AVR 8-bit Microcontroller

AVR125: ADC of tinyAVR in Single Ended Mode

APPLICATION NOTE

Introduction

Atmel® 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. 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. It contains code examples for Atmel ATtiny104 and ATtiny88 devices to get started. The code examples are written using C language in Atmel Studio 7.

Note: For more details about features and functionality of ADC modules, refer the specific device datasheet.

Features

- Up to 10-bit resolution
- Up to 15kSPS
- Auto triggered and single conversion mode
- Optional left adjustment for ADC result readout
- Driver source code included for
  - ATtiny88 ADC in single conversion mode
  - ATtiny88 ADC in free running mode
  - ATtiny88 ADC for temperature measurement
  - ATtiny88 ADC for bandgap measurement
  - ATtiny104 ADC in single conversion mode
  - ATtiny104 ADC in free running mode

Table of Contents

1. ADC Functionality and Basic Configuration

The section ADC Operation provides an overview of the functionality and basic configuration options of the ADC. The basic steps to get started with the ADC is provided along with the register descriptions and configuration details in the following sections.

1.1. ADC Operation

To use the ADC, the PRADC bit in the Power Reduction Register must be disabled 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.

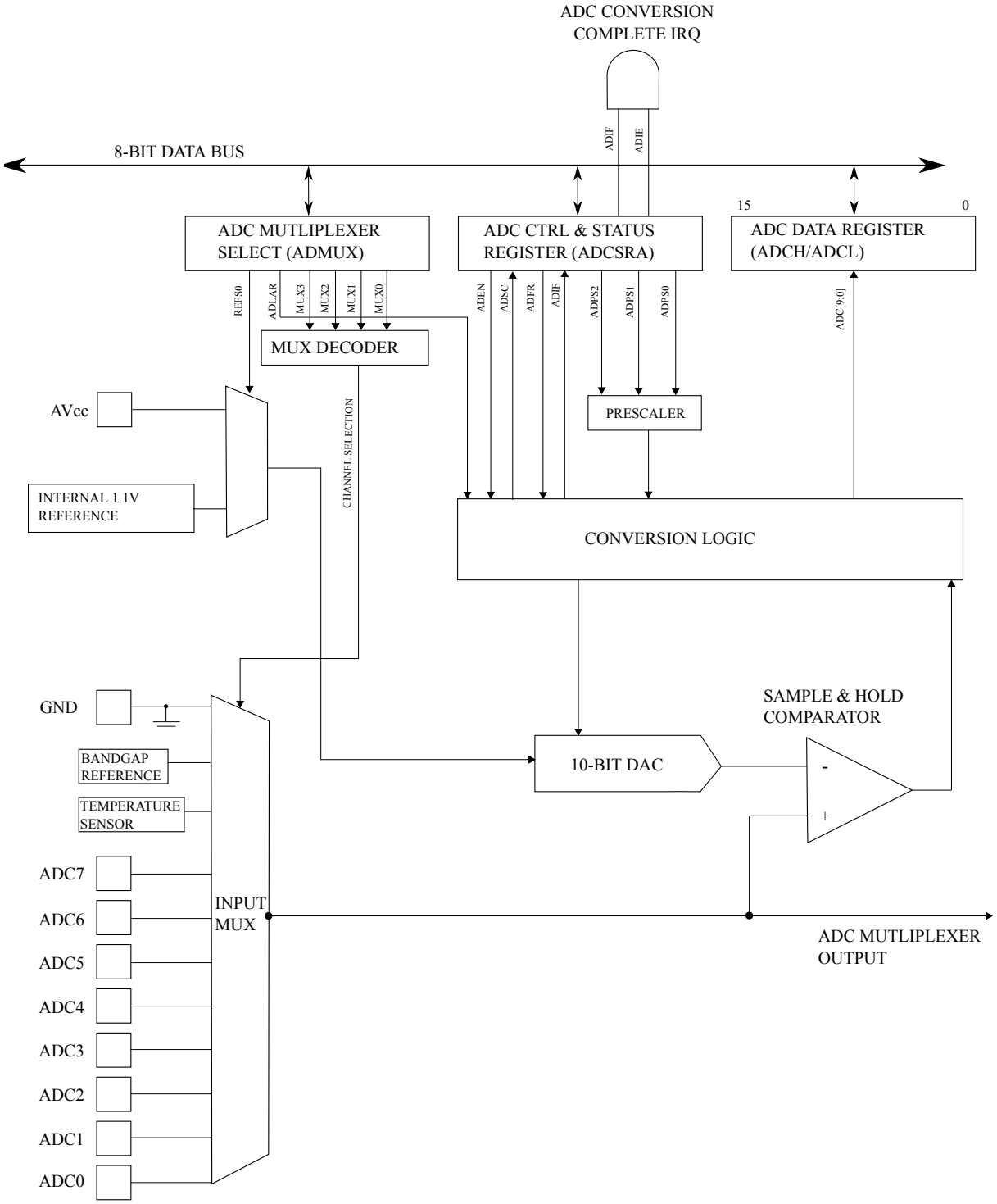
Note: For details about the available levels of reference voltages, refer the specific device datasheet.

The analog input channel for conversion is configured by selecting appropriate bits in the ADMUX register. To enable the ADC, ADEN bit in the ADCSRA register must be set. The channels selected for conversion will not be effect until ADEN bit is set. Before entering Sleep mode, the ADC module can be disabled by clearing ADEN bit. This reduces the power consumption due to 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 performed by setting the ADLAR bit in the ADMUX register, if necessary. If ADLAR is enabled and the application requires 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, after ADCL is read. It is re-enabled only after ADCH is read.

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

Figure 1-1. Analog to Digital Converter Block Diagram of ATtiny88



Note: A few modules in the above block diagram may not be available in ADC of ATtiny104. Refer the device datasheet for ADC architecture of ATtiny104.

1.2. Input Source

The input sources for the ADC are the analog voltage inputs that the ADC canmeasure and convert. Two types of measurements can be selected:

- Single ended input
- Internal input

Note: Internal Input is not applicable for ATtiny104 devices.

1.2.1. 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.

1.2.2. Internal inputs

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

- Temperature sensor
- Bandgap voltage

The temperature in MCU can be measured by selecting the temperature sensor output as ADC input. ADC result will give 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.

1.3. Starting a Conversion

In single conversion mode, for starting a conversion, the ADSC bit in the ADCSRA register must be set to 1 (high). 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 using 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: In auto triggered mode, the ADIF must be cleared manually for the next event to generate an interrupt. 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.

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. It is not recommended to use ADC clock with a frequency higher than 1MHz. 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. The timing for ADC clock and converstion varies slightly for ATtiny88 and ATtiny104.

ATtiny88:

1. If ADC resolution less than 10-bit is required, then the ADC clock frequency can be higher than 200kHz.
2. 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.

ATtiny104:

1. If ADC resolution less than 10-bit required, then the ADC clock frequency can be higher than 100kHz.
2. A normal conversion takes 15 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 26 ADC clock cycles in order to initialize the analog circuitry.

1.5. 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 trigger mode 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

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 preferable 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.

1.6. 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.

- Ensure that the ADC is enabled and is not busy performing any conversion. 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 after 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

When 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:

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

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

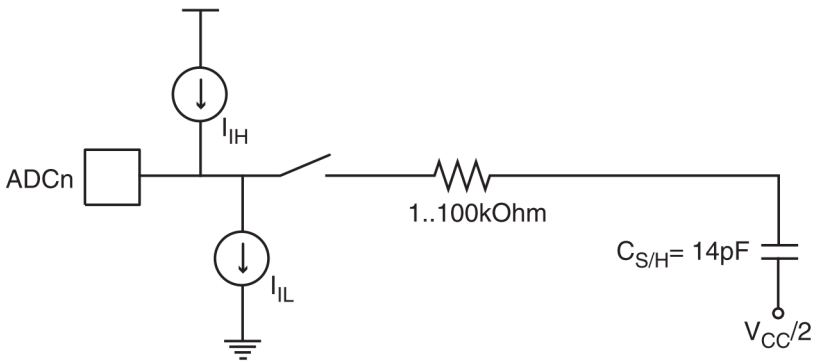
1.8. Analog Input Circuitry

The analog input circuitry for single ended channels is depicted irrespective 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 ATtiny104/102 is optimized for analog signals with an output impedance of 10kΩ or less. It is important to ensure 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. Refer the device datasheet for impedance value.

Figure 1-2. Analog Input Circuitry



1.9. 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 improving the accuracy of 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 of the the analog signal paths like analog reference (VREF) and analog power supply (AVCC) while designing. Use filtering if the analog power supply is connected to digital power supply.

- Keep analog signal paths as short as possible.
- Ensure that the 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 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.
- Use the device in “ADC Noise Reduction” mode for 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.

2. Getting started - ATtiny88

This section provides the basic steps for simple conversion and experimenting with MUX settings for ATtiny88 device. The necessary registers are described along with relevant bit settings.
Note: Only manual polling of status bits is covered here.

2.1. Single conversion

Task: One single-ended conversion of ADC input 1

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

2.2. Free-running mode

Task: Free running conversion on ADC channel 1

1. Set the Mux bitfield (MUX3:0) in ADC's MUX register (ADMUX) equal to 0001 to select ADC Channel 1.
2. Set the Voltage Reference bitfield (REFS0) in ADMUX equal to 0 to select Internal 1.1V reference.
3. Set the ADC Enable bit (ADEN) in ADC Control and Status Register A (ADCSRA) to enable the ADC module.
4. Set the ADC Prescaler bitfield (ADPS2:0) in ADCSRA equal to 100 to prescale system clock by 16.
5. Set the Auto Trigger Enable bit (ADATE) in ADCSRA equal to 1 to enable auto triggered mode.
6. 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.
7. Set the Start Conversion bit (ADSC) in ADCSRA to start the first conversion.
8. Optionally wait for the Interrupt Flag bit in the ADCSRA register to be set, indicating that a new conversion is finished.
9. Read the Result register pair for (ADCL/ADCH) to get the 10-bit conversion result as a 2-byte value.

Note: It is not necessary to wait for the interrupt flag when using free-running mode. However, to make sure of a fresh conversion, wait for the flag to set, clear it and then read the result.

2.3. 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. 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.

2.4. Bandgap measurement

Task: Measure bandgap reference from Internal bandgap reference channel

1. Initialize ADC module as mentioned in Temperature measurement.
2. Set ADC reference to AVcc.
3. Keep measuring input on ADC Channel 0, until the switch is pressed.
4. 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.
5. 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.
6. Read the *Result* register (ADC) to get the conversion result.

3. Getting Started - ATtiny104

This section provides the basic steps for getting up and running with simple conversion and experimenting with MUX settings for ATtiny104 devices. The necessary registers are described along with relevant bit settings. This section only covers manual polling of status bits. The applications are tested using ATtiny104 Xplained Nano kit.

Note: For more details about working with ATtiny104 Xplained Nano kit, refer the Hardware User Guide.

3.1. Single Conversion mode

1. Set the Mux bit field (MUX3:0) in ADC's MUX register (ADMUX) equal to 0001 to select ADC Channel 1.
2. Set the Voltage Reference bit field (REFS1) in ADMUX equal to 0 to select Internal 2.2V reference.
3. Set the ADC Enable bit (ADEN) in ADC Control and Status Register A (ADCSRA) to enable the ADC module.
4. Set the ADC Prescaler bit field (ADPS2:0) in ADCSRA equal to 100 to prescale system clock by 16.
5. Set the Start Conversion bit (ADSC) in ADCSRA to start a single conversion.
6. Poll for the Interrupt Flag bit (ADIF) in ADCSRA to be set, indicating that the conversion is finished.
7. Read the Result register pair for (ADCL/ADCH) to get the 10-bit conversion result as a 2-byte value.
8. In this application note the ADC results are displayed in the terminal window via UART module. (UART Baud rate is 2400).

3.2. Free-Running Mode

1. Set the Mux bit field (MUX3:0) in ADC's MUX register (ADMUX) equal to 0001 to select ADC Channel 1.
2. Set the Voltage Reference bit field (REFS1) in ADMUX equal to 0 to select Internal 2.2V reference.
3. Set the ADC Enable bit (ADEN) in ADC Control and Status Register A (ADCSRA) to enable the ADC module.
4. Set the ADC Prescaler bit field (ADPS2:0) in ADCSRA equal to 100 to prescale system clock by 16.
5. Set the Auto Trigger Enable bit (ADATE) in ADCSRA equal to 1 to enable auto triggered mode.
6. Set the ADC Interrupt Enable bit (ADIE) in ADCSRA equal to 1 to enable the interrupt.
7. By default, the Auto Trigger Source bit field (ADTS2:0) in ADC Control and Status Register B (ADCSRB) is set to 000, which represents Free-running mode.
8. Set the Start Conversion bit (ADSC) in ADCSRA to start the first conversion.
9. Optionally wait for the Interrupt Flag bit in the ADCSRA register to be set, indicating that a new conversion is finished.
10. Read the Result register pair for (ADCL/ADCH) to get the 10-bit conversion result as a 2-byte value.
11. In this application note the ADC results are displayed in the terminal window via UART module (UART Baud rate is 2400).

4. Driver Implementation

This application note includes a source code package with a basic ADC driver necessary to get started. It is tested for Atmel Studio 7.

Note: The driver supplied with the application note is to get started with the ADC and may not be fully optimized for code size.

1. ATtiny104 ADC in single conversion mode
2. ATtiny104 ADC in free running mode
3. ATtiny88 ADC in single conversion mode
4. ATtiny88 ADC in free running mode
5. ATtiny88 ADC for temperature measurement
6. ATtiny88 ADC for bandgap measurement

5. References

- 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 explains the conditions that must be fulfilled to make this method work as expected to achieve a higher resolution without using an external ADC
- ATTINY104-XNANO - The Atmel ATTINY104-XNANO Xplained Nano evaluation kit is a hardware platform for evaluating ATtiny102/ATtiny104 microcontrollers.

6. Revision History

Doc. Rev.	Date	Comments
8352B	06/2016	Updated to include ATtiny104 device.
8352A	01/2012	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**