AVR125: ADC of tinyAVR in Single Ended Mode

8-bit AVR® Microcontrollers

Application Note

Features

- **Up to 10bit resolution**
- **Up to 15kSPS**
- **Auto triggered and single conversion mode**
- **Optional left adjustment for ADC result readout**
- **Driver source code included for ATtiny88**
  - **ATtiny88 ADC in Single Conversion Mode**
  - **ATtiny88 ADC in free running mode**
  - **ATtiny88 ADC for temperature measurement**
  - **ATtiny88 ADC for bandgap measurement**

1 Introduction

ATtiny 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 two internal channel from internal temperature sensor and 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® 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.

Rev. 8352A-AVR-01/12

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 ADC operation

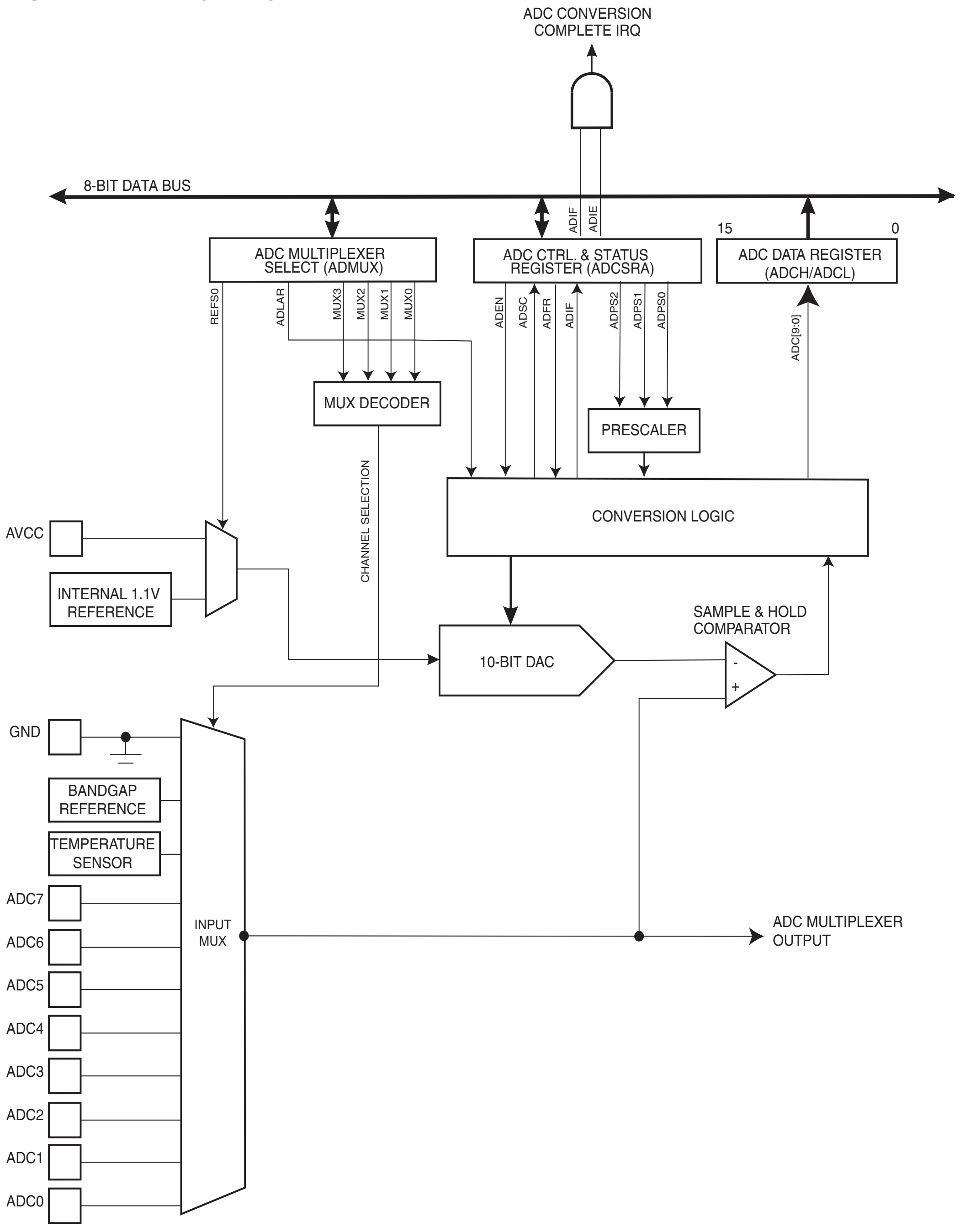
To make use of the ADC, the PRADC bit in the Power Reduction Register must be disabled. This is done by clearing the PRADC bit. ADC module must be disabled before disabling in PRR. The ADC module converts the analog input voltage to a 10-bit digital value. The minimum value represents GND and maximum value denotes the reference voltage used. The Reference voltage is chosen by the REFS0 bit in the ADMUX register. The possible reference voltages are internal 1.1V and AVcc.

The analog input channel for conversion is selected by the bits in the ADMUX register. This includes the ADC input pins along with internal voltage from temperature sensor, GND and fixed bandgap reference voltage. To enable the ADC, ADEN bit in the ADCSRA register must be set. 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 10-bit digital value after conversion is stored in ADCH and ADCL. 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 necessary. If ADLAR is enabled and the application needs only 8 bit accuracy then ADCH alone can be read. 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.

ADC module owns one interrupt which is triggered once a conversion is complete. If an interrupt occurs between reading ADCL and ADCH, it will get triggered and the result will be lost.

Figure 2-1. Analog to digital converter block schematic operation.



2.2 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

2.2.1 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 then the reference voltage selected for ADC.

2.2.2 Internal inputs

Two internal analog signals can be selected as input and measured by the ADC.

- Temperature sensor
- Bandgap voltage

The voltage output from an internal temperature reference can be measured with the ADC and the voltage output will give an ADC result presenting the current temperature in the microcontroller.

The bandgap voltage is an accurate voltage reference inside the microcontroller that is the source for other internal voltage reference.

2.3 Starting a conversion

In single conversion mode, for starting a conversion, the ADSC bit in the ADCSRA register must be written a logical one. This bit remains at high logic till the conversion is complete 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, ADTS in ADCSRB.

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. The ADC then operates in Free Running mode, in which next conversion is triggered once the previous conversion completes and sets the ADIF. Note that ADIF must be cleared manually in order to trigger at next interrupt event in auto triggered mode. For free running mode, the ADC will perform successive conversions independent of whether ADIF is cleared or not. The first conversion must be started by setting ADSC bit.

2.4 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. It is not recommended to use ADC clock with a frequency higher than 1MHz. If ADC resolution of less than 10 bits required, then the ADC clock frequency can be higher than 200kHz. The prescalar value is selected with ADPS bits in ADCSRA. At 1MHz we can expect maximum 8 bits of resolution.

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.

A normal conversion takes 13 ADC clock cycles. 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.

2.5 Changing channel or reference selection

Bits MUXn and REFS0 in the ADMUX register are single buffered through a temporary register to which CPU has random access. If Auto Triggering is used, then ADMUX can be safely updated:

- 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 till the conversion completes and then change the channel.

In Free running mode, select the channel before starting the first conversion. It can be changed one clock cycle after setting ADSC bit. It is better to wait till first conversion is complete and then change the channel selection. But since the next conversion has already started automatically, the changes will be reflected in the next following conversion.

2.6 ADC noise canceller

The Atmel ATtiny ADC has a noise canceller that enables conversion during sleep mode which reduces the noise induced from CPU core and other peripherals. This feature is available in ADC Noise Reduction and Idle mode.

- 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

2.7 Conversion result

Once the ADC completes conversion, the 10 bit result will be available in the ADCH and ADCL registers and ADC Interrupt Flag will be set.

For single conversion, the result is:

$$\text{ADC} = \frac{V_{IN}.1024}{V_{REF}}$$

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.

2.8 Temperature measurement

The temperature measurement is done by an on chip temperature sensor. It is internally coupled to ADC input channel 8. Internal 1.1V reference must be selected for temperature measurement and ADC8 must be selected with MUX3:0 bits in ADMUX register as "1000".

The temperature sensor has an approximate sensitivity of 1 LSB/˚C and the accuracy depends on the method of user calibration. Using single temperature calibration, the measurement accuracy is ±10˚C, assuming calibration at room temperature. Better accuracies are achieved by using two temperature points for calibration.

Table 2-1. Temperature vs. sensor output volyage (typical case).

Temperature	**-40°C**	**+25°C**	**+85°C**
ADC	230 LSB	300 LSB	370 LSB

The values depicted are typical values. More accurate results can be achieved with software calibration using the formula:

$$T = k * [(ADCH << 8) \mid ADCL] + T_{OS}$$

where ADCH and ADCL are the ADC result registers, T_{OS} is the temperature sensor offset and k is the fixed slop coefficient. K must be evaluated based on measurements at two temperatures for higher accuracy. Typically, k is very close to 1.0 and in single-point calibration the coefficient may be omitted.

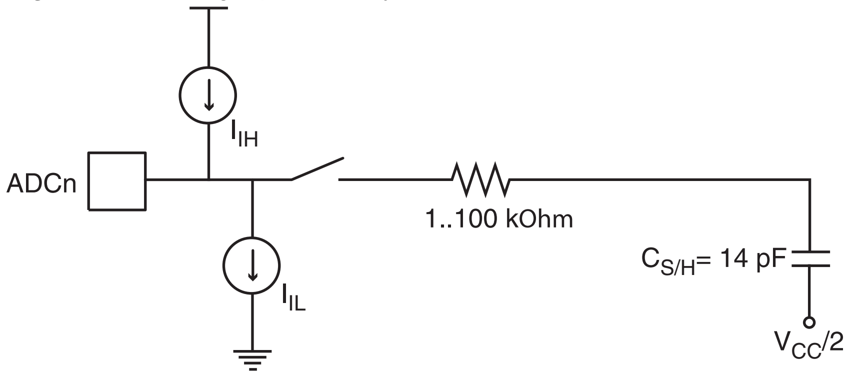
2.9 Analog input circuitry

The analog input circuitry for single ended channels is depicted in Figure 2-2. Regardless of whether a channel is selected as input for ADC, it is subjected to pin capacitance and input leakage of that pin. When particular channel is selected, it should drive the S/H capacitor through the series resistance which is the combined resistance in the input path.

The ADC module for Atmel ATtiny88 is optimized for analog signals with an output impedance of 10kΩ or less. It is important to make sure that the source impedance is either 10KΩ or less because sampling time will be negligible for such a source.

If the source impedance is higher than 10KΩ then the time taken to charge the capacitor will increase and the results will not be accurate. For example if you are using a voltage divider at the ADC input using a resistor network make sure that the source impedance is less then 10kΩ. Low impedance must be used for slowly varying signals since this minimizes the time for charge transfer. Frequency components higher than Nyquist frequency ($f_{ADC}/2$) must be removed with a low pass filter, in order to avoid distortion from unpredictable signal convolution. Please refer to device datasheet for impedance value.

Figure 2-2. Analog input circuitry.



2.10 Best practices for improving accuracy

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

- Understand the ADC, its features and how they are intended to be used
- Understand the application requirements
- Ensure that the source impedance is not too high compared to the sampling rate that is used. 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
- It is important to take great care when designing the analog signal paths like analog reference (V_{REF}) and analog power supply (A_{VCC}). Filtering should be used if the analog power supply is connected to digital power supply
- Keep analog signal paths as short as possible
- Make sure analog tracks run over the analog ground plane
- Avoid having the analog signal path close to digital signal path with high switching noise (that is communication lines, clock signals)
- Consider decoupling of the analog signal. Decoupling between signal and ground for single-ended inputs
- Try to toggle as few pins as possible while the ADC is converting, to avoid switching noise internally and on the power supply. The ADC is most sensitive to switching the I/O pins that are powered by the analog power supply (PORTC)
- Disable digital input on the corresponding ADC channel to minimize the power consumption
- Switch off the unused peripherals by setting PRR registers to eliminate noise from unused peripherals
- Put the device in “ADC Noise Reduction” mode to get more accurate results with ADC
- Wait until the ADC, reference or sources are stabilized before sampling, as some sources (for example, 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

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.

3.1 Single conversion

Task: One single-ended conversion of ADC input 1

- Set the *Mux* bitfield (MUX3:0) in ADC's MUX register (ADMUX) equal to 0001 to select ADC Channel 1
- Set the *Voltage Reference* bitfield (REFS0) in ADMUX equal to 0 to select Internal 1.1V reference
- Set the *ADC Enable* bit (ADEN) in ADC Control and Status Register A (ADCSRA) to enable the ADC module
- Set the *ADC Prescaler* bitfield (ADPS2:0) in ADCSRA equal to 100 to prescale system clock by 16
- Set the *Start Conversion* bit (ADSC) in ADCSRA to start a single conversion
- Wait for the *Interrupt Flag* bit (ADIF) in ADCSRA to be set, indicating that the conversion is finished
- Read the *Result* register pair for (ADCL/ADCH) to get the 10-bit conversion result as a 2-byte value

3.2 Free-running mode

Task: Free running conversion on ADC channel 1

- Set the Mux bitfield (MUX3:0) in ADC's MUX register (ADMUX) equal to 0001 to select ADC Channel 1
- Set the Voltage Reference bitfield (REFS0) in ADMUX equal to 0 to select Internal 1.1V reference
- Set the ADC Enable bit (ADEN) in ADC Control and Status Register A (ADCSRA) to enable the ADC module
- Set the ADC Prescaler bitfield (ADPS2:0) in ADCSRA equal to 100 to prescale system clock by 16
- Set the Auto Trigger Enable bit (ADATE) in ADCSRA equal to 1 to enable auto triggered mode
- By default, the Auto Trigger Source bitfield (ADTS2:0) in ADC Control and Status Register B (ADCSRB) is set to 000, which represents Free-running mode
- Set the Start Conversion bit (ADSC) in ADCSRA to start the first conversion
- Optionally wait for the Interrupt Flag bit in the ADCSRA register to be set, indicating that a new conversion is finished
- Read the Result register pair for (ADCL/ADCH) to get the 10-bit conversion result 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.

3.3 Temperature measurement

Task: Measure temperature from Internal Temperature Sensor (ADC8)

This can be done either in Single conversion mode or Free Running mode if continuous temperature monitoring is required.

The steps to be followed are same as mentioned above with ADC input channel selected as ADC8. That is, set the *Mux* bitfield (MUX3:0) in ADC's MUX register (ADMUX) equal to 1000 to select ADC Channel 8 and repeat the rest of the steps.

3.4 Bandgap measurement

Task: Measure bandgap reference from Internal bandgap reference channel

- Initialize ADC module as mentioned above
- Set ADC reference to AVcc
- Keep measuring input on ADC Channel 0, until the switch is pressed
- If the switch is pressed, configure 0V (GND) in ADC by setting *Mux* bitfield (MUX3:0) equal to 1111. This is done to discharge the capacitor of ADC
- After 70µS, measure the bandgap reference value by configuring *Mux* bitfield (MUX3:0) equal to 1110. It should be measured after 70µS delay, because of the time taken to charge the capacitor of ADC
- Read the *Result* register (ADC) to get the conversion result

4 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 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.

1. ATtiny88 ADC in Single Conversion Mode
2. ATtiny88 ADC in free running mode
3. ATtiny88 ADC for temperature measurement
4. ATtiny88 ADC for bandgap measurement

5 Recommended reading

It is recommended to read the following Application Notes to get an overall idea on ADC:

- AVR042: AVR Hardware Design Considerations – This application note covers most of the problems encountered with power supply design and other physical design problems
- AVR121: Enhancing ADC resolution by oversampling - 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's internal temperature reference - This application note describes how to calibrate and compensate the temperature measurements from the Atmel ATtiny25/45/85. It can also be used on other Atmel AVR® microcontrollers with internal temperature sensors

6 Table of Contents