AVR 8-bit Microcontrollers

AVR126: ADC of megaAVR in Single Ended Mode

APPLICATION NOTE

Introduction

Atmel® megaAVR® devices have a successive approximation Analog-to-Digital Converter (ADC) capable of conversion rates up to 15ksps with a resolution of 10-bits. It features a flexible multiplexer, which allows the ADC to measure the voltage at multiple single ended input pins and an internal channel from bandgap reference in the device. Single ended input channels are referred to ground.

This application note describes the basic functionality of the ADC in Atmel megaAVR devices in Single ended mode with code examples on Atmel ATmega88 to get started. The code examples are written in 'C' language and have been tested on the Atmel STK®600 starter kit for functionality.

Features

- Up to 10-bit resolution
- Up to 76.9ksps for Atmel ATmega88
- Up to 15ksps at maximum resolution
- Auto triggered and single conversion mode
- Optional left adjustment for ADC result read out
- Sleep mode noise canceler
- Driver source code included for ATmega88
  - ATmega88 ADC - Single conversion mode
  - ATmega88 ADC - Free running mode and conversion complete interrupt
  - ATmega88 ADC - Auto triggering using Timer0 compare event as trigger source
  - ATmega88 ADC - Measurement of bandgap reference voltage
  - ATmega88 ADC - Noise Reduction Sleep

Table of Contents

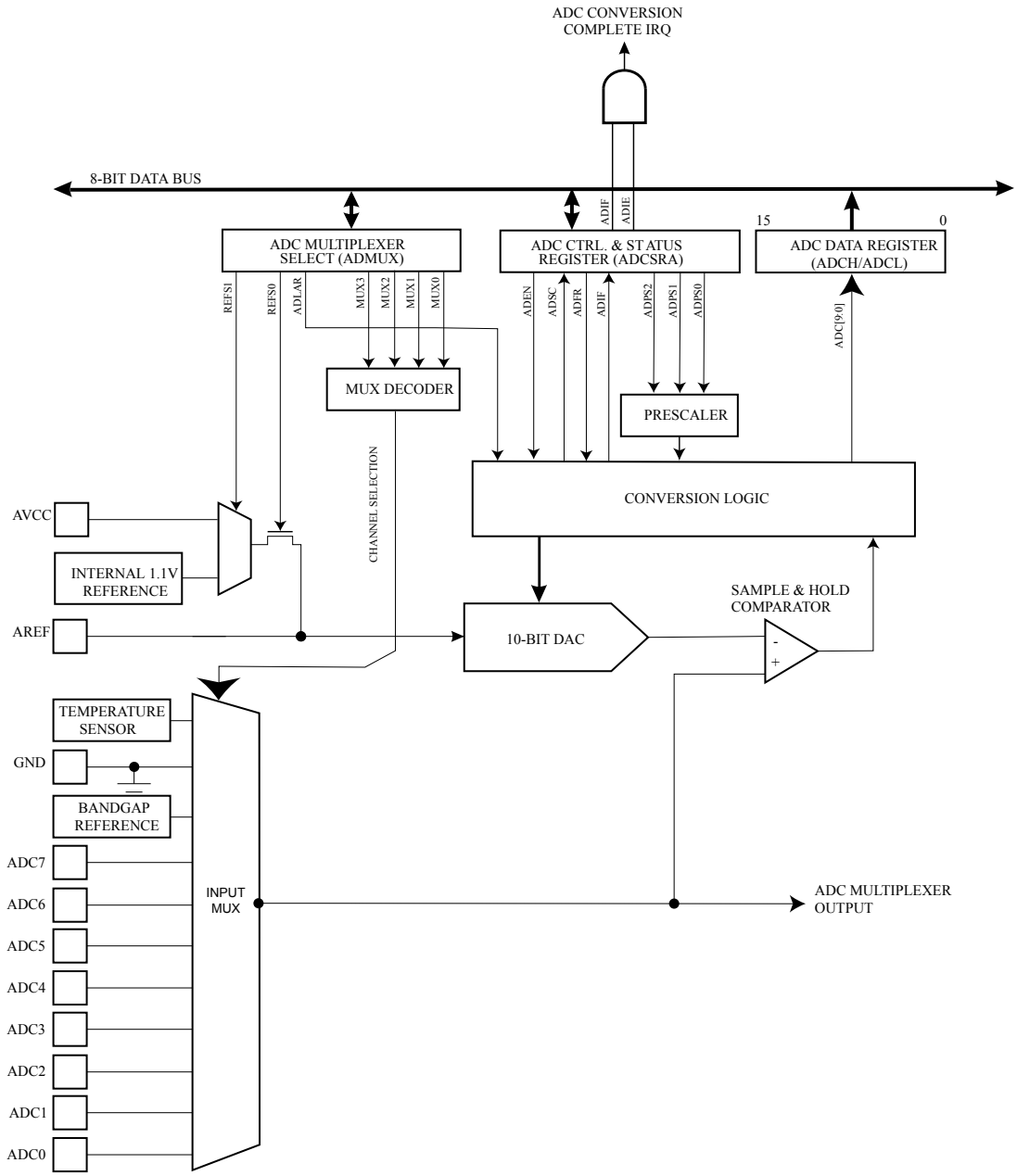
1. Module Overview

This chapter provides an overview of the functionality and basic configuration options of the ADC whereas the next section describes the basic steps to configure and run the ADC module with register description details.

1.1. ADC Operation

The ADC module converts the analog input voltage to a 10-bit digital value. The minimum value represents GND and maximum value denotes the reference voltage used.

Figure 1-1. Analog to Digital Converter Block Schematic



To make use of the ADC, the power reduction bit (PRADC) in the Power Reduction Register (PRR) must be written to 0. By default this bit has value 0, thus clearing this bit is required only if the ADC was

shutdown previously. To enable the ADC, the ADEN bit in the ADCSRA register must be set. The ADC module (the ADEN bit in the ADCSRA register) must be disabled before disabling in PRR.

The reference voltage is chosen by the REFS1:0 bits in the ADMUX register. The possible reference voltages are internal 1.1V or internal AVCC or external reference (AREF) pin. The analog input channel for conversion is selected by the MUXn bits in the ADMUX register. This includes the ADC input pins, internal voltage from fixed band gap reference voltage, and ground. The channels selected for conversion will not go into effect until this ADEN bit is set.

Before entering Sleep mode, the ADC module can be disabled by clearing ADEN bit. This reduces the power consumption caused by ADC.

The ADC's 10-bit digital value after conversion is stored in ADCH and ADCL result registers. The ADCH holds the higher byte and ADCL holds the lower byte. Optionally left adjustment of the result can be done by setting the ADLAR bit in the ADMUX register if required.

If ADLAR is enabled and the application needs only eight bit accuracy then it is sufficient to read the ADCH. Otherwise ADCL must be read first followed by ADCH, to ensure that the content of the data registers belong to the same conversion. Access to ADC is blocked, once ADCL is read. It is re-enabled only after ADCH is read.

The ADC module has one interrupt source which can be triggered when conversion completes. If an interrupt occurs between reading ADCL and ADCH, the result will be lost.

1.2. Input Sources

The input sources for the ADC are the analog voltage inputs that the ADC can measure and convert.

Two types of measurements can be selected:

- Single ended input
- Internal input

1.2.1. Internal Inputs

The internal bandgap analog voltage or ground signal can be selected as input and measured by the ADC. This bandgap voltage is an accurate voltage reference inside the microcontroller, which is the source for internal voltage reference.

1.2.2. Single Ended Input

For single ended measurements all analog input pins can be used as inputs. All single ended channels are referred to GND. The analog input voltages cannot be more than the reference voltage selected for ADC.

1.3. Starting a Conversion

In single conversion mode the ADSC bit in the ADCSRA register must be written a logical one to start the ADC conversion. This bit remains at logic high while conversion is in progress and is cleared by the hardware, once the conversion is complete.

In auto triggered mode, conversion is triggered automatically by various sources. To enable auto triggering, the ADATE bit in the ADCSRA register must be set. The source of trigger can be selected with the help of ADC Trigger Select bits (ADTS2:0) in ADCSRB register. The auto triggering mode provides a method of starting conversions at fixed intervals which is configurable based on the triggering source.

The interrupt flag will be set even if the specific interrupt or global interrupts are disabled. Thus a conversion can be triggered using ADIF flag without causing an interrupt. Note that ADIF must be cleared manually in order to trigger at next interrupt event in auto triggered mode. If the ADC runs in free running mode, in which next conversion is triggered once the previous conversion completes. For free running mode, the ADC will perform successive conversions independent of whether the ADIF is cleared or not. The first conversion must be started by setting ADSC bit.

1.4. ADC Clock and Conversion Timing

The ADC can prescale the system clock to provide an ADC clock that is between 50kHz and 200kHz to get maximum resolution. If an ADC resolution of less than 10-bits is required, then the ADC clock frequency can be higher than 200kHz. At 1MHz it is possible to achieve up to eight bits of resolution.

The prescaler value is selected with ADPS bits (ADPS2:0) in ADCSRA. When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle.

The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry. Then, for further conversions, it takes 13 ADC clock cycles (13.5 for Auto triggered conversions). When band gap voltage is used as input to ADC it will take a certain time for the voltage to stabilize. Start-up time for the bandgap reference voltage is available in the datasheet section *System and reset characteristics*.

1.5. Changing Channel or Reference Selection

Bits MUXn and REFS1:0 bits in the ADMUX register are single buffered through a temporary register to which the CPU has random access. If auto triggering is used, then ADMUX can be safely updated in the following ways:

- When ADATE or ADEN is cleared
- During conversion, minimum one ADC clock cycle after the trigger event
- After a conversion, before the Interrupt Flag used as trigger source is cleared. In these ways, the new settings will affect the next ADC conversion.

In single conversion mode, the channel must be selected before starting the conversion. The channel can be changed one clock cycle after setting ADSC bit, but it is better to wait for the conversion to complete before changing the channel.

In Free Running mode, select the channel before starting the first conversion. It can be changed one clock cycle after setting the ADSC bit but it is better to wait for the conversion to complete while changing the channel. Since the next conversion has already started automatically, the changes will be reflected in the subsequent conversion.

1.6. ADC Noise Canceler

The Atmel megaAVR ADC has a noise canceler that enables conversion during sleep mode which reduces the noise induced from CPU core and other I/O peripherals. This feature is available in ADC noise reduction and idle mode. To use this feature:

- Make sure that the ADC is enabled and is not busy converting. Single Conversion mode must be selected and the ADC conversion complete interrupt must be enabled.
- Enter ADC noise reduction mode (or idle mode). The ADC will start a conversion once the CPU has been halted.

- If no other interrupts occur before the ADC conversion completes, the ADC interrupt will wake up the CPU and execute the ADC conversion complete interrupt routine. If another interrupt wakes up the CPU before the ADC conversion is complete, that interrupt will be executed, and an ADC conversion complete interrupt request will be generated when the ADC conversion completes. The CPU will remain in active mode until a new sleep command is executed.

1.7. Conversion Result

After the ADC completes conversion (ADIF flag set) then the 10-bit result will be available in the ADCH and ADCL registers.

For single conversion, the result is:

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

where V_{IN} represents analog input voltage and V_{REF} represents the selected reference voltage. 0x000 represents GND and 0x3FF represents the reference voltage minus one LSB.

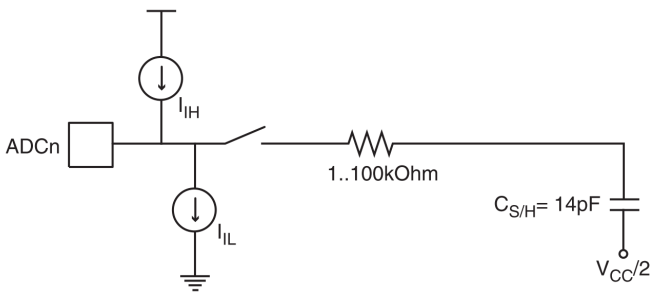
1.8. Analog Input Circuitry

The analog input circuitry for single ended channels is shown in the following figure. An analog source applied to the ADC input pin is subjected to pin capacitance and input leakage of that pin even if is not selected as input for ADC. When particular channel is selected, the source must drive the 'sample and hold' capacitor through the series (combined) resistance in the input path.

The ADC module is optimized for analog signals with an output impedance of 10kΩ or less. If such a source is used then the sampling time will be negligible.

If the source impedance is higher than 10kΩ then the time taken to charge the capacitor will increase (which can vary widely) and may produce inaccurate results. For example if the voltage divider used at the ADC input uses a resistor network, make sure that the source impedance is less than 10kΩ. Low impedance must be used for slowly varying signals since this minimizes the time for charge transfer. It is recommended to remove the frequency components higher than Nyquist frequency ($f_{ADC}/2$) with a low pass filter (to avoid distortion from unpredictable signal convolution).

Figure 1-2. Analog Input Circuitry



1.9. Best Practices for Improving ADC Performance

The accuracy of ADC depends on the quality of the input signals and power supplies. The following points should be taken into consideration to improve the accuracy of the ADC measurements.

- Understand the ADC, its features and how they are intended to be used

- Understand the application requirements
- It is important to take great care when designing the analog signal paths such as analog reference (V_{REF}) and analog power supply (AVCC). The AVCC is supply voltage to the ADC.
- Filtering should be used if the analog power supply (AVCC) is connected to digital power supply. That is, the AVCC pin on the device should be connected to the digital supply pin (VCC) via an LC network. For more information on this LC network connection, refer the respective Atmel megaAVR device datasheet. Note that the AVCC and VCC are internally connected by high impedance path.
- AVCC must not differ more than ±0.3V from VCC for Atmel megaAVR devices
- The reference voltage can be made more immune to noise by connecting a capacitor between AREF pin and ground
- Keep analog signal paths as short as possible. It is also important that the impedance on the PCB tracks for the ADC channels are not high which will result in longer charging period for the sample/hold capacitor of the ADC.
- Make sure analog tracks run over the analog ground plane
- Avoid having the analog signal path close to a digital signal path with high switching noise (that is, communication lines and clock signals)
- Consider decoupling of the analog signal between signal input and ground for single-ended inputs
- For differential signals the decoupling has to be between the positive and negative inputs. The decoupling capacitor value depends on the input signal. If the signals are switching fast, the decoupling capacitor must be lower.
- Ensure that the source impedance is not too high with respect to the sampling rate. If source impedance is too high, the internal sampling capacitor will not be charged to the correct level and the result will not be accurate.
- Try to avoid toggling of port pins while the ADC conversion is in progress, to prevent the switching noise from affecting the accuracy. The ADC is especially more sensitive to switching of the I/O pins (**PORTC**) that are powered by the analog power supply.
- Disable the digital input on the I/O pin corresponding to the ADC input channel to minimize the power consumption
- Switch off the unused peripherals by setting the respective bit in PRR register to eliminate any noise from unused peripherals
- Apply offset and gain calibration to the measurement to improve the accuracy
- Use ADC Noise Reduction sleep mode to get more accurate results
- Before triggering an ADC conversion, wait until the ADC completes any ongoing conversions. Ensure that enough time is given for the reference and input source to be stabilized. For example, the bandgap voltage needs certain amount of time to stabilize when it is selected as ADC input.
- Use oversampling to increase resolution and eliminate random noise
- Whenever the input MUX setting or reference voltage selection is modified, it is recommended to discard the first conversion result
- When switching to a differential channel (with gain settings), the first conversion result may have a poor accuracy due to the required settling time for the automatic offset cancellation circuitry. Thus it is better to discard the first sample result.
- Linear interpolation methods such as one point (offset) calibration and two point (offset and gain) calibration method can be used based on the application needs

2. Getting Started

This chapter walks you through the basic steps for getting started with simple ADC conversion and experimenting with its MUX settings. The necessary registers are described along with relevant bit settings. In the code supplied with this application note, multiple steps which write to the same register are combined and the effective value is written to the register in a single step for easiness.

2.1. General Instructions to Test the Code on STK600

- Place the Atmel ATmega88 device on the STK600 using the specific routing card and socket card (STK600-RC032M-29 and STK600-TQFP32)
- Try accessing the device from the menu **Tools > Device Programming** in Atmel Studio
- The voltage from AREF1 can be used as input to the ADC. Adjust the AREF1 voltage via the menu **Tools > Device Programming > Board settings** in Atmel Studio.
- Connect AREF1 on the STK600 to the ADC input channel ADC0 (PC0) used in the examples
- On the STK600, connect PB0 to an LED. This pin is used in the following examples to give visual indication.

2.2. Single Conversion Mode

Task: Single Conversion on ADC channel 0

In this program the `initialize()` routine is used to initialize the ADC module. The `convert()` routine has to be called whenever the application needs an ADC conversion.

1. Set the MUX bit fields (MUX3:0) in ADC's MUX register (ADMUX) equal to 0000 to select ADC Channel 0.
2. Set the ADC Enable bit (ADEN) in ADC Control and Status Register A (ADCSRA) to enable the ADC module.
3. Set the ADC Pre-scalar bit fields (ADPS2:0) in ADCSRA equal to 100 to prescale the system clock by 16.
4. Set the Voltage Reference bit fields (REFS1:0) in ADMUX equal to 11 to select Internal 1.1V reference.
5. Set the Start Conversion bit (ADSC) in ADCSRA to start a single conversion.
6. Poll (wait) for the Interrupt Flag (ADIF) bit in the ADCSRA register to be set, indicating that the conversion is completed.
7. After ADIF bit becomes high, read the ADC data register pair (ADCL/ADCH) to get the 10-bit ADC result.

2.2.1. Test Steps

1. Build the project and load the hex file into the device.
2. Make the arrangements on STK600 as described in General Instructions to Test the Code on STK600 on page 8.
3. Adjust the voltage applied to PC0 and check whether the LED indication is updated depending on the voltage.
4. If the voltage on PC0 is higher than ~0.5V, the LED connected to PB0 will remain OFF, otherwise the LED will remain ON.

2.3. Free Running Mode and Conversion Complete Interrupt

Task: Free running conversion on ADC channel 0. Use of conversion complete interrupt.

In this program the `initialize()` routine is used to initialize the ADC module. As the ADC conversion complete interrupt is enabled, the Interrupt Service Routine for this interrupt will be triggered as soon as conversion is completed. In Free Running mode, a new conversion will be started immediately after a conversion completes. ADSC bit remains high during a conversion. The time between two consecutive ADC samples depends on the ADC conversion time.

1. Repeat the steps 1-4 from Single Conversion Mode on page 8.
2. Set the ADC Interrupt Enable bit (ADIE) in ADCSRA equal to 1 to enable the ADC conversion complete interrupt.
3. Set the Auto Trigger Enable bit (ADATE) in ADCSRA equal to 1 to enable auto triggered mode. By default, the Auto Trigger Source bit fields (ADTS2:0) in ADC Control and Status Register B (ADCSRB) is set to 000, which represents Free-running mode.
4. Set the Start Conversion bit (ADSC) in ADCSRA to start the first conversion.
5. After the conversion is over (ADIF bit becomes high) the CPU executes ADC interrupt service routine where the ADC data register pair (ADCL/ADCH) is read to get the 10-bit ADC result.

Test Steps

Refer to Test Steps on page 8.

2.4. Auto Triggering Using Timer0 Compare Event as Trigger Source

Task: Auto triggered conversion on ADC channel 0 by using Timer as trigger source.

In this program the `initialize_adc()` routine is used to initialize the ADC module. The `initialize_timer()` routine configures the Timer0 for compare A match event. The ADC result is read inside the corresponding ISR as soon as conversion is completed. The ADC module will start the conversion whenever the timer reaches its compare match value (that is, every 10 milliseconds in this example).

Note that by changing the 'OCR0A' compare register and timer0 clock prescaler bits in 'TCCR0B' register, the conversion interval (from 10 milliseconds) can be changed as per the application needs.

1. Repeat the steps 1-4 from Single Conversion Mode on page 8.
2. Set the ADC Interrupt Enable bit (ADIE) in ADCSRA to enable the ADC interrupt.
3. Set the Auto Trigger Enable bit (ADATE) in ADCSRA to enable auto triggered mode.
4. Set the Auto Trigger Source bit fields (ADTS2:0) in ADC Control and Status Register B (ADCSRB) to 011 to use the Timer0 Compare Match A event as ADC start trigger.
5. Set the Waveform Generation Mode bit fields (WGM1:0) in TCCR0A to '10' (Clear Timer on Compare Match).
6. Set the Timer0 Clock Select bit fields (CS2:0) in TCCR0B to '011' (ClkIO/64 > Prescaler).
7. Set the Output Compare register A to the desired value (in this example 156) so that ADC will convert the analog value at every 10 milliseconds (compare match event) interval.
8. After the conversion is over (ADIF bit becomes high) the CPU executes ADC interrupt service routine where the ADC data register pair (ADCL/ADCH) is read to get the 10-bit ADC result. An ADC conversion is triggered whenever the configured Timer0 compare match happens.

Test Steps

Refer to Test Steps on page 8.

2.5. Measurement of Bandgap Reference Voltage

Task: Measure the bandgap reference voltage.

The `initialize()` routine is used to initialize the ADC module. The `measure_gnd()` routine measures the ground value. The `measure_bandgap()` routine measures the internal band gap reference voltage.

1. Repeat the steps 1-3 from Single Conversion Mode.
2. Set the Voltage Reference bit fields (REFS1:0) in ADMUX equal to 01 to select AVCC as ADC voltage reference.
3. Set the ADC Interrupt Enable bit (ADIE) in ADCSRA equal to 1 to enable the ADC interrupt.
4. If the switch is pressed, configure the ADC to measure GND by setting MUX bit fields (MUX3:0) equal to 1111. This is done to discharge the capacitor of ADC.
5. While measuring the ground the ADC interrupt and auto trigger (free running) mode are disabled.
6. Set the Start Conversion bit (ADSC) in ADCSRA to start the conversion.
7. After the conversion is over (ADIF bit becomes high) read the ADC data register pair (ADCL/ADCH) to get the 10-bit ADC result value.
8. Configure MUX bit field (MUX3:0) equal to 1110 to select the bangap reference voltage as ADC input. It should be measured after 70µs delay, because of the start-up time for the Bandgap reference.
9. Polling method is used for checking conversion complete. Auto triggering mode is disabled.
10. Set the Start Conversion bit (ADSC) in ADCSRA to start the conversion.
11. After ADIF bit becomes high, read the ADC data register pair (ADCL/ADCH) to get the 10-bit ADC result.

Test Steps

1. Connect PB0 to one of the switches available on STK600.
2. Open the project in Atmel Studio 7. Press Alt+F5 to start debugging.
3. If debugWIRE is not already enabled, Atmel Studio will prompt to enable debugWIRE.
4. After it goes to debug mode, set a breakpoint at the end of the function `measure_bandgap`.
5. Run the code and press the switch connected to PB0.
6. At this point, execution will hit the break point set inside the `measure_bandgap` function.
7. Add the variable `bg_val` to watch window to see the ADC reading. The ADC result register can also be checked via I/O view (using the menu **Debug** > **Windows** >**I/O**).

2.6. Noise Reduction Sleep

Task: Use of ADC noise reduction sleep.

In this program the `initialize()` routine is used to initialize the ADC module. The example uses ADC noise reduction sleep mode. Enter the sleep mode by executing 'sleep' instruction. The ADC will start a conversion after the CPU has been halted. As the ADC conversion complete interrupt is enabled, the Interrupt Service Routine for this interrupt will be triggered as soon as conversion is completed. ADSC bit remains high during a conversion. When the execution of ISR is finished, the execution comes to the main routine and it executes 'sleep' again, which triggers another conversion.

1. Repeat the steps 1-4 from Single Conversion Mode.
2. Set the ADC Interrupt Enable bit (ADIE) in ADCSRA equal to 1 to enable the ADC conversion complete interrupt.
3. Set the Auto Trigger Enable bit (ADATE) in ADCSRA equal to 1 to enable auto triggered mode. By default, the Auto Trigger Source bit fields (ADTS2:0) in ADC Control and Status Register B (ADCSRB) is set to 000, which represents Free-running mode.
4. Set the Start Conversion bit (ADSC) in ADCSRA to start the first conversion.
5. After the conversion is over (ADIF bit becomes high) the CPU executes ADC interrupt service routine where the ADC data register pair (ADCL/ADCH) is read to get the 10-bit ADC result.

Test Steps

Refer to Test Steps.

3. Device Datasheet Reference

This application note describes the Atmel ATmega88's ADC module. The operation of on chip ADC on different Atmel megaAVR devices are almost similar. Refer the respective device datasheet for more details. For example some Atmel megaAVR device variants (such as Atmel ATmega48P) have on-chip temperature sensor as one of the ADC input, some devices have differential ADC channels with configurable gains and 2.56 internal voltage reference (such as Atmel ATmega2560), some devices support high ADC speed such as 125ksps (like Atmel AT90PWM1). Refer to datasheet section *Analog to Digital Converter* for detailed information on the ADC of the specific device.

Datasheet section '*Electrical Characteristics >ADC Characteristics*, *Electrical Characteristics > System and Reset Characteristics* contains ADC related characterization data such as INL, DNL, absolute accuracy, conversion time, offset error, gain error, input resistance, band gap reference start up time, etc.

Datasheet section *Errata* contains information about any known ADC related errata for a device. If there are any errata which has a workaround mentioned, ensure that the workaround has been implemented.

4. Driver Implementation

This application note includes a source code package with a basic ADC driver implemented in C. It is developed with Atmel Studio 7 for Atmel ATmega88 device. 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.

Following are the features covered by the examples in this application note:

1. Single conversion mode.
2. Free running mode and conversion complete interrupt.
3. Auto triggering using Timer0 compare event as trigger source.
4. Measurement of bandgap reference voltage.
5. Noise Reduction Sleep.

5. Recommended Reading

- **AVR042:** AVR Hardware Design Considerations - This application note provides answers to some of the questions and problems faced when starting designs involving Atmel AVR microcontrollers.
- **AVR120:** Characterization and Calibration of the ADC on an AVR - This application note explains various ADC (Analog to Digital Converter) characterization parameters, how they affect ADC measurements and how to measure them and how to perform run-time compensation.
- **AVR121:** Enhancing ADC resolution by over sampling - This Application Note explains the method called 'Oversampling and Decimation' and which conditions need to be fulfilled to make this method work properly to achieve a higher resolution without using an external ADC.
- **AVR122:** Calibration of the tinyAVR internal temperature reference - This application note describes how to calibrate and compensate the temperature measurements from the ATtiny25/45/85. It can also be used on other AVR microcontrollers with internal temperature sensors.
- **AVR125:** ADC of tinyAVR in single ended mode - This application note describes the basic functionality of the ADC in Atmel tinyAVR devices in Single ended mode with code examples on Atmel ATtiny88 to get started. The code examples are written in assembly language and C language.
- **AVR127:** Understanding ADC parameters - This application note discusses about the basic concepts of analog-to-digital converter (ADC) and the various parameters that determine the performance of an ADC. These ADC parameters are of good importance since they are a part of deciding the accuracy of the ADC's output.

6. Revision History

Doc Rev.	Date	Comments
8444B	03/2016	Added new example.
8444A	10/2011	Initial document release.

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**