AVR1300: Using the XMEGA ADC

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**

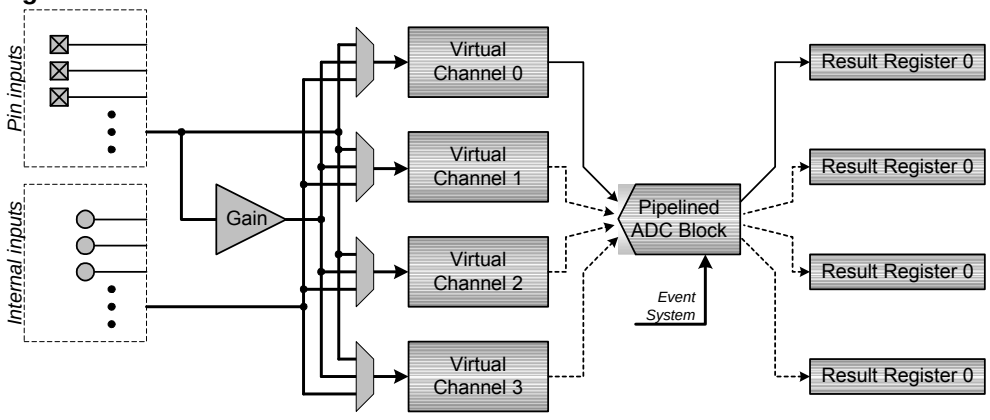
1 Introduction

The XMEGA™ ADC module is a high-performance Analog-to-Digital converter capable of conversion rates up to 2 million samples per second (Msps) 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 wide 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 (DMA) 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



8-bit AVR® Microcontrollers

Application Note

Preliminary

Rev. 8032C-AVR-09/09

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.

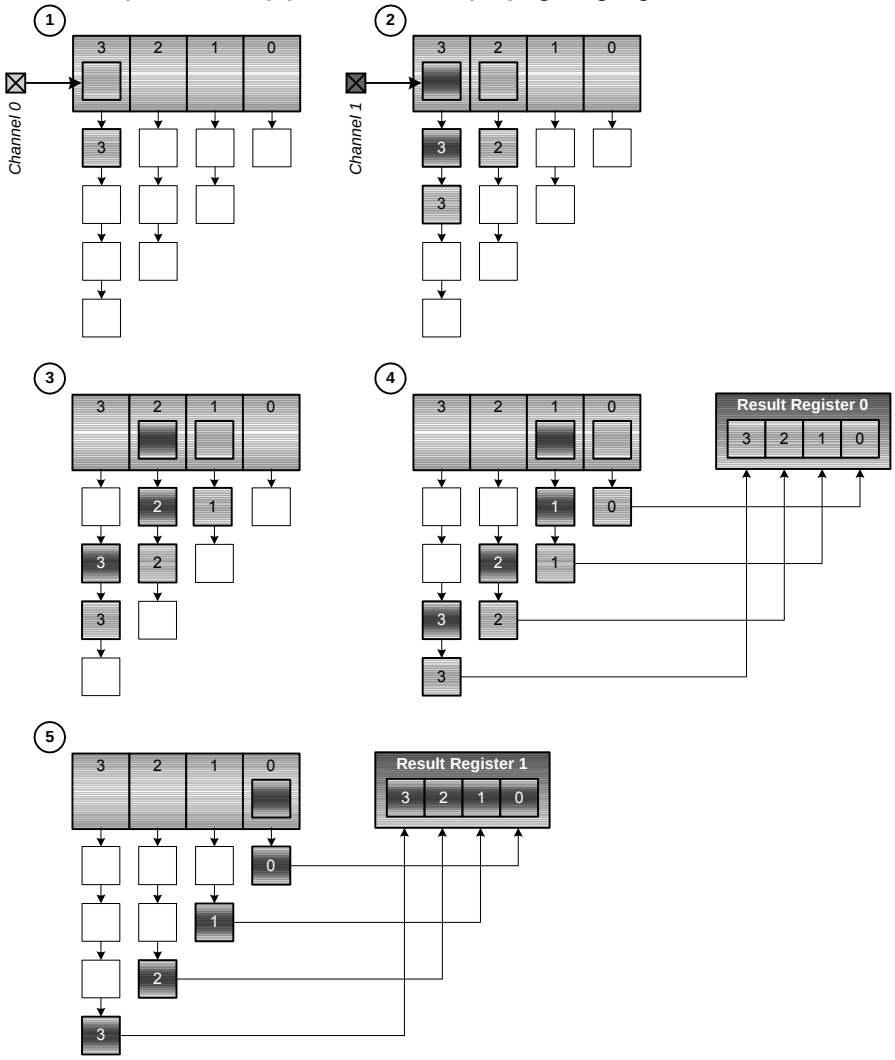
2.1 Pipeline Architecture and Virtual Channels

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 signal conversion 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 by one cycle. At full utilization the ADC delivers one result every ADC clock cycle. 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 first signal. While the second stage is converting the next bit of the signal, the first stage now 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.

All 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

The ADC has an internal gain stage which can be configured to amplify a voltage to allow measurement of smaller voltages.

This is a shared gain stage that can be used by all the channels. When the channel is configured to use gain, the gain stage is inserted between the channel 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.

It is possible to have individual gain settings for all the virtual channels.

The propagation delay for an ADC sample through the ADC increases by one ADC clock cycle when using the gain stage.

Using the gain setting of 1x gives no amplification, but it can be used to give the propagation delay for a channel with no amplification identical to channels that use amplification. However, to minimize the analog signal path for best possible ADC results it is recommended to configure the channel without gain unless having identical propagation delay is important in the application.

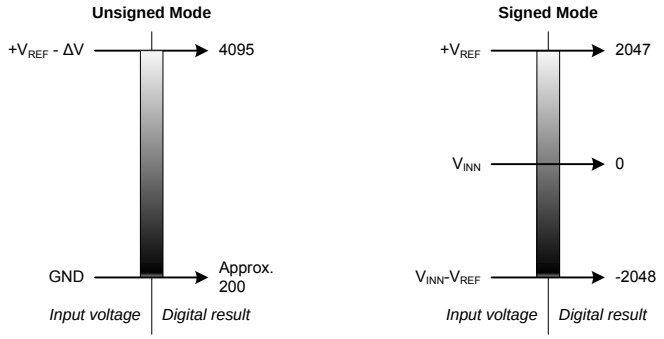
2.3 Conversion Mode

The conversion block can be put in the *unsigned* or *signed* conversion mode.

Signed mode can be used as input mode for both differential and single-ended inputs, while *unsigned* mode is only available for the *single-ended* or *internal* input.

In unsigned mode the conversion range is from ground to the reference voltage (more precisely to Vref – ΔV). In signed mode the range is from negative to positive reference voltage. The figure below shows the difference in conversion ranges.

Figure 2-2. 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 B* (CTRLB).

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.

2.4 Multiplexer Settings

The MUXes are used to select input signal for each virtual channel. There are four distinct configuration choices that can be selected using the *Channel Input mode* bitfield (INPUTMODE) in the *Channel Control* register (CHnCTRL):

- Differential Input
- Differential Input with Gain Stage
- Single-ended Input
- Internal Input

The positive and negative inputs are selected using the *MUX Positive Input* and *MUX Negative Input* bitfields (MUXPOS and MUXNEG) in the *Channel Mux Control* register (MUXCTRL). An alternative name for the *MUX Positive Input* bitfield used in the header files is *MUX Internal Input* (MUXINT) when measuring internal inputs.

In devices with two ADCs, the inputs 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 (PIN0 – PIN3) of the corresponding port for differential without Gain and the second four input signals (PIN4 –PIN7) for differential with Gain.

In devices with only one ADC but several analog ports, the positive input can be connected to any of the available input signals from both PORT A and PORT B. The negative input can be connected to one of the first four input signals (PIN0 – PIN3) of the corresponding port for differential without Gain and the second four input signals (PIN4 –PIN7) for differential with Gain.

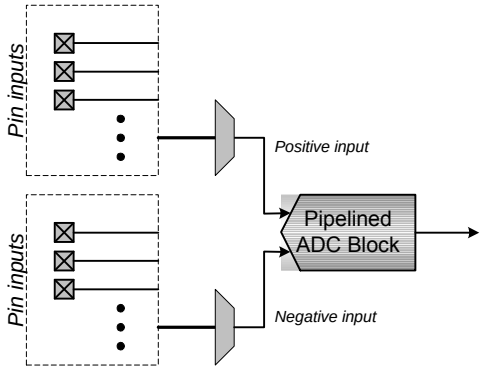
Refer to the datasheet to determine the number of ADC and the devices pin configuration.

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.

2.4.1 Differential Input

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

Figure 2-3. Differential input without gain stage

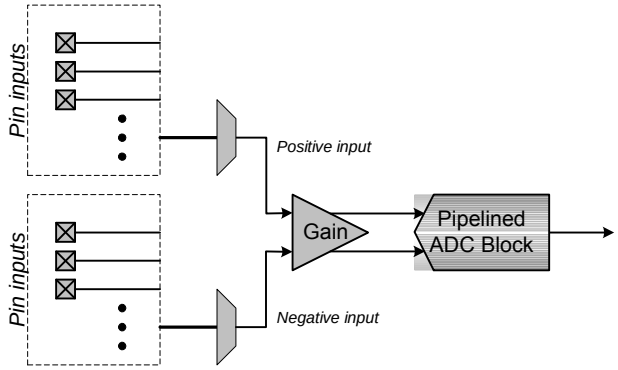


2.4.2 Differential Input with Gain Stage

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. When the gain stage is used, the propagation delay through the ADC block is increased by 1 ADC clock cycle.

Figure 2-4. Differential input with gain stage

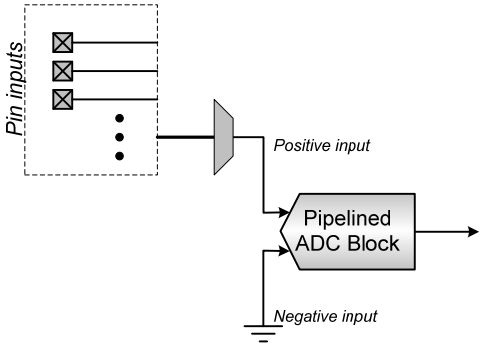


Note that the gain stage does not load the input and external signal source will see very high input impedance for channels that use the gain stage. This is useful for measuring weak signal sources. Details can be found in the datasheet for the device.

2.4.3 Single-ended Input

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 internally in the single-ended setting. If the ADC block is configured for signed mode single-ended input the negative input is connected to GND.

Figure 2-5. Single-ended input in signed mode

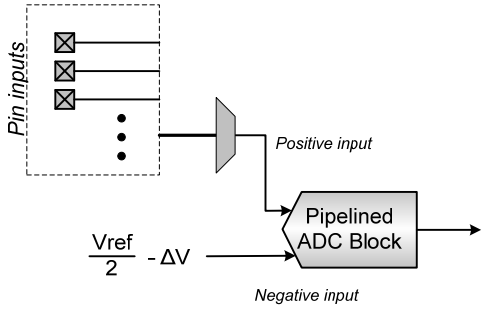


If the ADC block has been configured for unsigned mode the negative input is connected to Vref/2 – ΔV. ΔV is a fixed internally generated voltage of approximately 0.05*Vref. This offset *needs to be measured* by connecting the positive input to ground (GND). The offset will typically correspond to a value of about 200 when measured.

The advantage of ΔV is that it will be possible to measure a negative offset in the ADC block because ΔV will be larger than any offset. ΔV will allow the XMEAG ADC to be used in applications where it is essential to know and compensate for offset errors. The disadvantage is that some of the upper range is lost since any measurement above Vref – ΔV will saturate to the top value.

In addition to connecting the negative input the ADC will in unsigned single ended mode automatically add 2048 to the result. This gives an possible output range from 0 to 4096 as opposed to -2048 to 2047 for signed mode.

Figure 2-6. Single-ended input in unsigned mode



2.4.4 Internal Input

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, 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 internal DAC input can be used for calibration of the DAC. For more information about DAC configuration, please refer to the device datasheet or the application note “AVR1301: Getting Started with the XMEGA DAC”.

The Bandgap Reference could be used to measure an unknown external reference like a battery voltage. 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 2-7. Internal input in signed mode

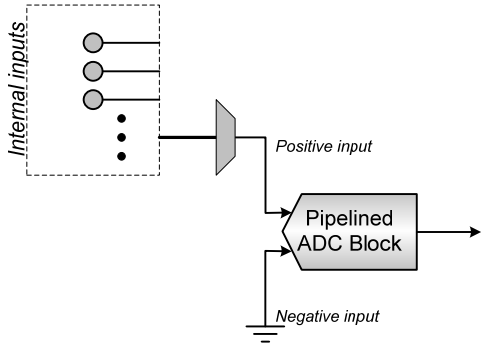
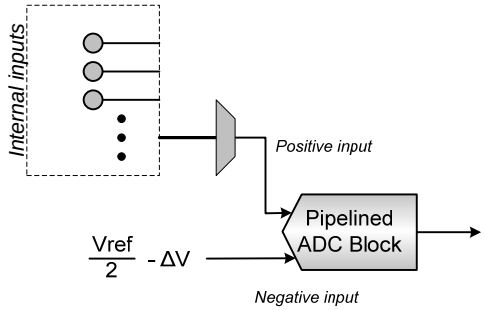


Figure 2-8. Internal input in unsigned mode



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 both Bandgap and Temperature Reference, hence should be enabled in due time before starting any conversions.

Note that input sampling speed of the internal inputs can be slower than the maximum conversion range of the device. (example: 100ksps/2Msps) See device datasheet for more information.

2.5 Conversion Result

2.5.1 Signed mode

In signed mode the conversion result from the ADC is:

$$RES = \frac{V_{INP} - V_{INN}}{V_{REF}} * GAIN * TOP$$

V_{INP} is the positive input and V_{INN} is the negative input to the ADC. GAIN corresponds to the gain setting used. GAIN is 1 if gain is not used. TOP is the top value given by the configured resolution, which is 2048 for 12 bit mode and 128 for 8 bit mode.

In signed mode the result is returned as a signed number represented on a two's complement format where the MSB represents the sign bit. In 12-bit right adjusted mode, the sign bit (bit 11) is padded to bits 12-15 to create a signed 16-bit number directly. In 8-bit mode, the sign bit (bit 7) is padded to the entire high byte.

With 12-bit resolution the range from $-V_{REF}$ to $+V_{REF}$ will be -2048 to +2047 (0xF800 - 0x07FF).

2.5.2 Unsigned mode

In unsigned mode the conversion result from the ADC is:

$$RES = \frac{V_{INP} - (-\Delta V)}{V_{REF}} * GAIN * TOP$$

V_{INP} is the positive input and V_{INN} is the internally connected negative input to the ADC. GAIN corresponds to the gain setting used. GAIN is 1 if gain is not used. TOP is the top value given by the configured resolution. For 12 bit mode TOP is 4096 and 8 bit mode TOP is 256.

The positive offset given by ΔV is typically $0.05*V_{REF}$. This typically corresponds to a measurement result of approximately 200 when the input pin is connected to ground. In order to measure this offset accurately the ADC should be configured as it will be used in the application (ie. voltages, speed and other settings) and the input pin should be connected externally to ground.

This offset is not compensated for automatically, and the software needs to subtract the measured positive offset from the conversion results.

With 12-bit resolution the range from GND to VREF – ΔV will be from approximately 200 to +4095 (0x00C8 - 0x0FFF).

2.6 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 *Conversion result Resolution* bitfield (`RESOLUTION`) in *Control Register B* (`CTRLB`).

2.7 Voltage References

The application can choose between the following voltage references (V_{REF}) for conversion results:

- Internal reference of 1.0 V
- Internal reference of V_{CC} / 1.6 V
- External reference (V_{REF})

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

Note that for the external reference (V_{REF}) the maximum voltage to be used is Vcc – 0.6V.

2.8 Conversion Speed

The ADC clock is derived from a prescaled 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. The ADC clock has to be set within the minimum and maximum recommended speed for the ADC module to guarantee correct operation. The ADC Clock is configured using the *Clock Prescaler* register (PRESCALER).

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 a signal at a much slower rate. For instance, an application could sample at a rate of 10kHz even if the ADC clock is 8MHz. However, it is not possible to sample at a rate higher than one fourth of the ADC clock speed since the maximum ADC clock is 1/4th of the peripheral clock. See device datasheet for more information.

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 Interrupt

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. Interrupts on compare match (above/below) can be configured individually on each channel, but the compare register is shared between all four virtual channels.

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.

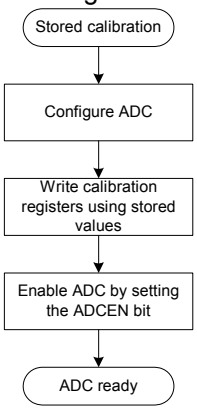
2.12 Calibration

The ADC module has been calibrated during production of the device. This calibration value is stored in the production signature row of the device. The calibration value compensates for mismatch between the individual steps of the ADC pipeline and it improves the linearity of the ADC.

The calibration value is not loaded automatically, and should always be loaded from the production signature row (ADCxCAL0/1) and written to the corresponding *ADC calibration registers* (CALL/CALH) before enabling the ADC.

Flowcharts for loading stored calibration settings are shown in Figure 2-9 below.

Figure 2-9. Using stored calibration settings



The calibration value is factory calibrated to the datasheet accuracy, and is not intended for user calibration.

The application note “AVR120: ADC Characteristics and Calibration” contains more information on characteristics of ADCs and how to compensate for gain and offset errors.

2.13 Improving accuracy

The accuracy of the XMEGA ADC depends on the quality of the input signals and power supplies. The following items should be taken into consideration for best possible accuracy of the ADC measurements:

- It is important to take great care when designing the analog signal paths like analog reference (V_{REF}) and analog power supply (A_{VCC}).
- Try to toggle as few pins as possible while the ADC is converting to avoid switching noise internally and on power supply. The ADC is most sensitive to switching the I/O pins that are powered by the analog power supply (PORTA/PORTB).
- Switch off unused peripherals by setting PRR registers to eliminate noise from unused peripherals.
- Put the XMEGA in the “Idle” sleep mode directly after starting the ADC conversion to reduce noise from the CPU.

- Use over-sampling to reduce increase resolution and eliminate random noise.

For randomly distributed noise using oversampling will help reducing any noise and improve accuracy. Using 8x oversampling will increase resolution by 2 bits, and due to the pipelined design of the ADC only take 8 additional ADC clock cycles.

See application note “AVR121: Enhancing ADC resolution by oversampling” for more information on oversampling.

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 0x01 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 0x01 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 A* (CTRLA) 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, as 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 for IAR Embedded Workbench.
- *adc_driver_asm.h* – ADC driver header file for the Assembly.
- *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
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
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
http://www.atmel.com/

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature