AN1982 Interrupt System in tinyAVR 0- and 1-series, and megaAVR 0-series

Features

- Interrupt Controller Overview
- Interrupt Priority Configuration
- Vector Table Configuration
- Code Examples

Introduction

Microcontrollers (MCUs) contain a wide range of hardware modules (peripherals) designed to perform specialized tasks like communication, timing and waveform generation. Typically, each peripheral has a number of signals indicating its status, an operation is completed, data availability, or the peripherals is ready to receive new commands. It is up to the central processing unit (CPU) to check each module for relevant updates. The CPU then has to service the requests from each module. With an increasing number of peripheral modules, the CPU spends increasingly more time to check and service all modules. This CPU load causes longer response time and higher power consumption. Traditionally, MCUs use two main methods to monitor and service peripherals: polling and interrupts.

Polling means manually reading and checking a status bit updated by a monitored peripheral. This gives the designer a high degree of freedom to decide how often and in which order to check the different peripherals. In a simple use case where the application only waits for a specific status, a very tight loop can be made; checking this status bit and immediately servicing it when it occurs. Even though this approach would allow a very fast response time, there are few major drawbacks using the polling method. First, the responsiveness decreases as the number of status bits to check increases. Second, the CPU needs to run in active mode while it is executing the code testing every status bit, which increases power consumption.

As the number of peripherals and status bits involved in an application increases, an interrupt system provides better responsiveness. An interrupt system is a scheme in which the peripherals can send a request to interrupt the CPU execution. Since the peripherals prompt the CPU when service is needed, there is no need for the CPU to actively poll the status bits. However, when a peripheral can request service from the CPU at any time, with no regard to other peripherals' needs, this creates the risk of an interrupt request pile-up. Therefore, an interrupt controller is used to determine the order in which pending interrupts will be serviced by the CPU. Traditionally, interrupt handling for tinyAVR® and megaAVR® devices uses a predefined priority, based on the interrupt vector address. The XMEGA® MCU family makes it possible to tailor the priority queue using a Programmable Multilevel Interrupt Controller (PMIC), which allows the user to assign interrupts to three priority levels. Interrupt handling techniques have become more configurable in the tinyAVR 0- and 1-series, and megaAVR 0-series. This application note describes the functionality of the new interrupt controller in detail, which uses a combination of predefined priority and multilevel controller.
1. Relevant Devices
This chapter lists the relevant devices for this document.

1.1 tinyAVR 1-Series
The figure below shows the tinyAVR 1-series devices, illustrating pin count variants and memory sizes:

- Vertical migration can be done upwards without code modification, since these devices are pin compatible and provide the same or even more features. Downward migration may require code modification due to fewer available instances of some peripherals.
- Horizontal migration to the left reduces the pin count, therefore, the available features.

Figure 1-1. tinyAVR® 1-Series Device Overview

![tinyAVR 1-Series Diagram]

Devices with different Flash memory size typically also have different SRAM and EEPROM.

1.2 megaAVR 0-Series
Figure 1-2 shows the feature compatible devices in the megaAVR device family, including pinout variants and memory variants.

Migration within the vertical direction can be done without modifications to the code, as these devices are fully pin and feature compatible.

Migration in the horizontal direction will introduce a change in pin count and therefore also in the available features. The peripherals are however fully compatible in the horizontal direction as well. It is just a matter of how many instances of the peripherals are present in a device with more or less pins.
The fully compatible variants of the megaAVR devices, that is the vertical migration option in Figure 1-2, come with both smaller and larger Flash memories.

Devices with different Flash memory size typically also have different SRAM and EEPROM.
2. **Interrupt Controller Overview**

The tinyAVR 0- and 1-series, and megaAVR 0-series Interrupt Controller has support for multiple individual Interrupt Requests (IRQs), with dedicated interrupt vectors. This helps reduce latency when an interrupt is triggered.

When an interrupt is enabled and the interrupt condition occurs, the CPU will receive an IRQ from the Interrupt Controller. Based on the interrupt's priority level and the priority level of any ongoing Interrupt Service Routine (ISR), the IRQ is either acknowledged (and its ISR is executed) or kept pending. See Figure 2-1 for a graphical representation of the Interrupt Controller interaction.

**Figure 2-1. Interrupt Controller Flowchart**

The Interrupt Controller decides the order of execution for pending ISRs by the combination of the interrupts priority level and vector address. The hardwired vector addresses and different priority modes are configurable from the CPUINT registers, with the following available priority levels:

1. Non-Maskable Interrupts
2. Level 1 Priority Interrupts
3. Level 0 Priority Interrupts

Refer to the device datasheet for the interrupt vector mapping.

**Interrupt Flags**

The Global Interrupt Enable flag (I-flag) must be set for IRQs to be generated, with Non-Maskable Interrupts as the only exception. Information on these can be found under Non-Maskable Interrupts.

The majority of interrupt flags must be cleared in the corresponding ISR. This is because some interrupt sources share interrupt vectors. Other flags are cleared by hardware when entering the ISR or reading a data register. For detailed information, refer to each peripheral's chapter in the device datasheet.

**Interrupt Latency**

The CPU's interrupt latency is determined by the number of clock cycles required for the following actions:

1. Complete the ongoing instruction.
   - 1 - 3 clock cycles, dependent on instruction time.
2. Push the program counter on the stack.
3. Execute the jump instruction stored in the interrupt vector table.
   - 2 - 3 clock cycles, dependent on Flash size. See Table 2-1.

Table 2-1. Interrupt Jump Instruction

<table>
<thead>
<tr>
<th>Flash size</th>
<th>Jump instruction</th>
<th>Jump Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ 8kB</td>
<td>RJMP</td>
<td>2 clock cycles</td>
</tr>
<tr>
<td>&gt; 8kB</td>
<td>JMP</td>
<td>3 clock cycles</td>
</tr>
</tbody>
</table>

The total interrupt latency will be five to eight clock cycles, depending on the parameters listed above. See the AVR Instruction Set Manual for more information.

If the device is in Sleep mode, the response time increases by five clock cycles, in addition to the CPU start-up time from the used Sleep mode.
3. Static Priority Interrupts

The tinyAVR 0- and 1-series, and megaAVR 0-series by default use an interrupt vector table with static priority. This is identical to other classic tinyAVR® and megaAVR® MCUs and for most standard applications this is the preferred method of operation.

The Static Priority Interrupt scheme is applicable for priority level 0 interrupts only. Information on higher level interrupts is located in the High Priority Interrupt Vector and Non-Maskable Interrupts chapters.

3.1 Operation

When using a static priority interrupt vector table, the lowest interrupt vector address has the highest priority. This means, when two or more interrupts are pending, the order of execution will be set by the interrupt vector address, in ascending order. See Figure 3-1 for a graphical representation.

While the CPU is executing an ISR, any new IRQs will be kept pending until the running routine is complete, and at that time the interrupt with the lowest vector address (highest priority) will be executed first.

Figure 3-1. Interrupt priority using Static Priority Scheme

![Diagram showing interrupt priority using Static Priority Scheme]

In Static Priority Interrupt Scheme the CPUINT.LVL0PRI register defines the starting point for the highest priority interrupt, so it is possible to statically shift the priority in the vector table by writing the desired interrupt vector address to LVL0PRI. The default value of this register is 0x00, so after reset the lowest address interrupt vector will have highest priority.

3.2 Example

Static Priority Interrupt scheme is the default setting for the interrupt controller in tinyAVR 0- and 1-series, and megaAVR 0-series devices, so the CPUINT registers are configured for this at startup. The following code snippet shows an example on how to configure the Static Priority Interrupt scheme.
Static Priority Interrupt scheme configuration

// io.h includes the device header file with register and vector defines
#include <avr/io.h>
// interrupt.h contains the ISR related content
#include <avr/interrupt.h>

void static_priority_interrupt_example(void) {
  // If needed, clear the Round-robin Scheduling Enable bit.
  CPUINT.CTRLA &= ~CPUINT_LVL0RR_bm;

  // Example on how to shift the interrupt priority.
  CPUINT.LVL0PRI = PORTA_PORT_vect_num;

  // Enable global interrupts
  sei();
}

ISR(PORTA_PORT_vect){
  // This interrupt will have highest priority.
}
4. Round-Robin Priority Scheme

In some cases where Static Priority Interrupt schemes are used, interrupt starvation may occur. This is when applications have higher priority (lower vector address) interrupts that are triggered too often, causing a lack of service for the lower priority interrupts. The Round-Robin Priority Scheme aims to prevent interrupt starvation.

The Round-Robin Priority Scheme is valid for priority level 0 interrupts. Descriptions of higher level interrupts are located under the High Priority Interrupt Vector and Non-Maskable Interrupts chapters.

4.1 Operation

When using the Round-Robin Priority Scheme, the interrupt controller will store the address of the latest acknowledged interrupt; in the event that more than one interrupt request is pending, the stored address will have the lowest priority when the next interrupt is due to be serviced. Consequently, the next higher interrupt vector address will be assigned the highest priority. See Figure 4-1 for a graphical representation of the scheme.

**Figure 4-1. Interrupt priority using Round-Robin Interrupt Scheme**

The Round-Robin Priority Scheme is enabled by writing ‘1’ to the LVL0RR bit in the CPUINT.CTRLA register. The CPUINT.LVL0PRI register will contain the address of the last acknowledged interrupt. This becomes the lowest priority interrupt when the interrupt controller decides which interrupt to execute next.

By changing the value in CPUINT.LVL0PRI it is possible for the CPU to change which interrupts have the lowest and highest priority next. The default value of this register is 0x00, so after reset the lowest interrupt vector will have highest priority.

4.2 Example

The following code snippet shows an example on how to configure the Round-robin Scheduling scheme.
Round-robin Interrupt Scheduling configuration

```c
#include <avr/io.h>
#include <avr/interrupt.h>

void round_robin_interrupt_scheduling_example(void) {
    // Set the Round-robin Scheduling Enable bit.
    CPUINT.CTRLA |= CPUINT_LVL0RR_bm;

    // Example on how to shift the initial interrupt priority.
    CPUINT.LVL0PRI = PORTA_PORT_vect_num;

    // Enable global interrupts
    sei();
}

ISR(PORTA_PORT_vect){
    // This interrupt will initially have lowest priority, however
    // that will change since the LVL0PRI will be updated by the interrupt
    // controller each time an interrupt is executed.
}
```

Atmel | START example
A use case example for Round-robin Interrupt Scheduling using Atmel | START is also available. For more information, see the Get Source Code from Atmel | START chapter.
5. **High Priority Interrupt Vector**

When a higher interrupt priority is required over what is already defined, one interrupt vector can be assigned priority level 1. That interrupt will have priority over all level 0 interrupts, and has the ability to interrupt an ongoing level 0 interrupt handler.

All interrupt sources are by default assigned level 0 priority, except for Non-Maskable Interrupts (NMI). These are the only interrupts that have higher priority than the level 1 interrupt. For more info see the Non-Maskable Interrupts chapter.

5.1 **Operation**

To use the high priority vector feature, simply write the desired interrupt vector address to the CPUINT.LVL1VEC register and this interrupt will be assigned level 1 priority. This feature is disabled when the register value is 0x00. Refer to the device datasheet for the interrupt vector mapping.

If interrupted by the level 1 interrupt, the level 0 handler will resume execution when the level 1 handler is done. It is important to make sure a level 1 interrupt will not interfere with the execution of any time critical level 0 interrupts. The LVL0EX bit in the CPUINT.STATUS register can be read to check if the level 1 interrupt has interrupted an executing level 0 ISR.

5.2 **Example**

The following code snippet shows an example on how to configure the High Priority Interrupt vector.

```c
#include <avr/io.h>
#include <avr/interrupt.h>

void high_priority_interrupt_example(void) {
    // Set the level 1 interrupt vector
    CPUINT.LVL1VEC = PORTA_PORT_vect_num;
    // Enable global interrupts
    sei();
}

ISR(PORTA_PORT_vect) {
    // This interrupt will have level 1 priority
    // after the configuration in CPUINT.LVL1VEC.

    // Check if a level 0 ISR has been interrupted
    if (CPUINT.STATUS & CPUINT_LVL0EX_bm) {
    }
}
```
6. **Non-Maskable Interrupts**

Non-Maskable Interrupts (NMI) are usually system critical interrupts and will always take priority over all other interrupts. Also, they will be acknowledged even when global interrupts are disabled.

6.1 **Operation**

There are specific vectors that have a hardwired NMI priority level, which cannot be changed. They must be enabled the same way as other interrupts. Refer to the Interrupt Vector Mapping of the device datasheet for available NMI sources.

By reading the LVL0EX and LVL1EX bits in the CPUINT.STATUS register, the application can check whether any level 0 or level 1 ISR has been interrupted by an NMI.

6.2 **Example**

The following code snippet demonstrates how a Non-Maskable Interrupt can be implemented.

```c
// io.h includes the device header file with register and vector defines
#include <avr/io.h>
// interrupt.h contains the ISR related content
#include <avr/interrupt.h>

void nmi_example(void) {
    // Configure the CRCSCAN peripheral with NMI
    CRCSCAN.CTRLB = CRCSCAN_MODE_PRIORITY_gc | CRCSCAN_SRC_FLASH_gc;
    CRCSCAN.CTRLA = CRCSCAN_NMIEN_bm | CRCSCAN_ENABLE_bm;
}

ISR(CRCSCAN_NMI_vect) { // If the CRC scan fails, the NMI interrupt will be handled here
    // The NMI request remains active until a system Reset, and can not be disabled
    // Check if level 0 or 1 ISRs were interrupted
    if (CPUINT.STATUS & (CPUINT_LVL0EX_bm | CPUINT_LVL1EX_bm)) {
    }
}
```

For more information on how to prepare a code project for using the CRCSCAN peripheral, refer to Application Note AN2521 - CRCSCAN on Devices in the tinyAVR® 1-Series
7. **Compact Vector Table**

If an application is using only a few interrupts, only those vector locations are needed; therefore, most of the vector table will be unused space. The Compact Vector Table (CVT) is an approach to a more efficient interrupt management scheme, since this will truncate the interrupt vector table to a few vectors.

7.1 **Operation**

The tinyAVR 0- and 1-series, and megaAVR 0-series MCUs support a three-vector CVT mode, where the interrupt vector table is reorganized per Table 7-1. Each vector will be used to service all interrupts of corresponding priority level.

**Table 7-1. Compact Vector Mapping**

<table>
<thead>
<tr>
<th>Vector Number</th>
<th>Peripheral Source</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RESET</td>
<td>RESET</td>
</tr>
<tr>
<td>1</td>
<td>NMI</td>
<td>Non-Maskable Interrupt vector</td>
</tr>
<tr>
<td>2</td>
<td>LVL1</td>
<td>Level 1 Interrupt vector</td>
</tr>
<tr>
<td>3</td>
<td>LVL0</td>
<td>Level 0 Interrupt vector</td>
</tr>
</tbody>
</table>

This can be very useful when an application only needs a few interrupts. Reducing the vector table size means it will occupy less Flash, freeing up space for application code. Be aware however, when using this approach, additional code may be needed in the ISR to find out which interrupt is actually requesting service as this will add execution time.

The Compact Vector Table is enabled by writing '1' to the CVT bit in CPUINT.CTRLA. This system critical register bit has Configuration Change Protection (CCP) to prevent accidental modification. Refer to the CPU chapter in the relevant device datasheet for details on CCP.

7.2 **Example**

To take advantage of the smaller vector table it is necessary to replace the standard start files used by AVR-GCC. These are disabled by setting the linker flag `-nostartfiles` when compiling the project. In Atmel Studio 7.0 this can be found in Project Properties (Alt+F7) - Toolchain - AVR/GNU Linker - General, as seen in Figure 7-1.
The following code snippet shows an example for the code needed to replace the standard start files. In a future Toolchain release this step will not be needed to configure the Compact Vector Table.

```c
// io.h includes the device header file with necessary defines
#include <avr/io.h>

// Letting the compiler know there will be something called main
extern int main(void);

// Being nice to the compiler by predeclaring the ISRs
void __vector_cvt_nmi(void) __attribute__((weak, alias("dummy_handler")));
void __vector_cvt_lvl1(void) __attribute__((weak, alias("dummy_handler")));
void __vector_cvt_lvl0(void) __attribute__((weak, alias("dummy_handler")));

// Setting up the vector section
// The rjmp instruction can handle 8k code space, so this is used for
// vector tables on devices with 8k flash or smaller. Other devices
// use the jmp instruction.
__attribute__((section(".vectors"), naked)) void vectors(void)
{
    #if (PROGMEM_SIZE <= 0x2000)

```
The following code snippet shows an example on how to configure the Compact Vector Table, using the interrupt vector names defined in the example startup code from above.

```c
#include <avr/io.h>
#include <avr/interrupt.h>

void compact_vector_table_example(void) {
  // Set the Compact Vector Table
  // read-modify-write to preserve other configured bits
  _PROTECTED_WRITE(CPUINT.CTRLA, (CPUINT.CTRLA | CPUINT_CVT_bm));

  //Enable global interrupts
  sei();
}

ISR(__vector_cvt_lvl0) {
  // All level 0 interrupt will be handled here
  if (PORTA.INTFLAGS) {
    // Example on how to handle PORTA interrupts in CVT mode
  }
}

ISR(__vector_cvt_lvl1) {
  // If enabled, level 1 interrupt will be handled here
```
Atmel | START example
A use case example for Compact Vector Table using Atmel | START is also available. For more information, see the Get Source Code from Atmel | START chapter.
8. Relocating Vector Table

By default, the vector table is located at the start of Flash in the tinyAVR 0- and 1-series, and megaAVR 0-series devices. However, it is possible to relocate the vector table to the start of the application section; a bootloader can use the default vector table location in its code, and then relocate the vector table when starting the main application. This way the bootloader and application code can have separate interrupt vector tables.

It is also possible to combine the relocation of the vector table with use of the Compact Vector Table (CVT), for more information on the CVT see the Compact Vector Table chapter.

8.1 Operation

Moving the interrupt vector table to the start of the application section of Flash is enabled by writing ‘1’ to the IVSEL bit in CPUINT.CTRLA. This system critical register bit has Configuration Change Protection (CCP) to prevent accidental modification. Refer to the CPU chapter in the relevant device data sheet for details on CCP.

Since the application section is placed after the boot section, the actual address of the interrupt vector table is determined by the BOOTEND fuse. This fuse sets the size of the boot section, in blocks of 256 bytes. When IVSEL is cleared, the interrupt vector table will be at the start address of the Flash. Be aware that if both the BOOTEND fuse and APPEND fuse (size of application section) are set to 0x00 the entire Flash is configured to be boot, and the IVSEL bit will not move the vector table.

The Reset vector address is 0x0000 (Start of Flash) even if the IVSEL bit is set. This corresponds with the lowest address in the boot interrupt vector table, which means the Reset vector will be the first instruction to execute after a reset has occurred.

8.2 Example

This example will use a bootloader section with 512 bytes size. This means the BOOTEND fuse needs to be configured to 512/256 = 0x02. In Atmel Studio 7.0 this fuse can be written to the device using Device Programming (Ctrl+Shift+P) - Fuses, as seen in Figure 8-1.
The AVR-GCC project must also be configured to shift the start of the application section by 512 bytes. This is done by changing the start of the text segment in the linker, and this value must be set in hexadecimal word size, for example: `-Wl,-section-start=bootloader=0x100`. In Atmel Studio 7.0 this can be found in Project Properties (Alt+F7) - Toolchain - AVR/GNU Linker - Memory Settings and adding `.text=0x100` to the FLASH segment, as seen in Figure 8-2.
The following code example shows how to relocate the vector table.

```c
#include <avr/io.h>
#include <avr/interrupt.h>

void relocating_vector_table_example(void) {
    // Set the Interrupt Vector Select bit,
    // read-modify-write to preserve other configured bits
    _PROTECTED_WRITE(CPUINT.CTRLA, (CPUINT.CTRLA | CPUINT_IVSEL_bm));
    //Enable global interrupts
    sei();
}

ISR(PORTA_PORT_vect){
    // After setting the IVSEL bit, the interrupt vector
    // will now be shifted by BOOTEND*256 bytes.
    // If the .text section in Flash starts at BOOTEND*128 words,
    // the ISRs will be executed correctly.
}
```

Relocating Vector Table configuration

// io.h includes the device header file with register and vector defines,
// it also includes xmega.h, containing the CCP macro _PROTECTED_WRITE().
#include <avr/io.h>
// interrupt.h contains the ISR related content
#include <avr/interrupt.h>

void relocating_vector_table_example(void) {
    // Set the Interrupt Vector Select bit,
    // read-modify-write to preserve other configured bits
    _PROTECTED_WRITE(CPUINT.CTRLA, (CPUINT.CTRLA | CPUINT_IVSEL_bm));
    //Enable global interrupts
    sei();
}

ISR(PORTA_PORT_vect){
    // After setting the IVSEL bit, the interrupt vector
    // will now be shifted by BOOTEND*256 bytes.
    // If the .text section in Flash starts at BOOTEND*128 words,
    // the ISRs will be executed correctly.
}
9. Summary

The choice between polling and interrupts depends heavily on the application requirements, however, it is possible to follow a few guidelines.

First, if your product is battery powered or otherwise power restricted, avoid polling. Polling requires the CPU to stay in Active mode, potentially consuming a lot more power than an interrupt driven system. However, if increased power consumption is not an issue, and only one or two status bits need checking, polling is very easy to implement in software.

Second, it can be difficult to manage and maintain a complex application in an effective and predictable way. This is made easier with the available interrupt priority schemes and the possibility to elevate one interrupt to a higher priority level.

Third, if an application is restricted in code size, this can be reduced by using the compact vector table. In some cases, it could be advantageous to use a compact vector table only for the bootloader, and then switch back to the full size table in the application section. This could potentially reduce boot section size, leaving more space for the application section.

As shown in this application note, the new tinyAVR 0- and 1-series, and megaAVR 0-series MCUs offer powerful flow control features with respect to interrupt handling. This allows designers to customize power consumption and responsiveness to fit their design requirements.
10. **Related Documents**

The following documents are related to the devices and topics covered by this application note.

- AVR Instruction Set Manual:
- AN2521 - CRCSCAN on Devices in the tinyAVR® 1-Series:
11. **Get Source Code from Atmel | START**

The example code is available through Atmel | START, which is a web-based tool that enables configuration of application code through a Graphical User Interface (GUI). The code can be downloaded for both Atmel Studio 7.0 and IAR Embedded Workbench® via the direct example code-link(s) below or the *BROWSE EXAMPLES* button on the Atmel | START front page.

Atmel | START web page: [http://microchip.com/start](http://microchip.com/start)

**Example Code**

- **CPUINT Round-Robin Interrupt Scheme:**
  - [http://start.atmel.com/#example/Atmel:cpuint_examples_tinyavr_megaavr_01:1.0.0::Application:CPUINT_Round-Robin_Scheme](http://start.atmel.com/#example/Atmel:cpuint_examples_tinyavr_megaavr_01:1.0.0::Application:CPUINT_Round-Robin_Scheme)

- **CPUINT Compact Vector Table:**
  - [http://start.atmel.com/#example/Atmel:cpuint_examples_tinyavr_megaavr_01:1.0.0::Application:CPUINT_Compact_Vector_Table](http://start.atmel.com/#example/Atmel:cpuint_examples_tinyavr_megaavr_01:1.0.0::Application:CPUINT_Compact_Vector_Table)

Press *User guide* in Atmel | START for details and information about example projects. The *User guide* button can be found in the example browser, and by clicking the project name in the dashboard view within the Atmel | START project configurator.

**Atmel Studio**

Download the code as an .atzip file for Atmel Studio from the example browser in Atmel | START, by clicking *DOWNLOAD SELECTED EXAMPLE*. To download the file from within Atmel | START, click *EXPORT PROJECT* followed by *DOWNLOAD PACK*.

Double-click the downloaded .atzip file and the project will be imported to Atmel Studio 7.0.

**IAR Embedded Workbench**

For information on how to import the project in IAR Embedded Workbench, open the Atmel | START user guide, select *Using Atmel Start Output in External Tools*, and *IAR Embedded Workbench*. A link to the Atmel | START user guide can be found by clicking *About* from the Atmel | START front page or *Help And Support* within the project configurator, both located in the upper right corner of the page.
## 12. Revision History

<table>
<thead>
<tr>
<th>Doc. Rev.</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12/2017</td>
<td>Initial document release</td>
</tr>
</tbody>
</table>
The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **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 (FAQ), technical support requests, online discussion groups, Microchip consultant 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

Customer Change Notification Service

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


Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

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

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip’s Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

**Legal Notice**

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. 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 ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. 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, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, KeeLoq logo, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntellIMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.


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

Silicon Storage Technology is a registered trademark 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.

All other trademarks mentioned herein are property of their respective companies.
Quality Management System Certified by DNV

ISO/TS 16949
Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company’s quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001:2000 certified.
## AMERICAS

<table>
<thead>
<tr>
<th>City</th>
<th>Address</th>
<th>Phone Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta</td>
<td>Duluth, GA</td>
<td>Tel: 678-957-9614, Fax: 678-957-1455</td>
</tr>
<tr>
<td>Austin, TX</td>
<td>Tel: 512-257-3370</td>
<td></td>
</tr>
<tr>
<td>Boston</td>
<td>Westborough, MA</td>
<td>Tel: 774-760-0087, Fax: 774-760-0088</td>
</tr>
<tr>
<td>Dallas</td>
<td>Tel: 214-308-0075</td>
<td></td>
</tr>
<tr>
<td>Denver</td>
<td>Tel: 720-720-0075</td>
<td></td>
</tr>
<tr>
<td>Detroit</td>
<td>Novi, MI</td>
<td>Tel: 248-848-4000</td>
</tr>
<tr>
<td>Houston, TX</td>
<td>Tel: 281-894-5983</td>
<td></td>
</tr>
<tr>
<td>Indianapolis</td>
<td>Noblesville, IN</td>
<td>Tel: 317-773-8323, Fax: 317-773-5453</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>Tel: 310-536-2380</td>
<td></td>
</tr>
<tr>
<td>Raleigh, NC</td>
<td>Tel: 919-844-7510</td>
<td></td>
</tr>
<tr>
<td>New York, NY</td>
<td>Tel: 631-435-6000</td>
<td></td>
</tr>
<tr>
<td>San Jose, CA</td>
<td>Tel: 408-735-9110, 408-436-4270</td>
<td></td>
</tr>
<tr>
<td>Canada - Toronto</td>
<td>Tel: 905-695-1980, 905-695-2078</td>
<td></td>
</tr>
</tbody>
</table>

## ASIA/PACIFIC

<table>
<thead>
<tr>
<th>Country</th>
<th>City</th>
<th>Phone Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australia</td>
<td>- Sydney</td>
<td>Tel: 61-2-9868-6733</td>
</tr>
<tr>
<td>China</td>
<td>- Beijing</td>
<td>Tel: 86-10-8569-7000</td>
</tr>
<tr>
<td>China</td>
<td>- Chengdu</td>
<td>Tel: 86-28-8665-5511</td>
</tr>
<tr>
<td>China</td>
<td>- Chongqing</td>
<td>Tel: 86-23-8980-9588</td>
</tr>
<tr>
<td>China</td>
<td>- Dongguan</td>
<td>Tel: 86-769-8702-9880</td>
</tr>
<tr>
<td>China</td>
<td>- Guangzhou</td>
<td>Tel: 86-20-8755-8029</td>
</tr>
<tr>
<td>China</td>
<td>- Hangzhou</td>
<td>Tel: 86-571-8792-8115</td>
</tr>
<tr>
<td>China</td>
<td>- Hong Kong SAR</td>
<td>Tel: 852-2943-5100</td>
</tr>
<tr>
<td>China</td>
<td>- Nanjing</td>
<td>Tel: 86-25-8473-2460</td>
</tr>
<tr>
<td>China</td>
<td>- Qingdao</td>
<td>Tel: 86-632-8502-7355</td>
</tr>
<tr>
<td>China</td>
<td>- Shanghai</td>
<td>Tel: 86-21-3326-8000</td>
</tr>
<tr>
<td>China</td>
<td>- Shenzhen</td>
<td>Tel: 86-24-2334-2829</td>
</tr>
<tr>
<td>China</td>
<td>- Suzhou</td>
<td>Tel: 86-186-6233-1526</td>
</tr>
<tr>
<td>China</td>
<td>- Wuhan</td>
<td>Tel: 86-27-5980-5300</td>
</tr>
<tr>
<td>China</td>
<td>- Xian</td>
<td>Tel: 86-29-9833-2752</td>
</tr>
<tr>
<td>China</td>
<td>- Xiamen</td>
<td>Tel: 86-592-2388138</td>
</tr>
<tr>
<td>China</td>
<td>- Zuhai</td>
<td>Tel: 86-756-3210040</td>
</tr>
<tr>
<td>India</td>
<td>- Bangalore</td>
<td>Tel: 91-80-3090-4444</td>
</tr>
<tr>
<td>India</td>
<td>- New Delhi</td>
<td>Tel: 91-11-4160-8631</td>
</tr>
<tr>
<td>India</td>
<td>- Pune</td>
<td>Tel: 91-20-4121-0141</td>
</tr>
<tr>
<td>Japan</td>
<td>- Osaka</td>
<td>Tel: 81-6-6152-7160</td>
</tr>
<tr>
<td>Japan</td>
<td>- Tokyo</td>
<td>Tel: 81-3-6880-3770</td>
</tr>
<tr>
<td>Korea</td>
<td>- Daegu</td>
<td>Tel: 82-53-744-4301</td>
</tr>
<tr>
<td>Korea</td>
<td>- Seoul</td>
<td>Tel: 82-2-554-7200</td>
</tr>
<tr>
<td>Malaysia</td>
<td>- Kuala Lumpur</td>
<td>Tel: 60-3-7651-7906</td>
</tr>
<tr>
<td>Malaysia</td>
<td>- Penang</td>
<td>Tel: 60-4-227-8870</td>
</tr>
<tr>
<td>Philippines</td>
<td>- Manila</td>
<td>Tel: 63-2-634-9065</td>
</tr>
<tr>
<td>Singapore</td>
<td></td>
<td>Tel: 65-6334-8870</td>
</tr>
<tr>
<td>Taiwan</td>
<td>- Hsin Chu</td>
<td>Tel: 886-3-577-8366</td>
</tr>
<tr>
<td>Taiwan</td>
<td>- Kaohsiung</td>
<td>Tel: 886-7-213-7830</td>
</tr>
<tr>
<td>Taiwan</td>
<td>- Taipei</td>
<td>Tel: 886-2-2508-8600</td>
</tr>
<tr>
<td>Thailand</td>
<td>- Bangkok</td>
<td>Tel: 66-2-694-1351</td>
</tr>
<tr>
<td>Vietnam</td>
<td>- Ho Chi Minh</td>
<td>Tel: 84-29-5460-2100</td>
</tr>
</tbody>
</table>

## EUROPE

<table>
<thead>
<tr>
<th>Country</th>
<th>City</th>
<th>Phone Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austria</td>
<td>- Wels</td>
<td>Tel: 43-7242-2244-39, Fax: 43-7242-2244-393</td>
</tr>
<tr>
<td>Denmark</td>
<td>- Copenhagen</td>
<td>Tel: 45-4450-2828, Fax: 45-4485-2829</td>
</tr>
<tr>
<td>Finland</td>
<td>- Espoo</td>
<td>Tel: 358-9-4520-820</td>
</tr>
<tr>
<td>France</td>
<td>- Paris</td>
<td>Tel: 33-1-69-53-63-20, Fax: 33-1-69-30-90-79</td>
</tr>
<tr>
<td>Germany</td>
<td>- Garching</td>
<td>Tel: 49-8931-9700</td>
</tr>
<tr>
<td>Germany</td>
<td>- Haan</td>
<td>Tel: 49-2129-376400</td>
</tr>
<tr>
<td>Germany</td>
<td>- Heilbronn</td>
<td>Tel: 49-7131-67-3636</td>
</tr>
<tr>
<td>Germany</td>
<td>- Karlsruhe</td>
<td>Tel: 49-721-625370</td>
</tr>
<tr>
<td>Germany</td>
<td>- Munich</td>
<td>Tel: 49-89-627-144-0, Fax: 49-89-627-144-44</td>
</tr>
<tr>
<td>Germany</td>
<td>- Rosenheim</td>
<td>Tel: 49-8031-354-560</td>
</tr>
<tr>
<td>Israel</td>
<td>- Ra'anan</td>
<td>Tel: 972-9-744-7705</td>
</tr>
<tr>
<td>Italy</td>
<td>- Milan</td>
<td>Tel: 39-0331-742611, Fax: 39-0331-466781</td>
</tr>
<tr>
<td>Italy</td>
<td>- Padova</td>
<td>Tel: 39-049-7625286</td>
</tr>
<tr>
<td>Netherlands</td>
<td>- Drunen</td>
<td>Tel: 31-416-690399, Fax: 31-416-690340</td>
</tr>
<tr>
<td>Norway</td>
<td>- Trondheim</td>
<td>Tel: 47-7289-7561</td>
</tr>
<tr>
<td>Poland</td>
<td>- Warsaw</td>
<td>Tel: 48-22-3325737</td>
</tr>
<tr>
<td>Romania</td>
<td>- Bucharest</td>
<td>Tel: 40-21-407-87-60</td>
</tr>
<tr>
<td>Spain</td>
<td>- Madrid</td>
<td>Tel: 34-91-708-08-90, Fax: 34-91-708-08-91</td>
</tr>
<tr>
<td>Sweden</td>
<td>- Gothenberg</td>
<td>Tel: 46-31-704-60-40</td>
</tr>
<tr>
<td>Sweden</td>
<td>- Stockholm</td>
<td>Tel: 46-8-5090-4654</td>
</tr>
<tr>
<td>UK</td>
<td>- Wokingham</td>
<td>Tel: 44-118-921-5800, Fax: 44-118-921-5820</td>
</tr>
</tbody>
</table>

© 2017 Microchip Technology Inc.