

Atmel AVR1300: Using the Atmel AVR XMEGA ADC

Atmel 8-bit Microcontrollers

PRELIMINARY

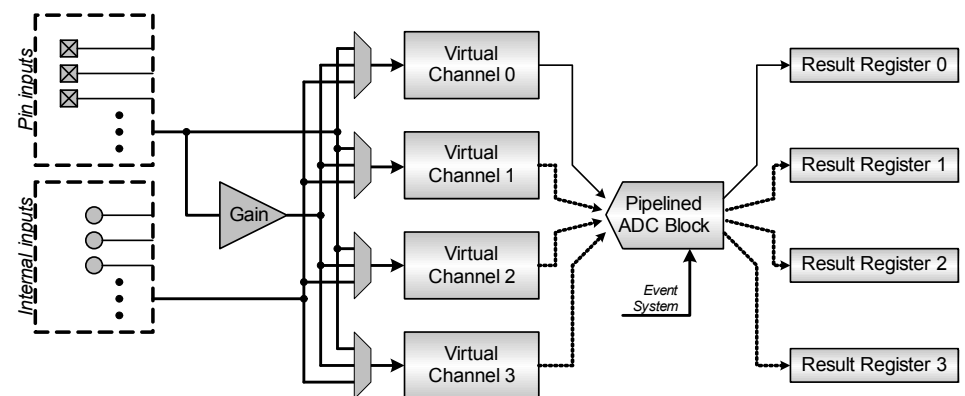
Introduction

The Atmel® AVR® 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. A wide range of 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. The examples are implemented in software framework (ASF 2.0), which is available for the Atmel AVR XMEGA and Atmel AVR UC3 microcontroller series.

Advanced usage, such as Direct Memory Access (DMA) and the XMEGA Event System, is outside the scope of this application note. Refer to the device datasheets and other relevant application notes such as [Atmel AVR1304: Using the XMEGA DMA Controller](#), and [AVR1001: Getting Started with the XMEGA Event System](#).

Figure 1. ADC Overview.



Features

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

Table of Contents

1. Module Overview	4
1.1 Pipeline Architecture and Virtual Channels	4
1.2 Gain Stage	6
1.3 Conversion Mode	6
1.4 Multiplexer Settings	7
1.4.1 Differential Input without Gain	7
1.4.2 Differential Input with Gain Stage	8
1.4.3 Single-ended Input	9
1.4.4 Internal Input	10
1.4.5 Temperature Sensor	11
1.5 Conversion Result	11
1.5.1 Signed Mode	11
1.5.2 Unsigned Mode	12
1.6 Result Presentation	12
1.7 Voltage References	12
1.8 Conversion Speed	13
1.9 Free-running Mode	14
1.10 Interrupts	14
1.11 Result Comparator Interrupt	14
1.12 Calibration	15
1.12.1 Offset Error	15
1.12.2 Offset Error – Single-ended Channels	16
1.12.3 Offset Error – Differential Channels	17
1.12.4 Gain Error	17
1.12.5 Non-linearity	18
1.12.6 Differential Non-linearity	19
1.12.7 Integral Non-linearity	19
1.12.8 Measurements and Compensation	19
1.12.9 Decoupling	19
1.12.10 Source Impedance	19
1.13 Tips for Improving Accuracy	20
2. Getting Started	22
2.1 Single Conversion	22
2.2 Multiple Channels	22
2.3 Free-running Mode	22
3. Advanced Features	24
3.1 DMA Controller	24
3.2 Event System	24
4. Driver Implementation	25
4.1 Files	25
4.2 Doxygen Documentation	25
5. Revision History	26

1. Module Overview

This chapter provides an overview of the functionality and basic configuration options of the ADC. Chapter 1 will walk you through the basic setup and usage of ADC on register level.

1.1 Pipeline Architecture and Virtual Channels

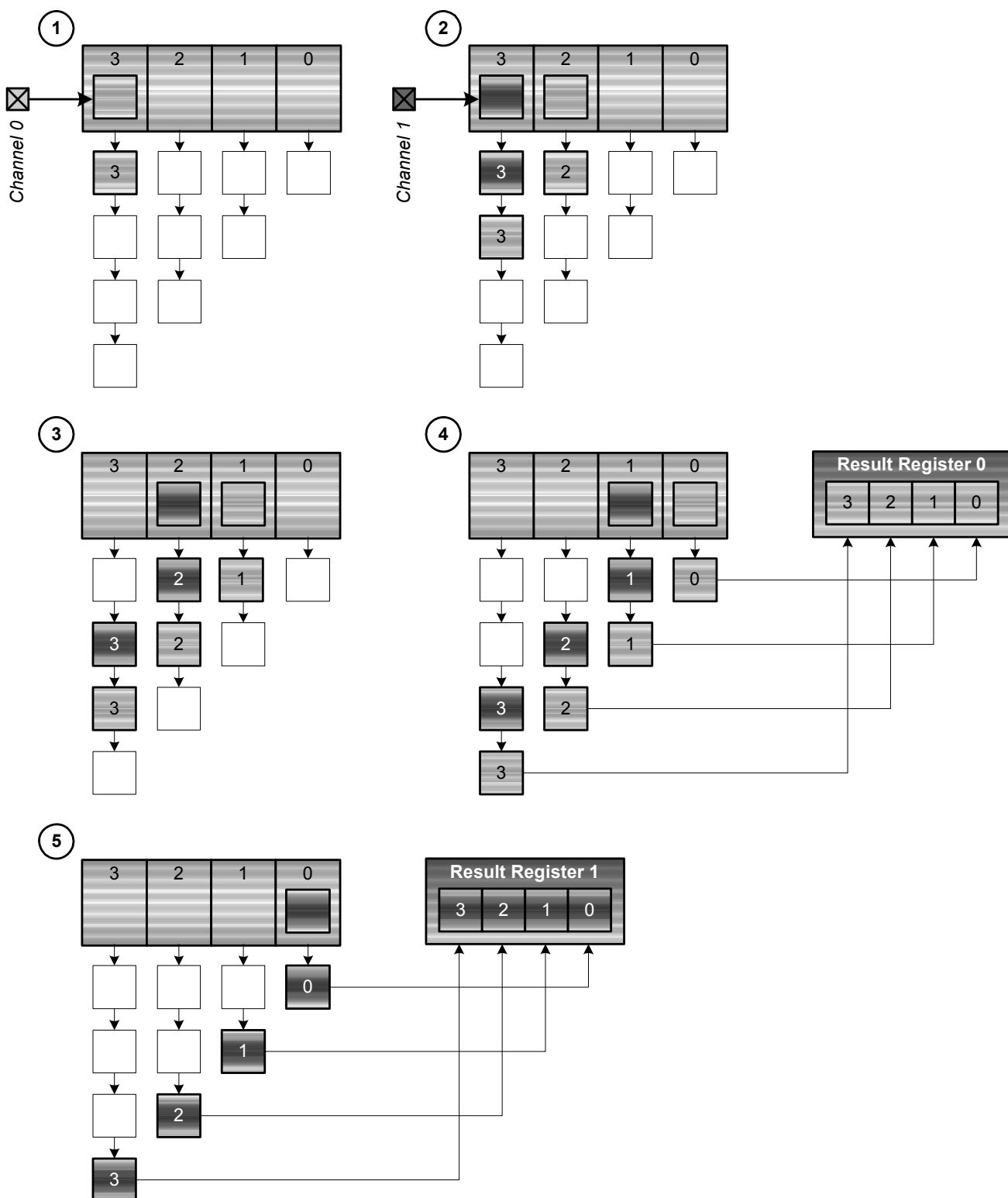
The ADC conversion block has a 12-stage pipelined architecture capable of sampling several signals in parallel. There are four input selection multiplexers with individual configurations. The separate configuration settings for the four multiplexers can be viewed as *virtual channels*, with one set of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

On the Atmel AVR XMEGA A series the multiplexer outputs can be sampled every ADC clock cycles. On the XMEGA D series a new measurement can be sampled once the previous conversion is done. Each signal propagates through the pipeline, where one bit is converted at each stage. In this way the ADC in the XMEGA A is capable of sampling one signal every ADC clock cycle, 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 seven ADC clock cycles for 12-bit conversions and five cycles for 8-bit conversions. If Gain is used the propagation time increases by one cycle. At full utilization the XMEGA A ADC delivers one result every ADC clock cycle while the XMEGA D ADC delivers one sample every 5 – 8 ADC clock cycle depending on operating mode. The relation between the XMEGA peripheral clock and the ADC clock is described in Section 1.8.

[Figure 1-1](#) on page 5, 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.

Note: The Atmel XMEGA D3/D4 families do not have a pipelined ADC and four virtual channels pr. ADC.

Figure 1-1. Simplified ADC Pipeline with TWO Propagating Signals.



All the four virtual channels have one *MUX Control* register ($CH_n.MUXCTRL$), one *Channel Control* register ($CH_n.CTRL$) and one *Result* register pair ($CH_n.RESL/CH_n.RESH$) each, in addition to several control bits distributed in shared registers.

1.2 Gain Stage

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

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 Atmel AVR XMEGA D series can also do 1/2x gain (div2). The *Gain Factor* bit field (GAINFACT) in the *Channel Control* register (CHn.CTRL) 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 module increases by one ADC clock cycle flat on the XMEGA A series when using the gain stage, on the XMEGA D series the propagation delay is depended of the gain setting.

- Propagation delay = 1 for 1/2x, 1x, 2x and 4x gain settings
- Propagation delay = 2 for 8x and 16x gain settings
- Propagation delay = 3 for 32x and 64x gain settings

To minimize the analog signal path for best possible ADC result it is recommended to disable gain when not needed.

1.3 Conversion Mode

The conversion block can be put in the *unsigned* or *signed* conversion mode. Changing between unsigned/signed and single-ended/differential modes will corrupt data already in the pipeline.

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. To be able to have zero-cross detection a ΔV is subtracted. The ΔV is approximately $0.05 \cdot V_{REF}$, so the ground level will be approximately 0.05 of the total value range ($0.05 \cdot 4095$ with 12-bit resolution). This will also limit the maximum input voltage of $0.05 \cdot V_{REF}$, so the maximum input voltage is $V_{REF} - \Delta V$. This is illustrated in Figure 1-2.

In signed mode the range is from negative to positive reference voltage, but the input voltage must be within GND and V_{REF} . Figure 1-2 shows the difference in conversion ranges.

Figure 1-2. Unsigned and Signed Conversion Mode.

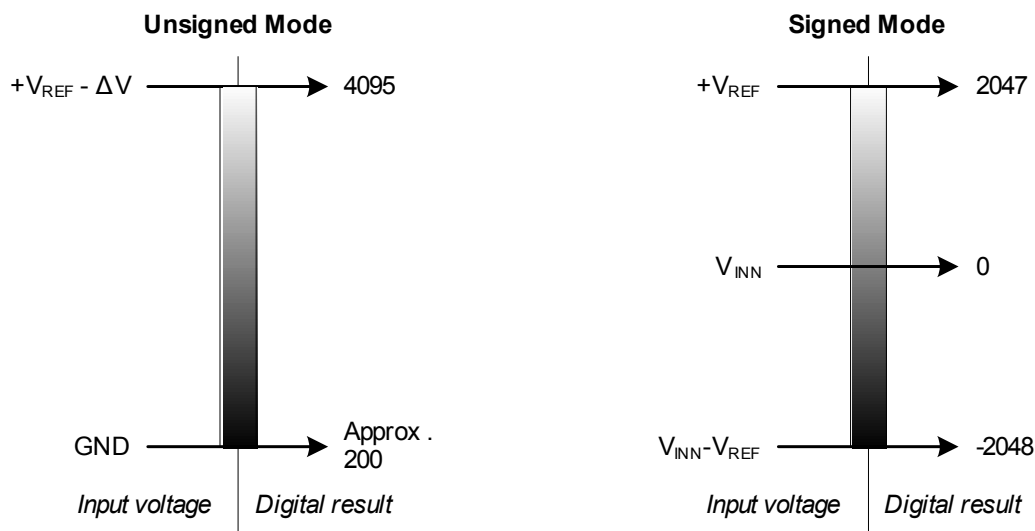


Figure 1-2 shows that the unsigned mode gives higher resolution on positive values than signed mode, but cannot convert negative values. The signed mode can convert negative values, but at the cost of lower resolution overall.

Note: Negative values are not negative inputs on the I/O pins, but higher voltage level on the negative input in respect to the positive input. Even though the resulting value can be negative, voltages below GND or above V_{CC} should under no circumstances be applied to any input port.

When the ADC uses *differential input*, signed mode must be used, when using single-ended input both signed and unsigned mode can be used.

Note: 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).

1.4 Multiplexer Settings

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

- XDifferential XInput without Gain (see Section 1.4.1)
- XDifferential Input with Gain Stage (see Section 1.4.2)
- XSingle-ended Input (see Section 1.4.3)
- XInternal Input (see Section 1.4.4)

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 only be connected to PORT A and ADC B can only be connected to PORT B. The positive input can be connected to any one of the eight input signals of the 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.

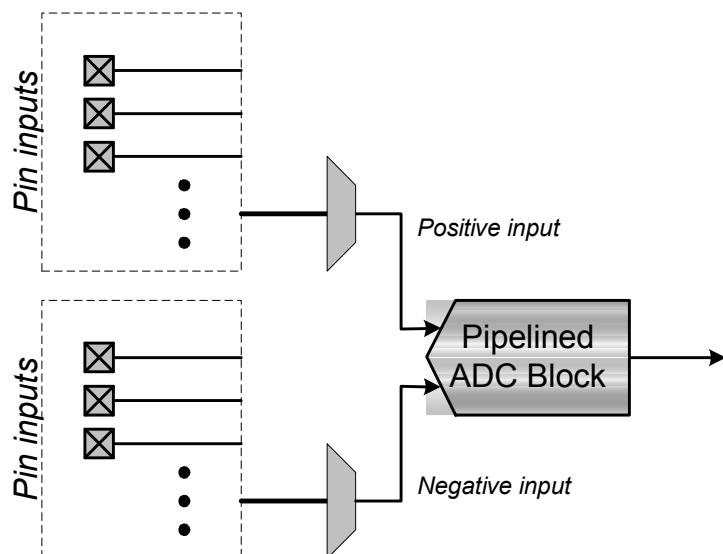
Refer to the datasheet to determine the number of ADCs and the devices pin configuration.

Note: The sampling capacitor is drained between each sample for all modes except for differential mode with gain. In the differential mode with gain, the charge on the sampling capacitor is maintained and this can be used to get a higher sample rate on slow changing signals. This can be used to get higher sampling rates on high impedance sources compared to single channel or differential channel (without gain) sampling. This will however propagate to other channels if the sample rate is too high compared to the source impedance.

1.4.1 Differential Input without Gain

With this setting, the MUX measures the difference between two input signals. In differential mode without gain all ADC inputs can be used for the positive input of the ADC but only the lower four pins can be used as the negative input. When using differential mode, offset can be measured quite easily by setting up the positive and negative input on the same pin, and the offset can be measure directly as the ADC does not need to know where the ground level is.

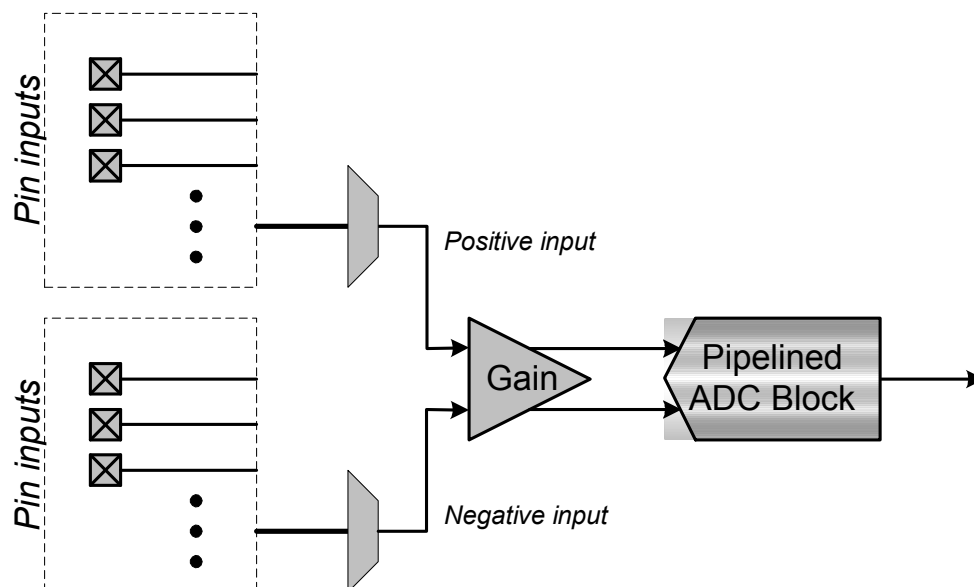
Figure 1-3. Differential Input without Gain Stage.



1.4.2 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. Maximum sampling speed while using the gain stage is 1 MSPS.

Figure 1-4. Differential Input with Gain Stage.



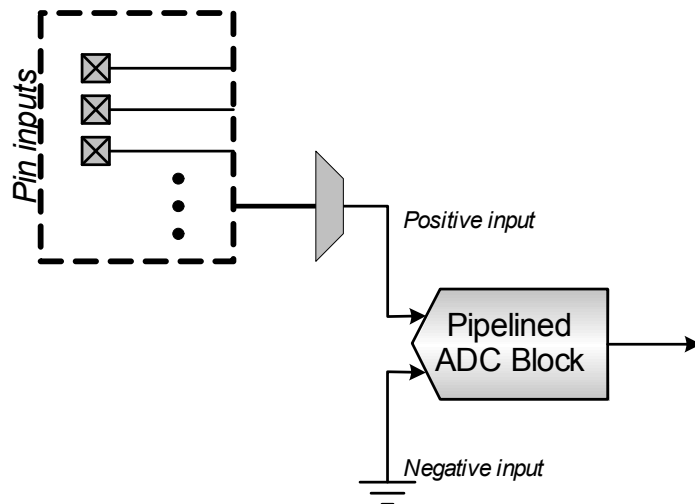
Note: The gain stage does not load the input; hence external signal sources will see very high input impedance for channels that use the gain stage. This is useful for measuring weak signal sources. More details can be found in the datasheet for the specific device.

The voltage on any of the two inputs can be between GND and V_{REF} , but the difference between them must not be larger than $V_{REF}/GAIN$ because this will saturate the ADC and the converted value will only equal the top value of the ADC.

1.4.3 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 to a defined level depending on if signed or unsigned mode is being used. For signed mode the negative input is tied to GND while in unsigned mode it is connected to $V_{REF}/2 - \Delta V$.

Figure 1-5. Single-ended Input in Signed Mode.

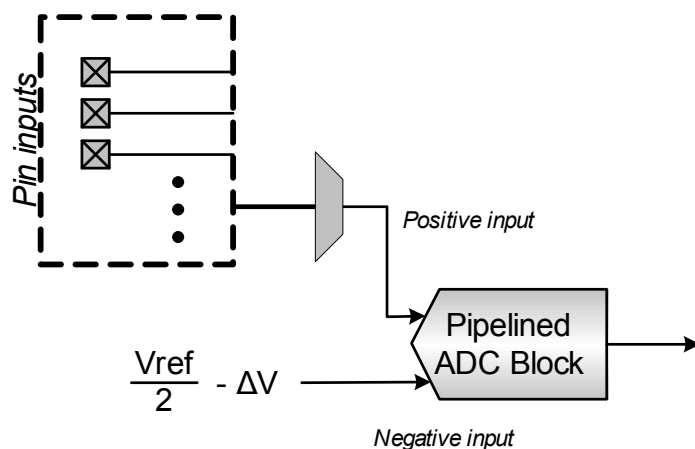


ΔV is a fixed internally generated voltage of approximately $0.05 \cdot V_{REF}$. This offset *needs to be measured* by connecting the positive input to ground (GND). The offset will typically correspond to a value of about 200 when measured.

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 Atmel XMEGA 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 $V_{REF} - \Delta V$ 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 a possible output range from 0 to 4096 as opposed to -2048 to 2047 for signed mode.

Figure 1-6. Single-ended Input in Unsigned Mode.



Note: Since the ADC is differential, unsigned mode is achieved by dividing the reference by two internally, resulting in an input range from V_{REF} to zero for the positive single ended input. The offset enables the ADC to measure zero cross detection in unsigned mode, and to calibrate any positive offset where the internal ground in the device is higher than the external ground.

1.4.4 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, $AV_{CC}/10$ (for supply voltage measurement) or Bandgap Reference.

Note: 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, refer to the device datasheet or the application note “[AVR1301: Using the XMEGA DAC](#)”.

The Bandgap Reference can be used as a reference to calculate the external reference, like a battery voltage, if the voltage is unknown. With a measurement of a known voltage (the Bandgap Reference, 1.1V) using an unknown reference, it is easy to calculate the actual voltage of the external reference.

Figure 1-7. Internal Input in Signed Mode.

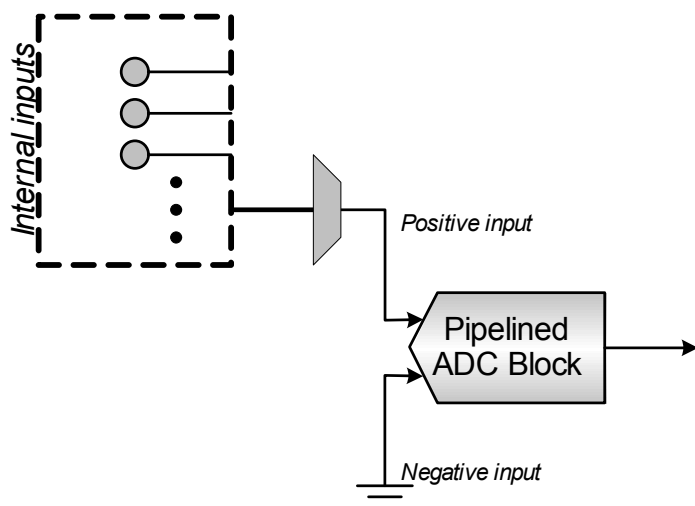
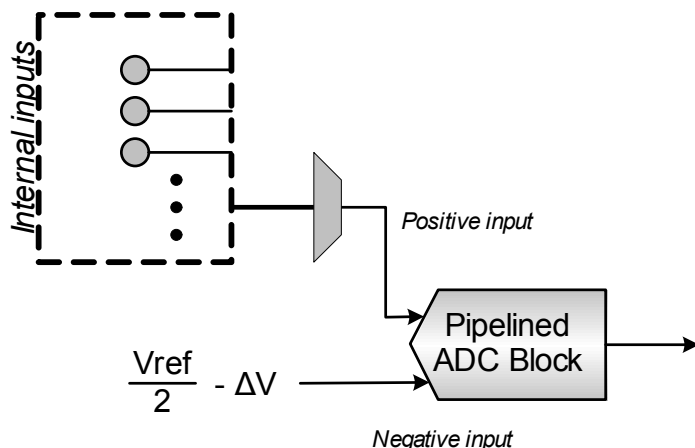


Figure 1-8. Internal Input in Unsigned Mode.



Note: 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 they should be enabled in due time before starting any conversions.

Note: Maximum sampling speed of the internal inputs is 125KSPS.

1.4.5 Temperature Sensor

The internal temperature sensor is linear, and is intended to give a rough approximation of the ambient temperature (not a PT100 sensor replacement). The target value at 0°K is 0mV from the analog sensor, resulting in “0x00” from the ADC (plus ΔV from single ended measurements – refer to the device manual/datasheet for this value). An approximate linear line can be made from the 0°K point to the production calibration value in the signature row. This value is stored in the signature row and corresponds to a temperature measurement done at 85°C (358°K) with a typical accuracy of $\pm 15^\circ\text{C}$. The inaccuracy will result in some gain error when measuring temperatures.

The measurement stored in the signature row is done in unsigned mode with 12-bit resolution with the internal 1V reference. V_{CC} is 3.2V and the ADC clock is 62.5kHz. The ADC setup has to be the same if this value is going to be used for calibration in the application.

The best way to get more accurate results is to do a 2-point calibration to get the incline of the curve. To do a 2-point calibration, select two temperatures where you can do the measurements where the temperature is known and is accurate. Do a measurement at both temperatures with the wanted setup of the ADC (mode, sample rate/frequency with a maximum of 125kHz, resolution). When you have these values, you can calculate the coefficient for the curve and you can use this in your application. The formula below can be used to calculate the temperature change per bit:

$$\text{Temp/bit} = \frac{\text{TempHigh} - \text{TempLow}}{\text{ValueHigh} - \text{ValueLow}}$$

For a single point calibration, use the same formula, but replace TempLow and ValueLow with 0°K and “0x00” result from the ADC (plus ΔV).

1.5 Conversion Result

1.5.1 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).

1.5.2 Unsigned Mode

In unsigned mode the conversion result from the ADC is:

$$RES = \frac{V_{INP} + \Delta V}{V_{REF}} * TOP$$

V_{INP} is the single ended input and $\Delta V = V_{REF} \times 0.05$. 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 $0.05 \times V_{REF}$. 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 (i.e. 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 offset from the conversion results.

With 12-bit resolution the range from GND to $V_{REF} - \Delta V$ will be from approximately 200 to +4095 (0x00C8 - 0x0FFF).

1.6 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: 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* bit field (**RESOLUTION**) in *Control Register B* (**CTRLB**).

The result will be stored in the result registers for each channel. The channels have separate flags to indicate when a new conversion is ready. If the result is not read before a new conversion is done, the current result will be lost.

The DMA can be set up to transfer the result from the result register into the SRAM when a new conversion is ready. This can be done for all channels when doing sweep and store all channels in one burst.

1.7 Voltage References

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

- INT1V - Internal reference of 1.0V
- INTVCC - Internal reference of $V_{CC} / 1.6V$
- AREFA - External reference pin on PORTA
- AREFB - External reference pin on PORTB
- AVCC/2 – Internal reference of $AV_{CC}/2$ for Atmel XMEGA D devices

The internal INT1V reference is a 1.00V reference from the bandgap of the device. The bandgap voltage is 1.10V and the reference voltage is 10/11 of the bandgap voltage, giving the 1.00V reference. The accuracy of the reference is dependent on the bandgap. The accuracy of the bandgap is stated in the device's datasheet.

The INTVCC is a reference voltage based on V_{CC} divided by 1.6. The accuracy is depending on the accuracy and stability of the analog supply voltage (AV_{CC}) and filtering should be used if the AV_{CC} is connected to the digital V_{CC} .

- Note: The external reference pin AREFA/B is shared with the DAC module. The voltage reference is configured using the *Reference Selection* bitfield (REFSEL) in the *Reference Control* register (REFCTRL). The external reference is located at pin 0 on PORTA and PORTB (AREFA and AREFB).
- Note: For the external references the maximum voltage to be used is $V_{CC} - 0.6V$ and the minimum voltage is 1V. The accuracy of the external reference is depending on the external circuitry and this has to be designed to satisfy the required accuracy for the ADC measurements.

1.8 Conversion Speed

The ADC clock is derived from a prescaled version of the Atmel AVR 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).

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 2MHz. However, it is not possible to sample at a rate higher than one fourth of the system clock speed since the maximum ADC clock is $1/4^{\text{th}}$ of the peripheral clock. For Atmel XMEGA A devices the ADC clock should not be set higher than 2 MHz, for Atmel XMEGA D devices the top frequency limit is 1.4Mhz. Lowest ADC clock frequency for both XMEGA A and XMEGA D devices are 100kHz. See device datasheet for more information.

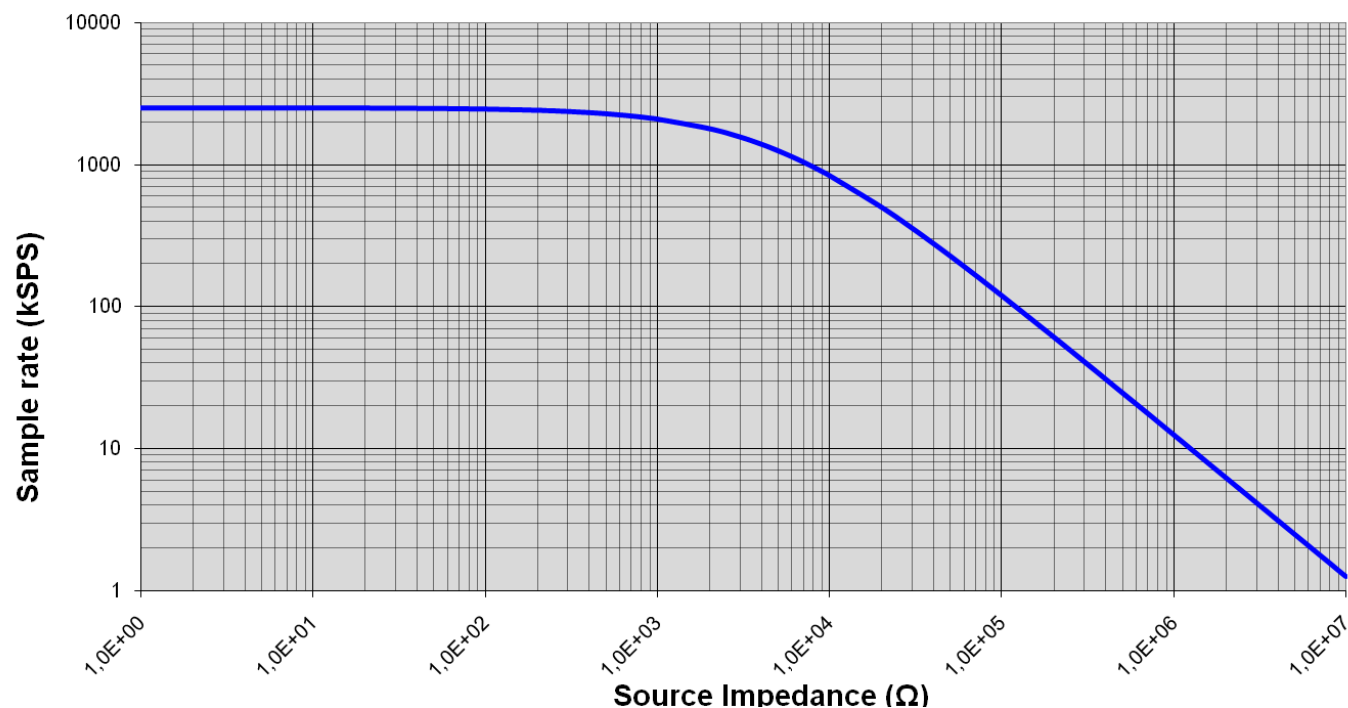
The conversion rate must satisfy the requirements for the given source impedance. If the sampling rate is too high compared to the source impedance, the results will not be accurate. It is important that you do not sample faster than the inclination rate of the signal to follow. The maximum sample rate is defined by the formula:

$$f_{ADC} \leq \frac{1}{2 \cdot (R_{source} + R_{channel}) \cdot C_{sample} \cdot \ln(2^{n+1})}$$

The values for C_{SAMPLE} and $R_{CHANNEL}$ ($R_{CHANNEL} = R_{CHANNEL} + R_{SWITCH}$) can be found in the datasheet for the device. The n represent the number of bits in the conversion and can be 8 or 12. The R_{SOURCE} is the impedance of the analog signal source which can be calculated from the circuitry or found in the datasheet of the device if using integrated sensors.

To give an illustration of how this will affect the sampling rate, we will use worst case numbers for $R_{CHANNEL}$, R_{SOURCE} and C_{SAMPLE} . This is 4.5kΩ for $R_{CHANNEL} + R_{SWITCH}$ and 5pF for C_{SAMPLE} . This will give the relationship between source impedance and the maximum sampling rate as shown in [Figure 1-9](#) page 14.

Figure 1-9. Sampling Rate vs. Source Impedance (log/log plot).



1.9 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: 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.

1.10 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, refer to the device datasheet or the application note [“AVR1305: XMEGA Interrupts and the Programmable Multi-level Interrupt Controller”](#).

1.11 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.

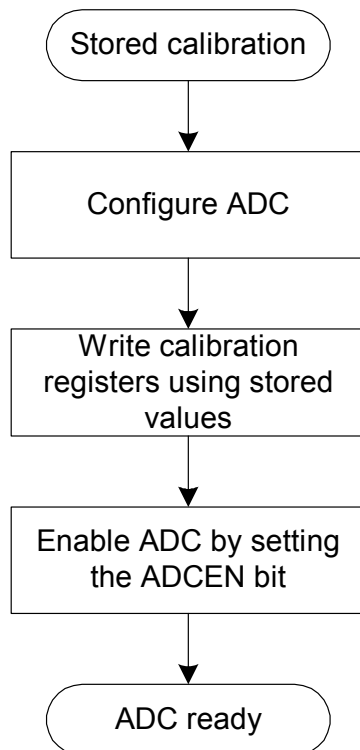
1.12 Calibration

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 1-10](#).

Figure 1-10. Using Stored Calibration Settings.



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

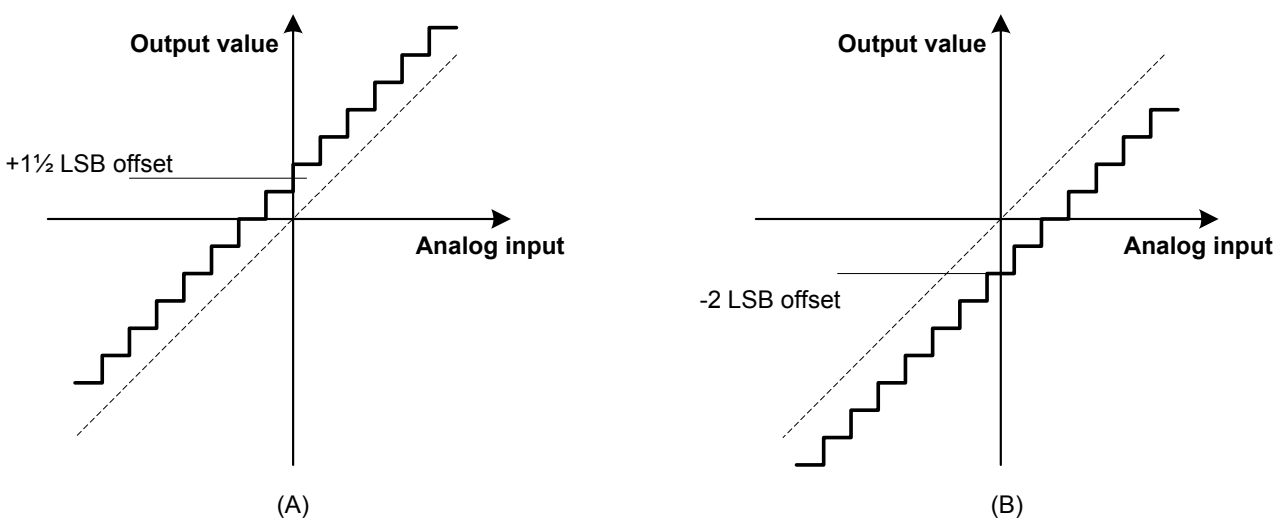
The application note “[AVR120: Characterization and Calibration of the ADC on an AVR](#)” contains more information on characteristics of ADCs and how to compensate for gain and offset errors.

1.12.1 Offset Error

The offset error is defined as the deviation of the actual ADC’s transfer function from the ideal straight line at zero input voltage.

When the transition from output value 0 to 1 does not occur at an input value of $\frac{1}{2}$ LSB, then we say that there is an offset error. With positive offset errors, the output value is larger than 0 when the input voltage approaches $\frac{1}{2}$ LSB from below. With negative offset errors, the input value is larger than $\frac{1}{2}$ LSB when the first output value transition occurs. In other words, if the actual transfer function lies below the ideal line, there is a negative offset and vice versa. Negative and positive offsets are shown in [Figure 1-11](#).

Figure 1-11. Examples of Positive (A) and Negative (B) Offset Errors.



Since single-ended conversion gives positive results only, the offset measurement procedures are different when using single-ended and differential channels.

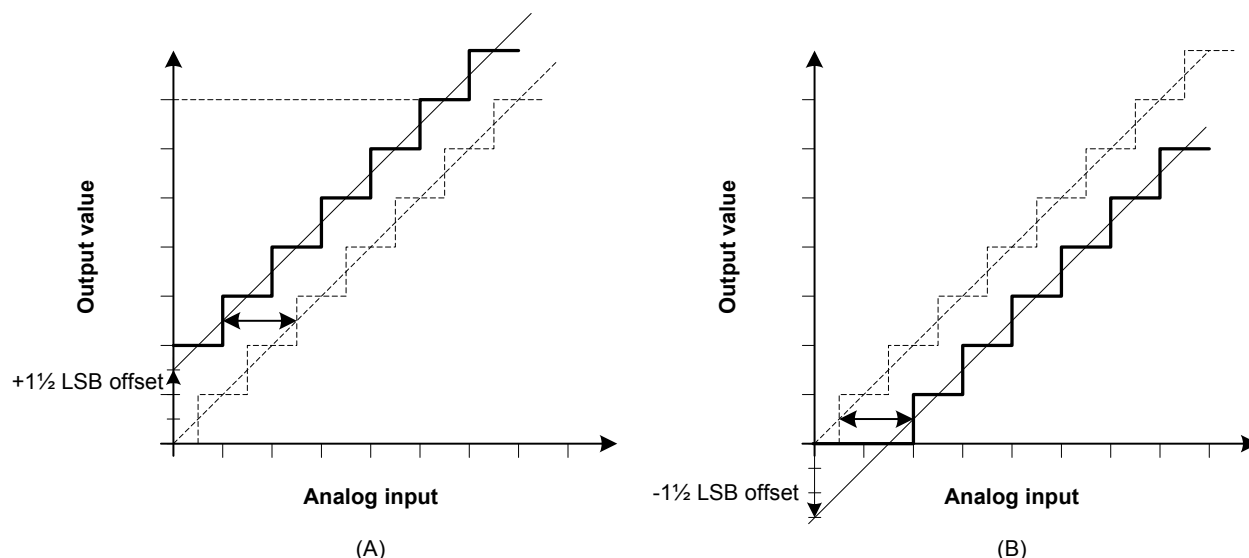
1.12.2 Offset Error – Single-ended Channels

To measure the offset error, increase the input voltage from GND until the first transition in the output value occurs. Calculate the difference between the input voltage for which the perfect ADC would have shown the same transition and the input voltage corresponding to the actual transition. This difference, converted to LSB, equals the offset error.

In [Figure 1-12A](#), the first transition occurs at 1 LSB. The transition is from 2 to 3, which equals an input voltage of $2\frac{1}{2}$ LSB for the perfect ADC. The difference is $+1\frac{1}{2}$ LSB, which equals the offset error. The double-headed arrows show the differences.

The same procedure applies to [Figure 1-12B](#). The first transition occurs at 2 LSB. The transition is from 0 to 1, which equals an input voltage of $\frac{1}{2}$ LSB for the perfect ADC. The difference is $-1\frac{1}{2}$ LSB, which equals the offset error.

Figure 1-12. Positive (A) and Negative (B) Offset Errors in Single-ended Mode.



To compensate for offset errors when using single ended channels, subtract the offset error from every measured value. Be aware that offset errors limit the available range for the ADC. A large positive offset error causes the output value to saturate at maximum before the input voltage reaches maximum. A large negative offset error gives output value 0 for the smallest input voltages.

1.12.3 Offset Error – Differential Channels

With differential channels, the offset measurement can be performed much easier since no external input voltage is required. The two differential inputs can be connected to the same voltage internally and the resulting output value is then the offset error. Since this method gives no exact information on where the first transition occurs, it gives an error of $\frac{1}{2}$ to 1 LSB (worst case).

To compensate for offset errors when using differential channels, subtract the offset error from every measured value.

In signed mode compensating for offset error caused by temperature, V_{CC} or reference drift can easily be done using two channels. E.g. set the ADC channel 0 MUXPOS to ADC pin 0 and MUXNEG to ADC pin 1. With channel 1 you do the opposite, set MUXPOS to ADC pin 1 and MUXNEG to ADC pin 0. Channel 0 should give you a positive value and channel 1 should give you a negative value. From these values you should get a real-time offset compensated result by applying this formula: $(CH0 + CH1 \cdot -1)/2$.

1.12.4 Gain Error

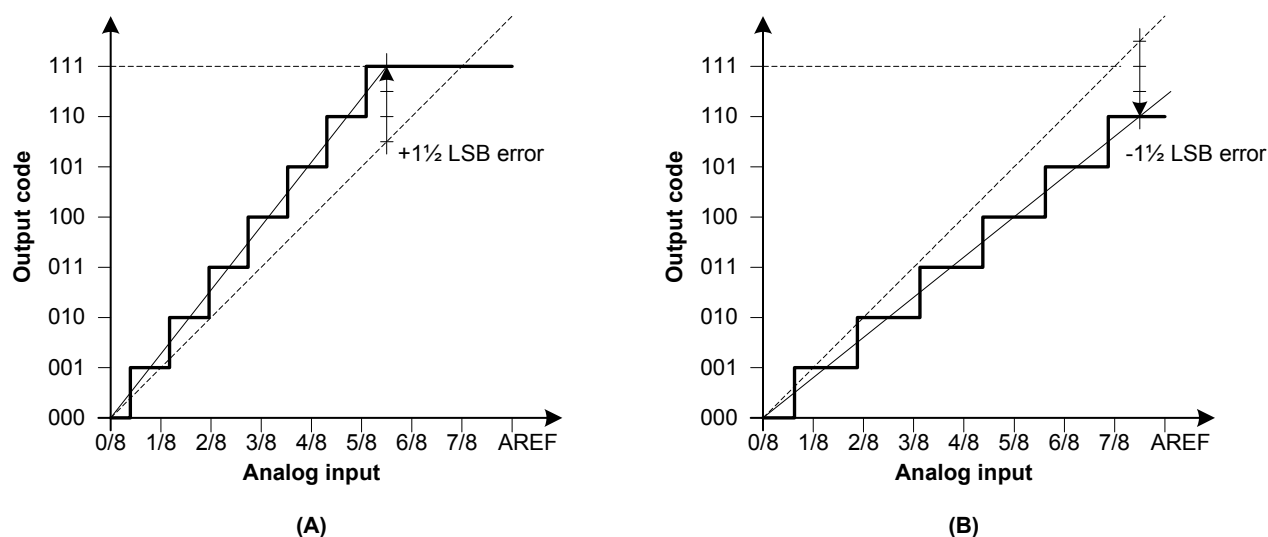
The gain error is defined as the deviation of the last output step's midpoint from the ideal straight line, after compensating for offset error.

After compensating for offset errors, applying an input voltage of 0 always give an output value of 0. However, gain errors cause the actual transfer function slope to deviate from the ideal slope. This gain error can be measured and compensated for by scaling the output values.

Run-time compensation often uses integer arithmetic, since floating point calculation takes too long to perform. Therefore, to get the best possible precision, the slope deviation should be measured as far from 0 as possible. The larger the values, the better precision you get. This is described in detail later in this document.

The example of a 3-bit ADC transfer functions with gain errors is shown in [Figure 1-13](#). The following description holds for both single-ended and differential modes.

Figure 1-13. Examples of Positive (A) and Negative (B) Gain Errors.

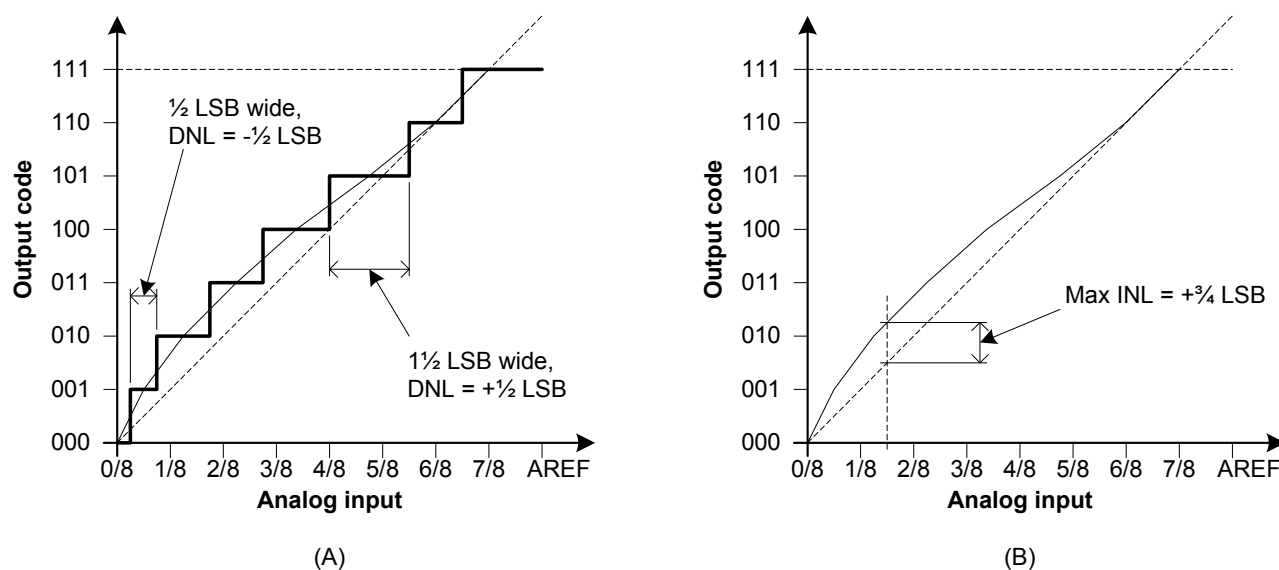


To measure the gain error, the input value is increased from 0 until the last output step is reached. The scaling factor for gain compensation equals the ideal output value for the midpoint of the last step divided by the actual value of the step.

1.12.5 Non-linearity

When offset and gain errors are compensated for, the actual transfer function should be equal to the transfer function of perfect ADC. However, non-linearity in the ADC may cause the actual curve to deviate slightly from the perfect curve, even if the two curves are equal around 0 and at the point where the gain error was measured. There are two methods for measuring non-linearity, both described below. Figure 1-14 shows examples of both measurements.

Figure 1-14. Example of Non-linear ADC Conversion Curve.



1.12.6 Differential Non-linearity

Differential Non-Linearity (DNL) is defined as the maximum and minimum difference between the step width and the perfect width (1 LSB) of any output step.

Non-linearity produces quantization steps with varying widths. All steps should be 1 LSB wide, but some are narrower or wider.

To measure DNL, a ramp input voltage is applied and all output value transitions are recorded. The step lengths are found from the distance between the transitions, and the most positive and negative deviations from 1 LSB are used to report the maximum and minimum DNL.

1.12.7 Integral Non-linearity

Integral Non-Linearity (INL) is defined as the maximum vertical difference between the actual and the perfect curve.

INL can be interpreted as a sum of DNLs. E.g. several consecutive negative DNLs raise the actual curve above the perfect curve as shown in [Figure 1-14](#). Negative INLs indicate that the actual curve is below the perfect curve.

The maximum and minimum INL are measured using the same ramp input voltage as in DNL measurement. Record the deviation at each conversion step midpoint and report the most positive and negative deviations as maximum and minimum INL.

1.12.8 Measurements and Compensation

It is important that DNL and INL values are measured after offset and gain error compensation. If not, the results will be infected by the offset and gain error and thus not reveal the true DNL and INL.

Non-linearity cannot be compensated for with simple calculations. Polynomial approximations or table lookups can be used for that purpose. However, the typical DNL and INL values are less than ± 2 LSB for the 12-bit ADC of the XMEGA, and are rarely of any concern in real life applications.

1.12.9 Decoupling

To get stable results, we need decoupling. This is needed both for the analog signals measured and for the reference used.

To be able to have a common reference for all signals measured over time, the voltage reference has to be exactly the same. To achieve this, the reference has to be decoupled with a large capacitor. For internal references this is not directly possible, but the AV_{CC} and V_{CC} have to be decoupled sufficiently. For both AV_{CC} and for the external references decoupling and filtering should be done. Using a filter inductor and a large capacitor will help on keeping the reference stable. The larger the capacitor, the better stability is achieved. A $1\mu\text{F}$ or larger capacitor is recommended. When using the ADC you have to wait until the capacitor is fully charged and stable, before measurements can be done. The time until stable have to be calculated from the rise time of the RC connection of the capacitor.

For the analog signals, decoupling should also be done. When having single ended signals, the decoupling should be done between the signal and ground and for differential signals the decoupling has to be between the positive and negative input. Decoupling of the signals is a more complex situation than the reference and this has to take the signals into account. If the signals are switching fast, the decoupling capacitor has to be lower. The decoupling capacitor should be as high as possible without changing the rise and fall time of the signal. It is therefore hard to give an exact value for the decoupling, and this has to be calculated as an RC circuit.

1.12.10 Source Impedance

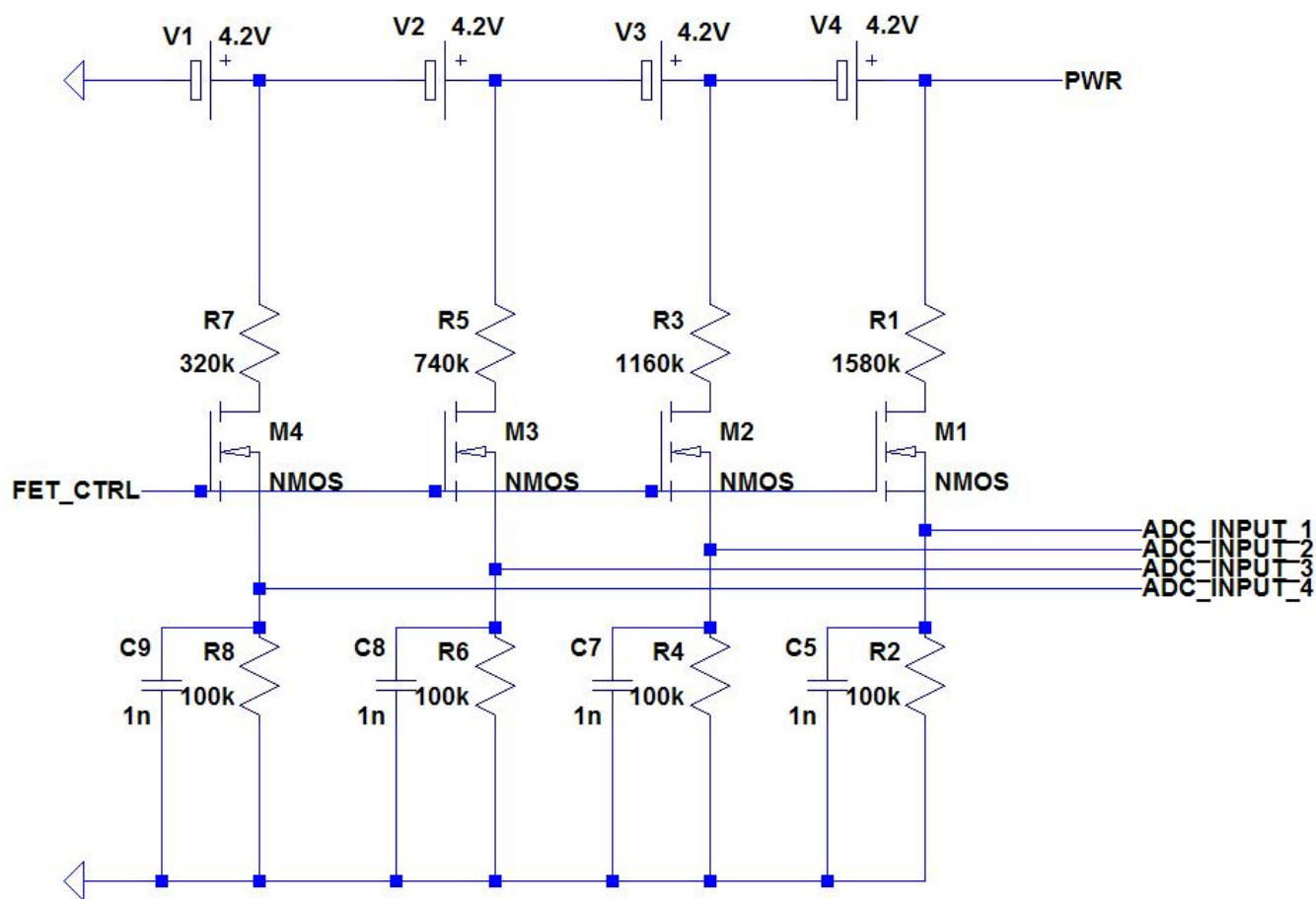
This is a very common problem when doing ADC designs. The source impedance is the sources capability to deliver charge to the internal sampling capacitor fast enough. If the internal capacitor is not charged to the same level as the analog signal, the result will be wrong.

When using a direct connection from a sensor IC, the impedance is usually stated in the datasheet of the device and it is easy to adjust the speed for the given impedance.

When the circuitry is made up by passive components, calculations have to be done to find the actual source impedance. For example, a resistor ladder dividing a high voltage down to a voltage that can be handled by the microcontroller can have very high impedance as large resistors are used to get the voltage sufficiently down. The solution is then to either lower the sample rate to be able to measure the correct signal or to lower the resistance value. Lowering the resistance would cause more leakage current and this can again be solved by adding switching FET to enable and disable the analogue source. An example of this can be seen in [Figure 1-15](#).

Another thing to be aware of in such an application is the time for the source to be stable, as the decoupling will take some time to charge to the correct level after enabling the analog source.

Figure 1-15. Schematic.



1.13 Tips for Improving Accuracy

The accuracy of the Atmel AVR 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:

- Understand the ADC, its features and how they are intended used
- Understand the applications requirements
- Make sure the source impedance is not too high compared to the sampling rate that is used. If the source impedance is too high, the internal sampling capacitor will not be charged to the correct level and the result will not be accurate

- It is important to take great care when designing the analog signal paths like analog reference (V_{REF}) and analog power supply (AV_{CC}). Filtering should be used if the analog power supply is connected to the digital power supply
- Avoid having the analog signal path close to digital signal path with high switching noise (i.e. communication lines, clock signals)
- Consider decoupling of the analog signal. Decoupling between signal and ground for single ended inputs and decoupling between the differential signal pair for differential measurements
- 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 Atmel XMEGA in the “Idle” sleep mode directly after starting the ADC conversion to reduce noise from the CPU
- Load the calibration values from the signature row into the calibration register in the ADC
- Use the lowest gain possible to avoid amplifying external noise
- Wait until the ADC, reference or sources are stabilized before sampling as some sources (e.g. bandgap) need time to stabilize after they are enabled
- Apply offset and gain calibration to the measurement
- Use over-sampling to increase resolution and eliminate random noise
- JTAG debugging interface create a lot of noise and should not be used while debugging an ADC application
- On the Atmel STK[®]600 starter kit the decoupling capacitors are far from the actual unit

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

See application note “[AVR121: Enhancing ADC resolution by oversampling](#)” for more information on oversampling.

2. Getting Started

This chapter 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: This chapter only covers manual polling of status bits. Interrupt control is not covered, but is an easy step after studying the application note “[AVR1305: XMEGA Interrupts and the Programmable Multi-level Interrupt Controller](#)”.

2.1 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. Wait for the ADC start-up time (typ. max. 24 ADC clocks)
- 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

2.2 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. Wait for the ADC start-up time (typ. max. 24 ADC clocks)
- 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

2.3 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. Wait for the ADC start-up time (typ. max. 24 ADC clocks).

- 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: 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.

3. Advanced Features

This chapter 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.

3.1 DMA Controller

Instead of using interrupt handlers to read and process the result registers, it is possible to use the Atmel AVR 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, refer to the device datasheet or the application note “[Atmel AVR1304: Using the XMEGA DMA Controller](#)”.

3.2 Event System

To improve conversion timing and further offload work from the CPU, the ADC is connected to the Atmel 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, refer to the device datasheet or the application note “[AVR1001: Getting Started with the XMEGA Event System](#)”.

4. Driver Implementation

This application note includes a source code package based on ASF 2.0 (Application Software Framework). The source code is compatible with Atmel AVR Studio®.

Note: 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. Refer to the driver source code and device datasheet for more details.

4.1 Files

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

4.2 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 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.

5. Revision History

Doc. Rev.	Date	Comments
8032I	05/2013	Chapter 3.4.5 has been changed. Also several other minor changes have been made. A new template is applied
8032H	12/2010	Doxygen updated
8032G	12/2010	Typos fixed
8032F	11/2010	Minor typos fixed
8032E	10/2010	Several corrections and a new section in chapter 3.12.3 is added
8032D	02/2010	Minor typos fixed
8032C	09/2009	Figures updated
8032B	08/2009	Several changes and updates
8032A	10/2008	Initial document release

**Atmel Corporation**

1600 Technology Drive
San Jose, CA 95110
USA

Tel: (+1)(408) 441-0311

Fax: (+1)(408) 487-2600

www.atmel.com

Atmel Asia Limited

Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG

Tel: (+852) 2245-6100

Fax: (+852) 2722-1369

Atmel Munich GmbH

Business Campus
Parking 4
D-85748 Garching b. Munich
GERMANY

Tel: (+49) 89-31970-0

Fax: (+49) 89-3194621

Atmel Japan G.K.

16F Shin-Osaki Kangyo Building
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN

Tel: (+81)(3) 6417-0300

Fax: (+81)(3) 6417-0370

© 2013 Atmel Corporation. All rights reserved. / Rev.: 8032I-AVR-05/2013

Atmel®, Atmel logo and combinations thereof, AVR®, AVR Studio®, Enabling Unlimited Possibilities®, STK®, XMEGA®, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

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 THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, 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 AND 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 products 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 products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.