Servicing Peripheral Interrupts While Writing Flash

TB3344

Introduction

Author: Henrik M. Arnesen, Egil Rotevatn, Microchip Technology Inc.

The Microchip AVR® EA Family of microcontrollers features Flash memory where the Flash is divided into one Read While Write (RWW) section and one Non Read While Write (NRWW) section, enabling the CPU to continue running instructions from the NRWW section while erasing or writing to the RWW section. Since the erase/write operations to Flash is timed in the order of milliseconds, the RWW functionality enables the user to utilize this time and not having to wait for it to finish. The purpose of this document is to describe how to use the NRWW Flash section to write to the RWW Flash section on AVR EA devices and why this is beneficial.

This technical brief explains the concept of RWW/NRWW and its implementation on the AVR EA Family of microcontrollers with the following use case:

Servicing Peripheral Interrupts While Writing Flash

- This example showcases the benefits of being able to write to RWW from NRWW Flash memory
- Different device pins are continuously toggled to show important events in the Erase/Write cycle
- The pins represent periodic interrupts, page buffer status, and Flash statuses
- The events have different occurrences when erasing and writing a program page in either NRWW or RWW areas

Note: For the use case described in this document, two code examples are set up, one made using Microchip Code Configurator (MCC) in MPLAB® and one bare metal example developed using Microchip Studio on AVR® EA. Both can be found on Microchip Discover:

- Example code for Microchip Studio
- Example code for MPLAB X

© 2023 Microchip Technology Inc. and its subsidiaries

Table of Contents

Introduction....1

1. Relevant Devices....3
2. Overview....4
3. Servicing Peripheral Interrupts While Writing Flash....7
4. Get Code Examples from MPLAB Discover....11
5. Revision History....12

Microchip Information....13

- The Microchip Website....13
- Product Change Notification Service....13
- Customer Support....13
- Microchip Devices Code Protection Feature....13
- Legal Notice....13
- Trademarks....14
- Quality Management System....15
- Worldwide Sales and Service....16

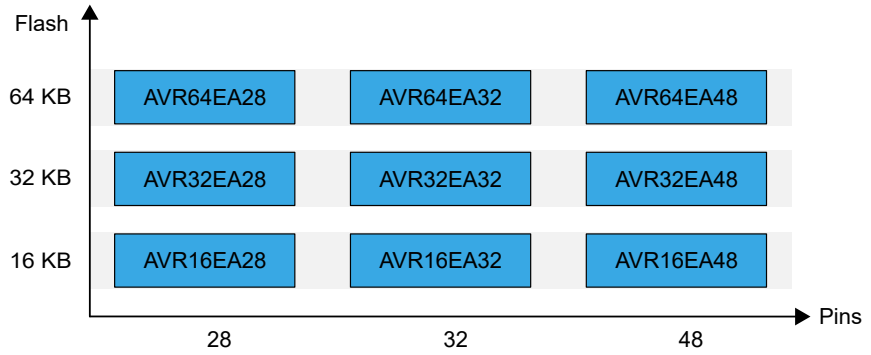
© 2023 Microchip Technology Inc. and its subsidiaries

1. Relevant Devices

This section lists the relevant devices for this document. The following figures show the different family devices, laying out pin count variants and memory sizes:

- Vertical migration upwards is possible without code modification, as these devices are pin-compatible and provide the same or more features
- Horizontal migration to the left reduces the pin count and, therefore, the available features
- Devices with different Flash memory sizes typically also have different SRAM and EEPROM

Figure 1-1. AVR® EA Family Overview

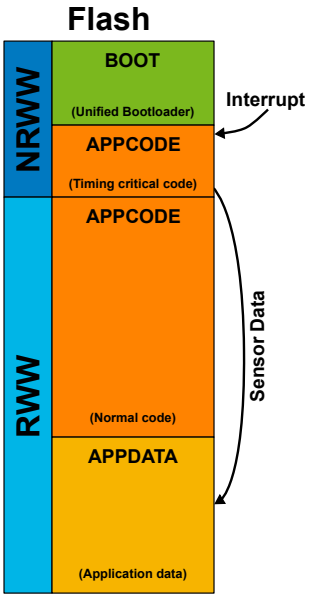


© 2023 Microchip Technology Inc. and its subsidiaries

2. Overview

Flash memory is organized into pages consisting of several words. In general for AVR, a word is 2 bytes or 16 bits. For the AVR EA, the page size is 128 bytes. Independent of page size, the Flash is split into two physical sections: Non Read-While-Write (NRWW) and Read-While-Write (NRWW). The NRWW section starts at the start of the Flash and, by that, overlaps the BOOT section and the APPCODE/APPDATA section, depending on the fuse settings. It's important to note that writing to the same section as the code is running from is impossible. In the example where the NRWW overlaps the boundary between the BOOT/APPCODE or BOOT/APPDATA sections, the uninterrupted code needs to be placed in the BOOT section to run while an erase/write cycle is underway in APPCODE/APPDATA. If configuring to have BOOT/APPCODE/APPDATA, the uninterrupted code can also be in the NRWW part of the APPCODE section to program APPDATA without halting the CPU.

Figure 2-1. RWW Memory Flow



The syntax "RWW section" refers to which section is being programmed (erased or written) and not the one read. Only code located inside the NRWW Flash is accessible by either executing a CPU instruction or reading data while programming the RWW section.

It follows that the main differences between the NRWW and RWW sections are:

1. When erasing or writing to a page located inside the RWW section using code running from the NRWW section, code and data can continue to be run and read from the NRWW section, enabling the continuous operation of the CPU while the program memory operation is running.
2. When erasing or writing to a page in the NRWW section, the CPU is halted during the operation. An exception to this is when writing from the BOOT section to the APPCODE or APPDATA section that overlaps with the NRWW section.

An application, leveraging the NRWW/RWW program memory split of AVR EA, prevents the CPU from blocking while the program memory is written to.

© 2023 Microchip Technology Inc. and its subsidiaries

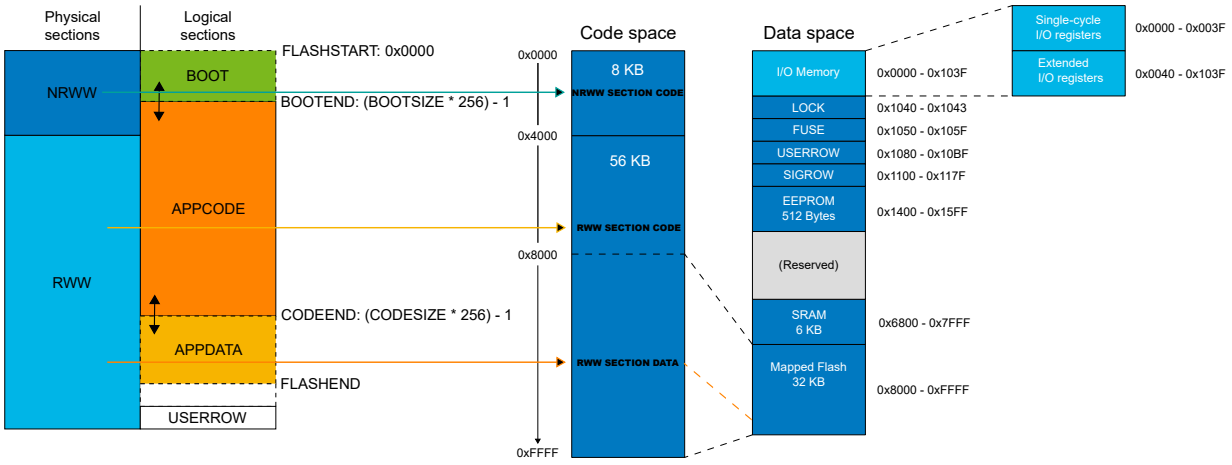
- One example is a Bootloader scenario where the CPU acts upon commands received via a communication peripheral (I^2C/USART) from the Host while simultaneously programming the APPCODE/APPDATA sections in program memory
- Another use case is a data logging scenario in which an analog peripheral interrupts CPU operation when data that at once must to be saved to program memory, becomes available

Both scenarios highlight the need for the CPU not to be blocked while the Nonvolatile Memory Controller (NVMCTRL) peripheral is erasing/writing to the program memory. By ensuring that the code that needs to run while an erase/write cycle is in progress is in the NRWW section, we can ensure that commands or interrupts are handled as they arrive and not after the Flash erase/write cycle.

To understand how to implement the Read-While-Write (RWW) feature on AVR EA, it helps to look at the program memory from a Physical, Logical, and Code/Data Space point of view. Physically, the program memory is designed with fixed NRWW and RWW sections. These are non-changeable and depend on the part number - see the "Memory Overview" section in the data sheet. Store the code and data in the respective physical sections to use the RWW use case.

The Logical sections are BOOT, APPCODE and APPDATA. These sections can be adjusted through the BOOTSIZE and CODESIZE fuses, as shown in the image below. Note - depending on the fuse configuration, the physical NRWW section can be used for both BOOT and APPCODE/APPDATA. See the data sheet in the NVMCTRL peripheral *Memory Organization* section for a detailed explanation. Note that you can access the code space via the data space through the Flash mapping feature, available on devices with a program memory exceeding 32kB. See the data sheet in the NVMCTRL peripheral *Memory Access* section for a detailed explanation.

Figure 2-2. AVR64EA48 Flash Sections



To implement this allocation, linker-named attributes with assigned addresses are given as linker options, and the named attributes are used when declaring both functions and data.

The RWW_DATA_SECTION attribute is used for data stored by the page write. These data can be written from either *BOOT* or *APPCODE* sections. The code snippet below shows the placement of an array in the APPDATA section. Define the .rww_data address in the Linker settings before compiling.

```
// the .rww_data section is at address 0x2000 (word address 0x1000)
#define RWW_DATA_SECTION __attribute__((used, section(".rww_data")))

const RWW_DATA_SECTION uint8_t rww_array[DATA_SIZE] = {0};
```

Use the NRWW_PROG_SECTION attribute for routines that **can be** executed in conjunction with Flash programming or erase (e.g., interrupts, app code Flash page write or erase, and boot loader

© 2023 Microchip Technology Inc. and its subsidiaries

host communication driver). In the code snippet below, a function called FillBuffer is placed in the NRWW section so that it can be called by interrupts running while writing to the Flash. Once again, the Linker settings must define the .nrww_program address.

```
// the .nrww_program section is at address 0x0400 (word address 0x0200)
#define NRWW_PROG_SECTION __attribute__((section(".nrww_program")))

void NRWW_SECTION_CODE FillBuffer(void);
```

Use the RWW_PROG_SECTION attribute for routines that **will not** execute during Flash programming or erase. These are blocking and must be placed in the physical RWW section. Examples of these are the setup routines.

Below is an example of the use of attribute sections.

If using a bootloader in the project, the bootloader code is placed first, from the Flash start (0x0000) until the BOOTEND (BOOTSIZE *256 -1). The BOOTSIZE fuse settings determine the BOOT section size. The Application code (App code) is placed after the BOOT (as long as CODESIZE > BOOTSIZE), which means that the application code may reside both in NRWW and RWW, making it possible to move functions into the NRWW section. Functions or interrupts placed in the NRWW section can run while erasing/writing the RWW section. However, take steps to ensure there are no interrupts or jumps to code located inside the RWW while operation on the RWW is ongoing, leading to the software ending up in an unknown state.

Table 2-1. Read-While-Write Scenarios

Flash Section Erased/Written	Flash Section Accessed	CPU
NRWW section	NRWW section	Halted
RWW section	NRWW section	Running
NRWW section	RWW section	Halted
RWW section	RWW section	Halted

Find the Flash memory size and the NRWW/RWW sections in the AVR EA's data sheet.

© 2023 Microchip Technology Inc. and its subsidiaries

3. Servicing Peripheral Interrupts While Writing Flash

This example shows the advantage of writing to the RWW Flash section from the NRWW Flash section. It is easy to get an overview of code flow and how it works with an oscilloscope or logic analyzer by using device pins for interrupt events and different memory statuses.

Testing

The program memory is divided into smaller pieces called pages. Each page is written and erased separately and consists of several bytes. The Flash has page granularity for the AVR EA, meaning it will perform an erase and/or write for every byte in the page whenever a change is necessary. In comparison, the EEPROM memory has byte granularity and can erase and write single bytes at a time.

The code works by first erasing the pages where data will be saved, filling up a buffer, and then performing a page write using the buffer values. A periodic timer interrupt continuously populate the buffer.

When the ring buffer is filled with new data >= `PROGMEM_PAGE_SIZE`, it copies the `PROGMEM_PAGE_SIZE` number of bytes from the ring buffer to the Flash page buffer and starts a Flash page write. It repeats this process a second time.

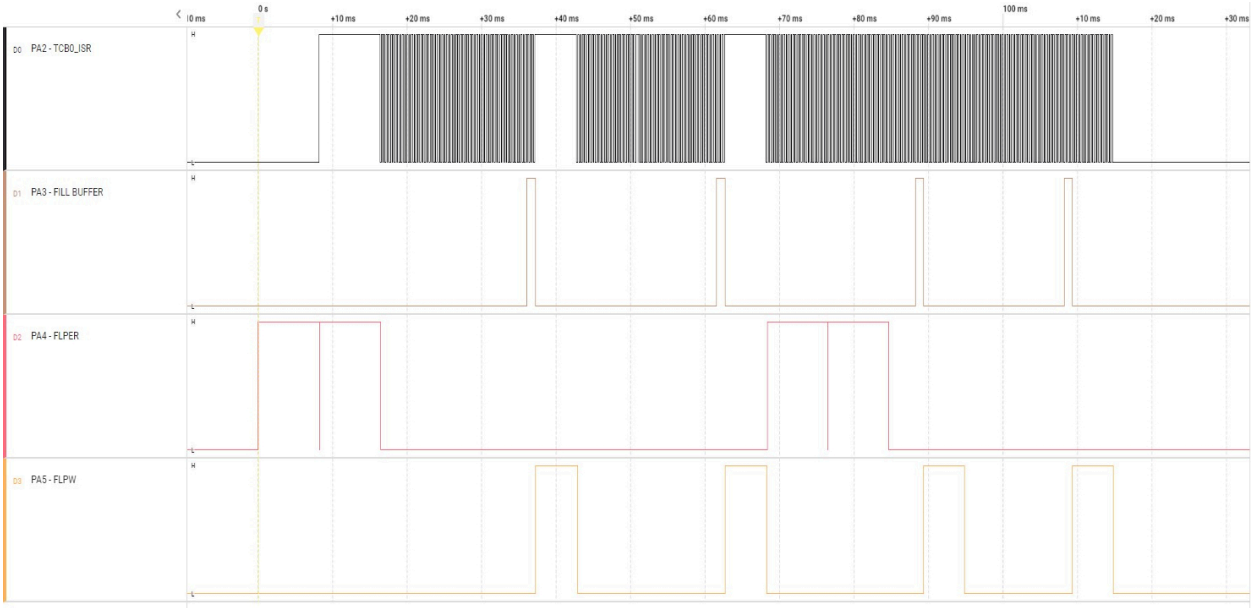
Writing to the Capture/Compare (CCMP) register in the TCB timer/counter changes the ISR period.

The code toggles some status pins, which makes it possible to scope the program flow:

- `PA2_ISR` toggles every time the TCB0 periodic interrupt ISR is handled
- `PA3_Buffer` is high when the Flash page buffer is being loaded
- `PA4_FLPER` is high while the Flash page erase is ongoing
- `PA5_FLPW` is high while the Flash page write is ongoing

As the application runs through its state machine, these pins will toggle, as seen in the image below. A more detailed description of this flow can be seen in the examples “README” file.

Figure 3-1. Pin Toggling Capture



Understanding The Code

© 2023 Microchip Technology Inc. and its subsidiaries

The linked code examples come in two flavors - an MPLAB X studio using MCC to generate drivers and a bare-metal project created in Microchip Studio. The projects have the same code flow and are divided into the following parts:

- Main file and function
- Timer 0 - Filling buffer
- Timer 1 - Switch debounce
- NVM controller

In addition to the parts above, to define the size of the BOOT and APPDATA sections set the fuses as shown below.

```
FUSES =
{
    .SYSCFG0 = FUSE_SYSCFG0_DEFAULT,
    .SYSCFG1 = FUSE_SYSCFG1_DEFAULT,

    .WDTCFG = FUSE_WDTCFG_DEFAULT,
    .BODCFG = FUSE_BODCFG_DEFAULT,
    .OSCCFG = FUSE_OSCCFG_DEFAULT,

    .BOOTSIZE = 31,     // 0x0000 -> 0x1F00 : BOOT Section
    .CODESIZE = 1,      // 0x01F00 -> FLASH_END : APPDATA Section
};
```

Most fuses are set to their default value, except BOOTSIZE, which is set to 31 blocks of 256 bytes and CODESIZE. As CODESIZE is set to 1, the program memory is logically split between the BOOT and APPDATA section.

Main file and setup

The main file of the system contains the initialization and main structure of the examples. The important things to notice are how the RWW/NRWW sections are defined and that the interrupt vector table is moved to the start of the BOOT section. Lastly, the mapped program memory points to the first section (the location of the .nrww_program, .nrww_data, and .rww_data sections). These initializations ensures that running code only exists in the program memory's NRWW section and that there exists memory addresses both in RWW and NRWW sections for data.

Timer 0

```
// TCB with capture interrupt
TCB0.INTCTRL = TCB_CAPT_bm;
TCB0.CCMP = TOP_VALUE;
TCB0.CTRLA = TCB_CLKSEL_DIV1_gc;
```

This timer is set up as a periodic interrupt that will trigger on capture when active. Note that the initialization does not enable the timer. The interrupt routine fills up the data buffer when called, as seen below.

```
// TCB0 capture interrupt
FillBuffer();
// Clear intflag
TCB0.INTFLAGS = TCB_CAPT_bm;
```

Timer 1

```
// TCB with periodic interrupt
TCB1.INTCTRL = TCB_CAPT_bm;
TCB1.CCMP = TCB1_TOP_VALUE;
TCB1.CTRLA = TCB_CLKSEL_DIV1_gc | TCB_ENABLE_bm;
```

The second timer is also set up to run periodic interrupts but is enabled all the time. The interrupt then runs the debounce function to avoid unwanted triggers.

```
// TCB1 capture interrupt
DebounceSW0();
```


© 2023 Microchip Technology Inc. and its subsidiaries

```
// Clear intflag
TCB1.INTFLAGS = TCB_CAPT_bm;
```

Erasing the Program Memory

```
NRWW_DATA_last_addr = ((((uint16_t) &nrww_array) & 0x7FFF) + MAPPED_PROGMEM_START +
DATA_SIZE);

while ((uint16_t)nrwwFlashPointer < NRWW_DATA_last_addr)
{
    _PROTECTED_WRITE_SPM(NVMCTRL.CTRLA, NVMCTRL_CMD_NOCMD_gc);

    // Perform a dummy write to this address to update the address register in NVMCTL
    *nrwwFlashPointer = 0;

    _PROTECTED_WRITE_SPM(NVMCTRL.CTRLA, NVMCTRL_CMD_FLPER_gc);

    while (NVMCTRL.STATUS & NVMCTRL_FLBUSY_bm)
    {
        ; // wait for Flash erase to complete
    }
    SCOPE_PORT.OUTCLR = SCOPE_FLPER_bm;

    nrwwFlashPointer = nrwwFlashPointer + PROGMEM_PAGE_SIZE;
}
```

The Flash erase operation for NRWW and RWW data sections is the same. It begins by setting the Flash page buffer to the area that will be erased. The FLPER pin is driven HIGH, so it's possible to see on the scope when the Flash erase operation is about to start. It sends NOCMD instructions to the NVM Controller to clear the previous NVM command to avoid a command collision error and is followed by a dummy write/store. The page erase command is sent to the NVM Controller. FLPER pin is driven LOW once the NVMCTRL.STATUS register shows that the Flash is not busy and the address of the next erase is incremented by a page size. The Flash erase operation is complete once the data section has been written to.

Writing the Program Memory

```
// Check if we have a full page to fill
if ((writeIndex - readIndex) >= PROGMEM_PAGE_SIZE)
{
    _PROTECTED_WRITE_SPM(NVMCTRL.CTRLA, NVMCTRL_CMD_NOCMD_gc);

    for (uint8_t i = 0; i < PROGMEM_PAGE_SIZE; i++)
    {
        *nrwwFlashPointer++ = buffer[readIndex++];
    }
    SCOPE_PORT.OUTCLR = SCOPE_BUFFER_bm;
    SCOPE_PORT.OUTSET = SCOPE_FLPW_bm;

    // Write the Flash page
    _PROTECTED_WRITE_SPM(NVMCTRL.CTRLA, NVMCTRL_CMD_FLPW_gc);
    while (NVMCTRL.STATUS & NVMCTRL_FLBUSY_bm)
    {
        ; // wait Flash write page operation to complete
    }
}
```

The Flash programming operation for NRWW and RWW data sections is the same. It begins by checking if enough data are available to write to a full Flash page. If not, it will wait until enough data have been generated. If yes, it sends NOCMD instructions to the NVM Controller to clear the previous NVM command to avoid a command collision error. Afterward, the BUFFER pin is driven HIGH, so it's possible to use the scope to see when the buffer load operation is about to start. Data is being read from the data buffer and stored in the Flash page buffer. When the equivalent of a complete page has been read and stored, the BUFFER pin is driven LOW. The FLPW pin is driven HIGH, and the page write command is sent to the NVM Controller. FLPW pin is driven LOW once the NVMCTRL.STATUS register shows that the Flash is not busy and the address of the next page write is incremented by a page size. The Flash write operation is complete once the data section has been written to.

© 2023 Microchip Technology Inc. and its subsidiaries

For the erase and write functions above, the code snippets are from the bare metal example and will be formatted differently for the MCC example. However, the functionality will be the same for each example.

Filling the buffer

```
void FillBuffer(void)
{
    // Fill the buffer with some data
    SCOPE_PORT.OUTTGL = SCOPE_ISR_bm;
    buffer[writeIndex++] = (data >> 2);
    data++;

    // Check if we have a buffer overflow
    if (writeIndex == readIndex)
    {
        // Overflow
        SCOPE_PORT.OUTTGL = SCOPE_OVERFLOW_bm;
    }
}
```

The FillBuffer function is triggered by the periodic interrupt on timer 0 when the timer compare counter reaches the set TOP value. The ISR pin toggles and the buffer is filled with data. OVERFLOW toggles if the buffer overflows.

SW0 Operation and Debouncing

The erase and write operation for both the NRWW and RWW data sections is started by a SW0 press. Subsequent SW0 presses will trigger the same process. The value of the last data location in the RWW data section is used as a seed for the new data. Due to this behavior, when SW0 is pressed and the program memory read, the user can observe a change in NRWW and RWW data sections. As a confirmation that SW0 has been pressed, LED0 lights up. It is necessary to trigger the whole operation of the example by a button press, as one needs to set up the external instruments (oscilloscope and/or logic analyzer) that can capture the toggling of the instrumentation pins. A simple debounce algorithm has been chosen and implemented. TCB1 is being employed for this purpose. Software debouncing the physical switch SW0 prevents the unwanted triggering of the erase and write operations. Please note that the programming Flash has a limited erase/write endurance beyond which operation is not guaranteed (consult the device's electrical characteristics for further information and details).

Moving the interrupt vector table

The vector interrupt placement depends on the Interrupt Vector Select (IVSEL) bit value in the Control A (CPUINT.CTRLA) register. If using no interrupt source, regular program code may be placed in these locations. By writing 1 to the IVSEL, the interrupt vector is placed at the start of the BOOT section of the Flash, enabling the interrupts to run as normal while an erase/write operation in the RWW section is running. As stated in the overview section, the syntax "RWW section" refers to which section is being programmed (erased or written) and not the one read. Only code located inside the NRWW Flash can be accessed by either executing a CPU instruction or reading data while the RWW section is being programmed.

© 2023 Microchip Technology Inc. and its subsidiaries

4. Get Code Examples from MPLAB Discover

MPLAB Discover bundles the available resources from multiple sources into one portal.

The MPLAB Discover webpage: MPLAB Discover

Code Examples

- Example code for Microchip Studio
- Example code for MPLAB X

You can download code examples as a `.zip` file directly from MPLAB Discover. Repositories ending with "*studio*" can be opened in Microchip Studio. Repositories ending with "*mplab-mcc*" can be opened in MPLAB X.

When the code example is hosted on GitHub, MPLAB Discover provides a link "*Open with Github*". There you can download the example code or use the `git` tool on your PC to create a local repository clone.

© 2023 Microchip Technology Inc. and its subsidiaries

5. Revision History

Document Revision	Date	Comments
A	10/2023	Initial document release

© 2023 Microchip Technology Inc. and its subsidiaries

Microchip Information

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable". Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure

© 2023 Microchip Technology Inc. and its subsidiaries

that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, TimeCesium, TimeHub, TimePictra, TimeProvider, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, Clockstudio, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, EyeOpen, GridTime, IdealBridge, IGaT, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, IntelliMOS, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, MarginLink, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, mSiC, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, Power MOS IV, Power MOS 7, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SmartHLS, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, Trusted Time, TSHARC, Turing, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

© 2023 Microchip Technology Inc. and its subsidiaries

All other trademarks mentioned herein are property of their respective companies.

© 2023, Microchip Technology Incorporated and its subsidiaries. All Rights Reserved.

ISBN: 978-1-6683-3305-1

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, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

© 2023 Microchip Technology Inc. and its subsidiaries

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office	**Australia - Sydney**	**India - Bangalore**	**Austria - Wels**
2355 West Chandler Blvd.	Tel: 61-2-9868-6733	Tel: 91-80-3090-4444	Tel: 43-7242-2244-39
Chandler, AZ 85224-6199	**China - Beijing**	**India - New Delhi**	Fax: 43-7242-2244-393
Tel: 480-792-7200	Tel: 86-10-8569-7000	Tel: 91-11-4160-8631	**Denmark - Copenhagen**
Fax: 480-792-7277	**China - Chengdu**	**India - Pune**	Tel: 45-4485-5910
Technical Support:	Tel: 86-28-8665-5511	Tel: 91-20-4121-0141	Fax: 45-4485-2829
www.microchip.com/support	**China - Chongqing**	**Japan - Osaka**	**Finland - Espoo**
Web Address:	Tel: 86-23-8980-9588	Tel: 81-6-6152-7160	Tel: 358-9-4520-820
www.microchip.com	**China - Dongguan**	**Japan - Tokyo**	**France - Paris**
Atlanta	Tel: 86-769-8702-9880	Tel: 81-3-6880- 3770	Tel: 33-1-69-53-63-20
Duluth, GA	**China - Guangzhou**	**Korea - Daegu**	Fax: 33-1-69-30-90-79
Tel: 678-957-9614	Tel: 86-20-8755-8029	Tel: 82-53-744-4301	**Germany - Garching**
Fax: 678-957-1455	**China - Hangzhou**	**Korea - Seoul**	Tel: 49-8931-9700
Austin, TX	Tel: 86-571-8792-8115	Tel: 82-2-554-7200	**Germany - Haan**
Tel: 512-257-3370	**China - Hong Kong SAR**	**Malaysia - Kuala Lumpur**	Tel: 49-2129-3766400
Boston	Tel: 852-2943-5100	Tel: 60-3-7651-7906	**Germany - Heilbronn**
Westborough, MA	**China - Nanjing**	**Malaysia - Penang**	Tel: 49-7131-72400
Tel: 774-760-0087	Tel: 86-25-8473-2460	Tel: 60-4-227-8870	**Germany - Karlsruhe**
Fax: 774-760-0088	**China - Qingdao**	**Philippines - Manila**	Tel: 49-721-625370
Chicago	Tel: 86-532-8502-7355	Tel: 63-2-634-9065	**Germany - Munich**
Itasca, IL	**China - Shanghai**	**Singapore**	Tel: 49-89-627-144-0
Tel: 630-285-0071	Tel: 86-21-3326-8000	Tel: 65-6334-8870	Fax: 49-89-627-144-44
Fax: 630-285-0075	**China - Shenyang**	**Taiwan - Hsin Chu**	**Germany - Rosenheim**
Dallas	Tel: 86-24-2334-2829	Tel: 886-3-577-8366	Tel: 49-8031-354-560
Addison, TX	**China - Shenzhen**	**Taiwan - Kaohsiung**	**Israel - Ra'anana**
Tel: 972-818-7423	Tel: 86-755-8864-2200	Tel: 886-7-213-7830	Tel: 972-9-744-7705
Fax: 972-818-2924	**China - Suzhou**	**Taiwan - Taipei**	**Italy - Milan**
Detroit	Tel: 86-186-6233-1526	Tel: 886-2-2508-8600	Tel: 39-0331-742611
Novi, MI	**China - Wuhan**	**Thailand - Bangkok**	Fax: 39-0331-466781
Tel: 248-848-4000	Tel: 86-27-5980-5300	Tel: 66-2-694-1351	**Italy - Padova**
Houston, TX	**China - Xian**	**Vietnam - Ho Chi Minh**	Tel: 39-049-7625286
Tel: 281-894-5983	Tel: 86-29-8833-7252	Tel: 84-28-5448-2100	**Netherlands - Drunen**
Indianapolis	**China - Xiamen**		Tel: 31-416-690399
Noblesville, IN	Tel: 86-592-2388138		Fax: 31-416-690340
Tel: 317-773-8323	**China - Zhuhai**		**Norway - Trondheim**
Fax: 317-773-5453	Tel: 86-756-3210040		Tel: 47-72884388
Tel: 317-536-2380			**Poland - Warsaw**
Los Angeles			Tel: 48-22-3325737
Mission Viejo, CA			**Romania - Bucharest**
Tel: 949-462-9523			Tel: 40-21-407-87-50
Fax: 949-462-9608			**Spain - Madrid**
Tel: 951-273-7800			Tel: 34-91-708-08-90
Raleigh, NC			Fax: 34-91-708-08-91
Tel: 919-844-7510			**Sweden - Gothenberg**
New York, NY			Tel: 46-31-704-60-40
Tel: 631-435-6000			**Sweden - Stockholm**
San Jose, CA			Tel: 46-8-5090-4654
Tel: 408-735-9110			**UK - Wokingham**
Tel: 408-436-4270			Tel: 44-118-921-5800
Canada - Toronto			Fax: 44-118-921-5820
Tel: 905-695-1980			
Fax: 905-695-2078			

© 2023 Microchip Technology Inc. and its subsidiaries