AN2515

AVR® Low-Power Techniques

Introduction

Author: Per Andreas Gulbrandsen, Microchip Technology Inc.

This application note will discuss several techniques available to help limit the power consumption of AVR microcontrollers. It covers techniques and concepts such as sleep modes, the choice of oscillator and operating frequency, using the event system, sleepwalking, using BOD, and what to do with unused pins.

Table of Contents

1. Relevant Devices

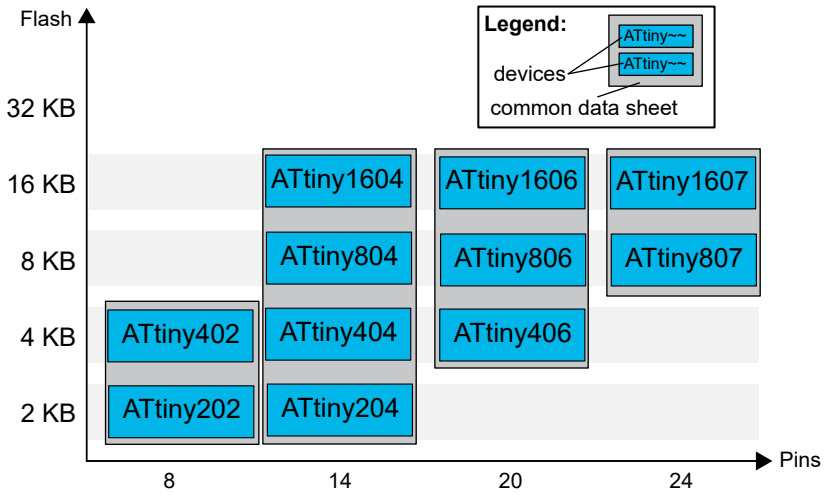
This chapter lists the relevant devices for this document.

1.1 tinyAVR® 0-series

The figure below shows the tinyAVR® 0-series, laying out pin count variants and memory sizes:

- Vertical migration is possible without code modification, as these devices are fully pin- and feature compatible.
- Horizontal migration to the left reduces the pin count and therefore, the available features.

Figure 1-1. tinyAVR® 0-series Overview



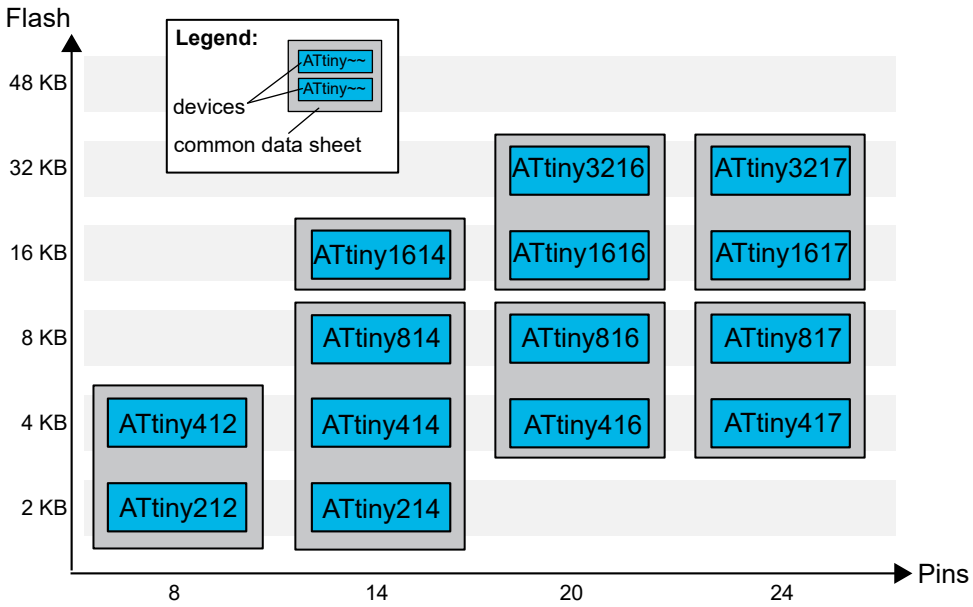
Devices with different Flash memory size typically also have different SRAM and EEPROM.

1.2 tinyAVR® 1-series

The figure below shows the tinyAVR® 1-series 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. Downward migration may require code modification due to fewer available instances of some peripherals.
- Horizontal migration to the left reduces the pin count and therefore, the available features.

Figure 1-2. tinyAVR® 1-series Overview



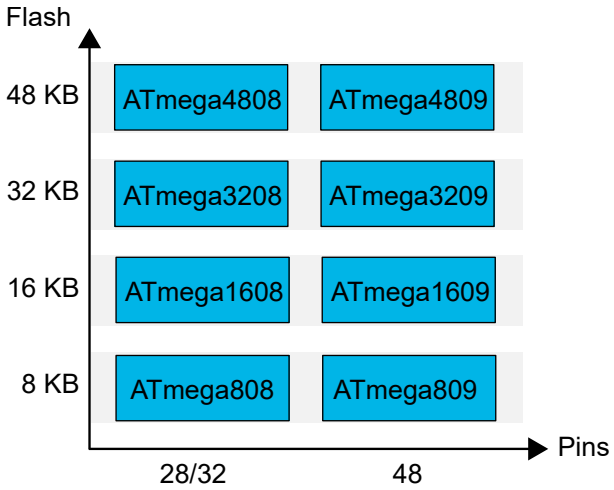
Devices with different Flash memory size typically also have different SRAM and EEPROM.

1.3 megaAVR® 0-series

The figure below shows the megaAVR® 0-series devices, laying out pin count variants and memory sizes:

- Vertical migration is possible without code modification, as these devices are fully pin and feature compatible.
- Horizontal migration to the left reduces the pin count and therefore the available features.

Figure 1-3. megaAVR® 0-series Overview



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

2. Conceptual Application

This application note will discuss several techniques available to help limit the power consumption. To illustrate this, a conceptual application is introduced, and the techniques will be applied to this application.

The conceptual application is a very simple application that monitors an external temperature sensor connected to a pin. In its basic form, the application busy-spins for one second then measures the value of the sensor using the Analog-to-Digital Converter (ADC) and compares this value to two predefined threshold values. Two threshold values are used to create a window, in order to avoid constant switching. Based on this comparison several actions can be taken, i.e. turn the heater OFF if the temperature is above the upper threshold value, or turn the heater ON if the temperature is below the lower threshold value. As long as the temperature is inside the window, no adjustments will be made. For simplicity, in this application, these actions are turning a pin ON or OFF.

3. Active Peripherals

When peripherals are enabled, they can impact power consumption even if they are not active. Therefore, a developer should make sure that only peripherals in use are enabled. On the Relevant Devices this is intuitive, as all peripherals are disabled by default.

On earlier AVR® devices, the Power Reduction Register (PRR) register has commonly been used to enable or disable peripherals. On the Relevant Devices, this register does not exist. Instead, in the register interface of each peripheral, there is an Enable bit. This, plus a more advanced clock system, with better clock gating, meaning that fewer gates are clocked in an unused clock path, replaces the PRR functionality. It is also more intuitive and user-friendly, as the Enable functionality is placed within the peripheral in question.

The peripheral operation can also be affected by sleep modes. This is discussed in 4. Operating Modes.

4. Operating Modes

Most applications do not require the CPU to run continuously after start-up. For such an application, sleep modes can be used to lower the power consumption.

4.1 Overview

Sleep modes are used to shut down peripherals and clock domains in the device in order to save power. The Sleep Controller (SLPCTRL) controls and handles the transitions between Active and Sleep mode.

There are in total four modes available:

- One Active mode in which the software is executed
- Three Sleep modes:
  - Idle
  - Standby
  - Power-Down

All Sleep modes are available and can be entered from Active mode. In active mode, the CPU is executing application code. When the device enters Sleep mode, program execution is stopped and interrupts or a Reset is used to wake the device again. The application code decides which Sleep mode to enter and when.

Interrupts are used to wake the device from sleep. The available interrupt wake-up sources depend on the configured Sleep mode. When an interrupt occurs, the device will wake up and execute the interrupt service routine before continuing normal program execution from the first instruction after the `SLEEP` instruction. Any Reset will take the device out of a Sleep mode.

The content of the register file, SRAM and registers are kept during Sleep. If a Reset occurs during Sleep, the device will Reset, start, and execute from the Reset vector.

4.2 Active Mode Operation

In Active mode all clocks are active, and the CPU is running. Power consumption in active mode is proportional to the operating frequency of the system clock. Applications that do not demand high-frequency operation can benefit from lowering the system clock.

Even though the rule of thumb is that lower frequency equals lower power, in some cases the opposite is true. Running at a higher clock frequency could mean that the CPU will finish its task faster, and return back to sleep quicker, resulting in a lower total power consumption. Imagine an application that wakes up from Standby sleep mode periodically and performs some calculations. If these calculations are done at a higher clock frequency the system will spend less time in Active and more time in Standby sleep mode, thus lowering the total power consumption. An example of the opposite could be where an operation such as USART communication is preventing the CPU from sleeping. In this situation, the timing of the USART transaction is the limiting factor, and higher CPU frequency will not help reduce time spent in Active mode.

Running the system at one static frequency can be sufficient for many applications. For some applications, there are advantages to using dynamic frequency scaling. As the operating frequency of the device can be changed on the fly, some parts of an application can execute at a lower frequency, while other parts can be executed at a higher frequency. This can be useful in e.g. applications that are very computationally heavy for short periods. An example of this can be an application that mainly sleeps

while collecting data, and periodically performs heavy computations on the sampled data. Using a higher clock frequency to perform the computations can reduce the total power consumption, even though the consumption when the computations are being performed is higher, because of the shorter duration of the computation period.

Note that this technique has a trade-off. As there is a cost related to executing the instructions needed to change the frequency, the code executed at higher frequency must compensate for this. I.e. the savings from executing code faster must outweigh the cost of adding code to change the frequency. Also, note that the frequency must most likely be changed both when entering and exiting the part of the application selected for high-frequency execution.

A convenient way of doing dynamic frequency scaling is to use the Main Clock Prescaler. This prescaler allows the clock to be scaled before going to the CPU and peripherals. Note that as this prescaler affects the clocks for both CPU and peripherals (any peripherals that are configured for a certain clock speed could malfunction). Examples of this could be violating the maximum clock speed of the ADC, or changing the BAUD rate of the USART when changing the peripheral clock. Refer to the chapter on Clock Controller in the corresponding data sheet, for details.

Creating a power budget can be a useful exercise when developing a low-power application. This can be done with a few simple steps. First, calculate the power consumption for each operating mode. Then, calculate how much time is spent in each mode. Lastly, using the numbers from the two previous steps, calculate the average and peak consumption. This allows the developer to calculate e.g. battery-life for that application. It can also be a valuable tool in profiling and optimizing the application.

4.3 Sleep Modes

In addition to Active mode, there are three different sleep modes, with decreasing power consumption and functionality.

Idle The CPU stops executing code, no peripherals are disabled.
All interrupt sources can wake up the device.

Standby The user can configure peripherals to be enabled or not, using the respective RUNSTBY bit. This means that the power consumption is highly dependent on what functionality is enabled, and thus may vary between the Idle and Power-Down levels.
SleepWalking is available for the ADC module.
The wake-up sources are pin interrupts, TWI address match, UART Start-of-Frame interrupt (if USART is enabled to run in Standby), RTC interrupt (if RTC enabled to run in Standby), and TCB interrupt.

Power-Down Only the WDT and the PIT (a component of the RTC) are active.
The only wake-up sources are the pin change interrupt and TWI address match.

Table 4-1. Sleep Mode Activity Overview

Group	Peripheral		Active in Sleep Mode		
		Clock	Idle	Standby	Power-Down
Oscillators	Main Clock Source		X	X*	
	RTC Clock Source		X	X*	
	WDT Oscillator		X	X	X

.....continued

Group	Peripheral		Active in Sleep Mode		
		Clock	Idle	Standby	Power-Down
Wake-Up Sources	INTn and Pin Change		X	X	X
	TWI Address Match		X	X	X
	Periodic Interrupt Timer		X	X	X
	UART Start-of-Frame		X	X*	
	ADC Window		X	X*	
	RTC Interrupt		X	X*	
	All other Interrupts		X		

Note:

- X means active. X* indicates that the RUNSTBY bit of the corresponding peripheral must be set to enter the active state.

4.4 Applying Operating Modes to the Conceptual Application

As the 2. Conceptual Application is inactive while waiting for another sample to be converted, great amounts of power can be saved by sleeping while waiting. In order to do this, the application needs a wake-up source. This is easily done by setting up the RTC to interrupt and wake the system once every second. When a wake-up happens, the CPU can initiate an ADC conversion, compare the result, take appropriate action if any, and turn back to sleep.

If we roughly estimate that the system will be awake for 1 ms per second for each sample (starting the system clock and CPU, starting the ADC conversion, waiting for ADC interrupt, reading out the result and comparing it, possibly toggling a pin, and going back to sleep), the expected time in Active mode is 0.1%. This is compared to not using sleep modes, and having the CPU stay continuously awake. The expected power consumption will be slightly above 0.1%, as power consumption in Sleep mode must also be taken into account.

Dynamic clock scaling can also be applied to the conceptual application. While the CPU is waiting for the ADC to complete there is no need for high CPU frequency. Based on the output value, the application decides if it needs to take action. If this is the case, the system can switch to a higher clock frequency to be able to complete these actions faster, and thus reduce the time spent in Active mode. However, as only a few instructions are executed after the comparison operation, there is no benefit to using this technique in the conceptual application.

4.5 Execution of the Sleep Instruction and Shared Variables Between ISR and Main

The potential problem with the sleep instruction is that an ISR can have executed right before the execution of the sleep instruction. If that ISR wrote to a flag which then needs to be handled in the main loop, it will not be handled until the next interrupt occurs and causes the device to wake up. The following code shows an example where this can happen:

```
volatile uint8_t shared_flag;
ISR(perip_vector){
    uint8_t i_flags = PERIP_INTFLAGS;
    shared_flag = 0x01;
```

```
    PERIP_INTFLAGS = i_flags;
}

main(){
    while(1){
        cli();
        if(shared_flag == 0x01){
            shared_flag &= ~0x01;
            sei();
            handle_event_etc();
        }
        sei();
        sleep();
    }
}
```

In this case, the *shared_flag* is protected from being altered by the ISR while it is being checked and cleared. However, if the ISR writes to the flag after the first *sei()*, inside the if statement, the sleep instruction will be executed without the *shared_flag* being checked again. As such, the events that need to happen in *main()* will not happen until the next time an interrupt occurs, waking the device from sleep.

To handle all flags inside the main loop, use the following code:

```
main(){
    while(1){
        cli();
        if(shared_flag == 0){
            sei();
            sleep();
        }
        sei();
        //handle shared_flags and run application
    }
}
```

This code ensures that no shared flags between ISRs and the main loop go unhandled before going to sleep. It does so because any instruction executing after the *sei()* instruction will be allowed to execute before a jump into a pending interrupt. The *shared_flag* variable can be shared between multiple ISRs. Note that any read-modify-write of the *shared_flag* inside the main loop will have to be protected by *cli()* and *sei()* as shown in the code below:

```
cli();
shared_flag &= ~0x01;
sei();
```

This makes the read-modify-write an atomic action, that is, it cannot be interrupted by any ISRs. However, this can be avoided by using General Purpose Input Output (GPIOR) registers as shared variables between ISRs. Registers GPIOR0 to GPIOR3 allow the compiler to compile any single-bit modification to one line of assembly code that will execute in a single CPU cycle. In comparison, using a read-modify-write of a *uint8_t* variable will take three cycles, or five cycles when taking the *cli()* and *sei()* instructions into account.

5. Event System

5.1 Overview

The Event System (EVSYS) enables direct peripheral-to-peripheral signaling. It allows a change in one peripheral (the event generator) to trigger actions in other peripherals (the event users) through event channels, without using the CPU. It is designed to provide short and predictable response times between peripherals, allowing for autonomous peripheral control and interaction, and also for the synchronized timing of actions in several peripheral modules. It is thus a powerful tool for reducing the complexity, size, and the execution time of the software.

A change of the event generator's state is referred to as an event and usually corresponds to one of the peripheral's interrupt conditions. Events can be directly forwarded to other peripherals using the dedicated event routing network. The routing of each channel is configured in software, including event generation and use.

Only one trigger from an event generator peripheral can be routed on each channel, but multiple channels can use the same generator source. Multiple peripherals can use events from the same channel.

A channel path can be either asynchronous or synchronous to the main clock. The mode must be selected based on the requirements of the application.

The Event System can directly connect analog and digital converters, analog comparators, I/O port pins, the real-time counter, timer/counters, and the configurable custom logic peripheral. Events can also be generated from software and the peripheral clock.

5.2 Applying Events to Conceptual Application

The 2. Conceptual Application can benefit from using Events. The RTC, which times the operation, can use an event to signal to the ADC that a conversion must be done, instead of waking the CPU to start a conversion. This allows the time spent in sleep mode to be slightly extended, as the CPU can now sleep while the ADC is converting. As the ADC conversion is very quick (13 ADC cycles, 52 CPU cycles), and the reduction of time spent in Active mode is equally short, the reduction in power consumption will be small. The gain in using events will be more visible when adding Sleepwalking functionality (explained in 6. Sleepwalking).

6. Sleepwalking

Some peripherals have the ability to process incoming data without waking the CPU. Depending on the incoming data, the peripherals can decide if the CPU should be woken up, or if there is no further action to take. On the Relevant Devices, there are two such peripherals. The TWI supports Wake on Address Match, where it examines the start of an incoming data frame, and only wakes the CPU if the address of the incoming frame matches its own address. The ADC supports Window mode, where each sample is compared with two values, representing a window. Based on configuration, the ADC can wake the CPU if the sample is below, above, inside, or outside the window.

6.1 Applying Sleepwalking to Example Application

Implementing ADC window mode in the example application could potentially allow the CPU to extend standby sleep. The ADC itself can compare the converted value, and check if it is below the window (if the heater is OFF) or above the window (if the heater is ON). The CPU will only wake up when it needs to turn the heater ON or OFF. In a perfectly stable environment, the CPU would never wake up, causing the total power consumption to be equal to the sleep mode consumption. In a constantly fluctuating environment, where the temperature alternates between below and above the thresholds for every sample (i.e. every second), the power consumption will be the same as when the CPU wakes up to do the comparison for every converted ADC sample.

7. Choice of Oscillator

As already mentioned, the system clock greatly impacts the power consumption of the system. In addition, the choice of oscillator also impacts the power consumption. As a general guideline, this is a tradeoff between accuracy and consumption. E.g. when selecting a 32 kHz clock, the internal 32K Ultra-Low Power (ULP) oscillator will offer drastically lower power consumption than an external crystal, but the accuracy is also worse. If the application is not dependent on high accuracy, choosing the ULP oscillator will help lower the power budget.

Allowing the clock sources to be disabled during sleep modes will lower the power consumption of the oscillator. However, for oscillators with a significant wake-up time, this will shorten the amount of time spent in sleep mode. For applications requiring high-accuracy timing, enabling oscillators to run in standby by writing the corresponding RUNSTDBY bit (e.g. XOSC32KCTRLA.RUNSTDBY for the external 32K oscillator) will eliminate the start-up time, and thus improve the timing. Refer to the corresponding data sheet for details.

There are techniques that can be used to improve the accuracy of the 32K ULP. By doing periodic run-time calibration, using a high-accuracy external crystal, the accuracy of the internal 32K ULP oscillator can be greatly improved. Refer to the application note on precise, ultra-low power timing using periodic enabling of the 32.768 kHz external crystal oscillator for recalibration of the ULP internal oscillator.

7.1 Applying Choice of Oscillator to Conceptual Application

The 2. Conceptual Application does not require great accuracy, and can, therefore, use the internal 32K ULP oscillator as the clock source of the RTC to keep track of time while sleeping.

8. BOD

A BOD protects the microcontroller when the supply voltage falls below its operating threshold by resetting the device. This keeps the microcontroller in a defined state when the V_{CC} is below its operating threshold. The BOD is not important to the microcontroller while it is in sleep mode but it is extremely important when it wakes up. Therefore, as a rule, most microcontrollers keep the BOD active during sleep mode and it contributes substantially to sleep mode power consumption.

The solution to this problem is to have the microcontroller shut down the BOD when it enters sleep mode and starts it again just before leaving sleep mode. This approach ensures the BOD is functioning when it is needed without any power consumption penalty while in sleep mode. However, this approach does add to the start-up time of the device, as the BOD needs additional start-up time.

8.1 BOD Sleep Mode Operation

The BOD can be configured in four ways for Active operation: Disabled, Enabled, Sampled and Enabled with wake-up halted until BOD is ready.

- Disabled: The BOD is disabled.
- Enabled: The BOD is continually sampling the supply voltage and comparing it to the selected threshold.
- Sampled: The supply voltage is sampled and compared to the threshold at either 125 Hz or 1 kHz.
- Enabled with wake-up halted until BOD is ready: The BOD will sample the supply voltage when a wake-up occurs, and wake-up will be halted until the BOD signals that the supply voltage is above the selected threshold. This adds to the wake-up time. In active mode, the BOD is continually sampling the supply voltage and comparing it to the selected threshold.

For Sleep operation, there are three configurations: Disabled, Enabled, and Sampled.

The Active operation can only be configured using fuses, while the Sleep operation can be configured using both fuses and registers. Refer to the corresponding data sheet for details.

8.2 Applying BOD to Conceptual Application

The conceptual application does not need a BOD in sleep. Disabling the BOD in sleep will reduce the power consumption.

In active mode, the conceptual application can use the Enabled with wake-up halted mode. This configuration means that the BOD will monitor the supply current when needed (active mode), but will not be active in sleep mode. Supply current is checked when a wake-up occurs, so that normal operation is safe. There will be a small penalty of added wake-up time, but as the example application is not time critical, this is tolerated.

9. Unused Pins

All digital I/O pins are by default floating to avoid hardware conflicts. However, since the pins have digital input buffers it is important to ensure that the level on an I/O pin is well-defined to avoid sporadic internal switching and leakage. The leakage caused by floating I/O is relatively small and is mainly observable in sleep, but can be minimized by enabling the internal pull-up. Using an external pull-up is also an option.

In addition, disabling the digital input buffer on unused pins will further lower the power consumption. This is also true for pins connected to an analog peripheral, e.g. the ADC. Both disabling of the digital input buffer, and enabling of the internal pull-up can be done in the PINCTRLn registers for the individual ports. Refer to the corresponding data sheet for details.

10. Revision History

Doc. Rev.	Date	Comments
C	10/2018	Updated list of tinyAVR and megaAVR relevant devices.
B	01/2018	New sections: • *tinyAVR 0-series* • *megaAVR 0-series* • *Execution of the Sleep Instruction and Shared Variables Between ISR and Main*
A	07/2017	Initial document release.

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.

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

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

Trademarks

ISBN: 978-1-5224-3650-8

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.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-67-3636 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820