Department of Computer Engineering
Faculty of Engineering, University of Peradeniya

CO321 Embedded Systems - 2020

Lab 5 - ADC

AVR Analog-to-Digital Converter Programming in C Language

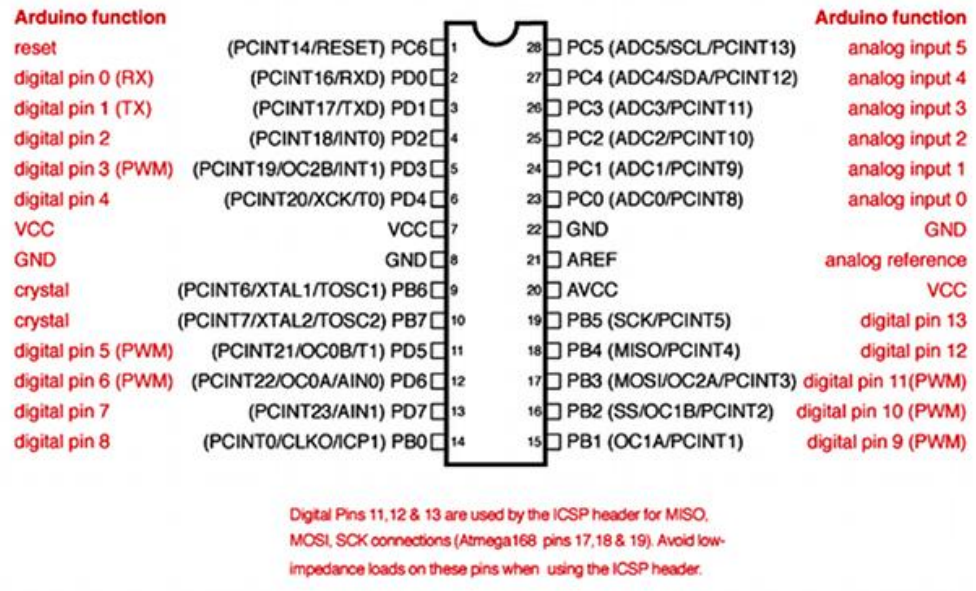


Figure 01: ATmega328P pin Mapping

Introduction of ADC devices

Analog-to-digital converters are among the most widely used devices for data acquisition. Digital computers use binary (discrete) values, but in the physical world, everything is analog (continuous). Temperature, pressure (wind or liquid), humidity, and velocity are a few examples of physical quantities that we deal with every day. A physical quantity is converted to electrical (voltage, current) signals using a device called as a transducer. Transducers are also referred to as sensors. Sensors for temperature, velocity, pressure, light and many other natural quantities produce an output that is an analog voltage (or current). Therefore, we need an analog-to-digital converter to translate the analog signals to digital numbers, so that the microcontroller can read and process them.

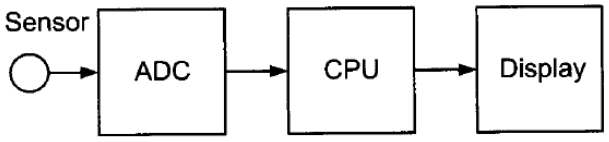


Figure 02: Microcontroller Connection to Sensor via ADC

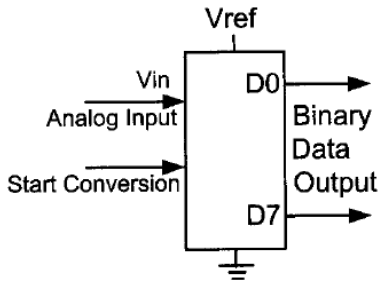


Figure 03: An 8-bit ADC Block diagram

Resolution

The ADC has n-bit resolution, where n can be 8,10,12,16, or even 24 bits. Higher-resolution ADCs provide a smaller step size, where step size is the smallest change that can be discerned by an ADC. Although the resolution of an ADC chip is decided at the time of its design and cannot be changed, we can control the step size with the help of what is called V_{ref}.

V_{ref} is an input voltage used for the reference voltage. The voltage connected to this pin, along with the resolution of the ADC chip, dictate the step size. For an 8-bit ADC, the step size is V_{ref} /256.because it is an 8-bit ADC, and 2 to the power of 8 gives us 256 steps.

Conversion time

In addition to resolution, conversion time is another major factor in judging an ADC. Conversion time is defined as the time it takes the ADC to convert the analog input to a digital (binary) number. The conversion time is dictated by the clock source connected to the ADC in addition to the method used for data conversion and technology used in the fabrication of the ADC chip such as MOS or TTL technology.

Digital data output

In an 8-bit ADC we have an 8-bit digital data output of D0-D7, while in the 10-bit ADC the data output is D0-D9. To calculate the output voltage,

$$Dout = \frac{Vin}{stepsize}$$

Dout = digital data output(in decimal)
V_{in}= analog input voltage
step size = $V_{ref}/2^n$ for an ADC with n-bit resolution.

An on-chip ADC eliminates the need for an external ADC connection, which leaves more pins for other I/O activities.

Analog Input Channels

Many data acquisition applications need more than one ADC. For this reason, we see chips with 2,4,8, or even 16 channels on a single chip. Multiplexing of analog inputs is widely used in the many chips allowing us to monitor multiple quantities such as temperature, pressure, heat and so on.

Start conversion and end-of-conversion signals

The fact that we have multiple analog input channels and a single digital output register creates the need for start conversion (SC) and end-of conversion (EOC) signals. When SC is activated, the ADC starts converting the analog input value of V_{in} to an n-bit digital number. The amount of time it takes to convert varies depending on the conversion method. When the data conversion is complete, the end of conversion signal notifies the CPU that the converted data is ready to be picked up.

ATmega328P ADC features

The ADC unit of the ATmega328P has the following characteristics:

(a) It is a 10-bit successive approximation ADC
(b) It has 8 analog input channels,
(c) The converted output binary data is held by 2 special function registers called ADCL (A/D result low) and ADCH (A/D result high)
(d) Because the ADCH; ADCL registers give us 16 bits and the ADC data out is only 10 bits wide, 6 bits of the 16 are unused. We have the option of making either the upper 6 bits or the lower 6 bits unused.
(e) We have 3 options for voltage reference (V_{ref}), it can be connected to AVCC (Analog V_{cc}), internal 1.1V references, or external AREF pin.

For digital logic signals, a small variation in the voltage level has no effect on the output. For example, 0.2 V may be considered LOW since, in TTL logic, anything less than 0.5 V will be directed as LOW logic. That is not the case when we are dealing with analog voltage. Figure 4 shows some techniques that must be followed and they are elaborated below.

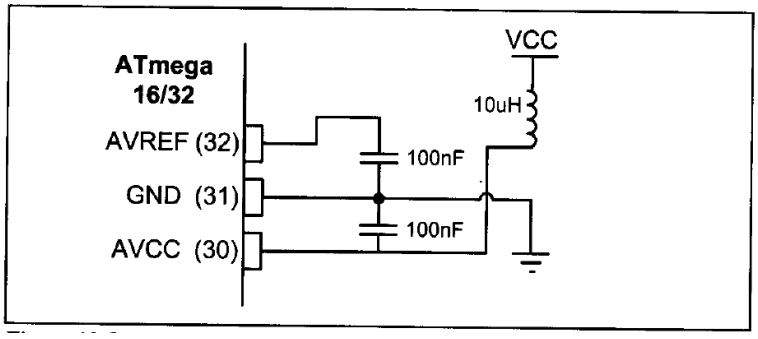


Figure 04: ADC recommended Connection.

Decoupling AVCC from V_{cc}: The AVCC pin provides the supply for analog ADC circuitry. To get a better accuracy of AVR ADC, we must provide a stable voltage source to the AVCC pin.

Connecting a capacitor between V_{ref} and GND: By connecting a capacitor between the AVREF pin and GND, you can make the V_{ref} voltage more stable and increase the precision of ADC.

Registers associated with ADC on ATmega328P

In the AVR microcontroller, 5 major registers are associated with the ADC. They are ADMUX (ADC multiplexer selection register), ADCSRA (ADC control and status register A), ADCH (high data), ADCL (low data), ADCSRB (ADC control and status register B) and DDR0 (Digital Input Disable Register 0). We will focus on the most important registers.

ADMUX register

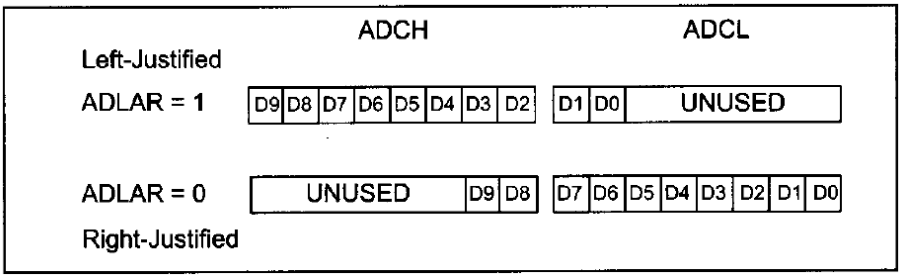
7	6	5	4	3	2	1	0	
REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

REFS1:0 - Bit 7:6 - Reference selection bits
These bits select the reference voltage for the ADC.

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

If you connect the AREF pin to an external fixed voltage you will not be able to use the other reference voltage options in the application, as they will be short circuited with the external voltage.

ADLAR - Bit 5 - ADC Left Adjust Results
This bit dictates whether the left-most 10-bits or the right-most 10-bits of the result registers pair ADCH: ADCL are used to store the result. If we write a '1' to ADLAR, the result will be left-justified; otherwise, the result is right-justified.



MUX[3:0] - Bits 3:0 – Analog Channel Selection Bits

The value of these bits selects which analog input channel (whether ADC0, ADC1, ADC3, ..., ADC7) is connected to the ADC.

ADCSRA register

The ADCSRA register is the status and control register of ADC. Bits of this register control or monitor the operation of the ADC.

7	6	5	4	3	2	1	0	
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

ADEN Bit 7 ADC Enable**:** This bit enables the ADC. Setting this bit to '1' will enable the ADC, and clearing this bit to '0' will disable it even while a conversion is in progress.

ADSC Bit 6 ADC Start Conversion**:** To start each conversion you have to set this bit to '1'.

ADATE Bit 5 ADC AutoTrigger Enable**:** Auto triggerring of the ADC is enabled when you set this bit to one.

ADIF Bit 4 ADC interrrupt flag**:** This flag bit is set when an ADC Conversion completes and the data registers are updated.

ADIE Bit 3 ADC Interrupt enable**:** Setting this bit to '1' enables the ADC conversion complete interrrupt.

ADPS2:0 Bits 2:0 ADC Prescaler Select Bits**:** These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

By using the ADPS2:0 bits of the ADCSRA register we can control the A/D conversion time. To select the conversion time, we can select any of $F_{osc/2}$, $F_{osc/4}$, $F_{osc/8}$, $F_{osc/16}$, $F_{osc/32}$, $F_{osc/64}$ or $F_{osc/128}$ for ADC clock, where F_{osc} is the frequency of the crystal oscillator connected to the AVR chip. For the AVR, the ADC requires an input clock frequency *less than 200 kHz* for the maximum accuracy.

A timing factor that we should know about is the **acquisition time**. After an ADC channel is selected, the ADC allows some time for the sample-and-hold capacitor to charge fully to the input voltage level present at the channel. In the AVR, the first conversion takes 25 ADC clock cycles in order to initialize the analog circuitry and pass the sample and hold time. Each consecutive conversion takes 13 ADC clock cycles.

ADCH and ADCL registers

After the A/D conversion is complete, the result sits in registers ADCL and ADCH. The ADLAR bit of the ADMUX is used for making it right-justified or left-justified because we need only 10 of the 16 bits.

ADC Programming in AVR

Steps in programming the A/D converter using polling

To program the A/D converter of the AVR, the following steps can be taken:

(1) Make the pin for the selected ADC channel an input pin.
(2) Turn on the ADC module of the AVR, because it is disabled upon power-on reset to save power.
(3) Select the conversion speed. We use registers ADPS2:0 to select the conversion speed.
(4) Select voltage reference and ADC input channels. We use the REFS0 and REFS1 bits in the ADMUX register to select voltage reference and the MUX4:0 bits in ADMUX to select the ADC input channel.
(5) Activate the start conversion bit by writing a one to the ADSC bit of ADCSRA.
(6) Wait for the conversion to be completed by polling the ADIF bit in the ADCSRA register.
(7) After the ADIF bit has gone HIGH, read the ADCL and ADCH; otherwise, the result will not be valid.
(8) If you want to read the selected channel again, go back to step 5.
(9) If you want to select another V_{ref} source or input channel, go back to step 4.

Exercise 1:

Build a digital voltmeter (one that always measures a voltage with respect to ground rather than a differential voltage) that measures voltages of the range 0V-5V with a 10-bit resolution. Use ADC1 pin of the microcontroller for the input and LEDs connected to Port D as the output. Use AVCC as the voltage reference. Demonstrate that it works by connecting to ground, 5V and 3.3V pins on the Arduino board.

Hint: Use left justification of the ADC result for your own convenience.

Exercise 2:

Using an LDR, LED and an ATmega328P microcontroller, design the model of a light bulb that automatically lights when it is dark or else turn off. Use external AREF as the voltage reference.