset Operation	222
15.6	Interrupts	222

15.6.1 General	222
15.6.2 Description of Interrupt Operation	222

Chapter 16

Development Support

16.1 Introduction	225
16.1.1 Forcing Active Background	225
16.1.2 Module Configuration	225
16.1.3 Features	226
16.2 Background Debug Controller (BDC)	226
16.2.1 BKGD Pin Description	227
16.2.2 Communication Details	227
16.2.3 BDC Commands	230
16.2.4 BDC Hardware Breakpoint	233
16.3 Register Definition	233
16.3.1 BDC Registers and Control Bits	234
16.3.2 System Background Debug Force Reset Register (SBDFFR)	236

Chapter 1

Device Overview

The MC9S08QL8 and MC9S08QL4 are members of the cost-effective, low-power, low voltage, high-performance HCS08 family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced HCS08 core and are available with a variety of modules, memory sizes, memory types, and package types.

1.1 Devices in the MC9S08QL8 Series

Table 1-1 summarizes the feature set available in the MC9S08QL8 series of MCUs.

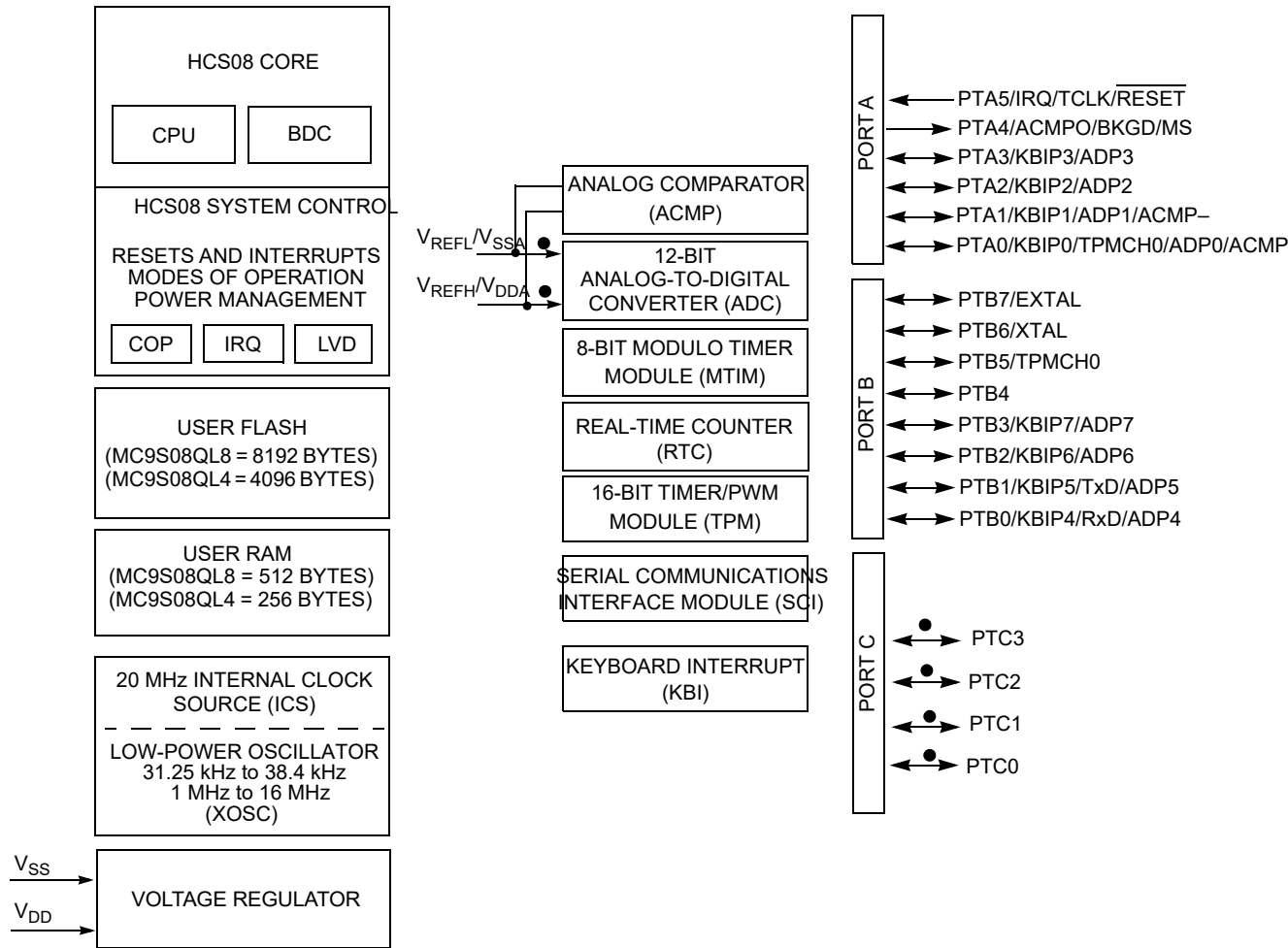
Table 1-1. MC9S08QL8 Series Features by MCU and Package

Feature	MC9S08QL8		MC9S08QL4	
Flash size (bytes)	8192		4096	
RAM size (bytes)	512		256	
Pin quantity	20	16	20	16
ACMP	yes			
ADC channels	8			
ADC Resolution	12	12	12	12
ICS	yes			
MTIM	yes			
IRQ	yes			
KBI	8			
Port I/O ¹	18	14	18	14
RTC	yes			
SCI	yes			
TPM channels	1			
XOSCVLP	yes			
Package	20-pin TSSOP, 16-pin TSSOP			

¹ Port I/O count includes the output-only PTA4 and the input-only PTA5 pins.

1.2 MCU Block Diagram

The block diagram in [Figure 1-1](#) shows the structure of the MC9S08QL8 series MCU.



● pins not available on 16-pin package

¹ V_{DDA}/V_{REFH} and V_{SSA}/V_{REFL} are double bonded to V_{DD} and V_{SS}

Figure 1-1. MC9S08QL8 Series Block Diagram

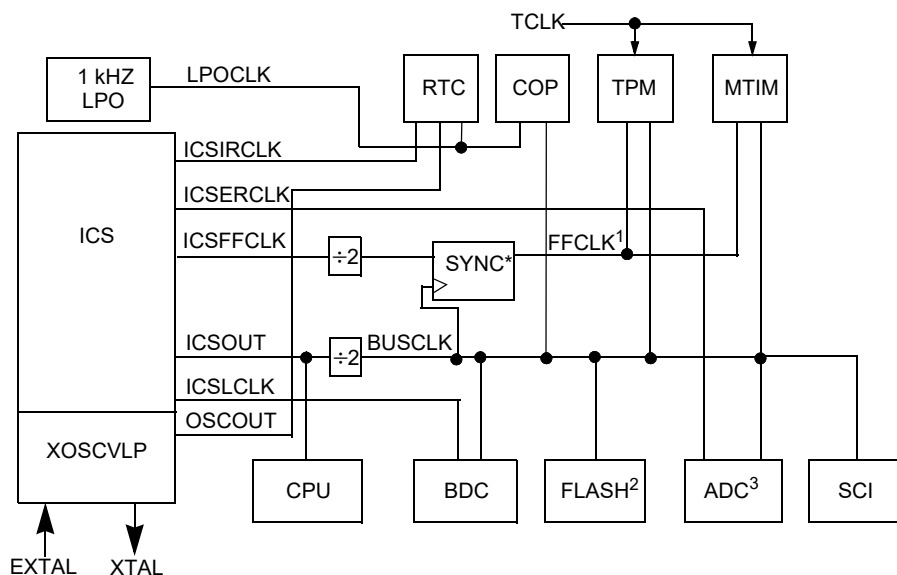
Table 1-2 provides the functional version of the on-chip modules.

Table 1-2. Module Versions

Module	Version
Analog Comparator (ACMPVLP)	1
Analog-to-Digital Converter (ADC12)	1
Central Processor Unit (CPU)	5
Internal Clock Source (ICS)	3
Keyboard Interrupt (KBI)	2
Low Power Oscillator (XOSCVLP)	1
Modulo Timer (MTIM)	1
Real-Time Counter (RTC)	1
Timer Pulse Width Modulator (TPM)	3
Serial Communications Interface (SCI)	4

1.3 System Clock Distribution

Figure 1-2 shows a simplified clock connection diagram. Some modules in the MCU have selectable clock inputs. The clock inputs to the modules indicate the clock(s) that are used to drive the module functions. All memory mapped registers associated with the modules are clocked with BUSCLK.



¹ The fixed frequency clock (FFCLK) is internally synchronized to the bus clock and must not exceed one half of the bus clock frequency. It is XCLK in [Chapter 12, Modulo Timer \(S08MTIMV1\)](#).

² Flash has frequency requirements for program and erase operation. See *MC9S08QL8 Series Data Sheet* for details.

³ ADC has minimum and maximum frequency requirements. See [Chapter 10, Analog-to-Digital Converter \(S08ADC12V1\)](#) and *MC9S08QL8 Data Sheet* for details.

Figure 1-2. System Clock Distribution Diagram

The ICS supplies the following clock sources:

- ICSOUT — This clock source is used as the CPU clock and is divided by 2 to generate the peripheral bus clock, BUSCLK. Control bits in the ICS control registers determine which of three clock sources is connected:
 - Internal reference clock
 - External reference clock
 - Frequency-locked loop (FLL) output

See [Chapter 11, Internal Clock Source \(S08ICSV3\)](#) for details on configuring the ICSOUT clock.

- ICSLCLK — This clock source is derived from the digitally controlled oscillator, DCO, of the ICS when the ICS is configured to run off of the internal or external reference clock. Development tools can select this internal self-clocked source (~ 8 MHz) to speed up BDC communications in systems where the bus clock is slow.
- ICSECLK — This is the external reference clock and can be selected as the alternate clock for the ADC module. [Section 11.4.7, External Reference Clock](#) explains the ICSECLK in more detail. See [Chapter 10, Analog-to-Digital Converter \(S08ADC12V1\)](#) for more information regarding the use of ICSECLK with these modules.
- ICSIRCLK — This is the internal reference clock and can be selected as the real-time counter clock source. [Chapter 11, Internal Clock Source \(S08ICSV3\)](#) explains the ICSIRCLK in more detail. See [Chapter 13, Real-Time Counter \(S08RTCV1\)](#) for more information regarding the use of ICSIRCLK.
- ICSFFCLK — This generates the fixed frequency clock (FFCLK) after being synchronized to the bus clock. It can be selected as clock source for the TPM module and MTIM module. The frequency of the ICSFFCLK is determined by the settings of the ICS. See [Section 11.4.8, Fixed Frequency Clock](#) in [Chapter 11, Internal Clock Source \(S08ICSV3\)](#) for details.
- LPOCLK — This clock is generated from an internal low power oscillator that is completely independent of the ICS module. The LPOCLK can be selected as the clock source to the RTC or COP modules. See [Chapter 13, Real-Time Counter \(S08RTCV1\)](#) and [Section 5.4, Computer Operating Properly \(COP\) Watchdog](#) for details on using the LPOCLK with these modules.
- OSCOUT — This is the output of the XOSCVLP module and can be selected as the real-time counter clock source.
- TCLK — TCLK is the optional external clock source for the TPM and MTIM modules. The TCLK must be limited to 1/4th the frequency of the bus clock for synchronization. See [Chapter 15, Timer/Pulse-Width Modulator \(S08TPMV3\)](#) and [Chapter 12, Modulo Timer \(S08MTIMV1\)](#) for more details.

Chapter 2

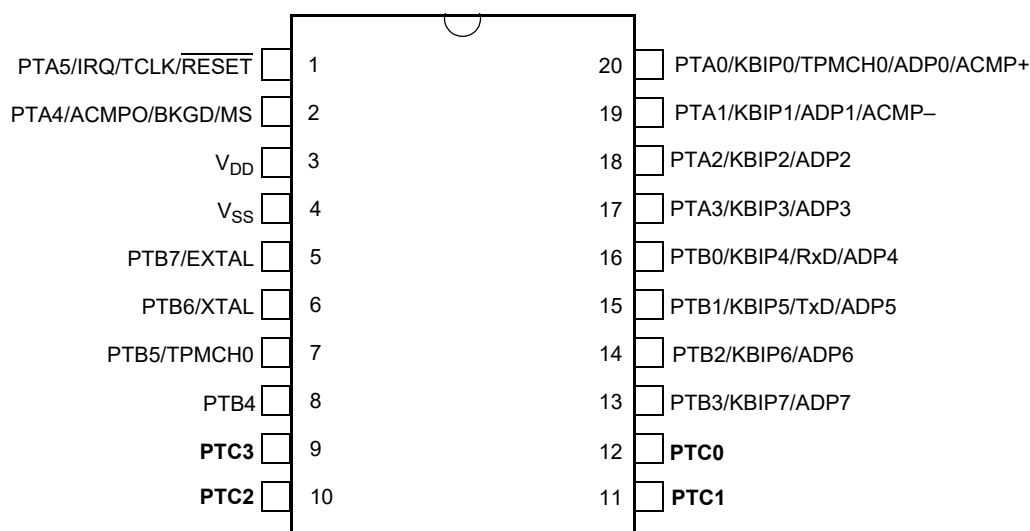
Pins and Connections

2.1 Introduction

This chapter describes signals that connect to package pins. It includes pinout diagrams, a signal properties table, and a detailed signal discussion.

2.2 Device Pin Assignment

Figure 2-1 and Figure 2-2 shows the pin assignments in the packages for the MC9S08QL8 series devices.



Pins shown in bold type are lost in the next lower pin count package.

Figure 2-1. MC9S08QL8 Series in 20-Pin TSSOP Package

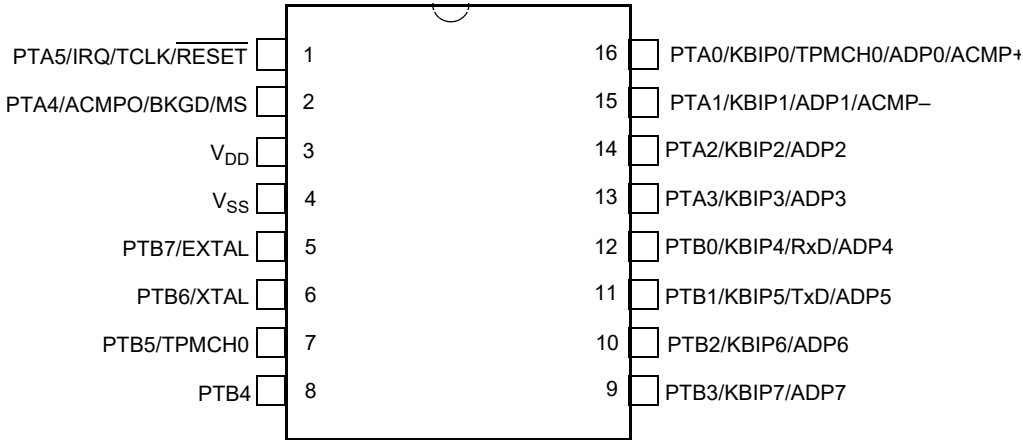
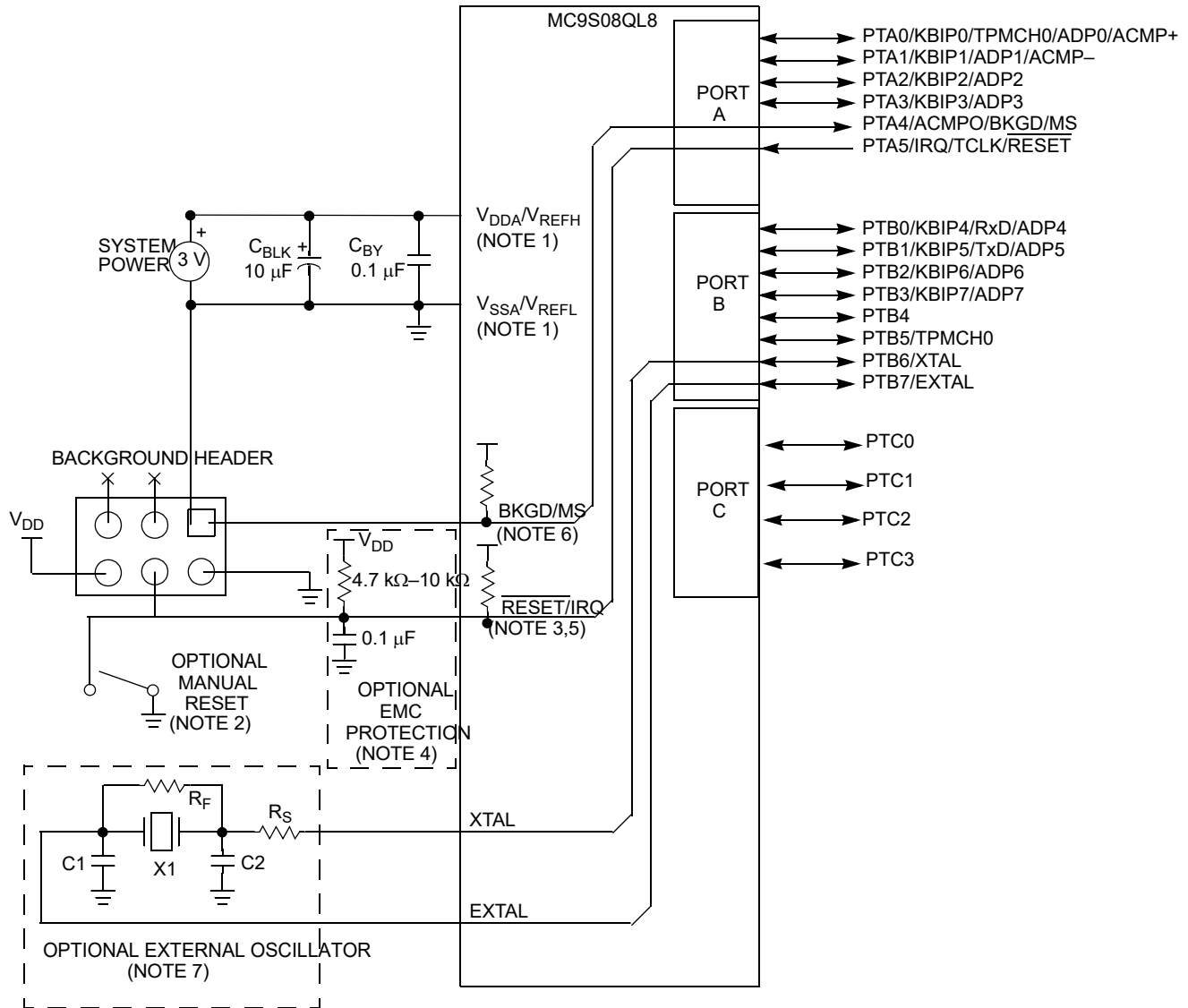


Figure 2-2. MC9S08QL8 Series in 16-Pin TSSOP Packages

2.3 Recommended System Connections

Figure 2-3 shows pin connections that are common to MC9S08QL8 series application systems.



NOTES:

- ¹ V_{DDA}/V_{REFH} and V_{SSA}/V_{REFL} are tied with V_{DD} and V_{SS} respectively.
- ² $\overline{RESET}$ pin can only be used to reset into user mode, you can not enter BDM using $\overline{RESET}$ pin. BDM can be entered by holding MS low during POR or writing a 1 to BDFR in SBDFFR with MS low after issuing BDM command.
- ³ When PTA5 is configured as IRQ, pin has optional internal pullup device.
- ⁴ RC filter on $\overline{RESET}/IRQ$ pin is recommended for noisy environments.
- ⁵ When PTA5 is configured as $\overline{RESET}$, pin becomes bi-directional with output being open-drain drive containing an internal pullup device.
- ⁶ When PTA4 is configured as BKGD, pin becomes bi-directional
- ⁷ When using the XOSCVLP module in low range and low power mode, the external components R_F , R_S , $C1$ and $C2$ are not required.

Figure 2-3. Basic System Connections

2.4 Pin Detail

This section provides a detailed description of system connections.

2.4.1 Power Pins

V_{DD} and V_{SS} are the primary power supply pins for the MCU. This voltage source supplies power to all I/O buffer circuitry and to an internal voltage regulator. The internal voltage regulator provides a regulated lower-voltage source for the CPU and other internal circuitry of the MCU.

Typically, application systems have two separate capacitors across the power pins. In this case, there must be a bulk electrolytic capacitor, such as a 10- μ F tantalum capacitor, to provide bulk charge storage for the overall system and a 0.1- μ F ceramic bypass capacitor located as near to the MCU power pins as practical to suppress high-frequency noise.

V_{DDA} and V_{SSA} are the analog power supply pins for the MCU. This voltage source supplies power to the ADC and ACMP modules.

The V_{REFH} and V_{REFL} pins are the voltage reference high and voltage reference low inputs, respectively for the ADC module. For this MCU, V_{DDA} shares the V_{REFH} pin and they are double bonded to the V_{DD} pin. For this MCU, V_{SSA} shares the V_{REFL} pin, and they are double bonded to the V_{SS} pin.

2.4.2 Oscillator

Immediately after reset, the MCU uses an internally generated clock provided by the internal clock source (ICS) module. The oscillator can be configured to run in stop2 or stop3 modes. For more information on the ICS, see [Chapter 11, Internal Clock Source \(S08ICSV3\)](#).

The oscillator (XOSCVLP) in this MCU is a Pierce oscillator that can accommodate a crystal or ceramic resonator. An external clock source can optionally be connected to the EXTAL input pin.

Refer to [Figure 2-3](#) for the following discussion. R_S (when used) and R_F must be low-inductance resistors such as carbon composition resistors. Wire-wound resistors and some metal film resistors have too much inductance. C1 and C2 normally must be high-quality ceramic capacitors that are specifically designed for high-frequency applications.

R_F provides a bias path to keep the EXTAL input in its linear range during crystal startup; its value is not generally critical. Typical systems use 1 M Ω to 10 M Ω . Higher values are sensitive to humidity and lower values reduce gain and (in extreme cases) could prevent startup.

C1 and C2 are typically in the 5 pF to 25 pF range and are chosen to match the requirements of a specific crystal or resonator. Be sure to consider printed circuit board (PCB) capacitance and MCU pin capacitance when selecting C1 and C2. The crystal manufacturer typically specifies a load capacitance which is the series combination of C1 and C2 (which are usually the same size). As a first-order approximation, use 10 pF as an estimate of combined pin and PCB capacitance for each oscillator pin (EXTAL and XTAL).

When using the oscillator in low range and low gain mode, the external components R_S , R_F , C1 and C2 are not required.

2.4.3 $\overline{\text{RESET}}$ Pin

After a power-on reset (POR), the PTA5/IRQ/TCLK/ $\overline{\text{RESET}}$ pin defaults to a general-purpose input port pin, PTA5. Setting RSTPE in SOPT1 configures the pin to be the $\overline{\text{RESET}}$ pin with an open-drain drive containing an internal pullup device. Once PTA5 is configured as $\overline{\text{RESET}}$, the pin will function as $\overline{\text{RESET}}$ until the next POR or LVD reset. When enabled, the $\overline{\text{RESET}}$ pin can be used to reset the MCU from an external source when the pin is driven low.

Internal power-on reset and low-voltage reset circuitry typically make external reset circuitry unnecessary. This pin is normally connected to the standard 6-pin background debug connector so a development system can directly reset the MCU system. A manual external reset can be added by supplying a simple switch to ground (pull reset pin low to force a reset).

Whenever any non-POR reset is initiated (whether from an external signal or from an internal system), the enabled $\overline{\text{RESET}}$ pin is driven low for about 34 bus cycles. The reset circuitry decodes the cause of reset and records it by setting a corresponding bit in the system reset status register (SRS).

NOTE

This pin does not contain a clamp diode to V_{DD} and must not be driven above V_{DD} .

NOTE

The voltage on the internally pulled up $\overline{\text{RESET}}$ pin when measured will be below V_{DD} . The internal gates connected to this pin are pulled to V_{DD} . If the $\overline{\text{RESET}}$ pin is required to drive to a V_{DD} level an external pullup must be used.

NOTE

In EMC-sensitive applications, an external RC filter is recommended on the $\overline{\text{RESET}}$ pin, if enabled. See [Figure 2-3](#) for an example.

2.4.4 Background / Mode Select (BKGD/MS)

During a power-on-reset (POR) or a background debug force reset (see [Section 5.8.3, System Background Debug Force Reset Register \(SBD FR\)](#) for more information), the PTA4/ACMPO/BKGD/MS pin functions as a mode select pin. Immediately after any reset, the pin functions as the background pin and can be used for background debug communication. When enabled as the BKGD/MS pin (BKGDPE = 1), an internal pullup device is automatically enabled.

The background debug communication function is enabled when BKGDPE in SOPT1 is set. BKGDPE is set following any reset of the MCU and must be cleared to use the PTA4/ACMPO/BKGD/MS pin's alternative pin functions.

If nothing is connected to this pin, the MCU will enter normal operating mode at the rising edge of the internal reset after a POR, or force a background debug controller (BDC) reset. If a debug system is connected to the standard 6-pin background debug header, it can hold BKGD/MS low during a POR or after issuing a background debug force reset. This will force the MCU into active background mode.

NOTE

A resistive or capacitive load on the PTA4/ACMPO/BKGD/MS pin could cause the MCU to enter active background mode on a POR if the pin voltage rises slower than V_{DD} .

The BKGD/MS pin is used primarily with BDC communications, and features a custom protocol that uses 16 clock cycles of the target MCU's BDC clock per bit time. The target MCU's BDC clock can run as fast as the bus clock, so no significant capacitance must be connected to the BKGD/MS pin that could interfere with background serial communications.

Although the BKGD/MS pin is a pseudo open-drain pin, the background debug communication protocol provides brief, actively driven, high speedup pulses to ensure fast rise times. Small capacitances from cables and the absolute value of the internal pullup device play a minimal role in determining rise and fall times on the BKGD/MS pin.

2.4.5 General-Purpose I/O (GPIO) and Peripheral Ports

The MC9S08QL8 series of MCUs support up to 16 general-purpose I/O pins, 1 input-only pin, and 1 output-only pin, which are shared with on-chip peripheral functions (timer, ADC, ACMP, etc.). The GPIO output-only (PTA4/ACMPO/BKGD/MS) and input-only (PTA5/IRQ/TCLK/ $\overline{\text{RESET}}$) pins are bi-directional when configured as BKGD and $\overline{\text{RESET}}$, respectively.

When a port pin is configured as a general-purpose output or a peripheral uses the port pin as an output, software can select one of two drive strengths and enable or disable slew rate control. When a port pin is configured as a general-purpose input or a peripheral uses the port pin as an input, software can enable a pullup device.

PTA5 is a special I/O pin. When the PTA5/IRQ/TCLK/ $\overline{\text{RESET}}$ pin is configured as PTA5 input with the pullup enabled, the voltage observed on the pin will not be pulled to V_{DD} . However, the internal voltage on the PTA5 node will be at V_{DD} .

When an on-chip peripheral system is controlling a pin, data direction control bits still determine what is read from port data registers even though the peripheral module controls the pin direction by controlling the enable for the pin's output buffer. For information about controlling these pins as general-purpose I/O pins, see [Chapter 6, Parallel Input/Output Control](#).

NOTE

To avoid extra current drain from floating input pins, the reset initialization routine in the application program must enable on-chip pullup devices or change the direction of unused or non-bonded pins to outputs so they do not float.

When using the 16-pin device, the user must either enable on-chip pullup devices or change the direction of non-bonded PTC3–PTC0 pins to outputs so the pins do not float.

Table 2-1. Pin Availability by Package Pin-Count

Pin Number		<-- Lowest Priority --> Highest				
20	16	Port Pin	Alt 1	Alt 2	Alt 3	Alt 4
1	1	PTA5	IRQ	TCLK	RESET	—
2	2	PTA4	ACMPO	BKGD	MS	—
3	3	—	—	—	—	V _{DD}
4	4	—	—	—	—	V _{SS}
5	5	PTB7	—	—	—	EXTAL
6	6	PTB6	—	—	—	XTAL
7	7	PTB5	TPMCH0 ¹	—	—	—
8	8	PTB4	—	—	—	—
9	—	PTC3	—	—	—	—
10	—	PTC2	—	—	—	—
11	—	PTC1	—	—	—	—
12	—	PTC0	—	—	—	—
13	9	PTB3	KBIP7	—	ADP7	—
14	10	PTB2	KBIP6	—	ADP6	—
15	11	PTB1	KBIP5	TxD	ADP5	—
16	12	PTB0	KBIP4	RxD	ADP4	—
17	13	PTA3	KBIP3	—	ADP3	—
18	14	PTA2	KBIP2	—	ADP2	—
19	15	PTA1	KBIP1	—	ADP1 ²	ACMP ⁻²
20	16	PTA0	KBIP0	TPMCH0	ADP0 ²	ACMP ⁺ ²

¹ TPMCH0 pin can be repositioned at PTB5 using TPMCH0PS in SOPT2, default reset location is PTA0.

² If ADC and ACMP are enabled, both modules will have access to the pin.

Chapter 3

Modes of Operation

3.1 Introduction

The operating modes of the MC9S08QL8 series are described in this chapter. Entry into each mode, exit from each mode, and functionality while in each of the modes are described.

3.2 Features

- Active background mode for code development
- Run mode — CPU clocks can be run at full speed and the internal supply is fully regulated.
- LPRUN mode — CPU and peripheral clocks are restricted to 125 kHz at maximum and the internal voltage regulator is in standby
- Wait mode — CPU shuts down to conserve power; system clocks are running and full regulation is maintained
- LPWAIT mode — CPU shuts down to conserve power; peripheral clocks are restricted to 125 kHz in maximum and the internal voltage regulator is in standby
- Stop modes — System clocks are stopped and voltage regulator is in standby
 - Stop3 — All internal circuits are powered for fast recovery
 - Stop2 — Partial power down of internal circuits, RAM content is retained, I/O states held

3.3 Run Mode

This is the normal operating mode for the MC9S08QL8 series. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at 0xFFFFE–0xFFFF after reset.

3.3.1 Low Power Run Mode (LPRun)

In the low power run mode, the on-chip voltage regulator is put into its standby state. This state uses the minimum power consumption necessary for CPU functionality. Power consumption is most reduced by disabling the clocks to all unused peripherals by clearing the corresponding bits in the SCGC1 and SCGC2 registers.

Before entering this mode, the following conditions must be met:

- FBELP is the selected clock mode for the ICS.
- The HGO bit in the ICSC2 register is clear.
- The bus frequency is less than 125 kHz.

- If enabled, the ADC must be configured to use the asynchronous clock source, ADACK, to meet the ADC minimum frequency requirements. The bandgap channel cannot be converted in low power run mode.
- The LVD and LVW must be disabled by clearing either the LVDE or LVDSE bits in the SPMSC1 register.
- Flash programming/erasing is not allowed.
- ACMP option to compare to internal bandgap reference is not allowed in LPRUN and LPWAIT.

Once these conditions are met, low power run mode can be entered by setting the LPR bit in the SPMSC2 register.

To re-enter standard run mode, clear the LPR bit. The LPRS bit in the SPMSC2 register is a read-only status bit that can be used to determine if the regulator is in full regulation mode or not. When LPRS is '0', the regulator is in full regulation mode and the MCU can run at full speed in any clock mode.

3.3.1.1 Interrupts in Low Power Run Mode

Low power run mode provides the option to return to full regulation if any interrupt occurs. This is done by setting the LPWUI bit in the SPMSC2 register. The ICS can then be set for full speed immediately in the interrupt service routine.

If the LPWUI bit is clear, interrupts will be serviced in low power run mode.

If the LPWUI bit is set, LPR and LPRS bits will be cleared and interrupts will be serviced with the regulator in full regulation.

3.3.1.2 Resets in Low Power Run Mode

Any reset will exit low power run mode, clear the LPR and LPRS bits, and return the device to normal run mode.

3.4 Active Background Mode

The active background mode functions are managed through the BDC in the HCS08 core. The BDC provides the means for analyzing MCU operation during software development.

Active background mode is entered in any of five ways:

- When the BKGD/MS pin is low during POR
- When the BKGD/MS pin is low immediately after issuing a background debug force reset (see [Section 5.8.3, System Background Debug Force Reset Register \(SBD FR\)](#)).
- When a BACKGROUND command is received through the BKGD/MS pin
- When a BGND instruction is executed
- When encountering a BDC breakpoint

After entering active background mode, the CPU is held in a suspended state while it waits for serial background commands instead of executing instructions from the user application program.

Background commands are of two types:

- Non-intrusive commands, defined as commands that can be issued while the user program is running. Non-intrusive commands can be issued through the BKGD pin while the MCU is in run mode; non-intrusive commands can also be executed when the MCU is in the active background mode. Non-intrusive commands include:
 - Memory access commands
 - Memory-access-with-status commands
 - BDC register access commands
 - The BACKGROUND command
- Active background commands, which can only be executed while the MCU is in active background mode. Active background commands include commands to:
 - Read or write CPU registers
 - Trace one user program instruction at a time
 - Leave active background mode to return to the user application program (GO)

The active background mode is used to program a bootloader or user application program into the flash program memory before the MCU is operated in run mode for the first time. When the MC9S08QL8 series are shipped from the NXP Semiconductors factory, the flash program memory is erased by default unless specifically noted. As a result, no program can be executed in run mode until the flash memory is initially programmed. The active background mode can also be used to erase and reprogram the flash memory after it has been previously programmed.

For additional information about the active background mode, refer to the [Chapter 16, Development Support](#).

3.5 Wait Mode

Wait mode is entered by executing a WAIT instruction. Upon execution of the WAIT instruction, the CPU enters a low-power state in which it is not clocked. The I bit in CCR is cleared when the CPU enters the wait mode, enabling interrupts. When an interrupt request occurs, the CPU exits the wait mode and resumes processing, beginning with the stacking operations that lead to the interrupt service routine.

While the MCU is in wait mode, there are some restrictions on which background debug commands can be used. Only the BACKGROUND command and memory-access-with-status commands are available when the MCU is in wait mode. The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in either stop or wait mode. The BACKGROUND command can be used to wake the MCU from wait mode and enter active background mode.

3.5.1 Low Power Wait Mode (LPWait)

Low power wait mode is entered by executing a WAIT instruction while the MCU is in low power run mode. In the low power wait mode, the on-chip voltage regulator remains in its standby state (as in the low power run mode). This state uses the minimum power consumption necessary for most modules to maintain functionality. Power consumption is most reduced by disabling the clocks to all unused peripherals by clearing the corresponding bits in the SCGC register.

The same restrictions on the low power run mode apply to low power wait mode.

3.5.1.1 Interrupts in Low Power Wait Mode

If the LPWUI bit is set when the WAIT instruction is executed, then the voltage regulator will return to full regulation when wait mode is exited. The ICS can be set for full speed immediately in the interrupt service routine.

If the LPWUI bit is clear when the WAIT instruction is executed, an interrupt will return the device to low power run mode.

If the LPWUI bit is set when the WAIT instruction is executed, an interrupt will return the device to normal run mode with full regulation and the LPR and LPRS bits will be cleared.

3.5.1.2 Resets in Low Power Wait Mode

Any reset will exit low power wait mode, clear the LPR and LPRS bits, and return the device to normal run mode.

3.6 Stop Modes

One of two stop modes (stop2 or stop3) is entered upon execution of a STOP instruction when the STOPE bit in the system option 1 register (SOPT1) is set. In both stop modes, the bus and CPU clocks are halted. In stop3 the voltage regulator is in standby. In stop2 the voltage regulator is in partial powerdown. The ICS module can be configured to leave the reference clocks running. See [Chapter 11, Internal Clock Source \(S08ICSV3\)](#) for more information.

If the STOPE bit is not set when the CPU executes a STOP instruction, the MCU will not enter either stop mode and an illegal opcode reset is forced. The stop modes are selected by setting the appropriate bits in the System Power Management Status and Control 2 Register (SPMSC2).

[Table 3-1](#) shows all of the control bits that affect stop mode selection and the mode selected under various conditions. The selected mode is entered following the execution of a STOP instruction.

Table 3-1. Stop Mode Selection

Register	SOPT1	BDCSCR	SPMSC1		SPMSC2	Stop Mode
Bit name	STOPE	ENBDM ¹	LVDE	LVDSE	PPDC	
	0	x	x		x	Stop modes disabled; illegal opcode reset if STOP instruction executed
	1	1	x		x	Stop3 with BDM enabled ²
	1	0	Both bits must be 1		x	Stop3 with voltage regulator active
	1	0	Either bit a 0		0	Stop3
	1	0	Either bit a 0		1	Stop2

¹ ENBDM is located in the BDCSCR which is accessible only through BDC commands, see [Chapter 16, Development Support](#).

² When in stop3 mode with BDM enabled, The S_{IDD} will be near R_{IDD} levels because internal clocks are enabled.

3.6.1 Stop2 Mode

Stop2 mode is entered by executing a STOP instruction under the conditions as shown in Table 3-1. Most of the internal circuitry of the MCU is powered off in stop2 with the exception of the RAM and optionally the RTC and low power oscillator. Upon entering stop2, all I/O pin control signals are latched so that the pins retain their states during stop2.

Exit from stop2 is performed by asserting the wakeup pin (PTA5/IRQ/TCLK/ $\overline{\text{RESET}}$) on the MCU.

NOTE

PTA5/IRQ/TCLK/ $\overline{\text{RESET}}$ is an active low wakeup. To avoid an immediate exit from stop 2, either the internal pullup must be enabled prior to executing a STOP instruction or an external pullup must be connected. If PTA5/IRQ/TCLK/ $\overline{\text{RESET}}$ is not to be used as the wakeup pin, configure it as PTA5 with the pullup enabled.

In addition, the real-time counter (RTC) can wake the MCU from stop2, if enabled.

Upon wakeup from stop2 mode, the MCU starts up as from a power-on reset (POR):

- All module control and status registers are reset, except for SPMSC1-SPMSC3, RTCSC, RTCCNT and RTCMOD.
- The LVD reset function is enabled and the MCU remains in the reset state if V_{DD} is below the LVD trip point
- The CPU takes the reset vector

In addition to the above, upon waking from stop2, the PPDF bit in SPMSC2 is set. This flag is used to direct user code to go to a stop2 recovery routine. PPDF remains set and the I/O pin states remain latched until a 1 is written to PPDACK in SPMSC2.

If using the low power oscillator during stop2, the user must reconfigure the ICSC2 register which contains oscillator control bits before PPDACK is written.

To maintain I/O states for pins that were configured as GPIO before entering stop2, the user must restore the contents of the I/O port registers, which have been saved in RAM, to the port registers before writing to the PPDACK bit. If the port registers are not restored from RAM before writing to PPDACK, then the pins will switch to their reset states when PPDACK is written.

For pins that were configured as peripheral I/O, the user must reconfigure the peripheral module that interfaces to the pin before writing to the PPDACK bit. If the peripheral module is not enabled before writing to PPDACK, the pins will be controlled by their associated port control registers when the I/O latches are opened.

3.6.1.1 Stop2 Mode Recovery Time

The stop2 recovery time is defined as the interval from the exit trigger to the first opcode fetch. There are three main components to this wakeup time: the voltage regulator recovery time, the clock source start up time, and the reset processing time.

The voltage regulator recovery time (t_{VRR}) is provided in the data sheet. This time is not influenced by the clock source frequency or V_{DD} and is therefore relatively consistent.

Since exiting from stop2 causes the MCU to wake as if a POR occurred, the standard reset processing will always occur which takes about 150 ICSOUT cycles after the clock source has started. Therefore, the equation for stop2 recovery time is

$$\text{Stop2 recovery time} = t_{VRR} + \text{clock start up time} + 150 \text{ ICSOUT cycles.} \quad \text{Eqn. 3-1}$$

Since ICSOUT defaults to FLL output running at 8.4 MHz during a reset, and the FLL takes about 1 ms to start outputting a clock signal (although it won't be stable initially) Equation 3-5 simplifies to

$$\text{Stop2 recovery time} = t_{VRR} + 1 \mu\text{sec} + 17.9 \mu\text{s.} \quad \text{Eqn. 3-2}$$

3.6.2 Stop3 Mode

Stop3 mode is entered by executing a STOP instruction under the conditions shown in Table 3-1. The states of all of the internal registers and logic, RAM contents, and I/O pin states are maintained.

Stop3 can be exited by asserting $\overline{\text{RESET}}$, or by an interrupt from one of the following sources: the RTC, LVD, LVW, ADC, ACMP, IRQ, SCI or the KBI.

If stop3 is exited by means of the $\overline{\text{RESET}}$ pin, then the MCU is reset and operation will resume after taking the reset vector. Exit by means of one of the internal interrupt sources results in the MCU taking the appropriate interrupt vector.

3.6.2.1 Stop3 Mode Recovery Time

The stop3 recovery time is defined as the interval from the exit trigger to the first opcode fetch. There are three main components to this wakeup time: the voltage regulator recovery time, the clock source start up time, and the reset or interrupt processing time.

The voltage regulator recovery time (t_{VRR}) is provided in the data sheet. This time is not influenced by the clock source frequency or V_{DD} and is therefore relatively consistent.

When an interrupt is used as the exit trigger, the clock must restart and ICSOUT must oscillate six times before the interrupt processing begins. The interrupt processing requires 11 bus cycles (22 ICSOUT cycles) for the stacking and vector fetch. Therefore, the first opcode of the interrupt service routine (ISR) will begin after

$$\text{Stop3 recovery time} = t_{VRR} + \text{clock start up time} + 28 \text{ ICSOUT cycles.} \quad \text{Eqn. 3-3}$$

The clock source start up time is dependent on the clock mode selected when the MCU enters stop mode. When the FLL output is selected as the clock source, the FLL starts up within a microsecond at roughly the same frequency as before stop mode is entered. Typical start up time for the internal reference is given in the data sheet. Typical start up times for the crystal oscillator are also given in the data sheet.

Assuming the FLL is the selected clock source upon entering stop3 and the FLL is configured for a 20 MHz ICSOUT frequency, then Equation 3-6 simplifies to

$$\text{Stop3 recovery time} = t_{VRR} + 1 \mu\text{sec} + 1.4 \mu\text{sec} \quad \text{Eqn. 3-4}$$

When reset is used as the exit trigger, more time is required for the reset processing, so Equation 3-3 becomes

$$\text{Stop3 recovery time} = t_{VRR} + \text{clock start up time} + 162 \text{ ICSOUT cycles.} \quad \text{Eqn. 3-5}$$

Since ICSOUT defaults to FLL output running at 8.4 MHz during a reset, Equation 3-5 simplifies to

$$\text{Stop3 recovery time} = t_{VRR} + 1 \mu\text{sec} + 19.3 \mu\text{sec.} \quad \text{Eqn. 3-6}$$

3.6.3 Active BDM Enabled in Stop Mode

Entry into the active background mode from run mode is enabled if the ENBDM bit in BDCSCR is set. This register is described in [Chapter 16, Development Support](#). If ENBDM is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode. Because of this, background debug communication remains possible. In addition, the voltage regulator does not enter its low-power standby state but maintains full internal regulation. If the user attempts to enter stop2 with ENBDM set, the MCU will instead enter stop3.

Most background commands are not available in stop mode. The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in either stop or wait mode. The BACKGROUND command can be used to wake the MCU from stop and enter active background mode if the ENBDM bit is set. After entering background debug mode, all background commands are available.

3.6.4 LVD Enabled in Stop Mode

The LVD system is capable of generating either an interrupt or a reset when the supply voltage drops below the LVD voltage. If the LVD is enabled in stop (LVDE and LVDSE bits in SPMSC1 both set) the voltage regulator remains active during stop mode. If the user attempts to enter stop2 with the LVD enabled for stop, the MCU will instead enter stop3.

3.6.5 Stop Modes in Low Power Run Mode

Stop2 mode cannot be entered from low power run mode. If the PPDC bit is set, then the LPR bit cannot be set. Likewise, if the LPR bit is set, the PPDC bit cannot be set.

Stop3 mode can be entered from low power run mode by executing the STOP instruction while in low power run. Exiting stop3 with a reset will put the device back into normal run mode. If LPWUI is clear, interrupts will exit stop3 mode, return the device to low power run mode, and then service the interrupt. If LPWUI is set, interrupts will exit stop3 mode, put the device into normal run mode, clear LPR and LPRS bits, and then service the interrupt.

3.7 Mode Selection

Several control signals are used to determine the current operating mode of the device. [Table 3-2](#) shows the conditions for each of the device's operating modes.

Table 3-2. Power Mode Selections

Mode of Operation	BDCSCR BDM	SPMSC1 PMC		SPMSC2 PMC		CPU & Periph CLKs	Affects on Sub-System	
	ENBDM ¹	LVDE	LVDSE	LPR	PPDC		BDM Clock	Voltage Regulator
RUN mode	0	x	x	0	x	on. ICS in any mode.	off	on
		1	1	1				
	1	x	x	x			on	
LPRUN mode	0	0	x	1	0	low freq required. ICS in FBELP mode only.	off	standby
		1	0					
WAIT mode - (Assumes WAIT instruction executed.)	0	x	x	0	x	CPU clock is off; peripheral clocks on. ICS state same as RUN mode.	off	on
		1	1	1				
	1	x	x	x			on	
LPWAIT mode - (Assumes WAIT instruction executed.)	0	0	x	1	0	CPU clock is off; peripheral clocks at low speed. ICS in FBELP mode.	off	standby
		1	0					
STOP3 - (Assumes STOPE bit is set and STOP instruction executed.) Note that STOP3 is used in place of STOP2 if the BDM or LVD is enabled.	0	0	x	x	0	ICS in STOP. ICSECLK, ICSIRCLK, and OSCOUT optionally on ²	off	standby
	0	1	0	x	0		off	
	0	1	1	x	x		off	
	1	x	x	x	x	ICSLCLK still active.	on	on - stop currents will be increased
STOP2 - (Assumes STOPE bit is set and STOP instruction executed.) If BDM or LVD is enabled, STOP3 will be invoked rather than STOP2.	0	0	x	0	1	OSCOUT optionally on ^{2,3}	off	partial powerdown
		1	0					

¹ ENBDM is located in the BDC status and control register (BDCSCR) which is write accessible only through BDC commands.

² Configured within the ICS module based on the settings of IREFSTEN, EFRESTEN, IRCLKEN and ERCLKEN.

³ In stop2, CPU, Flash, ICS and all peripheral modules are powered down except for the RTC.

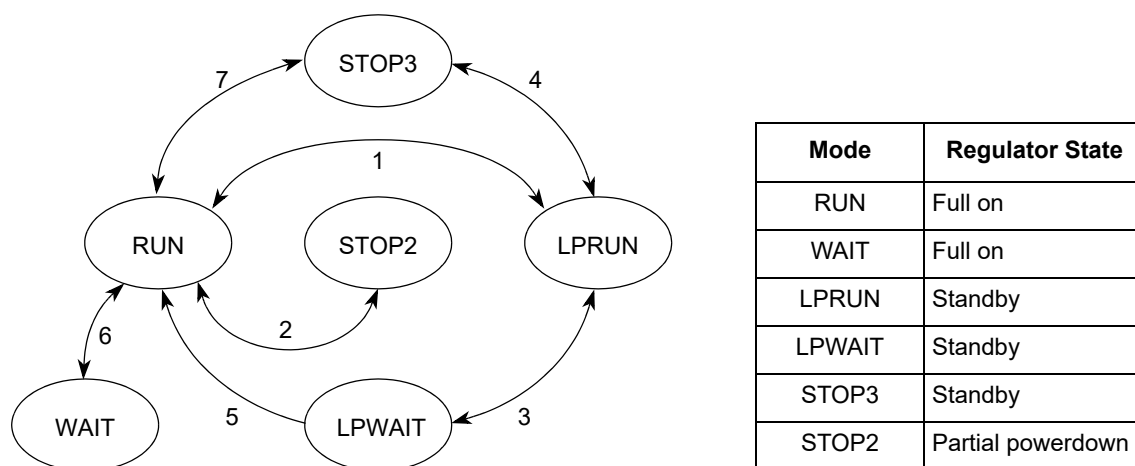


Figure 3-1. Allowable Power Mode Transitions for the MC9S08QL8 Series

Figure 3-1 illustrates mode state transitions allowed between the legal states shown in Table 3-1. PTA5/IRQ/TCLK/ $\overline{\text{RESET}}$ must be asserted low in order to exit stop2. Interrupts suffice for the other stop and wait modes.

Table 3-3 defines triggers for the various state transitions shown in Figure 3-1.

Table 3-3. Triggers for Transitions Shown in Figure 3-1

Transition #	From	To	Trigger
1	RUN	LPRUN	Configure settings shown in Table 3-1, switch LPR=1 last
	LPRUN	RUN	Clear LPR
			Interrupt when LPWUI=1
2	RUN	STOP2	Pre-configure settings shown in Table 3-1, issue STOP instruction
	STOP2	RUN	assert zero on PTA5/IRQ/TCLK/ $\overline{\text{RESET}}$ ¹ , reload environment from RAM
3	LPRUN	LPWAIT	WAIT instruction
	LPWAIT	LPRUN	Interrupt when LPWUI=0
4	LPRUN	STOP3	STOP instruction
	STOP3	LPRUN	Interrupt when LPWUI=0
5	LPWAIT	RUN	Interrupt when LPWUI=1
	RUN	LPWAIT	Not supported
6	RUN	WAIT	WAIT instruction
	WAIT	RUN	Interrupt or reset

Table 3-3. Triggers for Transitions Shown in Figure 3-1 (continued)

Transition #	From	To	Trigger
7	STOP3	RUN	Interrupt (if LPR = 0, or LPR = 1 and LPWUI =1) or reset
	RUN	STOP3	STOP instruction

¹ An analog connection from this pin to the on-chip regulator will wake the regulator, which will then initiate a power-on-reset sequence.

3.7.1 On-Chip Peripheral Modules in Stop and Low Power Modes

When the MCU enters any stop mode, system clocks to the internal peripheral modules are stopped. Even in the exception case (ENBDM = 1), where clocks to the background debug logic continue to operate, clocks to the peripheral systems are halted to reduce power consumption. Refer to [Section 3.6.1, Stop2 Mode](#) and [Section 3.6.2, Stop3 Mode](#) for specific information on system behavior in stop modes.

When the MCU enters LPWait or LPRun modes, system clocks to the internal peripheral modules continue based on the settings of the clock gating control registers (SCGC1 and SCGC2).

Table 3-4. Stop and Low Power Mode Behavior

Peripheral	Mode			
	Stop2	Stop3	LPWait	LPRun
CPU	Off	Standby	Standby	On
RAM	Standby	Standby	Standby	On
Flash	Off	Standby	Standby	On
Port I/O Registers	Off	Standby	Standby	On
ADC	Off	Optionally On ¹	Optionally On ¹	Optionally On ¹
ACMP	Off	Optionally On ²	Optionally On	Optionally On
BDM	Off ³	Optionally On	Off ⁴	Off ⁴
COP	Off	Off	Optionally On	Optionally On
ICS	Off	Optionally On ⁵	On ⁶	On ⁶
IRQ	Off	Optionally On	Optionally On	Optionally On
KBI	Off	Optionally On	Optionally On	Optionally On
LVD/LVW	Off ⁷	Optionally On	Off ⁸	Off ⁸
RTC	Optionally On	Optionally On	Optionally On	Optionally On
MTIM	Off	Standby	Optionally On	Optionally On
SCI	Off	Standby	Optionally On	Optionally On
TPM	Off	Standby	Optionally On	Optionally On
Voltage Regulator	Partial Powerdown	Optionally On ⁹	Standby	Standby
XOSCVLP	Optionally On ¹⁰	Optionally On ¹⁰	Optionally On	Optionally On
I/O Pins	States Held	Peripheral Control	Peripheral Control	On

¹ Requires the asynchronous ADC clock. For stop3, LVD must be enabled to run in stop if converting the bandgap channel. The bandgap channel cannot be converted in LPRun or LPWait.

² LVD must be enabled to run in stop if using the bandgap as a reference.

- ³ If ENBDM is set when entering stop2, the MCU will actually enter stop3.
- ⁴ If ENBDM is set when entering LPRun or LPWait, the MCU will actually stay in run mode or enter wait mode, respectively.
- ⁵ IRCLKEN and IREFSTEN set in ICSC1, else in standby.
- ⁶ ICS must be configured for FBELP, bus frequency limited to 125 kHz in LPRUN or LPWAIT.
- ⁷ If LVDSE is set when entering stop2, the MCU will actually enter stop3.
- ⁸ If LVDSE is set when entering LPRun or LPWait, the MCU will actually enter run or wait mode, respectively.
- ⁹ Requires the LVD to be enabled, else in standby. See [Section 3.6.4, LVD Enabled in Stop Mode](#).
- ¹⁰ ERCLKEN and EREFSTEN set in ICSC2, else in standby.

Chapter 4

Memory

4.1 MC9S08QL8 Series Memory Map

As shown in [Figure 4-1](#), on-chip memory in the MC9S08QL8 series of MCUs consists of RAM, flash program memory for nonvolatile data storage, and I/O and control/status registers. The registers are divided into three groups:

- Direct-page registers (0x0000 through 0x005F)
- High-page registers (0x1800 through 0x184F)
- Nonvolatile registers (0xFFB0 through 0xFFBF)

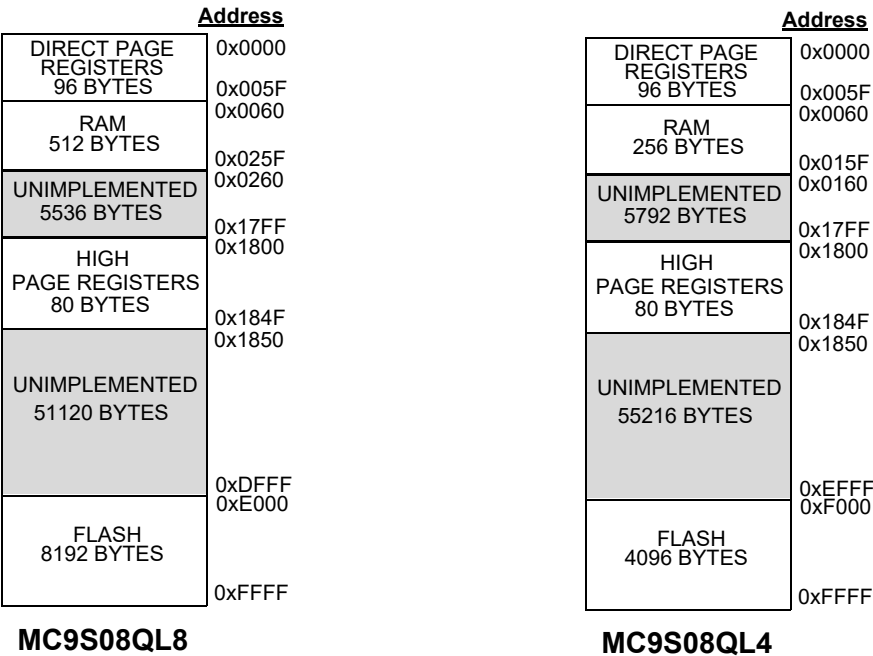


Figure 4-1. MC9S08QL8 Series Memory Maps

4.2 Reset and Interrupt Vector Assignments

[Table 4-1](#) shows address assignments for reset and interrupt vectors. The vector names shown in this table are the labels used in the NXP Semiconductors provided equate file for the MC9S08QL8 series.

Table 4-1. Reset and Interrupt Vectors

Address (High/Low)	Vector	Vector Name
0xFFC0:0xFFC1 ↕ 0xFFCC:0xFFCD	Unused Vector Space (available for user program)	
0xFFCE:0xFFCF	RTC	Vrtc
0xFFD0:0xFFD1	Reserved	—
0xFFD2:0xFFD3	Reserved	—
0xFFD4:0xFFD5	Reserved	—
0xFFD6:0xFFD7	ACMP	Vacmp
0xFFD8:0xFFD9	ADC Conversion	Vadc
0xFFDA:0xFFDB	KBI Interrupt	Vkeyboard
0xFFDC:0xFFDD	Reserved	—
0xFFDE:0xFFDF	SCI Transmit	Vscitx
0xFFE0:0xFFE1	SCI Receive	Vscirx
0xFFE2:0xFFE3	SCI Error	Vscierr
0xFFE4:0xFFE5	Reserved	—
0xFFE6:0xFFE7	MTIM Overflow	Vmtim
0xFFE8:0xFFE9	Reserved	—
0xFFEA:0xFFEB	Reserved	—
0xFFEC:0xFFED	Reserved	—
0xFFEE:0xFFEF	Reserved	—
0xFFFF0:0xFFFF1	TPM Overflow	Vtpmovf
0xFFFF2:0xFFFF3	Reserved	—
0xFFFF4:0xFFFF5	Reserved	—
0xFFFF6:0xFFFF7	TPM Channel 0	Vtpmch0
0xFFFF8:0xFFFF9	Low Voltage Detect or Low Voltage Warning	Vlvd
0xFFFFA:0xFFFFB	IRQ	Virq
0xFFFFC:0xFFFFD	SWI	Vswi
0xFFFFE:0xFFFFF	Reset	Vreset

4.3 Register Addresses and Bit Assignments

The registers in the MC9S08QL8 series are divided into these groups:

- Direct-page registers are the ones located in the first 96 locations in the memory map; these are accessible with efficient direct addressing mode instructions.
- High-page registers are used much less often, so they are located above 0x1800 in the memory map. This leaves more room in the direct page for more frequently used registers and RAM.
- The nonvolatile register area consists of a block of 16 locations in flash memory at 0xFFB0–0xFFBF. Nonvolatile register locations include:
 - NVPROT and NVOPT are loaded into working registers at reset
 - An 8-byte backdoor comparison key that optionally allows a user to gain controlled access to secure memory

Because the nonvolatile register locations are flash memory, they must be erased and programmed like other flash memory locations.

Direct-page registers can be accessed with efficient direct addressing mode instructions. Bit manipulation instructions can be used to access any bit in any direct-page register. [Table 4-2](#) is a summary of all user-accessible direct-page registers and control bits.

The direct-page registers in [Table 4-2](#) can use the more efficient direct addressing mode, which requires only the lower byte of the address. Because of this, the lower byte of the address in column one is shown in bold text. In [Table 4-3](#) and [Table 4-4](#), the whole address in column one is shown in bold. In [Table 4-2](#), [Table 4-3](#), and [Table 4-4](#), the register names in column two are shown in bold to set them apart from the bit names to the right. Cells that are not associated with named bits are shaded. A shaded cell with a 0 indicates this unused bit always reads as a 0. Shaded cells with dashes indicate unused or reserved bit locations that could read as 1s or 0s. When writing to these bits, write a 0 unless otherwise specified.

Table 4-2. Direct-Page Register Summary (Sheet 1 of 2)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0000	PTAD	0	0	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
0x0001	PTADD	0	0	0	0	PTADD3	PTADD2	PTADD1	PTADD0
0x0002	PTBD	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
0x0003	PTBDD	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
0x0004	PTCD	0	0	0	0	PTCD3	PTCD2	PTCD1	PTCD0
0x0005	PTCDD	0	0	0	0	PTCDD3	PTCDD2	PTCDD1	PTCDD0
0x0006	Reserved	—	—	—	—	—	—	—	—
—		—	—	—	—	—	—	—	—
0x000B	Reserved	—	—	—	—	—	—	—	—
—		—	—	—	—	—	—	—	—
0x000C	KBISC	0	0	0	0	KBF	KBACK	KBIE	KBIMOD
0x000D	KBIPE	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x000E	KBIES	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KB1EDG3	KB1EDG2	KB1EDG1	KB1EDG0
0x000F	IRQSC	0	IRQPDD	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD
0x0010	ADCSC1	COCO	AIEN	ADCO	ADCH				
0x0011	ADCSC2	ADACT	ADTRG	ACFE	ACFGT	0	0	R	R
0x0012	ADCRH	0	0	0	0	ADR11	ADR10	ADR9	ADR8
0x0013	ADCRL	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
0x0014	ADCCVH	0	0	0	0	ADCV11	ADCV10	ADCV9	ADCV8
0x0015	ADCCVL	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
0x0016	ADCCFG	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x0017	APCTL1	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
0x0018	Reserved	—	—	—	—	—	—	—	—
—		—	—	—	—	—	—	—	—
0x0019	Reserved	—	—	—	—	—	—	—	—
—		—	—	—	—	—	—	—	—
0x001A	ACMPSC	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD1	ACMOD0
0x001B	Reserved	—	—	—	—	—	—	—	—
—		—	—	—	—	—	—	—	—
0x001F	Reserved	—	—	—	—	—	—	—	—
—		—	—	—	—	—	—	—	—
0x0020	SCIBDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0021	SCIBDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x0022	SCIC1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x0023	SCIC2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x0024	SCIS1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x0025	SCIS2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x0026	SCIC3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x0027	SCID	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
0x0028	Reserved	—	—	—	—	—	—	—	—
—		—	—	—	—	—	—	—	—
0x0037	Reserved	—	—	—	—	—	—	—	—
—		—	—	—	—	—	—	—	—
0x0038	ICSC1	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN

Table 4-2. Direct-Page Register Summary (Sheet 2 of 2)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0039	ICSC2	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFTEN
0x003A	ICSTRM	TRIM							
0x003B	ICSSC	DRST/DRS		DMX32	IREFST	CLKST		OSCINIT	FTRIM
0x003C	MTIMSC	TOF	TOIE	TRST	TSTP	0	0	0	0
0x003D	MTIMCLK	0	0	CLKS		PS			
0x003E	MTIMCNT	COUNT							
0x003F	MTIMMOD	MOD							
0x0040	TPMSC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0041	TPMCNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0042	TPMCNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0043	TPMMODH	Bit 15	14	13	12	11	10	9	Bit 8
0x0044	TPMMODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0045	TPMC0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0046	TPMC0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0047	TPMC0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0048	Reserved	—	—	—	—	—	—	—	—
—		—	—	—	—	—	—	—	—
0x005F		—	—	—	—	—	—	—	—

High-page registers, shown in [Table 4-3](#), are accessed much less often than other I/O and control registers so they have been located outside the direct addressable memory space, starting at 0x1800.

Table 4-3. High-Page Register Summary (Sheet 1 of 2)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1800	SRS	POR	PIN	COP	ILOP	ILAD	0	LVD	0
0x1801	SBDPR	0	0	0	0	0	0	0	BDFR
0x1802	SOPT1	COPE	COPT	STOPE	0	0	0	BKGDPE	RSTPE
0x1803	SOPT2	COPCLKS	0	0	TPMCH0PS	0	0	0	ACIC
0x1804–	Reserved	—	—	—	—	—	—	—	—
0x1805		—	—	—	—	—	—	—	—
0x1806	SDIDH	—	—	—	—	ID11	ID10	ID9	ID8
0x1807	SDIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
0x1808	SPMSC1	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	0	BGBE
0x1809	SPMSC2	LPR	LPRS	LPWUI	0	PPDF	PPDACK	PPDE	PPDC
0x180A	Reserved	—	—	—	—	—	—	—	—
0x180B	SPMSC3	LVWF	LVWACK	—	—	LVWIE	0	0	0
0x180C	Reserved	—	—	—	—	—	—	—	—
0x180D	Reserved	—	—	—	—	—	—	—	—
0x180E	SCGC1	MTIM	1	TPM	ADC	1	1	1	SCI
0x180F	SCGC2	1	FLS	IRQ	KBI	ACMP	RTC	1	1

Table 4-3. High-Page Register Summary (Sheet 2 of 2)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1810– 0x181F	Reserved	—	—	—	—	—	—	—	—
0x1820	FCDIV	DIVLD	PRDIV8	DIV					
0x1821	FOPT	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
0x1822	Reserved	—	—	—	—	—	—	—	—
0x1823	FCNFG	0	0	KEYACC	0	0	0	0	0
0x1824	FPROT	FPS							FPDIS
0x1825	FSTAT	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
0x1826	FCMD	FCMD							
0x1827– 0x182F	Reserved	—	—	—	—	—	—	—	—
0x1830	RTCSC	RTIF	RTCLKS		RTIE	RTCPS			
0x1831	RTCCNT	RTCCNT							
0x1832	RTCMOD	RTCMOD							
0x1833– 0x183F	Reserved	—	—	—	—	—	—	—	—
0x1840	PTAPE	0	0	PTAPE5	0	PTAPE3	PTAPE2	PTAPE1	PTAPE0
0x1841	PTASE	0	0	0	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
0x1842	PTADS	0	0	0	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
0x1843	Reserved	—	—	—	—	—	—	—	—
0x1844	PTBPE	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x1845	PTBSE	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x1846	PTBDS	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
0x1847	Reserved	—	—	—	—	—	—	—	—
0x1848	PTCPE	0	0	0	0	PTCPE3	PTCPE2	PTCPE1	PTCPE0
0x1849	PTCSE	0	0	0	0	PTCSE3	PTCSE2	PTCSE1	PTCSE0
0x184A	PTCDS	0	0	0	0	PTCDS3	PTCDS2	PTCDS1	PTCDS0
0x184B– 0x184F	Reserved	—	—	—	—	—	—	—	—

Several reserved flash memory locations, shown in [Table 4-4](#), are used for storing values used by several registers. These registers include an 8-byte backdoor key, NVBACKKEY, which can be used to gain access to secure memory resources. During reset events, the contents of NVPROT and NVOPT in the reserved flash memory are transferred into corresponding FPROT and FOPT registers in the high-page registers area to control security and block protection options.

The factory ICS trim value is stored in the IFR and will be loaded into the ICSTRM and ICSSC registers after any reset. The internal reference trim values stored in flash, TRIM and FTRIM, can be programmed by third party programmers and must be copied into the corresponding ICS registers by user code to override the factory trim.

Table 4-4. Reserved Flash Memory Addresses

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0xFFAE	Reserved for Storage of FTRIM	0	0	0	0	0	0	0	FTRIM
0xFFAF	Reserved for Storage of ICSTRM	TRIM							
0xFFB0– 0xFFB7	NVBACKKEY	8-Byte Comparison Key							
0xFFB8– 0xFFBC	Reserved	—	—	—	—	—	—	—	—
0xFFBD	NVPROT	FPS							FPDIS
0xFFBE	Reserved	—	—	—	—	—	—	—	—
0xFFBF	NVOPT	KEYEN	FNORED	0	0	0	0	SEC	

Provided the key enable (KEYEN) bit is 1, the 8-byte comparison key can be used to temporarily disengage memory security. This key mechanism can be accessed only through user code running in secure memory. (A security key cannot be entered directly through background debug commands.) This security key can be disabled completely by programming the KEYEN bit to 0. If the security key is disabled, the only way to disengage security is by mass erasing the flash if needed (normally through the background debug interface) and verifying that flash is blank. To avoid returning to secure mode after the next reset, program the security bits (SEC) to the unsecured state (1:0).

4.4 RAM

The MC9S08QL8 series include static RAM. The locations in RAM below 0x0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

At power-on, the contents of RAM are uninitialized. RAM data is unaffected by any reset provided that the supply voltage does not drop below the minimum value for RAM retention (V_{RAM}).

For compatibility with M68HC05 MCUs, the HCS08 resets the stack pointer to 0x00FF. In the MC9S08QL8 series, it is usually best to reinitialize the stack pointer to the top of the RAM so the direct-page RAM can be used for frequently accessed RAM variables and bit-addressable program variables. Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM in the NXP Semiconductors-provided equate file).

```
LDHX    #RamLast+1    ;point one past RAM
TXS          ;SP<-(H:X-1)
```

When security is enabled, the RAM is considered a secure memory resource and is not accessible through background debug mode (BDM) or through code executing from non-secure memory. See [Section 4.6, Security](#) for a detailed description of the security feature.

4.5 Flash

The flash memory is primarily for program storage. In-circuit programming allows the operating program to be loaded into the flash memory after final assembly of the application product. It is possible to program the entire array through the single-wire background debug interface. Because no special voltages are needed for flash erase and programming operations, in-application programming is also possible through other software-controlled communication paths. For a more detailed discussion of in-circuit and in-application programming, refer to the *HCS08 Family Reference Manual, Volume I*, NXP Semiconductors document order number HCS08RMv1.

4.5.1 Features

Features of the flash memory include:

- Flash size
 - MC9S08QL8: 8,192 bytes (16 pages of 512 bytes each)
 - MC9S08QL4: 4,096 bytes (8 pages of 512 bytes each)
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection
- Security feature for flash and RAM
- Auto power-down for low-frequency read accesses

4.5.2 Program and Erase Times

Before any program or erase command can be accepted, the flash clock divider register (FCDIV) must be written to set the internal clock for the flash module to a frequency (f_{FCLK}) between 150 kHz and 200 kHz (see [Section 4.7.1, Flash Clock Divider Register \(FCDIV\)](#)). This register can be written only once, so normally this write is done during reset initialization. FCDIV cannot be written if the access error flag, FACCERR in FSTAT, is set. The user must ensure that FACCERR is not set before writing to the FCDIV register. One period of the resulting clock ($1/f_{FCLK}$) is used by the command processor to time program and erase pulses. An integer number of these timing pulses is used by the command processor to complete a program or erase command.

[Table 4-5](#) shows program and erase times. The bus clock frequency and FCDIV determine the frequency of FCLK (f_{FCLK}). The time for one cycle of FCLK is $t_{FCLK} = 1/f_{FCLK}$. The times are shown as a number of cycles of FCLK and as an absolute time for the case where $t_{FCLK} = 5 \mu s$. Program and erase times shown include overhead for the command state machine and enabling and disabling of program and erase voltages.

Table 4-5. Program and Erase Times

Parameter	Cycles of FCLK	Time if FCLK = 200 kHz
Byte program	9	45 μ s
Byte program (burst)	4	20 μ s ¹
Page erase	4000	20 ms
Mass erase	20,000	100 ms

¹ Excluding start/end overhead

4.5.3 Program and Erase Command Execution

The steps for executing any of the commands are listed below. The FCDIV register must be initialized and any error flags cleared before beginning command execution. The command execution steps are:

1. Write a data value to an address in the flash array. The address and data information from this write is latched into the flash interface. This write is a required first step in any command sequence. For erase and blank check commands, the value of the data is not important. For page erase commands, the address may be any address in the 512-byte page of flash to be erased. For mass erase and blank check commands, the address can be any address in the flash memory. Whole pages of 512 bytes are the smallest block of flash that may be erased.

NOTE

Do not program any byte in the flash more than once after a successful erase operation. Reprogramming bits to a byte that is already programmed is not allowed without first erasing the page in which the byte resides or mass erasing the entire flash memory. Programming without first erasing may disturb data stored in the flash.

2. Write the command code for the desired command to FCMD. The five valid commands are blank check (0x05), byte program (0x20), burst program (0x25), page erase (0x40), and mass erase (0x41). The command code is latched into the command buffer.
3. Write a 1 to the FCBEF bit in FSTAT to clear FCBEF and launch the command (including its address and data information).

A partial command sequence can be aborted manually by writing a 0 to FCBEF any time after the write to the memory array and before writing the 1 that clears FCBEF and launches the complete command. Aborting a command in this way sets the FACCERR access error flag, which must be cleared before starting a new command.

A strictly monitored procedure must be obeyed or the command will not be accepted. This minimizes the possibility of any unintended change to the flash memory contents. The command complete flag (FCCF) indicates when a command is complete. The command sequence must be completed by clearing FCBEF to launch the command. [Figure 4-2](#) is a flowchart for executing all of the commands except for burst programming. The FCDIV register must be initialized before using any flash command. This must be done only once following a reset.

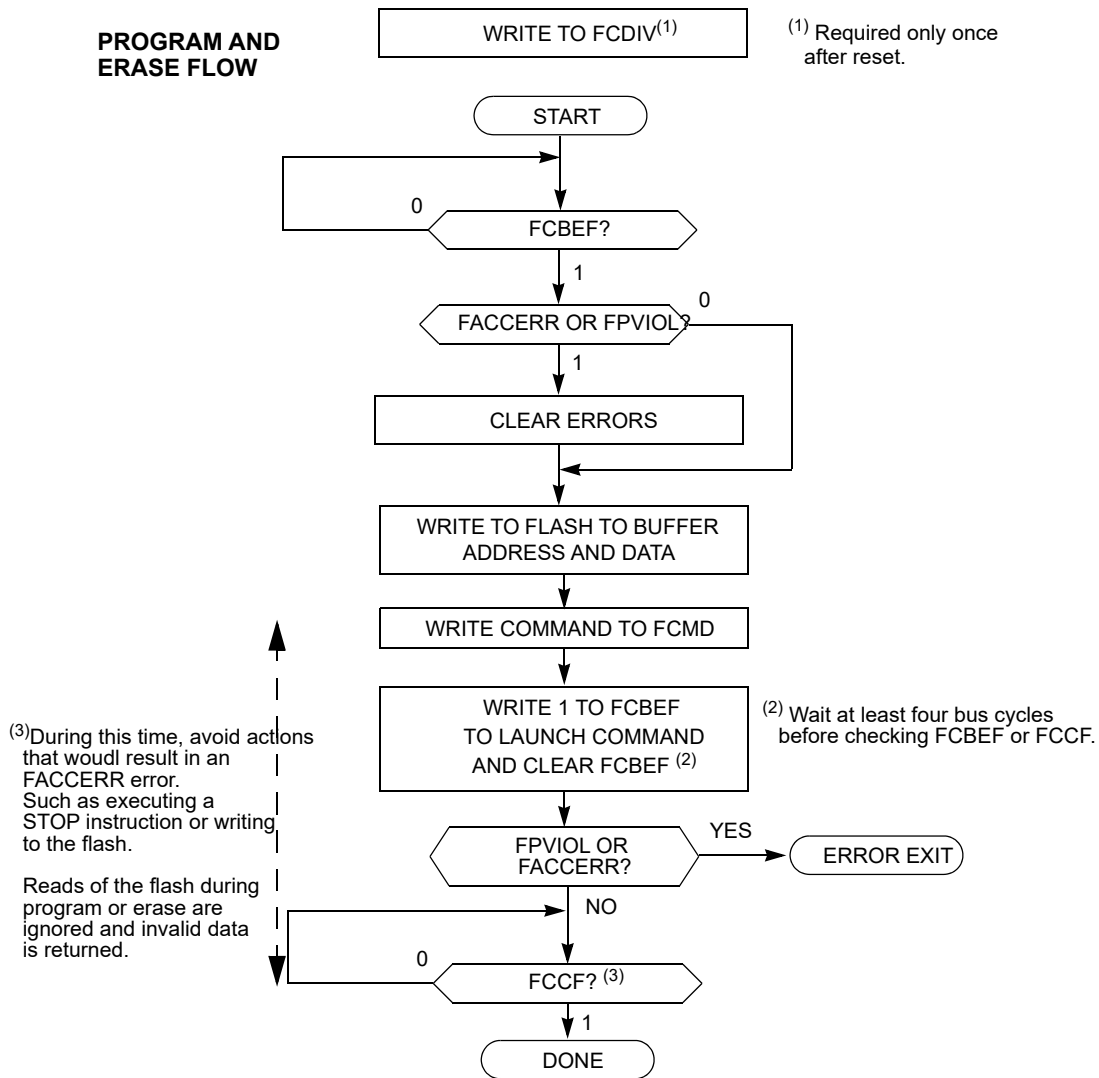


Figure 4-2. Flash Program and Erase Flowchart

4.5.4 Burst Program Execution

The burst program command is used to program sequential bytes of data in less time than would be required using the standard program command. This is possible because the high voltage to the flash array does not need to be disabled between program operations. Ordinarily, when a program or erase command is issued, an internal charge pump associated with the flash memory must be enabled to supply high voltage to the array. Upon completion of the command, the charge pump is turned off. When a burst program command is issued, the charge pump is enabled and then remains enabled after completion of the burst program operation if these two conditions are met:

- The next burst program command has been queued before the current program operation has completed.

- The next sequential address selects a byte on the same physical row as the current byte being programmed. A row of flash memory consists of 64 bytes. A byte within a row is selected by addresses A5 through A0. A new row begins when addresses A5 through A0 are all zero.

The first byte of a series of sequential bytes being programmed in burst mode will take the same amount of time to program as a byte programmed in standard mode. Subsequent bytes will program in the burst program time provided that the conditions above are met. In the case the next sequential address is the beginning of a new row, the program time for that byte will be the standard time instead of the burst time. This is because the high voltage to the array must be disabled and then enabled again. If a new burst command has not been queued before the current command completes, then the charge pump will be disabled and high voltage removed from the array.

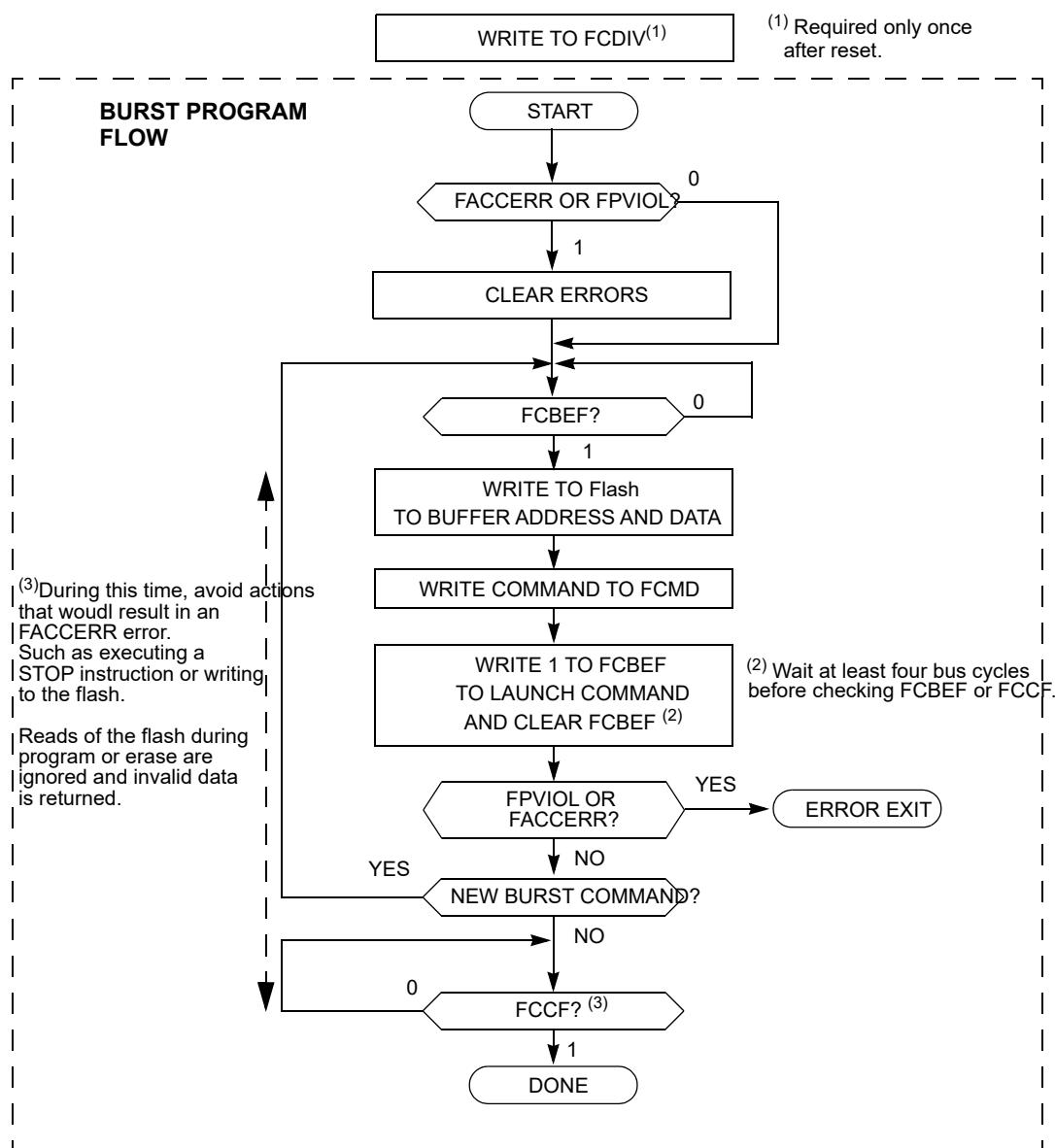


Figure 4-3. Flash Burst Program Flowchart

4.5.5 Access Errors

An access error occurs whenever the command execution protocol is violated.

Any of the following actions will cause the access error flag (FACCERR) in FSTAT to be set. FACCERR must be cleared by writing a 1 to FACCERR in FSTAT before any command can be processed.

- Writing to a flash address before the internal flash clock frequency has been set by writing to the FCDIV register
- Writing to a flash address while FCBEF is not set (A new command cannot start until the command buffer is empty.)
- Writing a second time to a flash address before launching the previous command (There is only one write to flash for every command.)
- Writing a second time to FCMD before launching the previous command (There is only one write to FCMD for every command.)
- Writing to any flash control register other than FCMD after writing to a flash address
- Writing any command code to FCMD other than the five allowed codes (0x05, 0x20, 0x25, 0x40, or 0x41)
- Accessing (read or write) any flash control register other than the write to FSTAT (to clear FCBEF and launch the command) after writing the command to FCMD
- The MCU enters stop mode while a program or erase command is in progress (The command is aborted.)
- Writing the byte program, burst program, or page erase command code (0x20, 0x25, or 0x40) with a background debug command while the MCU is secured (The background debug controller can only do blank check and mass erase commands when the MCU is secure.)
- Writing 0 to FCBEF to cancel a partial command

4.5.6 Flash Block Protection

The block protection feature prevents the protected region of flash from program or erase changes. Block protection is controlled through the flash protection register (FPROT). When enabled, block protection begins at any 512 byte boundary below the last address of flash, 0xFFFF. (See [Section 4.7.4, Flash Protection Register \(FPROT and NVPROT\)](#).)

After exit from reset, FPROT is loaded with the contents of the NVPROT location, which is in the nonvolatile register block of the flash memory. FPROT cannot be changed directly from application software to prevent runaway programs from altering the block protection settings. Because NVPROT is within the last 512 bytes of flash, if any amount of memory is protected, NVPROT is itself protected and cannot be altered (intentionally or unintentionally) by the application software. FPROT can be written through background debug commands, which allows a protected flash memory to be erased and reprogrammed.

The block protection mechanism is illustrated in [Figure 4-4](#). The FPS bits are used as the upper bits of the last address of unprotected memory. This address is formed by concatenating FPS7:FPS1 with logic 1 bits as shown. For example, to protect the last 1536 bytes of memory (addresses 0xFA00 through 0xFFFF), the FPS bits must be set to 1111 100, which results in the value 0xF9FF as the last address of unprotected

memory. In addition to programming the FPS bits to the appropriate value, FPDIS (bit 0 of NVPROT) must be programmed to logic 0 to enable block protection. Therefore the value 0xF8 must be programmed into NVPROT to protect addresses 0xFA00 through 0xFFFF.

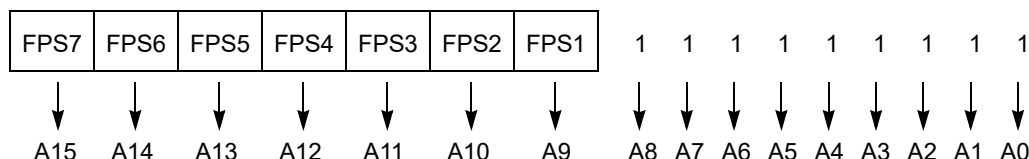


Figure 4-4. Block Protection Mechanism

One use of block protection is to block protect an area of flash memory for a bootloader program. This bootloader program then can be used to erase the rest of the flash memory and reprogram it. Because the bootloader is protected, it remains intact even if MCU power is lost in the middle of an erase and reprogram operation.

4.5.7 Vector Redirection

Whenever any block protection is enabled, the reset and interrupt vectors will be protected. Vector redirection allows users to modify interrupt vector information without unprotecting bootloader and reset vector space. Vector redirection is enabled by programming the FNORED bit in the NVOPT register located at address 0xFFBF to zero. For redirection to occur, at least some portion but not all of the flash memory must be block protected by programming the NVPROT register located at address 0xFFBD. All of the interrupt vectors (memory locations 0xFFC0–0xFFFD) are redirected, though the reset vector (0xFFFE:FFFF) is not.

For example, if 512 bytes of flash are protected, the protected address region is from 0xFE00 through 0xFFFF. The interrupt vectors (0xFFC0–0xFFFD) are redirected to the locations 0xFDC0–0xFDFD. For instance, if a TPM overflow interrupt is taken, the values in the locations 0xFDF0:FDF1 are used for the vector instead of the values in the locations 0xFFFF0:FFF1. This allows the user to reprogram the unprotected portion of the flash with new program code including new interrupt vector values while leaving the protected area, which includes the default vector locations, unchanged.

4.6 Security

The MC9S08QL8 series include circuitry to prevent unauthorized access to the contents of flash and RAM memory. When security is engaged, flash and RAM are considered secure resources. Direct-page registers, high-page registers, and the background debug controller are considered unsecured resources. Programs executing within secure memory have normal access to any MCU memory locations and resources. Attempts to access a secure memory location with a program executing from an unsecured memory space or through the background debug interface are blocked (writes are ignored and reads return all 0s).

Security is engaged or disengaged based on the state of two nonvolatile register bits (SEC01:SEC00) in the FOPT register. During reset, the contents of the nonvolatile location NVOPT are copied from flash into the working FOPT register in high-page register space. A user engages security by programming the NVOPT location which can be done at the same time the flash memory is programmed. The 1:0 state disengages security and the other three combinations engage security. Notice the erased state (1:1) makes

the MCU secure. During development, whenever the flash is erased, user code must immediately program the SEC00 bit to 0 in NVOPT so SEC01:SEC00 = 1:0. This would allow the MCU to remain unsecured after a subsequent reset.

The background debug controller can still be used for background memory access commands of unsecured resources.

A user can choose to allow or disallow a security unlocking mechanism through an 8-byte backdoor security key. If the nonvolatile KEYEN bit in NVOPT/FOPT is 0, the backdoor key is disabled and there is no way to disengage security without completely erasing all flash locations. If KEYEN is 1, a secure user program can temporarily disengage security by:

1. Writing 1 to KEYACC in the FCNFG register. This makes the flash module interpret writes to the backdoor comparison key locations (NVBACKKEY through NVBACKKEY+7) as values to be compared against the key rather than as the first step in a flash program or erase command.
2. Writing the user-entered key values to the NVBACKKEY through NVBACKKEY+7 locations. These writes must be done in order starting with the value for NVBACKKEY and ending with NVBACKKEY+7. STHX must not be used for these writes because these writes cannot be done on adjacent bus cycles. User software normally would get the key codes from outside the MCU system through a communication interface such as a serial I/O.
3. Writing 0 to KEYACC in the FCNFG register. If the 8-byte key that was just written matches the key stored in the flash locations, SEC01:SEC00 are automatically changed to 1:0 and security will be disengaged until the next reset.

The security key can be written only from secure memory (either RAM or flash), so it cannot be entered through background commands without the cooperation of a secure user program.

The backdoor comparison key (NVBACKKEY through NVBACKKEY+7) is located in flash memory locations in the nonvolatile register space so users can program these locations exactly as they would program any other flash memory location. The nonvolatile registers are in the same 512-byte block of flash as the reset and interrupt vectors, so block protecting that space also block protects the backdoor comparison key. Block protects cannot be changed from user application programs, so if the vector space is block protected, the backdoor security key mechanism cannot permanently change the block protect, security settings, or the backdoor key.

Security can always be disengaged through the background debug interface by taking these steps:

1. Disable any block protections by writing FPROT. FPROT can be written only with background debug commands, not from application software.
2. Mass erase flash if necessary.
3. Blank check flash. Provided flash is completely erased, security is disengaged until the next reset. To avoid returning to secure mode after the next reset, program NVOPT so SEC01:SEC00 = 1:0.

4.7 Flash Registers and Control Bits

The flash module has six 8-bit registers in the high-page register space. Two locations (NVOPT, NVPROT) in the nonvolatile register space in flash memory are copied into corresponding high-page control registers (FOPT, FPROT) at reset. There is also an 8-byte comparison key in flash memory. Refer

to [Table 4-3](#) and [Table 4-4](#) for the absolute address assignments for all flash registers. This section refers to registers and control bits only by their names. An NXP Semiconductors-provided equate or header file is normally used to translate these names into the appropriate absolute addresses.

4.7.1 Flash Clock Divider Register (FCDIV)

Bit 7 of this register is a read-only flag. Bits 6:0 may be read at any time but can be written only one time. Before any erase or programming operations are possible, write to this register to set the frequency of the clock for the nonvolatile memory system within acceptable limits.

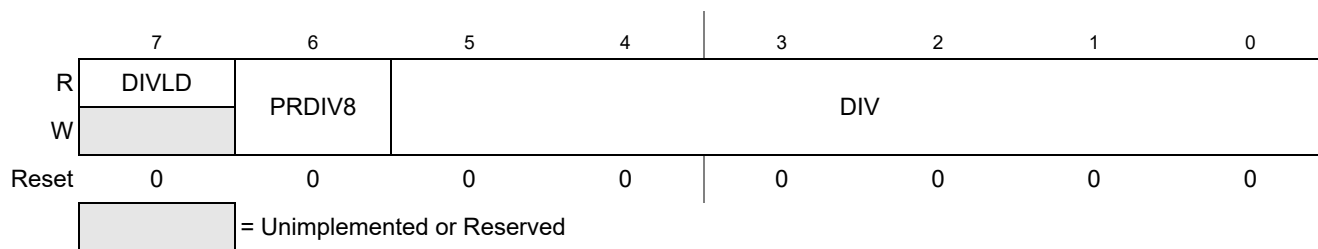


Figure 4-5. Flash Clock Divider Register (FCDIV)

Table 4-6. FCDIV Register Field Descriptions

Field	Description
7 DIVLD	Divisor Loaded Status Flag — When set, this read-only status flag indicates that the FCDIV register has been written since reset. Reset clears this bit and the first write to this register causes this bit to become set regardless of the data written. 0 FCDIV has not been written since reset; erase and program operations disabled for flash. 1 FCDIV has been written since reset; erase and program operations enabled for flash.
6 PRDIV8	Prescale (Divide) Flash Clock by 8 0 Clock input to the flash clock divider is the bus rate clock. 1 Clock input to the flash clock divider is the bus rate clock divided by 8.
5:0 DIV	Divisor for Flash Clock Divider — The flash clock divider divides the bus rate clock (or the bus rate clock divided by 8 if PRDIV8 = 1) by the value in the 6-bit DIV field plus one. The resulting frequency of the internal flash clock must fall within the range of 200 kHz to 150 kHz for proper flash operations. Program/Erase timing pulses are one cycle of this internal flash clock which corresponds to a range of 5 μs to 6.7 μs. The automated programming logic uses an integer number of these pulses to complete an erase or program operation. See Equation 4-1 and Equation 4-2 .

$$\text{if PRDIV8} = 0 \text{ — } f_{\text{CLK}} = f_{\text{Bus}} \div (\text{DIV} + 1) \quad \text{Eqn. 4-1}$$

$$\text{if PRDIV8} = 1 \text{ — } f_{\text{CLK}} = f_{\text{Bus}} \div (8 \times (\text{DIV} + 1)) \quad \text{Eqn. 4-2}$$

[Table 4-7](#) shows the appropriate values for PRDIV8 and DIV for selected bus frequencies.

Table 4-7. Flash Clock Divider Settings

f_{Bus}	PRDIV8 (Binary)	DIV (Decimal)	f_{CLK}	Program/Erase Timing Pulse (5 μs Min, 6.7 μs Max)
20 MHz	1	12	192.3 kHz	5.2 μs
10 MHz	0	49	200 kHz	5 μs
8 MHz	0	39	200 kHz	5 μs
4 MHz	0	19	200 kHz	5 μs
2 MHz	0	9	200 kHz	5 μs
1 MHz	0	4	200 kHz	5 μs
200 kHz	0	0	200 kHz	5 μs
150 kHz	0	0	150 kHz	6.7 μs

4.7.2 Flash Options Register (FOPT and NVOPT)

During reset, the contents of the nonvolatile location NVOPT are copied from flash into FOPT. To change the value in this register, erase and reprogram the NVOPT location in flash memory as usual and then issue a new MCU reset.

	7	6	5	4	3	2	1	0
R	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
W								

Reset This register is loaded from nonvolatile location NVOPT during reset.

 = Unimplemented or Reserved

Figure 4-6. Flash Options Register (FOPT)

Table 4-8. FOPT Register Field Descriptions

Field	Description
7 KEYEN	Backdoor Key Mechanism Enable — When this bit is 0, the backdoor key mechanism cannot be used to disengage security. The backdoor key mechanism is accessible only from user (secured) firmware. BDM commands cannot be used to write key comparison values that would unlock the backdoor key. For more detailed information about the backdoor key mechanism, refer to Section 4.6, Security . 0 No backdoor key access allowed. 1 If user firmware writes an 8-byte value that matches the nonvolatile backdoor key (NVBACKKEY through NVBACKKEY+7 in that order), security is temporarily disengaged until the next MCU reset.
6 FNORED	Vector Redirection Disable — When this bit is 1, then vector redirection is disabled. 0 Vector redirection enabled. 1 Vector redirection disabled.
1:0 SEC0[1:0]	Security State Code — This 2-bit field determines the security state of the MCU as shown in Table 4-9 . When the MCU is secure, the contents of RAM and flash memory cannot be accessed by instructions from any unsecured source including the background debug interface. SEC01:SEC00 changes to 1:0 after successful backdoor key entry or a successful blank check of flash. For more detailed information about security, refer to Section 4.6, Security .

Table 4-9. Security States¹

SEC01:SEC00	Description
0:0	secure
0:1	secure
1:0	unsecured
1:1	secure

¹ SEC01:SEC00 changes to 1:0 after successful backdoor key entry or a successful blank check of flash.

4.7.3 Flash Configuration Register (FCNFG)

	7	6	5	4	3	2	1	0
R	0	0	KEYACC	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

Figure 4-7. Flash Configuration Register (FCNFG)

Table 4-10. FCNFG Register Field Descriptions

Field	Description
5 KEYACC	Enable Writing of Access Key — This bit enables writing of the backdoor comparison key. For more detailed information about the backdoor key mechanism, refer to Section 4.6, Security . 0 Writes to 0xFFB0–0xFFB7 are interpreted as the start of a flash programming or erase command. 1 Writes to NVBACKKEY (0xFFB0–0xFFB7) are interpreted as comparison key writes.

4.7.4 Flash Protection Register (FPROT and NVPROT)

During reset, the contents of the nonvolatile location NVPROT is copied from flash into FPROT. FPROT can be read at any time. With FPDIS set, all bits are writable, but with FPDIS clear the FPS bits are writable as long as the size of the protected region is being increased. Any FPROT write that attempts to decrease the size of the protected region will be ignored.

	7	6	5	4	3	2	1	0
R	FPS ⁽¹⁾							FPDIS ⁽¹⁾
W								
Reset	This register is loaded from nonvolatile location NVPROT during reset.							

¹ Background commands can be used to change the contents of these bits in FPROT.

Figure 4-8. Flash Protection Register (FPROT)

Table 4-11. FPROT Register Field Descriptions

Field	Description
7:1 FPS	Flash Protect Select Bits — When FPDIS = 0, this 7-bit field determines the ending address of unprotected flash locations at the high address end of the flash. Protected flash locations cannot be erased or programmed.
0 FPDIS	Flash Protection Disable 0 Flash block specified by FPS7:FPS1 is block protected (program and erase not allowed). 1 No flash block is protected.

4.7.5 Flash Status Register (FSTAT)

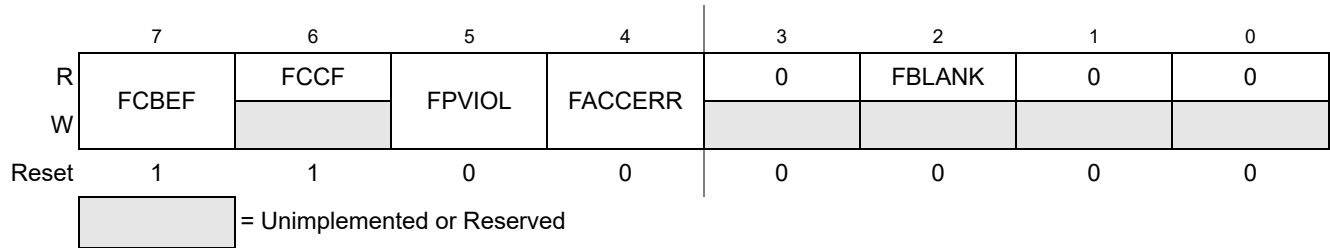


Figure 4-9. Flash Status Register (FSTAT)

Table 4-12. FSTAT Register Field Descriptions

Field	Description
7 FCBEF	Flash Command Buffer Empty Flag — The FCBEF bit is used to launch commands. It also indicates that the command buffer is empty so that a new command sequence can be executed when performing burst programming. The FCBEF bit is cleared by writing a 1 to it or when a burst program command is transferred to the array for programming. Only burst program commands can be buffered. 0 Command buffer is full (not ready for additional commands). 1 A new burst program command can be written to the command buffer.
6 FCCF	Flash Command Complete Flag — FCCF is set automatically when the command buffer is empty and no command is being processed. FCCF is cleared automatically when a new command is started (by writing 1 to FCBEF to register a command). Writing to FCCF has no meaning or effect. 0 Command in progress 1 All commands complete
5 FPVIOL	Protection Violation Flag — FPVIOL is set automatically when a command is written that attempts to erase or program a location in a protected block (the erroneous command is ignored). FPVIOL is cleared by writing a 1 to FPVIOL. 0 No protection violation. 1 An attempt was made to erase or program a protected location.

Table 4-12. FSTAT Register Field Descriptions (continued)

Field	Description
4 FACCERR	Access Error Flag — FACCERR is set automatically when the proper command sequence is not obeyed exactly (the erroneous command is ignored), if a program or erase operation is attempted before the FCDIV register has been initialized, or if the MCU enters stop while a command was in progress. For a more detailed discussion of the exact actions that are considered access errors, see Section 4.5.5, Access Errors . FACCERR is cleared by writing a 1 to FACCERR. Writing a 0 to FACCERR has no meaning or effect. 0 No access error. 1 An access error has occurred.
2 FBLANK	Flash Verified as All Blank (erased) Flag — FBLANK is set automatically at the conclusion of a blank check command if the entire flash array was verified to be erased. FBLANK is cleared by clearing FCBEF to write a new valid command. Writing to FBLANK has no meaning or effect. 0 After a blank check command is completed and FCCF = 1, FBLANK = 0 indicates the flash array is not completely erased. 1 After a blank check command is completed and FCCF = 1, FBLANK = 1 indicates the flash array is completely erased (all 0xFF).

4.7.6 Flash Command Register (FCMD)

Only five command codes are recognized in normal user modes as shown in [Table 4-13](#). Refer to [Section 4.5.3, Program and Erase Command Execution](#) for a detailed discussion of flash programming and erase operations.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	FCMD							
Reset	0	0	0	0	0	0	0	0

Figure 4-10. Flash Command Register (FCMD)

Table 4-13. Flash Commands

Command	FCMD	Equate File Label
Blank check	0x05	mBlank
Byte program	0x20	mByteProg
Byte program — burst mode	0x25	mBurstProg
Page erase (512 bytes/page)	0x40	mPageErase
Mass erase (all flash)	0x41	mMassErase

All other command codes are illegal and generate an access error.

It is not necessary to perform a blank check command after a mass erase operation. Only blank check is required as part of the security unlocking mechanism.

Chapter 5

Resets, Interrupts, and General System Control

5.1 Introduction

This section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupt in the MC9S08QL8 series. Some interrupt sources from peripheral modules are discussed in greater detail in other sections of this document. This section gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and interrupt sources, including the computer operating properly (COP) watchdog are not part of on-chip peripheral systems and have their own chapters.

5.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation
- Reset status register (SRS) to indicate source of most recent reset
- Separate interrupt vector for most modules (reduces polling overhead) (see [Table 5-2](#))

5.3 MCU Reset

Resetting the MCU provides a way to start processing from a known set of initial conditions. During reset, most control and status registers are forced to initial values and the program counter is loaded from the reset vector (0xFFFF:0xFFFF). On-chip peripheral modules are disabled and I/O pins are initially configured as general-purpose high-impedance inputs with pullup devices disabled. The I bit in the condition code register (CCR) is set to block maskable interrupts so the user program has a chance to initialize the stack pointer (SP) and system control settings. SP is forced to 0x00FF at reset.

The MC9S08QL8 series have the following sources for reset:

- Power-on reset (POR)
- External pin reset (PIN)
- Computer operating properly (COP) timer
- Illegal opcode detect (IOP)
- Illegal address detect (ILAD)
- Low-voltage detect (LVD)
- Background debug forced reset

Each of these sources, with the exception of the background debug forced reset, has an associated bit in the system reset status register (SRS).

5.4 Computer Operating Properly (COP) Watchdog

The COP watchdog is intended to force a system reset when the application software fails to execute as expected. To prevent a system reset from the COP timer (when it is enabled), application software must reset the COP counter periodically. If the application program gets lost and fails to reset the COP counter before it times out, a system reset is generated to force the system back to a known starting point.

After any reset, the COPE becomes set in SOPT1 enabling the COP watchdog (see [Section 5.8.4, System Options Register 1 \(SOPT1\)](#) for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing COPE. The COP counter is reset by writing any value to the address of SRS. This write does not affect the data in the read-only SRS. Instead, the act of writing to this address is decoded and sends a reset signal to the COP counter.

The COPCLKS bit in SOPT2 (see [Section 5.8.5, System Options Register 2 \(SOPT2\)](#) for additional information) selects the clock source used for the COP timer. The clock source options are either the bus clock or an internal 1 kHz clock source. With each clock source, there is an associated short and long time-out controlled by COPT in SOPT1. [Table 5-1](#) summarizes the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the 1 kHz clock source and the associated long time-out (2^8 cycles).

Table 5-1. COP Configuration Options

Control Bits		Clock Source	COP Overflow Count
COPCLKS	COPT		
0	0	~1 kHz	2^5 cycles (32 ms) ¹
0	1	~1 kHz	2^8 cycles (256 ms) ¹
1	0	Bus	2^{13} cycles
1	1	Bus	2^{18} cycles

¹ Values are shown in this column based on $t_{LPO} = 1$ ms. See t_{LPO} in the data sheet for the tolerance of this value.

Even if the application will use the reset default settings of COPE, COPCLKS, and COPT, the user must write to the write-once SOPT1 and SOPT2 registers during reset initialization to lock in the settings. That way, they cannot be changed accidentally if the application program gets lost. The initial writes to SOPT1 and SOPT2 will reset the COP counter.

The write to SRS that services (clears) the COP counter must not be placed in an interrupt service routine (ISR) because the ISR could continue to be executed periodically even if the main application program fails.

In background debug mode, the COP counter will not increment.

When the bus clock source is selected, the COP counter does not increment while the system is in stop mode. The COP counter resumes as soon as the MCU exits stop mode.

When the 1 kHz clock source is selected, the COP counter is re-initialized to zero upon entry to stop mode. The COP counter begins from zero after the MCU exits stop mode.

5.5 Interrupts

Interrupts provide a way to save the current CPU status and registers, execute an interrupt service routine (ISR), and then restore the CPU status so processing can resume where it left off before the interrupt. Other than the software interrupt (SWI), which is a program instruction, interrupts are caused by hardware events such as an edge on the IRQ pin or a timer-overflow event.

If an event occurs in an enabled interrupt source, an associated read-only status flag will become set. The CPU will not respond unless the local interrupt enable is a 1 (enabled) and the I bit in the CCR is 0 to allow interrupts. The global interrupt mask (I bit) in the CCR is initially set after reset which prevents all maskable interrupt sources. The user program initializes the stack pointer and performs other system setup before clearing the I bit to allow the CPU to respond to interrupts.

When the CPU receives a qualified interrupt request, it completes the current instruction before responding to the interrupt. The interrupt sequence obeys the same cycle-by-cycle sequence as the SWI instruction and consists of:

- Saving the CPU registers on the stack
- Setting the I bit in the CCR to mask further interrupts
- Fetching the interrupt vector for the highest-priority interrupt that is currently pending
- Filling the instruction queue with the first three bytes of program information starting from the address fetched from the interrupt vector locations

While the CPU is responding to the interrupt, the I bit is automatically set to avoid the possibility of another interrupt interrupting the ISR itself (this is called nesting of interrupts). Normally, the I bit is restored to 0 when the CCR is restored from the value stacked on entry to the ISR. In rare cases, the I bit can be cleared inside an ISR (after clearing the status flag that generated the interrupt) so that other interrupts can be serviced without waiting for the first service routine to finish. This practice is not recommended for anyone other than the most experienced programmers because it can lead to subtle program errors that are difficult to debug.

The interrupt service routine ends with a return-from-interrupt (RTI) instruction which restores the CCR, A, X, and PC registers to their pre-interrupt values by reading the previously saved information from the stack.

NOTE

For compatibility with M68HC08 devices, the H register is not automatically saved and restored. It is good programming practice to push H onto the stack at the start of the interrupt service routine (ISR) and restore it immediately before the RTI that is used to return from the ISR.

If more than one interrupt is pending when the I bit is cleared, the highest priority source is serviced first (see [Table 5-2](#)).

5.5.1 Interrupt Stack Frame

Figure 5-1 shows the contents and organization of a stack frame. Before the interrupt, the stack pointer (SP) points at the next available byte location on the stack. The current values of CPU registers are stored on the stack starting with the low-order byte of the program counter (PCL) and ending with the CCR. After stacking, the SP points at the next available location on the stack. This address is one less than the address where the CCR was saved. The PC value that is stacked is the address of the instruction in the main program that would have executed next if the interrupt had not occurred.

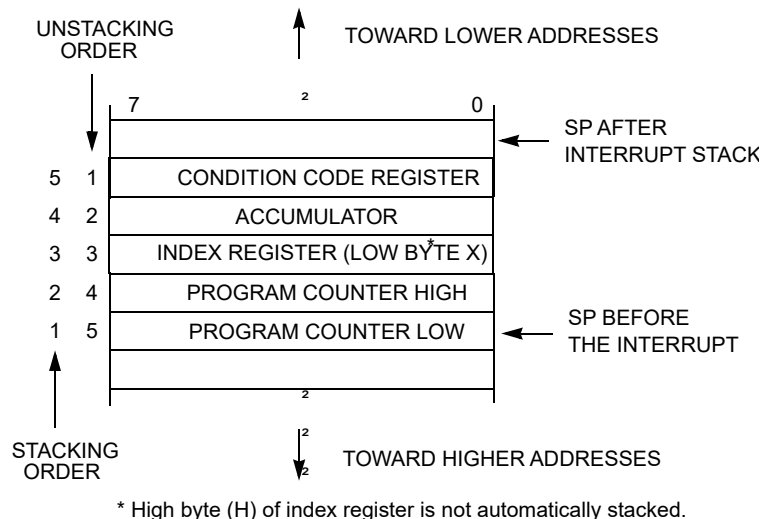


Figure 5-1. Interrupt Stack Frame

When an RTI instruction is executed, these values are recovered from the stack in reverse order. As part of the RTI sequence, the CPU fills the instruction pipeline by reading three bytes of program information, starting from the PC address recovered from the stack.

The status flag corresponding to the interrupt source must be acknowledged (cleared) before returning from the ISR. Typically, the flag is cleared at the beginning of the ISR so that if another interrupt is generated by this same source it will be registered so it can be serviced after completion of the current ISR.

5.5.2 External Interrupt Request (IRQ) Pin

External interrupts are managed by the IRQ status and control register (IRQSC). When the IRQ function is enabled, synchronous logic monitors the pin for edge-only or edge-and-level events. When the MCU is in stop3 mode and system clocks are shut down, a separate asynchronous path is used so the IRQ pin (if enabled) can wake the MCU.

5.5.2.1 Pin Configuration Options

The IRQ pin enable (IRQPE) control bit in IRQSC must be 1 in order for the IRQ pin to act as the interrupt request (IRQ) input. As an IRQ input, the user can choose the polarity of edges or levels detected (IRQEDG), whether the pin detects edges-only or edges and levels (IRQMOD) and whether an event causes an interrupt or only sets the IRQF flag which can be polled by software (IRQIE).

The IRQ pin, when enabled, defaults to use an internal pull device ($IRQPDD = 0$), configured as a pullup or pulldown depending on the polarity chosen. If the user desires to use an external pullup or pulldown, the $IRQPDD$ can be written to a 1 to turn off the internal device.

BIH and BIL instructions may be used to detect the level on the IRQ pin when the pin is configured to act as the IRQ input.

NOTE

This pin does not contain a clamp diode to V_{DD} and must not be driven above V_{DD} .

NOTE

The voltage measured on the internally pulled up IRQ pin will not be pulled to V_{DD} . The internal gates connected to this pin are pulled to V_{DD} . The IRQ pullup must not be used to pullup components external to the MCU.

5.5.2.2 Edge and Level Sensitivity

The $IRQMOD$ control bit reconfigures the detection logic so it detects edge events and pin levels. In the edge and level detection mode, the $IRQF$ status flag becomes set when an edge is detected (when the IRQ pin changes from the deasserted to the asserted level), but the flag is continuously set (and cannot be cleared) as long as the IRQ pin remains at the asserted level.

5.5.3 Interrupt Vectors, Sources, and Local Masks

Table 5-2 provides a summary of all interrupt sources. Higher-priority sources are located toward the bottom of the table. The high-order byte of the address for the interrupt service routine is located at the first address in the vector address column, and the low-order byte of the address for the interrupt service routine is located at the next higher address.

When an interrupt condition occurs, an associated flag bit becomes set. If the associated local interrupt enable is 1, an interrupt request is sent to the CPU. Within the CPU, if the global interrupt mask (I bit in the CCR) is 0, the CPU will finish the current instruction; stack the PCL, PCH, X, A, and CCR CPU registers; set the I bit; and then fetch the interrupt vector for the highest priority pending interrupt. Processing then continues in the interrupt service routine.

Table 5-2. Vector Summary

Vector Priority	Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
Lowest ↑ ↓ Highest	31 through 25	0xFFC0:FFC1 through 0xFFCC:FFCD	Unused Vector Space (available for user program)				
	24	0xFFCE/0xFFCF	Vrtc	RTC	RTIF	RTIE	Real-time Counter
	23	0xFFD0/0xFFD1	—	—	—	—	—
	22	0xFFD2/0xFFD3	—	—	—	—	—
	21	0xFFD4/0xFFD5	—	—	—	—	—
	20	0xFFD6/0xFFD7	Vacmp	ACMP	ACF	ACIE	Analog comparator
	19	0xFFD8/0xFFD9	Vadc	ADC	COCO	AIEN	ADC
	18	0xFFDA/0xFFDB	Vkeyboard	KBI	KBF	KBIE	Keyboard pins
	17	0xFFDC/0xFFDD	—	—	—	—	—
	16	0xFFDE/0xFFDF	Vscitx	SCI	TDRE TC	TIE TCIE	SCI transmit
	15	0xFFE0/0xFFE1	Vscirx	SCI	IDLE RDRF LBKDIF RXEDGIF	ILIE RIE LBKDIE RXEDGIE	SCI receive
	14	0xFFE2/0xFFE3	Vscierr	SCI	OR NF FE PF	ORIE NFIE FEIE PFIE	SCI error
	13	0xFFE4/0xFFE5	—	—	—	—	—
	12	0xFFE6/0xFFE7	Vmtim	MTIM	TOF	TOIE	MTIM
	11	0xFFE8/0xFFE9	—	—	—	—	—
	10	0xFFEA/0xFFEB	—	—	—	—	—
	9	0xFFEC/0xFFED	—	—	—	—	—
	8	0xFFEE/0xFFEF	—	—	—	—	—
	7	0xFFFF0/0xFFFF1	Vtpmovf	TPM	TOF	TOIE	TPM overflow
	6	0xFFFF2/0xFFFF3	—	—	—	—	—
	5	0xFFFF4/0xFFFF5	—	—	—	—	—
	4	0xFFFF6/0xFFFF7	Vtpmch0	TPM	CH0F	CH0IE	TPM channel 0
	3	0xFFFF8/0xFFFF9	Vlvd	System control	LVDF, LVWF	LVDIE, LVWIE	Low-voltage detect, Low-voltage warning
	2	0xFFFFA/0xFFFFB	Virq	IRQ	IRQF	IRQIE	IRQ pin
	1	0xFFFFC/0xFFFFD	Vswi	Core	SWI Instruction	—	Software interrupt
	0	0xFFFFE/0xFFFFF	Vreset	System control	COP LVD RESET pin Illegal opcode Illegal address POR	COPE LVDRE RSTPE — —	Watchdog timer Low-voltage detect External pin Illegal opcode Illegal address

5.6 Low-Voltage Detect (LVD) System

The MC9S08QL8 series include a system to protect against low voltage conditions to protect memory contents and control MCU system states during supply voltage variations. The system is comprised of a power-on reset (POR) circuit and a LVD circuit. The LVD circuit is enabled when LVDE in SPMSC1 is set. The LVD is disabled upon entering either of the stop modes unless LVDSE is set in SPMSC1. If LVDSE and LVDE are both set, then the MCU will enter stop3 instead of stop2, and the current consumption in stop3 with the LVD enabled will be greater.

5.6.1 Power-On Reset Operation

When power is initially applied to the MCU, or when the supply voltage drops below the power-on reset rearm voltage level, V_{POR} , the POR circuit will cause a reset condition. As the supply voltage rises, the LVD circuit will hold the MCU in reset until the supply has risen above the low voltage detection low threshold, V_{LVDL} . Both the POR bit and the LVD bit in SRS are set following a POR.

5.6.2 Low-Voltage Detection (LVD) Reset Operation

The LVD can be configured to generate a reset upon detection of a low voltage condition by setting LVDRE to 1. After an LVD reset has occurred, the LVD system will hold the MCU in reset until the supply voltage has risen above the low voltage detection threshold. The LVD bit in the SRS register is set following either an LVD reset or POR.

5.6.3 Low-Voltage Detection (LVD) Interrupt Operation

When a low voltage condition is detected and the LVD circuit is configured using SPMSC1 for interrupt operation (LVDE set, LVDIE set, and LVDRE clear), then LVDF in SPMSC1 will be set and an LVD interrupt request will occur. The LVDF bit is cleared by writing a 1 to the LVDACK bit in SPMSC1.

5.6.4 Low-Voltage Warning (LVW) Interrupt Operation

The LVD system has a low voltage warning flag (LVWF) to indicate to the user that the supply voltage is approaching, but remains above, the LVD voltage. The LVW also has an interrupt associated with it, enabled by setting the LVWIE bit in the SPMSC3 register. If enabled, an LVW interrupt request will occur when the LVWF is set. LVWF is cleared by writing a 1 to the LVWACK bit in SPMSC3.

5.7 Peripheral Clock Gating

The MC9S08QL8 series include a clock gating system to manage the bus clock sources to the individual peripherals. Using this system, the user can enable or disable the bus clock to each of the peripherals at the clock source, eliminating unnecessary clocks to peripherals which are not in use and thereby reducing the overall run and wait mode currents.

Out of reset, all peripheral clocks will be enabled. For lowest possible run or wait currents, user software must disable the clock source to any peripheral not in use. The actual clock will be enabled or disabled immediately following the write to the clock gating control registers (SCGC1 and SCGC2). Any

peripheral with a gated clock cannot be used unless its clock is enabled. Writing to the registers of a peripheral with a disabled clock has no effect.

NOTE

User software must disable the peripheral before disabling the clocks to the peripheral. When clocks are re-enabled to a peripheral, the peripheral registers need to be re-initialized by user software.

In stop modes, the bus clock is disabled for all gated peripherals, regardless of the settings in SCGC1 and SCGC2.

5.8 Reset, Interrupt, and System Control Registers and Control Bits

One 8-bit register in the direct page register space and eight 8-bit registers in the high-page register space are related to reset and interrupt systems.

Refer to Table 4-2 and Table 4-3 in Chapter 4, Memory for the absolute address assignments for all registers. This section refers to registers and control bits only by their names. An NXP-provided equate or header file is used to translate these names into the appropriate absolute addresses.

Some control bits in the SOPT1 and SPMSC2 registers are related to modes of operation. Although brief descriptions of these bits are provided here, the related functions are discussed in greater detail in Chapter 3, Modes of Operation.

5.8.1 Interrupt Pin Request Status and Control Register (IRQSC)

This direct-page register includes status and control bits which are used to configure the IRQ function, report status, and acknowledge IRQ events.

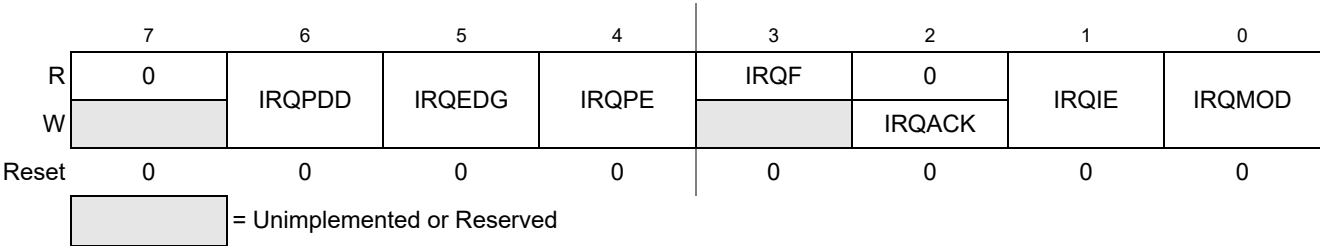


Figure 5-2. Interrupt Request Status and Control Register (IRQSC)

Table 5-3. IRQSC Register Field Descriptions

Field	Description
6 IRQPDD	Interrupt Request (IRQ) Pull Device Disable — This read/write control bit is used to disable the internal pullup/pulldown device when the IRQ pin is enabled (IRQPE = 1) allowing for an external device to be used. 0 IRQ pull device enabled if IRQPE = 1. 1 IRQ pull device disabled if IRQPE = 1.
5 IRQEDG	Interrupt Request (IRQ) Edge Select — This read/write control bit is used to select the polarity of edges or levels on the IRQ pin that cause IRQF to be set. The IRQMOD control bit determines whether the IRQ pin is sensitive to both edges and levels or only edges. When IRQEDG = 1 and the internal pull device is enabled, the pullup device is reconfigured as an optional pulldown device. 0 IRQ is falling edge or falling edge/low-level sensitive. 1 IRQ is rising edge or rising edge/high-level sensitive.
4 IRQPE	IRQ Pin Enable — This read/write control bit enables the IRQ pin function. When this bit is set the IRQ pin can be used as an interrupt request. 0 IRQ pin function is disabled. 1 IRQ pin function is enabled.
3 IRQF	IRQ Flag — This read-only status bit indicates when an interrupt request event has occurred. 0 No IRQ request. 1 IRQ event detected.
2 IRQACK	IRQ Acknowledge — This write-only bit is used to acknowledge interrupt request events (write 1 to clear IRQF). Writing 0 has no meaning or effect. Reads always return 0. If edge-and-level detection is selected (IRQMOD = 1), IRQF cannot be cleared while the IRQ pin remains at its asserted level.
1 IRQIE	IRQ Interrupt Enable — This read/write control bit determines whether IRQ events generate an interrupt request. 0 Interrupt request when IRQF set is disabled (use polling). 1 Interrupt requested whenever IRQF = 1.
0 IRQMOD	IRQ Detection Mode — This read/write control bit selects either edge-only detection or edge-and-level detection. The IRQEDG control bit determines the polarity of edges and levels that are detected as interrupt request events. See Section 5.5.2.2, Edge and Level Sensitivity for more details. 0 IRQ event on falling edges or rising edges only. 1 IRQ event on falling edges and low levels or on rising edges and high levels.

5.8.2 System Reset Status Register (SRS)

This high-page register includes read-only status flags to indicate the source of the most recent reset. When a debug host forces reset by writing 1 to BDFR in the SBDIFR register, none of the status bits in SRS will be set. Writing any value to this register address clears the COP watchdog timer without affecting the contents of this register. The reset state of these bits depends on what caused the MCU to reset.

	7	6	5	4	3	2	1	0
R	POR	PIN	COP	ILOP	ILAD	0	LVD	0
W	Writing any value to SRS address clears COP watchdog timer.							
POR:	1	0	0	0	0	0	1	0
LVD:	u ¹	0	0	0	0	0	1	0
Any other reset:	0	Note ²	Note ²	Note ²	Note ²	0	0	0

1. u = unaffected

2. Any of these reset sources that are active at the time of reset entry will cause the corresponding bit(s) to be set; bits corresponding to sources that are not active at the time of reset entry will be cleared.

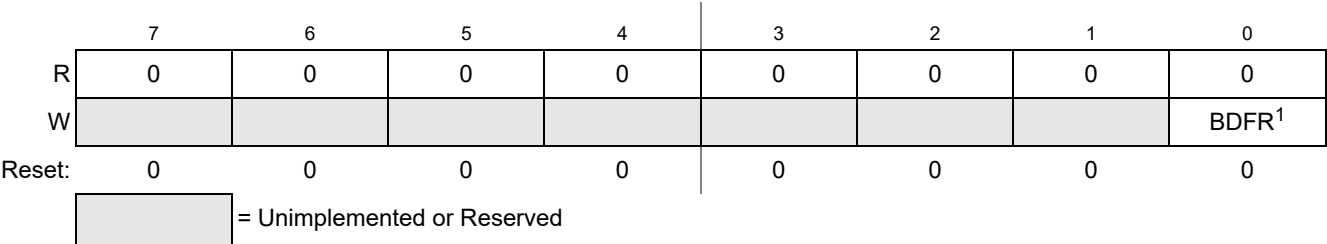
Figure 5-3. System Reset Status (SRS)

Table 5-4. SRS Register Field Descriptions

Field	Description
7 POR	Power-On Reset — Reset was caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD threshold. 0 Reset not caused by POR. 1 POR caused reset.
6 PIN	External Reset Pin — Reset was caused by an active-low level on the external reset pin. 0 Reset not caused by external reset pin. 1 Reset came from external reset pin.
5 COP	Computer Operating Properly (COP) Watchdog — Reset was caused by the COP watchdog timer timing out. This reset source can be blocked by COPE = 0. 0 Reset not caused by COP timeout. 1 Reset caused by COP timeout.
4 ILOP	Illegal Opcode — Reset was caused by an attempt to execute an unimplemented or illegal opcode. The STOP instruction is considered illegal if stop is disabled by STOPE = 0 in the SOPT register. The BGND instruction is considered illegal if active background mode is disabled by ENBDM = 0 in the BDCSC register. 0 Reset not caused by an illegal opcode. 1 Reset caused by an illegal opcode.
3 ILAD	Illegal Address — Reset was caused by an attempt to access either data or an instruction at an unimplemented memory address. 0 Reset not caused by an illegal address 1 Reset caused by an illegal address
1 LVD	Low Voltage Detect — If the LVDRE bit is set and the supply drops below the LVD trip voltage, an LVD reset will occur. This bit is also set by POR. 0 Reset not caused by LVD trip or POR. 1 Reset caused by LVD trip or POR.

5.8.3 System Background Debug Force Reset Register (SBDFR)

This high-page register contains a single write-only control bit. A serial background command such as WRITE_BYTE must be used to write to SBDFR. Attempts to write this register from a user program are ignored. Reads always return 0x00.



1. BDFR is writable only through serial background debug commands, not from user programs.

Figure 5-4. System Background Debug Force Reset Register (SBDFR)

Table 5-5. SBDFR Register Field Descriptions

Field	Description
0 BDFR	Background Debug Force Reset — A serial background command such as WRITE_BYTE can be used to allow an external debug host to force a target system reset. Writing 1 to this bit forces an MCU reset. This bit cannot be written from a user program. To enter user mode, PTA4/ACMPO/BKGD/MS must be high immediately after issuing WRITE_BYTE command. To enter BDM, PTA4/ACMPO/BKGD/MS must be low immediately after issuing WRITE_BYTE command. See the data sheet for more information.

5.8.4 System Options Register 1 (SOPT1)

This high-page register is a write-once register, so only the first write after reset is honored. It can be read at any time. Any subsequent attempt to write to SOPT1 (intentionally or unintentionally) is ignored to avoid accidental changes to these sensitive settings. SOPT1 must be written during the user's reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

	7	6	5	4	3	2	1	0
R	COPE	COPT	STOPE	0	0	0	BKGDPE	RSTPE
W								
Reset:	1	1	0	0	0	0	1	u ¹
POR:	1	1	0	0	0	0	1	0
LVR:	1	1	0	0	0	0	1	0

 = Unimplemented or Reserved

1. u = unaffected

Figure 5-5. System Options Register 1 (SOPT1)

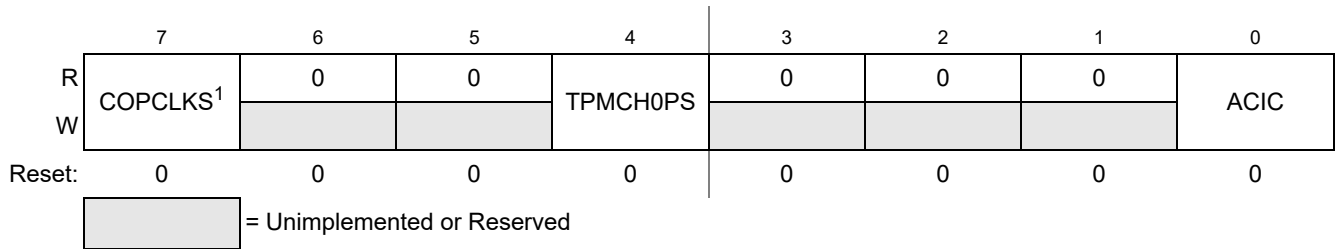
Table 5-6. SOPT1 Register Field Descriptions

Field	Description
7 COPE	COP Watchdog Enable — This write-once bit selects whether the COP watchdog is enabled. 0 COP watchdog timer disabled. 1 COP watchdog timer enabled (force reset on timeout).
6 COPT	COP Watchdog Timeout — This write-once bit selects the timeout period of the COP. COPT along with COPCLKS in SOPT2 defines the COP timeout period. 0 Short timeout period selected. 1 Long timeout period selected.
5 STOPE	Stop Mode Enable — This write-once bit is used to enable stop mode. If stop mode is disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset is forced. 0 Stop mode disabled. 1 Stop mode enabled.
1 BKGDPE	Background Debug Mode Pin Enable — This write-once bit when set enables the PTA4/ACMPO/BKGD/MS pin to function as BKGD/MS. When clear, the pin functions as one of its output only alternative functions. This pin defaults to the BKGD/MS function following any MCU reset. 0 PTA4/ACMPO/BKGD/MS pin functions as PTA4 or ACMPO 1 PTA4/ACMPO/BKGD/MS pin functions as BKGD/MS.
0 RSTPE ¹	RESET Pin Enable — This write-once bit when set enables the PTA5/IRQ/TCLK/RESET pin to function as RESET. When clear, the pin functions as one of its input only alternative functions. This pin defaults to its PTA5 function following an MCU POR. When RSTPE is set, an internal pullup device is enabled on RESET. 0 PTA5/IRQ/TCLK/RESET pin functions as PTA5, IRQ or TCLK. 1 PTA5/IRQ/TCLK/RESET pin functions as RESET.

¹ The RSTPE bit will be cleared by the stop2 recovery and must not be set before writing to the PPDACK bit. Doing so will cause a second reset event and the PPDF bit will be cleared at the end of the second reset. The RESET pin must be monitored by reading PTA5 port as input so that RSTPE is enabled only after the pin is confirmed as "1". By doing so, RESET pin can avoid being low and cause another reset. Mind that COP must be refreshed during pin monitor to prevent unwanted COP reset.

5.8.5 System Options Register 2 (SOPT2)

This high page register contains bits to configure MCU specific features on the MC9S08QL8 series devices.



1. This bit can be written only one time after reset. Additional writes are ignored.

Figure 5-6. System Options Register 2 (SOPT2)

Table 5-7. SOPT2 Register Field Descriptions

Field	Description
7 COPCLKS	COP Watchdog Clock Select — This write-once bit selects the clock source of the COP watchdog. 0 Internal 1-kHz clock is source to COP. 1 Bus clock is source to COP.
4 TPMCH0PS	TPMCH0 Pin Select — This bit selects the location of the TPMCH0 pins of the TPM module. 0 TPMCH0 on PTA0. 1 TPMCH0 on PTB5.
0 ACIC	Analog Comparator to Input Capture Enable — This bit connects the output of ACMP to TPM input channel 0. See Chapter 9, Analog Comparator (S08ACMPVLPV1) and Chapter 15, Timer/Pulse-Width Modulator (S08TPMV3) for more details on this feature. 0 ACMP output not connected to TPM input channel 0. 1 ACMP output connected to TPM input channel 0.

5.8.6 System Device Identification Register (SDIDH, SDIDL)

These high-page read-only registers are included so host development systems can identify the HCS08 derivative. This allows the development software to recognize where specific memory blocks, registers, and control bits are located in a target MCU.

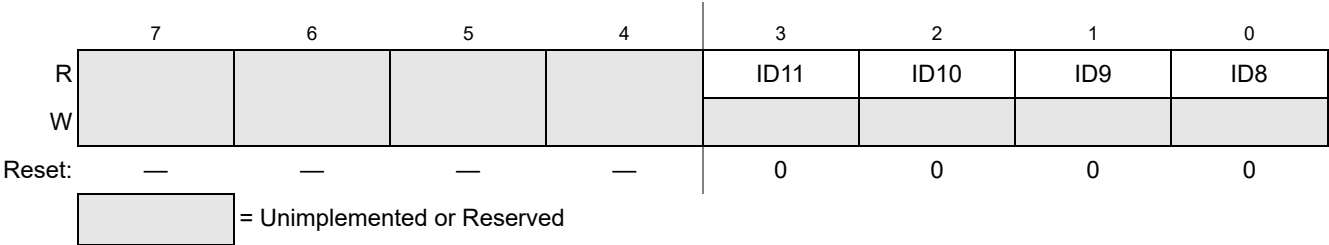


Figure 5-7. System Device Identification Register — High (SDIDH)

Table 5-8. SDIDH Register Field Descriptions

Field	Description
7:4 Reserved	Bits 7:4 are reserved. Reading these bits will result in an indeterminate value; writes have no effect.
3:0 ID[11:8]	Part Identification Number — Each derivative in the HCS08 family has a unique identification number. The MC9S08QL8 is hard coded to the value 0x023. See also ID bits in Table 5-9 .

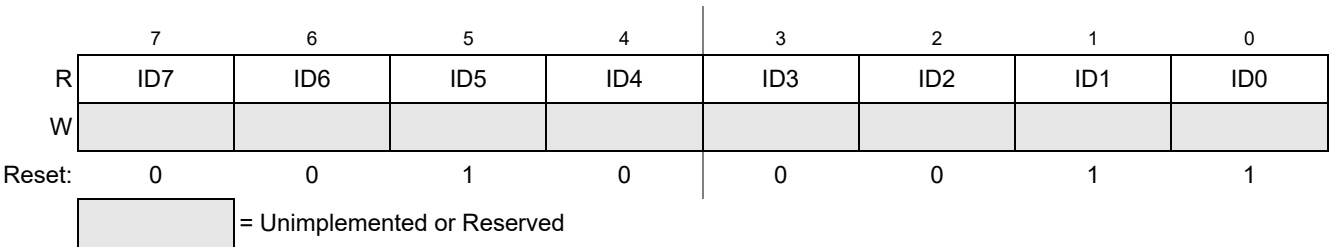


Figure 5-8. System Device Identification Register — Low (SDIDL)

Table 5-9. SDIDL Register Field Descriptions

Field	Description
7:0 ID[7:0]	Part Identification Number — Each derivative in the HCS08 Family has a unique identification number. The MC9S08QL8 is hard coded to the value 0x023. See also ID bits in Table 5-8 .

5.8.7 System Power Management Status and Control 1 Register (SPMSC1)

This high-page register contains status and control bits to support the low voltage detect function, and to enable the bandgap voltage reference for use by the ADC or ACMP modules. To configure the low voltage detect trip voltage, see [Table 5-12](#) for the LVDV bit description in SPMSC3.

	7	6	5	4	3	2	1	0
R	LVDF	0	LVDIE	LVDRE ²	LVDSE	LVDE ²	0	BGBE
W		LVDACK						
Reset:	0	0	0	1	1	1	0	0
Stop2 wakeup:	u	0	u	u	u	u	0	u

= Unimplemented or Reserved u= Unaffected by reset

1. Bit 1 is a reserved bit that must always be written to 0.

2. This bit can be written only one time after reset. Additional writes are ignored.

Figure 5-9. System Power Management Status and Control 1 Register (SPMSC1)

Table 5-10. SPMSC1 Register Field Descriptions

Field	Description
7 LVDF	Low-Voltage Detect Flag — Provided LVDE = 1, this read-only status bit indicates a low-voltage detect event.
6 LVDACK	Low-Voltage Detect Acknowledge — This write-only bit is used to acknowledge low voltage detection errors (write 1 to clear LVDF). Reads always return 0.
5 LVDIE	Low-Voltage Detect Interrupt Enable — This bit enables hardware interrupt requests for LVDF. 0 Hardware interrupt disabled (use polling). 1 Request a hardware interrupt when LVDF = 1.
4 LVDRE	Low-Voltage Detect Reset Enable — This write-once bit enables LVDF events to generate a hardware reset (provided LVDE = 1). 0 LVDF does not generate hardware resets. 1 Force an MCU reset when LVDF = 1.
3 LVDSE	Low-Voltage Detect Stop Enable — Provided LVDE = 1, this read/write bit determines whether the low-voltage detect function operates when the MCU is in stop mode. 0 Low-voltage detect disabled during stop mode. 1 Low-voltage detect enabled during stop mode.
2 LVDE	Low-Voltage Detect Enable — This write-once bit enables low-voltage detect logic and qualifies the operation of other bits in this register. 0 LVD logic disabled. 1 LVD logic enabled.
0 BGBE	Bandgap Buffer Enable — This bit enables an internal buffer for the bandgap voltage reference for use by the ADC module on one of its internal channels or as a voltage reference for ACMP module. 0 Bandgap buffer disabled. 1 Bandgap buffer enabled.

5.8.8 System Power Management Status and Control 2 Register (SPMSC2)

This high-page register contains status and control bits to configure the low power run and wait modes as well as configure the stop mode behavior of the MCU. See [Section 3.3.1, Low Power Run Mode \(LPRun\)](#), [Section 3.5.1, Low Power Wait Mode \(LPWait\)](#), and [Section 3.6, Stop Modes](#) for more information.

	7	6	5	4	3	2	1	0
R	LPR	LPRS	LPWUI	0	PPDF	0	PPDE ¹	PPDC
W						PPDACK		
Reset:	0	0	0	0	0	0	1	0
Stop2 wakeup:	0	0	u	0	1	0	1	1

= Unimplemented or Reserved
 u= Unaffected by reset

1. This bit can be written only one time after reset. Additional writes are ignored.

Figure 5-10. System Power Management Status and Control 2 Register (SPMSC2)

Table 5-11. SPMSC2 Register Field Descriptions

Field	Description
7 LPR	Low Power Regulator Control — The LPR bit controls entry into the low power run and wait modes in which the voltage regulator is put into standby. This bit cannot be set if PPDC=1. If PPDC and LPR are set in a single write instruction, only PPDC will actually be set. Automatically cleared when LPWUI is set and an interrupt occurs. 0 Low power run and wait modes are disabled. 1 Low power run and wait modes are enabled.
6 LPRS	Low Power Regulator Status — This read-only status bit indicates that the voltage regulator has entered into standby for the low power run or wait mode. 0 The voltage regulator is not currently in standby. 1 The voltage regulator is currently in standby.
5 LPWUI	Low Power Wake Up on Interrupt — This bit controls whether or not the voltage regulator exits standby when any active MCU interrupt occurs. 0 The voltage regulator will remain in standby on an interrupt. 1 The voltage regulator will exit standby on an interrupt.
3 PPDF	Partial Power Down Flag — This read-only status bit indicates that the MCU has recovered from stop2 mode. 0 MCU has not recovered from stop2 mode. 1 MCU recovered from stop2 mode.
2 PPDACK	Partial Power Down Acknowledge — Writing a 1 to PPDACK clears the PPDF bit.
1 PPDE	Partial Power Down Enable — The write-once PPDE bit can be used to “lockout” the partial power down mode. 0 Partial power down is not enabled. 1 Partial power down is enabled and controlled via the PPDC bit.
0 PPDC	Partial Power Down Control — The PPDC bit controls which power down mode is selected. This bit cannot be set if LPR=1. If PPDC and LPR are set in a single write instruction, only PPDC will actually be set. 0 Stop3 low power mode enabled. 1 Stop2 partial power down mode enabled.

5.8.9 System Power Management Status and Control 3 Register (SPMSC3)

This high-page register is used to report the status of the low voltage warning function and to select the low voltage detect trip voltage.

	7	6	5	4	3	2	1	0
R	LVWF	0	—	—	LVWIE	0	0	0
W		LVWACK						
Reset:	0 ¹	0	0	0	0	0	0	0
Stop2 wakeUp:	u	0	u	u	u	0	0	0

= Unimplemented or Reserved
 u = Unaffected by reset

1. LVWF will be set if V_{Supply} transitions below the trip point or after reset and V_{Supply} is already below V_{LVW} .

Figure 5-11. System Power Management Status and Control 3 Register (SPMSC3)

Table 5-12. SPMSC3 Register Field Descriptions

Field	Description
7 LVWF	Low-Voltage Warning Flag — The LVWF bit indicates the low voltage warning status. 0 Low voltage warning not present. 1 Low voltage warning is present or was present.
6 LVWACK	Low-Voltage Warning Acknowledge — The LVWF bit indicates the low voltage warning status. Writing a 1 to LVWACK clears LVWF to a 0 if a low voltage warning is not present.
3 LVWIE	Low-Voltage Warning Interrupt Enable — This bit enables hardware interrupt requests for LVWF. 0 Hardware interrupt disabled (use polling). 1 Request a hardware interrupt when LVWF = 1.

Table 5-13. LVD and LVW Trip Point Typical Values¹

LVW Trip Point	LVD Trip Point
$V_{\text{LVW}} = 2.14 \text{ V}$	$V_{\text{LVD}} = 1.84 \text{ V}$

¹ See MC9S08QL8 Series *Data Sheet* for minimum and maximum values.

5.8.10 System Clock Gating Control 1 Register (SCGC1)

This high page register contains control bits to enable or disable the bus clock to the TPM and ADC modules. Gating off the clocks to unused peripherals is used to reduce the MCU's run and wait currents. See [Section 5.7, Peripheral Clock Gating](#) for more information.

NOTE

User software must disable the peripheral before disabling the clocks to the peripheral. When clocks are re-enabled to a peripheral, the peripheral registers need to be re-initialized by user software.

	7	6	5	4	3	2	1	0
R	MTIM	1	TPM	ADC	1	1	1	SCI
W		—			—	—	—	
Reset:	1	1	1	1	1	1	1	1

Figure 5-12. System Clock Gating Control 1 Register (SCGC1)

Table 5-14. SCGC1 Register Field Descriptions

Field	Description
7 MTIM	MTIM Clock Gate Control — This bit controls the clock gate to the MTIM module. 0 Bus clock to the MTIM module is disabled. 1 Bus clock to the MTIM module is enabled.
5 TPM	TPM Clock Gate Control — This bit controls the clock gate to the TPM module. 0 Bus clock to the TPM module is disabled. 1 Bus clock to the TPM module is enabled.
4 ADC	ADC Clock Gate Control — This bit controls the clock gate to the ADC module. 0 Bus clock to the ADC module is disabled. 1 Bus clock to the ADC module is enabled.
0 SCI	SCI Clock Gate Control — This bit controls the clock gate to the SCI module. 0 Bus clock to the SCI module is disabled. 1 Bus clock to the SCI module is enabled.

5.8.11 System Clock Gating Control 2 Register (SCGC2)

This high page register contains control bits to enable or disable the bus clock to the Flash, IRQ, KBI, ACMP and RTC modules. Gating off the clocks to unused peripherals is used to reduce the MCU's run and wait currents. See [Section 5.7, Peripheral Clock Gating](#) for more information.

NOTE

User software must disable the peripheral before disabling the clocks to the peripheral. When clocks are re-enabled to a peripheral, the peripheral registers must be re-initialized by user software.

	7	6	5	4	3	2	1	0
R	1	FLS	IRQ	KBI	ACMP	RTC	1	1
W	—						—	—
Reset:	1	1	1	1	1	1	1	1

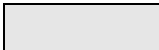
 = Unimplemented or Reserved

Figure 5-13. System Clock Gating Control 2 Register (SCGC2)

Table 5-15. SCGC2 Register Field Descriptions

Field	Description
6 FLS	Flash Register Clock Gate Control — This bit controls the bus clock gate to the Flash module. 0 Bus clock to the Flash module is disabled. 1 Bus clock to the Flash module is enabled.
5 IRQ	IRQ Clock Gate Control — This bit controls the bus clock gate to the IRQ module. 0 Bus clock to the IRQ module is disabled. 1 Bus clock to the IRQ module is enabled.
4 KBI	KBI Clock Gate Control — This bit controls the clock gate to the KBI module. 0 Bus clock to the KBI module is disabled. 1 Bus clock to the KBI module is enabled.
3 ACMP	ACMP Clock Gate Control — This bit controls the clock gate to both of the ACMP modules. 0 Bus clock to the ACMP modules is disabled. 1 Bus clock to the ACMP modules is enabled.
2 RTC	RTC Clock Gate Control — This bit controls the bus clock gate to the RTC module. Only the bus clock is gated, the IC SERCLK and LPOCLK are still available to the RTC. 0 Bus clock to the RTC module is disabled. 1 Bus clock to the RTC module is enabled.

Chapter 6

Parallel Input/Output Control

This chapter explains software controls related to parallel input/output (I/O) and pin control. The MC9S08QL8 has three parallel I/O ports which include a total of 22 I/O pins, one output-only pin and one input-only pin. See [Chapter 2, Pins and Connections](#) for more information about pin assignments and external hardware considerations of these pins.

Many of these pins are shared with on-chip peripherals such as timer systems, analog modules, or keyboard interrupts, as shown in [Table 2-1](#). The peripheral modules have priority over the general-purpose I/O functions so that when a peripheral is enabled, the I/O functions associated with the shared pins may be disabled.

After reset, the shared peripheral functions are disabled and the pins are configured as inputs ($PTxDDn = 0$). The pin control functions for each pin are configured as follows: slew rate control enabled ($PTxSEn = 1$), low drive strength selected ($PTxDSn = 0$), and internal pullups disabled ($PTxPEN = 0$).

NOTE

Not all general-purpose I/O pins are available on all packages. To avoid extra current drain from floating input pins, the user's reset initialization routine in the application program must either enable on-chip pullup devices or change the direction of unconnected pins to outputs so the pins do not float.

6.1 Port Data and Data Direction

Reading and writing of parallel I/Os are performed through the port data registers. The direction, either input or output, is controlled through the port data direction registers. The parallel I/O port function for an individual pin is illustrated in the block diagram shown in [Figure 6-1](#).

The data direction control bit ($PTxDDn$) determines whether the output buffer for the associated pin is enabled, and also controls the source for port data register reads. The input buffer for the associated pin is always enabled unless the pin is enabled as an analog function or is an output-only pin.

When a shared digital function is enabled for a pin, the output buffer is controlled by the shared function. However, the data direction register bit will continue to control the source for reads of the port data register.

When a shared analog function is enabled for a pin, both the input and output buffers are disabled. A value of 0 is read for any port data bit where the bit is an input ($PTxDDn = 0$) and the input buffer is disabled. In general, whenever a pin is shared with both an alternate digital function and an analog function, the analog function has priority such that if both the digital and analog functions are enabled, the analog function controls the pin.

It is good programming practice to write to the port data register before changing the direction of a port pin so it becomes an output. This ensures that the pin will not be driven momentarily with an old data value that happened to be in the port data register.

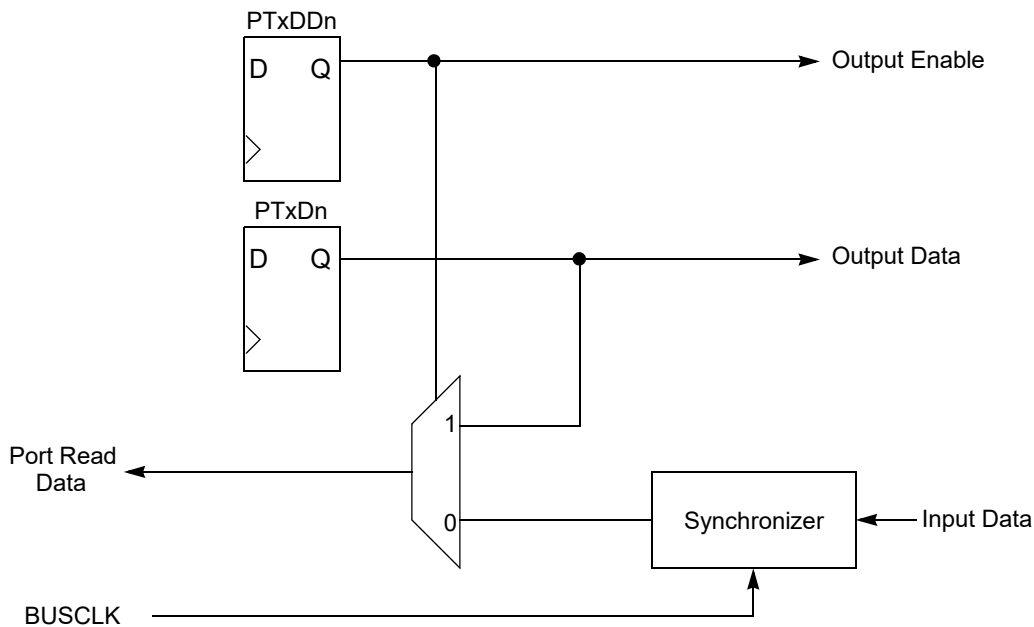


Figure 6-1. Parallel I/O Block Diagram

6.2 Pullup, Slew Rate, and Drive Strength

Associated with the parallel I/O ports is a set of registers located in the high page register space that operate independently of the parallel I/O registers. These registers are used to control pullups, slew rate, and drive strength for the pins and may be used in conjunction with the peripheral functions on these pins.

6.2.1 Port Internal Pullup Enable

An internal pullup device can be enabled for each port pin by setting the corresponding bit in the pullup enable register (PTxPEN). The pullup device is disabled if the pin is configured as an output by the parallel I/O control logic or any shared peripheral function regardless of the state of the corresponding pullup enable register bit. The pullup device is also disabled if the pin is controlled by an analog function.

6.2.2 Port Slew Rate Enable

Slew rate control can be enabled for each port pin by setting the corresponding bit in the slew rate control register (PTxSEN). When enabled, slew control limits the rate at which an output can transition in order to reduce EMC emissions. Slew rate control has no effect on pins that are configured as inputs.

6.2.3 Port Drive Strength Select

An output pin can be configured for high output drive strength by setting the corresponding bit in the drive strength select register (PTxDSn). When high drive is selected, a pin is capable of sourcing and sinking

greater current. Even though every I/O pin can be selected as high drive, the user must ensure that the total current source and sink limits for the MCU are not exceeded. Drive strength selection is intended to affect the DC behavior of I/O pins. However, the AC behavior is also affected. High drive allows a pin to drive a greater load with the same switching speed as a low drive enabled pin into a smaller load. Because of this, the EMC emissions may be affected by enabling pins as high drive.

6.3 Pin Behavior in Stop Modes

Pin behavior following execution of a STOP instruction depends on the stop mode that is entered. An explanation of pin behavior for the various stop modes follows:

- Stop2 mode is a partial power-down mode, whereby I/O latches are maintained in their pre-STOP instruction state. CPU register status and the state of I/O registers must be saved in RAM before the STOP instruction is executed to place the MCU in stop2 mode. Upon recovery from stop2 mode, before accessing any I/O, the user must examine the state of the PPDF bit in the SPMSC2 register. If the PPDF bit is 0, I/O must be initialized as if a power-on reset had occurred. If the PPDF bit is 1, I/O register states must be restored from the values saved in RAM before the STOP instruction was executed. Peripherals may require initialization or restoration to their pre-stop condition. The user must then write a 1 to the PPDACK bit in the SPMSC2 register. Access to I/O is again permitted in the user application program.
- In stop3 mode, all I/O is maintained because internal logic circuitry stays powered up. Upon recovery, normal I/O function is available to the user.

6.4 Parallel I/O and Pin Control Registers

This section provides information about the registers associated with the parallel I/O ports. The data and data direction registers are located in page zero of the memory map. The pull up, slew rate, drive strength, and interrupt control registers are located in the high page section of the memory map.

Refer to tables in [Chapter 4, Memory](#) for the absolute address assignments for all parallel I/O and their pin control registers. This section refers to registers and control bits only by their names. An NXP-provided equate or header file is normally to translate these names into the appropriate absolute addresses.

6.4.1 Port A Registers

Port A is controlled by the registers listed below.

The pins PTA4 and PTA5 are unique. PTA4 is an output only, so the control bits for the input functions will not have any effect on this pin. PTA5 is an input only, so the control bits for the output functions will not have any effect on this pin. BKGDPE bit in SOPT1 register is set following any reset of the MCU and must be cleared to use the PTA4/ACMPO/BKGD/MS pin's alternative pin functions.

6.4.1.1 Port A Data Register (PTAD)

	7	6	5	4	3	2	1	0
R	—	—	PTAD5 ¹	PTAD4 ²	PTAD3	PTAD2	PTAD1	PTAD0
W	—	—						
Reset:	0	0	0	0	0	0	0	0

1. Reads of bit PTAD5 always return the pin value of PTA5, regardless of the value stored in bit PTADD5.
2. Reads of bit PTAD4 always return the contents of PTAD4, regardless of the value stored in bit PTADD4.

Figure 6-2. Port A Data Register (PTAD)

Table 6-1. PTAD Register Field Descriptions

Field	Description
5:0 PTAD[5:0]	Port A Data Register Bits — For port A pins that are inputs, reads return the logic level on the pin. For port A pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port A pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTAD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups/pulldowns disabled.

6.4.1.2 Port A Data Direction Register (PTADD)

	7	6	5	4	3	2	1	0
R	—	—	—	—	PTADD3	PTADD2	PTADD1	PTADD0
W	—	—	—	—				
Reset:	0	0	0	0	0	0	0	0

Figure 6-3. Port A Data Direction Register (PTADD)

Table 6-2. PTADD Register Field Descriptions

Field	Description
3:0 PTADD[3:0]	Data Direction for Port A Bits — These read/write bits control the direction of port A pins and what is read for PTAD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port A bit n and PTAD reads return the contents of PTADn.

6.4.1.3 Port A Pull Enable Register (PTAPE)

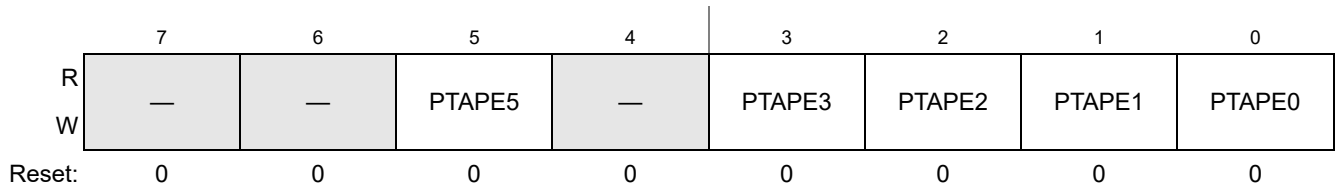


Figure 6-4. Internal Pull Enable for Port A Register (PTAPE)

Table 6-3. PTAPE Register Field Descriptions

Field	Description
5, 3:0 PTAPE[5][3:0]]	Internal Pull Enable for Port A Bits — Each of these control bits determines if the internal pullup or pulldown device is enabled for the associated PTA pin. For port A pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pullup/pulldown device disabled for port A bit n. 1 Internal pullup/pulldown device enabled for port A bit n.

6.4.1.4 Port A Slew Rate Enable Register (PTASE)

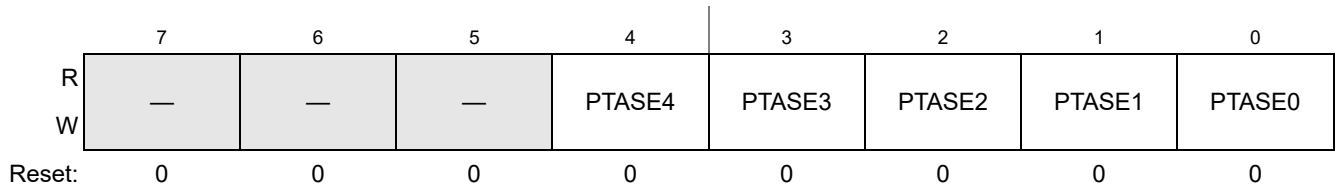


Figure 6-5. Slew Rate Enable for Port A Register (PTASE)

Table 6-4. PTASE Register Field Descriptions

Field	Description
4:0 PTASE[4:0]	Output Slew Rate Enable for Port A Bits — Each of these control bits determines if the output slew rate control is enabled for the associated PTA pin. For port A pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port A bit n. 1 Output slew rate control enabled for port A bit n.

6.4.1.5 Port A Drive Strength Selection Register (PTADS)

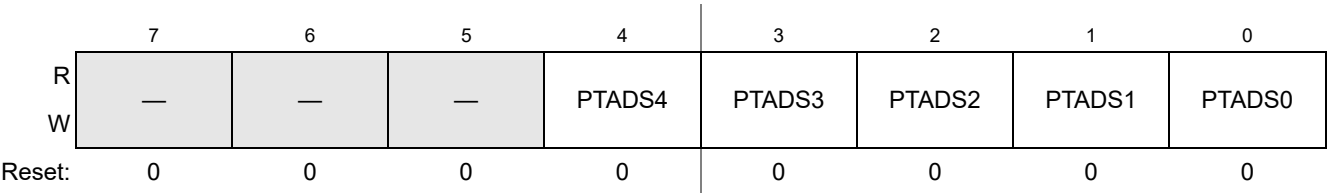


Figure 6-6. Drive Strength Selection for Port A Register (PTADS)

Table 6-5. PTADS Register Field Descriptions

Field	Description
4:0 PTADS[4:0]	Output Drive Strength Selection for Port A Bits — Each of these control bits selects between low and high output drive for the associated PTA pin. For port A pins that are configured as inputs, these bits have no effect. 0 Low output drive strength selected for port A bit n. 1 High output drive strength selected for port A bit n.

6.4.2 Port B Registers

Port B is controlled by the registers listed below.

6.4.2.1 Port B Data Register (PTBD)

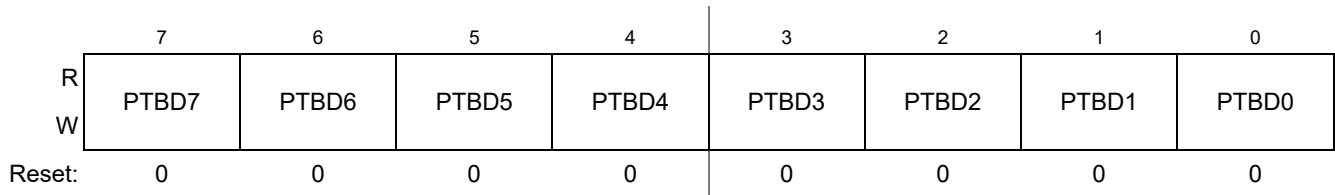


Figure 6-7. Port B Data Register (PTBD)

Table 6-6. PTBD Register Field Descriptions

Field	Description
7:0 PTBD[7:0]	Port B Data Register Bits — For port B pins that are inputs, reads return the logic level on the pin. For port B pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port B pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTBD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups/pulldowns disabled.

6.4.2.2 Port B Data Direction Register (PTBDD)

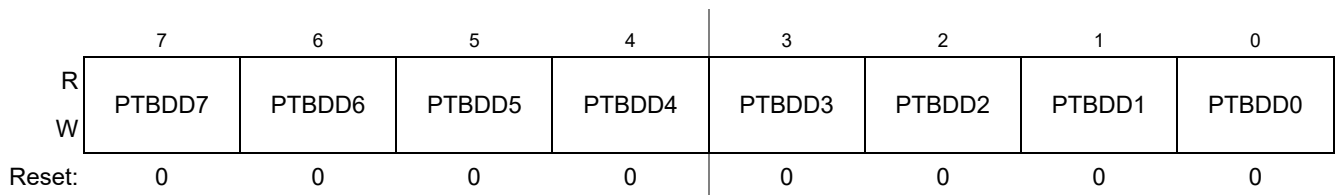


Figure 6-8. Port B Data Direction Register (PTBDD)

Table 6-7. PTBDD Register Field Descriptions

Field	Description
7:0 PTBDD[7:0]	Data Direction for Port B Bits — These read/write bits control the direction of port B pins and what is read for PTBD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port B bit n and PTBD reads return the contents of PTBDn.

6.4.2.3 Port B Pull Enable Register (PTBPE)

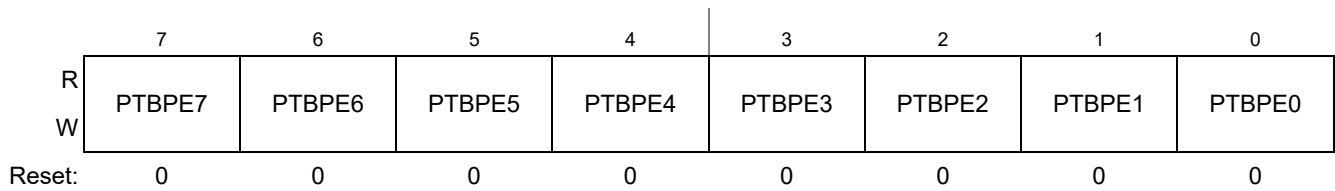


Figure 6-9. Internal Pull Enable for Port B Register (PTBPE)

Table 6-8. PTBPE Register Field Descriptions

Field	Description
7:0 PTBPE[7:0]	Internal Pull Enable for Port B Bits — Each of these control bits determines if the internal pullup or pulldown device is enabled for the associated PTB pin. For port B pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pullup/pulldown device disabled for port B bit n. 1 Internal pullup/pulldown device enabled for port B bit n.

6.4.2.4 Port B Slew Rate Enable Register (PTBSE)

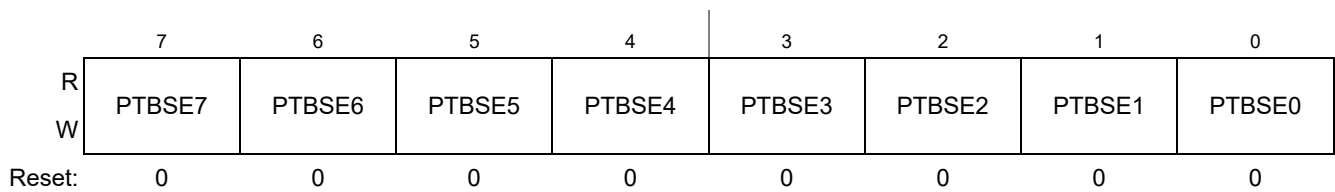


Figure 6-10. Slew Rate Enable for Port B Register (PTBSE)

Table 6-9. PTBSE Register Field Descriptions

Field	Description
7:0 PTBSE[7:0]	Output Slew Rate Enable for Port B Bits — Each of these control bits determines if the output slew rate control is enabled for the associated PTB pin. For port B pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port B bit n. 1 Output slew rate control enabled for port B bit n.

6.4.2.5 Port B Drive Strength Selection Register (PTBDS)

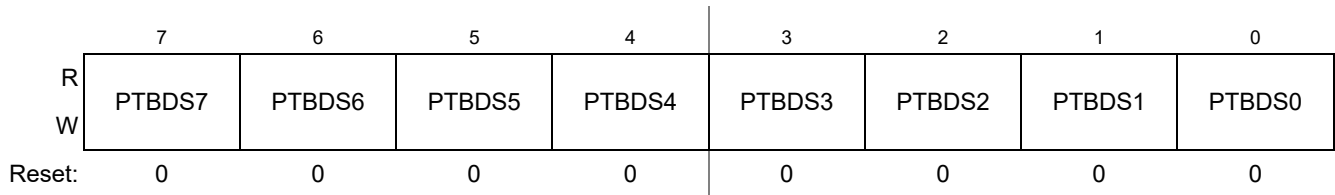


Figure 6-11. Drive Strength Selection for Port B Register (PTBDS)

Table 6-10. PTBDS Register Field Descriptions

Field	Description
7:0 PTBDS[7:0]	Output Drive Strength Selection for Port B Bits — Each of these control bits selects between low and high output drive for the associated PTB pin. For port B pins that are configured as inputs, these bits have no effect. 0 Low output drive strength selected for port B bit n. 1 High output drive strength selected for port B bit n.

6.4.3 Port C Registers

Port C is controlled by the registers listed below.

6.4.3.1 Port C Data Register (PTCD)

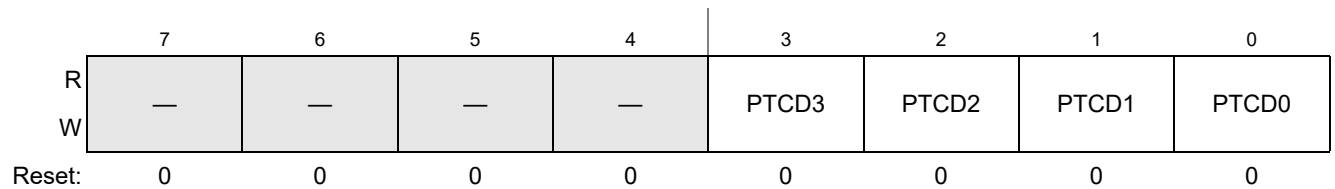


Figure 6-12. Port C Data Register (PTCD)

Table 6-11. PTCDD Register Field Descriptions

Field	Description
3:0 PTCD[3:0]	Port C Data Register Bits — For port C pins that are inputs, reads return the logic level on the pin. For port C pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port C pins that are configured as outputs, the logic level is driven out of the corresponding MCU pin. Reset forces PTCDD to all 0s, but these 0s are not driven out of the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

6.4.3.2 Port C Data Direction Register (PTCDD)

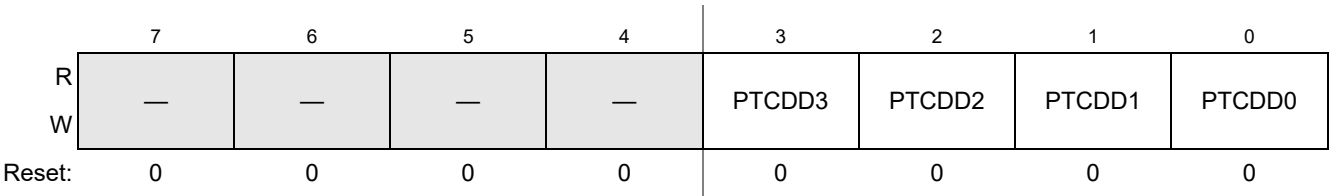


Figure 6-13. Port C Data Direction Register (PTCDD)

Table 6-12. PTCDD Register Field Descriptions

Field	Description
3:0 PTCDD[3:0]	Data Direction for Port C Bits — These read/write bits control the direction of port C pins and what is read for PTCDD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port C bit n and PTCDD reads return the contents of PTCDDn.

6.4.3.3 Port C Pull Enable Register (PTCPE)

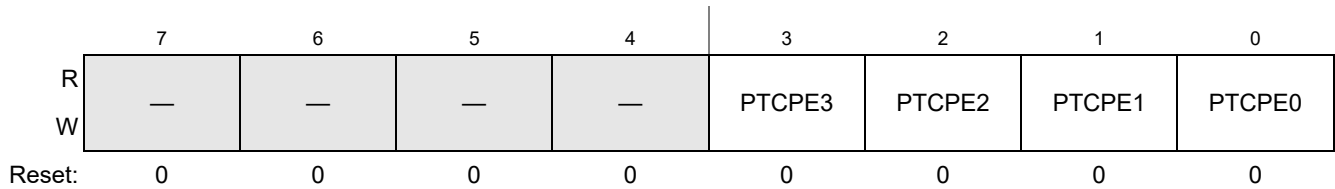


Figure 6-14. Internal Pull Enable for Port C Register (PTCPE)

Table 6-13. PTCPE Register Field Descriptions

Field	Description
3:0 PTCPE[3:0]	Internal Pull Enable for Port C Bits — Each of these control bits determines if the internal pullup device is enabled for the associated PTC pin. For port C pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pullup device disabled for port C bit n. 1 Internal pullup device enabled for port C bit n.

6.4.3.4 Port C Slew Rate Enable Register (PTCSE)

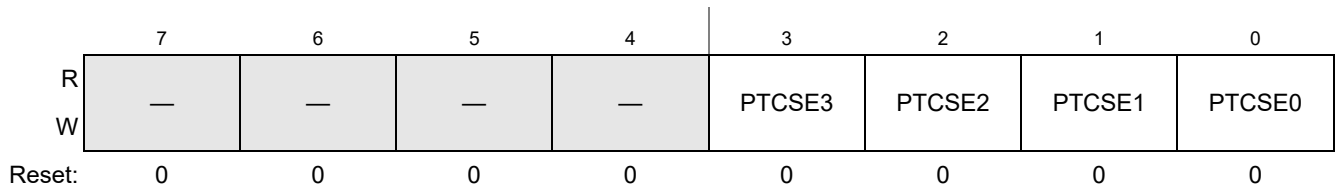


Figure 6-15. Slew Rate Enable for Port C Register (PTCSE)

Table 6-14. PTCSE Register Field Descriptions

Field	Description
3:0 PTCSE[3:0]	Output Slew Rate Enable for Port C Bits — Each of these control bits determines if the output slew rate control is enabled for the associated PTC pin. For port C pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port C bit n. 1 Output slew rate control enabled for port C bit n.

6.4.3.5 Port C Drive Strength Selection Register (PTCDS)

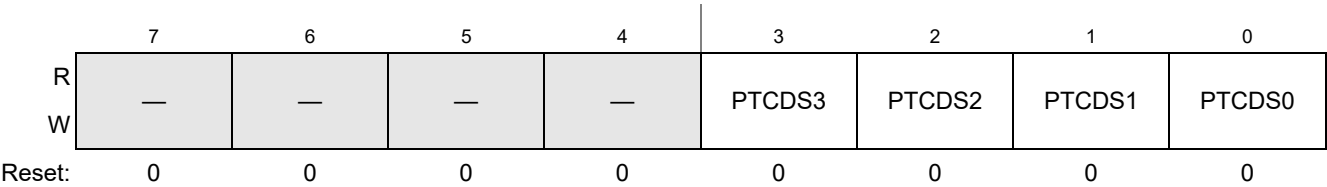


Figure 6-16. Drive Strength Selection for Port C Register (PTCDS)

Table 6-15. PTCDS Register Field Descriptions

Field	Description
3:0 PTCDS[3:0]	Output Drive Strength Selection for Port C Bits — Each of these control bits selects between low and high output drive for the associated PTC pin. For port C pins that are configured as inputs, these bits have no effect. 0 Low output drive strength selected for port C bit n. 1 High output drive strength selected for port C bit n.

Chapter 7

Keyboard Interrupt (S08KBIV2)

7.1 Introduction

The keyboard interrupt (KBI) module provides up to eight independently enabled external interrupt sources.

7.1.1 KBI Clock Gating

The bus clock to the KBI can be gated on and off using the KBI bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, this bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, Peripheral Clock Gating](#) for details.

7.1.2 Features

The KBI features include:

- Up to eight keyboard interrupt pins with individual pin enable bits.
- Each keyboard interrupt pin is programmable as falling edge (or rising edge) only, or both falling edge and low level (or both rising edge and high level) interrupt sensitivity.
- One software enabled keyboard interrupt.
- Exit from low-power modes.

7.1.3 Modes of Operation

This section defines the KBI operation in wait, stop, and background debug modes.

7.1.3.1 KBI in Wait Mode

The KBI continues to operate in wait mode if enabled before executing the WAIT instruction. Therefore, an enabled KBI pin ($KBPEX = 1$) can be used to bring the MCU out of wait mode if the KBI interrupt is enabled ($KBIE = 1$).

7.1.3.2 KBI in Stop Modes

The KBI operates asynchronously in stop3 mode if enabled before executing the STOP instruction. Therefore, an enabled KBI pin ($KBPEX = 1$) can be used to bring the MCU out of stop3 mode if the KBI interrupt is enabled ($KBIE = 1$).

During stop2 mode, the KBI is disabled. Upon wakeup from stop2 mode, the KBI module will be in the reset state.

7.1.3.3 KBI in Active Background Mode

When the microcontroller is in active background mode, the KBI will continue to operate normally.

7.1.4 Block Diagram

The block diagram for the keyboard interrupt module is shown [Figure 7-1](#).

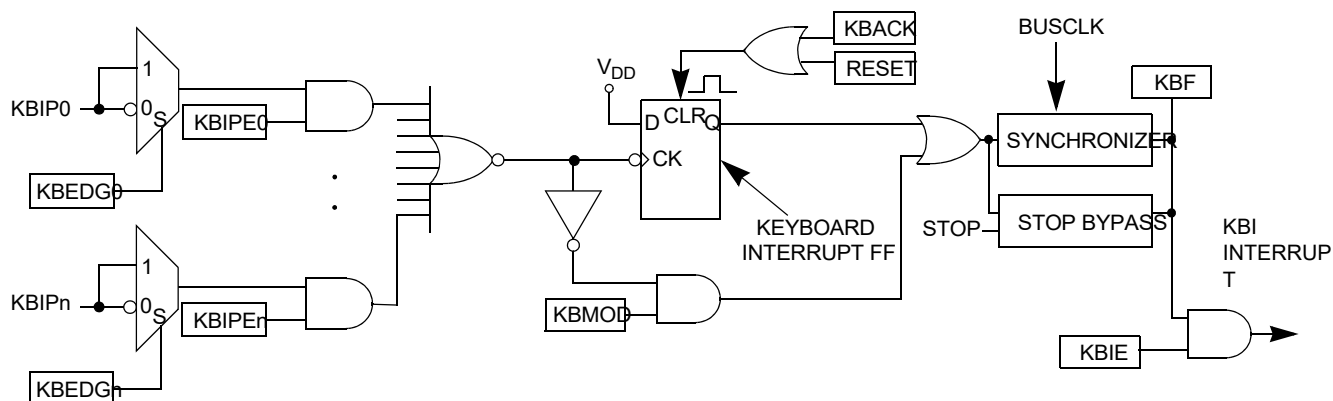


Figure 7-1. Keyboard Interrupt (KBI) Block Diagram

7.2 External Signal Description

The KBI input pins can be used to detect either falling edges, or both falling edge and low level interrupt requests. The KBI input pins can also be used to detect either rising edges, or both rising edge and high level interrupt requests.

Table 7-1. KBI Pin Mapping

Port pin	PTB3	PTB2	PTB1	PTB0	PTA3	PTA2	PTA1	PTA0
KBI pin	KBIP7	KBIP6	KBIP5	KBIP4	KBIP3	KBIP2	KBIP1	KBIP0

7.3 Register Definition

The KBI includes three registers:

- An 8-bit pin status and control register.
- An 8-bit pin enable register.
- An 8-bit edge select register.

Refer to the direct-page register summary in [Chapter 4, Memory](#) for the absolute address assignments for all KBI registers. This section refers to registers and control bits only by their names and relative address offsets.

7.3.1 KBI Interrupt Status and Control Register (KBISC)

	7	6	5	4	3	2	1	0
R	0	0	0	0	KBF	0	KBIE	KBIMOD
W						KBACK		
Reset:	0	0	0	0	0	0	0	0

Figure 7-2. KBI Interrupt Status and Control Register (KBISC)

Table 7-2. KBISC Register Field Descriptions

Field	Description
3 KBF	KBI Interrupt Flag — KBF indicates when a KBI interrupt is detected. Writes have no effect on KBF. 0 No KBI interrupt detected. 1 KBI interrupt detected.
2 KBACK	KBI Interrupt Acknowledge — Writing a 1 to KBACK is part of the flag clearing mechanism. KBACK always reads as 0.
1 KBIE	KBI Interrupt Enable — KBIE determines whether a KBI interrupt is requested. 0 KBI interrupt request not enabled. 1 KBI interrupt request enabled.
0 KBIMOD	KBI Detection Mode — KBIMOD (along with the KBIES bits) controls the detection mode of the KBI interrupt pins. 0 KBI pins detect edges only. 1 KBI pins detect both edges and levels.

7.3.2 KBI Interrupt Pin Select Register (KBIPE)

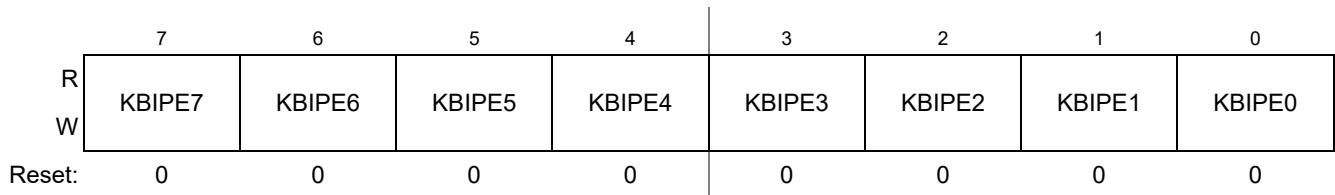


Figure 7-3. KBI Interrupt Pin Select Register (KBIPE)

Table 7-3. KBIPE Register Field Descriptions

Field	Description
7:0 KBIPE[7:0]	KBI Interrupt Pin Selects — Each of the KBIPE _n bits enable the corresponding KBI interrupt pin. 0 Pin not enabled as interrupt. 1 Pin enabled as interrupt.

7.3.3 KBI Interrupt Edge Select Register (KBIES)

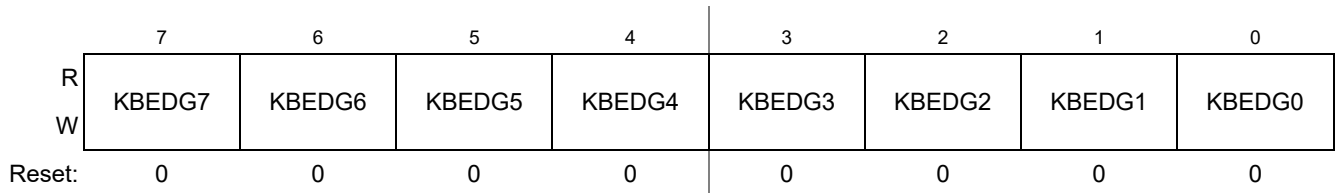


Figure 7-4. KBI Edge Select Register (KBIES)

Table 7-4. KBIES Register Field Descriptions

Field	Description
7:0 KBEDG[7:0]	KBI Edge Selects — Each of the KBEDG _n bits serves a dual purpose by selecting the polarity of the active interrupt edge as well as selecting a pullup or pulldown device if enabled. 0 A pullup device is connected to the associated pin and detects falling edge/low level for interrupt generation. 1 A pulldown device is connected to the associated pin and detects rising edge/high level for interrupt generation.

7.4 Functional Description

This on-chip peripheral module is called a keyboard interrupt (KBI) module because originally it was designed to simplify the connection and use of row-column matrices of keyboard switches. However, these inputs are also useful as extra external interrupt inputs and as an external means of waking the MCU from stop or wait low-power modes. The KBI module allows up to eight pins to act as additional interrupt sources.

Writing to the KBIPE_n bits in the keyboard interrupt pin enable register (KBIPE) independently enables or disables each port pin. Each port can be configured as edge sensitive or edge and level sensitive based on the KBIMOD bit in the keyboard interrupt status and control register (KBISC). Edge sensitivity can be software programmed to be either falling or rising; the level can be either low or high. The polarity of the

edge or edge and level sensitivity is selected using the KBEDGn bits in the keyboard interrupt edge select register (KBIES).

Synchronous logic is used to detect edges. Prior to detecting an edge, enabled port inputs must be at the deasserted logic level. A falling edge is detected when an enabled port input signal is seen as a logic 1 (the deasserted level) during one bus cycle and then a logic 0 (the asserted level) during the next cycle. A rising edge is detected when the input signal is seen as a logic 0 during one bus cycle and then a logic 1 during the next cycle.

7.4.1 Edge Only Sensitivity

A valid edge on an enabled port pin will set KBF in KBISC. If KBIE in KBISC is set, an interrupt request will be presented to the CPU. Clearing of KBF is accomplished by writing a 1 to KBACK in KBISC.

7.4.2 Edge and Level Sensitivity

A valid edge or level on an enabled port pin will set KBF in KBISC. If KBIE in KBISC is set, an interrupt request will be presented to the CPU. Clearing of KBF is accomplished by writing a 1 to KBACK in KBISC provided all enabled port inputs are at their deasserted levels. KBF will remain set if any enabled port pin is asserted while attempting to clear by writing a 1 to KBACK.

7.4.3 Pullup/Pulldown Resistors

The keyboard interrupt pins can be configured to use an internal pullup/pulldown resistor using the associated I/O port pullup enable register. If an internal resistor is enabled, the KBIES register is used to select whether the resistor is a pullup (KBEDGn = 0) or a pulldown (KBEDGn = 1).

7.4.4 Keyboard Interrupt Initialization

When an interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request during pin interrupt initialization, the user must do the following:

1. Mask interrupts by clearing KBIE in KBISC.
2. Select the pin polarity by setting the appropriate KBEDGn bits in KBIES.
3. If using internal pullup/pulldown device, configure the associated pull enable bits in KBIPE.
4. Enable the interrupt pins by setting the appropriate KBIPEn bits in KBIPE.
5. Write to KBACK in KBISC to clear any false interrupts.
6. Set KBIE in KBISC to enable interrupts.

Chapter 8

Central Processor Unit (S08CPUV5)

8.1 Introduction

This section provides summary information about the registers, addressing modes, and instruction set of the CPU of the HCS08 Family. For a more detailed discussion, refer to the *HCS08 Family Reference Manual, volume 1*, NXP Semiconductor document order number HCS08RMV1/D.

The HCS08 CPU is fully source- and object-code-compatible with the M68HC08 CPU. Several instructions and enhanced addressing modes were added to improve C compiler efficiency and to support a new background debug system which replaces the monitor mode of earlier M68HC08 microcontrollers (MCU).

8.1.1 Features

Features of the HCS08 CPU include:

- Object code fully upward-compatible with M68HC05 and M68HC08 Families
- All registers and memory are mapped to a single 64-Kbyte address space
- 16-bit stack pointer (any size stack anywhere in 64-KB CPU address space)
- 16-bit index register (H:X) with powerful indexed addressing modes
- 8-bit accumulator (A)
- Many instructions treat X as a second general-purpose 8-bit register
- Seven addressing modes:
 - Inherent — Operands in internal registers
 - Relative — 8-bit signed offset to branch destination
 - Immediate — Operand in next object code byte(s)
 - Direct — Operand in memory at 0x0000–0x00FF
 - Extended — Operand anywhere in 64-Kbyte address space
 - Indexed relative to H:X — Five submodes including auto increment
 - Indexed relative to SP — Improves C efficiency dramatically
- Memory-to-memory data move instructions with four address mode combinations
- Overflow, half-carry, negative, zero, and carry condition codes support conditional branching on the results of signed, unsigned, and binary-coded decimal (BCD) operations
- Efficient bit manipulation instructions
- Fast 8-bit by 8-bit multiply and 16-bit by 8-bit divide instructions
- STOP and WAIT instructions to invoke low-power operating modes

8.2 Programmer's Model and CPU Registers

Figure 8-1 shows the five CPU registers. CPU registers are not part of the memory map.

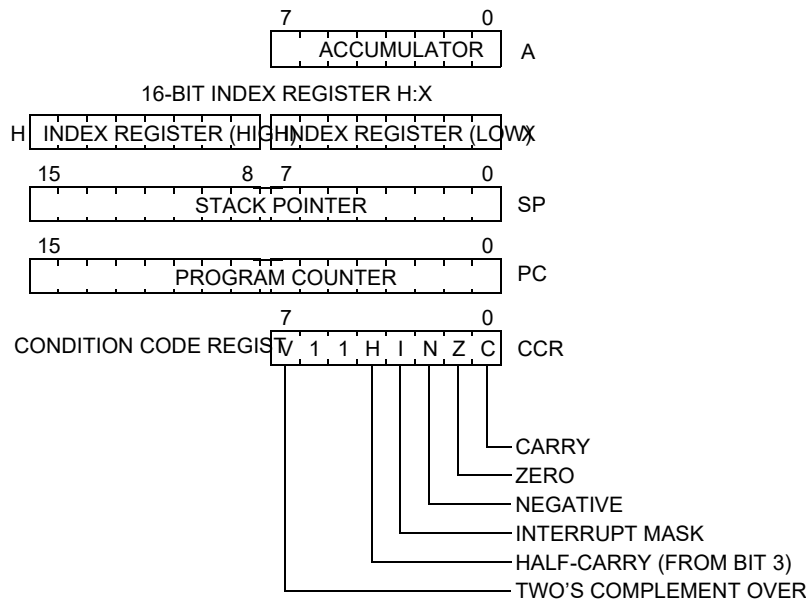


Figure 8-1. CPU Registers

8.2.1 Accumulator (A)

The A accumulator is a general-purpose 8-bit register. One operand input to the arithmetic logic unit (ALU) is connected to the accumulator and the ALU results are often stored into the A accumulator after arithmetic and logical operations. The accumulator can be loaded from memory using various addressing modes to specify the address where the loaded data comes from, or the contents of A can be stored to memory using various addressing modes to specify the address where data from A will be stored.

Reset has no effect on the contents of the A accumulator.

8.2.2 Index Register (H:X)

This 16-bit register is actually two separate 8-bit registers (H and X), which often work together as a 16-bit address pointer where H holds the upper byte of an address and X holds the lower byte of the address. All indexed addressing mode instructions use the full 16-bit value in H:X as an index reference pointer; however, for compatibility with the earlier M68HC05 Family, some instructions operate only on the low-order 8-bit half (X).

Many instructions treat X as a second general-purpose 8-bit register that can be used to hold 8-bit data values. X can be cleared, incremented, decremented, complemented, negated, shifted, or rotated. Transfer instructions allow data to be transferred from A or transferred to A where arithmetic and logical operations can then be performed.

For compatibility with the earlier M68HC05 Family, H is forced to 0x00 during reset. Reset has no effect on the contents of X.

8.2.3 Stack Pointer (SP)

This 16-bit address pointer register points at the next available location on the automatic last-in-first-out (LIFO) stack. The stack may be located anywhere in the 64-Kbyte address space that has RAM and can be any size up to the amount of available RAM. The stack is used to automatically save the return address for subroutine calls, the return address and CPU registers during interrupts, and for local variables. The AIS (add immediate to stack pointer) instruction adds an 8-bit signed immediate value to SP. This is most often used to allocate or deallocate space for local variables on the stack.

SP is forced to 0x00FF at reset for compatibility with the earlier M68HC05 Family. HCS08 programs normally change the value in SP to the address of the last location (highest address) in on-chip RAM during reset initialization to free up direct page RAM (from the end of the on-chip registers to 0x00FF).

The RSP (reset stack pointer) instruction was included for compatibility with the M68HC05 Family and is seldom used in new HCS08 programs because it only affects the low-order half of the stack pointer.

8.2.4 Program Counter (PC)

The program counter is a 16-bit register that contains the address of the next instruction or operand to be fetched.

During normal program execution, the program counter automatically increments to the next sequential memory location every time an instruction or operand is fetched. Jump, branch, interrupt, and return operations load the program counter with an address other than that of the next sequential location. This is called a change-of-flow.

During reset, the program counter is loaded with the reset vector that is located at 0xFFFFE and 0xFFFF. The vector stored there is the address of the first instruction that will be executed after exiting the reset state.

8.2.5 Condition Code Register (CCR)

The 8-bit condition code register contains the interrupt mask (I) and five flags that indicate the results of the instruction just executed. Bits 6 and 5 are set permanently to 1. The following paragraphs describe the functions of the condition code bits in general terms. For a more detailed explanation of how each instruction sets the CCR bits, refer to the *HCS08 Family Reference Manual, volume 1*, NXP document order number HCS08RMv1.

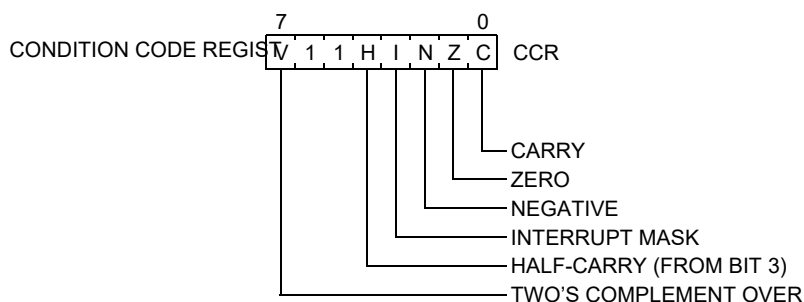


Figure 8-2. Condition Code Register

Table 8-1. CCR Register Field Descriptions

Field	Description
7 V	Two's Complement Overflow Flag — The CPU sets the overflow flag when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow flag. 0 No overflow 1 Overflow
4 H	Half-Carry Flag — The CPU sets the half-carry flag when a carry occurs between accumulator bits 3 and 4 during an add-without-carry (ADD) or add-with-carry (ADC) operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations. The DAA instruction uses the states of the H and C condition code bits to automatically add a correction value to the result from a previous ADD or ADC on BCD operands to correct the result to a valid BCD value. 0 No carry between bits 3 and 4 1 Carry between bits 3 and 4
3 I	Interrupt Mask Bit — When the interrupt mask is set, all maskable CPU interrupts are disabled. CPU interrupts are enabled when the interrupt mask is cleared. When a CPU interrupt occurs, the interrupt mask is set automatically after the CPU registers are saved on the stack, but before the first instruction of the interrupt service routine is executed. Interrupts are not recognized at the instruction boundary after any instruction that clears I (CLI or TAP). This ensures that the next instruction after a CLI or TAP will always be executed without the possibility of an intervening interrupt, provided I was set. 0 Interrupts enabled 1 Interrupts disabled
2 N	Negative Flag — The CPU sets the negative flag when an arithmetic operation, logic operation, or data manipulation produces a negative result, setting bit 7 of the result. Simply loading or storing an 8-bit or 16-bit value causes N to be set if the most significant bit of the loaded or stored value was 1. 0 Non-negative result 1 Negative result
1 Z	Zero Flag — The CPU sets the zero flag when an arithmetic operation, logic operation, or data manipulation produces a result of 0x00 or 0x0000. Simply loading or storing an 8-bit or 16-bit value causes Z to be set if the loaded or stored value was all 0s. 0 Non-zero result 1 Zero result
0 C	Carry/Borrow Flag — The CPU sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some instructions — such as bit test and branch, shift, and rotate — also clear or set the carry/borrow flag. 0 No carry out of bit 7 1 Carry out of bit 7

8.3 Addressing Modes

Addressing modes define the way the CPU accesses operands and data. In the HCS08, memory, status and control registers, and input/output (I/O) ports share a single 64-Kbyte CPU address space. This arrangement means that the same instructions that access variables in RAM can also be used to access I/O and control registers or nonvolatile program space.

Some instructions use more than one addressing mode. For instance, move instructions use one addressing mode to specify the source operand and a second addressing mode to specify the destination address. Instructions such as BRCLR, BRSET, CBEQ, and DBNZ use one addressing mode to specify the location of an operand for a test and then use relative addressing mode to specify the branch destination address when the tested condition is true. For BRCLR, BRSET, CBEQ, and DBNZ, the addressing mode listed in the instruction set tables is the addressing mode needed to access the operand to be tested, and relative addressing mode is implied for the branch destination.

8.3.1 Inherent Addressing Mode (INH)

In this addressing mode, operands needed to complete the instruction (if any) are located within CPU registers so the CPU does not need to access memory to get any operands.

8.3.2 Relative Addressing Mode (REL)

Relative addressing mode is used to specify the destination location for branch instructions. A signed 8-bit offset value is located in the memory location immediately following the opcode. During execution, if the branch condition is true, the signed offset is sign-extended to a 16-bit value and is added to the current contents of the program counter, which causes program execution to continue at the branch destination address.

8.3.3 Immediate Addressing Mode (IMM)

In immediate addressing mode, the operand needed to complete the instruction is included in the object code immediately following the instruction opcode in memory. In the case of a 16-bit immediate operand, the high-order byte is located in the next memory location after the opcode, and the low-order byte is located in the next memory location after that.

8.3.4 Direct Addressing Mode (DIR)

In direct addressing mode, the instruction includes the low-order eight bits of an address in the direct page (0x0000–0x00FF). During execution a 16-bit address is formed by concatenating an implied 0x00 for the high-order half of the address and the direct address from the instruction to get the 16-bit address where the desired operand is located. This is faster and more memory efficient than specifying a complete 16-bit address for the operand.

8.3.5 Extended Addressing Mode (EXT)

In extended addressing mode, the full 16-bit address of the operand is located in the next two bytes of program memory after the opcode (high byte first).

8.3.6 Indexed Addressing Mode

Indexed addressing mode has seven variations including five that use the 16-bit H:X index register pair and two that use the stack pointer as the base reference.

8.3.6.1 Indexed, No Offset (IX)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair as the address of the operand needed to complete the instruction.

8.3.6.2 Indexed, No Offset with Post Increment (IX+)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair as the address of the operand needed to complete the instruction. The index register pair is then incremented ($H:X = H:X + 0x0001$) after the operand has been fetched. This addressing mode is only used for MOV and CBEQ instructions.

8.3.6.3 Indexed, 8-Bit Offset (IX1)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction.

8.3.6.4 Indexed, 8-Bit Offset with Post Increment (IX1+)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction. The index register pair is then incremented ($H:X = H:X + 0x0001$) after the operand has been fetched. This addressing mode is used only for the CBEQ instruction.

8.3.6.5 Indexed, 16-Bit Offset (IX2)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus a 16-bit offset included in the instruction as the address of the operand needed to complete the instruction.

8.3.6.6 SP-Relative, 8-Bit Offset (SP1)

This variation of indexed addressing uses the 16-bit value in the stack pointer (SP) plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction.

8.3.6.7 SP-Relative, 16-Bit Offset (SP2)

This variation of indexed addressing uses the 16-bit value in the stack pointer (SP) plus a 16-bit offset included in the instruction as the address of the operand needed to complete the instruction.

8.4 Special Operations

The CPU performs a few special operations that are similar to instructions but do not have opcodes like other CPU instructions. In addition, a few instructions such as STOP and WAIT directly affect other MCU circuitry. This section provides additional information about these operations.

8.4.1 Reset Sequence

Reset can be caused by a power-on-reset (POR) event, internal conditions such as the COP (computer operating properly) watchdog, or by assertion of an external active-low reset pin. When a reset event occurs, the CPU immediately stops whatever it is doing (the MCU does not wait for an instruction boundary before responding to a reset event). For a more detailed discussion about how the MCU recognizes resets and determines the source, refer to [Chapter 13, Real-Time Counter \(S08RTCV1\)](#).

The reset event is considered concluded when the sequence to determine whether the reset came from an internal source is done and when the reset pin is no longer asserted. At the conclusion of a reset event, the CPU performs a 6-cycle sequence to fetch the reset vector from 0xFFFFE and 0xFFFF and to fill the instruction queue in preparation for execution of the first program instruction.

8.4.2 Interrupt Sequence

When an interrupt is requested, the CPU completes the current instruction before responding to the interrupt. At this point, the program counter is pointing at the start of the next instruction, which is where the CPU should return after servicing the interrupt. The CPU responds to an interrupt by performing the same sequence of operations as for a software interrupt (SWI) instruction, except the address used for the vector fetch is determined by the highest priority interrupt that is pending when the interrupt sequence started.

The CPU sequence for an interrupt is:

1. Store the contents of PCL, PCH, X, A, and CCR on the stack, in that order.
2. Set the I bit in the CCR.
3. Fetch the high-order half of the interrupt vector.
4. Fetch the low-order half of the interrupt vector.
5. Delay for one free bus cycle.
6. Fetch three bytes of program information starting at the address indicated by the interrupt vector to fill the instruction queue in preparation for execution of the first instruction in the interrupt service routine.

After the CCR contents are pushed onto the stack, the I bit in the CCR is set to prevent other interrupts while in the interrupt service routine. Although it is possible to clear the I bit with an instruction in the

interrupt service routine, this would allow nesting of interrupts (which is not recommended because it leads to programs that are difficult to debug and maintain).

For compatibility with the earlier M68HC05 MCUs, the high-order half of the H:X index register pair (H) is not saved on the stack as part of the interrupt sequence. The user must use a PSHH instruction at the beginning of the service routine to save H and then use a PULH instruction just before the RTI that ends the interrupt service routine. It is not necessary to save H if you are certain that the interrupt service routine does not use any instructions or auto-increment addressing modes that might change the value of H.

The software interrupt (SWI) instruction is like a hardware interrupt except that it is not masked by the global I bit in the CCR and it is associated with an instruction opcode within the program so it is not asynchronous to program execution.

8.4.3 Wait Mode Operation

The WAIT instruction enables interrupts by clearing the I bit in the CCR. It then halts the clocks to the CPU to reduce overall power consumption while the CPU is waiting for the interrupt or reset event that will wake the CPU from wait mode. When an interrupt or reset event occurs, the CPU clocks will resume and the interrupt or reset event will be processed normally.

If a serial BACKGROUND command is issued to the MCU through the background debug interface while the CPU is in wait mode, CPU clocks will resume and the CPU will enter active background mode where other serial background commands can be processed. This ensures that a host development system can still gain access to a target MCU even if it is in wait mode.

8.4.4 Stop Mode Operation

Usually, all system clocks, including the crystal oscillator (when used), are halted during stop mode to minimize power consumption. In such systems, external circuitry is needed to control the time spent in stop mode and to issue a signal to wake the target MCU when it is time to resume processing. Unlike the earlier M68HC05 and M68HC08 MCUs, the HCS08 can be configured to keep a minimum set of clocks running in stop mode. This optionally allows an internal periodic signal to wake the target MCU from stop mode.

When a host debug system is connected to the background debug pin (BKGD) and the ENBDM control bit has been set by a serial command through the background interface (or because the MCU was reset into active background mode), the oscillator is forced to remain active when the MCU enters stop mode. In this case, if a serial BACKGROUND command is issued to the MCU through the background debug interface while the CPU is in stop mode, CPU clocks will resume and the CPU will enter active background mode where other serial background commands can be processed. This ensures that a host development system can still gain access to a target MCU even if it is in stop mode.

Recovery from stop mode depends on the particular HCS08 and whether the oscillator was stopped in stop mode. Refer to [Chapter 3, Modes of Operation](#) for more details.

8.4.5 BGND Instruction

The BGND instruction is new to the HCS08 compared to the M68HC08. BGND would not be used in normal user programs because it forces the CPU to stop processing user instructions and enter the active background mode. The only way to resume execution of the user program is through reset or by a host debug system issuing a GO, TRACE1, or TAGGO serial command through the background debug interface.

Software-based breakpoints can be set by replacing an opcode at the desired breakpoint address with the BGND opcode. When the program reaches this breakpoint address, the CPU is forced to active background mode rather than continuing the user program.

8.5 HCS08 Instruction Set Summary

Instruction Set Summary Nomenclature

The nomenclature listed here is used in the instruction descriptions in [Table 8-2](#).

Operators

()	=	Contents of register or memory location shown inside parentheses
←	=	Is loaded with (read: “gets”)
&	=	Boolean AND
	=	Boolean OR
⊕	=	Boolean exclusive-OR
×	=	Multiply
÷	=	Divide
:	=	Concatenate
+	=	Add
–	=	Negate (two’s complement)

CPU registers

A	=	Accumulator
CCR	=	Condition code register
H	=	Index register, higher order (most significant) 8 bits
X	=	Index register, lower order (least significant) 8 bits
PC	=	Program counter
PCH	=	Program counter, higher order (most significant) 8 bits
PCL	=	Program counter, lower order (least significant) 8 bits
SP	=	Stack pointer

Memory and addressing

M	=	A memory location or absolute data, depending on addressing mode
M:M + 0x0001	=	A 16-bit value in two consecutive memory locations. The higher-order (most significant) 8 bits are located at the address of M, and the lower-order (least significant) 8 bits are located at the next higher sequential address.

Condition code register (CCR) bits

V	=	Two’s complement overflow indicator, bit 7
H	=	Half carry, bit 4
I	=	Interrupt mask, bit 3
N	=	Negative indicator, bit 2
Z	=	Zero indicator, bit 1
C	=	Carry/borrow, bit 0 (carry out of bit 7)

CCR activity notation

–	=	Bit not affected
---	---	------------------

0	=	Bit forced to 0
1	=	Bit forced to 1
→	=	Bit set or cleared according to results of operation
U	=	Undefined after the operation

Machine coding notation

dd	=	Low-order 8 bits of a direct address 0x0000–0x00FF (high byte assumed to be 0x00)
ee	=	Upper 8 bits of 16-bit offset
ff	=	Lower 8 bits of 16-bit offset or 8-bit offset
ii	=	One byte of immediate data
jj	=	High-order byte of a 16-bit immediate data value
kk	=	Low-order byte of a 16-bit immediate data value
hh	=	High-order byte of 16-bit extended address
ll	=	Low-order byte of 16-bit extended address
pg	=	Page
rr	=	Relative offset

Source form

Everything in the source forms columns, *except expressions in italic characters*, is literal information that must appear in the assembly source file exactly as shown. The initial 3- to 5-letter mnemonic is always a literal expression. All commas, pound signs (#), parentheses, and plus signs (+) are literal characters.

<i>n</i>	—	Any label or expression that evaluates to a single integer in the range 0–7
<i>opr8i</i>	—	Any label or expression that evaluates to an 8-bit immediate value
<i>opr16i</i>	—	Any label or expression that evaluates to a 16-bit immediate value
<i>opr8a</i>	—	Any label or expression that evaluates to an 8-bit value. The instruction treats this 8-bit value as the low order 8 bits of an address in the direct page of the 64-Kbyte address space (0x00xx).
<i>opr16a</i>	—	Any label or expression that evaluates to a 16-bit value. The instruction treats this value as an address in the 64-Kbyte address space.
<i>opr8</i>	—	Any label or expression that evaluates to an unsigned 8-bit value, used for indexed addressing
<i>opr16</i>	—	Any label or expression that evaluates to a 16-bit value. Because the HCS08 has a 16-bit address bus, this can be either a signed or an unsigned value.
<i>page</i>	—	Any label or expression that evaluates to a valid bank number for the PPAGE register.
<i>rel</i>	—	Any label or expression that refers to an address that is within –128 to +127 locations from the next address after the last byte of object code for the current instruction. The assembler will calculate the 8-bit signed offset and include it in the object code for this instruction.

Address modes

INH	=	Inherent (no operands)
IMM	=	8-bit or 16-bit immediate

DIR	=	8-bit direct
EXT	=	16-bit extended
IX	=	16-bit indexed no offset
IX+	=	16-bit indexed no offset, post increment (CBEQ and MOV only)
IX1	=	16-bit indexed with 8-bit offset from H:X
IX1+	=	16-bit indexed with 8-bit offset, post increment (CBEQ only)
IX2	=	16-bit indexed with 16-bit offset from H:X
REL	=	8-bit relative offset
SP1	=	Stack pointer with 8-bit offset
SP2	=	Stack pointer with 16-bit offset

Table 8-2. HCS08 Instruction Set Summary (Sheet 1 of 7)

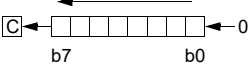
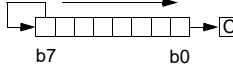
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
ADC #opr8i ADC opr8a ADC opr16a ADC oprx16,X ADC oprx8,X ADC ,X ADC oprx16,SP ADC oprx8,SP	Add with Carry	$A \leftarrow (A) + (M) + (C)$	→	→	—	→	→	→	IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 B9 dd C9 hh ll D9 ee ff E9 ff F9 9ED9 ee ff 9EE9 ff	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 3 5 4
ADD #opr8i ADD opr8a ADD opr16a ADD oprx16,X ADD oprx8,X ADD ,X ADD oprx16,SP ADD oprx8,SP	Add without Carry	$A \leftarrow (A) + (M)$	→	→	—	→	→	→	IMM DIR EXT IX2 IX1 IX SP2 SP1	AB BB dd CB hh ll DB ee ff EB ff FB 9EDB ee ff 9EEB ff	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 3 5 4
AIS #opr8i	Add Immediate Value (Signed) to Stack Pointer	$SP \leftarrow (SP) + (M)$ M is sign extended to a 16-bit value	—	—	—	—	—	—	IMM	A7	ii	2
AIX #opr8i	Add Immediate Value (Signed) to Index Register (H:X)	$H:X \leftarrow (H:X) + (M)$ M is sign extended to a 16-bit value	—	—	—	—	—	—	IMM	AF	ii	2
AND #opr8i AND opr8a AND opr16a AND oprx16,X AND oprx8,X AND ,X AND oprx16,SP AND oprx8,SP	Logical AND	$A \leftarrow (A) \& (M)$	0	—	—	→	→	—	IMM DIR EXT IX2 IX1 IX SP2 SP1	A4 B4 dd C4 hh ll D4 ee ff E4 ff F4 9ED4 ee ff 9EE4 ff	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 3 5 4
ASL opr8a ASLA ASLX ASL oprx8,X ASL ,X ASL oprx8,SP	Arithmetic Shift Left (Same as LSL)		→	—	—	→	→	→	DIR INH INH IX1 IX SP1	38 48 58 68 ff 78 9E68 ff	dd ff ff ff	5 1 1 5 4 6
ASR opr8a ASRA ASRX ASR oprx8,X ASR ,X ASR oprx8,SP	Arithmetic Shift Right		→	—	—	→	→	→	DIR INH INH IX1 IX SP1	37 47 57 67 ff 77 9E67 ff	dd ff ff ff	5 1 1 5 4 6
BCC rel	Branch if Carry Bit Clear	Branch if (C) = 0	—	—	—	—	—	—	REL	24	rr	3

Table 8-2. HCS08 Instruction Set Summary (Sheet 2 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
BCLR <i>n,opr8a</i>	Clear Bit n in Memory	$M_n \leftarrow 0$	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BCS <i>rel</i>	Branch if Carry Bit Set (Same as BLO)	Branch if (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BEQ <i>rel</i>	Branch if Equal	Branch if (Z) = 1	–	–	–	–	–	–	REL	27	rr	3
BGE <i>rel</i>	Branch if Greater Than or Equal To (Signed Operands)	Branch if $(N \oplus V) = 0$	–	–	–	–	–	–	REL	90	rr	3
BGND	Enter Active Background if ENBDM = 1	Waits For and Processes BDM Commands Until GO, TRACE1, or TAGGO	–	–	–	–	–	–	INH	82		5+
BGT <i>rel</i>	Branch if Greater Than (Signed Operands)	Branch if $(Z) (N \oplus V) = 0$	–	–	–	–	–	–	REL	92	rr	3
BHCC <i>rel</i>	Branch if Half Carry Bit Clear	Branch if (H) = 0	–	–	–	–	–	–	REL	28	rr	3
BHCS <i>rel</i>	Branch if Half Carry Bit Set	Branch if (H) = 1	–	–	–	–	–	–	REL	29	rr	3
BHI <i>rel</i>	Branch if Higher	Branch if $(C) (Z) = 0$	–	–	–	–	–	–	REL	22	rr	3
BHS <i>rel</i>	Branch if Higher or Same (Same as BCC)	Branch if (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH <i>rel</i>	Branch if IRQ Pin High	Branch if IRQ pin = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL <i>rel</i>	Branch if IRQ Pin Low	Branch if IRQ pin = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT <i>#opr8i</i> BIT <i>opr8a</i> BIT <i>opr16a</i> BIT <i>opr16,X</i> BIT <i>opr8,X</i> BIT <i>,X</i> BIT <i>opr16,SP</i> BIT <i>opr8,SP</i>	Bit Test	(A) & (M) (CCR Updated but Operands Not Changed)	0	–	–	→	→	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A5 B5 C5 D5 E5 F5 9ED5 9EE5	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 3 5 4
BLE <i>rel</i>	Branch if Less Than or Equal To (Signed Operands)	Branch if $(Z) (N \oplus V) = 1$	–	–	–	–	–	–	REL	93	rr	3
BLO <i>rel</i>	Branch if Lower (Same as BCS)	Branch if (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BLS <i>rel</i>	Branch if Lower or Same	Branch if $(C) (Z) = 1$	–	–	–	–	–	–	REL	23	rr	3
BLT <i>rel</i>	Branch if Less Than (Signed Operands)	Branch if $(N \oplus V) = 1$	–	–	–	–	–	–	REL	91	rr	3
BMC <i>rel</i>	Branch if Interrupt Mask Clear	Branch if (I) = 0	–	–	–	–	–	–	REL	2C	rr	3
BMI <i>rel</i>	Branch if Minus	Branch if (N) = 1	–	–	–	–	–	–	REL	2B	rr	3
BMS <i>rel</i>	Branch if Interrupt Mask Set	Branch if (I) = 1	–	–	–	–	–	–	REL	2D	rr	3
BNE <i>rel</i>	Branch if Not Equal	Branch if (Z) = 0	–	–	–	–	–	–	REL	26	rr	3
BPL <i>rel</i>	Branch if Plus	Branch if (N) = 0	–	–	–	–	–	–	REL	2A	rr	3
BRA <i>rel</i>	Branch Always	No Test	–	–	–	–	–	–	REL	20	rr	3

Table 8-2. HCS08 Instruction Set Summary (Sheet 3 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
BRCLR <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Clear	Branch if (Mn) = 0	–	–	–	–	–	→	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN <i>rel</i>	Branch Never	Uses 3 Bus Cycles	–	–	–	–	–	–	REL	21	rr	3
BRSET <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Set	Branch if (Mn) = 1	–	–	–	–	–	→	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET <i>n,opr8a</i>	Set Bit <i>n</i> in Memory	Mn ← 1	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BSR <i>rel</i>	Branch to Subroutine	PC ← (PC) + 0x0002 push (PCL); SP ← (SP) – 0x0001 push (PCH); SP ← (SP) – 0x0001 PC ← (PC) + <i>rel</i>	–	–	–	–	–	–	REL	AD	rr	5
CBEQ <i>opr8a,rel</i> CBEQA <i>#opr8i,rel</i> CBEQX <i>#opr8i,rel</i> CBEQ <i>opr8,X+,rel</i> CBEQ <i>,X+,rel</i> CBEQ <i>opr8,SP,rel</i>	Compare and Branch if Equal	Branch if (A) = (M) Branch if (A) = (M) Branch if (X) = (M) Branch if (A) = (M) Branch if (A) = (M) Branch if (A) = (M)	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr rr ff rr	5 4 4 5 5 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask Bit	I ← 0	–	–	0	–	–	–	INH	9A		1
CLR <i>opr8a</i> CLRA CLR X CLR H CLR <i>opr8,X</i> CLR <i>,X</i> CLR <i>opr8,SP</i>	Clear	M ← 0x00 A ← 0x00 X ← 0x00 H ← 0x00 M ← 0x00 M ← 0x00 M ← 0x00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd ff ff ff	5 1 1 1 5 4 6
CMP <i>#opr8i</i> CMP <i>opr8a</i> CMP <i>opr16a</i> CMP <i>opr16,X</i> CMP <i>opr8,X</i> CMP <i>,X</i> CMP <i>opr16,SP</i> CMP <i>opr8,SP</i>	Compare Accumulator with Memory	(A) – (M) (CCR Updated But Operands Not Changed)	→	–	–	→	→	→	IMM DIR EXT IX2 IX1 IX SP2 SP1	A1 B1 C1 D1 E1 F1 9ED1 9EE1	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 3 5 4
COM <i>opr8a</i> COMA COM X COM <i>opr8,X</i> COM <i>,X</i> COM <i>opr8,SP</i>	Complement (One's Complement)	M ← (M) = 0xFF – (M) A ← (A) = 0xFF – (A) X ← (X) = 0xFF – (X) M ← (M) = 0xFF – (M) M ← (M) = 0xFF – (M) M ← (M) = 0xFF – (M)	0	–	–	→	→	1	DIR INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd ff ff ff	5 1 1 5 4 6
CPHX <i>opr16a</i> CPHX <i>#opr16i</i> CPHX <i>opr8a</i> CPHX <i>opr8,SP</i>	Compare Index Register (H:X) with Memory	(H:X) – (M:M + 0x0001) (CCR Updated But Operands Not Changed)	→	–	–	→	→	→	EXT IMM DIR SP1	3E 65 75 9EF3	hh ll jj kk dd ff	6 3 5 6

Table 8-2. HCS08 Instruction Set Summary (Sheet 4 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
CPX #opr8i CPX opr8a CPX opr16a CPX oprx16,X CPX oprx8,X CPX ,X CPX oprx16,SP CPX oprx8,SP	Compare X (Index Register Low) with Memory	$(X) - (M)$ (CCR Updated But Operands Not Changed)	→	-	-	→	→	→	IMM DIR EXT IX2 IX1 IX SP2 SP1	A3 B3 C3 D3 E3 F3 9ED3 9EE3	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 3 5 4
DAA	Decimal Adjust Accumulator After ADD or ADC of BCD Values	$(A)_{10}$	U	-	-	→	→	→	INH	72		1
DBNZ opr8a,rel DBNZ rel DBNZX rel DBNZ oprx8,X,rel DBNZ ,X,rel DBNZ oprx8,SP,rel	Decrement and Branch if Not Zero	Decrement A, X, or M Branch if (result) $\neq 0$ DBNZX Affects X Not H	-	-	-	-	-	-	DIR INH INH IX1 IX SP1	3B 4B 5B 6B 7B 9E6B	dd rr rr rr ff rr ff ff rr	7 4 4 7 6 8
DEC opr8a DECA DECX DEC oprx8,X DEC ,X DEC oprx8,SP	Decrement	$M \leftarrow (M) - 0x01$ $A \leftarrow (A) - 0x01$ $X \leftarrow (X) - 0x01$ $M \leftarrow (M) - 0x01$ $M \leftarrow (M) - 0x01$ $M \leftarrow (M) - 0x01$	→	-	-	→	→	-	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd ff ff ff ff	5 1 1 5 4 6
DIV	Divide	$A \leftarrow (H:A) \div (X)$ H ← Remainder	-	-	-	-	→	→	INH	52		6
EOR #opr8i EOR opr8a EOR opr16a EOR oprx16,X EOR oprx8,X EOR ,X EOR oprx16,SP EOR oprx8,SP	Exclusive OR Memory with Accumulator	$A \leftarrow (A \oplus M)$	0	-	-	→	→	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	A8 B8 C8 D8 E8 F8 9ED8 9EE8	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 3 5 4
INC opr8a INCA INCX INC oprx8,X INC ,X INC oprx8,SP	Increment	$M \leftarrow (M) + 0x01$ $A \leftarrow (A) + 0x01$ $X \leftarrow (X) + 0x01$ $M \leftarrow (M) + 0x01$ $M \leftarrow (M) + 0x01$ $M \leftarrow (M) + 0x01$	→	-	-	→	→	-	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff ff	5 1 1 5 4 6
JMP opr8a JMP opr16a JMP oprx16,X JMP oprx8,X JMP ,X	Jump	$PC \leftarrow \text{Jump Address}$	-	-	-	-	-	-	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	3 4 4 3 3
JSR opr8a JSR opr16a JSR oprx16,X JSR oprx8,X JSR ,X	Jump to Subroutine	$PC \leftarrow (PC) + n$ ($n = 1, 2, \text{ or } 3$) Push (PCL); $SP \leftarrow (SP) - 0x0001$ Push (PCH); $SP \leftarrow (SP) - 0x0001$ $PC \leftarrow \text{Unconditional Address}$	-	-	-	-	-	-	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff	5 6 6 5 5
LDA #opr8i LDA opr8a LDA opr16a LDA oprx16,X LDA oprx8,X LDA ,X LDA oprx16,SP LDA oprx8,SP	Load Accumulator from Memory	$A \leftarrow (M)$	0	-	-	→	→	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	A6 B6 C6 D6 E6 F6 9ED6 9EE6	ii dd hh ll ee ff ff ff	2 3 4 4 3 3 5 4
LDHX #opr16i LDHX opr8a LDHX opr16a LDHX ,X LDHX oprx16,X LDHX oprx8,X LDHX oprx8,SP	Load Index Register (H:X) from Memory	$H:X \leftarrow (M:M + 0x0001)$	0	-	-	→	→	-	IMM DIR EXT IX IX2 IX1 SP1	45 55 32 9EAE 9EBE 9ECE 9EFE	jj kk dd ll hh ll ee ff ff ff	3 4 5 5 6 5 5

Table 8-2. HCS08 Instruction Set Summary (Sheet 5 of 7)

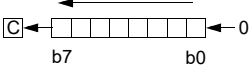
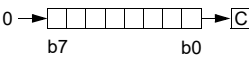
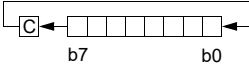
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
LDX #opr8i LDX opr8a LDX opr16a LDX oprx16,X LDX oprx8,X LDX ,X LDX oprx16,SP LDX oprx8,SP	Load X (Index Register Low) from Memory	$X \leftarrow (M)$	0	-	-	-	-	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	AE BE CE DE EE FE 9EDE 9EEE	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
LSL opr8a LSLA LSLX LSL oprx8,X LSL ,X LSL oprx8,SP	Logical Shift Left (Same as ASL)		-	-	-	-	-	-	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	5 1 1 5 4 6
LSR opr8a LSRA LSRX LSR oprx8,X LSR ,X LSR oprx8,SP	Logical Shift Right		-	-	-	0	-	-	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	5 1 1 5 4 6
MOV opr8a,opr8a MOV opr8a,X+ MOV #opr8i,opr8a MOV ,X+,opr8a	Move	$(M)_{\text{destination}} \leftarrow (M)_{\text{source}}$ $H:X \leftarrow (H:X) + 0x0001$ in IX+/DIR and DIR/IX+ Modes	0	-	-	-	-	-	DIR/DIR DIR/IX+ IMM/DIR IX+/DIR	4E 5E 6E 7E	dd dd dd ii ii dd dd	5 5 4 5
MUL	Unsigned multiply	$X:A \leftarrow (X) \times (A)$	-	0	-	-	-	0	INH	42		5
NEG opr8a NEGA NEGX NEG oprx8,X NEG ,X NEG oprx8,SP	Negate (Two's Complement)	$M \leftarrow -(M) = 0x00 - (M)$ $A \leftarrow -(A) = 0x00 - (A)$ $X \leftarrow -(X) = 0x00 - (X)$ $M \leftarrow -(M) = 0x00 - (M)$ $M \leftarrow -(M) = 0x00 - (M)$ $M \leftarrow -(M) = 0x00 - (M)$	-	-	-	-	-	-	DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	5 1 1 5 4 6
NOP	No Operation	Uses 1 Bus Cycle	-	-	-	-	-	-	INH	9D		1
NSA	Nibble Swap Accumulator	$A \leftarrow (A[3:0]:A[7:4])$	-	-	-	-	-	-	INH	62		1
ORA #opr8i ORA opr8a ORA opr16a ORA oprx16,X ORA oprx8,X ORA ,X ORA oprx16,SP ORA oprx8,SP	Inclusive OR Accumulator and Memory	$A \leftarrow (A) (M)$	0	-	-	-	-	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	AA BA CA DA EA FA 9EDA 9EEA	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 3 5 4
PSHA	Push Accumulator onto Stack	Push (A); $SP \leftarrow (SP) - 0x0001$	-	-	-	-	-	-	INH	87		2
PSHH	Push H (Index Register High) onto Stack	Push (H); $SP \leftarrow (SP) - 0x0001$	-	-	-	-	-	-	INH	8B		2
PSHX	Push X (Index Register Low) onto Stack	Push (X); $SP \leftarrow (SP) - 0x0001$	-	-	-	-	-	-	INH	89		2
PULA	Pull Accumulator from Stack	$SP \leftarrow (SP + 0x0001)$; Pull (A)	-	-	-	-	-	-	INH	86		3
PULH	Pull H (Index Register High) from Stack	$SP \leftarrow (SP + 0x0001)$; Pull (H)	-	-	-	-	-	-	INH	8A		3
PULX	Pull X (Index Register Low) from Stack	$SP \leftarrow (SP + 0x0001)$; Pull (X)	-	-	-	-	-	-	INH	88		3
ROL opr8a ROLA ROLX ROL oprx8,X ROL ,X ROL oprx8,SP	Rotate Left through Carry		-	-	-	-	-	-	DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	5 1 1 5 4 6

Table 8-2. HCS08 Instruction Set Summary (Sheet 6 of 7)

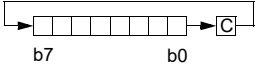
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
ROR <i>opr8a</i> RORA RORX ROR <i>opr8,X</i> ROR <i>,X</i> ROR <i>opr8,SP</i>	Rotate Right through Carry		→	→	→	→	→	→	DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	5 1 1 5 4 6
RSP	Reset Stack Pointer	SP ← 0xFF (High Byte Not Affected)	→	→	→	→	→	→	INH	9C		1
RTI	Return from Interrupt	SP ← (SP) + 0x0001; Pull (CCR) SP ← (SP) + 0x0001; Pull (A) SP ← (SP) + 0x0001; Pull (X) SP ← (SP) + 0x0001; Pull (PCH) SP ← (SP) + 0x0001; Pull (PCL)	→	→	→	→	→	→	INH	80		9
RTS	Return from Subroutine	SP ← SP + 0x0001; Pull (PCH) SP ← SP + 0x0001; Pull (PCL)	→	→	→	→	→	→	INH	81		6
SBC <i>#opr8i</i> SBC <i>opr8a</i> SBC <i>opr16a</i> SBC <i>opr8,X</i> SBC <i>opr16,X</i> SBC <i>,X</i> SBC <i>opr8,SP</i> SBC <i>opr16,SP</i> SBC <i>opr8,SP</i>	Subtract with Carry	A ← (A) – (M) – (C)	→	→	→	→	→	→	IMM DIR EXT IX2 IX1 IX SP2 SP1	A2 B2 C2 D2 E2 F2 9ED2 9EE2	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 3 5 4
SEC	Set Carry Bit	C ← 1	→	→	→	→	→	1	INH	99		1
SEI	Set Interrupt Mask Bit	I ← 1	→	→	1	→	→	→	INH	9B		1
STA <i>opr8a</i> STA <i>opr16a</i> STA <i>opr8,X</i> STA <i>opr16,X</i> STA <i>,X</i> STA <i>opr8,SP</i> STA <i>opr16,SP</i> STA <i>opr8,SP</i>	Store Accumulator in Memory	M ← (A)	0	→	→	→	→	→	DIR EXT IX2 IX1 IX SP2 SP1	B7 C7 D7 E7 F7 9ED7 9EE7	dd hh ll ee ff ff ff ee ff ff	3 4 4 3 2 5 4
STHX <i>opr8a</i> STHX <i>opr16a</i> STHX <i>opr8,SP</i>	Store H:X (Index Reg.)	(M:M + 0x0001) ← (H:X)	0	→	→	→	→	→	DIR EXT SP1	35 96 9EFF	dd hh ll ff	4 5 5
STOP	Enable Interrupts: Stop Processing Refer to MCU Documentation	I bit ← 0; Stop Processing	→	→	0	→	→	→	INH	8E		2+
STX <i>opr8a</i> STX <i>opr16a</i> STX <i>opr8,X</i> STX <i>opr16,X</i> STX <i>,X</i> STX <i>opr8,SP</i> STX <i>opr16,SP</i> STX <i>opr8,SP</i>	Store X (Low 8 Bits of Index Register) in Memory	M ← (X)	0	→	→	→	→	→	DIR EXT IX2 IX1 IX SP2 SP1	BF CF DF EF FF 9EDF 9EEF	dd hh ll ee ff ff ff ee ff ff	3 4 4 3 2 5 4
SUB <i>#opr8i</i> SUB <i>opr8a</i> SUB <i>opr16a</i> SUB <i>opr8,X</i> SUB <i>opr16,X</i> SUB <i>,X</i> SUB <i>opr8,SP</i> SUB <i>opr16,SP</i> SUB <i>opr8,SP</i>	Subtract	A ← (A) – (M)	→	→	→	→	→	→	IMM DIR EXT IX2 IX1 IX SP2 SP1	A0 B0 C0 D0 E0 F0 9ED0 9EE0	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 3 5 4
SWI	Software Interrupt	PC ← (PC) + 0x0001 Push (PCL); SP ← (SP) – 0x0001 Push (PCH); SP ← (SP) – 0x0001 Push (X); SP ← (SP) – 0x0001 Push (A); SP ← (SP) – 0x0001 Push (CCR); SP ← (SP) – 0x0001 I ← 1; PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	→	→	1	→	→	→	INH	83		11

Table 8-2. HCS08 Instruction Set Summary (Sheet 7 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
TAP	Transfer Accumulator to CCR	$CCR \leftarrow (A)$	→	→	→	→	→	→	INH	84		1
TAX	Transfer Accumulator to X (Index Register Low)	$X \leftarrow (A)$	–	–	–	–	–	–	INH	97		1
TPA	Transfer CCR to Accumulator	$A \leftarrow (CCR)$	–	–	–	–	–	–	INH	85		1
TST <i>opr8a</i> TSTA TSTX TST <i>opr8,X</i> TST <i>,X</i> TST <i>opr8,SP</i>	Test for Negative or Zero	(M) – 0x00 (A) – 0x00 (X) – 0x00 (M) – 0x00 (M) – 0x00 (M) – 0x00	0	–	–	→	→	–	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff	4 1 1 4 3 5
TSX	Transfer SP to Index Reg.	$H:X \leftarrow (SP) + 0x0001$	–	–	–	–	–	–	INH	95		2
TXA	Transfer X (Index Reg. Low) to Accumulator	$A \leftarrow (X)$	–	–	–	–	–	–	INH	9F		1
TXS	Transfer Index Reg. to SP	$SP \leftarrow (H:X) - 0x0001$	–	–	–	–	–	–	INH	94		2
WAIT	Enable Interrupts; Wait for Interrupt	$I \text{ bit} \leftarrow 0$; Halt CPU	–	–	0	–	–	–	INH	8F		2+

¹ Bus clock frequency is one-half of the CPU clock frequency.

Table 8-3. Opcode Map (Sheet 1 of 2)

Bit-Manipulation		Branch		Read-Modify-Write						Control		Register/Memory																			
00	5	10	5	20	3	30	5	40	1	50	5	60	5	70	4	80	9	90	3	A0	2	B0	3	C0	4	D0	4	E0	3	F0	3
BRSET0	DIR	BSET0	DIR	BRA	REL	NEG	DIR	NEGA	INH	NEGX	INH	NEG	IX1	NEG	IX	RTI	INH	BGE	REL	SUB	IMM	SUB	DIR	SUB	EXT	SUB	IX2	SUB	IX1	SUB	IX
01	5	11	5	21	3	31	5	41	4	51	4	61	5	71	5	81	6	91	3	A1	2	B1	3	C1	4	D1	4	E1	3	F1	3
BRCLR0	DIR	BCLR0	DIR	BRN	REL	CBEQ	DIR	CBEQA	IMM	CBEQX	IMM	CBEQ	IX1+	CBEQ	IX+	RTS	INH	BLT	REL	CMP	IMM	CMP	DIR	CMP	EXT	CMP	IX2	CMP	IX1	CMP	IX
02	5	12	5	22	3	32	5	42	5	52	6	62	5	72	1	82	5+	92	3	A2	2	B2	3	C2	4	D2	4	E2	3	F2	3
BRSET1	DIR	BSET1	DIR	BHI	REL	LDHX	EXT	MUL	INH	DIV	INH	NSA	INH	DAA	INH	BGND	INH	BGT	REL	SBC	IMM	SBC	DIR	SBC	EXT	SBC	IX2	SBC	IX1	SBC	IX
03	5	13	5	23	3	33	5	43	1	53	1	63	5	73	4	83	11	93	3	A3	2	B3	3	C3	4	D3	4	E3	3	F3	3
BRCLR1	DIR	BCLR1	DIR	BLS	REL	COM	DIR	COMA	INH	COMX	INH	COM	IX1	COM	IX	SWI	INH	BLE	REL	CPX	IMM	CPX	DIR	CPX	EXT	CPX	IX2	CPX	IX1	CPX	IX
04	5	14	5	24	3	34	5	44	1	54	1	64	5	74	4	84	1	94	2	A4	2	B4	3	C4	4	D4	4	E4	3	F4	3
BRSET2	DIR	BSET2	DIR	BCC	REL	LSR	DIR	LSRA	INH	LSRX	INH	LSR	IX1	LSR	IX	TAP	INH	TXS	INH	AND	IMM	AND	DIR	AND	EXT	AND	IX2	AND	IX1	AND	IX
05	5	15	5	25	3	35	4	45	3	55	4	65	3	75	5	85	1	95	2	A5	2	B5	3	C5	4	D5	4	E5	3	F5	3
BRCLR2	DIR	BCLR2	DIR	BCS	REL	STHX	DIR	LDHX	IMM	LDHX	DIR	CPHX	IMM	CPHX	DIR	TPA	INH	TSX	INH	BIT	IMM	BIT	DIR	BIT	EXT	BIT	IX2	BIT	IX1	BIT	IX
06	5	16	5	26	3	36	5	46	1	56	1	66	5	76	4	86	3	96	5	A6	2	B6	3	C6	4	D6	4	E6	3	F6	3
BRSET3	DIR	BSET3	DIR	BNE	REL	ROR	DIR	RORA	INH	RORX	INH	ROR	IX1	ROR	IX	PULA	INH	STHX	EXT	LDA	IMM	LDA	DIR	LDA	EXT	LDA	IX2	LDA	IX1	LDA	IX
07	5	17	5	27	3	37	5	47	1	57	1	67	5	77	4	87	2	97	1	A7	2	B7	3	C7	4	D7	4	E7	3	F7	2
BRCLR3	DIR	BCLR3	DIR	BEQ	REL	ASR	DIR	ASRA	INH	ASRX	INH	ASR	IX1	ASR	IX	PSHA	INH	TAX	INH	AIS	IMM	STA	DIR	STA	EXT	STA	IX2	STA	IX1	STA	IX
08	5	18	5	28	3	38	5	48	1	58	1	68	5	78	4	88	3	98	1	A8	2	B8	3	C8	4	D8	4	E8	3	F8	3
BRSET4	DIR	BSET4	DIR	BHCC	REL	LSL	DIR	LSLA	INH	LSLX	INH	LSL	IX1	LSL	IX	PULX	INH	CLC	INH	EOR	IMM	EOR	DIR	EOR	EXT	EOR	IX2	EOR	IX1	EOR	IX
09	5	19	5	29	3	39	5	49	1	59	1	69	5	79	4	89	2	99	1	A9	2	B9	3	C9	4	D9	4	E9	3	F9	3
BRCLR4	DIR	BCLR4	DIR	BHCS	REL	ROL	DIR	ROLA	INH	ROLX	INH	ROL	IX1	ROL	IX	PSHX	INH	SEC	INH	ADC	IMM	ADC	DIR	ADC	EXT	ADC	IX2	ADC	IX1	ADC	IX
0A	5	1A	5	2A	3	3A	5	4A	1	5A	1	6A	5	7A	4	8A	3	9A	1	AA	2	BA	3	CA	4	DA	4	EA	3	FA	3
BRSET5	DIR	BSET5	DIR	BPL	REL	DEC	DIR	DECA	INH	DECX	INH	DEC	IX1	DEC	IX	PULH	INH	CLI	INH	ORA	IMM	ORA	DIR	ORA	EXT	ORA	IX2	ORA	IX1	ORA	IX
0B	5	1B	5	2B	3	3B	7	4B	4	5B	4	6B	7	7B	6	8B	2	9B	1	AB	2	BB	3	CB	4	DB	4	EB	3	FB	3
BRCLR5	DIR	BCLR5	DIR	BMI	REL	DBNZ	DIR	DBNZA	INH	DBNZX	INH	DBNZ	IX1	DBNZ	IX	PSHH	INH	SEI	INH	ADD	IMM	ADD	DIR	ADD	EXT	ADD	IX2	ADD	IX1	ADD	IX
0C	5	1C	5	2C	3	3C	5	4C	1	5C	1	6C	5	7C	4	8C	1	9C	1			BC	3	CC	4	DC	4	EC	3	FC	3
BRSET6	DIR	BSET6	DIR	BMC	REL	INC	DIR	INCA	INH	INCX	INH	INC	IX1	INC	IX	CLRH	INH	RSP	INH			JMP	DIR	JMP	EXT	JMP	IX2	JMP	IX1	JMP	IX
0D	5	1D	5	2D	3	3D	4	4D	1	5D	1	6D	4	7D	3			9D	1	AD	5	BD	5	CD	6	DD	6	ED	5	FD	5
BRCLR6	DIR	BCLR6	DIR	BMS	REL	TST	DIR	TSTA	INH	TSTX	INH	TST	IX1	TST	IX			NOP	INH	BSR	REL	JSR	DIR	JSR	EXT	JSR	IX2	JSR	IX1	JSR	IX
0E	5	1E	5	2E	3	3E	6	4E	5	5E	5	6E	4	7E	5	8E	2+	9E	Page 2	AE	2	BE	3	CE	4	DE	4	EE	3	FE	3
BRSET7	DIR	BSET7	DIR	BIL	REL	CPHX	EXT	MOV	DD	MOV	DIX+	MOV	IMD	MOV	IX+D	STOP	INH			LDX	IMM	LDX	DIR	LDX	EXT	LDX	IX2	LDX	IX1	LDX	IX
0F	5	1F	5	2F	3	3F	5	4F	1	5F	1	6F	5	7F	4	8F	2+	9F	1	AF	2	BF	3	CF	4	DF	4	EF	3	FF	2
BRCLR7	DIR	BCLR7	DIR	BIH	REL	CLR	DIR	CLRA	INH	CLR	INH	CLR	IX1	CLR	IX	WAIT	INH	TXA	INH	AIX	IMM	STX	DIR	STX	EXT	STX	IX2	STX	IX1	STX	IX

INH Inherent
 IMM Immediate
 DIR Direct
 EXT Extended
 DD DIR to DIR
 IX+D IX+ to DIR
 REL Relative
 IX Indexed, No Offset
 IX1 Indexed, 8-Bit Offset
 IX2 Indexed, 16-Bit Offset
 IMM to DIR
 DIR to IX+
 SP1 Stack Pointer, 8-Bit Offset
 SP2 Stack Pointer, 16-Bit Offset
 IX+ Indexed, No Offset with Post Increment
 IX1+ Indexed, 1-Byte Offset with Post Increment

Opcode in Hexadecimal F0 SUB 3
 Number of Bytes 1 SUB IX
 HCS08 Cycles Instruction Mnemonic Addressing Mode

Table 8-3. Opcode Map (Sheet 2 of 2)

Bit-Manipulation	Branch	Read-Modify-Write				Control				Register/Memory							
					9E60 6 NEG 3 SP1						9ED0 5 SUB 4 SP2	9EE0 4 SUB 3 SP1					
					9E61 6 CBEQ 4 SP1						9ED1 5 CMP 4 SP2	9EE1 4 CMP 3 SP1					
											9ED2 5 SBC 4 SP2	9EE2 4 SBC 3 SP1					
					9E63 6 COM 3 SP1						9ED3 5 CPX 4 SP2	9EE3 4 CPX 3 SP1	9EF3 6 CPHX 3 SP1				
					9E64 6 LSR 3 SP1						9ED4 5 AND 4 SP2	9EE4 4 AND 3 SP1					
											9ED5 5 BIT 4 SP2	9EE5 4 BIT 3 SP1					
					9E66 6 ROR 3 SP1						9ED6 5 LDA 4 SP2	9EE6 4 LDA 3 SP1					
					9E67 6 ASR 3 SP1						9ED7 5 STA 4 SP2	9EE7 4 STA 3 SP1					
					9E68 6 LSL 3 SP1						9ED8 5 EOR 4 SP2	9EE8 4 EOR 3 SP1					
					9E69 6 ROL 3 SP1						9ED9 5 ADC 4 SP2	9EE9 4 ADC 3 SP1					
					9E6A 6 DEC 3 SP1						9EDA 5 ORA 4 SP2	9EEA 4 ORA 3 SP1					
					9E6B 8 DBNZ 4 SP1						9EDB 5 ADD 4 SP2	9EEB 4 ADD 3 SP1					
					9E6C 6 INC 3 SP1												
					9E6D 5 TST 3 SP1												
										9EAE 5 LDHX 2 IX	9EBE 6 LDHX 4 IX2	9ECE 5 LDHX 3 IX1	9EDE 5 LDX 4 SP2	9EEE 4 LDX 3 SP1	9EFE 5 LDHX 3 SP1		
					9E6F 6 CLR 3 SP1						9EDF 5 STX 4 SP2	9EEF 4 STX 3 SP1	9EFF 5 STHX 3 SP1				

INH Inherent REL Relative SP1 Stack Pointer, 8-Bit Offset
 IMM Immediate IX Indexed, No Offset SP2 Stack Pointer, 16-Bit Offset
 DIR Direct IX1 Indexed, 8-Bit Offset IX+ Indexed, No Offset with
 EXT Extended IX2 Indexed, 16-Bit Offset Post Increment
 DD DIR to DIR IMD IMM to DIR IX1+ Indexed, 1-Byte Offset with
 IX+D IX+ to DIR DIX+ DIR to IX+ Post Increment

Note: All Sheet 2 Opcodes are Preceded by the Page 2 Prebyte (9E)

Prebyte (9E) and Opcode in
 Hexadecimal

9E60 6	NEG
3 SP1	

 HCS08 Cycles
 Instruction Mnemonic
 Addressing Mode
 Number of Bytes

Chapter 9

Analog Comparator (S08ACMPVLPV1)

9.1 Introduction

MC9S08QL8 series MCUs have one analog comparator, named ACMP.

The analog comparator module (ACMP) provides a circuit for comparing two analog input voltages or for comparing one analog input voltage to an internal reference voltage. The comparator circuit is designed to operate across the full range of the supply voltage (rail-to-rail operation).

The ACMP option to compare to internal bandgap reference is not allowed in LPRUN and LPWAIT.

9.1.1 ACMP Configuration Information

When using the bandgap reference voltage for input to ACMP+, the user must enable the bandgap buffer by setting BGBE = 1 in SPMSC1 see [Section 5.8.7, System Power Management Status and Control 1 Register \(SPMSC1\)](#) for value of bandgap voltage reference see the data sheet.

BKGDPE bit in SOPT1 register is set following any reset of the MCU and must be cleared to use the PTA4/ACMPO/BKGD/MS pin's alternative pin functions.

9.1.2 ACMP/TPM Configuration Information

The ACMP module can be configured to connect the output of the analog comparator to the TPM input capture channel 0 by setting the ACIC bit in SOPT2. With ACIC set, the TPMCH0 pin is not available externally regardless of the configuration of the TPM module.

9.1.3 ACMP Clock Gating

The bus clock to the ACMP can be gated on and off using the ACMP bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the ACMP bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, Peripheral Clock Gating](#) for details.

9.1.4 Stop1 Not Available

Stop1 is not available on this MCU. Therefore, ignore references the stop1 within this chapter.

9.1.5 Features

The ACMP has the following features:

- Full rail-to-rail supply operation
- Low input offset and hysteresis
- Selectable interrupt on rising edge, falling edge, or rising and falling edges of comparator output
- Option to compare to fixed internal bandgap reference voltage

9.1.6 Modes of Operation

9.1.6.1 Wait Mode Operation

During wait mode the ACMP, if enabled, continues to operate normally. Also, if enabled, the interrupt can wake the MCU.

9.1.6.2 Stop3 Mode Operation

If enabled, the ACMP continues to operate in stop3 mode and compare operation remains active. If ACOPE is enabled, comparator output operates in the normal operating mode and comparator output is placed onto the external pin. The MCU is brought out of stop when a compare event occurs and ACIE is enabled; ACF flag sets accordingly.

If stop is exited with a reset, the ACMP will be put into its reset state.

9.1.6.3 Stop2 Mode Operation

During stop2 mode, the ACMP module is fully powered down. Upon wakeup from stop2 mode, the ACMP module is in the reset state.

9.1.6.4 Active Background Mode Operation

When the microcontroller is in active background mode, the ACMP continues to operate normally. When ACMPO is shared with the BKGD pin and the BKGD pin is enabled, the ACMPO function is not available.

9.1.7 Block Diagram

The block diagram for the ACMP module follows.

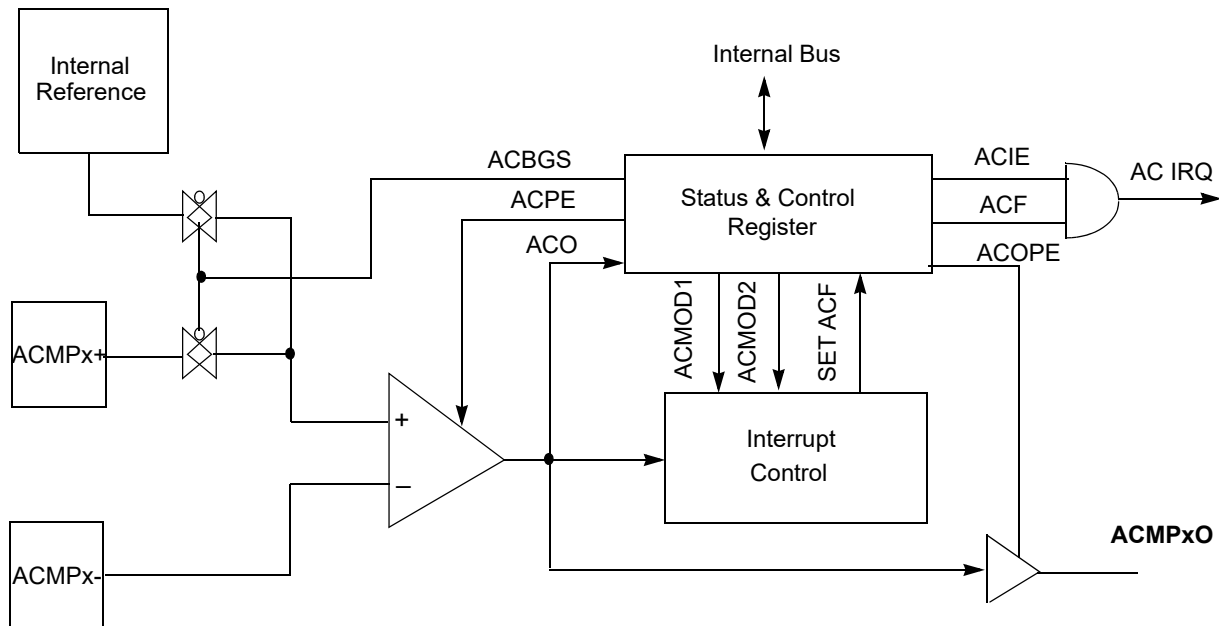


Figure 9-1. Analog Comparator Module Block Diagram

9.2 External Signal Description

The ACMP has two analog input pins: ACMP- and ACMP+. Each of these pins can accept an input voltage that varies across the full operating voltage range of the MCU. If the module is not enabled, each of these pins can be used as digital inputs or outputs. Consult the specific MCU documentation to determine what functions are shared with these analog inputs. As shown in the block diagram, the ACMP+ pin is connected to the comparator non-inverting input if ACBGS is equal to logic zero, and the ACMP- pin is connected to the inverting input of the comparator.

9.3 Register Definition

9.3.1 Status and Control Register (ACMPxSC)

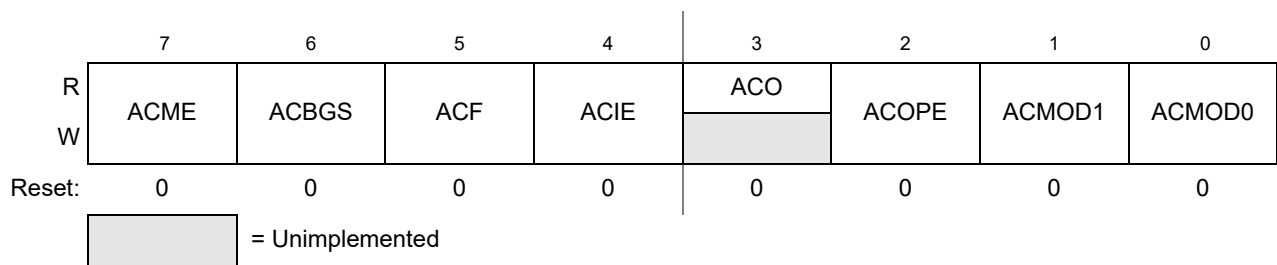


Figure 9-2. ACMP Status and Control Register (ACMPxSC)

Table 9-1. ACMPxSC Field Descriptions

Field	Description
7 ACME	Analog Comparator Module Enable — The ACME bit enables the ACMP module. When the module is not enabled, it remains in a low power state. 0 Analog Comparator disabled. 1 Analog Comparator enabled.
6 ACBGS	Analog Comparator Bandgap Select — The ACBGS bit selects the internal bandgap as the comparator reference. 0 External pin ACMP+ selected as comparator non-inverting input. 1 Internal bandgap reference selected as comparator non-inverting input.
5 ACF	Analog Comparator Flag — The ACF bit is set when a compare event occurs. Compare events are defined by the ACMOD0 and ACMOD1 bits. The ACF bit is cleared by writing a logic one to the bit. 0 Compare event has not occurred. 1 Compare event has occurred.
4 ACIE	Analog Comparator Interrupt Enable — The ACIE bit enables the interrupt from the ACM. When this bit is set, an interrupt is asserted when the ACF bit is set. 0 Interrupt disabled. 1 Interrupt enabled.
3 ACO	Analog Comparator Output — Reading the ACO bit returns the current value of the analog comparator output. The register bit is reset to zero and reads as logic zero when the ACMP module is disabled (ACME = 0).
2 ACOPE	Analog Comparator Output Pin Enable — ACOPE enables the comparator output to be placed onto the external pin, ACMPxO. 0 Analog comparator output not available on ACMPxO. 1 Analog comparator output is driven out on ACMPxO.
1:0 ACMOD	Analog Comparator Modes — The ACMOD1 and ACMOD0 bits select the flag setting mode that controls the type of compare event that sets the ACF bit. 00 Comparator output falling edge. 01 Comparator output rising edge. 10 Comparator output falling edge. 11 Comparator output rising and falling edge.

9.4 Functional Description

The ACMP module can be used to compare:

- Two analog input voltages applied to ACMP- and ACMP+ or
- An analog input voltage applied to ACMP- with an internal bandgap reference voltage

The ACBGS bit selects the mode of operation. The comparator output is high when the non-inverting input is greater than the inverting input, and low when the non-inverting input is less than the inverting input. The ACMOD0 and ACMOD1 bits select the condition that causes the ACF bit to be set. The ACF bit can be set on a rising edge of the comparator output, a falling edge of the comparator output, or a rising and a falling edge (toggle). The comparator output can be read directly through the ACO bit.

9.5 Interrupts

The ACMP module is capable of generating an interrupt on a compare event. The interrupt request is asserted when both the ACIE bit and the ACF bit are set. The interrupt is deasserted by clearing either the

ACIE bit or the ACF bit. The ACIE bit is cleared by writing a logic zero and the ACF bit is cleared by writing a logic one.

Chapter 10

Analog-to-Digital Converter (S08ADC12V1)

10.1 Introduction

The 12-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

10.1.1 ADC Clock Gating

The bus clock to the ADC can be gated on and off using the ADC bit in SCGC1. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the ADC bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, Peripheral Clock Gating](#) for details.

10.1.2 Module Configurations

This section provides device-specific information for configuring the ADC on MC9S08QL8 series.

10.1.2.1 Analog Supply and Voltage Reference Connections

The MC9S08QL8 series V_{DDA}/V_{REFH} and V_{SSA}/V_{REFL} pins are double bonded to V_{DD} and V_{SS} , in the 20-pin and 16-pin packages.

10.1.2.2 Configurations for Stop and Low Power Modes

The ADC, if enabled, must be configured to use the asynchronous clock source, ADACK, to meet the ADC minimum frequency requirements. To convert the internal bandgap channel (AD27), the LVD must be enabled in stop mode (LVDE and LVDSE bits set in the SPMSC1 register). The bandgap channel cannot be converted in low power run or low power wait modes.

10.1.2.3 Channel Assignments

The ADC channel assignments for the MC9S08QL8 series devices are shown in [Table 10-1](#). Reserved channels convert to an unknown value.

Table 10-1. ADC Channel Assignment

ADCH	Channel	Input	Pin Control	ADCH	Channel	Input	Pin Control
00000	AD0	PTA0/ADP0	ADPC0	10000	AD16	V_{REFL}	N/A
00001	AD1	PTA1/ADP1	ADPC1	10001	AD17	V_{REFL}	N/A
00010	AD2	PTA2/ADP2	ADPC2	10010	AD18	V_{REFL}	N/A

Table 10-1. ADC Channel Assignment (continued)

ADCH	Channel	Input	Pin Control	ADCH	Channel	Input	Pin Control
00011	AD3	PTA3/ADP3	ADPC3	10011	AD19	V _{REFL}	N/A
00100	AD4	PTB0/ADP4	ADPC4	10100	AD20	V _{REFL}	N/A
00101	AD5	PTB1/ADP5	ADPC5	10101	AD21	V _{REFL}	N/A
00110	AD6	PTB2/ADP6	ADPC6	10110	AD22	V _{REFL}	N/A
00111	AD7	PTB3/ADP7	ADPC7	10111	AD23	Reserved	N/A
01000	AD8	Reserved	N/A	11000	AD24	Reserved	N/A
01001	AD9	Reserved	N/A	11001	AD25	Reserved	N/A
01010	AD10	Reserved	N/A	11010	AD26	Temperature Sensor ¹	N/A
01011	AD11	Reserved	N/A	11011	AD27	Internal bandgap	N/A
01100	AD12	V _{REFL}	N/A	11100	—	Reserved	N/A
01101	AD13	V _{REFL}	N/A	11101	V _{REFH} ²	V _{REFH}	N/A
01110	AD14	V _{REFL}	N/A	11110	V _{REFL} ³	V _{SS}	N/A
01111	AD15	V _{REFL}	N/A	11111	Module Disabled	None	N/A

¹ For information, see [Section 10.1.2.6, Temperature Sensor](#).

² V_{REFH}/V_{DDA} is tied to V_{DD}.

³ V_{REFL}/V_{SSA} is tied to V_{SS}.

NOTE

Selecting the internal bandgap channel requires BGBE = 1 in SPMSC1, see [Section 5.8.7, System Power Management Status and Control 1 Register \(SPMSC1\)](#) for value of bandgap voltage reference, see the data sheet.

10.1.2.4 Alternate Clock

The ADC is capable of performing conversions using the MCU bus clock, the bus clock divided by two, the local asynchronous clock (ADACK) within the module, or the alternate clock (ALTCLK). The ALTCLK on the MC9S08QL8 series is the ICSERCLK. See [Chapter 11, Internal Clock Source \(S08ICSV3\)](#) for more information.

10.1.2.5 Hardware Trigger

The RTC can be enabled as a hardware trigger for the ADC module by setting the ADTRG bit in the ADCSC2 register. When enabled, the ADC will be triggered every time RTCINT matches RTCMOD. The RTC interrupt does not have to be enabled to trigger the ADC.

The RTC can be clocked by either ICSIRCLK, OSCOUT or LPO. The period of the RTC is determined by the input clock frequency and the RTC configuration bits. When the ADC hardware trigger is enabled, a conversion is initiated upon a RTC overflow.

The RTC can be configured to cause a hardware trigger in MCU run, wait, low power run, low power wait, and stop3.

10.1.2.6 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. [Equation 10-1](#) provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - ((V_{\text{TEMP}} - V_{\text{TEMP25}}) \div m) \quad \text{Eqn. 10-1}$$

where:

- V_{TEMP} is the voltage of the temperature sensor channel at the ambient temperature.
- V_{TEMP25} is the voltage of the temperature sensor channel at 25°C.
- m is the hot or cold voltage versus temperature slope in V/°C.

For temperature calculations, use the V_{TEMP25} and m values in the data sheet.

In application code, the user reads the temperature sensor channel, calculates V_{TEMP} , and compares it to V_{TEMP25} . If V_{TEMP} is greater than V_{TEMP25} the cold slope value is applied in [Equation 10-1](#). If V_{TEMP} is less than V_{TEMP25} the hot slope value is applied in [Equation 10-1](#).

10.1.3 Features

Features of the ADC module include:

- Linear successive approximation algorithm with 12-bit resolution
- Up to 28 analog inputs
- Output formatted in 12-, 10-, or 8-bit right-justified unsigned format
- Single or continuous conversion (automatic return to idle after single conversion)
- Configurable sample time and conversion speed/power
- Conversion complete flag and interrupt
- Input clock selectable from up to four sources
- Operation in wait or stop3 modes for lower noise operation
- Asynchronous clock source for lower noise operation
- Selectable asynchronous hardware conversion trigger
- Automatic compare with interrupt for less-than, or greater-than or equal-to, programmable value
- Temperature sensor

10.1.4 ADC Module Block Diagram

Figure 10-1 provides a block diagram of the ADC module.

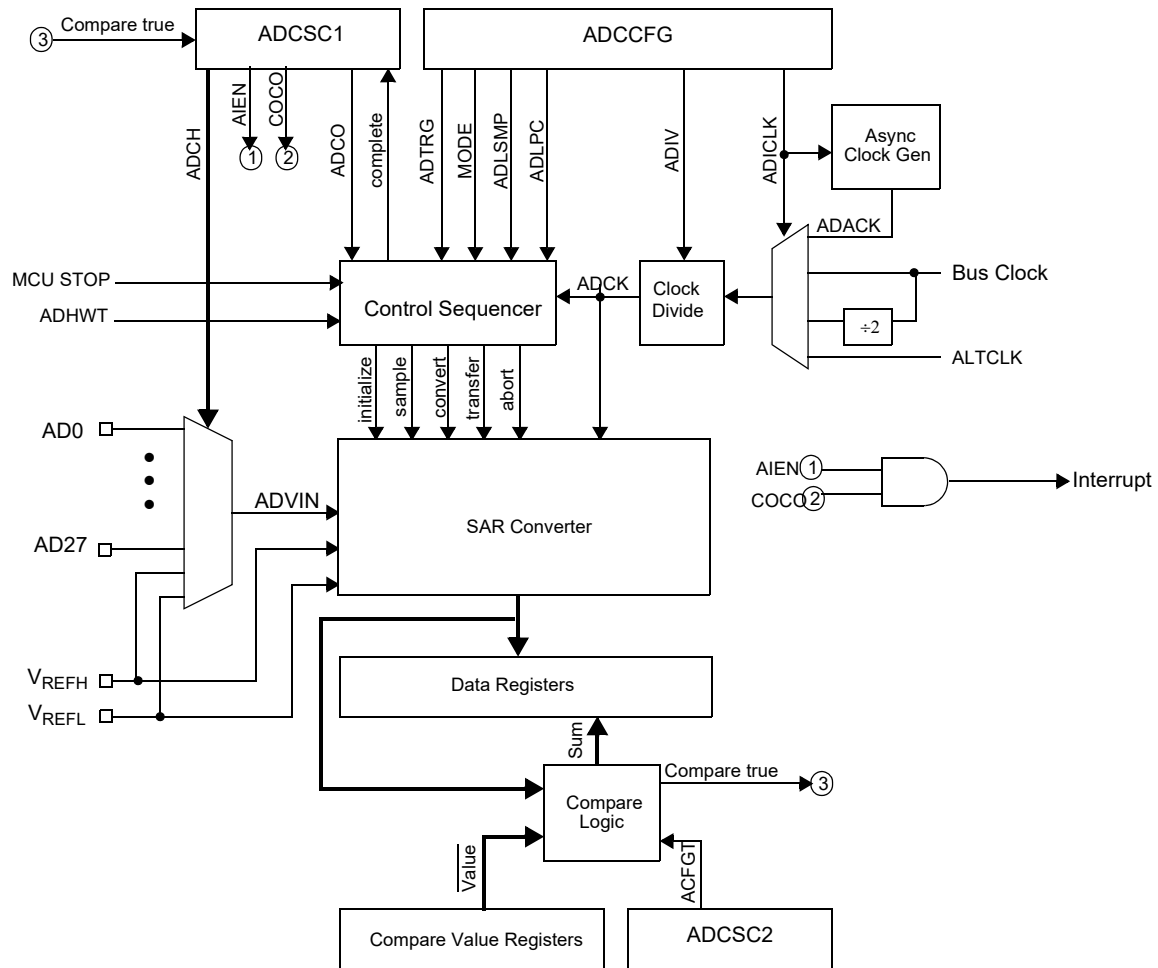


Figure 10-1. ADC Block Diagram

10.2 External Signal Description

The ADC module supports up to 28 separate analog inputs. It also requires four supply/reference/ground connections.

Table 10-2. Signal Properties

Name	Function
AD27–AD0	Analog Channel inputs
V _{REFH}	High reference voltage
V _{REFL}	Low reference voltage
V _{DDA}	Analog power supply
V _{SSA}	Analog ground

10.2.1 Analog Power (V_{DDA})

The ADC analog portion uses V_{DDA} as its power connection. In some packages, V_{DDA} is connected internally to V_{DD} . If externally available, connect the V_{DDA} pin to the same voltage potential as V_{DD} . External filtering may be necessary to ensure clean V_{DDA} for good results.

10.2.2 Analog Ground (V_{SSA})

The ADC analog portion uses V_{SSA} as its ground connection. In some packages, V_{SSA} is connected internally to V_{SS} . If externally available, connect the V_{SSA} pin to the same voltage potential as V_{SS} .

10.2.3 Voltage Reference High (V_{REFH})

V_{REFH} is the high reference voltage for the converter. In some packages, V_{REFH} is connected internally to V_{DDA} . If externally available, V_{REFH} may be connected to the same potential as V_{DDA} or may be driven by an external source between the minimum V_{DDA} spec and the V_{DDA} potential (V_{REFH} must never exceed V_{DDA}).

10.2.4 Voltage Reference Low (V_{REFL})

V_{REFL} is the low-reference voltage for the converter. In some packages, V_{REFL} is connected internally to V_{SSA} . If externally available, connect the V_{REFL} pin to the same voltage potential as V_{SSA} .

10.2.5 Analog Channel Inputs (ADx)

The ADC module supports up to 28 separate analog inputs. An input is selected for conversion through the ADCH channel select bits.

10.3 Register Definition

These memory-mapped registers control and monitor operation of the ADC:

- Status and control register, ADCSC1
- Status and control register, ADCSC2
- Data result registers, ADCRH and ADCRL
- Compare value registers, ADCCVH and ADCCVL
- Configuration register, ADCCFG
- Pin control registers, APCTL1, APCTL2, APCTL3

10.3.1 Status and Control Register 1 (ADCSC1)

This section describes the function of the ADC status and control register (ADCSC1). Writing ADCSC1 aborts the current conversion and initiates a new conversion (if the ADCH bits are equal to a value other than all 1s).

**Figure 10-2. Status and Control Register (ADCSC1)****Table 10-3. ADCSC1 Field Descriptions**

Field	Description
7 COCO	Conversion Complete Flag. The COCO flag is a read-only bit set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1), the COCO flag is set upon completion of a conversion only if the compare result is true. This bit is cleared when ADCSC1 is written or when ADCRL is read. 0 Conversion not completed 1 Conversion completed
6 AIEN	Interrupt Enable AIEN enables conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted. 0 Conversion complete interrupt disabled 1 Conversion complete interrupt enabled
5 ADCO	Continuous Conversion Enable. ADCO enables continuous conversions. 0 One conversion following a write to the ADCSC1 when software triggered operation is selected, or one conversion following assertion of ADHWT when hardware triggered operation is selected. 1 Continuous conversions initiated following a write to ADCSC1 when software triggered operation is selected. Continuous conversions are initiated by an ADHWT event when hardware triggered operation is selected.
4:0 ADCH	Input Channel Select. The ADCH bits form a 5-bit field that selects one of the input channels. The input channels are detailed in Table 10-4 . The successive approximation converter subsystem is turned off when the channel select bits are all set. This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional, single conversion from being performed. It is not necessary to set the channel select bits to all ones to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.

Table 10-4. Input Channel Select

ADCH	Input Select
00000–01111	AD0–15
10000–11011	AD16–27
11100	Reserved
11101	V _{REFH}
11110	V _{REFL}
11111	Module disabled

10.3.2 Status and Control Register 2 (ADCSC2)

The ADCSC2 register controls the compare function, conversion trigger, and conversion active of the ADC module.

	7	6	5	4	3	2	1	0
R	ADACT	ADTRG	ACFE	ACFGT	0	0	R ¹	R ¹
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-3. Status and Control Register 2 (ADCSC2)

¹ Bits 1 and 0 are reserved bits that must always be written to 0.

Table 10-5. ADCSC2 Register Field Descriptions

Field	Description
7 ADACT	Conversion Active. Indicates that a conversion is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress 1 Conversion in progress
6 ADTRG	Conversion Trigger Select. Selects the type of trigger used for initiating a conversion. Two types of triggers are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write to ADCSC1. When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input. 0 Software trigger selected 1 Hardware trigger selected
5 ACFE	Compare Function Enable. Enables the compare function. 0 Compare function disabled 1 Compare function enabled
4 ACFGT	Compare Function Greater Than Enable. Configures the compare function to trigger when the result of the conversion of the input being monitored is greater than or equal to the compare value. The compare function defaults to triggering when the result of the compare of the input being monitored is less than the compare value. 0 Compare triggers when input is less than compare value 1 Compare triggers when input is greater than or equal to compare value

10.3.3 Data Result High Register (ADCRH)

In 12-bit operation, ADCRH contains the upper four bits of 12-bit conversion data. In 10-bit operation, ADCRH contains the upper two bits of 10-bit conversion data. In 12-bit and 10-bit mode, ADCRH is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met. When configured for 10-bit mode, ADR[11:10] are cleared. When configured for 8-bit mode, ADR[11:8] are cleared.

When automatic compare is not enabled, the value stored in ADCRH are the upper bits of the conversion result. When automatic compare is enabled, the conversion result is manipulated as described in [Section 10.4.5, Automatic Compare Function](#) prior to storage in ADCRH:ADCRL registers.

In 12-bit and 10-bit mode, reading ADCRH prevents the ADC from transferring subsequent conversion data into the result registers until ADCRL is read. If ADCRL is not read until after the next conversion is completed, the intermediate conversion data is lost. In 8-bit mode, there is no interlocking with ADCRL. If the MODE bits are changed, any data in ADCRH becomes invalid.

	7	6	5	4	3	2	1	0
R	0	0	0	0	ADR11	ADR10	ADR9	ADR8
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-4. Data Result High Register (ADCRH)

10.3.4 Data Result Low Register (ADCRL)

ADCRL contains the lower eight bits of a 12-bit or 10-bit conversion data, and all eight bits of 8-bit conversion data. ADCRL is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met.

When automatic compare is not enabled, the value stored in ADCRL is the lower eight bits of the conversion result. When automatic compare is enabled, the conversion result is manipulated as described in [Section 10.4.5, Automatic Compare Function](#) prior to storage in ADCRH:ADCRL registers.

In 12-bit and 10-bit mode, reading ADCRH prevents the ADC from transferring subsequent conversion data into the result registers until ADCRL is read. If ADCRL is not read until the after next conversion is completed, the intermediate conversion data is lost. In 8-bit mode, there is no interlocking with ADCRH. If the MODE bits are changed, any data in ADCRL becomes invalid.

	7	6	5	4	3	2	1	0
R	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-5. Data Result Low Register (ADCRL)

10.3.5 Compare Value High Register (ADCCVH)

In 12-bit mode, the ADCCVH register holds the upper four bits of the 12-bit compare value. When the compare function is enabled, these bits are compared to the upper four bits of the result following a conversion in 12-bit mode.

	7	6	5	4	3	2	1	0
R	0	0	0	0	ADCV11	ADCV10	ADCV9	ADCV8
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-6. Compare Value High Register (ADCCVH)

In 10-bit mode, the ADCCVH register holds the upper two bits of the 10-bit compare value (ADCV[9:8]). These bits are compared to the upper two bits of the result following a conversion in 10-bit mode when the compare function is enabled.

In 8-bit mode, ADCCVH is not used during compare.

10.3.6 Compare Value Low Register (ADCCVL)

This register holds the lower eight bits of the 12-bit or 10-bit compare value or all eight bits of the 8-bit compare value. When the compare function is enabled, bits ADCV[7:0] are compared to the lower eight bits of the result following a conversion in 12-bit, 10-bit or 8-bit mode.

	7	6	5	4	3	2	1	0
R	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-7. Compare Value Low Register(ADCCVL)

10.3.7 Configuration Register (ADCCFG)

ADCCFG selects the mode of operation, clock source, clock divide, and configures for low power and long sample time.

	7	6	5	4	3	2	1	0
R	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-8. Configuration Register (ADCCFG)

Table 10-6. ADCCFG Register Field Descriptions

Field	Description
7 ADLPC	Low-Power Configuration. ADLPC controls the speed and power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required. 0 High speed configuration 1 Low power configuration: The power is reduced at the expense of maximum clock speed.
6:5 ADIV	Clock Divide Select. ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK. Table 10-7 shows the available clock configurations.
4 ADLSMP	Long Sample Time Configuration. ADLSMP selects between long and short sample time. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. 0 Short sample time 1 Long sample time
3:2 MODE	Conversion Mode Selection. MODE bits are used to select between 12-, 10-, or 8-bit operation. See Table 10-8 .
1:0 ADICLK	Input Clock Select. ADICLK bits select the input clock source to generate the internal clock ADCK. See Table 10-9 .

Table 10-7. Clock Divide Select

ADIV	Divide Ratio	Clock Rate
00	1	Input clock
01	2	Input clock ÷ 2
10	4	Input clock ÷ 4
11	8	Input clock ÷ 8

Table 10-8. Conversion Modes

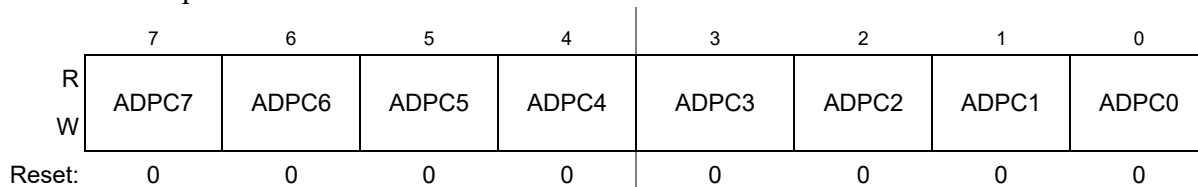
MODE	Mode Description
00	8-bit conversion (N=8)
01	12-bit conversion (N=12)
10	10-bit conversion (N=10)
11	Reserved

Table 10-9. Input Clock Select

ADICLK	Selected Clock Source
00	Bus clock
01	Bus clock divided by 2
10	Alternate clock (ALTCLK)
11	Asynchronous clock (ADACK)

10.3.8 Pin Control 1 Register (APCTL1)

The pin control registers disable the I/O port control of MCU pins used as analog inputs. APCTL1 is used to control the pins associated with channels 0–7 of the ADC module.

**Figure 10-9. Pin Control 1 Register (APCTL1)****Table 10-10. APCTL1 Register Field Descriptions**

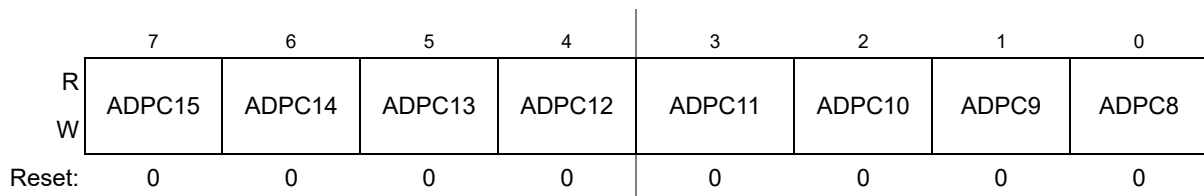
Field	Description
7 ADPC7	ADC Pin Control 7. ADPC7 controls the pin associated with channel AD7. 0 AD7 pin I/O control enabled 1 AD7 pin I/O control disabled
6 ADPC6	ADC Pin Control 6. ADPC6 controls the pin associated with channel AD6. 0 AD6 pin I/O control enabled 1 AD6 pin I/O control disabled
5 ADPC5	ADC Pin Control 5. ADPC5 controls the pin associated with channel AD5. 0 AD5 pin I/O control enabled 1 AD5 pin I/O control disabled
4 ADPC4	ADC Pin Control 4. ADPC4 controls the pin associated with channel AD4. 0 AD4 pin I/O control enabled 1 AD4 pin I/O control disabled
3 ADPC3	ADC Pin Control 3. ADPC3 controls the pin associated with channel AD3. 0 AD3 pin I/O control enabled 1 AD3 pin I/O control disabled
2 ADPC2	ADC Pin Control 2. ADPC2 controls the pin associated with channel AD2. 0 AD2 pin I/O control enabled 1 AD2 pin I/O control disabled

Table 10-10. APCTL1 Register Field Descriptions (continued)

Field	Description
1 ADPC1	ADC Pin Control 1. ADPC1 controls the pin associated with channel AD1. 0 AD1 pin I/O control enabled 1 AD1 pin I/O control disabled
0 ADPC0	ADC Pin Control 0. ADPC0 controls the pin associated with channel AD0. 0 AD0 pin I/O control enabled 1 AD0 pin I/O control disabled

10.3.9 Pin Control 2 Register (APCTL2)

APCTL2 controls channels 8–15 of the ADC module.

**Figure 10-10. Pin Control 2 Register (APCTL2)****Table 10-11. APCTL2 Register Field Descriptions**

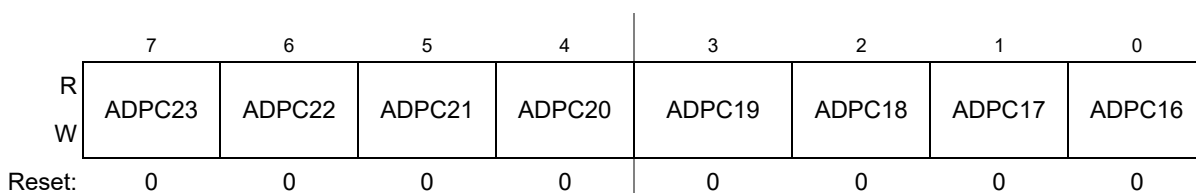
Field	Description
7 ADPC15	ADC Pin Control 15. ADPC15 controls the pin associated with channel AD15. 0 AD15 pin I/O control enabled 1 AD15 pin I/O control disabled
6 ADPC14	ADC Pin Control 14. ADPC14 controls the pin associated with channel AD14. 0 AD14 pin I/O control enabled 1 AD14 pin I/O control disabled
5 ADPC13	ADC Pin Control 13. ADPC13 controls the pin associated with channel AD13. 0 AD13 pin I/O control enabled 1 AD13 pin I/O control disabled
4 ADPC12	ADC Pin Control 12. ADPC12 controls the pin associated with channel AD12. 0 AD12 pin I/O control enabled 1 AD12 pin I/O control disabled
3 ADPC11	ADC Pin Control 11. ADPC11 controls the pin associated with channel AD11. 0 AD11 pin I/O control enabled 1 AD11 pin I/O control disabled
2 ADPC10	ADC Pin Control 10. ADPC10 controls the pin associated with channel AD10. 0 AD10 pin I/O control enabled 1 AD10 pin I/O control disabled

Table 10-11. APCTL2 Register Field Descriptions (continued)

Field	Description
1 ADPC9	ADC Pin Control 9. ADPC9 controls the pin associated with channel AD9. 0 AD9 pin I/O control enabled 1 AD9 pin I/O control disabled
0 ADPC8	ADC Pin Control 8. ADPC8 controls the pin associated with channel AD8. 0 AD8 pin I/O control enabled 1 AD8 pin I/O control disabled

10.3.10 Pin Control 3 Register (APCTL3)

APCTL3 controls channels 16–23 of the ADC module.

**Figure 10-11. Pin Control 3 Register (APCTL3)****Table 10-12. APCTL3 Register Field Descriptions**

Field	Description
7 ADPC23	ADC Pin Control 23. ADPC23 controls the pin associated with channel AD23. 0 AD23 pin I/O control enabled 1 AD23 pin I/O control disabled
6 ADPC22	ADC Pin Control 22. ADPC22 controls the pin associated with channel AD22. 0 AD22 pin I/O control enabled 1 AD22 pin I/O control disabled
5 ADPC21	ADC Pin Control 21. ADPC21 controls the pin associated with channel AD21. 0 AD21 pin I/O control enabled 1 AD21 pin I/O control disabled
4 ADPC20	ADC Pin Control 20. ADPC20 controls the pin associated with channel AD20. 0 AD20 pin I/O control enabled 1 AD20 pin I/O control disabled
3 ADPC19	ADC Pin Control 19. ADPC19 controls the pin associated with channel AD19. 0 AD19 pin I/O control enabled 1 AD19 pin I/O control disabled
2 ADPC18	ADC Pin Control 18. ADPC18 controls the pin associated with channel AD18. 0 AD18 pin I/O control enabled 1 AD18 pin I/O control disabled

Table 10-12. APCTL3 Register Field Descriptions (continued)

Field	Description
1 ADPC17	ADC Pin Control 17. ADPC17 controls the pin associated with channel AD17. 0 AD17 pin I/O control enabled 1 AD17 pin I/O control disabled
0 ADPC16	ADC Pin Control 16. ADPC16 controls the pin associated with channel AD16. 0 AD16 pin I/O control enabled 1 AD16 pin I/O control disabled

10.4 Functional Description

The ADC module is disabled during reset or when the ADCH bits are all high. The module is idle when a conversion has completed and another conversion has not been initiated. When idle, the module is in its lowest power state.

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. In 12-bit and 10-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 12-bit digital result. In 8-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 9-bit digital result.

When the conversion is completed, the result is placed in the data registers (ADCRH and ADCRL). In 10-bit mode, the result is rounded to 10 bits and placed in the data registers (ADCRH and ADCRL). In 8-bit mode, the result is rounded to 8 bits and placed in ADCRL. The conversion complete flag (COCO) is then set and an interrupt is generated if the conversion complete interrupt has been enabled (AIEN = 1).

The ADC module has the capability of automatically comparing the result of a conversion with the contents of its compare registers. The compare function is enabled by setting the ACFE bit and operates with any of the conversion modes and configurations.

10.4.1 Clock Select and Divide Control

One of four clock sources can be selected as the clock source for the ADC module. This clock source is then divided by a configurable value to generate the input clock to the converter (ADCK). The clock is selected from one of the following sources by means of the ADICLK bits.

- The bus clock, which is equal to the frequency at which software is executed. This is the default selection following reset.
- The bus clock divided by two. For higher bus clock rates, this allows a maximum divide by 16 of the bus clock.
- ALTCLK, as defined for this MCU (See module section introduction).
- The asynchronous clock (ADACK). This clock is generated from a clock source within the ADC module. When selected as the clock source, this clock remains active while the MCU is in wait or stop3 mode and allows conversions in these modes for lower noise operation.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC do not perform according to specifications. If the available clocks

are too fast, the clock must be divided to the appropriate frequency. This divider is specified by the ADIV bits and can be divide-by 1, 2, 4, or 8.

10.4.2 Input Select and Pin Control

The pin control registers (APCTL3, APCTL2, and APCTL1) disable the I/O port control of the pins used as analog inputs. When a pin control register bit is set, the following conditions are forced for the associated MCU pin:

- The output buffer is forced to its high impedance state.
- The input buffer is disabled. A read of the I/O port returns a zero for any pin with its input buffer disabled.
- The pullup is disabled.

10.4.3 Hardware Trigger

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when the ADTRG bit is set. This source is not available on all MCUs. Consult the module introduction for information on the ADHWT source specific to this MCU.

When ADHWT source is available and hardware trigger is enabled (ADTRG=1), a conversion is initiated on the rising edge of ADHWT. If a conversion is in progress when a rising edge occurs, the rising edge is ignored. In continuous convert configuration, only the initial rising edge to launch continuous conversions is observed. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

10.4.4 Conversion Control

Conversions can be performed in 12-bit mode, 10-bit mode, or 8-bit mode as determined by the MODE bits. Conversions can be initiated by a software or hardware trigger. In addition, the ADC module can be configured for low power operation, long sample time, continuous conversion, and automatic compare of the conversion result to a software determined compare value.

10.4.4.1 Initiating Conversions

A conversion is initiated:

- Following a write to ADCSC1 (with ADCH bits not all 1s) if software triggered operation is selected.
- Following a hardware trigger (ADHWT) event if hardware triggered operation is selected.
- Following the transfer of the result to the data registers when continuous conversion is enabled.

If continuous conversions are enabled, a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, continuous conversions begin after ADCSC1 is written and continue until aborted. In hardware triggered operation, continuous conversions begin after a hardware trigger event and continue until aborted.

10.4.4.2 Completing Conversions

A conversion is completed when the result of the conversion is transferred into the data result registers, ADCRH and ADCRL. This is indicated by the setting of COCO. An interrupt is generated if AIEN is high at the time that COCO is set.

A blocking mechanism prevents a new result from overwriting previous data in ADCRH and ADCRL if the previous data is in the process of being read while in 12-bit or 10-bit MODE (the ADCRH register has been read but the ADCRL register has not). When blocking is active, the data transfer is blocked, COCO is not set, and the new result is lost. In the case of single conversions with the compare function enabled and the compare condition false, blocking has no effect and ADC operation is terminated. In all other cases of operation, when a data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled).

If single conversions are enabled, the blocking mechanism could result in several discarded conversions and excess power consumption. To avoid this issue, the data registers must not be read after initiating a single conversion until the conversion completes.

10.4.4.3 Aborting Conversions

Any conversion in progress is aborted when:

- A write to ADCSC1 occurs (the current conversion will be aborted and a new conversion will be initiated, if ADCH are not all 1s).
- A write to ADCSC2, ADCCFG, ADCCVH, or ADCCVL occurs. This indicates a mode of operation change has occurred and the current conversion is therefore invalid.
- The MCU is reset.
- The MCU enters stop mode with ADACK not enabled.

When a conversion is aborted, the contents of the data registers, ADCRH and ADCRL, are not altered. However, they continue to be the values transferred after the completion of the last successful conversion. If the conversion was aborted by a reset, ADCRH and ADCRL return to their reset states.

10.4.4.4 Power Control

The ADC module remains in its idle state until a conversion is initiated. If ADACK is selected as the conversion clock source, the ADACK clock generator is also enabled.

Power consumption when active can be reduced by setting ADLPC. This results in a lower maximum value for f_{ADCK} (see the electrical specifications).

10.4.4.5 Sample Time and Total Conversion Time

The total conversion time depends on the sample time (as determined by ADLSMP), the MCU bus frequency, the conversion mode (8-bit, 10-bit or 12-bit), and the frequency of the conversion clock (f_{ADCK}). After the module becomes active, sampling of the input begins. ADLSMP selects between short (3.5 ADCK cycles) and long (23.5 ADCK cycles) sample times. When sampling is complete, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the

digital value of the analog signal. The result of the conversion is transferred to ADCRH and ADCRL upon completion of the conversion algorithm.

If the bus frequency is less than the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when short sample is enabled (ADLSMP=0). If the bus frequency is less than 1/11th of the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when long sample is enabled (ADLSMP=1).

The maximum total conversion time for different conditions is summarized in [Table 10-13](#).

Table 10-13. Total Conversion Time vs. Control Conditions

Conversion Type	ADICLK	ADLSMP	Max Total Conversion Time
Single or first continuous 8-bit	0x, 10	0	20 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	0	23 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	0x, 10	1	40 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	1	43 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	11	0	5 μ s + 20 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	0	5 μ s + 23 ADCK + 5 bus clock cycles
Single or first continuous 8-bit	11	1	5 μ s + 40 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	1	5 μ s + 43 ADCK + 5 bus clock cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	17 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	20 ADCK cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	37 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	40 ADCK cycles

The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by the ADICLK bits, and the divide ratio is specified by the ADIV bits. For example, in 10-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-1 ratio selected, and a bus frequency of 8 MHz, then the conversion time for a single conversion is:

$$\text{Conversion time} = \frac{23 \text{ ADCK Cyc}}{8 \text{ MHz}/1} + \frac{5 \text{ bus Cyc}}{8 \text{ MHz}} = 3.5 \mu\text{s}$$

$$\text{Number of bus cycles} = 3.5 \mu\text{s} \times 8 \text{ MHz} = 28 \text{ cycles}$$

NOTE

The ADCK frequency must be between f_{ADCK} minimum and f_{ADCK} maximum to meet ADC specifications.

10.4.5 Automatic Compare Function

The compare function is enabled by the ACFE bit. The compare function can be configured to check for an upper or lower limit. After the input is sampled and converted, the compare value (ADCCVH and ADCCVL) is subtracted from the conversion result. When comparing to an upper limit (ACFGT = 1), if the conversion result is greater-than or equal-to the compare value, COCO is set. When comparing to a lower limit (ACFGT = 0), if the result is less than the compare value, COCO is set. An ADC interrupt is generated upon the setting of COCO if the ADC interrupt is enabled (AIEN = 1).

The subtract operation of two positive values (the conversion result less the compare value) results in a signed value that is 1-bit wider than the bit-width of the two terms. The final value transferred to the ADCRH and ADCRL registers is the result of the subtraction operation, excluding the sign bit. The value of the sign bit can be derived based on ACFGT control setting. When ACFGT=1, the sign bit of any value stored in ADCRH and ADCRL is always 0, indicating a positive result for the subtract operation. When ACFGT = 1, the sign bit of any result is always 1, indicating a negative result for the subtract operation.

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, COCO is not set and no data is transferred to the result registers.

NOTE

The compare function can monitor the voltage on a channel while the MCU is in wait or stop3 mode. The ADC interrupt wakes the MCU when the compare condition is met.

An example of compare operation eases understanding of the compare feature. If the ADC is configured for 10-bit operation, ACFGT=0, and ADCCVH:ADCCVL= 0x200, then a conversion result of 0x080 causes the compare condition to be met and the COCO bit is set. A value of 0x280 is stored in ADCRH:ADCRL. This is signed data without the sign bit and must be combined with a derived sign bit to have meaning. The value stored in ADCRH:ADCRL is calculated as follows.

The value to interpret from the data is (Result – Compare Value) = (0x080 – 0x200) = –0x180. A standard method for handling subtraction is to convert the second term to its 2's complement, and then add the two terms. First calculate the 2's complement of 0x200 by complementing each bit and adding 1. Note that prior to complementing, a sign bit of 0 is added so that the 10-bit compare value becomes a 11-bit signed value that is always positive.

$$\begin{array}{rcl}
 \%101\ 1111\ 1111 & & \leq 1\text{'s complement of } 0x200 \text{ compare value} \\
 + & & \%1 \\
 \hline
 \%110\ 0000\ 0000 & & \leq 2\text{'s complement of } 0x200 \text{ compare value}
 \end{array}$$

Then the conversion result of 0x080 is added to 2's complement of 0x200:

$$\begin{array}{rcl}
 \%000\ 1000\ 0000 & & \\
 + & & \%110\ 0000\ 0000 \\
 \hline
 \%110\ 1000\ 0000 & & \leq \text{Subtraction result is } -0x180 \text{ in signed 11-bit data}
 \end{array}$$

The subtraction result is an 11-bit signed value. The lower 10 bits (0x280) are stored in ADCRH:ADCRL. The sign bit is known to be 1 (negative) because the ACFG_T=0, the COCO bit was set, and conversion data was updated in ADCRH:ADCRL.

A simpler way to use the data stored in ADCRH:ADCRL is to apply the following rules. When comparing for upper limit (ACFG_T=1), the value in ADCRH:ADCRL is a positive value and does not need to be manipulated. This value is the difference between the conversion result and the compare value. When comparing for lower limit (ACFG_T=0), ADCRH:ADCRL is a negative value without the sign bit. If the value from these registers is complemented and then a value of 1 is added, then the calculated value is the unsigned (i.e., absolute) difference between the conversion result and the compare value. In the previous example, 0x280 is stored in ADCRH:ADCRL. The following example shows how the absolute value of the difference is calculated.

```

%01 0111 1111 <= Complement of 10-bit value stored in ADCRH:ADCRL
+
%1
-----
%01 1000 0000 <= Unsigned value 0x180 is the absolute value of (Result - Compare Value)

```

10.4.6 MCU Wait Mode Operation

Wait mode is a lower power-consumption standby mode from which recovery is fast because the clock sources remain active. If a conversion is in progress when the MCU enters wait mode, it continues until completion. Conversions can be initiated while the MCU is in wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, bus clock divided by two, and ADACK are available as conversion clock sources while in wait mode. The use of ALTCLK as the conversion clock source in wait is dependent on the definition of ALTCLK for this MCU. Consult the module introduction for information on ALTCLK specific to this MCU.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from wait mode if the ADC interrupt is enabled (AIEN = 1).

10.4.7 MCU Stop3 Mode Operation

Stop mode is a low power-consumption standby mode during which most or all clock sources on the MCU are disabled.

10.4.7.1 Stop3 Mode With ADACK Disabled

If the asynchronous clock, ADACK, is not selected as the conversion clock, executing a stop instruction aborts the current conversion and places the ADC in its idle state. The contents of ADCRH and ADCRL are unaffected by stop3 mode. After exiting from stop3 mode, a software or hardware trigger is required to resume conversions.

10.4.7.2 Stop3 Mode With ADACK Enabled

If ADACK is selected as the conversion clock, the ADC continues operation during stop3 mode. For guaranteed ADC operation, the MCU's voltage regulator must remain active during stop3 mode. Consult the module introduction for configuration information for this MCU.

If a conversion is in progress when the MCU enters stop3 mode, it continues until completion. Conversions can be initiated while the MCU is in stop3 mode by means of the hardware trigger or if continuous conversions are enabled.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from stop3 mode if the ADC interrupt is enabled (AIEN = 1).

NOTE

The ADC module can wake the system from low-power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure the data transfer blocking mechanism (discussed in [Section 10.4.4.2, Completing Conversions](#)) is cleared when entering stop3 and continuing ADC conversions.

10.4.8 MCU Stop2 Mode Operation

The ADC module is automatically disabled when the MCU enters stop2 mode. All module registers contain their reset values following exit from stop2. Therefore, the module must be re-enabled and re-configured following exit from stop2.

10.5 Initialization Information

This section gives an example that provides some basic direction on how to initialize and configure the ADC module. You can configure the module for 8-, 10-, or 12-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to [Table 10-7](#), [Table 10-8](#), and [Table 10-9](#) for information used in this example.

NOTE

Hexadecimal values designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

10.5.1 ADC Module Initialization Example

10.5.1.1 Initialization Sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be performed. A typical sequence is as follows:

1. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.

2. Update status and control register 2 (ADCSC2) to select the conversion trigger (hardware or software) and compare function options, if enabled.
3. Update status and control register 1 (ADCSC1) to select whether conversions will be continuous or completed only once, and to enable or disable conversion complete interrupts. The input channel on which conversions will be performed is also selected here.

10.5.1.2 Pseudo-Code Example

In this example, the ADC module is set up with interrupts enabled to perform a single 10-bit conversion at low power with a long sample time on input channel 1, where the internal ADCK clock is derived from the bus clock divided by 1.

ADCCFG = 0x98 (%10011000)

Bit 7	ADLPC	1	Configures for low power (lowers maximum clock speed)
Bit 6:5	ADIV	00	Sets the ADCK to the input clock ÷ 1
Bit 4	ADLSMP	1	Configures for long sample time
Bit 3:2	MODE	10	Sets mode at 10-bit conversions
Bit 1:0	ADICLK	00	Selects bus clock as input clock source

ADCSC2 = 0x00 (%00000000)

Bit 7	ADACT	0	Flag indicates if a conversion is in progress
Bit 6	ADTRG	0	Software trigger selected
Bit 5	ACFE	0	Compare function disabled
Bit 4	ACFGT	0	Not used in this example
Bit 3:2		00	Reserved, always reads zero
Bit 1:0		00	Reserved for NXP's internal use; always write zero

ADCSC1 = 0x41 (%01000001)

Bit 7	COCO	0	Read-only flag which is set when a conversion completes
Bit 6	AIEN	1	Conversion complete interrupt enabled
Bit 5	ADCO	0	One conversion only (continuous conversions disabled)
Bit 4:0	ADCH	00001	Input channel 1 selected as ADC input channel

ADCRH/L = 0xxx

Holds results of conversion. Read high byte (ADCRH) before low byte (ADCRL) so that conversion data cannot be overwritten with data from the next conversion.

ADCCVH/L = 0xxx

Holds compare value when compare function enabled

APCTL1=0x02

AD1 pin I/O control disabled. All other AD pins remain general purpose I/O pins

APCTL2=0x00

All other AD pins remain general purpose I/O pins

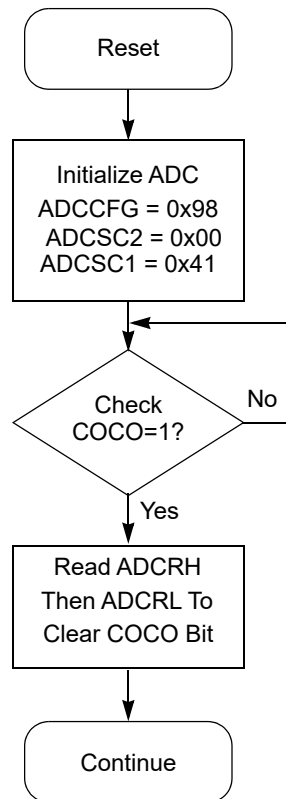


Figure 10-12. Initialization Flowchart for Example

10.6 Application Information

This section contains information for using the ADC module in applications. The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an A/D converter.

10.6.1 External Pins and Routing

The following sections discuss the external pins associated with the ADC module and how they should be used for best results.

10.6.1.1 Analog Supply Pins

The ADC module has analog power and ground supplies (V_{DDA} and V_{SSA}) available as separate pins on some devices. V_{SSA} is shared on the same pin as the MCU digital V_{SS} on some devices. On other devices, V_{SSA} and V_{DDA} are shared with the MCU digital supply pins. In these cases, there are separate pads for the analog supplies bonded to the same pin as the corresponding digital supply so that some degree of isolation between the supplies is maintained.

When available on a separate pin, both V_{DDA} and V_{SSA} must be connected to the same voltage potential as their corresponding MCU digital supply (V_{DD} and V_{SS}) and must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

If separate power supplies are used for analog and digital power, the ground connection between these supplies must be at the V_{SSA} pin. This should be the only ground connection between these supplies if possible. The V_{SSA} pin makes a good single point ground location.

10.6.1.2 Analog Reference Pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs. The high reference is V_{REFH} , which may be shared on the same pin as V_{DDA} on some devices. The low reference is V_{REFL} , which may be shared on the same pin as V_{SSA} on some devices.

When available on a separate pin, V_{REFH} may be connected to the same potential as V_{DDA} , or may be driven by an external source between the minimum V_{DDA} spec and the V_{DDA} potential (V_{REFH} must never exceed V_{DDA}). When available on a separate pin, V_{REFL} must be connected to the same voltage potential as V_{SSA} . V_{REFH} and V_{REFL} must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the V_{REFH} and V_{REFL} loop. The best external component to meet this current demand is a 0.1 μF capacitor with good high frequency characteristics. This capacitor is connected between V_{REFH} and V_{REFL} and must be placed as near as possible to the package pins. Resistance in the path is not recommended because the current causes a voltage drop that could result in conversion errors. Inductance in this path must be minimum (parasitic only).

10.6.1.3 Analog Input Pins

The external analog inputs are typically shared with digital I/O pins on MCU devices. The pin I/O control is disabled by setting the appropriate control bit in one of the pin control registers. Conversions can be performed on inputs without the associated pin control register bit set. It is recommended that the pin control register bit always be set when using a pin as an analog input. This avoids problems with contention because the output buffer is in its high impedance state and the pullup is disabled. Also, the input buffer draws DC current when its input is not at V_{DD} or V_{SS} . Setting the pin control register bits for all pins used as analog inputs should be done to achieve lowest operating current.

Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. Use of 0.01 μF capacitors with good high-frequency characteristics is sufficient. These capacitors are not necessary in all cases, but when used they must be placed as near as possible to the package pins and be referenced to V_{SSA} .

For proper conversion, the input voltage must fall between V_{REFH} and V_{REFL} . If the input is equal to or exceeds V_{REFH} , the converter circuit converts the signal to 0xFFF (full scale 12-bit representation), 0x3FF (full scale 10-bit representation) or 0xFF (full scale 8-bit representation). If the input is equal to or less than V_{REFL} , the converter circuit converts it to 0x000. Input voltages between V_{REFH} and V_{REFL} are straight-line linear conversions. There is a brief current associated with V_{REFL} when the sampling

capacitor is charging. The input is sampled for 3.5 cycles of the ADCK source when ADLSMP is low, or 23.5 cycles when ADLSMP is high.

For minimal loss of accuracy due to current injection, pins adjacent to the analog input pins should not be transitioning during conversions.

10.6.2 Sources of Error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

10.6.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7k Ω and input capacitance of approximately 5.5 pF, sampling to within 1/4LSB (at 12-bit resolution) can be achieved within the minimum sample window (3.5 cycles @ 8 MHz maximum ADCK frequency) provided the resistance of the external analog source (R_{AS}) is kept below 2 k Ω .

Higher source resistances or higher-accuracy sampling is possible by setting ADLSMP (to increase the sample window to 23.5 cycles) or decreasing ADCK frequency to increase sample time.

10.6.2.2 Pin Leakage Error

Leakage on the I/O pins can cause conversion error if the external analog source resistance (R_{AS}) is high. If this error cannot be tolerated by the application, keep R_{AS} lower than $V_{DDA} / (2^N \cdot I_{LEAK})$ for less than 1/4LSB leakage error ($N = 8$ in 8-bit, 10 in 10-bit or 12 in 12-bit mode).

10.6.2.3 Noise-Induced Errors

System noise that occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1 μ F low-ESR capacitor from V_{REFH} to V_{REFL} .
- There is a 0.1 μ F low-ESR capacitor from V_{DDA} to V_{SSA} .
- If inductive isolation is used from the primary supply, an additional 1 μ F capacitor is placed from V_{DDA} to V_{SSA} .
- V_{SSA} (and V_{REFL} , if connected) is connected to V_{SS} at a quiet point in the ground plane.
- Operate the MCU in wait or stop3 mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.
 - For software triggered conversions, immediately follow the write to ADCSC1 with a wait instruction or stop instruction.
 - For stop3 mode operation, select ADACK as the clock source. Operation in stop3 reduces V_{DD} noise but increases effective conversion time due to stop recovery.
- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive V_{DD} noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in wait or stop3 or I/O activity cannot be halted, these recommended actions may reduce the effect of noise on the accuracy:

- Place a 0.01 μF capacitor (C_{AS}) on the selected input channel to V_{REFL} or V_{SSA} (this improves noise issues, but affects the sample rate based on the external analog source resistance).
- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

10.6.2.4 Code Width and Quantization Error

The ADC quantizes the ideal straight-line transfer function into 4096 steps (in 12-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8, 10 or 12), defined as 1LSB, is:

$$1 \text{ lsb} = (V_{REFH} - V_{REFL}) / 2^N \quad \text{Eqn. 10-2}$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be $\pm 1/2$ lsb in 8- or 10-bit mode. As a consequence, however, the code width of the first (0x000) conversion is only 1/2 lsb and the code width of the last (0xFF or 0x3FF) is 1.5 lsb.

For 12-bit conversions the code transitions only after the full code width is present, so the quantization error is -1 lsb to 0 lsb and the code width of each step is 1 lsb.

10.6.2.5 Linearity Errors

The ADC may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the system should be aware of them because they affect overall accuracy. These errors are:

- Zero-scale error (E_{ZS}) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2 lsb in 8-bit or 10-bit modes and 1 lsb in 12-bit mode). If the first conversion is 0x001, the difference between the actual 0x001 code width and its ideal (1 lsb) is used.
- Full-scale error (E_{FS}) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5 lsb in 8-bit or 10-bit modes and 1LSB in 12-bit mode). If the last conversion is 0x3FE, the difference between the actual 0x3FE code width and its ideal (1LSB) is used.
- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.

- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function and includes all forms of error.

10.6.2.6 Code Jitter, Non-Monotonicity, and Missing Codes

Analog-to-digital converters are susceptible to three special forms of error. These are code jitter, non-monotonicity, and missing codes.

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around $\pm 1/2$ lsb in 8-bit or 10-bit mode, or around 2 lsb in 12-bit mode, and increases with noise.

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in [Section 10.6.2.3](#) reduces this error.

Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage. Missing codes are those values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and have no missing codes.

Chapter 11

Internal Clock Source (S08ICSV3)

11.1 Introduction

The internal clock source (ICS) module provides clock source choices for the MCU. The module contains a frequency-locked loop (FLL) as a clock source that is controllable by either an internal or an external reference clock. The module can provide this FLL clock or either of the internal or external reference clocks as a source for the MCU system clock. There are also signals provided to control a low power oscillator (XOSCVLP) module to allow the use of an external crystal/resonator as the external reference clock.

Whichever clock source is chosen, it is passed through a reduced bus divider (BDIV) which allows a lower final output clock frequency to be derived.

11.1.1 DCO Select bits

The ICS on the MC9S08QL8 series is configured to support only the low range DCO, therefore the DRS and DRST bits in ICSSC have no effect.

11.1.2 Features

Key features of the ICS module are:

- Frequency-locked loop (FLL) is trimmable for accuracy
- Internal or external reference clocks can be used to control the FLL
- Reference divider is provided for external clock
- Internal reference clock has 9 trim bits available
- Internal or external reference clocks can be selected as the clock source for the MCU
- Whichever clock is selected as the source can be divided down
 - 2-bit select for clock divider is provided
 - Allowable dividers are: 1, 2, 4, 8
- Control signals for a low power oscillator clock generator (OSCOUT) as the ICS external reference clock are provided
 - HGO, RANGE, EREFS, ERCLKEN, EREFSTEN
- FLL Engaged Internal mode is automatically selected out of reset
- BDC clock is provided as a constant divide by 2 of the low range DCO output
- Three selectable digitally-controlled oscillators (DCO) optimized for different frequency ranges.
- Option to maximize output frequency for a 32768 Hz external reference clock source.

11.1.3 Block Diagram

Figure 11-1 is the ICS block diagram.

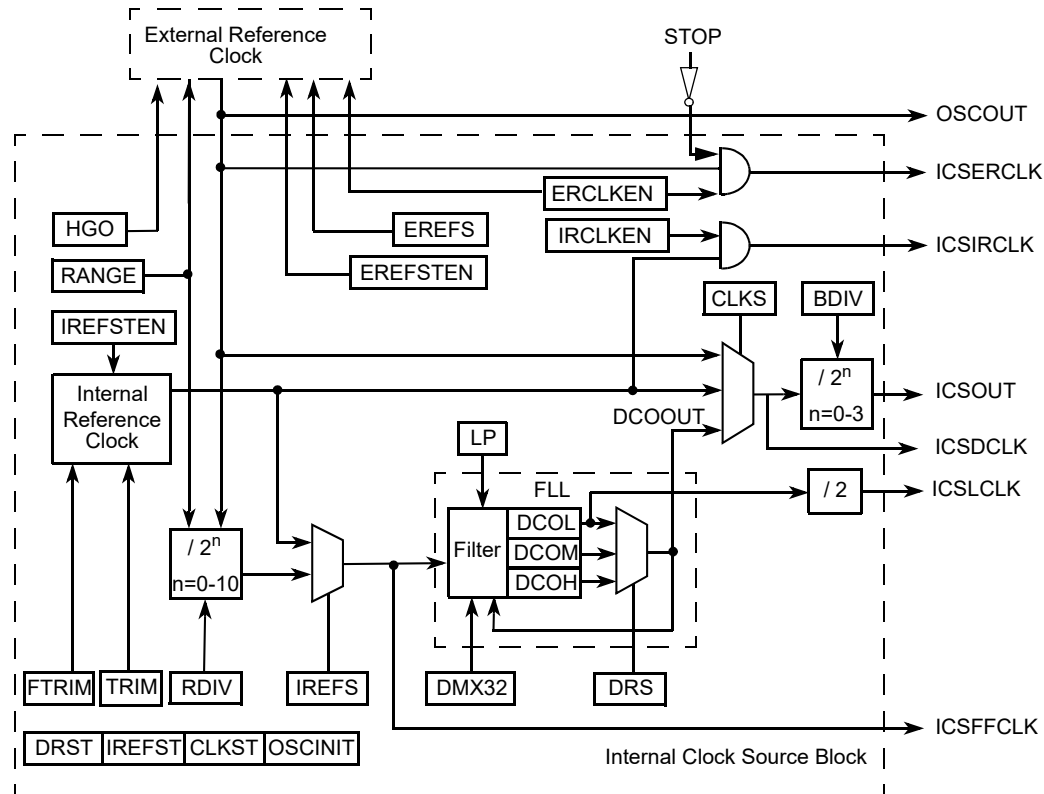


Figure 11-1. Internal Clock Source (ICS) Block Diagram

11.1.4 Modes of Operation

There are seven modes of operation for the ICS: FEI, FEE, FBI, FBILP, FBE, FBELP, and stop.

11.1.4.1 FLL Engaged Internal (FEI)

In FLL engaged internal mode, which is the default mode, the ICS supplies a clock derived from the FLL which is controlled by the internal reference clock. The BDC clock is supplied from the FLL.

11.1.4.2 FLL Engaged External (FEE)

In FLL engaged external mode, the ICS supplies a clock derived from the FLL which is controlled by an external reference clock source. The BDC clock is supplied from the FLL.

11.1.4.3 FLL Bypassed Internal (FBI)

In FLL bypassed internal mode, the FLL is enabled and controlled by the internal reference clock, but is bypassed. The ICS supplies a clock derived from the internal reference clock. The BDC clock is supplied from the FLL.

11.1.4.4 FLL Bypassed Internal Low Power (FBILP)

In FLL bypassed internal low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the internal reference clock. The BDC clock is not available.

11.1.4.5 FLL Bypassed External (FBE)

In FLL bypassed external mode, the FLL is enabled and controlled by an external reference clock, but is bypassed. The ICS supplies a clock derived from the external reference clock source. The BDC clock is supplied from the FLL.

11.1.4.6 FLL Bypassed External Low Power (FBELP)

In FLL bypassed external low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the external reference clock. The BDC clock is not available.

11.1.4.7 Stop (STOP)

In stop mode, the FLL is disabled and the internal or the ICS external reference clocks source (OSCOUT) can be selected to be enabled or disabled. The BDC clock is not available and the ICS does not provide an MCU clock source.

NOTE

The DCO frequency changes from the pre-stop value to its reset value and the FLL will need to re-acquire the lock before the frequency is stable. Timing sensitive operations should wait for the FLL acquisition time, $t_{Acquire}$, before executing.

11.2 External Signal Description

There are no ICS signals that connect off chip.

11.3 Register Definition

Figure 11-1 is a summary of ICS registers.

Table 11-1. ICS Register Summary

Name		7	6	5	4	3	2	1	0
ICSC1	R	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
	W								
ICSC2	R	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
	W								
ICSTRM	R	TRIM							
	W								

Table 11-1. ICS Register Summary (continued)

Name		7	6	5	4	3	2	1	0
ICSSC	R	DRST		DMX32	IREFST	CLKST		OSCINIT	FTRIM
	W	DRS							

11.3.1 ICS Control Register 1 (ICSC1)

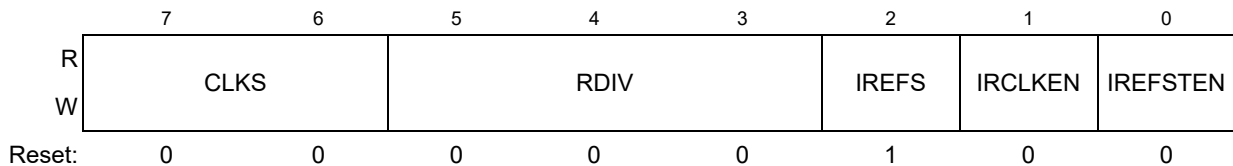


Figure 11-2. ICS Control Register 1 (ICSC1)

Table 11-2. ICS Control Register 1 Field Descriptions

Field	Description
7:6 CLKS	Clock Source Select — Selects the clock source that controls the bus frequency. The actual bus frequency depends on the value of the BDIV bits. 00 Output of FLL is selected. 01 Internal reference clock is selected. 10 External reference clock is selected. 11 Reserved, defaults to 00.
5:3 RDIV	Reference Divider — Selects the amount to divide down the external reference clock. Resulting frequency must be in the range 31.25 kHz to 39.0625 kHz. See Table 11-3 for the divide-by factors.
2 IREFS	Internal Reference Select — The IREFS bit selects the reference clock source for the FLL. 1 Internal reference clock selected. 0 External reference clock selected.
1 IRCLKEN	Internal Reference Clock Enable — The IRCLKEN bit enables the internal reference clock for use as ICSIRCLK. 1 ICSIRCLK active. 0 ICSIRCLK inactive.
0 IREFSTEN	Internal Reference Stop Enable — The IREFSTEN bit controls whether or not the internal reference clock remains enabled when the ICS enters stop mode. 1 Internal reference clock stays enabled in stop if IRCLKEN is set before entering stop. 0 Internal reference clock is disabled in stop.

Table 11-3. Reference Divide Factor

RDIV	RANGE=0	RANGE=1
0	1 ¹	32
1	2	64
2	4	128
3	8	256

Table 11-3. Reference Divide Factor

RDIV	RANGE=0	RANGE=1
4	16	512
5	32	1024
6	64	Reserved
7	128	Reserved

¹ Reset default

11.3.2 ICS Control Register 2 (ICSC2)

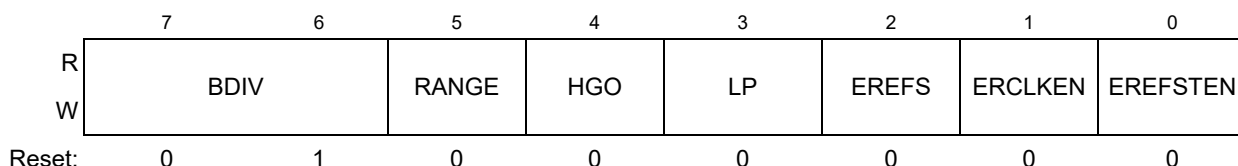
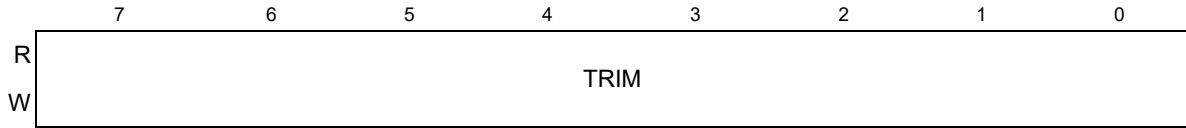


Figure 11-3. ICS Control Register 2 (ICSC2)

Table 11-4. ICS Control Register 2 Field Descriptions

Field	Description
7:6 BDIV	Bus Frequency Divider — Selects the amount to divide down the clock source selected by the CLKS bits. This controls the bus frequency. 00 Encoding 0 — Divides selected clock by 1. 01 Encoding 1 — Divides selected clock by 2 (reset default). 10 Encoding 2 — Divides selected clock by 4. 11 Encoding 3 — Divides selected clock by 8.
5 RANGE	Frequency Range Select — Selects the frequency range for the external oscillator. 1 High frequency range selected for the external oscillator. 0 Low frequency range selected for the external oscillator.
4 HGO	High Gain Oscillator Select — The HGO bit controls the external oscillator mode of operation. 1 Configure external oscillator for high gain operation. 0 Configure external oscillator for low power operation.
3 LP	Low Power Select — The LP bit controls whether the FLL is disabled in FLL bypassed modes. 1 FLL is disabled in bypass modes unless BDM is active. 0 FLL is not disabled in bypass mode.
2 EREFS	External Reference Select — The EREFS bit selects the source for the external reference clock. 1 Oscillator requested. 0 External Clock Source requested.
1 ERCLKEN	External Reference Enable — The ERCLKEN bit enables the external reference clock for use as ICSECLK. 1 ICSECLK active. 0 ICSECLK inactive.
0 EREFSTEN	External Reference Stop Enable — The EREFSTEN bit controls whether or not the external reference clock source (OSCAOUT) remains enabled when the ICS enters stop mode. 1 External reference clock source stays enabled in stop if ERCLKEN is set before entering stop. 0 External reference clock source is disabled in stop.

11.3.3 ICS Trim Register (ICSTRM)



Reset: Note: TRIM is loaded during reset from a factory programmed location when not in BDM mode. If in a BDM mode, a default value of 0x80 is loaded.

Figure 11-4. ICS Trim Register (ICSTRM)

Table 11-5. ICS Trim Register Field Descriptions

Field	Description
7:0 TRIM	ICS Trim Setting — The TRIM bits control the internal reference clock frequency by controlling the internal reference clock period. The bits' effect are binary weighted (in other words, bit 1 adjusts twice as much as bit 0). Increasing the binary value in TRIM will increase the period, and decreasing the value will decrease the period. An additional fine trim bit is available in ICSSC as the FTRIM bit.

11.3.4 ICS Status and Control (ICSSC)

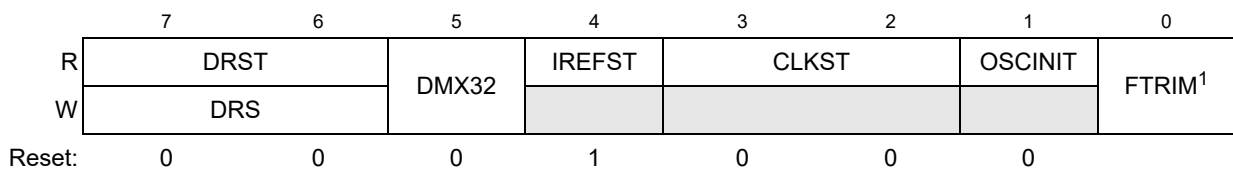


Figure 11-5. ICS Status and Control Register (ICSSC)

¹ FTRIM is loaded during reset from a factory programmed location when not in any BDM mode. If in a BDM mode, FTRIM gets loaded with a value of 1'b0.

Table 11-6. ICS Status and Control Register Field Descriptions

Field	Description
7-6 DRST DRS	DCO Range Status — The DRST read field indicates the current frequency range for the FLL output, DCOOUT. See Table 11-7. The DRST field does not update immediately after a write to the DRS field due to internal synchronization between clock domains. Writing the DRS bits to 2'b11 is ignored and the DRST bits remain with the current setting. DCO Range Select — The DRS field selects the frequency range for the FLL output, DCOOUT. Writes to the DRS field while the LP bit is set are ignored. 00 Low range. 01 Mid range. 10 High range. 11 Reserved.
5 DMX32	DCO Maximum frequency with 32.768 kHz reference — The DMX32 bit controls whether or not the DCO frequency range is narrowed to its maximum frequency with a 32.768 kHz reference. See Table 11-7. 0 DCO has default range of 25%. 1 DCO is fined tuned for maximum frequency with 32.768 kHz reference.

Table 11-6. ICS Status and Control Register Field Descriptions (continued)

Field	Description
4 IREFST	Internal Reference Status — The IREFST bit indicates the current source for the reference clock. The IREFST bit does not update immediately after a write to the IREFS bit due to internal synchronization between clock domains. 0 Source of reference clock is external clock. 1 Source of reference clock is internal clock.
3-2 CLKST	Clock Mode Status — The CLKST bits indicate the current clock mode. The CLKST bits don't update immediately after a write to the CLKS bits due to internal synchronization between clock domains. 00 Output of FLL is selected. 01 FLL Bypassed, Internal reference clock is selected. 10 FLL Bypassed, External reference clock is selected. 11 Reserved.
1 OSCINIT	OSC Initialization — If the external reference clock is selected by ERCLKEN or by the ICS being in FEE, FBE, or FBELP mode, and if EREFS is set, then this bit is set after the initialization cycles of the external oscillator clock have completed. This bit is only cleared when either ERCLKEN or EREFS are cleared.
0 FTRIM	ICS Fine Trim — The FTRIM bit controls the smallest adjustment of the internal reference clock frequency. Setting FTRIM will increase the period and clearing FTRIM will decrease the period by the smallest amount possible.

Table 11-7. DCO frequency range¹

DRS	DMX32	Reference range	FLL factor	DCO range
00	0	31.25 - 39.0625 kHz	512	16 - 20 MHz
	1	32.768 kHz	608	19.92 MHz
01	0	31.25 - 39.0625 kHz	1024	32 - 40 MHz
	1	32.768 kHz	1216	39.85 MHz
10	0	31.25 - 39.0625 kHz	1536	48 - 60 MHz
	1	32.768 kHz	1824	59.77 MHz
11	Reserved			

¹ The resulting bus clock frequency should not exceed the maximum specified bus clock frequency of the device.

11.4 Functional Description

11.4.1 Operational Modes

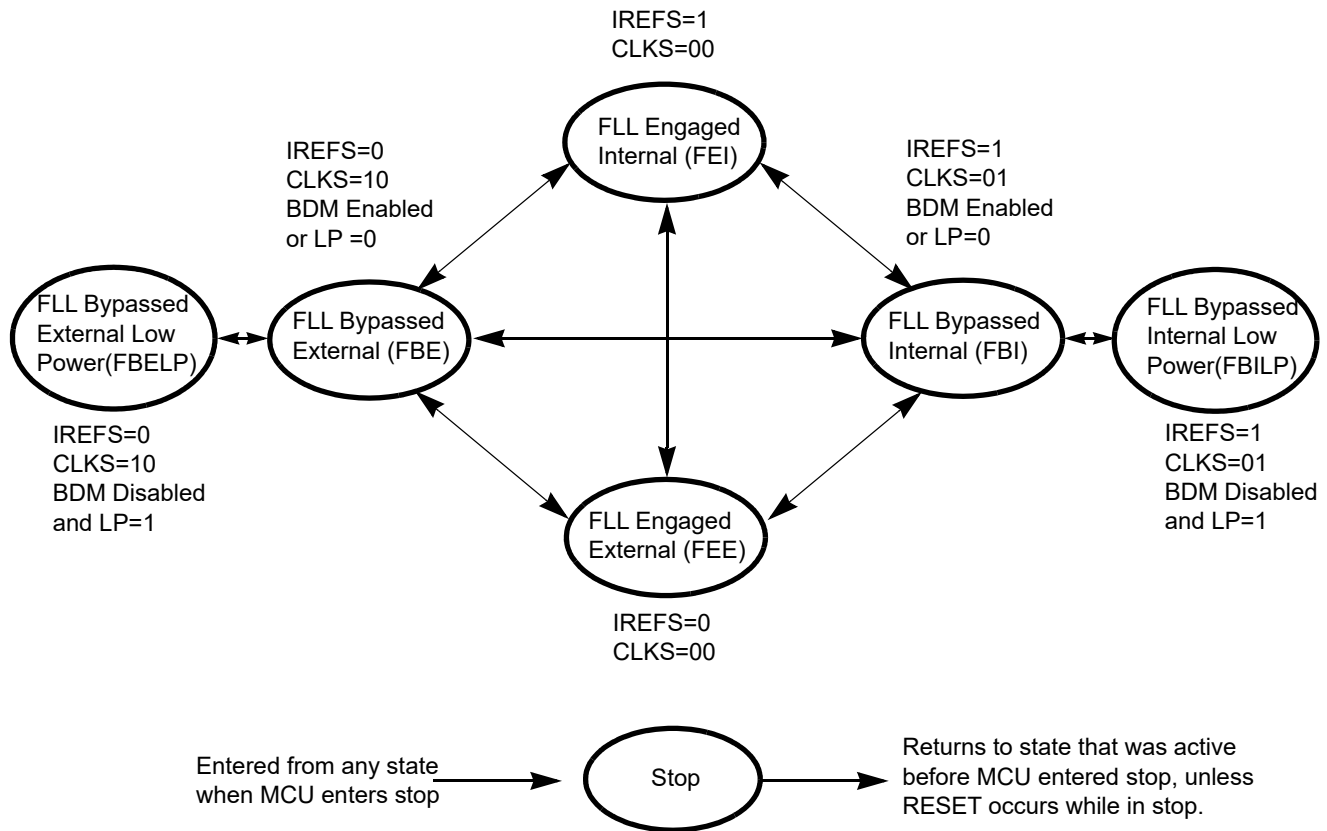


Figure 11-6. Clock Switching Modes

The seven states of the ICS are shown as a state diagram and are described below. The arrows indicate the allowed movements between the states.

11.4.1.1 FLL Engaged Internal (FEI)

FLL engaged internal (FEI) is the default mode of operation and is entered when all the following conditions occur:

- CLKS bits are written to 00.
- IREFS bit is written to 1.

In FLL engaged internal mode, the ICSOUT clock is derived from the FLL clock, which is controlled by the internal reference clock. The FLL loop locks the frequency to the FLL factor times the internal reference frequency. The ICSLCLK is available for BDC communications, and the internal reference clock is enabled.

11.4.1.2 FLL Engaged External (FEE)

The FLL engaged external (FEE) mode is entered when all the following conditions occur:

- CLKS bits are written to 00.
- IREFS bit is written to 0.
- RDIV bits are written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz.

In FLL engaged external mode, the ICSOUT clock is derived from the FLL clock which is controlled by the external reference clock source. The FLL loop locks the frequency to the FLL factor times the external reference frequency, as selected by the RDIV bits. The ICSLCLK is available for BDC communications, and the external reference clock is enabled.

11.4.1.3 FLL Bypassed Internal (FBI)

The FLL bypassed internal (FBI) mode is entered when all the following conditions occur:

- CLKS bits are written to 01.
- IREFS bit is written to 1.
- BDM mode is active or LP bit is written to 0.

In FLL bypassed internal mode, the ICSOUT clock is derived from the internal reference clock. The FLL clock is controlled by the internal reference clock, and the FLL loop locks the FLL frequency to the FLL factor times the internal reference frequency. The ICSLCLK will be available for BDC communications, and the internal reference clock is enabled.

11.4.1.4 FLL Bypassed Internal Low Power (FBILP)

The FLL bypassed internal low power (FBILP) mode is entered when all the following conditions occur:

- CLKS bits are written to 01.
- IREFS bit is written to 1.
- BDM mode is not active and LP bit is written to 1.

In FLL bypassed internal low power mode, the ICSOUT clock is derived from the internal reference clock and the FLL is disabled. The ICSLCLK will not be available for BDC communications, and the internal reference clock is enabled.

11.4.1.5 FLL Bypassed External (FBE)

The FLL bypassed external (FBE) mode is entered when all the following conditions occur:

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- RDIV bits are written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz.
- BDM mode is active or LP bit is written to 0.

In FLL bypassed external mode, the ICSOUT clock is derived from the external reference clock source. The FLL clock is controlled by the external reference clock, and the FLL loop locks the FLL frequency to the FLL factor times the external reference frequency, as selected by the RDIV bits, so that the ICSLCLK will be available for BDC communications, and the external reference clock is enabled.

11.4.1.6 FLL Bypassed External Low Power (FBELP)

The FLL bypassed external low power (FBELP) mode is entered when all the following conditions occur:

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- BDM mode is not active and LP bit is written to 1.

In FLL bypassed external low power mode, the ICSOUT clock is derived from the external reference clock source and the FLL is disabled. The ICSLCLK will not be available for BDC communications. The external reference clock source is enabled.

11.4.1.7 Stop

Stop mode is entered whenever the MCU enters a STOP state. In this mode, all ICS clock signals are static except in the following cases:

ICSIRCLK will be active in stop mode when all the following conditions occur:

- IRCLKEN bit is written to 1.
- IREFSTEN bit is written to 1.

OSCOUT will be active in stop mode when all the following conditions occur:

- ERCLKEN bit is written to 1.
- EREFSTEN bit is written to 1.

11.4.2 Mode Switching

The IREF bit can be changed at anytime, but the actual switch to the newly selected clock is shown by the IREFST bit. When switching between FLL engaged internal (FEI) and FLL engaged external (FEE) modes, the FLL begins locking again after the switch is completed.

The CLKS bits can also be changed at anytime, but the actual switch to the newly selected clock is shown by the CLKST bits. If the newly selected clock is not available, the previous clock remains selected.

The DRS bits can be changed at anytime except when LP bit is 1. If the DRS bits are changed while in FLL engaged internal (FEI) or FLL engaged external (FEE), the bus clock remains at the previous DCO range until the new DCO starts. When the new DCO starts the bus clock switches to it. After switching to the new DCO the FLL remains unlocked for several reference cycles. Once the selected DCO startup time is over, the FLL is locked. The completion of the switch is shown by the DRST bits.

11.4.3 Bus Frequency Divider

The BDIV bits can be changed at anytime and the actual switch to the new frequency occurs immediately.

11.4.4 Low Power Bit Usage

The low power bit (LP) is provided to allow the FLL to be disabled and thus conserve power when it is not being used. The DRS bits can not be written while LP bit is 1.

However, in some applications it may be desirable to allow the FLL to be enabled and to lock for maximum accuracy before switching to an FLL engaged mode. To do this, write the LP bit to 0.

11.4.5 DCO Maximum Frequency with 32.768 kHz Oscillator

The FLL has an option to change the clock multiplier for the selected DCO range such that it results in the maximum bus frequency with a common 32.768 kHz crystal reference clock.

11.4.6 Internal Reference Clock

When IRCLKEN is set the internal reference clock signal is presented as ICSIRCLK, which can be used as an additional clock source. To re-target the ICSIRCLK frequency, write a new value to the TRIM bits in the ICSTRM register to trim the period of the internal reference clock:

- Writing a larger value slows down the ICSIRCLK frequency.
- Writing a smaller value to the ICSTRM register speeds up the ICSIRCLK frequency.

The TRIM bits effect the ICSOUT frequency if the ICS is in FLL engaged internal (FEI), FLL bypassed internal (FBI), or FLL bypassed internal low power (FBILP) mode.

Until ICSIRCLK is trimmed, programming low reference divider (RDIV) factors may result in ICSOUT frequencies that exceed the maximum chip-level frequency and violate the chip-level clock timing specifications (see [Chapter 1, Device Overview](#)).

If IREFSTEN is set and the IRCLKEN bit is written to 1, the internal reference clock keeps running during stop mode in order to provide a fast recovery upon exiting stop.

All MCU devices are factory programmed with a trim value in a reserved memory location. This value is uploaded to the ICSTRM register and ICS FTRIM register during any reset initialization. For finer precision, trim the internal oscillator in the application and set the FTRIM bit accordingly.

11.4.7 External Reference Clock

The ICS module supports an external reference clock with frequencies between 31.25 kHz to 40 MHz in all modes. When the ERCLKEN is set, the external reference clock signal is presented as ICSECLK, which can be used as an additional clock source in run mode. When IREFS = 1, the external reference clock is not used by the FLL and will only be used as ICSECLK. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications support (see [Chapter 1, Device Overview](#)).

If EREFSTEN is set and the ERCLKEN bit is written to 1, the external reference clock source (OSCOUT) keeps running during stop mode in order to provide a fast recovery upon exiting stop.

11.4.8 Fixed Frequency Clock

The ICS presents the divided FLL reference clock as ICSFFCLK for use as an additional clock source. ICSFFCLK frequency must be no more than 1/4 of the ICSOUT frequency to be valid.

11.4.9 Local Clock

The ICS presents the low range DCO output clock divided by two as ICSLCLK for use as a clock source for BDC communications. ICSLCLK is not available in FLL bypassed internal low power (FBILP) and FLL bypassed external low power (FBELP) modes.

Chapter 12

Modulo Timer (S08MTIMV1)

12.1 Introduction

The MTIM is a simple 8-bit timer with several software selectable clock sources and a programmable interrupt.

The central component of the MTIM is the 8-bit counter, which can operate as a free-running counter or a modulo counter. A timer overflow interrupt can be enabled to generate periodic interrupts for time-based software loops.

12.1.1 MTIM/TPM Configuration Information

The external clock for the MTIM module, TCLK, is selected by setting CLKS = 1:1 or 1:0 in MTIMCLK, which selects the TCLK pin input. The TCLK input on PTA5 can be enabled as external clock inputs to both the MTIM and TPM modules simultaneously.

12.1.2 MTIM Clock Gating

The bus clock to the MTIM can be gated on and off using the MTIM bit in SCGC1. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the MTIM bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, Peripheral Clock Gating](#) for details.

12.1.3 Features

Timer system features include:

- 8-bit up-counter
 - Free-running or 8-bit modulo limit
 - Software controllable interrupt on overflow
 - Counter reset bit (TRST)
 - Counter stop bit (TSTP)
- Four software selectable clock sources for input to prescaler:
 - System bus clock — rising edge
 - Fixed frequency clock (XCLK) — rising edge
 - External clock source on the TCLK pin — rising edge
 - External clock source on the TCLK pin — falling edge
- Nine selectable clock prescale values:
 - Clock source divide by 1, 2, 4, 8, 16, 32, 64, 128, or 256

12.1.4 Modes of Operation

This section defines the MTIM's operation in stop, wait and background debug modes.

12.1.4.1 MTIM in Wait Mode

The MTIM continues to run in wait mode if enabled before executing the WAIT instruction. Therefore, the MTIM can be used to bring the MCU out of wait mode if the timer overflow interrupt is enabled. For lowest possible current consumption, the MTIM should be stopped by software if not needed as an interrupt source during wait mode.

12.1.4.2 MTIM in Stop Modes

The MTIM is disabled in all stop modes, regardless of the settings before executing the STOP instruction. Therefore, the MTIM cannot be used as a wakeup source from stop modes.

Waking from stop1 and stop2 modes, the MTIM will be put into its reset state. If stop3 is exited with a reset, the MTIM will be put into its reset state. If stop3 is exited with an interrupt, the MTIM continues from the state it was in when stop3 was entered. If the counter was active upon entering stop3, the count will resume from the current value.

12.1.4.3 MTIM in Active Background Mode

The MTIM suspends all counting until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as an MTIM reset did not occur (TRST written to a 1 or MTIMMOD written).

12.1.5 Block Diagram

The block diagram for the modulo timer module is shown [Figure 12-1](#).

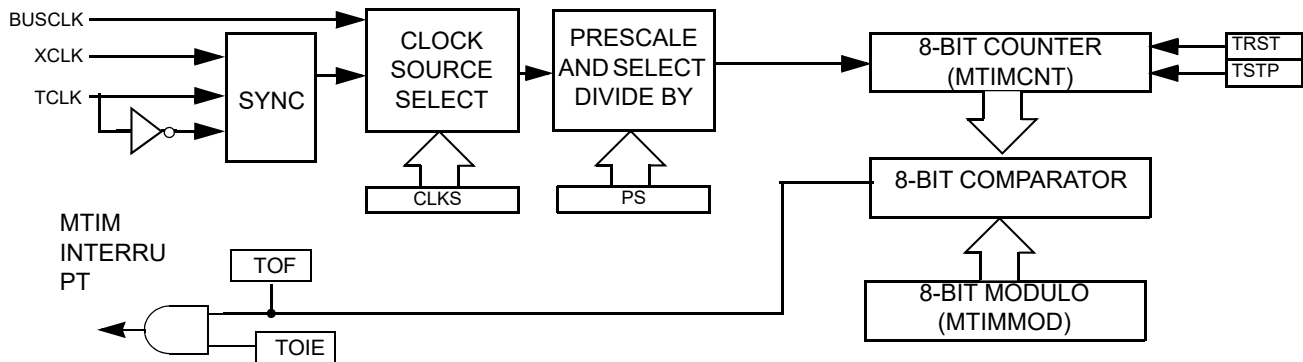


Figure 12-1. Modulo Timer (MTIM) Block Diagram

12.2 External Signal Description

The MTIM includes one external signal, TCLK, used to input an external clock when selected as the MTIM clock source. The signal properties of TCLK are shown in [Table 12-1](#).

Table 12-1. Signal Properties

Signal	Function	I/O
TCLK	External clock source input into MTIM	I

The TCLK input must be synchronized by the bus clock. Also, variations in duty cycle and clock jitter must be accommodated. Therefore, the TCLK signal must be limited to one-fourth of the bus frequency.

The TCLK pin can be muxed with a general-purpose port pin. See [Chapter 2, Pins and Connections](#) for the pin location and priority of this function.

12.3 Register Definition

[Figure 12-2](#) is a summary of MTIM registers.

Figure 12-2. MTIM Register Summary

Name		7	6	5	4	3	2	1	0
MTIMSC	R	TOF	TOIE	0	TSTP	0	0	0	0
	W			TRST					
MTIMCLK	R	0	0	CLKS		PS			
	W								
MTIMCNT	R	COUNT							
	W								
MTIMMOD	R	MOD							
	W								

Each MTIM includes four registers:

- An 8-bit status and control register
- An 8-bit clock configuration register
- An 8-bit counter register
- An 8-bit modulo register

Refer to the direct-page register summary in the [Memory](#) chapter of this data sheet for the absolute address assignments for all MTIM registers. This section refers to registers and control bits only by their names and relative address offsets.

Some MCUs may have more than one MTIM, so register names include placeholder characters to identify which MTIM is being referenced.

12.3.1 MTIM Status and Control Register (MTIMSC)

MTIMSC contains the overflow status flag and control bits which are used to configure the interrupt enable, reset the counter, and stop the counter.

	7	6	5	4	3	2	1	0
R	TOF	TOIE	0	TSTP	0	0	0	0
W			TRST					
Reset:	0	0	0	1	0	0	0	0

Figure 12-3. MTIM Status and Control Register

Table 12-2. MTIM Status and Control Register Field Descriptions

Field	Description
7 TOF	MTIM Overflow Flag — This read-only bit is set when the MTIM counter register overflows to 0x00 after reaching the value in the MTIM modulo register. Clear TOF by reading the MTIMSC register while TOF is set, then writing a 0 to TOF. TOF is also cleared when TRST is written to a 1 or when any value is written to the MTIMMOD register. 0 MTIM counter has not reached the overflow value in the MTIM modulo register. 1 MTIM counter has reached the overflow value in the MTIM modulo register.
6 TOIE	MTIM Overflow Interrupt Enable — This read/write bit enables MTIM overflow interrupts. If TOIE is set, then an interrupt is generated when TOF = 1. Reset clears TOIE. Do not set TOIE if TOF = 1. Clear TOF first, then set TOIE. 0 TOF interrupts are disabled. Use software polling. 1 TOF interrupts are enabled.
5 TRST	MTIM Counter Reset — When a 1 is written to this write-only bit, the MTIM counter register resets to 0x00 and TOF is cleared. Reading this bit always returns 0. 0 No effect. MTIM counter remains at current state. 1 MTIM counter is reset to 0x00.
4 TSTP	MTIM Counter Stop — When set, this read/write bit stops the MTIM counter at its current value. Counting resumes from the current value when TSTP is cleared. Reset sets TSTP to prevent the MTIM from counting. 0 MTIM counter is active. 1 MTIM counter is stopped.
3:0	Unused register bits, always read 0.

12.3.2 MTIM Clock Configuration Register (MTIMCLK)

MTIMCLK contains the clock select bits (CLKS) and the prescaler select bits (PS).

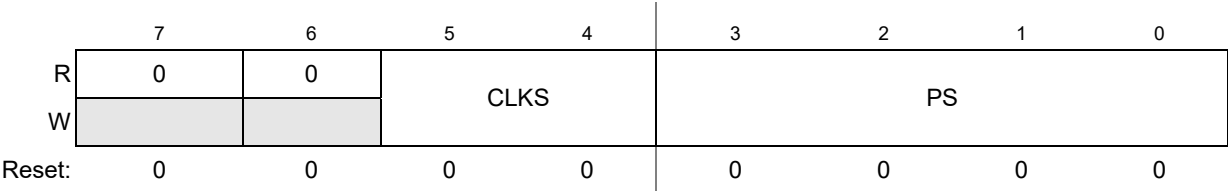


Figure 12-4. MTIM Clock Configuration Register

Table 12-3. MTIM Clock Configuration Register Field Description

Field	Description
7:6	Unused register bits, always read 0.
5:4 CLKS	Clock Source Select — These two read/write bits select one of four different clock sources as the input to the MTIM prescaler. Changing the clock source while the counter is active does not clear the counter. The count continues with the new clock source. Reset clears CLKS to 000. 00 Encoding 0. Bus clock (BUSCLK) 01 Encoding 1. Fixed-frequency clock (XCLK) 10 Encoding 3. External source (TCLK pin), falling edge 11 Encoding 4. External source (TCLK pin), rising edge All other encodings default to the bus clock (BUSCLK).
3:0 PS	Clock Source Prescaler — These four read/write bits select one of nine outputs from the 8-bit prescaler. Changing the prescaler value while the counter is active does not clear the counter. The count continues with the new prescaler value. Reset clears PS to 0000. 0000 Encoding 0. MTIM clock source ÷ 1 0001 Encoding 1. MTIM clock source ÷ 2 0010 Encoding 2. MTIM clock source ÷ 4 0011 Encoding 3. MTIM clock source ÷ 8 0100 Encoding 4. MTIM clock source ÷ 16 0101 Encoding 5. MTIM clock source ÷ 32 0110 Encoding 6. MTIM clock source ÷ 64 0111 Encoding 7. MTIM clock source ÷ 128 1000 Encoding 8. MTIM clock source ÷ 256 All other encodings default to MTIM clock source ÷ 256.

12.3.3 MTIM Counter Register (MTIMCNT)

MTIMCNT is the read-only value of the current MTIM count of the 8-bit counter.

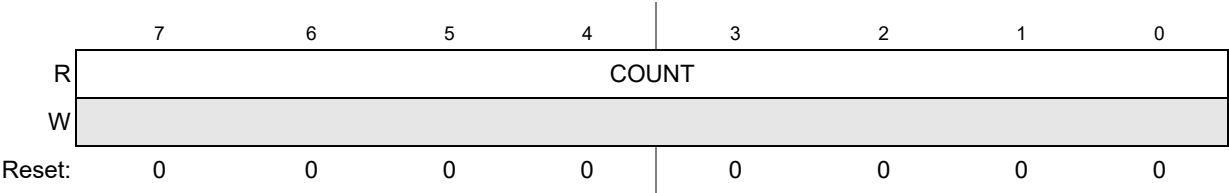


Figure 12-5. MTIM Counter Register

Table 12-4. MTIM Counter Register Field Description

Field	Description
7:0 COUNT	MTIM Count — These eight read-only bits contain the current value of the 8-bit counter. Writes have no effect to this register. Reset clears the count to 0x00.

12.3.4 MTIM Modulo Register (MTIMMOD)

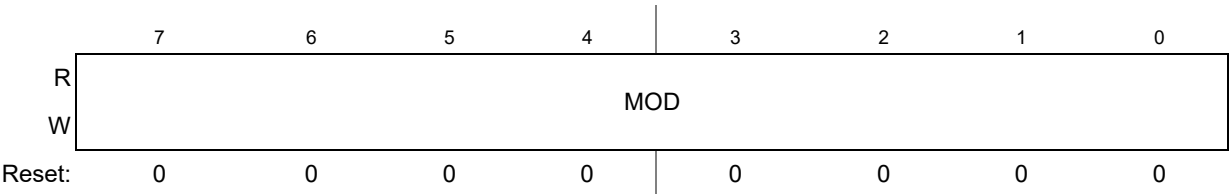


Figure 12-6. MTIM Modulo Register

Table 12-5. MTIM Modulo Register Field Descriptions

Field	Description
7:0 MOD	MTIM Modulo — These eight read/write bits contain the modulo value used to reset the count and set TOF. A value of 0x00 puts the MTIM in free-running mode. Writing to MTIMMOD resets the COUNT to 0x00 and clears TOF. Reset sets the modulo to 0x00.

12.4 Functional Description

The MTIM is composed of a main 8-bit up-counter with an 8-bit modulo register, a clock source selector, and a prescaler block with nine selectable values. The module also contains software selectable interrupt logic.

The MTIM counter (MTIMCNT) has three modes of operation: stopped, free-running, and modulo. Out of reset, the counter is stopped. If the counter is started without writing a new value to the modulo register, then the counter will be in free-running mode. The counter is in modulo mode when a value other than 0x00 is in the modulo register while the counter is running.

After any MCU reset, the counter is stopped and reset to 0x00, and the modulus is set to 0x00. The bus clock is selected as the default clock source and the prescale value is divide by 1. To start the MTIM in free-running mode, simply write to the MTIM status and control register (MTIMSC) and clear the MTIM stop bit (TSTP).

Four clock sources are software selectable: the internal bus clock, the fixed frequency clock (XCLK), and an external clock on the TCLK pin, selectable as incrementing on either rising or falling edges. The MTIM clock select bits (CLKS1:CLKS0) in MTIMSC are used to select the desired clock source. If the counter is active (TSTP = 0) when a new clock source is selected, the counter will continue counting from the previous value using the new clock source.

Nine prescale values are software selectable: clock source divided by 1, 2, 4, 8, 16, 32, 64, 128, or 256. The prescaler select bits (PS[3:0]) in MTIMSC select the desired prescale value. If the counter is active (TSTP = 0) when a new prescaler value is selected, the counter will continue counting from the previous value using the new prescaler value.

The MTIM modulo register (MTIMMOD) allows the overflow compare value to be set to any value from 0x01 to 0xFF. Reset clears the modulo value to 0x00, which results in a free running counter.

When the counter is active (TSTP = 0), the counter increments at the selected rate until the count matches the modulo value. When these values match, the counter overflows to 0x00 and continues counting. The MTIM overflow flag (TOF) is set whenever the counter overflows. The flag sets on the transition from the modulo value to 0x00. Writing to MTIMMOD while the counter is active resets the counter to 0x00 and clears TOF.

Clearing TOF is a two-step process. The first step is to read the MTIMSC register while TOF is set. The second step is to write a 0 to TOF. If another overflow occurs between the first and second steps, the clearing process is reset and TOF will remain set after the second step is performed. This will prevent the second occurrence from being missed. TOF is also cleared when a 1 is written to TRST or when any value is written to the MTIMMOD register.

The MTIM allows for an optional interrupt to be generated whenever TOF is set. To enable the MTIM overflow interrupt, set the MTIM overflow interrupt enable bit (TOIE) in MTIMSC. TOIE should never be written to a 1 while TOF = 1. Instead, TOF should be cleared first, then the TOIE can be set to 1.

12.4.1 MTIM Operation Example

This section shows an example of the MTIM operation as the counter reaches a matching value from the modulo register.

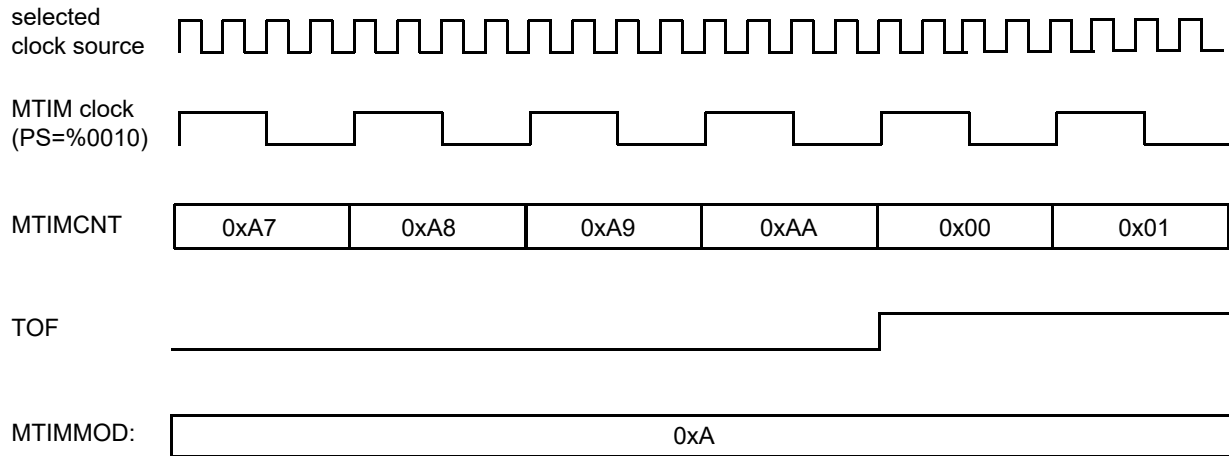


Figure 12-7. MTIM counter overflow example

In the example of [Figure 12-7](#), the selected clock source could be any of the five possible choices. The prescaler is set to PS = %0010 or divide-by-4. The modulo value in the MTIMMOD register is set to 0xAA. When the counter, MTIMCNT, reaches the modulo value of 0xAA, the counter overflows to 0x00 and continues counting. The timer overflow flag, TOF, sets when the counter value changes from 0xAA to 0x00. An MTIM overflow interrupt is generated when TOF is set, if TOIE = 1.

Chapter 13

Real-Time Counter (S08RTCV1)

13.1 Introduction

The real-time counter (RTC) consists of one 8-bit counter, one 8-bit comparator, several binary-based and decimal-based prescaler dividers, two clock sources, and one programmable periodic interrupt. This module can be used for time-of-day, calendar or any task scheduling functions. It can also serve as a cyclic wakeup from low power modes without the need of external components.

13.1.1 ADC Hardware Trigger

This RTC can be enabled as a hardware trigger for the ADC module by setting the ADTRG bit in the ADCSC2 register. When enabled, the ADC will be triggered every time RTCINT matches RTCMOD. The RTC interrupt does not have to be enabled to trigger the ADC.

13.1.2 RTC Clock Sources

The RTC module on MC9S08QL8 series can be clocked from the ICSIRCLK, OSCOUT or the LPO. ICSIRCLK is connected to the IRCLK input and OSCOUT is connected to ERCLK input.

ICSERCLK is not available as a source to the RTC in this MCU.

13.1.3 RTC Modes of Operation

All clock sources are available in all modes except stop2. The OSCOUT and LPO can be enabled as the clock source of the RTC in stop2.

13.1.4 RTC Status after Stop2 Wakeup

The registers associated with the RTC will be unaffected after a stop2 wakeup.

13.1.5 RTC Clock Gating

The bus clock to the RTC can be gated on and off using the RTC bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the RTC bit can be cleared to disable the clock to this module when not in use. If the bus clock is not used as the RTC clock source, the bus clock must still be enabled to the RTC for proper operation. Interrupts from the RTC will not occur if the bus clock is gated off. See [Section 5.7, Peripheral Clock Gating](#) for details.

13.1.6 Features

Features of the RTC module include:

- 8-bit up-counter
 - 8-bit modulo match limit
 - Software controllable periodic interrupt on match
- Three software selectable clock sources for input to prescaler with selectable binary-based and decimal-based divider values
 - 1-kHz internal low-power oscillator (LPO)
 - External clock (ERCLK)
 - 32-kHz internal clock (IRCLK)

13.1.7 Modes of Operation

This section defines the operation in stop, wait and background debug modes.

13.1.7.1 Wait Mode

The RTC continues to run in wait mode if enabled before executing the appropriate instruction. Therefore, the RTC can bring the MCU out of wait mode if the real-time interrupt is enabled. For lowest possible current consumption, the RTC should be stopped by software if not needed as an interrupt source during wait mode.

13.1.7.2 Stop Modes

The RTC continues to run in stop2 or stop3 mode if the RTC is enabled before executing the STOP instruction. Therefore, the RTC can bring the MCU out of stop modes with no external components, if the real-time interrupt is enabled.

The LPO clock can be used in stop2 and stop3 modes. ERCLK and IRCLK clocks are only available in stop3 mode.

Power consumption is lower when all clock sources are disabled, but in that case, the real-time interrupt cannot wake up the MCU from stop modes.

13.1.7.3 Active Background Mode

The RTC suspends all counting during active background mode until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as the RTCMOD register is not written and the RTCPS and RTCLKS bits are not altered.

13.1.8 Block Diagram

The block diagram for the RTC module is shown in [Figure 13-1](#).

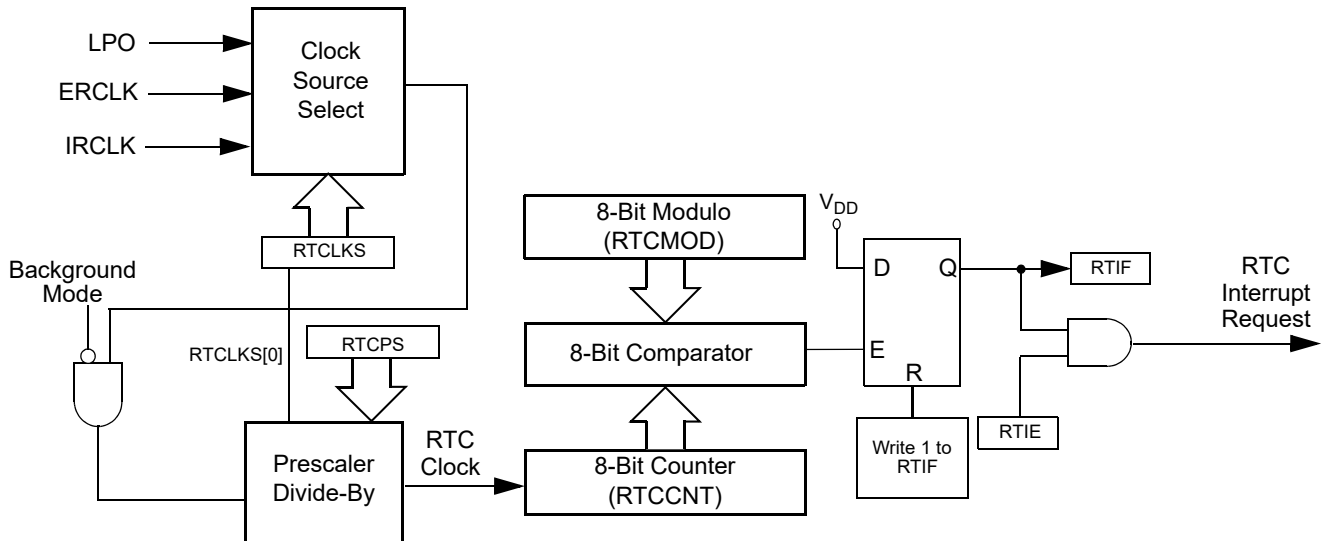


Figure 13-1. Real-Time Counter (RTC) Block Diagram

13.2 External Signal Description

The RTC does not include any off-chip signals.

13.3 Register Definition

The RTC includes a status and control register, an 8-bit counter register, and an 8-bit modulo register.

Refer to the direct-page register summary in the memory section of this document for the absolute address assignments for all RTC registers. This section refers to registers and control bits only by their names and relative address offsets.

[Table 13-1](#) is a summary of RTC registers.

Table 13-1. RTC Register Summary

Name		7	6	5	4	3	2	1	0
RTCSC	R	RTIF	RTCLKS		RTIE	RTCPS			
	W								
RTCCNT	R	RTCCNT							
	W								
RTCMOD	R	RTCMOD							
	W								

13.3.1 RTC Status and Control Register (RTCSC)

RTCSC contains the real-time interrupt status flag (RTIF), the clock select bits (RTCLKS), the real-time interrupt enable bit (RTIE), and the prescaler select bits (RTCPS).

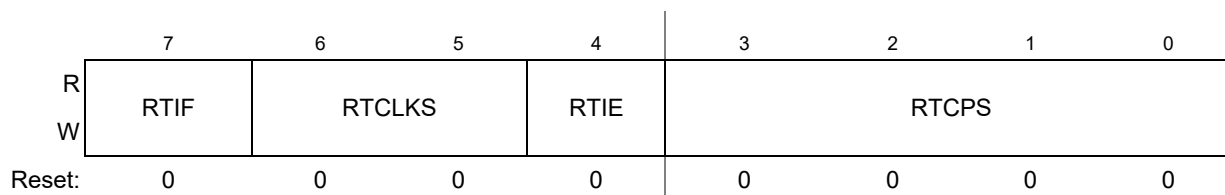


Figure 13-2. RTC Status and Control Register (RTCSC)

Table 13-2. RTCSC Field Descriptions

Field	Description
7 RTIF	Real-Time Interrupt Flag This status bit indicates the RTC counter register reached the value in the RTC modulo register. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the real-time interrupt request. Reset clears RTIF. 0 RTC counter has not reached the value in the RTC modulo register. 1 RTC counter has reached the value in the RTC modulo register.
6–5 RTCLKS	Real-Time Clock Source Select. These two read/write bits select the clock source input to the RTC prescaler. Changing the clock source clears the prescaler and RTCCNT counters. When selecting a clock source, ensure that the clock source is properly enabled (if applicable) to ensure correct operation of the RTC. Reset clears RTCLKS. 00 Real-time clock source is the 1-kHz low power oscillator (LPO) 01 Real-time clock source is the external clock (ERCLK) 1x Real-time clock source is the internal clock (IRCLK)
4 RTIE	Real-Time Interrupt Enable. This read/write bit enables real-time interrupts. If RTIE is set, then an interrupt is generated when RTIF is set. Reset clears RTIE. 0 Real-time interrupt requests are disabled. Use software polling. 1 Real-time interrupt requests are enabled.
3–0 RTCPS	Real-Time Clock Prescaler Select. These four read/write bits select binary-based or decimal-based divide-by values for the clock source. See Table 13-3 . Changing the prescaler value clears the prescaler and RTCCNT counters. Reset clears RTCPS.

Table 13-3. RTC Prescaler Divide-by values

RTCLKS[0]	RTCPS															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Off	2 ³	2 ⁵	2 ⁶	2 ⁷	2 ⁸	2 ⁹	2 ¹⁰	1	2	2 ²	10	2 ⁴	10 ²	5x10 ²	10 ³
1	Off	2 ¹⁰	2 ¹¹	2 ¹²	2 ¹³	2 ¹⁴	2 ¹⁵	2 ¹⁶	10 ³	2x10 ³	5x10 ³	10 ⁴	2x10 ⁴	5x10 ⁴	10 ⁵	2x10 ⁵

13.3.2 RTC Counter Register (RTCCNT)

RTCCNT is the read-only value of the current RTC count of the 8-bit counter.

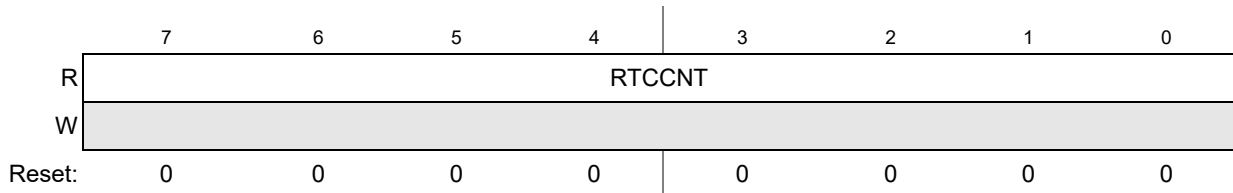


Figure 13-3. RTC Counter Register (RTCCNT)

Table 13-4. RTCCNT Field Descriptions

Field	Description
7:0 RTCCNT	RTC Count. These eight read-only bits contain the current value of the 8-bit counter. Writes have no effect to this register. Reset, writing to RTCMOD, or writing different values to RTCLKS and RTCPS clear the count to 0x00.

13.3.3 RTC Modulo Register (RTCMOD)

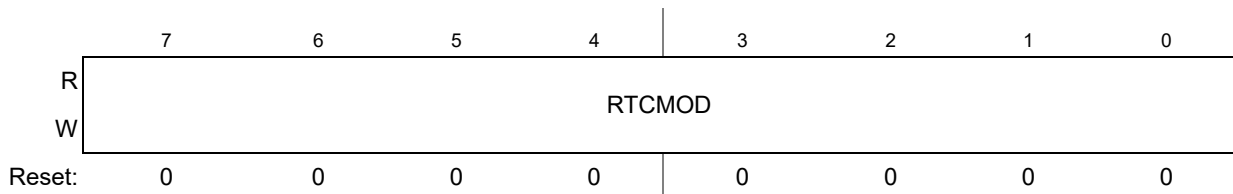


Figure 13-4. RTC Modulo Register (RTCMOD)

Table 13-5. RTCMOD Field Descriptions

Field	Description
7:0 RTCMOD	RTC Modulo. These eight read/write bits contain the modulo value used to reset the count to 0x00 upon a compare match and set the RTIF status bit. A value of 0x00 sets the RTIF bit on each rising edge of the prescaler output. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x00. Reset sets the modulo to 0x00.

13.4 Functional Description

The RTC is composed of a main 8-bit up-counter with an 8-bit modulo register, a clock source selector, and a prescaler block with binary-based and decimal-based selectable values. The module also contains software selectable interrupt logic.

After any MCU reset, the counter is stopped and reset to 0x00, the modulus register is set to 0x00, and the prescaler is off. The 1-kHz internal oscillator clock is selected as the default clock source. To start the prescaler, write any value other than zero to the prescaler select bits (RTCPS).

Three clock sources are software selectable: the low power oscillator clock (LPO), the external clock (ERCLK), and the internal clock (IRCLK). The RTC clock select bits (RTCLKS) select the desired clock source. If a different value is written to RTCLKS, the prescaler and RTCCNT counters are reset to 0x00.

RTCPS and the RTCLKS[0] bit select the desired divide-by value. If a different value is written to RTCPS, the prescaler and RTCCNT counters are reset to 0x00. Table 13-6 shows different prescaler period values.

Table 13-6. Prescaler Period

RTCPS	1-kHz Internal Clock (RTCLKS = 00)	1-MHz External Clock (RTCLKS = 01)	32-kHz Internal Clock (RTCLKS = 10)	32-kHz Internal Clock (RTCLKS = 11)
0000	Off	Off	Off	Off
0001	8 ms	1.024 ms	250 μ s	32 ms
0010	32 ms	2.048 ms	1 ms	64 ms
0011	64 ms	4.096 ms	2 ms	128 ms
0100	128 ms	8.192 ms	4 ms	256 ms
0101	256 ms	16.4 ms	8 ms	512 ms
0110	512 ms	32.8 ms	16 ms	1.024 s
0111	1.024 s	65.5 ms	32 ms	2.048 s
1000	1 ms	1 ms	31.25 μ s	31.25 ms
1001	2 ms	2 ms	62.5 μ s	62.5 ms
1010	4 ms	5 ms	125 μ s	156.25 ms
1011	10 ms	10 ms	312.5 μ s	312.5 ms
1100	16 ms	20 ms	0.5 ms	0.625 s
1101	0.1 s	50 ms	3.125 ms	1.5625 s
1110	0.5 s	0.1 s	15.625 ms	3.125 s
1111	1 s	0.2 s	31.25 ms	6.25 s

The RTC modulo register (RTCMOD) allows the compare value to be set to any value from 0x00 to 0xFF. When the counter is active, the counter increments at the selected rate until the count matches the modulo value. When these values match, the counter resets to 0x00 and continues counting. The real-time interrupt flag (RTIF) is set when a match occurs. The flag sets on the transition from the modulo value to 0x00. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x00.

The RTC allows for an interrupt to be generated when RTIF is set. To enable the real-time interrupt, set the real-time interrupt enable bit (RTIE) in RTCSC. RTIF is cleared by writing a 1 to RTIF.

13.4.1 RTC Operation Example

This section shows an example of the RTC operation as the counter reaches a matching value from the modulo register.

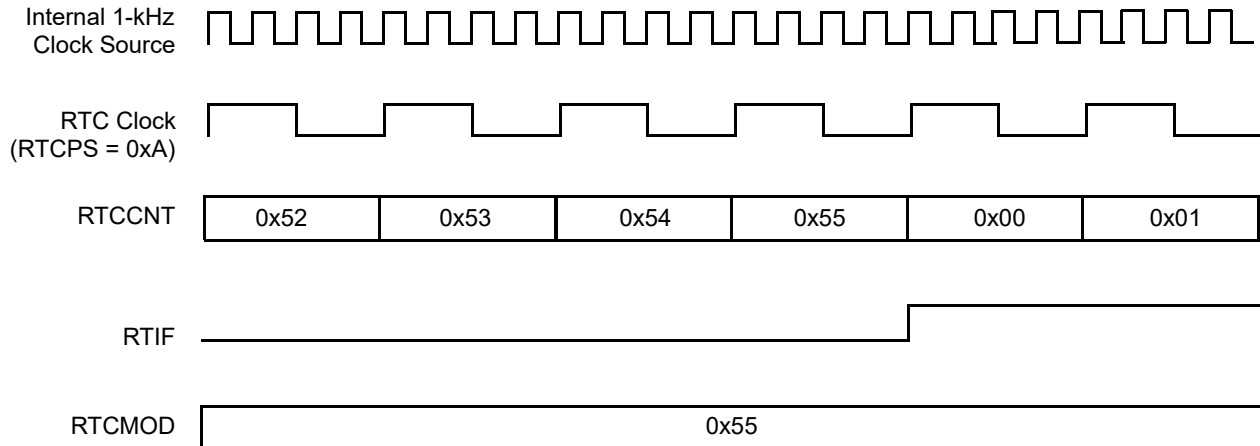


Figure 13-5. RTC Counter Overflow Example

In the example of [Figure 13-5](#), the selected clock source is the 1-kHz internal oscillator clock source. The prescaler (RTCPs) is set to 0xA or divide-by-4. The modulo value in the RTCMOD register is set to 0x55. When the counter, RTCCNT, reaches the modulo value of 0x55, the counter overflows to 0x00 and continues counting. The real-time interrupt flag, RTIF, sets when the counter value changes from 0x55 to 0x00. A real-time interrupt is generated when RTIF is set, if RTIE is set.

13.5 Initialization/Application Information

This section provides example code to give some basic direction to a user on how to initialize and configure the RTC module. The example software is implemented in C language.

The example below shows how to implement time of day with the RTC using the 1-kHz clock source to achieve the lowest possible power consumption. Because the 1-kHz clock source is not as accurate as a crystal, software can be added for any adjustments. For accuracy without adjustments at the expense of additional power consumption, the external clock (ERCLK) or the internal clock (IRCLK) can be selected with appropriate prescaler and modulo values.

```
/* Initialize the elapsed time counters */
Seconds = 0;
Minutes = 0;
Hours = 0;
Days=0;

/* Configure RTC to interrupt every 1 second from 1-kHz clock source */
RTCMOD.byte = 0x00;
RTCSC.byte = 0x1F;

/*****
Function Name : RTC_ISR
Notes : Interrupt service routine for RTC module.
*****/
```

```
#pragma TRAP_PROC
void RTC_ISR(void)
{
    /* Clear the interrupt flag */
    RTCSC.byte = RTCSC.byte | 0x80;
    /* RTC interrupts every 1 Second */
    Seconds++;
    /* 60 seconds in a minute */
    if (Seconds > 59){
        Minutes++;
        Seconds = 0;
    }
    /* 60 minutes in an hour */
    if (Minutes > 59){
        Hours++;
        Minutes = 0;
    }
    /* 24 hours in a day */
    if (Hours > 23){
        Days ++;
        Hours = 0;
    }
}
```

Chapter 14

Serial Communications Interface (S08SCIV4)

14.1 Introduction

NOTE

Stop1 mode is not available in this device.

14.1.1 SCI Clock Gating

The bus clock to the SCI can be gated on and off using the SCI bit in SCGC1. This bit is set after any reset, which enables the bus clock to this module. To conserve power, this bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, Peripheral Clock Gating](#) for details.

Module Initialization:

Write: SCIBDH:SCIBDL to set baud rate

Write: SCFC1 to configure 1-wire/2-wire, 9/8-bit data, wakeup, and parity, if used.

Write: SCIC2 to configure interrupts, enable Rx and Tx, RWU
Enable Rx wakeup, SBK sends break character

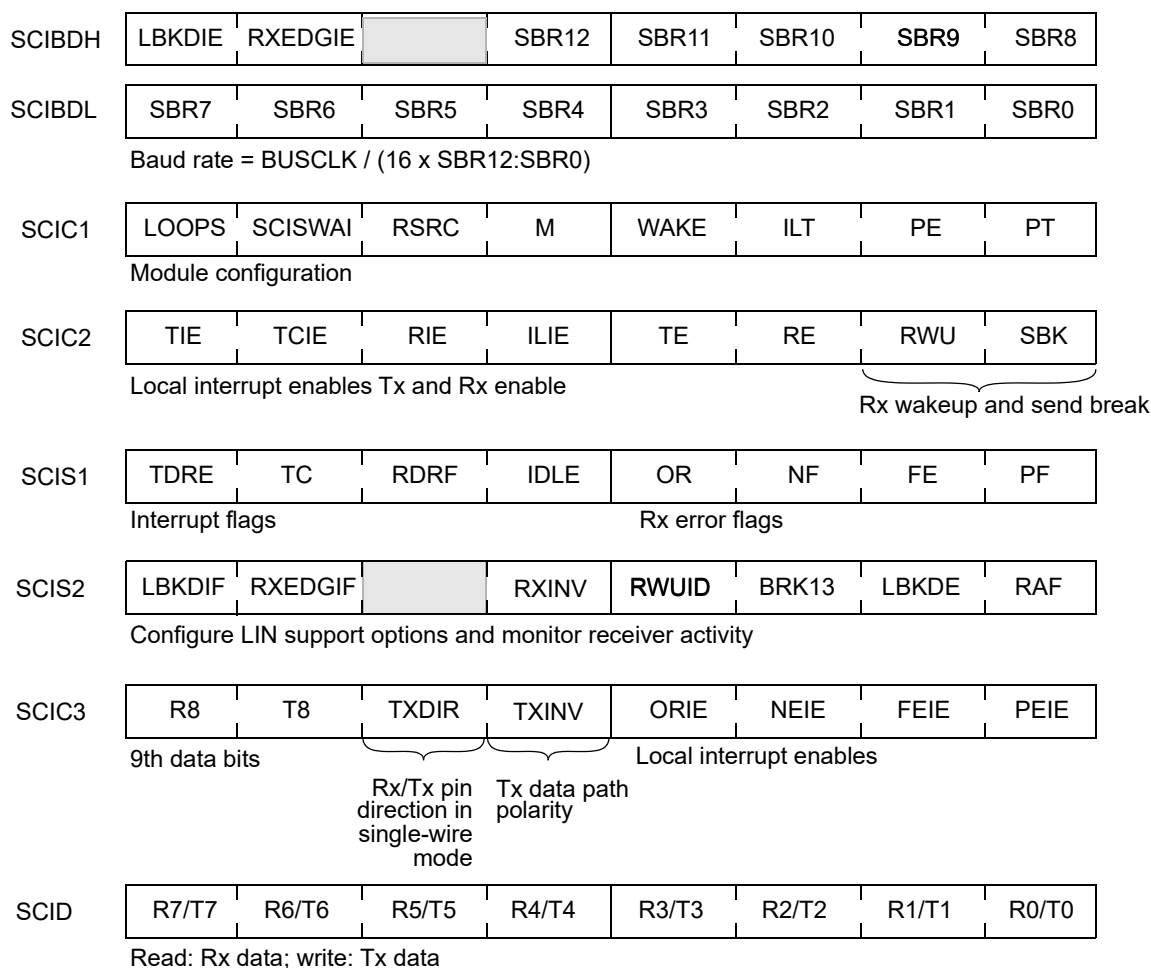
Write: SCIC3 to enable Rx error interrupt sources. Also controls pin direction in 1-wire modes. R8 and T8 only used in 9-bit data modes.

Module Use:

Wait for TDRE, then write data to SCID

Wait for RDRF, then read data from SCID

A small number of applications will use RWU to manage automatic receiver wakeup, SBK to send break characters, and R8 and T8 for 9-bit data.

**Figure 14-1. SCI Module Quick Start**

14.1.2 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
 - Transmit data register empty and transmission complete
 - Receive data register full
 - Receive overrun, parity error, framing error, and noise error
 - Idle receiver detect
 - Active edge on receive pin
 - Break detect supporting LIN
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark
- Optional 13-bit break character generation / 11-bit break character detection
- Independently selectable polarity for transmitter output and receiver input

14.1.3 Modes of Operation

See [Section 14.3, Functional Description](#) for details concerning SCI operation in these modes:

- 8- and 9-bit data modes
- Stop mode operation
- Loop mode
- Single-wire mode

14.1.4 Block Diagram

[Figure 14-2](#) shows the transmitter portion of the SCI.

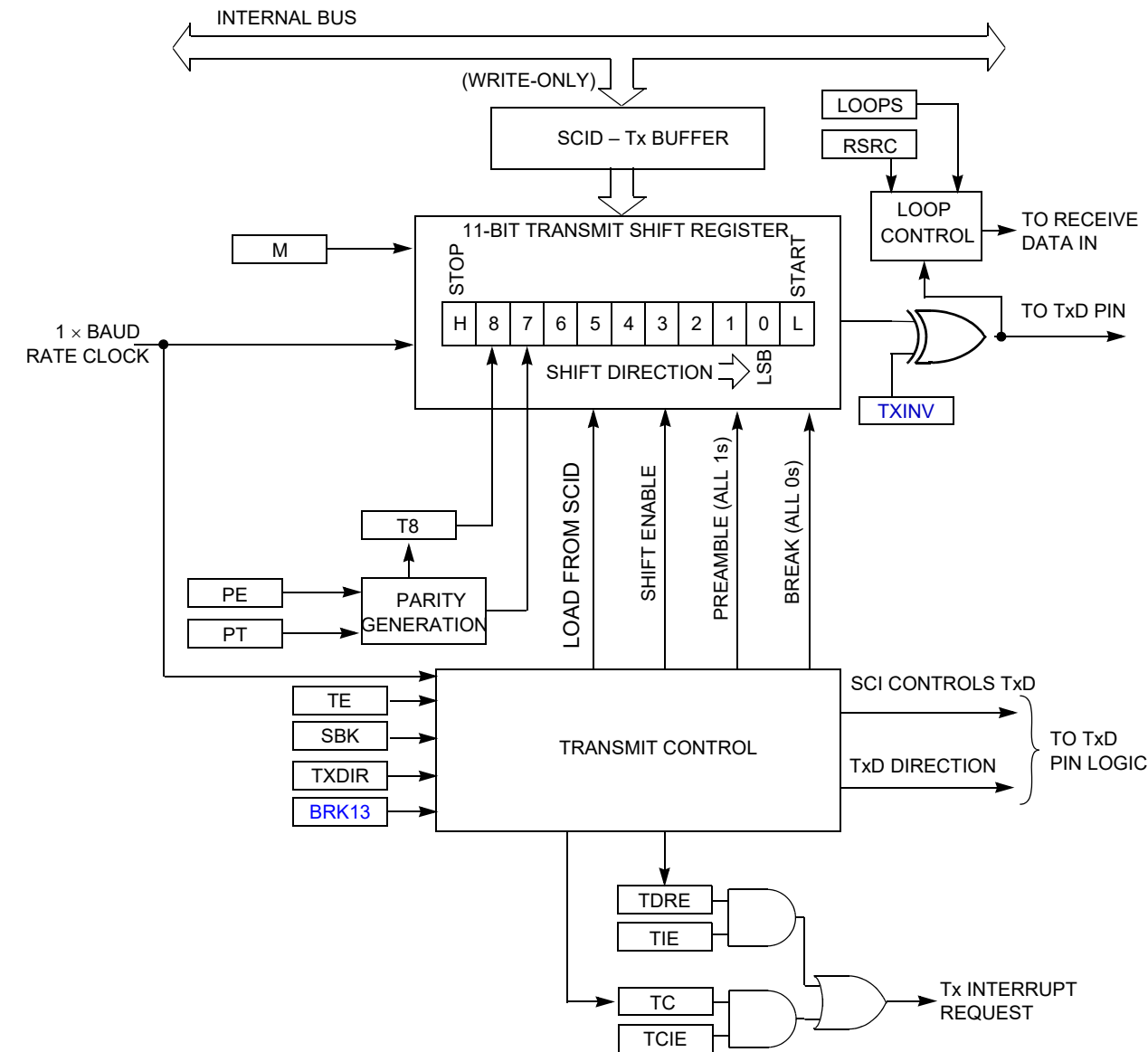


Figure 14-2. SCI Transmitter Block Diagram

Figure 14-3 shows the receiver portion of the SCI.

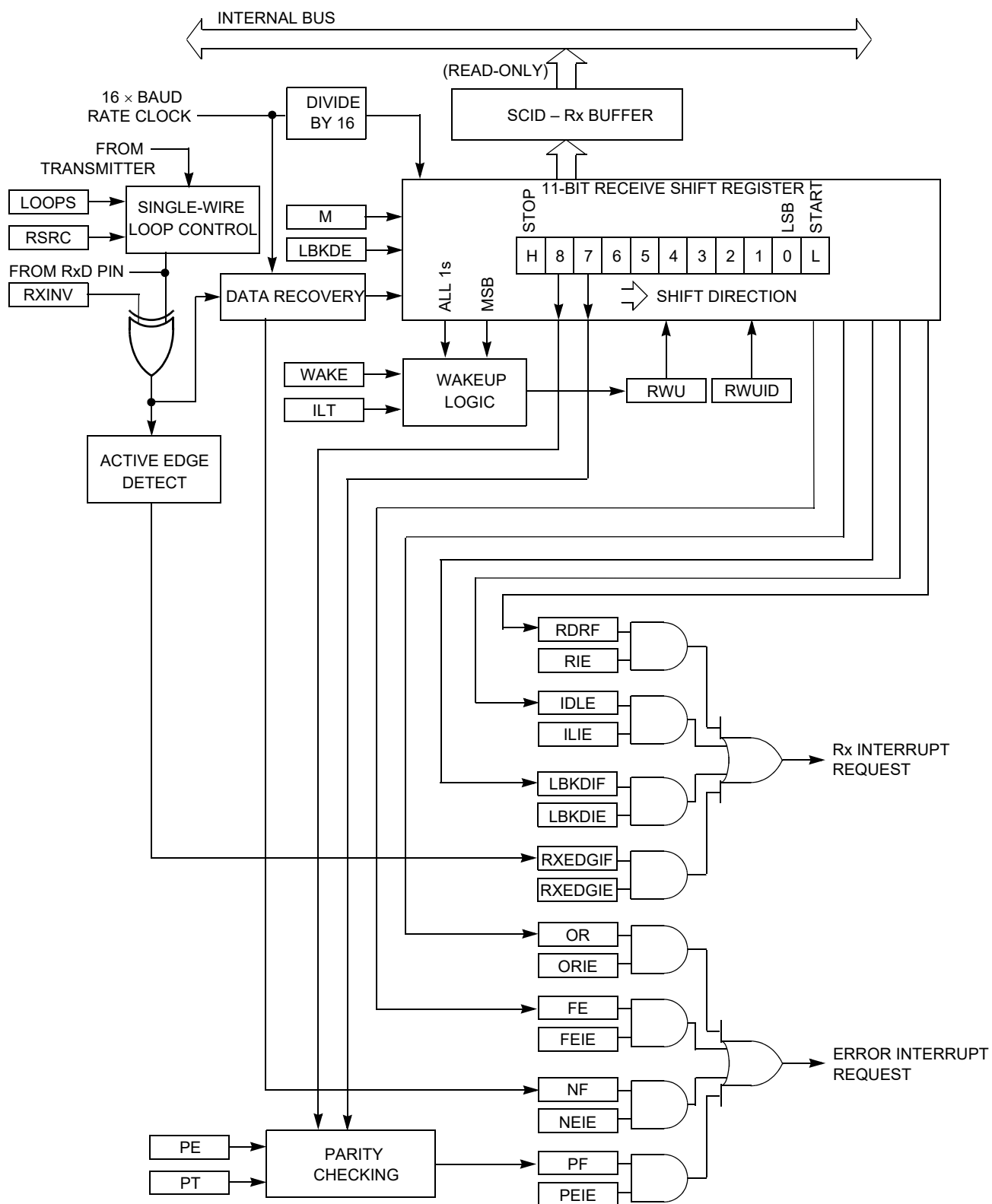


Figure 14-3. SCI Receiver Block Diagram

14.2 Register Definition

The SCI has eight 8-bit registers to control baud rate, select SCI options, report SCI status, and for transmit/receive data.

Refer to the direct-page register summary in the [Memory](#) chapter of this data sheet for the absolute address assignments for all SCI registers. This section refers to registers and control bits only by their names. A NXP-provided equate or header file is used to translate these names into the appropriate absolute addresses.

14.2.1 SCI Baud Rate Registers (SCIBDH, SCIBDL)

This pair of registers controls the prescale divisor for SCI baud rate generation. To update the 13-bit baud rate setting [SBR12:SBR0], first write to SCIBDH to buffer the high half of the new value and then write to SCIBDL. The working value in SCIBDH does not change until SCIBDL is written.

SCIBDL is reset to a non-zero value, so after reset the baud rate generator remains disabled until the first time the receiver or transmitter is enabled (RE or TE bits in SCIC2 are written to 1).

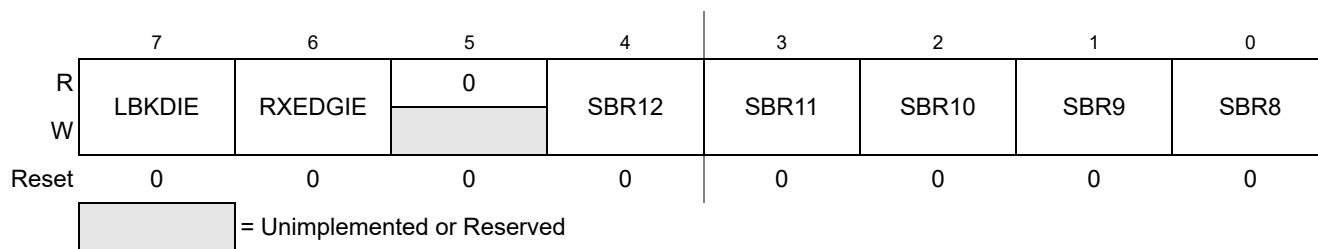


Figure 14-4. SCI Baud Rate Register (SCIBDH)

Table 14-1. SCIBDH Field Descriptions

Field	Description
7 LBKDIE	LIN Break Detect Interrupt Enable (for LBKDIF) 0 Hardware interrupts from LBKDIF disabled (use polling). 1 Hardware interrupt requested when LBKDIF flag is 1.
6 RXEDGIE	RxD Input Active Edge Interrupt Enable (for RXEDGIF) 0 Hardware interrupts from RXEDGIF disabled (use polling). 1 Hardware interrupt requested when RXEDGIF flag is 1.
4:0 SBR[12:8]	Baud Rate Modulo Divisor — The 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = $BUSCLK/(16 \times BR)$. See also BR bits in Table 14-2 .

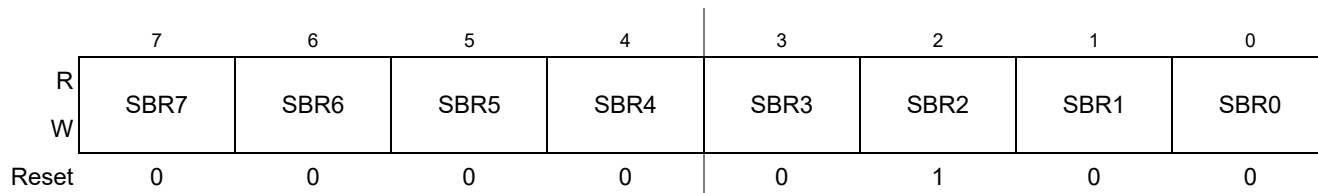


Figure 14-5. SCI Baud Rate Register (SCIBDL)

Table 14-2. SCIBDL Field Descriptions

Field	Description
7:0 SBR[7:0]	Baud Rate Modulo Divisor — These 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in Table 14-1 .

14.2.2 SCI Control Register 1 (SCIC1)

This read/write register is used to control various optional features of the SCI system.

	7	6	5	4	3	2	1	0
R	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
W								
Reset	0	0	0	0	0	0	0	0

Figure 14-6. SCI Control Register 1 (SCIC1)

Table 14-3. SCIC1 Field Descriptions

Field	Description
7 LOOPS	Loop Mode Select — Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS = 1, the transmitter output is internally connected to the receiver input. 0 Normal operation — RxD and TxD use separate pins. 1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See RSRC bit.) RxD pin is not used by SCI.
6 SCISWAI	SCI Stops in Wait Mode 0 SCI clocks continue to run in wait mode so the SCI can be the source of an interrupt that wakes up the CPU. 1 SCI clocks freeze while CPU is in wait mode.
5 RSRC	Receiver Source Select — This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS = 1, the receiver input is internally connected to the TxD pin and RSRC determines whether this connection is also connected to the transmitter output. 0 Provided LOOPS = 1, RSRC = 0 selects internal loop back mode and the SCI does not use the RxD pins. 1 Single-wire SCI mode where the TxD pin is connected to the transmitter output and receiver input.
4 M	9-Bit or 8-Bit Mode Select 0 Normal — start + 8 data bits (LSB first) + stop. 1 Receiver and transmitter use 9-bit data characters start + 8 data bits (LSB first) + 9th data bit + stop.
3 WAKE	Receiver Wakeup Method Select — Refer to Section 14.3.3.2, Receiver Wakeup Operation for more information. 0 Idle-line wakeup. 1 Address-mark wakeup.
2 ILT	Idle Line Type Select — Setting this bit to 1 ensures that the stop bit and logic 1 bits at the end of a character do not count toward the 10 or 11 bit times of logic high level needed by the idle line detection logic. Refer to Section 14.3.3.2.1, Idle-Line Wakeup for more information. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.

Table 14-3. SCIC1 Field Descriptions (continued)

Field	Description
1 PE	Parity Enable — Enables hardware parity generation and checking. When parity is enabled, the most significant bit (MSB) of the data character (eighth or ninth data bit) is treated as the parity bit. 0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	Parity Type — Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0 Even parity. 1 Odd parity.

14.2.3 SCI Control Register 2 (SCIC2)

This register can be read or written at any time.

	7	6	5	4	3	2	1	0
R	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
W								
Reset	0	0	0	0	0	0	0	0

Figure 14-7. SCI Control Register 2 (SCIC2)**Table 14-4. SCIC2 Field Descriptions**

Field	Description
7 TIE	Transmit Interrupt Enable (for TDRE) 0 Hardware interrupts from TDRE disabled (use polling). 1 Hardware interrupt requested when TDRE flag is 1.
6 TCIE	Transmission Complete Interrupt Enable (for TC) 0 Hardware interrupts from TC disabled (use polling). 1 Hardware interrupt requested when TC flag is 1.
5 RIE	Receiver Interrupt Enable (for RDRF) 0 Hardware interrupts from RDRF disabled (use polling). 1 Hardware interrupt requested when RDRF flag is 1.
4 ILIE	Idle Line Interrupt Enable (for IDLE) 0 Hardware interrupts from IDLE disabled (use polling). 1 Hardware interrupt requested when IDLE flag is 1.
3 TE	Transmitter Enable 0 Transmitter off. 1 Transmitter on. TE must be 1 in order to use the SCI transmitter. When TE = 1, the SCI forces the TxD pin to act as an output for the SCI system. When the SCI is configured for single-wire operation (LOOPS = RSRC = 1), TXDIR controls the direction of traffic on the single SCI communication line (TxD pin). TE also can be used to queue an idle character by writing TE = 0 then TE = 1 while a transmission is in progress. Refer to Section 14.3.2.1, Send Break and Queued Idle for more details. When TE is written to 0, the transmitter keeps control of the port TxD pin until any data, queued idle, or queued break character finishes transmitting before allowing the pin to revert to a general-purpose I/O pin.

Table 14-4. SCIC2 Field Descriptions (continued)

Field	Description
2 RE	Receiver Enable — When the SCI receiver is off, the RxD pin reverts to being a general-purpose port I/O pin. If LOOPS = 1 the RxD pin reverts to being a general-purpose I/O pin even if RE = 1. 0 Receiver off. 1 Receiver on.
1 RWU	Receiver Wakeup Control — This bit can be written to 1 to place the SCI receiver in a standby state where it waits for automatic hardware detection of a selected wakeup condition. The wakeup condition is either an idle line between messages (WAKE = 0, idle-line wakeup), or a logic 1 in the most significant data bit in a character (WAKE = 1, address-mark wakeup). Application software sets RWU and (normally) a selected hardware condition automatically clears RWU. Refer to Section 14.3.3.2, Receiver Wakeup Operation for more details. 0 Normal SCI receiver operation. 1 SCI receiver in standby waiting for wakeup condition.
0 SBK	Send Break — Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 (13 or 14 if BRK13 = 1) bit times of logic 0 are queued as long as SBK = 1. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. Refer to Section 14.3.2.1, Send Break and Queued Idle for more details. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

14.2.4 SCI Status Register 1 (SCIS1)

This register has eight read-only status flags. Writes have no effect. Special software sequences (which do not involve writing to this register) are used to clear these status flags.

	7	6	5	4	3	2	1	0
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W								
Reset	1	1	0	0	0	0	0	0

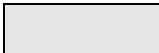
 = Unimplemented or Reserved

Figure 14-8. SCI Status Register 1 (SCIS1)

Table 14-5. SCIS1 Field Descriptions

Field	Description
7 TDRE	Transmit Data Register Empty Flag — TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SCIS1 with TDRE = 1 and then write to the SCI data register (SCID). 0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.
6 TC	Transmission Complete Flag — TC is set out of reset and when TDRE = 1 and no data, preamble, or break character is being transmitted. 0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete). TC is cleared automatically by reading SCIS1 with TC = 1 and then doing one of the following three things: • Write to the SCI data register (SCID) to transmit new data • Queue a preamble by changing TE from 0 to 1 • Queue a break character by writing 1 to SBK in SCIC2
5 RDRF	Receive Data Register Full Flag — RDRF becomes set when a character transfers from the receive shifter into the receive data register (SCID). To clear RDRF, read SCIS1 with RDRF = 1 and then read the SCI data register (SCID). 0 Receive data register empty. 1 Receive data register full.
4 IDLE	Idle Line Flag — IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT = 0, the receiver starts counting idle bit times after the start bit. So if the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or 11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When ILT = 1, the receiver doesn't start counting idle bit times until after the stop bit. So the stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line. To clear IDLE, read SCIS1 with IDLE = 1 and then read the SCI data register (SCID). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE will get set only once even if the receive line remains idle for an extended period. 0 No idle line detected. 1 Idle line was detected.
3 OR	Receiver Overrun Flag — OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from SCID yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCID. To clear OR, read SCIS1 with OR = 1 and then read the SCI data register (SCID). 0 No overrun. 1 Receive overrun (new SCI data lost).
2 NF	Noise Flag — The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF will be set at the same time as the flag RDRF gets set for the character. To clear NF, read SCIS1 and then read the SCI data register (SCID). 0 No noise detected. 1 Noise detected in the received character in SCID.

Table 14-5. SCIS1 Field Descriptions (continued)

Field	Description
1 FE	Framing Error Flag — FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read SCIS1 with FE = 1 and then read the SCI data register (SCID). 0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.
0 PF	Parity Error Flag — PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SCIS1 and then read the SCI data register (SCID). 0 No parity error. 1 Parity error.

14.2.5 SCI Status Register 2 (SCIS2)

This register has one read-only status flag.

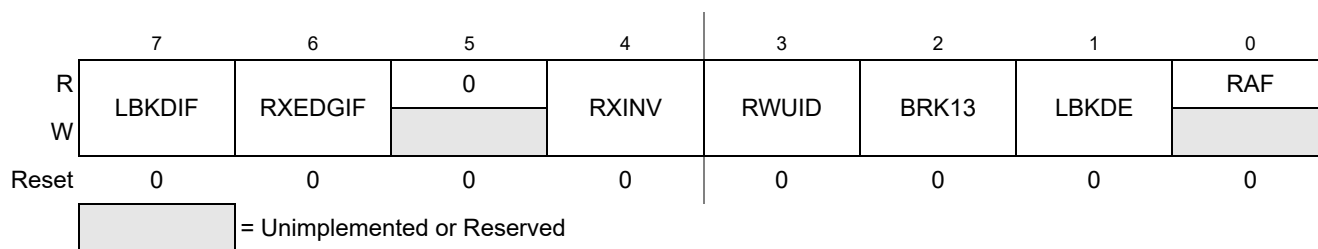


Figure 14-9. SCI Status Register 2 (SCIS2)

Table 14-6. SCIS2 Field Descriptions

Field	Description
7 LBKDIF	LIN Break Detect Interrupt Flag — LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a “1” to it. 0 No LIN break character has been detected. 1 LIN break character has been detected.
6 RXEDGIF	RxD Pin Active Edge Interrupt Flag — RXEDGIF is set when an active edge (falling if RXINV = 0, rising if RXINV=1) on the RxD pin occurs. RXEDGIF is cleared by writing a “1” to it. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
4 RXINV ¹	Receive Data Inversion — Setting this bit reverses the polarity of the received data input. 0 Receive data not inverted 1 Receive data inverted
3 RWUID	Receive Wake Up Idle Detect — RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. 0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. 1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character.
2 BRK13	Break Character Generation Length — BRK13 is used to select a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. 0 Break character is transmitted with length of 10 bit times (11 if M = 1) 1 Break character is transmitted with length of 13 bit times (14 if M = 1)

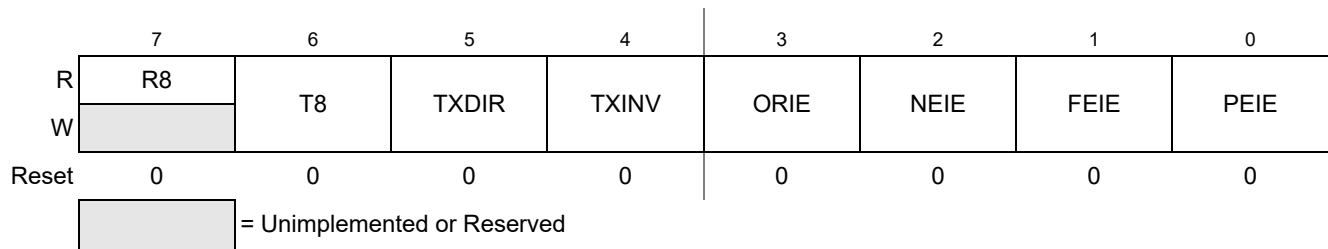
Table 14-6. SCIS2 Field Descriptions (continued)

Field	Description
1 LBKDE	LIN Break Detection Enable — LBKDE is used to select a longer break character detection length. While LBKDE is set, framing error (FE) and receive data register full (RDRF) flags are prevented from setting. 0 Break character is detected at length of 10 bit times (11 if M = 1). 1 Break character is detected at length of 11 bit times (12 if M = 1).
0 RAF	Receiver Active Flag — RAF is set when the SCI receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an SCI character is being received before instructing the MCU to go to stop mode. 0 SCI receiver idle waiting for a start bit. 1 SCI receiver active (RxD input not idle).

¹ Setting RXINV inverts the RxD input for all cases: data bits, start and stop bits, break, and idle.

When using an internal oscillator in a LIN system, it is necessary to raise the break detection threshold by one bit time. Under the worst case timing conditions allowed in LIN, it is possible that a 0x00 data character can appear to be 10.26 bit times long at a slave which is running 14% faster than the master. This would trigger normal break detection circuitry which is designed to detect a 10 bit break symbol. When the LBKDE bit is set, framing errors are inhibited and the break detection threshold changes from 10 bits to 11 bits, preventing false detection of a 0x00 data character as a LIN break symbol.

14.2.6 SCI Control Register 3 (SCIC3)

**Figure 14-10. SCI Control Register 3 (SCIC3)****Table 14-7. SCIC3 Field Descriptions**

Field	Description
7 R8	Ninth Data Bit for Receiver — When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the MSB of the buffered data in the SCID register. When reading 9-bit data, read R8 before reading SCID because reading SCID completes automatic flag clearing sequences which could allow R8 and SCID to be overwritten with new data.
6 T8	Ninth Data Bit for Transmitter — When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the MSB of the data in the SCID register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCID is written so T8 should be written (if it needs to change from its previous value) before SCID is written. If T8 does not need to change in the new value (such as when it is used to generate mark or space parity), it need not be written each time SCID is written.
5 TXDIR	TxD Pin Direction in Single-Wire Mode — When the SCI is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TxD pin. 0 TxD pin is an input in single-wire mode. 1 TxD pin is an output in single-wire mode.

Table 14-7. SCIC3 Field Descriptions (continued)

Field	Description
4 TXINV ¹	Transmit Data Inversion — Setting this bit reverses the polarity of the transmitted data output. 0 Transmit data not inverted 1 Transmit data inverted
3 ORIE	Overrun Interrupt Enable — This bit enables the overrun flag (OR) to generate hardware interrupt requests. 0 OR interrupts disabled (use polling). 1 Hardware interrupt requested when OR = 1.
2 NEIE	Noise Error Interrupt Enable — This bit enables the noise flag (NF) to generate hardware interrupt requests. 0 NF interrupts disabled (use polling). 1 Hardware interrupt requested when NF = 1.
1 FEIE	Framing Error Interrupt Enable — This bit enables the framing error flag (FE) to generate hardware interrupt requests. 0 FE interrupts disabled (use polling). 1 Hardware interrupt requested when FE = 1.
0 PEIE	Parity Error Interrupt Enable — This bit enables the parity error flag (PF) to generate hardware interrupt requests. 0 PF interrupts disabled (use polling). 1 Hardware interrupt requested when PF = 1.

¹ Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle.

14.2.7 SCI Data Register (SCID)

This register is actually two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer. Reads and writes of this register are also involved in the automatic flag clearing mechanisms for the SCI status flags.

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0

Figure 14-11. SCI Data Register (SCID)

14.3 Functional Description

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The SCI comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the SCI, writes the data to be transmitted, and processes received data. The following describes each of the blocks of the SCI.

14.3.1 Baud Rate Generation

As shown in [Figure 14-12](#), the clock source for the SCI baud rate generator is the bus-rate clock.

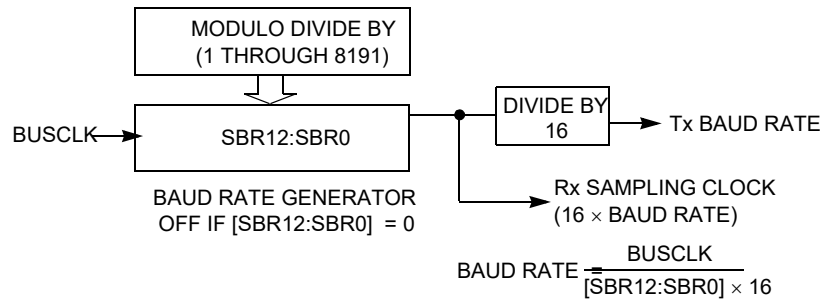


Figure 14-12. SCI Baud Rate Generation

SCI communications require the transmitter and receiver (which typically derive baud rates from independent clock sources) to use the same baud rate. Allowed tolerance on this baud frequency depends on the details of how the receiver synchronizes to the leading edge of the start bit and how bit sampling is performed.

The MCU resynchronizes to bit boundaries on every high-to-low transition, but in the worst case, there are no such transitions in the full 10- or 11-bit time character frame so any mismatch in baud rate is accumulated for the whole character time. For an NXP SCI system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about ± 4.5 percent for 8-bit data format and about ± 4 percent for 9-bit data format. Although baud rate modulo divider settings do not always produce baud rates that exactly match standard rates, it is normally possible to get within a few percent, which is acceptable for reliable communications.

14.3.2 Transmitter Functional Description

This section describes the overall block diagram for the SCI transmitter, as well as specialized functions for sending break and idle characters. The transmitter block diagram is shown in [Figure 14-2](#).

The transmitter output (TxD) idle state defaults to logic high ($TXINV = 0$ following reset). The transmitter output is inverted by setting $TXINV = 1$. The transmitter is enabled by setting the TE bit in SCIC2. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register (SCID).

The central element of the SCI transmitter is the transmit shift register that is either 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, we will assume $M = 0$, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new SCI character, the value waiting in the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCID.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD high, waiting for more characters to transmit.

Writing 0 to TE does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity that is in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

14.3.2.1 Send Break and Queued Idle

The SBK control bit in SCIC2 is used to send break characters which were originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). A longer break of 13 bit times can be enabled by setting BRK13 = 1. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK is still 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another NXP SCI, the break characters will be received as 0s in all eight data bits and a framing error (FE = 1) occurs.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while TE = 0, the SCI transmitter never actually releases control of the TxD pin. If there is a possibility of the shifter finishing while TE = 0, set the general-purpose I/O controls so the pin that is shared with TxD is an output driving a logic 1. This ensures that the TxD line will look like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

The length of the break character is affected by the BRK13 and M bits as shown below.

Table 14-8. Break Character Length

BRK13	M	Break Character Length
0	0	10 bit times
0	1	11 bit times
1	0	13 bit times
1	1	14 bit times

14.3.3 Receiver Functional Description

In this section, the receiver block diagram ([Figure 14-3](#)) is used as a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

The receiver input is inverted by setting RXINV = 1. The receiver is enabled by setting the RE bit in SCIC2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data mode, refer to [Section 14.3.5.1, 8- and 9-Bit Data Modes](#) for the remainder of this discussion, we assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF)

status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full ($RDRF = 1$), it gets the data from the receive data register by reading SCID. The RDRF flag is cleared automatically by a 2-step sequence which is normally satisfied in the course of the user's program that handles receive data. Refer to [Section 14.3.4, Interrupts and Status Flags](#) for more details about flag clearing.

14.3.3.1 Data Sampling Technique

The SCI receiver uses a $16\times$ baud rate clock for sampling. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the RxD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The $16\times$ baud rate clock is used to divide the bit time into 16 segments labeled RT1 through RT16. When a falling edge is located, three more samples are taken at RT3, RT5, and RT7 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

The receiver then samples each bit time, including the start and stop bits, at RT8, RT9, and RT10 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. In the case of the start bit, the bit is assumed to be 0 if at least two of the samples at RT3, RT5, and RT7 are 0 even if one or all of the samples taken at RT8, RT9, and RT10 are 1s. If any sample in any bit time (including the start and stop bits) in a character frame fails to agree with the logic level for that bit, the noise flag (NF) will be set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges, and if an edge is detected, the sample clock is resynchronized to bit times. This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

In the case of a framing error, the receiver is inhibited from receiving any new characters until the framing error flag is cleared. The receive shift register continues to function, but a complete character cannot transfer to the receive data buffer if FE is still set.

14.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCIC2. When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, IDLE, when RWUID bit is set) are inhibited from setting, thus eliminating the software overhead for handling the unimportant message

characters. At the end of a message, or at the beginning of the next message, all receivers automatically force RWU to 0 so all receivers wake up in time to look at the first character(s) of the next message.

14.3.3.2.1 Idle-Line Wakeup

When WAKE = 0, the receiver is configured for idle-line wakeup. In this mode, RWU is cleared automatically when the receiver detects a full character time of the idle-line level. The M control bit selects 8-bit or 9-bit data mode that determines how many bit times of idle are needed to constitute a full character time (10 or 11 bit times because of the start and stop bits).

When RWU is one and RWUID is zero, the idle condition that wakes up the receiver does not set the IDLE flag. The receiver wakes up and waits for the first data character of the next message which will set the RDRF flag and generate an interrupt if enabled. When RWUID is one, any idle condition sets the IDLE flag and generates an interrupt if enabled, regardless of whether RWU is zero or one.

The idle-line type (ILT) control bit selects one of two ways to detect an idle line. When ILT = 0, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When ILT = 1, the idle bit counter does not start until after a stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

14.3.3.2.2 Address-Mark Wakeup

When WAKE = 1, the receiver is configured for address-mark wakeup. In this mode, RWU is cleared automatically when the receiver detects a logic 1 in the most significant bit of a received character (eighth bit in M = 0 mode and ninth bit in M = 1 mode).

Address-mark wakeup allows messages to contain idle characters but requires that the MSB be reserved for use in address frames. The logic 1 MSB of an address frame clears the RWU bit before the stop bit is received and sets the RDRF flag. In this case the character with the MSB set is received even though the receiver was sleeping during most of this character time.

14.3.4 Interrupts and Status Flags

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF, IDLE, RXEDGIF and LBKDIF events, and a third vector is used for OR, NF, FE, and PF error conditions. Each of these ten interrupt sources can be separately masked by local interrupt enable masks. The flags can still be polled by software when the local masks are cleared to disable generation of hardware interrupt requests.

The SCI transmitter has two status flags that optionally can generate hardware interrupt requests. Transmit data register empty (TDRE) indicates when there is room in the transmit data buffer to write another transmit character to SCID. If the transmit interrupt enable (TIE) bit is set, a hardware interrupt will be requested whenever TDRE = 1. Transmit complete (TC) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TxD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (TCIE) bit is set, a hardware interrupt will be requested whenever TC = 1.

Instead of hardware interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are 0s.

When a program detects that the receive data register is full ($RDRF = 1$), it gets the data from the receive data register by reading SCID. The RDRF flag is cleared by reading SCIS1 while $RDRF = 1$ and then reading SCID.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, SCIS1 must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD line remains idle for an extended period of time. IDLE is cleared by reading SCIS1 while $IDLE = 1$ and then reading SCID. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set RDRF.

If the associated error was detected in the received character that caused RDRF to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — get set at the same time as RDRF. These flags are not set in overrun cases.

If RDRF was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag gets set instead the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the RxD serial data input pin causes the RXEDGIF flag to set. The RXEDGIF flag is cleared by writing a “1” to it. This function does depend on the receiver being enabled ($RE = 1$).

14.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

14.3.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in SCIC1. In 9-bit mode, there is a ninth data bit to the left of the MSB of the SCI data register. For the transmit data buffer, this bit is stored in T8 in SCIC3. For the receiver, the ninth bit is held in R8 in SCIC3.

For coherent writes to the transmit data buffer, write to the T8 bit before writing to SCID.

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to T8 again. When data is transferred from the transmit data buffer to the transmit shifter, the value in T8 is copied at the same time data is transferred from SCID to the shifter.

9-bit data mode typically is used in conjunction with parity to allow eight bits of data plus the parity in the ninth bit. Or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

14.3.5.2 Stop Mode Operation

During all stop modes, clocks to the SCI module are halted.

In stop1 and stop2 modes, all SCI register data is lost and must be re-initialized upon recovery from these two stop modes. No SCI module registers are affected in stop3 mode.

The receive input active edge detect circuit is still active in stop3 mode, but not in stop2. . An active edge on the receive input brings the CPU out of stop3 mode if the interrupt is not masked (RXEDGIE = 1).

Note, because the clocks are halted, the SCI module will resume operation upon exit from stop (only in stop3 mode). Software should ensure stop mode is not entered while there is a character being transmitted out of or received into the SCI module.

14.3.5.3 Loop Mode

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RxD pin is not used by the SCI, so it reverts to a general-purpose port I/O pin.

14.3.5.4 Single-Wire Operation

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Single-wire mode is used to implement a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD pin. The RxD pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the TXDIR bit in SCIC3 controls the direction of serial data on the TxD pin. When TXDIR = 0, the TxD pin is an input to the SCI receiver and the transmitter is temporarily disconnected from the TxD pin so an external device can send serial data to the receiver. When TXDIR = 1, the TxD pin is an output driven by the transmitter. In single-wire mode, the internal loop back connection from the transmitter to the receiver causes the receiver to receive characters that are sent out by the transmitter.

Chapter 15

Timer/Pulse-Width Modulator (S08TPMV3)

15.1 Introduction

15.1.1 ACMP/TPM Configuration Information

The ACMP module can be configured to connect the output of the analog comparator to the TPM input capture channel 0 by setting the ACIC bit in SOPT2. With ACIC set, the TPMCH0 pin is not available externally regardless of the configuration of the TPM module.

15.1.2 TPM External Clock

The TPM module on the MC9S08QL8 series use the TCLK pin as the external clock source.

15.1.3 TPM Pin Repositioning

The TPM module pin, TPMCH0, can be repositioned under software control using TPMCH0PS bits in SOPT2 as shown in [Table 15-1](#).

Table 15-1. TPM Position Options

TPMCH0PS in SOPT2	Port Pin for TPMCH0
0 (default)	PTA0
1	PTB5

15.1.4 TPM Clock Gating

The bus clock to the TPM can be gated on and off using the TPM bit in SCGC1. This bit is set after any reset, which enables the bus clock to this module. To conserve power, this bit can be cleared to disable the clock to this module when not in use. See [Section 5.7, Peripheral Clock Gating](#) for details.

15.1.5 TPMV3 Differences from Previous Versions

The TPMV3 is the latest version of the Timer/PWM module that addresses errata found in previous versions. The following section outlines the differences between TPMV3 and TPMV2 modules, and any considerations that should be taken when porting code.

Table 15-2. TPMV2 and TPMV3 Porting Considerations

Action	TPMV3	TPMV2
Write to TPMxCnTH:L registers¹		
Any write to TPMxCNTH or TPMxCNTL registers	Clears the TPM counter (TPMxCNTH:L) and the prescaler counter.	Clears the TPM counter (TPMxCNTH:L) only.
Read of TPMxCNTH:L registers¹		
In BDM mode, any read of TPMxCNTH:L registers	Returns the value of the TPM counter that is frozen.	If only one byte of the TPMxCNTH:L registers was read before the BDM mode became active, returns the latched value of TPMxCNTH:L from the read buffer (instead of the frozen TPM counter value).
In BDM mode, a write to TPMxSC, TPMxCNTH or TPMxCNTL	Clears this read coherency mechanism.	Does not clear this read coherency mechanism.
Read of TPMxCnVH:L registers²		
In BDM mode, any read of TPMxCnVH:L registers	Returns the value of the TPMxCnVH:L register.	If only one byte of the TPMxCnVH:L registers was read before the BDM mode became active, returns the latched value of TPMxCNTH:L from the read buffer (instead of the value in the TPMxCnVH:L registers).
In BDM mode, a write to TPMxCnSC	Clears this read coherency mechanism.	Does not clear this read coherency mechanism.
Write to TPMxCnVH:L registers		
In Input Capture mode, writes to TPMxCnVH:L registers ³	Not allowed.	Allowed.
In Output Compare mode, when (CLKSB:CLKSA not = 0:0), writes to TPMxCnVH:L registers ³	Update the TPMxCnVH:L registers with the value of their write buffer at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.	Always update these registers when their second byte is written.

Table 15-2. TPMV2 and TPMV3 Porting Considerations (continued)

Action	TPMV3	TPMV2
In Edge-Aligned PWM mode when (CLKSB:CLKSA not = 00), writes to TPMxCnVH:L registers	Update the TPMxCnVH:L registers with the value of their write buffer after both bytes were written and when the TPM counter changes from (TPMxMODH:L - 1) to (TPMxMODH:L). Note: If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from 0xFFFE to 0xFFFF.	Update after both bytes are written and when the TPM counter changes from TPMxMODH:L to 0x0000.
In Center-Aligned PWM mode when (CLKSB:CLKSA not = 00), writes to TPMxCnVH:L registers ⁴	Update the TPMxCnVH:L registers with the value of their write buffer after both bytes are written and when the TPM counter changes from (TPMxMODH:L - 1) to (TPMxMODH:L). Note: If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from 0xFFFE to 0xFFFF.	Update after both bytes are written and when the TPM counter changes from TPMxMODH:L to (TPMxMODH:L - 1).
Center-Aligned PWM		
When TPMxCnVH:L = TPMxMODH:L ⁵	Produces 100% duty cycle.	Produces 0% duty cycle.
When TPMxCnVH:L = (TPMxMODH:L - 1) ⁶	Produces a near 100% duty cycle.	Produces 0% duty cycle.
TPMxCnVH:L is changed from 0x0000 to a non-zero value ⁷	Waits for the start of a new PWM period to begin using the new duty cycle setting.	Changes the channel output at the middle of the current PWM period (when the count reaches 0x0000).
TPMxCnVH:L is changed from a non-zero value to 0x0000 ⁸	Finishes the current PWM period using the old duty cycle setting.	Finishes the current PWM period using the new duty cycle setting.
Write to TPMxMODH:L registers in BDM mode		
In BDM mode, a write to TPMxSC register	Clears the write coherency mechanism of TPMxMODH:L registers.	Does not clear the write coherency mechanism.

¹ For more information, refer to [Section 15.3.2, TPM-Counter Registers \(TPMCNTH:TPMCNTL\)](#). [SE110-TPM case 7]² For more information, refer to [Section 15.3.5, TPM Channel Value Registers \(TPMCnVH:TPMCnVL\)](#).³ For more information, refer to [Section 15.4.2.1, Input Capture Mode](#).⁴ For more information, refer to [Section 15.4.2.4, Center-Aligned PWM Mode](#).⁵ For more information, refer to [Section 15.4.2.4, Center-Aligned PWM Mode](#). [SE110-TPM case 1]⁶ For more information, refer to [Section 15.4.2.4, Center-Aligned PWM Mode](#). [SE110-TPM case 2]⁷ For more information, refer to [Section 15.4.2.4, Center-Aligned PWM Mode](#). [SE110-TPM case 3 and 5]

⁸ For more information, refer to [Section 15.4.2.4, Center-Aligned PWM Mode](#). [SE110-TPM case 4]

15.1.6 Migrating from TPMV1

In addition to [Section 15.1.5, TPMV3 Differences from Previous Versions](#), keep in mind the following considerations when migrating from a device that uses TPMV1.

- You can write to the Channel Value register (TPMxCnV) when the timer is not in input capture mode for TPMV2, not TPMV3.
- In edge- or center- aligned modes, the Channel Value register (TPMxCnV) registers only update when the timer changes from TPMMOD-1 to TPMMOD, or in the case of a free running timer from 0xFFFE to 0xFFFF.
- Also, when configuring the TPM modules, it is best to write to TPMxSC before TPMxCnV as a write to TPMxSC resets the coherency mechanism on the TPMxCnV registers.

Table 15-3. Migrating to TPMV3 Considerations

When...	Action / Best Practice
Writing to the Channel Value Register (TPMxCnV) register...	Timer must be in Input Capture mode.
Updating the Channel Value Register (TPMxCnV) register in edge-aligned or center-aligned modes...	Only occurs when the timer changes from TPMMOD-1 to TPMMOD (or in the case of a free running timer, from 0xFFFE to 0xFFFF).
Resetting the coherency mechanism for the Channel Value Register (TPMxCnV) register...	Write to TPMxSC.
Configuring the TPM modules...	Write first to TPMxSC and then to TPMxCnV register.

15.1.7 Features

The TPM includes these distinctive features:

- One to eight channels:
 - Each channel is input capture, output compare, or edge-aligned PWM
 - Rising-edge, falling-edge, or any-edge input capture trigger
 - Set, clear, or toggle output compare action
 - Selectable polarity on PWM outputs
- Module is configured for buffered, center-aligned pulse-width-modulation (CPWM) on all channels
- Timer clock source selectable as bus clock, fixed frequency clock, or an external clock
 - Prescale taps for divide-by 1, 2, 4, 8, 16, 32, 64, or 128 used for any clock input selection
 - Fixed frequency clock is an additional clock input to allow the selection of an on chip clock source other than bus clock
 - Selecting external clock connects TPM clock to a chip level input pin therefore allowing to synchronize the TPM counter with an off chip clock source
- 16-bit free-running or modulus count with up/down selection
- One interrupt per channel and one interrupt for TPM counter overflow

15.1.8 Modes of Operation

In general, TPM channels are independently configured to operate in input capture, output compare, or edge-aligned PWM modes. A control bit allows the whole TPM (all channels) to switch to center-aligned PWM mode. When center-aligned PWM mode is selected, input capture, output compare, and edge-aligned PWM functions are not available on any channels of this TPM module.

When the MCU is in active BDM background or BDM foreground mode, the TPM temporarily suspends all counting until the MCU returns to normal user operating mode. During stop mode, all TPM input clocks are stopped, so the TPM is effectively disabled until clocks resume. During wait mode, the TPM continues to operate normally. If the TPM does not need to produce a real time reference or provide the interrupt sources needed to wake the MCU from wait mode, the power can then be saved by disabling TPM functions before entering wait mode.

- Input capture mode

When a selected edge event occurs on the associated MCU pin, the current value of the 16-bit timer counter is captured into the channel value register and an interrupt flag bit is set. Rising edges, falling edges, any edge, or no edge (disable channel) are selected as the active edge that triggers the input capture.
- Output compare mode

When the value in the timer counter register matches the channel value register, an interrupt flag bit is set, and a selected output action is forced on the associated MCU pin. The output compare action is selected to force the pin to zero, force the pin to one, toggle the pin, or ignore the pin (used for software timing functions).
- Edge-aligned PWM mode

The value of a 16-bit modulo register plus 1 sets the period of the PWM output signal. The channel value register sets the duty cycle of the PWM output signal. You can also choose the polarity of the PWM output signal. Interrupts are available at the end of the period and at the duty-cycle transition point. This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period that is same for all channels within a TPM.

- Center-aligned PWM mode

Twice the value of a 16-bit modulo register sets the period of the PWM output, and the channel-value register sets the half-duty-cycle duration. The timer counter counts up until it reaches the modulo value and then counts down until it reaches zero. As the count matches the channel value register while counting down, the PWM output becomes active. When the count matches the channel value register while counting up, the PWM output becomes inactive. This type of PWM signal is called center-aligned because the centers of the active duty cycle periods for all channels are aligned with a count value of zero. This type of PWM is required for types of motors used in small appliances.

This is a high-level description only. Detailed descriptions of operating modes are in later sections.

15.1.9 Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPMxCHn (timer channel n) where n is the channel number (1–8). The TPM shares its I/O pins with general purpose I/O port pins (refer to I/O pin descriptions in full-chip specification for the specific chip implementation).

Figure 15-1 shows the TPM structure. The central component of the TPM is the 16-bit counter that can operate as a free-running counter or a modulo up/down counter. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture, output compare, and edge-aligned PWM functions. The timer counter modulo registers, TPMxMODH:TPMxMODL, control the modulo value of the counter (the values 0x0000 or 0xFFFF effectively make the counter free running). Software can read the counter value at any time without affecting the counting sequence. Any write to either half of the TPMxCNT counter resets the counter, regardless of the data value written.

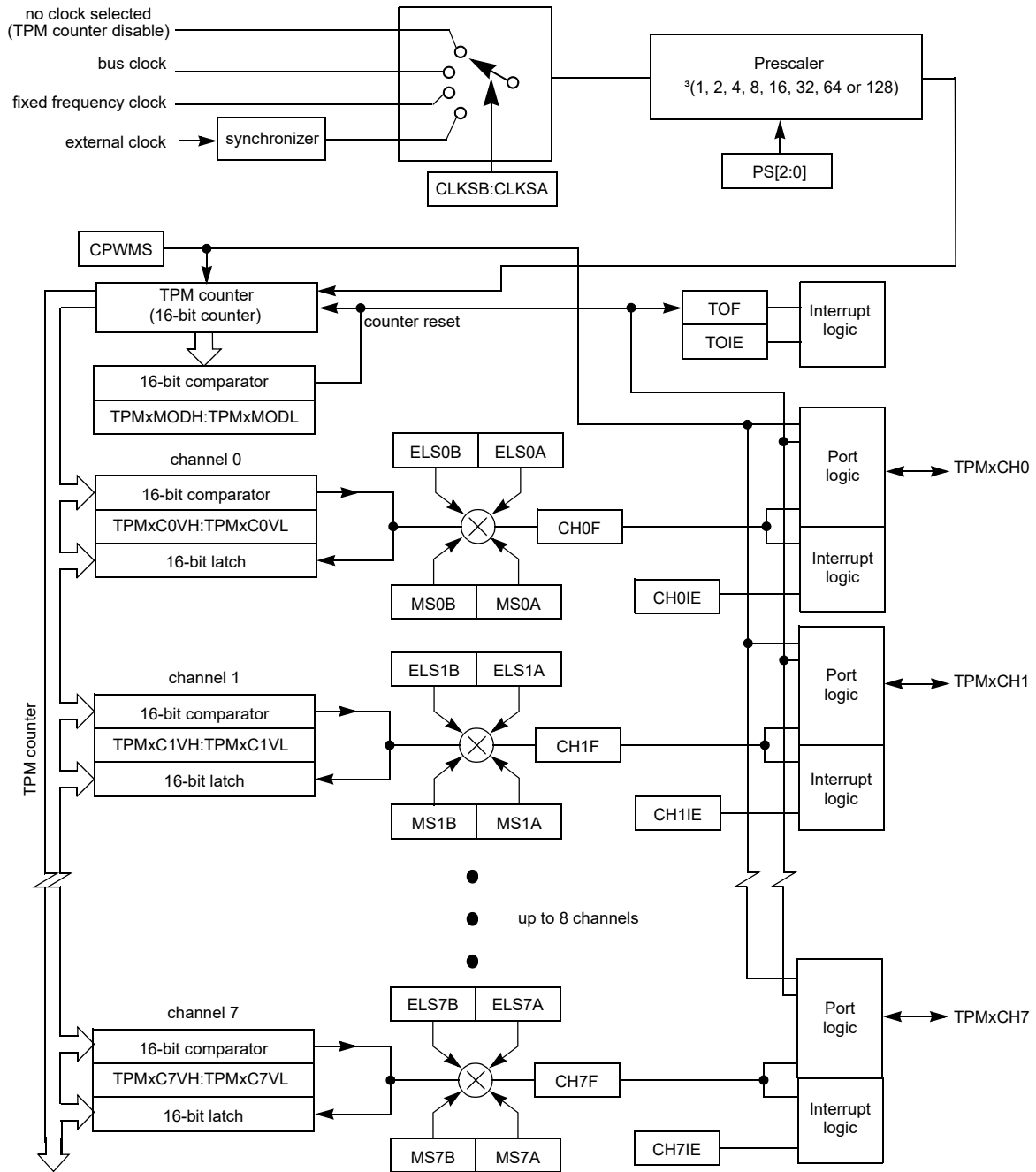


Figure 15-1. TPM Block Diagram

The TPM channels are programmable independently as input capture, output compare, or edge-aligned PWM channels. Alternately, the TPM can be configured to produce CPWM outputs on all channels. When the TPM is configured for CPWMs (the counter operates as an up/down counter) input capture, output compare, and EPWM functions are not practical.

15.2 Signal Description

Table 15-4 shows the user-accessible signals for the TPM. The number of channels are varied from one to eight. When an external clock is included, it can be shared with the same pin as any TPM channel; however, it could be connected to a separate input pin. Refer to the I/O pin descriptions in full-chip specification for the specific chip implementation.

Table 15-4. Signal Properties

Name	Function
EXTCLK ¹	External clock source that is selected to drive the TPM counter.
TPMxCHn ²	I/O pin associated with TPM channel n.

¹ The external clock pin can be shared with any channel pin. However, depending upon full-chip implementation, this signal could be connected to a separate external pin.

² n = channel number (1–8)

15.2.1 Detailed Signal Descriptions

15.2.1.1 EXTCLK — External Clock Source

The external clock signal can share the same pin as a channel pin, however the channel pin can not be used for channel I/O function when external clock is selected. If this pin is used as an external clock (CLKSB:CLKSA = 1:1), the channel can still be configured to output compare mode therefore allowing its use as a timer (ELSnB:ELSnA = 0:0).

For proper TPM operation, the external clock frequency must not exceed one-fourth of the bus clock frequency.

15.2.1.2 TPMxCHn — TPM Channel n I/O Pins

The TPM channel does not control the I/O pin when ELSnB:ELSnA or CLKSb:CLKSA are cleared so it normally reverts to general purpose I/O control. When CPWMS is set and ELSnB:ELSnA are not cleared, all TPM channels are configured for center-aligned PWM and the TPMxCHn pins are all controlled by TPM. When CPWMS is cleared, the MSnB:MSnA control bits determine whether the channel is configured for input capture, output compare, or edge-aligned PWM.

When a channel is configured for input capture (CPWMS = 0, MSnB:MSnA = 0:0, and ELSnB:ELSnA ≠ 0:0), the TPMxCHn pin is forced to act as an edge-sensitive input to the TPM. ELSnB:ELSnA control bits determine what polarity edge or edges trigger input capture events. The channel input signal is synchronized on the bus clock. This implies the minimum pulse width—that can

be reliably detected—on an input capture pin is four bus clock periods (with ideal clock pulses as near as two bus clocks can be detected).

When a channel is configured for output compare (CPWMS = 0, MSnB:MSnA = 0:1, and ELSnB:ELSnA $\neq$ 0:0), the TPMxCHn pin is an output controlled by the TPM. The ELSnB:ELSnA bits determine whether the TPMxCHn pin is toggled, cleared, or set each time the 16-bit channel value register matches the TPM counter.

When the output compare toggle mode is initially selected, the previous value on the pin is driven out until the next output compare event, the pin is then toggled.

When a channel is configured for edge-aligned PWM (CPWMS = 0, MSnB = 1, and ELSnB:ELSnA $\neq$ 0:0), the TPMxCHn pin is an output controlled by the TPM, and ELSnB:ELSnA bits control the polarity of the PWM output signal. When ELSnB is set and ELSnA is cleared, the TPMxCHn pin is forced high at the start of each new period (TPMxCNT=0x0000), and it is forced low when the channel value register matches the TPM counter. When ELSnA is set, the TPMxCHn pin is forced low at the start of each new period (TPMxCNT=0x0000), and it is forced high when the channel value register matches the TPM counter.

TPMxMODH:TPMxMODL = 0x0008
TPMxCnVH:TPMxCnVL = 0x0005

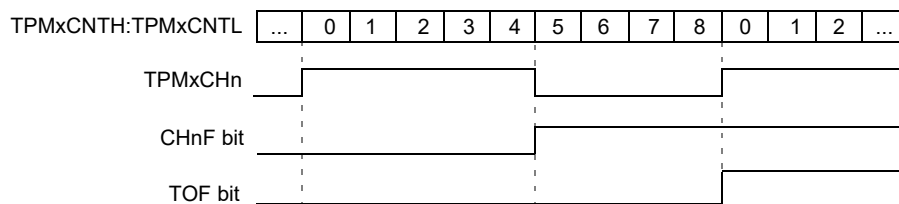


Figure 15-2. High-true pulse of an edge-aligned PWM

TPMxMODH:TPMxMODL = 0x0008
TPMxCnVH:TPMxCnVL = 0x0005

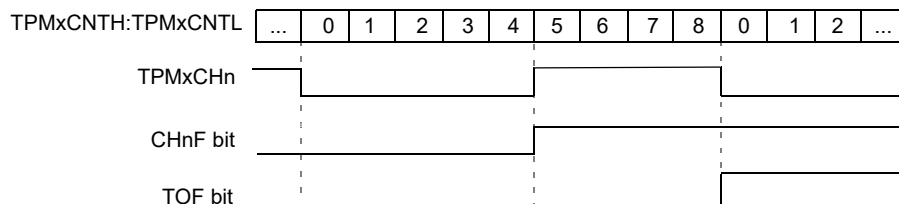


Figure 15-3. Low-true pulse of an edge-aligned PWM

When the TPM is configured for center-aligned PWM (CPWMS = 1 and ELSnB:ELSnA $\neq$ 0:0), the TPMxCHn pins are outputs controlled by the TPM, and ELSnB:ELSnA bits control the polarity of the PWM output signal. If ELSnB is set and ELSnA is cleared, the corresponding TPMxCHn pin is cleared when the TPM counter is counting up, and the channel value register matches the TPM counter; and it is

set when the TPM counter is counting down, and the channel value register matches the TPM counter. If ELSnA is set, the corresponding TPMxCHn pin is set when the TPM counter is counting up and the channel value register matches the TPM counter; and it is cleared when the TPM counter is counting down and the channel value register matches the TPM counter.

TPMxMODH:TPMxMODL = 0x0008
TPMxCnVH:TPMxCnVL = 0x0005

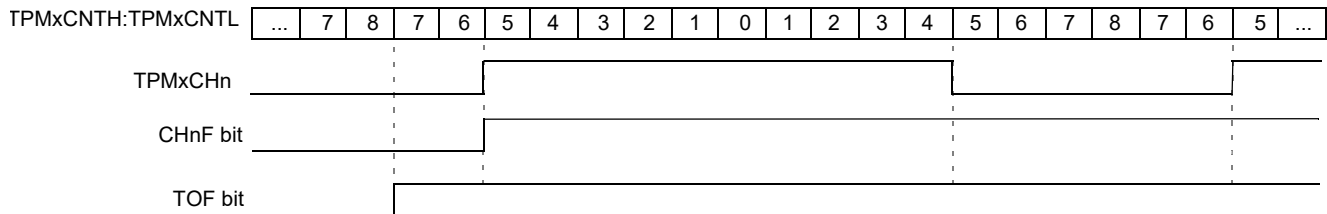


Figure 15-4. High-true pulse of a center-aligned PWM

TPMxMODH:TPMxMODL = 0x0008
TPMxCnVH:TPMxCnVL = 0x0005

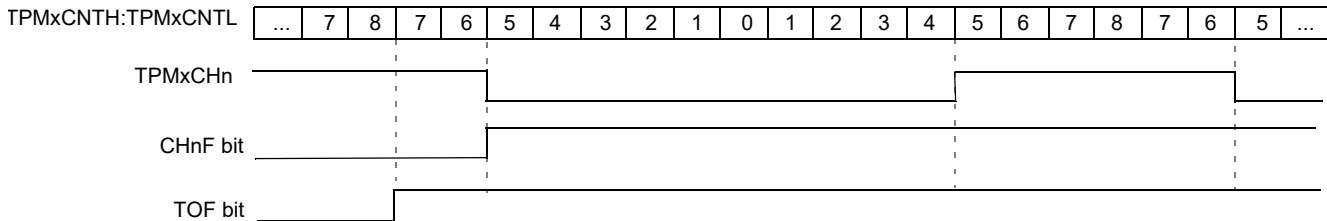


Figure 15-5. Low-true pulse of a center-aligned PWM

15.3 Register Definition

15.3.1 TPM Status and Control Register (TPMxSC)

TPMxSC contains the overflow status flag and control bits used to configure the interrupt enable, TPM configuration, clock source, and prescale factor. These controls relate to all channels within this timer module.

	7	6	5	4	3	2	1	0
R	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
W	0							
Reset	0	0	0	0	0	0	0	0

Figure 15-6. TPM Status and Control Register (TPMxSC)

Table 15-5. TPMxSC Field Descriptions

Field	Description
7 TOF	Timer overflow flag. This read/write flag is set when the TPM counter resets to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. Clear TOF by reading the TPM status and control register when TOF is set and then writing a logic 0 to TOF. If another TPM overflow occurs before the clearing sequence is completed, the sequence is reset so TOF remains set after the clear sequence was completed for the earlier TOF. This is done so a TOF interrupt request cannot be lost during the clearing sequence for a previous TOF. Reset clears TOF. Writing a logic 1 to TOF has no effect. 0 TPM counter has not reached modulo value or overflow. 1 TPM counter has overflowed.
6 TOIE	Timer overflow interrupt enable. This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals one. Reset clears TOIE. 0 TOF interrupts inhibited (use for software polling). 1 TOF interrupts enabled.
5 CPWMS	Center-aligned PWM select. This read/write bit selects CPWM operating mode. By default, the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up/down counting mode for CPWM functions. Reset clears CPWMS. 0 All channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MSnB:MSnA control bits in each channel's status and control register. 1 All channels operate in center-aligned PWM mode.
4–3 CLKS[B:A]	Clock source selection bits. As shown in Table 15-6, this 2-bit field is used to disable the TPM counter or select one of three clock sources to TPM counter and counter prescaler.
2–0 PS[2:0]	Prescale factor select. This 3-bit field selects one of eight division factors for the TPM clock as shown in Table 15-7. This prescaler is located after any clock synchronization or clock selection so it affects the clock selected to drive the TPM counter. The new prescale factor affects the selected clock on the next bus clock cycle after the new value is updated into the register bits.

Table 15-6. TPM Clock Selection

CLKSB:CLKSA	TPM Clock to Prescaler Input
00	No clock selected (TPM counter disable)

Table 15-6. TPM Clock Selection

CLKSB:CLKSA	TPM Clock to Prescaler Input
01	Bus clock
10	Fixed frequency clock
11	External clock

Table 15-7. Prescale Factor Selection

PS[2:0]	TPM Clock Divided-by
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

15.3.2 TPM-Counter Registers (TPMxCNTH:TPMxCNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPMxCNTH or TPMxCNTL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This allows coherent 16-bit reads in big-endian or little-endian order that makes this more friendly to various compiler implementations. The coherency mechanism is automatically restarted by an MCU reset or any write to the timer status/control register (TPMxSC).

Reset clears the TPM counter registers. Writing any value to TPMxCNTH or TPMxCNTL also clears the TPM counter (TPMxCNTH:TPMxCNTL) and resets the coherency mechanism, regardless of the data involved in the write.

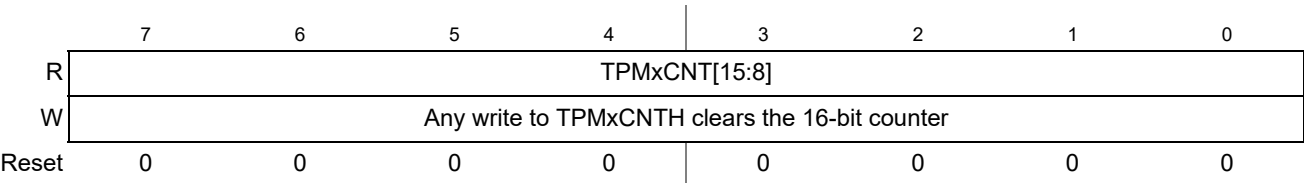


Figure 15-7. TPM Counter Register High (TPMxCNTH)

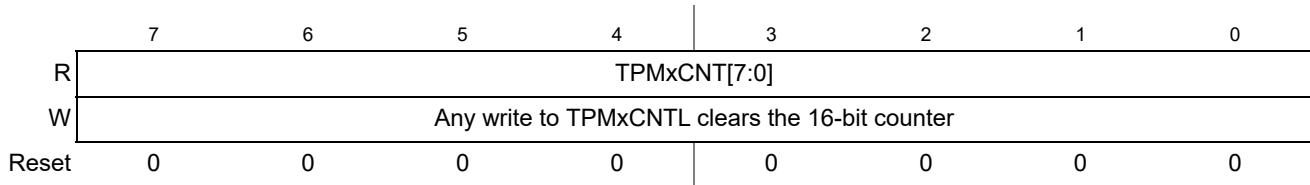


Figure 15-8. TPM Counter Register Low (TPMxCNTL)

When BDM is active, the timer counter is frozen (this is the value you read). The coherency mechanism is frozen so the buffer latches remain in the state they were in when the BDM became active, even if one or both counter halves are read while BDM is active. This assures that if you were in the middle of reading a 16-bit register when BDM became active, it reads the appropriate value from the other half of the 16-bit value after returning to normal execution.

In BDM mode, writing any value to TPMxSC, TPMxCNTH, or TPMxCNTL registers resets the read coherency mechanism of the TPMxCNTH:TPMxCNTL registers, regardless of the data involved in the write.

15.3.3 TPM Counter Modulo Registers (TPMxMODH:TPMxMODL)

The read/write TPM modulo registers contain the modulo value for the TPM counter. After the TPM counter reaches the modulo value, the TPM counter resumes counting from 0x0000 at the next clock, and the overflow flag (TOF) becomes set. Writing to TPMxMODH or TPMxMODL inhibits the TOF bit and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to 0x0000 that results in a free running timer counter (modulo disabled).

Writes to any of the registers TPMxMODH and TPMxMODL actually writes to buffer registers and the registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written
- If CLKSB and CLKSA are not cleared, the registers are updated after both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL – 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF

The latching mechanism is manually reset by writing to the TPMxSC address (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMxSC register) so the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any write to the modulo registers bypasses the buffer latches and directly writes to the modulo register while BDM is active.

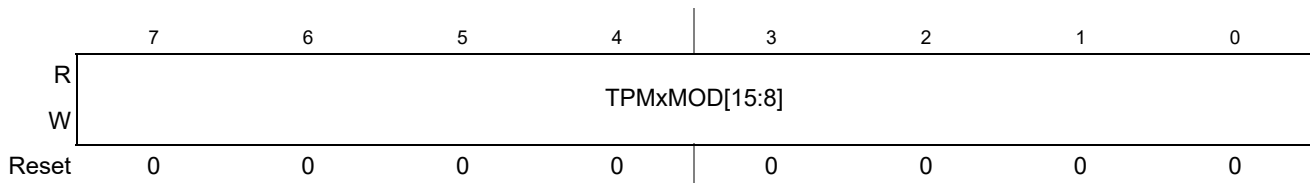
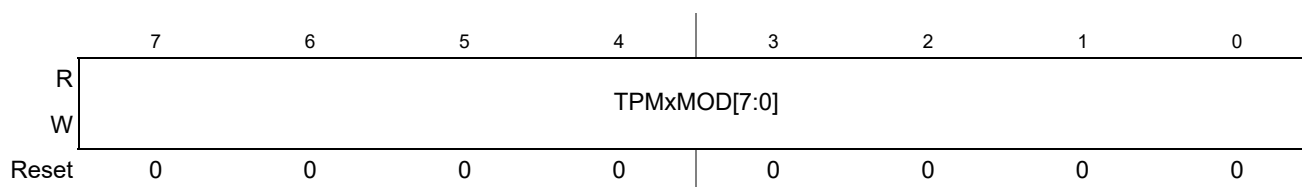


Figure 15-9. TPM Counter Modulo Register High (TPMxMODH)



Reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow occurs.

15.3.4 TPM Channel n Status and Control Register (TPMxCnSC)

TPMxCnSC contains the channel-interrupt-status flag and control bits that configure the interrupt enable, channel configuration, and pin function.

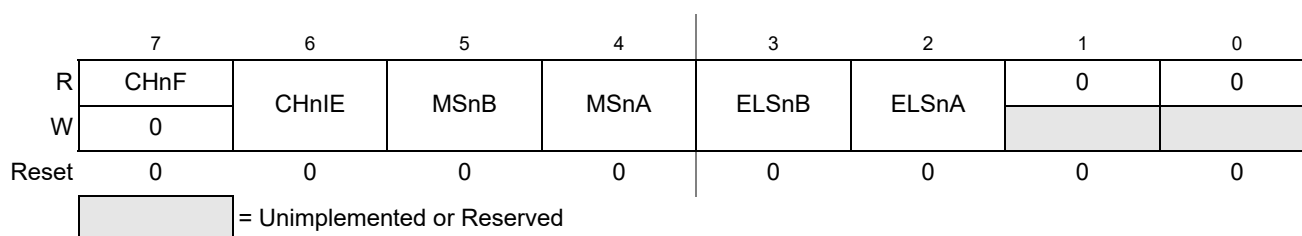


Figure 15-11. TPM Channel n Status and Control Register (TPMxCnSC)

Table 15-8. TPMxCnSC Field Descriptions

Field	Description
7 CHnF	Channel n flag. When channel n is an input capture channel, this read/write bit is set when an active edge occurs on the channel n input. When channel n is an output compare or edge-aligned/center-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. When channel n is an edge-aligned/center-aligned PWM channel and the duty cycle is set to 0% or 100%, CHnF is not set even when the value in the TPM counter registers matches the value in the TPM channel n value registers. A corresponding interrupt is requested when this bit is set and channel n interrupt is enabled (CHnIE = 1). Clear CHnF by reading TPMxCnSC while this bit is set and then writing a logic 0 to it. If another interrupt request occurs before the clearing sequence is completed CHnF remains set. This is done so a CHnF interrupt request is not lost due to clearing a previous CHnF. Reset clears this bit. Writing a logic 1 to CHnF has no effect. 0 No input capture or output compare event occurred on channel n. 1 Input capture or output compare event on channel n.
6 CHnIE	Channel n interrupt enable. This read/write bit enables interrupts from channel n. Reset clears this bit. 0 Channel n interrupt requests disabled (use for software polling). 1 Channel n interrupt requests enabled.
5 MSnB	Mode select B for TPM channel n. When CPWMS is cleared, setting the MSnB bit configures TPM channel n for edge-aligned PWM mode. Refer to the summary of channel mode and setup controls in Table 15-9 .

Table 15-8. TPMxCnSC Field Descriptions (continued)

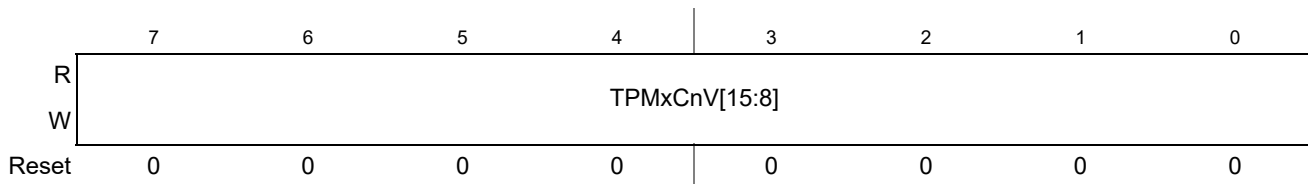
Field	Description
4 MSnA	Mode select A for TPM channel n. When CPWMS and MSnB are cleared, the MSnA bit configures TPM channel n for input capture mode or output compare mode. Refer to Table 15-9 for a summary of channel mode and setup controls. Note: If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger.
3–2 ELSnB ELSnA	Edge/level select bits. Depending upon the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in Table 15-9 , these bits select the polarity of the input edge that triggers an input capture event, select the level that is driven in response to an output compare match, or select the polarity of the PWM output. If ELSnB and ELSnA bits are cleared, the channel pin is not controlled by TPM. This configuration can be used by software compare only, because it does not require the use of a pin for the channel.

Table 15-9. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	XX	00	Pin is not controlled by TPM. It is reverted to general purpose I/O or other peripheral control	
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	00	Output compare	Software compare only
		01		Toggle output on channel match
		10		Clear output on channel match
		11		Set output on channel match
	1X	10	Edge-aligned PWM	High-true pulses (clear output on channel match)
		X1		Low-true pulses (set output on channel match)
1	XX	10	Center-aligned PWM	High-true pulses (clear output on channel match when TPM counter is counting up)
		X1		Low-true pulses (set output on channel match when TPM counter is counting up)

15.3.5 TPM Channel Value Registers (TPMxCnVH:TPMxCnVL)

These read/write registers contain the captured TPM counter value of the input capture function or the output compare value for the output compare or PWM functions. The channel registers are cleared by reset.

**Figure 15-12. TPM Channel Value Register High (TPMxCnVH)**

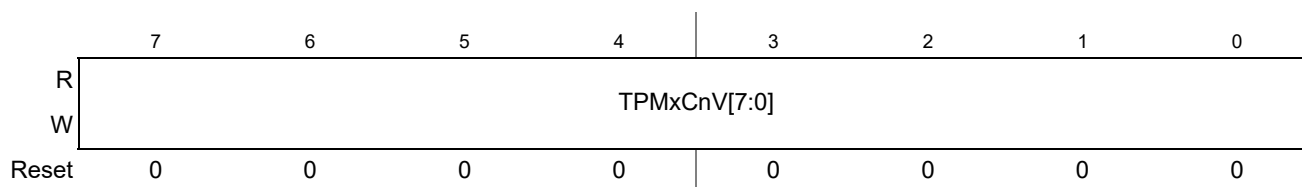


Figure 15-13. TPM Channel Value Register Low (TPMxCnVL)

In input capture mode, reading either byte (TPMxCnVH or TPMxCnVL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This latching mechanism also resets (becomes unlatched) when the TPMxCnSC register is written (whether BDM mode is active or not). Any write to the channel registers is ignored during the input capture mode.

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMxCnSC register) so the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the channel register are read while BDM is active. This assures that if you were in the middle of reading a 16-bit register when BDM became active, it reads the appropriate value from the other half of the 16-bit value after returning to normal execution. The value read from the TPMxCnVH and TPMxCnVL registers in BDM mode is the value of these registers and not the value of their read buffer.

In output compare or PWM modes, writing to either byte (TPMxCnVH or TPMxCnVL) latches the value into a buffer. After both bytes were written, they are transferred as a coherent 16-bit value into the timer-channel registers according to the value of CLKSB:CLKSA bits and the selected mode:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written.
- If CLKSB and CLKSA are not cleared and in output compare mode, the registers are updated after the second byte is written and on the next change of the TPM counter (end of the prescaler counting).
- If CLKSB and CLKSA are not cleared and in EPWM or CPWM modes, the registers are updated after both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL – 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF.

The latching mechanism is manually reset by writing to the TPMxCnSC register (whether BDM mode is active or not). This latching mechanism allows coherent 16-bit writes in either big-endian or little-endian order that is friendly to various compiler implementations.

When BDM is active, the coherency mechanism is frozen so the buffer latches remain in the state they were in when the BDM became active even if one or both halves of the channel register are written while BDM is active. Any write to the channel registers bypasses the buffer latches and directly write to the channel register while BDM is active. The values written to the channel register while BDM is active are used for PWM and output compare operation after normal execution resumes. Writes to the channel registers while BDM is active do not interfere with partial completion of a coherency sequence. After the coherency mechanism is fully exercised, the channel registers are updated using the buffered values (while BDM was not active).

15.4 Functional Description

All TPM functions are associated with a central 16-bit counter that allows flexible selection of the clock and prescale factor. There is also a 16-bit modulo register associated with this counter.

The CPWMS control bit chooses between center-aligned PWM operation for all channels in the TPM (CPWMS=1) or general purpose timing functions (CPWMS=0) where each channel can independently be configured to operate in input capture, output compare, or edge-aligned PWM mode. The CPWMS control bit is located in the TPM status and control register because it affects all channels within the TPM and influences the way the main counter operates. (In CPWM mode, the counter changes to an up/down mode rather than the up-counting mode used for general purpose timer functions.)

The following sections describe TPM counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation and interrupt activity depend upon the operating mode, these topics are covered in the associated mode explanation sections.

15.4.1 Counter

All timer functions are based on the main 16-bit counter (TPMxCNTH:TPMxCNTL). This section discusses selection of the clock, end-of-count overflow, up-counting vs. up/down counting, and manual counter reset.

15.4.1.1 Counter Clock Source

The 2-bit field, CLKSB:CLKSA, in the timer status and control register (TPMxSC) disables the TPM counter or selects one of three clock sources to TPM counter ([Table 15-6](#)). After any MCU reset, CLKSB and CLKSA are cleared so no clock is selected and the TPM counter is disabled (TPM is in a very low power state). You can read or write these control bits at any time. Disabling the TPM counter by writing 00 to CLKSB:CLKSA bits, does not affect the values in the TPM counter or other registers.

The fixed frequency clock is an alternative clock source for the TPM counter that allows the selection of a clock other than the bus clock or external clock. This clock input is defined by chip integration. You can refer chip specific documentation for further information. Due to TPM hardware implementation limitations, the frequency of the fixed frequency clock must not exceed the bus clock frequency. The fixed frequency clock has no limitations for low frequency operation.

The external clock passes through a synchronizer clocked by the bus clock to assure that counter transitions are properly aligned to bus clock transitions. Therefore, in order to meet Nyquist criteria considering also jitter, the frequency of the external clock source must not exceed 1/4 of the bus clock frequency.

When the external clock source is shared with a TPM channel pin, this pin must not be used in input capture mode. However, this channel can be used in output compare mode with ELSnB:ELSnA = 0:0 for software timing functions. In this case, the channel output is disabled, but the channel match events continue to set the appropriate flag.

15.4.1.2 Counter Overflow and Modulo Reset

An interrupt flag and enable are associated with the 16-bit main counter. The flag (TOF) is a software-accessible indication that the timer counter has overflowed. The enable signal selects between software polling (TOIE = 0) where no interrupt is generated, or interrupt-driven operation (TOIE = 1) where the interrupt is generated whenever the TOF is set.

The conditions causing TOF to become set depend on whether the TPM is configured for center-aligned PWM (CPWMS = 1). If CPWMS is cleared and there is no modulus limit, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF is set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF is set at the transition from the value set in the modulus register to 0x0000. When the TPM is in center-aligned PWM mode (CPWMS = 1), the TOF flag is set as the counter changes direction at the end of the count value set in the modulus register (at the transition from the value set in the modulus register to the next lower count value). This corresponds to the end of a PWM period (the 0x0000 count value corresponds to the center of a period).

15.4.1.3 Counting Modes

The main timer counter has two counting modes. When center-aligned PWM is selected (CPWMS = 1), the counter operates in up/down counting mode. Otherwise, the counter operates as a simple up counter. As an up counter, the timer counter counts from 0x0000 through its terminal count and continues with 0x0000. The terminal count is 0xFFFF or a modulus value in TPMxMODH:TPMxMODL.

When center-aligned PWM operation is specified, the counter counts up from 0x0000 through its terminal count and then down to 0x0000 where it changes back to up counting. The terminal count value and 0x0000 are normal length counts (one timer clock period long). In this mode, the timer overflow flag (TOF) is set at the end of the terminal-count period (as the count changes to the next lower count value).

15.4.1.4 Manual Counter Reset

The main timer counter can be manually reset at any time by writing any value to TPMxCNTH or TPMxCNTL. Resetting the counter in this manner also resets the coherency mechanism in case only half of the counter was read before resetting the count.

15.4.2 Channel Mode Selection

If CPWMS is cleared, MSnB and MSnA bits determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and edge-aligned PWM.

15.4.2.1 Input Capture Mode

With the input capture function, the TPM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the TPM latches the contents of the TPM counter into the channel-value registers (TPMxCnVH:TPMxCnVL). Rising edges, falling edges, or any edge is chosen as the active edge that triggers an input capture.

In input capture mode, the TPMxCnVH and TPMxCnVL registers are read only.

When either half of the 16-bit capture register is read, the other half is latched into a buffer to support coherent 16-bit accesses in big-endian or little-endian order. The coherency sequence can be manually reset by writing to TPMxCnSC.

An input capture event sets a flag bit (CHnF) that optionally generates a CPU interrupt request.

While in BDM, the input capture function works as configured. When an external event occurs, the TPM latches the contents of the TPM counter (frozen because of the BDM mode) into the channel value registers and sets the flag bit.

15.4.2.2 Output Compare Mode

With the output compare function, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter reaches the value in TPMxCnVH:TPMxCnVL registers of an output compare channel, the TPM can set, clear, or toggle the channel pin.

Writes to any of TPMxCnVH and TPMxCnVL registers actually write to buffer registers. In output compare mode, the TPMxCnVH:TPMxCnVL registers are updated with the value of their write buffer only after both bytes were written and according to the value of CLKSB:CLKSA bits:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written
- If CLKSB and CLKSA are not cleared, the registers are updated at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.

The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An output compare event sets a flag bit (CHnF) that optionally generates a CPU interrupt request.

15.4.2.3 Edge-Aligned PWM Mode

This type of PWM output uses the normal up-counting mode of the timer counter (CPWMS=0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the value of the modulus register (TPMxMODH:TPMxMODL) plus 1. The duty cycle is determined by the value of the timer channel register (TPMxCnVH:TPMxCnVL). The polarity of this PWM signal is determined by ELSnA bit. 0% and 100% duty cycle cases are possible.

The time between the modulus overflow and the channel match value (TPMxCnVH:TPMxCnVL) is the pulse width or duty cycle ([Figure 15-14](#)). If ELSnA is cleared, the counter overflow forces the PWM signal high, and the channel match forces the PWM signal low. If ELSnA is set, the counter overflow forces the PWM signal low, and the channel match forces the PWM signal high.

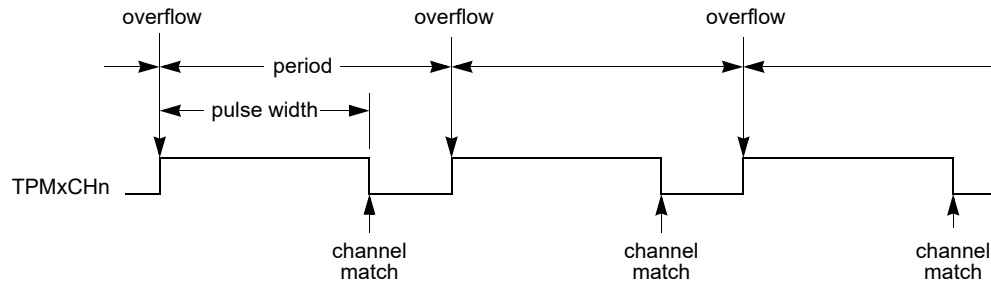


Figure 15-14. EPWM period and pulse width (ELSnA=0)

When the channel value register is set to 0x0000, the duty cycle is 0%. A 100% duty cycle is achieved by setting the timer-channel register (TPMxCnVH:TPMxCnVL) to a value greater than the modulus setting. This implies that the modulus setting must be less than 0xFFFF in order to get 100% duty cycle.

The timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMxCnVH and TPMxCnVL actually write to buffer registers. In edge-aligned PWM mode, the TPMxCnVH:TPMxCnVL registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written
- If CLKSB and CLKSA are not cleared, the registers are updated after both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL – 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF.

15.4.2.4 Center-Aligned PWM Mode

This type of PWM output uses the up/down counting mode of the timer counter (CPWMS=1). The channel match value in TPMxCnVH:TPMxCnVL determines the pulse width (duty cycle) of the PWM signal while the period is determined by the value in TPMxMODH:TPMxMODL. TPMxMODH:TPMxMODL must be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA determines the polarity of the CPWM signal.

$$\text{pulse width} = 2 \times (\text{TPMxCnVH:TPMxCnVL})$$

$$\text{period} = 2 \times (\text{TPMxMODH:TPMxMODL}); \text{TPMxMODH:TPMxMODL} = 0x0001\text{--}0x7FFF$$

If TPMxCnVH:TPMxCnVL is zero or negative (bit 15 set), the duty cycle is 0%. If TPMxCnVH:TPMxCnVL is a positive value (bit 15 clear) and is greater than the non-zero modulus setting, the duty cycle is 100% because the channel match never occurs. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FFF if you do not need to generate 100% duty cycle). This is not a significant limitation. The resulting period is much longer than required for normal applications.

All zeros in TPMxMODH:TPMxMODL is a special case that must not be used with center-aligned PWM mode. When CPWMS is cleared, this case corresponds to the counter running free from 0x0000 through 0xFFFF. When CPWMS is set, the counter needs a valid match to the modulus register somewhere other than at 0x0000 in order to change directions from up-counting to down-counting.

The channel match value in the TPM channel registers (times two) determines the pulse width (duty cycle) of the CPWM signal (Figure 15-15). If ELSnA is cleared, a channel match occurring while counting up clears the CPWM output signal and a channel match occurring while counting down sets the output. The counter counts up until it reaches the modulo setting in TPMxMODH:TPMxMODL, then counts down until it reaches zero. This sets the period equal to two times TPMxMODH:TPMxMODL.

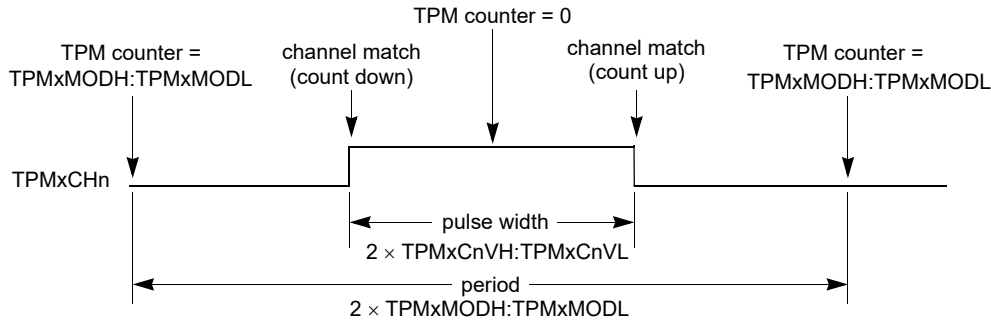


Figure 15-15. CPWM period and pulse width (ELSnA=0)

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

Input capture, output compare, and edge-aligned PWM functions do not make sense when the counter is operating in up/down counting mode so this implies that all active channels within a TPM must be used in CPWM mode when CPWMS is set.

The timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMxCnVH and TPMxCnVL actually write to buffer registers. In center-aligned PWM mode, the TPMxCnVH:TPMxCnVL registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written
- If CLKSB and CLKSA are not cleared, the registers are updated after both bytes were written, and the TPM counter changes from $(\text{TPMxMODH:TPMxMODL} - 1)$ to $(\text{TPMxMODH:TPMxMODL})$. If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF.

When TPMxCNTH:TPMxCNTL equals TPMxMODH:TPMxMODL, the TPM can optionally generate a TOF interrupt (at the end of this count).

15.5 Reset Overview

15.5.1 General

The TPM is reset whenever any MCU reset occurs.

15.5.2 Description of Reset Operation

Reset clears TPMxSC that disables TPM counter clock and overflow interrupt (TOIE=0). CPWMS, MSnB, MSnA, ELSnB, and ELSnA are all cleared. This configures all TPM channels for input capture operation and the associated pins are not controlled by TPM.

15.6 Interrupts

15.6.1 General

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on each channel's mode of operation. If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized. If the channel is configured for output compare or PWM modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register.

All TPM interrupts are listed in [Table 15-10](#).

Table 15-10. Interrupt Summary

Interrupt	Local Enable	Source	Description
TOF	TOIE	Counter overflow	Set each time the TPM counter reaches its terminal count (at transition to its next count value)
CHnF	CHnIE	Channel event	An input capture event or channel match took place on channel n

The TPM module provides high-truth interrupt signals.

15.6.2 Description of Interrupt Operation

For each interrupt source in the TPM, a flag bit is set upon recognition of the interrupt condition such as timer overflow, channel input capture, or output compare events. This flag is read (polled) by software to determine that the action has occurred, or an associated enable bit (TOIE or CHnIE) can be set to enable the interrupt generation. While the interrupt enable bit is set, the interrupt is generated whenever the associated interrupt flag is set. Software must perform a sequence of steps to clear the interrupt flag before returning from the interrupt-service routine.

TPM interrupt flags are cleared by a two-step process including a read of the flag bit while it is set followed by a write of zero to the bit. If a new event is detected between these two steps, the sequence is reset and the interrupt flag remains set after the second step to avoid the possibility of missing the new event.

15.6.2.1 Timer Overflow Interrupt (TOF) Description

The meaning and details of operation for TOF interrupts varies slightly depending upon the mode of operation of the TPM system (general purpose timing functions versus center-aligned PWM operation). The flag is cleared by the two step sequence described above.

15.6.2.1.1 Normal Case

When CPWMS is cleared, TOF is set when the timer counter changes from the terminal count (the value in the modulo register) to 0x0000. If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFF to 0x0000.

15.6.2.1.2 Center-Aligned PWM Case

When CPWMS is set, TOF is set when the timer counter changes direction from up-counting to down-counting at the end of the terminal count (the value in the modulo register).

15.6.2.2 Channel Event Interrupt Description

The meaning of channel interrupts depends on the channel's current mode (input capture, output compare, edge-aligned PWM, or center-aligned PWM).

15.6.2.2.1 Input Capture Events

When a channel is configured as an input capture channel, the ELSnB:ELSnA bits select if channel pin is not controlled by TPM, rising edges, falling edges, or any edge as the edge that triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the two-step sequence described in [Section 15.6.2, Description of Interrupt Operation](#).

15.6.2.2.2 Output Compare Events

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the two-step sequence described in [Section 15.6.2, Description of Interrupt Operation](#).

15.6.2.2.3 PWM End-of-Duty-Cycle Events

When the channel is configured for edge-aligned PWM, the channel flag is set when the timer counter matches the channel value register that marks the end of the active duty cycle period. When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle period when the timer counter matches the channel value register. The flag is cleared by the two-step sequence described in [Section 15.6.2, Description of Interrupt Operation](#).

Chapter 16

Development Support

16.1 Introduction

Development support systems in the HCS08 include the background debug controller (BDC). The BDC provides a single-wire debug interface to the target MCU that provides a convenient interface for programming the on-chip flash and other nonvolatile memories. The BDC is also the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as CPU register modify, breakpoints, and single instruction trace commands.

In the HCS08 family, address and data bus signals are not available on external pins. Debug is done through commands fed into the target MCU via the single-wire background debug interface.

16.1.1 Forcing Active Background

The method for forcing active background mode depends on the specific HCS08 derivative. For the MC9S08QL8 series, you can force active background after a power-on reset by holding the BKGD pin low as the device exits the reset condition. You can also force active background by driving BKGD low immediately after a serial background command that writes a one to the BDFR bit in the SBDFR register. Other causes of reset, including an external pin reset or an internally generated error reset ignore the state of the BKGD pin and reset into normal user mode. If no debug pod is connected to the BKGD pin, the MCU will always reset into normal operating mode.

16.1.2 Module Configuration

The alternate BDC clock source is the ICSLCLK. This clock source is selected by clearing the CLKSW bit in the BDCSCR register.

16.1.3 Features

Features of the BDC module include:

- Single pin for mode selection and background communications
- BDC registers are not located in the memory map
- SYNC command to determine target communications rate
- Non-intrusive commands for memory access
- Active background mode commands for CPU register access
- GO and TRACE1 commands
- BACKGROUND command can wake CPU from stop or wait modes
- One hardware address breakpoint built into BDC
- Oscillator runs in stop mode, if BDC enabled
- COP watchdog disabled while in active background mode

16.2 Background Debug Controller (BDC)

All MCUs in the HCS08 Family contain a single-wire background debug interface that supports in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities. Unlike debug interfaces on earlier 8-bit MCUs, this system does not interfere with normal application resources. It does not use any user memory or locations in the memory map and does not share any on-chip peripherals.

BDC commands are divided into two groups:

- Active background mode commands require that the target MCU is in active background mode (the user program is not running). Active background mode commands allow the CPU registers to be read or written, and allow the user to trace one user instruction at a time, or GO to the user program from active background mode.
- Non-intrusive commands can be executed at any time even while the user's program is running. Non-intrusive commands allow a user to read or write MCU memory locations or access status and control registers within the background debug controller.

Typically, a relatively simple interface pod is used to translate commands from a host computer into commands for the custom serial interface to the single-wire background debug system. Depending on the development tool vendor, this interface pod may use a standard RS-232 serial port, a parallel printer port, or some other type of communications such as a universal serial bus (USB) to communicate between the host PC and the pod. The pod typically connects to the target system with ground, the BKGD pin, $\overline{\text{RESET}}$, and sometimes V_{DD} . An open-drain connection to reset allows the host to force a target system reset, which is useful to regain control of a lost target system or to control startup of a target system before the on-chip nonvolatile memory has been programmed. Sometimes V_{DD} can be used to allow the pod to use power from the target system to avoid the need for a separate power supply. However, if the pod is powered separately, it can be connected to a running target system without forcing a target system reset or otherwise disturbing the running application program.

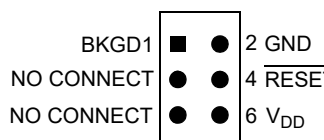


Figure 16-1. BDM Tool Connector

16.2.1 BKGD Pin Description

BKGD is the single-wire background debug interface pin. The primary function of this pin is for bidirectional serial communication of active background mode commands and data. During reset, this pin is used to select between starting in active background mode or starting the user's application program. This pin is also used to request a timed sync response pulse to allow a host development tool to determine the correct clock frequency for background debug serial communications.

BDC serial communications use a custom serial protocol first introduced on the M68HC12 Family of microcontrollers. This protocol assumes the host knows the communication clock rate that is determined by the target BDC clock rate. All communication is initiated and controlled by the host that drives a high-to-low edge to signal the beginning of each bit time. Commands and data are sent most significant bit first (MSB first). For a detailed description of the communications protocol, refer to [Section 16.2.2, Communication Details](#).

If a host is attempting to communicate with a target MCU that has an unknown BDC clock rate, a SYNC command may be sent to the target MCU to request a timed sync response signal from which the host can determine the correct communication speed.

BKGD is a pseudo-open-drain pin and there is an on-chip pullup so no external pullup resistor is required. Unlike typical open-drain pins, the external RC time constant on this pin, which is influenced by external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively driven speedup pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to [Section 16.2.2, Communication Details](#), for more detail.

When no debugger pod is connected to the 6-pin BDM interface connector, the internal pullup on BKGD chooses normal operating mode. When a debug pod is connected to BKGD it is possible to force the MCU into active background mode after reset. The specific conditions for forcing active background depend upon the HCS08 derivative (refer to the introduction to this Development Support section). It is not necessary to reset the target MCU to communicate with it through the background debug interface.

16.2.2 Communication Details

The BDC serial interface requires the external controller to generate a falling edge on the BKGD pin to indicate the start of each bit time. The external controller provides this falling edge whether data is transmitted or received.

BKGD is a pseudo-open-drain pin that can be driven either by an external controller or by the MCU. Data is transferred MSB first at 16 BDC clock cycles per bit (nominal speed). The interface times out if 512 BDC clock cycles occur between falling edges from the host. Any BDC command that was in progress

when this timeout occurs is aborted without affecting the memory or operating mode of the target MCU system.

The custom serial protocol requires the debug pod to know the target BDC communication clock speed.

The clock switch (CLKSW) control bit in the BDC status and control register allows the user to select the BDC clock source. The BDC clock source can either be the bus or the alternate BDC clock source.

The BKGD pin can receive a high or low level or transmit a high or low level. The following diagrams show timing for each of these cases. Interface timing is synchronous to clocks in the target BDC, but asynchronous to the external host. The internal BDC clock signal is shown for reference in counting cycles.

Figure 16-2 shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target HCS08 MCU. The host is asynchronous to the target so there is a 0-to-1 cycle delay from the host-generated falling edge to where the target perceives the beginning of the bit time. Ten target BDC clock cycles later, the target senses the bit level on the BKGD pin. Typically, the host actively drives the pseudo-open-drain BKGD pin during host-to-target transmissions to speed up rising edges. Because the target does not drive the BKGD pin during the host-to-target transmission period, there is no need to treat the line as an open-drain signal during this period.

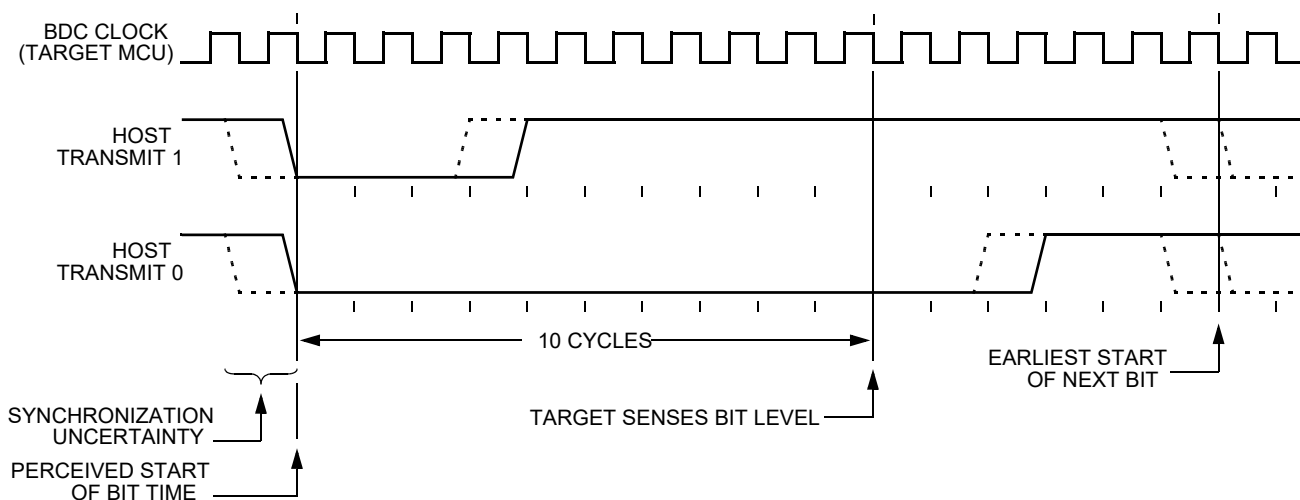


Figure 16-2. BDC Host-to-Target Serial Bit Timing

Figure 16-3 shows the host receiving a logic 1 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target BDC cycles). The host must release the low drive before the target MCU drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.

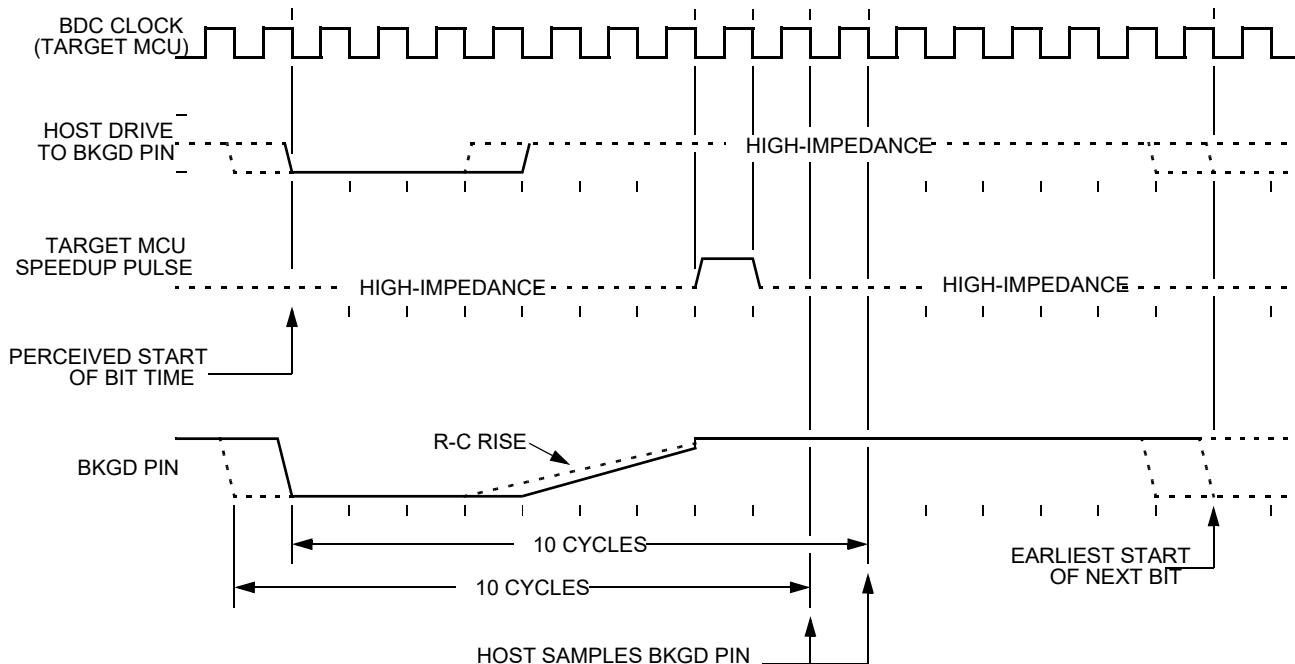


Figure 16-3. BDC Target-to-Host Serial Bit Timing (Logic 1)

Figure 16-4 shows the host receiving a logic 0 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target MCU. The host initiates the bit time but the target HCS08 finishes it. Because the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 BDC clock cycles, then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 cycles after starting the bit time.

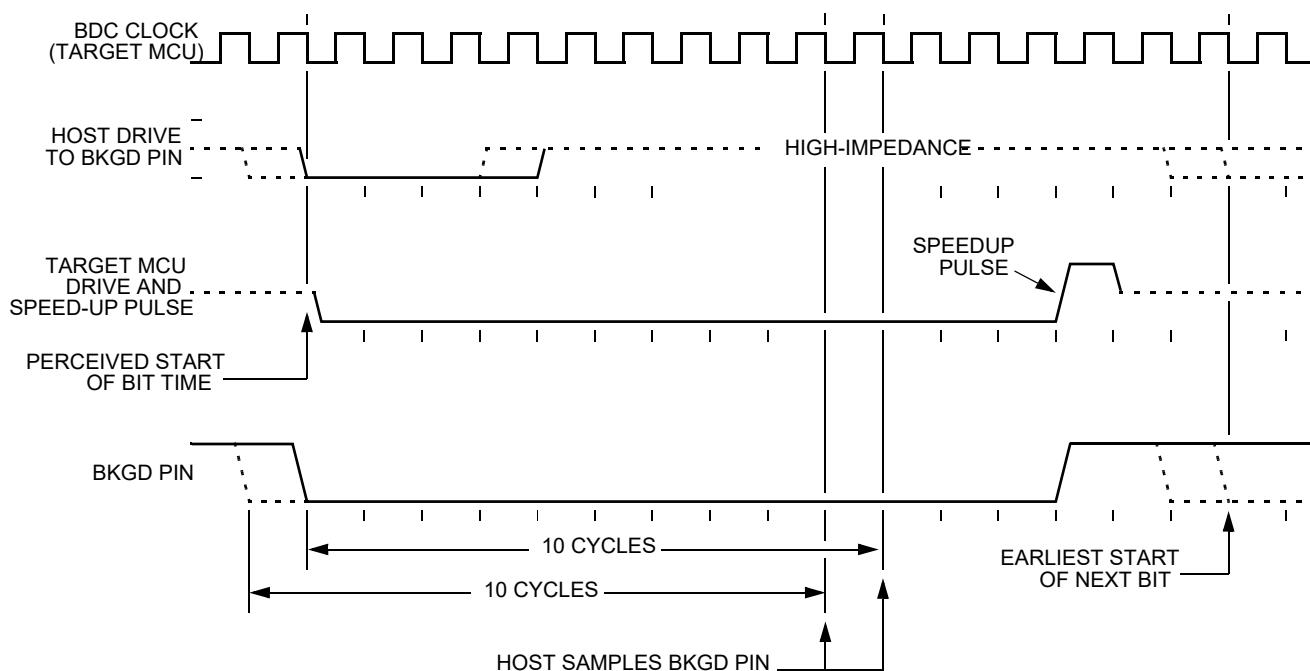


Figure 16-4. BDM Target-to-Host Serial Bit Timing (Logic 0)

16.2.3 BDC Commands

BDC commands are sent serially from a host computer to the BKGD pin of the target HCS08 MCU. All commands and data are sent MSB-first using a custom BDC communications protocol. Active background mode commands require that the target MCU is currently in the active background mode while non-intrusive commands may be issued at any time whether the target MCU is in active background mode or running a user application program.

Table 16-1 shows all HCS08 BDC commands, a shorthand description of their coding structure, and the meaning of each command.

Coding Structure Nomenclature

This nomenclature is used in Table 16-1 to describe the coding structure of the BDC commands.

Commands begin with an 8-bit hexadecimal command code in the host-to-target direction (most significant bit first)

/	=	separates parts of the command
d	=	delay 16 target BDC clock cycles
AAAA	=	a 16-bit address in the host-to-target direction
RD	=	8 bits of read data in the target-to-host direction
WD	=	8 bits of write data in the host-to-target direction
RD16	=	16 bits of read data in the target-to-host direction
WD16	=	16 bits of write data in the host-to-target direction
SS	=	the contents of BDCSCR in the target-to-host direction (STATUS)
CC	=	8 bits of write data for BDCSCR in the host-to-target direction (CONTROL)
RBKP	=	16 bits of read data in the target-to-host direction (from BDCBKPT breakpoint register)
WBKP	=	16 bits of write data in the host-to-target direction (for BDCBKPT breakpoint register)

Table 16-1. BDC Command Summary

Command Mnemonic	Active BDM/ Non-intrusive	Coding Structure	Description
SYNC	Non-intrusive	n/a ¹	Request a timed reference pulse to determine target BDC communication speed
ACK_ENABLE	Non-intrusive	D5/d	Enable acknowledge protocol. Refer to NXP document order no. HCS08RMv1/D.
ACK_DISABLE	Non-intrusive	D6/d	Disable acknowledge protocol. Refer to NXP document order no. HCS08RMv1/D.
BACKGROUND	Non-intrusive	90/d	Enter active background mode if enabled (ignore if ENBDM bit equals 0)
READ_STATUS	Non-intrusive	E4/SS	Read BDC status from BDCSCR
WRITE_CONTROL	Non-intrusive	C4/CC	Write BDC controls in BDCSCR
READ_BYTE	Non-intrusive	E0/AAAA/d/RD	Read a byte from target memory
READ_BYTE_WS	Non-intrusive	E1/AAAA/d/SS/RD	Read a byte and report status
READ_LAST	Non-intrusive	E8/SS/RD	Re-read byte from address just read and report status
WRITE_BYTE	Non-intrusive	C0/AAAA/WD/d	Write a byte to target memory
WRITE_BYTE_WS	Non-intrusive	C1/AAAA/WD/d/SS	Write a byte and report status
READ_BKPT	Non-intrusive	E2/RBKP	Read BDCBKPT breakpoint register
WRITE_BKPT	Non-intrusive	C2/WBKP	Write BDCBKPT breakpoint register
GO	Active BDM	08/d	Go to execute the user application program starting at the address currently in the PC
TRACE1	Active BDM	10/d	Trace 1 user instruction at the address in the PC, then return to active background mode
TAGGO	Active BDM	18/d	Same as GO but enable external tagging (HCS08 devices have no external tagging pin)
READ_A	Active BDM	68/d/RD	Read accumulator (A)
READ_CCR	Active BDM	69/d/RD	Read condition code register (CCR)
READ_PC	Active BDM	6B/d/RD16	Read program counter (PC)
READ_HX	Active BDM	6C/d/RD16	Read H and X register pair (H:X)
READ_SP	Active BDM	6F/d/RD16	Read stack pointer (SP)
READ_NEXT	Active BDM	70/d/RD	Increment H:X by one then read memory byte located at H:X
READ_NEXT_WS	Active BDM	71/d/SS/RD	Increment H:X by one then read memory byte located at H:X. Report status and data.
WRITE_A	Active BDM	48/WD/d	Write accumulator (A)
WRITE_CCR	Active BDM	49/WD/d	Write condition code register (CCR)
WRITE_PC	Active BDM	4B/WD16/d	Write program counter (PC)
WRITE_HX	Active BDM	4C/WD16/d	Write H and X register pair (H:X)
WRITE_SP	Active BDM	4F/WD16/d	Write stack pointer (SP)
WRITE_NEXT	Active BDM	50/WD/d	Increment H:X by one, then write memory byte located at H:X
WRITE_NEXT_WS	Active BDM	51/WD/d/SS	Increment H:X by one, then write memory byte located at H:X. Also report status.

¹ The SYNC command is a special operation that does not have a command code.

The SYNC command is unlike other BDC commands because the host does not necessarily know the correct communications speed to use for BDC communications until after it has analyzed the response to the SYNC command.

To issue a SYNC command, the host:

- Drives the BKGD pin low for at least 128 cycles of the slowest possible BDC clock (The slowest clock is normally the reference oscillator/64 or the self-clocked rate/64.)
- Drives BKGD high for a brief speedup pulse to get a fast rise time (This speedup pulse is typically one cycle of the fastest clock in the system.)
- Removes all drive to the BKGD pin so it reverts to high impedance
- Monitors the BKGD pin for the sync response pulse

The target, upon detecting the SYNC request from the host (which is a much longer low time than would ever occur during normal BDC communications):

- Waits for BKGD to return to a logic high
- Delays 16 cycles to allow the host to stop driving the high speedup pulse
- Drives BKGD low for 128 BDC clock cycles
- Drives a 1-cycle high speedup pulse to force a fast rise time on BKGD
- Removes all drive to the BKGD pin so it reverts to high impedance

The host measures the low time of this 128-cycle sync response pulse and determines the correct speed for subsequent BDC communications. Typically, the host can determine the correct communication speed within a few percent of the actual target speed and the communication protocol can easily tolerate speed errors of several percent.

16.2.4 BDC Hardware Breakpoint

The BDC includes one relatively simple hardware breakpoint that compares the CPU address bus to a 16-bit match value in the BDCBKPT register. This breakpoint can generate a forced breakpoint or a tagged breakpoint. A forced breakpoint causes the CPU to enter active background mode at the first instruction boundary following any access to the breakpoint address. The tagged breakpoint causes the instruction opcode at the breakpoint address to be tagged so that the CPU will enter active background mode rather than executing that instruction if and when it reaches the end of the instruction queue. This implies that tagged breakpoints can only be placed at the address of an instruction opcode while forced breakpoints can be set at any address.

The breakpoint enable (BKPTEN) control bit in the BDC status and control register (BDCSCR) is used to enable the breakpoint logic (BKPTEN = 1). When BKPTEN = 0, its default value after reset, the breakpoint logic is disabled and no BDC breakpoints are requested regardless of the values in other BDC breakpoint registers and control bits. The force/tag select (FTS) control bit in BDCSCR is used to select forced (FTS = 1) or tagged (FTS = 0) type breakpoints.

16.3 Register Definition

This section contains the descriptions of the BDC registers and control bits.

This section refers to registers and control bits only by their names. An NXP-provided equate or header file is used to translate these names into the appropriate absolute addresses.

16.3.1 BDC Registers and Control Bits

The BDC has two registers:

- The BDC status and control register (BDCSCR) is an 8-bit register containing control and status bits for the background debug controller.
- The BDC breakpoint match register (BDCBKPT) holds a 16-bit breakpoint match address.

These registers are accessed with dedicated serial BDC commands and are not located in the memory space of the target MCU (so they do not have addresses and cannot be accessed by user programs).

Some of the bits in the BDCSCR have write limitations; otherwise, these registers may be read or written at any time. For example, the ENBDM control bit may not be written while the MCU is in active background mode. (This prevents the ambiguous condition of the control bit forbidding active background mode while the MCU is already in active background mode.) Also, the four status bits (BDMACT, WS, WSF, and DVF) are read-only status indicators and can never be written by the WRITE_CONTROL serial BDC command. The clock switch (CLKSW) control bit may be read or written at any time.

16.3.1.1 BDC Status and Control Register (BDCSCR)

This register can be read or written by serial BDC commands (READ_STATUS and WRITE_CONTROL) but is not accessible to user programs because it is not located in the normal memory map of the MCU.

	7	6	5	4	3	2	1	0
R	ENBDM	BDMACT	BKPTEN	FTS	CLKSW	WS	WSF	DVF
W								
Normal Reset	0	0	0	0	0	0	0	0
Reset in Active BDM:	1	1	0	0	1	0	0	0

 = Unimplemented or Reserved

Figure 16-5. BDC Status and Control Register (BDCSCR)

Table 16-2. BDCSCR Register Field Descriptions

Field	Description
7 ENBDM	Enable BDM (Permit Active Background Mode) — Typically, this bit is written to 1 by the debug host shortly after the beginning of a debug session or whenever the debug host resets the target and remains 1 until a normal reset clears it. 0 BDM cannot be made active (non-intrusive commands still allowed) 1 BDM can be made active to allow active background mode commands
6 BDMACT	Background Mode Active Status — This is a read-only status bit. 0 BDM not active (user application program running) 1 BDM active and waiting for serial commands
5 BKPTEN	BDC Breakpoint Enable — If this bit is clear, the BDC breakpoint is disabled and the FTS (force tag select) control bit and BDCBKPT match register are ignored. 0 BDC breakpoint disabled 1 BDC breakpoint enabled
4 FTS	Force/Tag Select — When FTS = 1, a breakpoint is requested whenever the CPU address bus matches the BDCBKPT match register. When FTS = 0, a match between the CPU address bus and the BDCBKPT register causes the fetched opcode to be tagged. If this tagged opcode ever reaches the end of the instruction queue, the CPU enters active background mode rather than executing the tagged opcode. 0 Tag opcode at breakpoint address and enter active background mode if CPU attempts to execute that instruction 1 Breakpoint match forces active background mode at next instruction boundary (address need not be an opcode)
3 CLKSW	Select Source for BDC Communications Clock — CLKSW defaults to 0, which selects the alternate BDC clock source. 0 Alternate BDC clock source 1 MCU bus clock

Table 16-2. BDCSCR Register Field Descriptions (continued)

Field	Description
2 WS	Wait or Stop Status — When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop and into active background mode where all BDC commands work. Whenever the host forces the target MCU into active background mode, the host should issue a READ_STATUS command to check that BDMACT = 1 before attempting other BDC commands. 0 Target CPU is running user application code or in active background mode (was not in wait or stop mode when background became active) 1 Target CPU is in wait or stop mode, or a BACKGROUND command was used to change from wait or stop to active background mode
1 WSF	Wait or Stop Failure Status — This status bit is set if a memory access command failed due to the target CPU executing a wait or stop instruction at or about the same time. The usual recovery strategy is to issue a BACKGROUND command to get out of wait or stop mode into active background mode, repeat the command that failed, then return to the user program. (Typically, the host would restore CPU registers and stack values and re-execute the wait or stop instruction.) 0 Memory access did not conflict with a wait or stop instruction 1 Memory access command failed because the CPU entered wait or stop mode
0 DVF	Data Valid Failure Status — This status bit is not used in the MC9S08QL8 Series because it does not have any slow access memory. 0 Memory access did not conflict with a slow memory access 1 Memory access command failed because CPU was not finished with a slow memory access

16.3.1.2 BDC Breakpoint Match Register (BDCBKPT)

This 16-bit register holds the address for the hardware breakpoint in the BDC. The BKPTEN and FTS control bits in BDCSCR are used to enable and configure the breakpoint logic. Dedicated serial BDC commands (READ_BKPT and WRITE_BKPT) are used to read and write the BDCBKPT register but is not accessible to user programs because it is not located in the normal memory map of the MCU. Breakpoints are normally set while the target MCU is in active background mode before running the user application program. For additional information about setup and use of the hardware breakpoint logic in the BDC, refer to [Section 16.2.4, BDC Hardware Breakpoint](#).

16.3.2 System Background Debug Force Reset Register (SBDFR)

This register contains a single write-only control bit. A serial background mode command such as WRITE_BYTE must be used to write to SBDFR. Attempts to write this register from a user program are ignored. Reads always return 0x00.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W								BDFR ¹
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

¹ BDFR is writable only through serial background mode debug commands, not from user programs.

Figure 16-6. System Background Debug Force Reset Register (SBDFR)

Table 16-3. SBDFR Register Field Description

Field	Description
0 BDFR	Background Debug Force Reset — A serial active background mode command such as WRITE_BYTE allows an external debug host to force a target system reset. Writing 1 to this bit forces an MCU reset. This bit cannot be written from a user program.

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, μ Vision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2018 NXP B.V.

