AVR1300: Using the XMEGA ADC

8-bit AVR® Microcontrollers

Application Note

Features

- **Up to 12 bit resolution**
- **Up to 2M samples per second**
- **Signed and unsigned mode**
- **Selectable gain**
- **Pipelined architecture**
- **4 virtual channels**
- **Result comparator**
- **Automatic calibration**
- **Internal connection to DAC output**
- **Driver source code included**

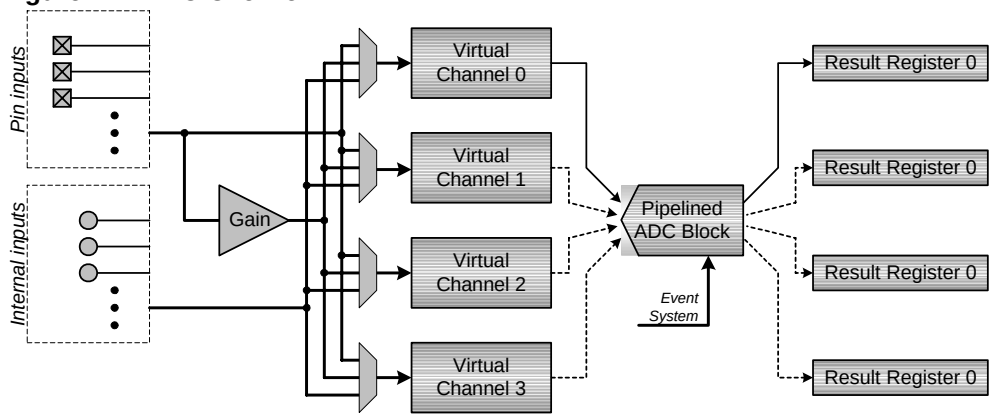
1 Introduction

The XMEGA™ ADC module is a high-performance Analog-to-Digital converter capable of conversion rates up to 2 M samples per second with a resolution of 12 bits. Flexible multiplexer (MUX) settings, integrated gain stage and four virtual input channels make this a flexible module suitable for a huge range of applications, such as data acquisition, embedded control and general signal processing.

This application note describes the basic functionality of the XMEGA ADC with code examples to get up and running quickly. A driver interface written in C is included as well.

Advanced usage, such as Direct Memory Access and the XMEGA Event System, is outside the scope of this application note. Please refer to the device datasheets and other relevant application notes for details.

Figure 1-1. ADC Overview



Rev. 8149A-AVR-04/08

2 Module Overview

This section provides an overview of the functionality and basic configuration options of the ADC. Section 3 then walks you through the basic steps to get you up and running, with register descriptions and configuration details.

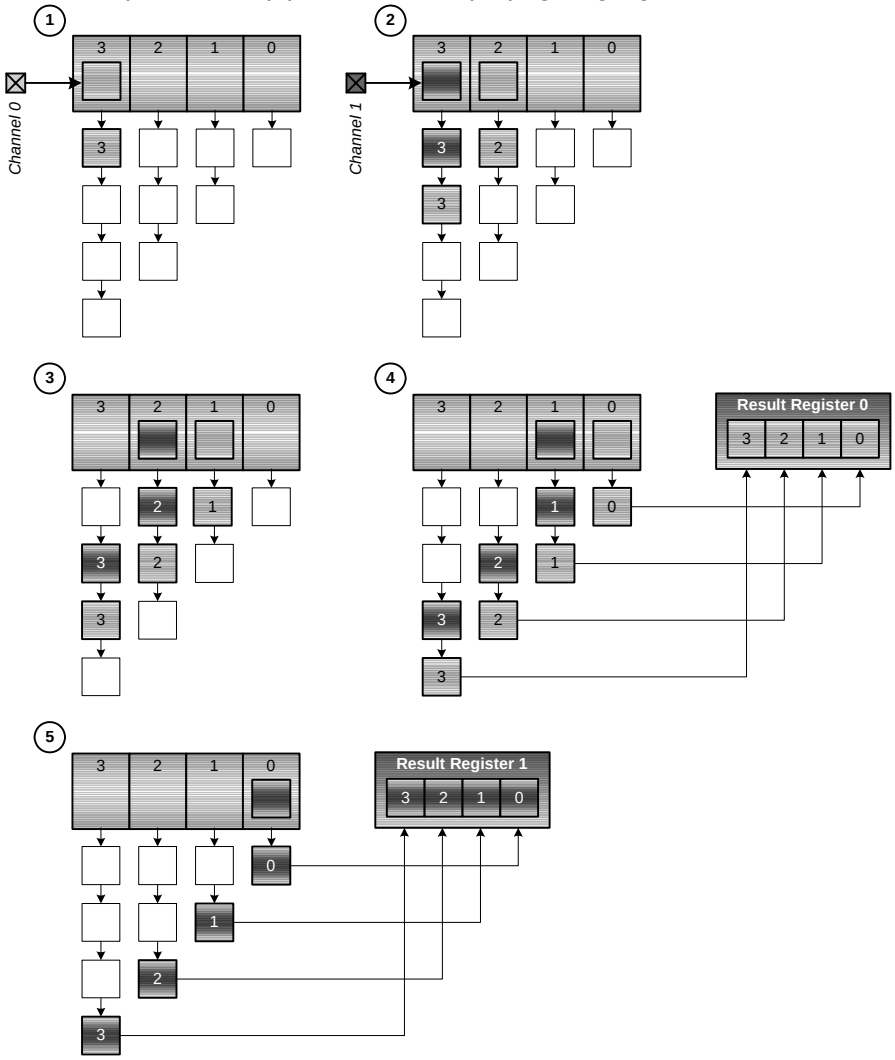
2.1 Pipeline Architecture and Virtual Channels

The ADC conversion block has a 12-stage pipelined architecture capable of sampling several signals in parallel. There are four input selection MUXes with individual configurations. The separate configuration settings for the four MUXes can be viewed as *virtual channels*, with one set of result registers each, all sharing the same ADC conversion block. Refer to Figure 1-1 above.

The MUX outputs can be sampled every four ADC clock cycles and each signal propagates through the pipeline, where one bit is converted at each stage. In this way the ADC is capable of sampling one signal every four ADC clock cycles, even if each signal must propagate through all stages in the pipeline before the result is ready in the result register. The propagation time for one single sample through the pipeline is 7 ADC clock cycles for 12-bit conversions and 5 cycles for 8-bit conversions. If Gain is used the propagation time increases with one cycle. At full utilization the ADC delivers one result every ADC clock cycles. The relation between the XMEGA peripheral clock and the ADC clock is described in Section 2.8.

Figure 2-1 below shows a simplified 4-stage pipeline during conversion of two input signals. The figure shows that once the signal has been sampled into the pipeline, the first stage converts the MSB of the signal. Next, the second stage converts the next bit of the signal while the first stage converts the MSB of the second signal.

Figure 2-1. Simplified ADC pipeline with two propagating signals



Note that it is also possible to repeatedly convert signals from a single channel. The pipeline can sample and convert a signal from one channel even if there is already a previous sample from that channel on its way through the pipeline.

The four virtual channels have one *MUX Control* register (`CHnMUXCTRL`), one *Channel Control* register (`CHnCTRL`) and one *Result* register pair (`CHnRESL`/`CHnRESH`) each, in addition to several control bits distributed in shared registers.

2.2 Gain Stage

There is one gain block that all the channels can use between its input selection MUX and the conversion block. The available gain settings are 1x, 2x, 4x, 8x, 16x, 32x and 64x. The Gain Factor bit field (`GAINFACT`) in the *Channel Control* register (`CHnCTRL`) set the gain factor for the channel. If one channel needs to convert very small values, you can configure that channel to use the gain stage to boost the voltages before converting them. It is possible to have different gain settings for all the virtual channels.

2.3 Multiplexer Settings

The MUXes are used to select which input signals to convert for each virtual channel. There are four distinct configuration choices that can be selected using the *INPUT mode* bit field (`INPUTMODE`) in the *Channel Control* register (`CHnCTRL`):

- Differential Input without Gain Stage
- Differential Input with Gain Stage
- Internal Inputs without Gain Stage
- Single-ended Input without Gain Stage

The positive and negative inputs are selected using the *MUX Positive Input* and *MUX Negative Input* bitfields (`MUXPOS` and `MUXNEG`). An alternative name for the *MUX Positive Input* bitfield is *MUX Internal Input* (`MUXINT`) when measuring internal inputs.

When a device has two ADC the input can only be connected to the corresponding port. Meaning that ADC A can be connected to PORT A and ADC B can be connected to PORT B. The positive input can be connected to any one of the eight input signals of corresponding port. The negative input can be connected to one of the first four input signals of corresponding port.

When the devise has one ADC the positive input can be connected to any of the available input signals of both PORT A and PORT B. The negative input can be connected to the first four inputs signals of corresponding PORT.

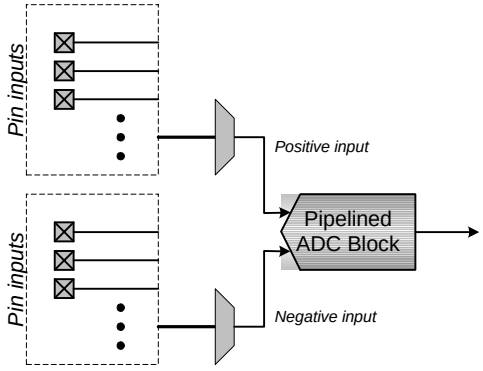
Refer to the datasheet to determine the number of ADC and the devices pin configuration.

Note that even if the difference between two inputs could be negative, voltages below GND or above VCC should under no circumstances be applied to any input pin.

2.3.1 Differential Input without Gain Stage

With this setting, the MUX measures the difference between two input signals.

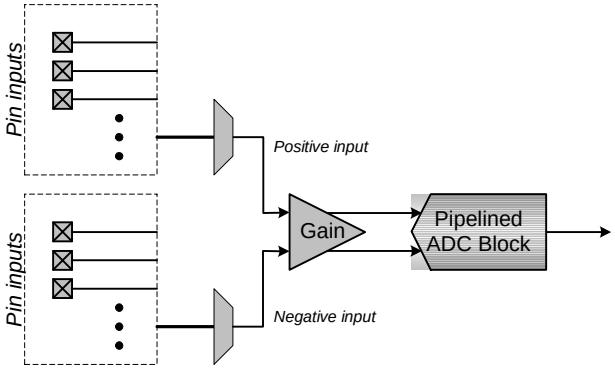
Figure 2-2. Differential input without gain stage



2.3.2 Differential Input with Gain Stage

This setting is almost identical to differential input without gain stage. With this setting the gain stage is inserted in the signal path for this channel, providing up to 64 times amplification of the differential input signal.

Figure 2-3. Differential input with gain stage



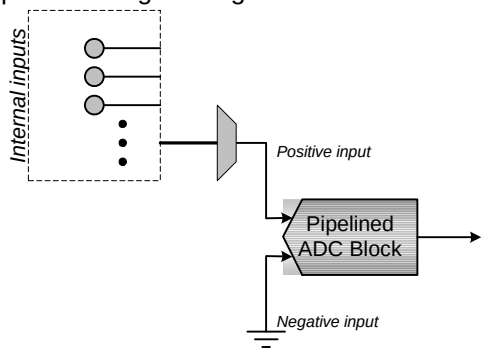
Note that the gain stage does not load the input and external signal source will see very high input impedance for channels that use the gain stage. This is useful for measuring weak signal sources.

2.3.3 Internal Inputs without Gain Stage

With this setting, the MUX measures one of several internal signals. The negative input is always connected to GND while the positive input can be connected to one of the following internal sources: Temperature Reference, DAC Internal Output (for DAC calibration), VCC/10 (for supply voltage measurement) or Bandgap Reference. Note that two channels can select different internal sources. They are not limited to one common setting, as opposed to the shared gain stage setting.

The Bandgap Reference could be used to measure an unknown external reference. With a measurement of a known voltage (the Bandgap Reference, 1.1 V) using an unknown reference, it is easy to calculate the voltage of the external reference.

Figure 2-4. Internal input without gain stage



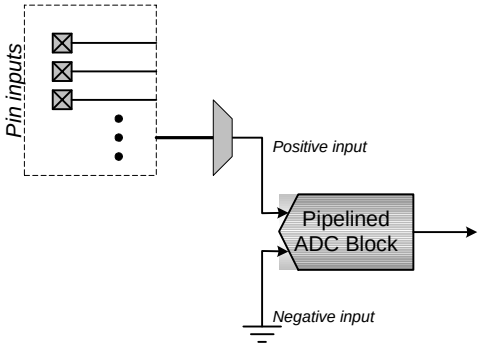
For more information on DAC configuration, please refer to the device datasheet or the application note “AVR1301: Getting Started with the XMEGA DAC”.

Note that if no other modules are using the Bandgap Reference. It must be turned on using the *Bandgap Enable* bit (BANDGAP) in the *Reference Control* register (REFCTRL). The same goes for the Temperature Reference, which is not shared with any other modules. The Temperature Reference is turned on using the *Temperature Reference Enable* bit (TEMPREF). Also note that there is a certain settling time for the Bandgap and Temperature Reference, and they should be enabled in due time before starting any conversions.

2.3.4 Single-ended Input without Gain Stage

With this setting, the ADC measures the value of one input signal. The difference between this setting and differential measurement is that the negative input is always connected to GND, which means that there is no need to reserve one ADC input pin to represent GND.

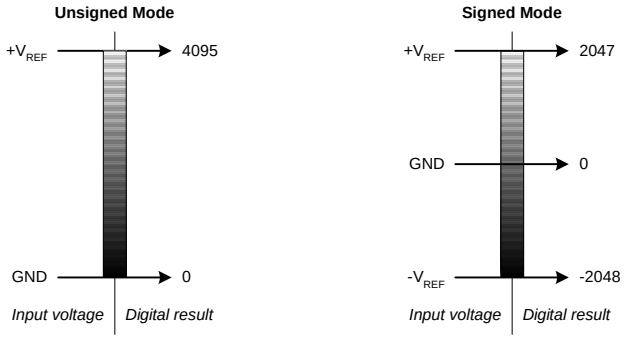
Figure 2-5. Single-ended input without gain stage



2.4 Conversion Mode

The conversion block can be put in the unsigned or signed conversion mode. In unsigned mode the conversion range is from zero to the reference voltage. In signed mode the range is from negative to positive reference voltage. The figure below shows the difference in conversion ranges.

Figure 2-6. Unsigned and signed conversion mode



The figure shows that the unsigned mode gives higher resolution on positive values than signed mode, but cannot convert negative values. The signed mode can convert negative values, but at the cost of lower resolution overall. When the ADC uses differential inputs signed mode must be used, while in the other modes both signed and unsigned mode can be used.

Note that conversion mode is configured for the whole ADC, not individually for each channel, which means that the ADC must be put in the signed mode even if only one of the channels uses differential inputs.

The conversion mode is configured using the *Conversion Mode* bit (`CONVMODE`) in *Control Register A* (`CTRLB`).

2.5 Result Presentation

The ADC can be configured to present conversion results in the following formats:

- 12 bits, right adjusted
- 8 bits, right adjusted
- 12 bits, left adjusted

Note that a lower resolution gives faster conversions, as there are fewer pipeline stages for the signal samples to propagate through. Therefore, selecting result presentation is a tradeoff between resolution and conversion speed.

The ADC resolution is configured using the *Resolution* bitfield (`RESOLUTION`) in *Control Register B* (`CTRLB`).

2.6 Voltage References

The application can choose between the following voltage references for conversion results:

- Internal reference of 1.0 V
- Internal reference of V_{CC} – 0.6 V
- External reference (V_{REF+} - V_{REF-})

For the internal references, the user has the choice of connecting the internal voltage source to the external reference pins for decoupling purposes. It is imperative to decouple the voltage reference, internal and external, to be able to achieve the full 12-bit accuracy of the ADC module. Also, great care should be taken when designing the analog signal paths as well.

Note that the external reference pin V_{REF+} is shared with the DAC module. The voltage reference is configured using the *Reference* bitfield (`REFSEL`) in the *Reference Control* register (`REFCTRL`).

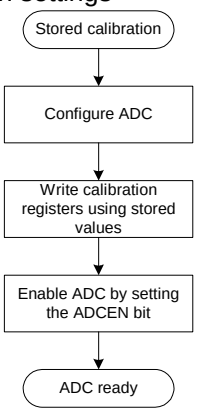
2.7 Calibration

The ADC module has been calibrated during production. Using the production calibration settings stored in flash will compensate for minor gain errors in the analog circuitry in the ADC module. However, the factory default calibration provides sufficient accuracy for most applications.

If you want to use stored production calibration settings, these must also be written to the calibration registers before enabling the ADC.

Flowcharts for loading stored calibration settings are shown in Figure 2-7 below.

Figure 2-7. Using stored calibration settings



2.8 Conversion Speed

The ADC clock is derived from a scaled version of the XMEGA peripheral clock, where the available factors are 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256 and 1/512. Regardless of clock speed, the ADC module has a minimum and maximum recommended speed to guarantee correct operation. Please consult the device datasheet for details on recommended minimum and maximum ADC clock speeds.

Note that having a fast ADC clock gives a short propagation time for each sample, but does not mean that you cannot sample data at a much slower rate. For instance, an application could have a sample rate of 10 kHz with an ADC clock of 8 MHz. However, it is not possible to sample at a higher rate than one fourth of the ADC clock speed.

The conversion speed is configured using the *Clock Prescaler* register (PRESCALER).

2.9 Free-running Mode

Instead of manually starting conversions by setting one or more of the *Start Conversion* bits (CHnSTART) in *Control Register A* (CTRLA) or assigning events to virtual channels, the ADC can be put in free-running mode. This means that a number of channels are repeatedly converted in sequence as long as the mode is active.

The *Channel Sweep Selection* bitfield (SWEEP) in the *Event Control* register (EVCTRL) selects which channels to include in free-running mode. You can choose between channel 0 only, channel 0 and 1, channel 0 to 2 or all four channels.

Note that the same bits are used to select the channels to include in an event-triggered conversion sweep, but that is outside the scope of this application note.

Care should be taken not to change any involved MUX settings when in free-running mode, as this would corrupt conversion results.

2.10 Interrupts

To avoid having to poll a register to check when conversions are finished, the ADC can be configured to issue interrupt requests upon conversion complete. This can be used to do result processing using interrupt handler code while leaving the CPU ready for other tasks most of the time.

For more information, please refer to the device datasheet or the application note “AVR1305: Getting Started with the XMEGA Interrupt Controller”.

2.11 Result Comparator

Instead of merely converting an input value, the ADC can be configured to compare the result to a given value and only issue an interrupt or event when the result is above or below that value. Typical use of this feature is to leave one or more ADC channels in free-running mode and configure the ADC to issue an interrupt when one of the input signals reach a certain threshold.

For more information, please refer to the device datasheet.

3 Getting Started

This section walks you through the basic steps for getting up and running with simple conversion and experimenting with MUX settings. The necessary registers are described along with relevant bit settings.

Note that this section only covers manual polling of status bits. Interrupt control is not covered, but is an easy step after studying the application note “AVR1305: Getting Started with the XMEGA Interrupt Controller”.

3.1 Single Conversion

Task: One single-ended conversion of ADC input 1 using virtual channel 2.

- Set the *Input Mode* bitfield (`INPUTMODE`) in *Channel 2 Control Register* (`CH2CTRL`) equal to 0x3 to select single-ended input.
- Set the *MUX Positive Input* bitfield (`MUXPOS`) in *Channel 2 MUX Control Register* (`CH2MUXCTRL`) equal to 0x01 to select ADC input 1.
- Set the *Enable* bit (`ENABLE`) in *Control Register A* (`CTRLA`) to enable the ADC module without calibrating.
- Set the *Start Conversion* bit for channel 2 (`CH2START`) in *Control Register A* (`CTRLA`) to start a single conversion.
- Wait for the *Interrupt Flag* bit for channel 2 (`CH2IF`) in the *Interrupt Flags* register (`INTFLAGS`) to be set, indicating that the conversion is finished.
- Read the *Result* register pair for channel 2 (`CH2RESL`/`CH2RESH`) to get the 12-bit conversion result as a 2-byte value.

3.2 Multiple Channels

Task: One single-ended conversion of ADC input 3 and 6 using virtual channel 1 and 3.

- Set the *Input Mode* bitfield (`INPUTMODE`) in *Channel 1 Control Register* (`CH1CTRL`) and *Channel 3 Control Register* (`CH3CTRL`) equal to 0x3 to select single-ended input on both channels.
- Set the *MUX Positive Input* bitfield (`MUXPOS`) in the *MUX Control Register* for channel 1 and 3 (`CH1MUXCTRL` and `CH3MUXCTRL`) equal to 0x03 and 0x06 respectively.
- Set the *Enable* bit (`ENABLE`) in *Control Register A* (`CTRLA`) to enable the ADC module without calibrating.

- Set the *Start Conversion* bit for channel 1 and 3 (`CH1START` and `CH3START`) in *Control Register A* (`CTRLA`) to start two conversions.
- Wait for the *Interrupt Flag* bits for channel 1 and 3 (`CH1IF` and `CH3IF`) in the *Interrupt Flags* register (`INTFLAGS`) to be set, indicating that the conversions are finished.
- Read the *Result* register pair for channel 1 and 3 (`CH1RESL`/`CH1RESH` and `CH3RESL`/`CH3RESH`) to get the 12-bit conversion results as 2-byte values.

3.3 Free-running Mode

Task: Free-running differential conversion on channel 0, using ADC0 and ADC3 as positive and negative inputs.

- Set the *MUX Positive Input* and *MUX Negative Input* bitfields (`MUXPOS` and `MUXNEG`) in *Channel 0* (`CH0MUXCTRL`) to 0x00 and 0x03 respectively.
- Set the *Free Run* bit (`FREERUN`) in *Control Register B* (`CTRLB`) to enable free running mode.
- Set the *Enable* bit (`ENABLE`) in *Control Register B* (`CTRLB`) to enable the ADC module without calibrating.
- Optionally wait for the *Interrupt Flag* bit for channel 0 (`CH0IF`) in the *Interrupt Flags* register (`INTFLAGS`) to be set, indicating that a new conversion is finished. Clear the flag by writing a one to it is going to be used later.
- Read the *Result* register pair for channel 0 (`CH0RESL`/`CH0RESH`) to retrieve the latest 12-bit conversion results as a 2-byte value.

Note that it is not strictly required to wait for the interrupt flag when using free-running mode. However, to make sure you have a fresh conversion, you should wait for the flag, clear it and then read the result. Also note that it is recommended to use the Free-running Mode together with DMA data transfer to offload work from the CPU.

4 Advanced Features

This section introduces more advanced features and possibilities with the ADC. In-depth treatment is outside the scope of this application note and the user is advised to study the device datasheet and relevant application notes.

4.1 DMA Controller

Instead of using interrupt handlers to read and process the result registers, it is possible to use the XMEGA DMA Controller to move data from one or more result registers to memory buffers or other peripheral modules. This moving of data is done without CPU intervention, and leaves the CPU ready for other tasks, even without having to execute interrupt handlers.

For more information, please refer to the device datasheet or the application note “AVR1304: Getting Started with the XMEGA DMA Controller”.

4.2 Event System

To improve conversion timing and further offload work from the CPU, the ADC is connected to the XMEGA Event System. This makes it possible to use incoming events to trigger single conversions or conversion sweeps across several channels. The ADC conversion complete conditions also serve as event sources available for other peripheral modules connected to the event system.

For more information, please refer to the device datasheet or the application note "AVR1001: Getting Started with the XMEGA Event System".

5 Driver Implementation

This application note includes a source code package with a basic ADC driver implemented in C and in Assembly. It is written for the IAR Embedded Workbench® compiler.

Note that this ADC driver is not intended for use with high-performance code. It is designed as a library to get started with the ADC. For timing and code space critical application development, you should access the ADC registers directly. Please refer to the driver source code and device datasheet for more details.

5.1 Files

The source code package consists of five files:

- *adc_driver.c* – ADC driver source file containing all the functions written in C.
- *adc_driver.h* – ADC driver header file.
- *adc_driver_asm.S90* – ADC driver source file containing all the functions written in Assembly.
- *adc_driver_asm.h* – ADC driver header file for the Assembly code.
- *adc_example_polled.c* – Example code using the polled driver.
- *adc_example_interrupt*.c – Example code using the interrupt driver.

Note that the driver and example code does not include support for DMA data transfer or the XMEGA Event System.

For a complete overview of the available driver interface functions and their use, please refer to the source code documentation.

5.2 Doxygen Documentation

All source code is prepared for automatic documentation generation using Doxygen. Doxygen is a tool for generating documentation from source code by analyzing the source code and using special keywords. For more details about Doxygen please visit http://www.doxygen.org. Precompiled Doxygen documentation is also supplied with the source code accompanying this application note, available from the *readme.html* file in the source code folder.

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2008 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, AVR® and others, are the registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

8149A-AVR-04/08