AVR Microcontroller

Microprocessor Course

Chapter 13

ADC, DAC, AND SENSOR INTERFACING

Azar 1393

ADC, DAC AND SENSOR INTERFACING
13.1 ADC CHARACTERISTICS

ADC devices

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 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 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. See Figures 13-1 and 13-2.

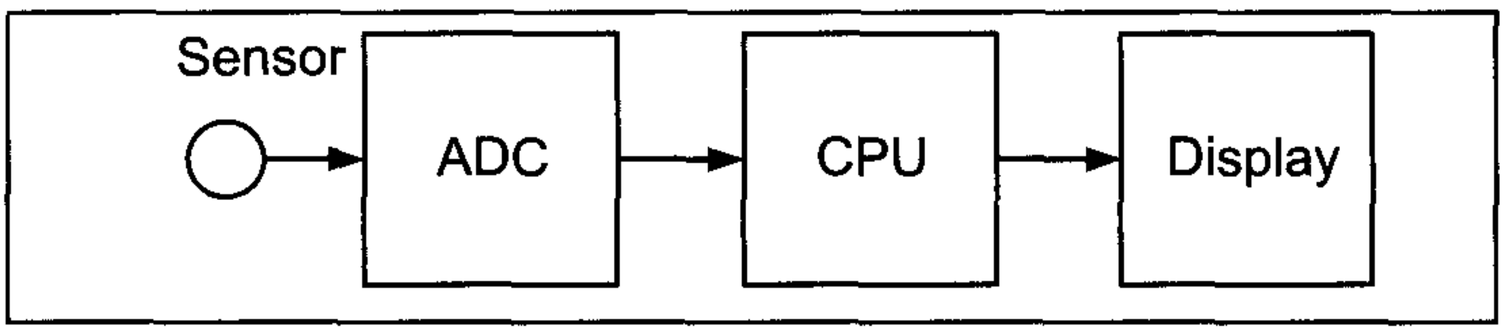


Figure 13-1. Microcontroller Connection to Sensor via ADC

ADC, DAC AND SENSOR INTERFACING

13.1 ADC CHARACTERISTICS

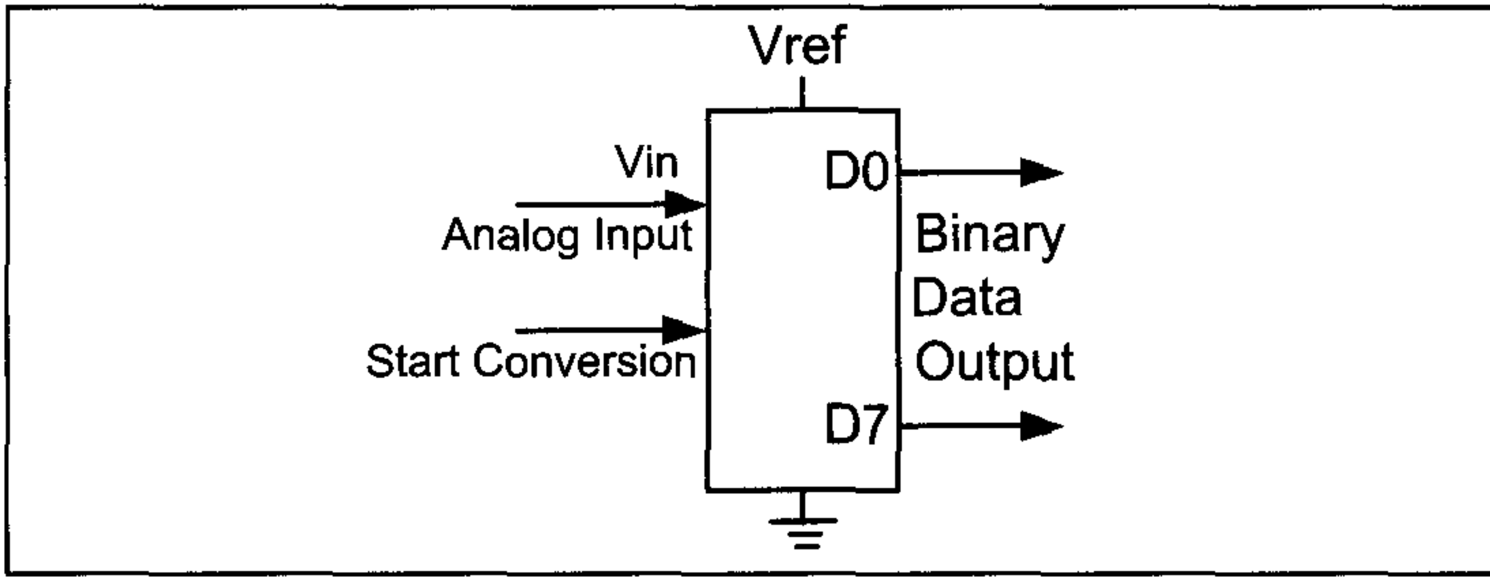


Figure 13-2. An 8-bit ADC Block Diagram

Page 474

ADC, DAC AND SENSOR INTERFACING
13.1 ADC CHARACTERISTICS

Some of the major characteristics of the ADC

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. Some widely used resolutions for ADCs are shown in Table 13-1. 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}

Table 13-1: Resolution versus Step Size for ADC (V_{ref} = 5 V)

n-bit	Number of steps	Step size (mV)
8	256	5/256 = 19.53
10	1024	5/1024 = 4.88
12	4096	5/4096 = 1.2
16	65,536	5/65,536 = 0.076

Notes: V_{CC} = 5 V

Step size (resolution) is the smallest change that can be discerned by an ADC.

ADC, DAC AND SENSOR INTERFACING

13.1 ADC CHARACTERISTICS

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.

13.1 ADC CHARACTERISTICS

Vref

Vref 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 Vref/256 because it is an 8-bit ADC, and 2 to the power of 8 gives us 256 steps.

13.1 ADC CHARACTERISTICS

. For example, if the analog input range needs to be 0 to 4 volts, Vref is connected to 4 volts. That gives 4V/256=15.62 mV for the step size of an 8-bit ADC. In another case, if we need a step size of 10 mV for an 8-bit ADC, then Vref=2.56 V, because 2.56V/256 =10 mV.

Table 13-2: V_{ref} Relation to V_{in} Range for an 8-bit ADC

V_{ref} (V)	V_{in} Range (V)	Step Size (mV)
5.00	0 to 5	5/256 = 19.53
4.0	0 to 4	4/256 = 15.62
3.0	0 to 3	3/256 = 11.71
2.56	0 to 2.56	2.56/256 = 10
2.0	0 to 2	2/256 = 7.81
1.28	0 to 1.28	1.28/256 = 5
1	0 to 1	1/256 = 3.90

Step size is V_{ref} / 256

13.1 ADC CHARACTERISTICS

For the 10-bit ADC, if the Vref=5V, then the step size is 4.88mV as shown in Table 13-1. Tables 13-2 and 13-3 show the relationship between the Vref and step size for the 8- and 10-bit ADCs, respectively. In some applications, we need the differential reference voltage where Vref=VRf (+) - Vref(-). Often the Vref (-) pin is connected to ground and the Vref (+) pin is used as the Vref.

Table 13-3: V_{ref} Relation to V_{in} Range for an 10-bit ADC

V_{ref} (V)	V_{in} (V)	Step Size (mV)
5.00	0 to 5	5/1024 = 4.88
4.096	0 to 4.096	4.096/1024 = 4
3.0	0 to 3	3/1024 = 2.93
2.56	0 to 2.56	2.56/1024 = 2.5
2.048	0 to 2.048	2.048/1024 = 2
1.28	0 to 1.28	1/1024 = 1.25
1.024	0 to 1.024	1.024/1024 = 1

ADC, DAC AND SENSOR INTERFACING
13.1 ADC CHARACTERISTICS

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, we use the following formula:

$$D_{out} = \frac{V_{in}}{step\ size}$$

where D_{out} = digital data output (in decimal), V_{in}, = analog input voltage, and step size (resolution) is the smallest change, which is $V_{ref}/256$ for an 8-bit ADC. See Example 13-1. This data is brought out of the ADC chip either one bit at a time (serially), or in one chunk, using a parallel line of outputs.

ADC, DAC AND SENSOR INTERFACING

13.1 ADC CHARACTERISTICS

Example 13-1

For an 8-bit ADC, we have V_{ref} = 2.56 V. Calculate the D0–D7 output if the analog input is: (a) 1.7 V, and (b) 2.1 V.

Solution:

Because the step size is 2.56/256 = 10 mV, we have the following:

(a) D_{out} = 1.7 V/10 mV=170 in decimal, which gives us 10101010 in binary for D7–D0.

(b) D_{out}= 2.1 V/10 mV = 210 in decimal, which gives us 11010010 in binary for D7–D0.

ADC, DAC AND SENSOR INTERFACING
13.1 ADC CHARACTERISTICS

Parallel versus serial ADC

The ADC chips are either parallel or serial. In parallel ADC, we have 8 or more pins dedicated to bringing out the binary data, but in serial ADC we have only one pin for data out. That means that inside the serial ADC, there is a parallel-in-serial-out shift register responsible for sending out the binary data one bit at a time. The DCD7 data pins of the 8-bit ADC provide an 8-bit parallel data path between the ADC chip and the CPU. Page 476

13.1 ADC CHARACTERISTICS

Parallel versus serial ADC

In the case of the 16-bit parallel ADC chip, we need 16 pins for the data path. In order to save pins, many 12- and 16-bit ADCs use pins D0-D7 to send out the upper and lower bytes of the binary data. In recent years, for many applications where space is a critical issue, using such a large number of pins for data is not feasible. For this reason, serial devices such as the serial ADC are becoming widely used.

13.1 ADC CHARACTERISTICS

Parallel versus serial ADC

While the serial ADCs use fewer pins and their smaller packages take much less space on the printed circuit board, more CPU time is needed to get the converted data from the ADC because the CPU must get data one bit at a time, instead of in one single read operation as with the parallel ADC. ADC848 is an example of a parallel ADC with 8 pins for the data output, while

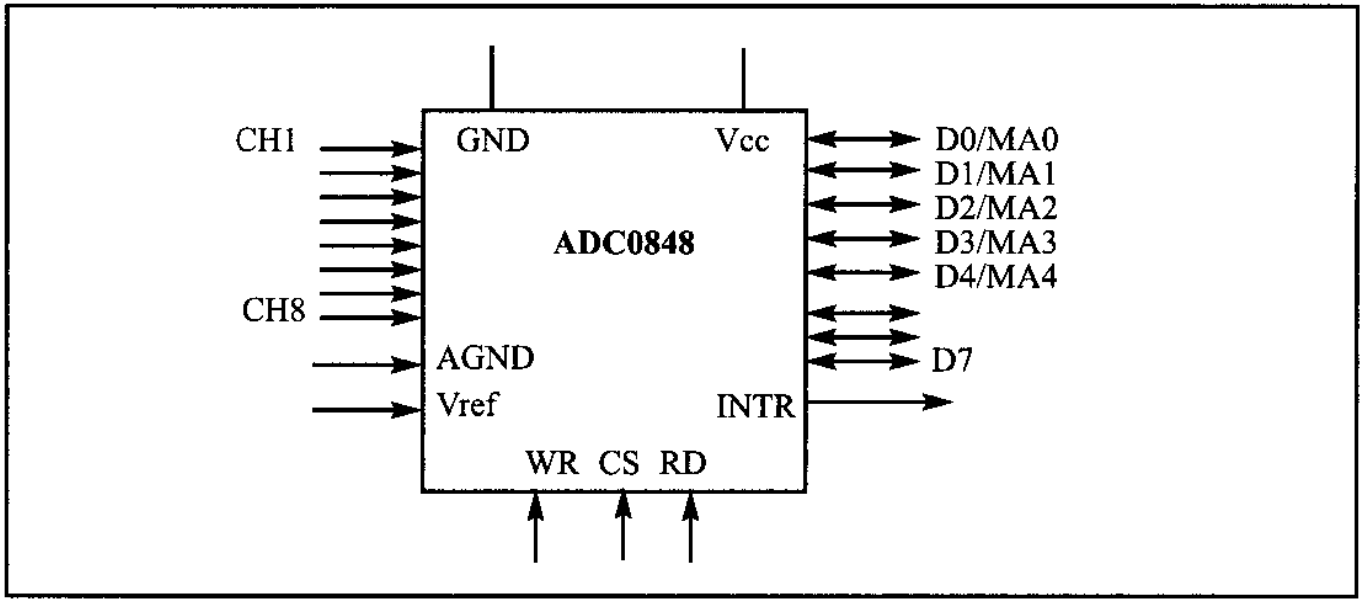


Figure 13-3. ADC0848 Parallel ADC Block Diagram

ADC, DAC AND SENSOR INTERFACING
13.1 ADC CHARACTERISTICS

The MAX1112 is an example of a serial ADC with a single pin for Dout.

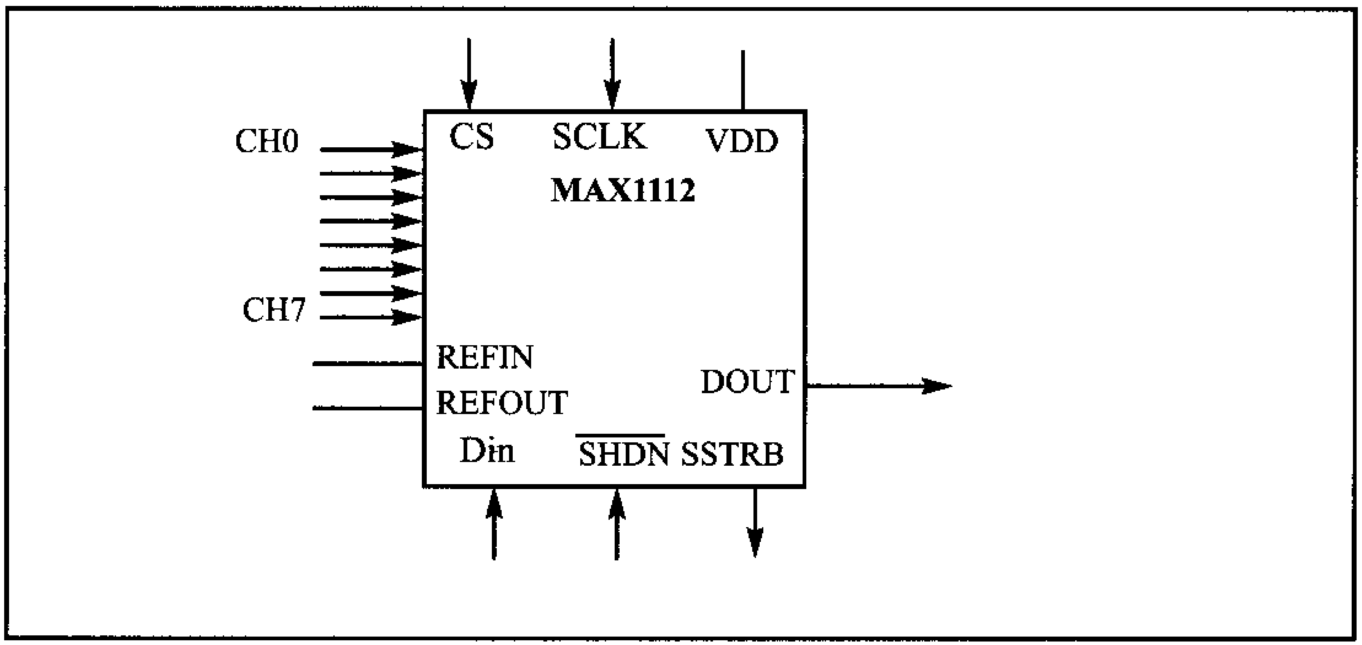


Figure 13-4. MAX1112 Serial ADC Block Diagram

ADC, DAC AND SENSOR INTERFACING

13.1 ADC CHARACTERISTICS

Analog input channels

Many data acquisition applications need more than one ADC. For this reason, we see ADC chips with 2, 4, 8, or even 16 channels on a single chip. Multiplexing of analog inputs is widely used as shown in the ADC848 and MAX1112. In these chips, we have 8 channels of analog inputs, allowing us to monitor multiple quantities such as temperature, pressure, heat, and so on. AVR microcontroller chips come with up to 16 ADC channels.

13.1 ADC CHARACTERISTICS

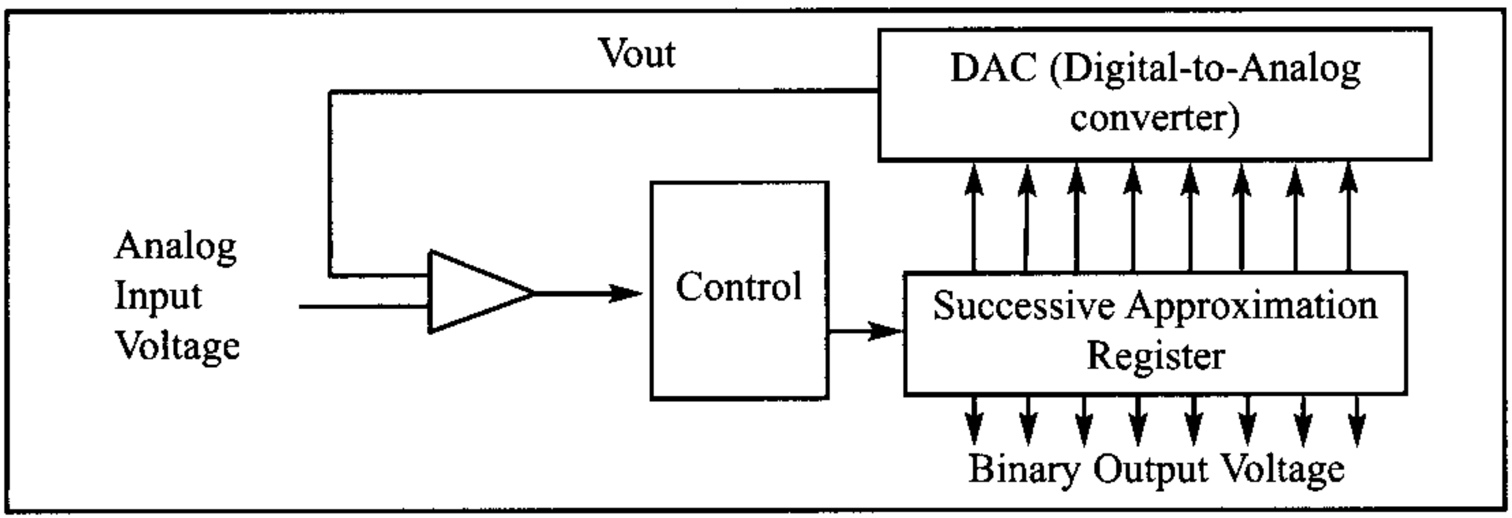
Start conversion and end-of-conversion signals

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 Vin to an n-bit digital number. The amount of time it takes to convert varies depending on the conversion method as was explained earlier. When the data conversion is complete, the end-of-conversion signal notifies the CPU that the converted data is ready to be picked up.

13.1 ADC CHARACTERISTICS

Successive Approximation ADC

Successive Approximation is a widely used method of converting an analog input to digital output. It has three main components: (a) successive approximation register (SAR), (b) comparator, and (c) control unit. See the figure below.



13.1 ADC CHARACTERISTICS

Assuming a step size of 10 mV, the 8-bit successive approximation ADC will go through the following steps to convert an input of 1 volt:

(1) It starts with binary 10000000. Since 128 x 10 mV = 1.28 V is greater than the 1 V input, bit 7 is cleared (dropped).

(2) 01000000 gives us 64 x 10 mV = 640 mV and bit 6 is kept since it is smaller than the 1V input.

(3) 01100000 gives us 96 x 10 mV = 960 mV and bit 5 is kept since it is smaller than the 1V input,

(4) 01110000 gives us 112 x 10 mV = 1120 mv and bit 4 is dropped since it is greater than the 1V input.

(5) 01 101000 gives us 108 x 10 mV = 1080 mV and bit **3 is** dropped since it is greater than the 1V input.

(6) 01100100 gives us 100 x 10 mV = 1000 mV = 1 V and bit 2 is kept since it is equal to input. Even though the answer is found it does not stop.

(7) 011000110 gives us 102 x 10 mV = 1020 mV and bit 1 is dropped since it is greater than the 1V input.

(8) 01100101 gives us 101 x 10 mV = 1010 mV and bit 0 is dropped since it is greater than the 1V input.

13.1 ADC CHARACTERISTICS

Notice that the Successive Approximation method goes through all the steps even if the answer is found in one of the earlier steps. The advantage of the Successive Approximation method is that the conversion time is fixed since it has to go through all the steps.

13.2 ADC PROGRAMMING IN THE AVR

ATmega32 ADC features

The ADC peripheral of the ATmega32 has the following characteristics:

- It is a 10-bit ADC
- It has 8 analog input channels, 7 differential input channels, and 2 differential input channels with optional gain of lox and 200x.
- The converted output binary data is held by two special function registers called ADCL (A/D Result Low) and ADCH (A/D Result High).
- 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.
- We have three options for V_{ref}. V_{ref} can be connected to AVCC (Analog V_{cc}), internal 2.56 V reference, or external AREF pin.
- The conversion time is dictated by the crystal frequency connected to the XTAL pins (Fosc) and ADPS0:2 bits.

13.2 ADC PROGRAMMING IN THE AVR

AVR ADC hardware considerations

For digital logic signals a small variation in voltage level has no effect on the output. For example, 0.2 V is considered LOW, since in TTL logic, anything less than 0.5 V will be detected as LOW logic. That is not the case when we are dealing with analog voltage. See Example 13-2.

We can use many techniques to reduce the impact of ADC supply voltage and Vref variation on the accuracy of ADC output. Next, we examine two of the most widely used techniques in the AVR.

13.2 ADC PROGRAMMING IN THE AVR

Example 13-2

For an 10-bit ADC, we have $V_{ref} = 2.56$ V. Calculate the D0–D9 output if the analog input is: (a) 0.2 V, and (b) 0 V. How much is the variation between (a) and (b)?

Solution:

Because the step size is 2.56/1024 = 2.5 mV, we have the following:

(a) $D_{out} = 0.2$ V/2.5 mV = 80 in decimal, which gives us 1010000 in binary.

(b) $D_{out} = 0$ V/2.5 mV = 0 in decimal, which gives us 0 in binary.

The difference is 1010000, which is 7 bits!

13.2 ADC PROGRAMMING IN THE AVR

Decoupling AVCC from VCC

As we mentioned in Chapter 8, 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. Figure 13-5 shows how to use an inductor and a capacitor to achieve this.

By connecting a capacitor between the AVREF pin and GND you can make the Vref voltage more stable and increase the precision of ADC.

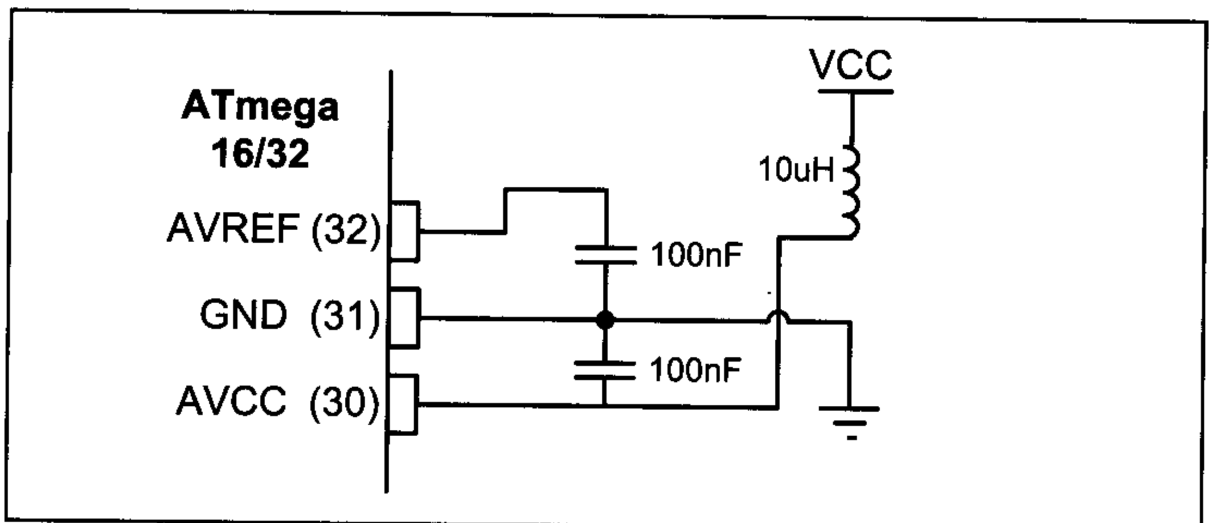


Figure 13-5. ADC Recommended Connection

13.2 ADC PROGRAMMING IN THE AVR

AVR programming in Assembly and C

In the AVR microcontroller five major registers are associated with the ADC that we deal with in this chapter. They are

- ADCH (high data),
- ADCL (low data),
- ADCSRA (ADC Control and Status Register),
- ADMUX (ADC multiplexer selection register), and
- SPIOR (Special Function I/O Register).

We examine each of them in this section.

13.2 ADC PROGRAMMING IN THE AVR

ADMUX register

Figure 13-6 shows the bits of ADMUX registers and their usage.

REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

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

ADLAR Bit 5 ADC Left Adjust Results
This bit dictates either the left bits or the right bits of the result registers ADCH:ADCL that are used to store the result. If we write a one to ADLAR, the result will be left adjusted; otherwise, the result is right adjusted.

MUX4:0 Bit 4:0 Analog Channel and Gain Selection Bits
The value of these bits selects the gain for the differential channels and also selects which combination of analog inputs are connected to the ADC.

Figure 13-6. ADMUX Register

13.2 ADC PROGRAMMING IN THE AVR

In this section we will focus more on the function of these bits.

V_{ref} *source*

Figure 13-7 shows the block diagram of internal circuitry of V_{ref} selection. As you can see we have three options: (a) AREF pin, (b) AVCC pin, or (c) internal 2.56 V.

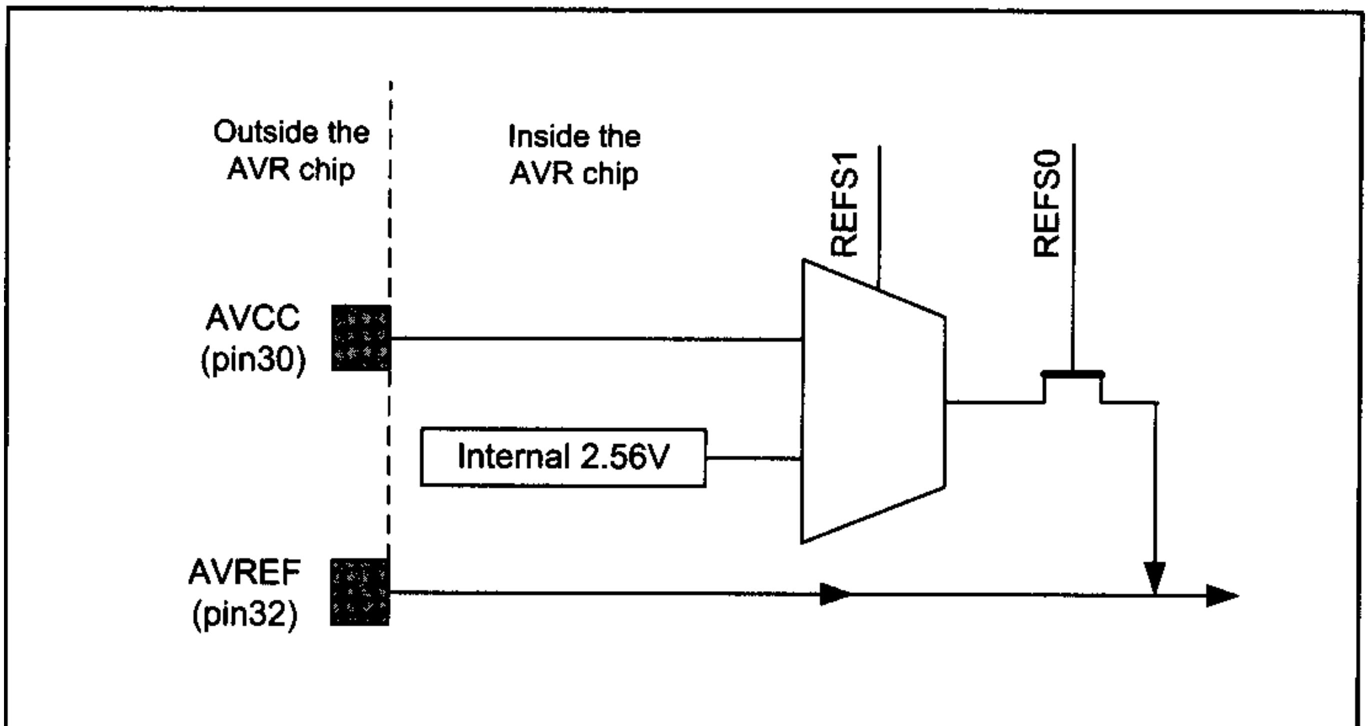


Figure 13-7. ADC Reference Source Selection

13.2 ADC PROGRAMMING IN THE AVR

Table 13-4 shows how the REFS1 and REFS0 bits of the ADMUX register can be used to select the Vref source.

Table 13-4: V_{ref} Source Selection Table for AVR

REFS1	REFS0	V_{ref}	
0	0	AREF pin	Set externally
0	1	AVCC pin	Same as VCC
1	0	Reserved	----
1	1	Internal 2.56 V	Fixed regardless of VCC value

Notice that if you connect the VREF 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 shorted with the external voltage.

If you choose 2.56 V as the Vref, the step size of ADC will be 2.56 / 1024 = 1014 = 2.5 mV. Such a round step size will reduce the calculations in software.

13.2 ADC PRO

ADC input channel source

Figure 13-8 shows the schematic of the internal circuitry of input channel selection. As you can see in the figure, either single-ended or the differential input can be selected to be converted to digital data.

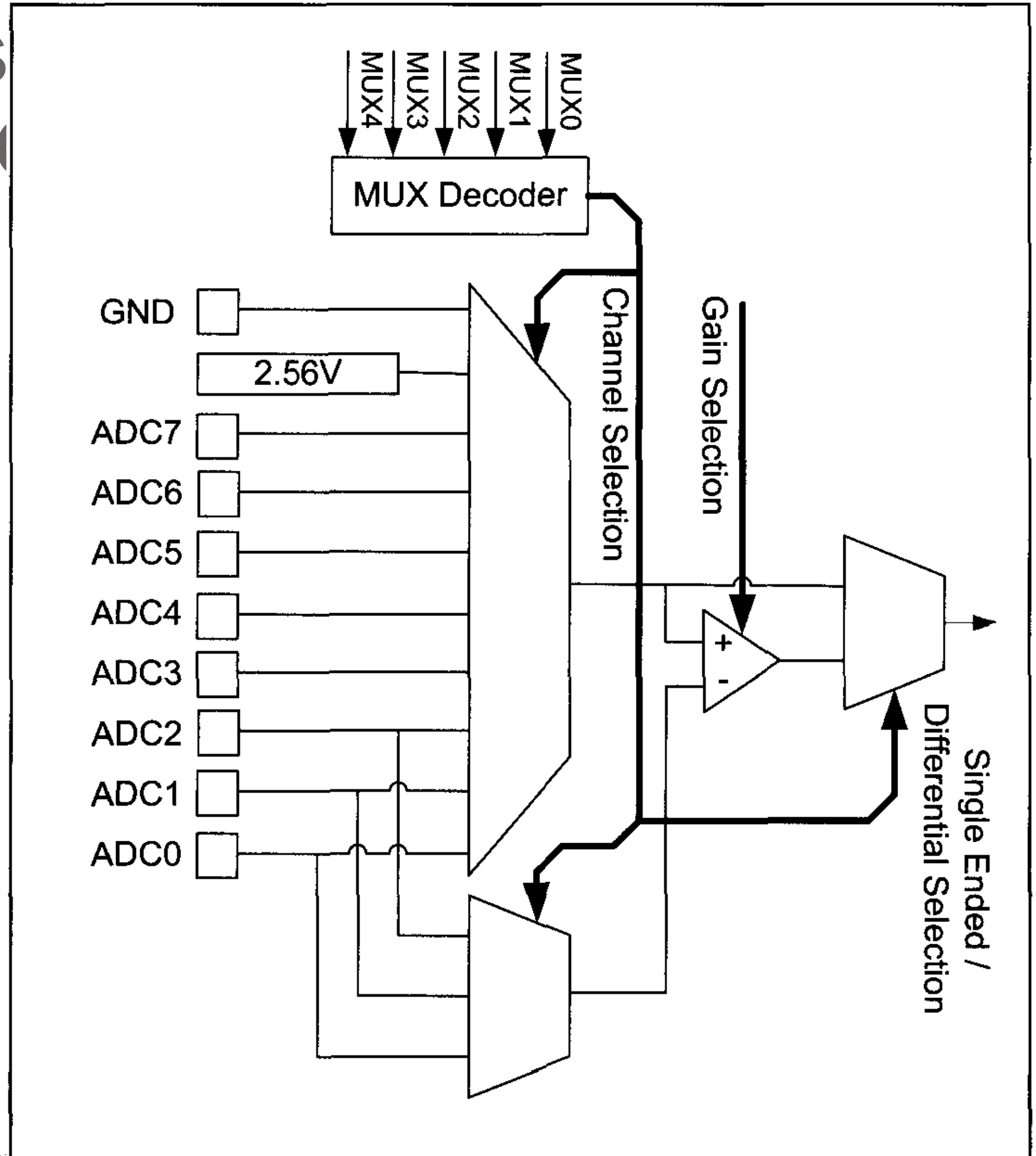


Figure 13-8. ADC Input Channel Selection

13.2 ADC PROGRAMMING IN THE AVR

If you select single-ended input, you can choose the input channel among ADC0 to ADC7. In this case a single pin is used as the analog line, and GND of the AVR chip is used as common ground. Table 13-5 lists the values of MUX4-MUX0 bits for different single ended inputs.

Table 13-5: Single-ended Channels

MUX4...0	Single-ended Input
00000	ADC0
00001	ADC1
00010	ADC2
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7

13.2 ADC PROGRAMMING IN THE AVR

If you choose differential input, you can also select the op-amp gain. You can choose the gain of the op-amp to be 1x, 10x, or 200x. You can select the positive input of the op-amp to be one of the pins ADC0 to ADC7, and the negative input of the op-amp can be any of ADC0, ADC1, or ADC2 pins.

Table 13-6: V_{ref} Source Selection Table

MUX4...0	+ Differential Input	– Differential Input	Gain
01000 *	ADC0	ADC0	10x
01001	ADC1	ADC0	10x
01010 *	ADC0	ADC0	200x
01011	ADC1	ADC0	200x
01100 *	ADC2	ADC2	10x
01101	ADC3	ADC2	10x
01110 *	ADC2	ADC2	200x
01111	ADC3	ADC2	200x
10000	ADC0	ADC1	1x
10001 *	ADC1	ADC1	1x
10010	ADC2	ADC1	1x
10011	ADC3	ADC1	1x
10100	ADC4	ADC1	1x
10101	ADC5	ADC1	1x
10110	ADC6	ADC1	1x
10111	ADC7	ADC1	1x
11000	ADC0	ADC2	1x
11001	ADC1	ADC2	1x
11010 *	ADC2	ADC2	1x
11011	ADC3	ADC2	1x
11100	ADC4	ADC2	1x
11101	ADC5	ADC2	1x

*Note: The rows with * are not applicable.*

ADC, DAC AND SENSOR INTERFACING

13.2 ADC PROGRAMMING IN THE AVR

ADLAR bit operation

The AVRs have a 10-bit ADC, which means that the result is 10 bits long and cannot be stored in a single byte. In AVR two 8-bit registers are dedicated to the ADC result, but only 10 of the 16 bits are used. You can select the position of used bits in the bytes. Notice that changing the ADLAR bit will affect the ADC data register immediately.

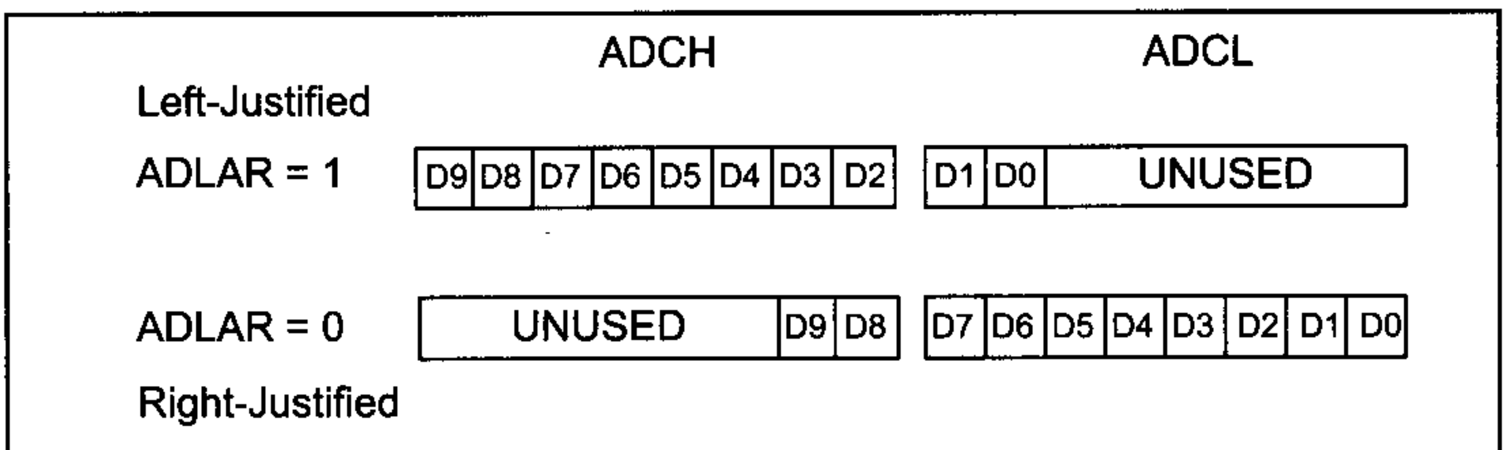


Figure 13-9. ADLAR Bit and ADCx Registers

13.2 ADC PROGRAMMING IN THE AVR

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.

ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

ADEN Bit 7 ADC Enable
This bit enables or disables the ADC. Setting this bit to one will enable the ADC, and clearing this bit to zero 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 one.

ADATE Bit 5 ADC Auto Trigger Enable
Auto triggering of the ADC is enabled when you set this bit to one.

ADIF Bit 4 ADC Interrupt Flag
This bit is set when an ADC conversion completes and the data registers are updated.

ADIE Bit 3 ADC Interrupt Enable
Setting this bit to one enables the ADC conversion complete interrupt.

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

Figure 13-10. ADCSRA (A/D Control and Status Register A)

13.2 ADC PROGRAMMING IN THE AVR

ADC Start Conversion bit

As we stated before, an ADC has a Start Conversion input. The AVR chip has a special circuit to trigger start conversion. As you see in Figure 13-1 1, in addition to the ADCSC bit of ADCSRA there are other sources to trigger start of conversion. If you set the ADATE bit of ADCSRA to high, you can select auto trigger source by updating ADTS2:0 in the SFIOR register. If ADATE is cleared, the ADTS2:0 settings will have no effect. Notice that there are many considerations if you want to use auto trigger mode.

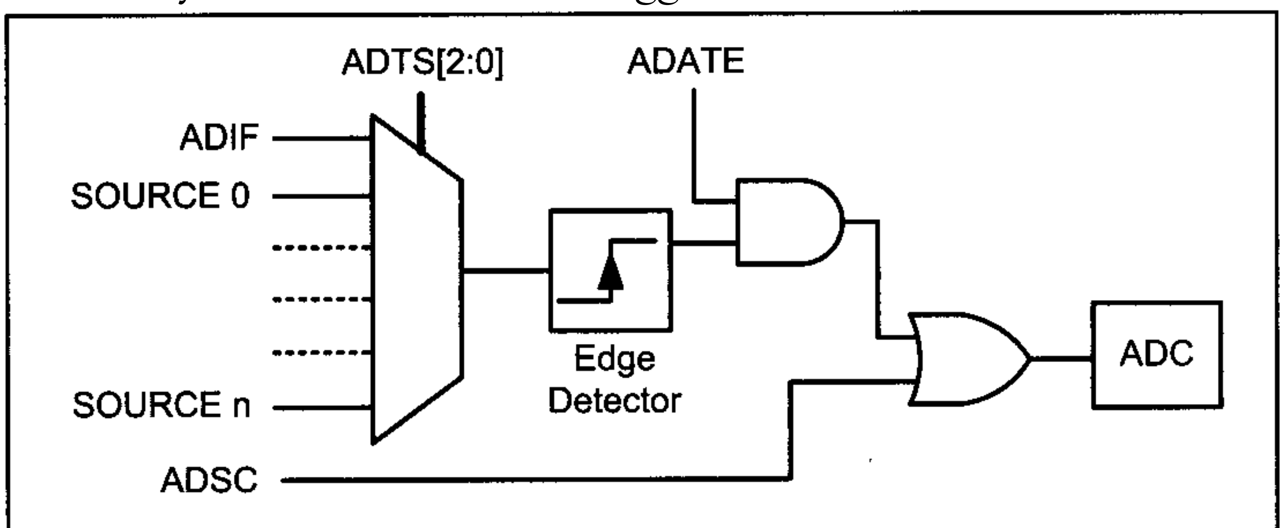
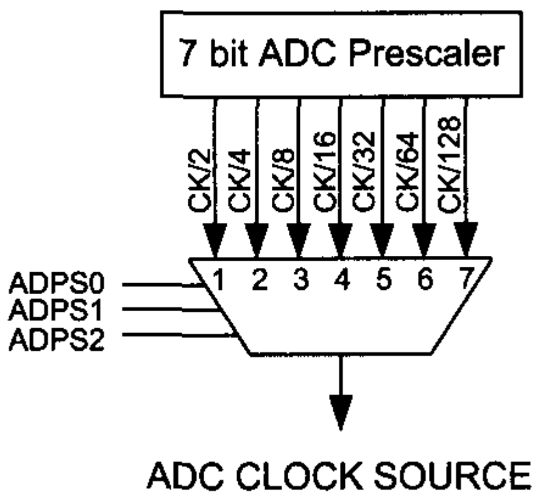


Figure 13-11. AVR ADC Trigger Source

ADC, DAC AND SENSOR INTERFACING
13.2 ADC PROGRAMMING IN THE AVR

AID conversion time

As you see in Figure 13-12, by using the ADPS2:0 bits of the ADCSRA register we can set the A/D conversion time. To select the conversion time, we can select any of Fosc/2, Fosc/4, Fosc/8, Fosc/16, Fosc/32, Fosc/64, or Fosc/128 for ADC clock, where Fosc is the speed of the crystal frequency connected to the AVR chip. For the AVR, the ADC requires an input clock frequency less than 200 kHz for the maximum accuracy.



ADPS2	ADPS1	ADPS0	ADC Clock
0	0	0	Reserved
0	0	1	CK/2
0	1	0	CK/4
0	1	1	CK/8
1	0	0	CK/16
1	0	1	CK/32
1	1	0	CK/64
1	1	1	CK/128

Figure 13-12. AVR ADC Clock Selection

13.2 ADC PROGRAMMING IN THE AVR

Example 13-3

An AVR is connected to the 8 MHz crystal oscillator. Calculate the ADC frequency for (a) ADPS2:0 = 001 (b) ADPS2:0 = 100 (c) ADPS2:0 = 111

Solution:

(a) Because ADPS2:0 = 001 (1 decimal), the ck/2 input will be activated; we have
8 MHz / 2 = 4 MHz (greater than 200 kHz and not valid)

(b) Because ADPS2:0 = 100 (4 decimal), the ck/8 input will be activated; we have
8 MHz / 16 = 500 kHz (greater than 200 kHz and not valid)

(c) Because ADPS2:0 = 111 (7 decimal), the ck/128 input will be activated; we have
8 MHz / = 62 kHz (a valid option since it is less than 200 kHz)

ADC, DAC AND SENSOR INTERFACING

13.2 ADC PROGRAMMING IN THE AVR

Sample-and-hold time in ADC

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 (C hold) 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. Then each consecutive conversion takes 13 ADC clock cycles.

Table 13-7: Conversion Time Table

Condition	Sample and Hold Time (Cycles)	Total Conversion Time (Cycles)
First Conversion	14.5	25
Normal Conversion, Single-ended	1.5	13
Normal Conversion, Differential	2	13.5
Auto trigger conversion	1.5 / 2.5	13/14

13.2 ADC PROGRAMMING IN THE AVR

Steps in programming the AID converter using polling

To program the A/D converter of the AVR, the following steps must 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.

mashhoun@iust.ac.ir Iran Univ of Science & Tech

13.2 ADC PROGRAMMING IN THE AVR

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 registers to get the digital data output. Notice that you have to read ADCL before 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.

ADC, DAC AND SENSOR INTERFACING
13.2 ADC PROGRAMMING IN THE AVR

Programming AVR ADC in Assembly and C

The Assembly language Program 13-1 illustrates the steps for ADC conversion shown above. Figure 13-13 shows the hardware connection of Program 13-1.

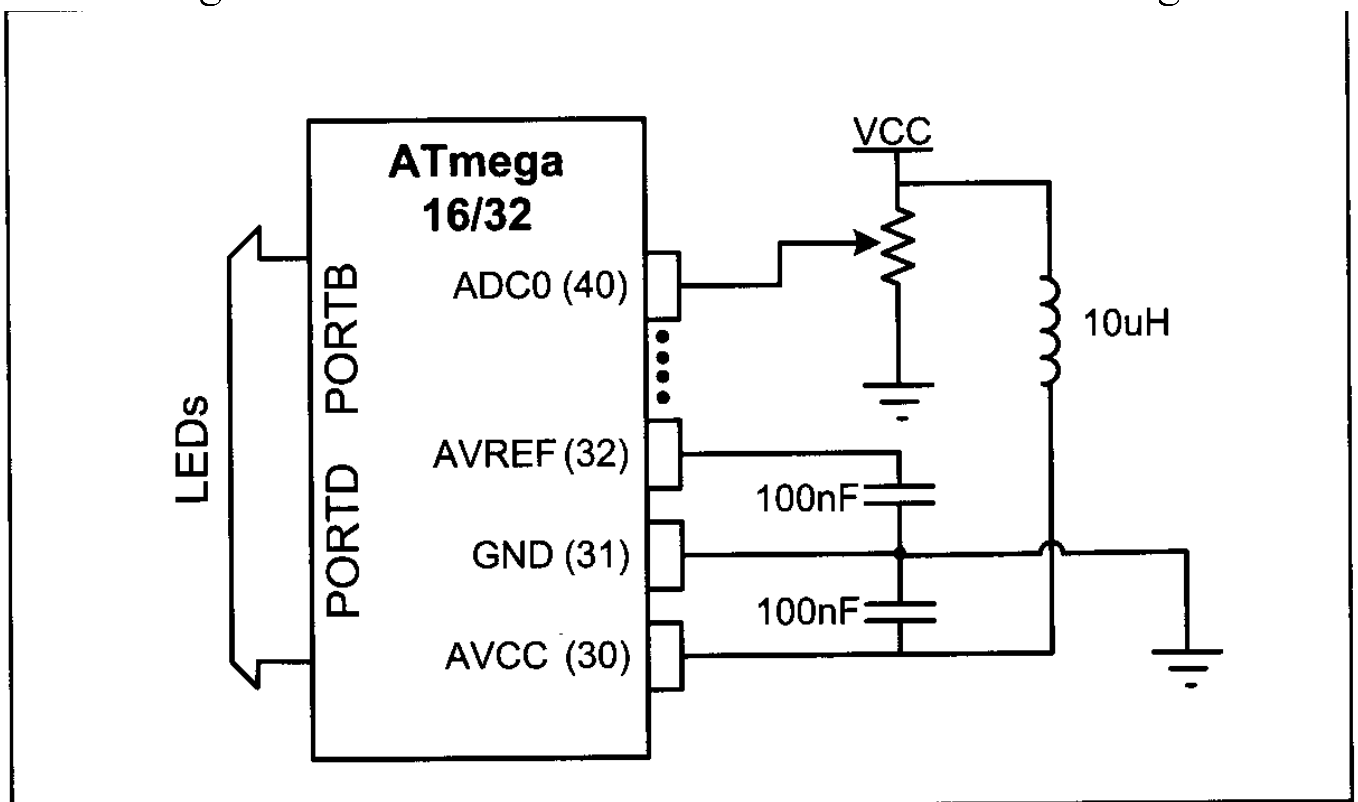


Figure 13-13. ADC Connection for Program 13-1

13.2 ADC PROGRAMMING IN THE AVR

```
;Program 13-1: This program gets data from channel 0 (ADC0) of
;ADC and displays the result on Port C and Port D. This is done
;forever.
;****************** Program 13-1 *************************

.INCLUDE "M32DEF.INC"
      LDI   R16,0xFF
      OUT   DDRB, R16        ;make Port B an output
      OUT   DDRD, R16        ;make Port D an output
      LDI   R16,0
      OUT   DDRA, R16        ;make Port A an input for ADC
      LDI   R16,0x87         ;enable ADC and select ck/128
      OUT   ADCSRA, R16
      LDI   R16,0xC0         ;2.56V Vref, ADC0 single ended
      OUT   ADMUX, R16       ;input, right-justified data
READ_ADC:
      SBI   ADCSRA,ADSC      ;start conversion
KEEP_POLING:                 ;wait for end of conversion
      SBIS  ADCSRA,ADIF      ;is it end of conversion yet?
      RJMP  KEEP_POLING      ;keep polling end of conversion
      SBI   ADCSRA,ADIF      ;write 1 to clear ADIF flag
      IN    R16,ADCL         ;YOU HAVE TO READ ADCL FIRST
      OUT   PORTD,R16        ;give the low byte to PORTD
      IN    R16,ADCH         ;READ ADCH AFTER ADCL
      OUT   PORTB,R16        ;give the high byte to PORTB
      RJMP  READ_ADC         ;keep repeating it
```

Program 13-1: Reading ADC Using Polling Method in Assembly

13.2 ADC PROGRAMMING IN THE AVR

Program 13-1C is the C version of the ADC conversion for Program 13-1.

```
#include <avr/io.h>           //standard AVR header
int main (void)
{
  DDRB = 0xFF;                //make Port B an output
  DDRD = 0xFF;                //make Port D an output
  DDRA = 0;                   //make Port A an input for ADC input
  ADCSRA= 0x87;               //make ADC enable and select ck/128
  ADMUX= 0xC0;                //2.56V Vref, ADC0 single ended input
                              //data will be right-justified
  while (1){
    ADCSRA|=(1<<ADSC);        //start conversion
    while((ADCSRA&(1<<ADIF))==0);//wait for conversion to finish
    PORTD = ADCL;             //give the low byte to PORTD
    PORTB = ADCH;             //give the high byte to PORTB
  }
  return 0;
}
```

Program 13-1C: Reading ADC Using Polling Method in C

13.2 ADC PROGRAMMING IN THE AVR

Programming A/D converter using interrupts

In Chapter 10, we showed how to use interrupts instead of polling to avoid tying down the microcontroller. To program the A/D using the interrupt method, we need to set HIGH the ADIE (A/D interrupt enable) flag. Upon completion of conversion, the ADIF (A/D interrupt flag) changes to HIGH; if ADIE = 1, it will force the CPU to jump to the ADC interrupt handler. Programs 13-2 and 13-2C show how to read ADC using interrupts.

```
.INCLUDE "M32DEF.INC"
.CSEG
      RJMP  MAIN
.ORG  ADCCaddr
      RJMP  ADC_INT_HANDLER
.ORG  40
;*****************************
```

13.2 ADC PROGRAMMING IN THE AVR

```
MAIN: LDI   R16, HIGH(RAMEND)
      OUT   SPH, R16
      LDI   R16, LOW(RAMEND)
      OUT   SPL,R16
      SEI
      LDI   R16,0xFF
      OUT   DDRB, R16         ;make Port B an output
      OUT   DDRD, R16         ;make Port D an output
      LDI   R16,0
      OUT   DDRA, R16         ;make Port A an input for ADC
      LDI   R16,0x8F          ;enable ADC and select ck/128
      OUT   ADCSRA, R16
      LDI   R16,0xC0          ;2.56V Vref, ADC0 single ended
      OUT   ADMUX, R16        ;input right-justified data
      SBI   ADCSRA,ADSC       ;start conversion
WAIT_HERE:
      RJMP  WAIT_HERE         ;keep repeating it
;****************************
ADC_INT_HANDLER:
      IN    R16,ADCL          ;YOU HAVE TO READ ADCL FIRST
      OUT   PORTD,R16         ;give the low byte to PORTD
      IN    R16,ADCH          ;READ ADCH AFTER ADCL
      OUT   PORTB,R16         ;give the high byte to PORTB
      SBI   ADCSRA,ADSC       ;start conversion again
      RETI
```

13.2 ADC PROGRAMMING IN THE AVR

Program 13-2C is the C version of Program 13-2. Notice that this program is checked under WinAVR (20080610).

```
#include <avr\io.h>
#include <avr\interrupt.h>
ISR(ADC_vect){
  PORTD = ADCL;               //give the low byte to PORTD
  PORTB = ADCH;               //give the high byte to PORTB
  ADCSRA|=(1<<ADSC);          //start conversion
}
int main (void){
  DDRB = 0xFF;                //make Port B an output
  DDRD = 0xFF;                //make Port D an output
  DDRA = 0;                   //make Port A an input for ADC input
  sei();                      //enable interrupts
  ADCSRA= 0x8F;               //enable and interrupt select ck/128
  ADMUX= 0xC0;                //2.56V Vref and ADC0 single-ended
                              //input right-justified data
  ADCSRA|=(1<<ADSC);          //start conversion
  while (1);                  //wait forever
  return 0;
}
```

Program 13-2C: Reading ADC Using Interrupts in C

Temperature sensors

Transducers convert physical data such as temperature, light intensity, flow, and speed to electrical signals. Depending on the transducer, the output produced is in the form of voltage, current, resistance, or capacitance. For example, temperature is converted to electrical signals using a transducer called a thermistor.

A *thermistor* responds to temperature change by changing resistance, but its response is not linear, as seen in Table 13-8.

Table 13-8: Thermistor Resistance vs. Temperature

Temperature (C)	Tf (K ohms)
0	29.490
25	10.000
50	3.893
75	1.700
100	0.817

From William Kleitz, Digital Electronics

13.3 SENSOR INTERFACING AND SIGNAL CONDITIONING

The complexity associated with writing software for such nonlinear devices has led many manufacturers to market a linear temperature sensor. Simple and widely used linear temperature sensors include the LM34 and LM35 series from National Semiconductor Corp. They are discussed next.

LM34 and LM35 temperature sensors

The sensors of the LM34 series are precision integrated-circuit temperature sensors whose output voltage is linearly proportional to the Fahrenheit temperature. See Table 13-9. The LM34 requires no external calibration because it is internally calibrated. It outputs 10 mV for each degree of Fahrenheit temperatur

Table 13-9: LM34 Temperature Sensor Series Selection Guide

Part Scale	Temperature Range	Accuracy	Output
LM34A	–50 F to +300 F	+2.0 F	10 mV/F
LM34	–50 F to +300 F	+3.0 F	10 mV/F
LM34CA	–40 F to +230 F	+2.0 F	10 mV/F
LM34C	–40 F to +230 F	+3.0 F	10 mV/F
LM34D	–32 F to +212 F	+4.0 F	10 mV/F

Note: Temperature range is in degrees Fahrenheit.

ADC, DAC AND SENSOR INTERFACING
13.3 SENSOR INTERFACING AND SIGNAL CONDITIONING

Table 13-10: LM35 Temperature Sensor Series Selection Guide

Part	Temperature Range	Accuracy	Output Scale
LM35A	–55 C to +150 C	+1.0 C	10 mV/C
LM35	–55 C to +150 C	+1.5 C	10 mV/C
LM35CA	–40 C to +110 C	+1.0 C	10 mV/C
LM35C	–40 C to +110 C	+1.5 C	10 mV/C
LM35D	0 C to +100 C	+2.0 C	10 mV/C

Note: Temperature range is in degrees Celsius.

The LM35 series sensors are precision integrated-circuit temperature sensors whose output voltage is linearly proportional to the Celsius (centigrade) temperature. The LM35 requires no external calibration because it is internally calibrated. It outputs 10 mV for each degree of centigrade temperature. Table 13-10 is the selection guide for the LM35. (For further information see http://www.national.com.)

Signal conditioning

Signal conditioning is widely used in the world of data acquisition. The most common transducers produce an output in the form of voltage, current, charge, capacitance, and resistance. We need to convert these signals to voltage, however, in order to send input to an A-to-D converter. This conversion (modification) is commonly called *signal conditioning. See* Figure 13-14. Signal conditioning can be current-to-voltage conversion or signal amplification. For example, the thermistor changes resistance with temperature. The change of resistance must be translated into voltages to be of any use to an ADC. We now look at the case of connecting an LM34 (or LM35) to an ADC of the

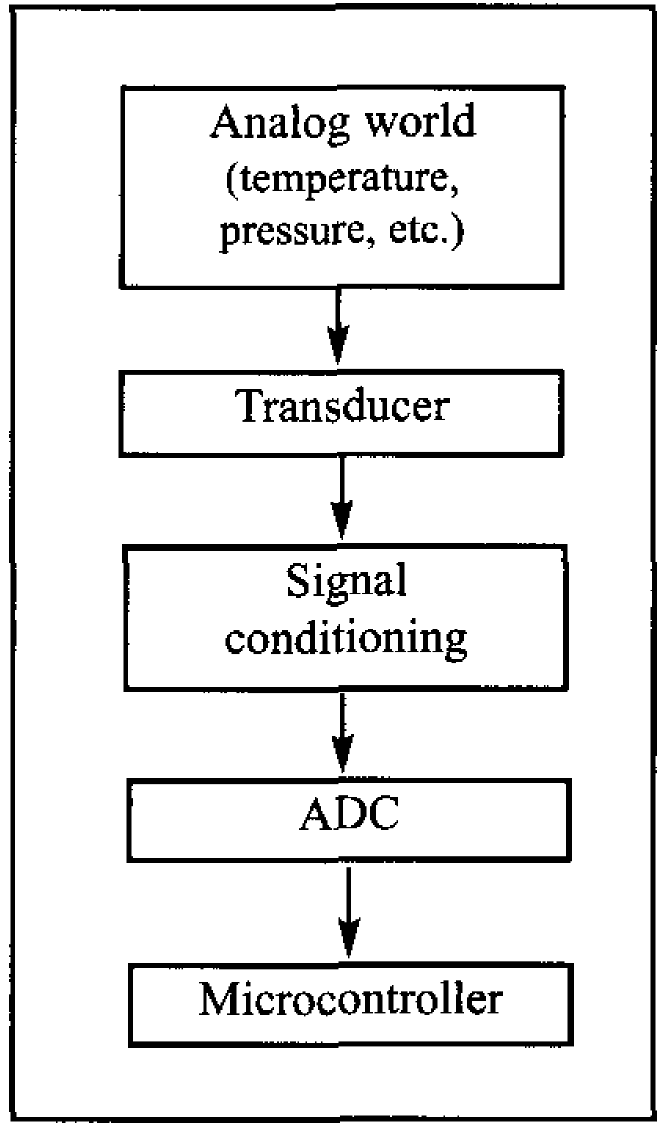


Figure 13-14. Getting Data from the Analog World

Interfacing the LM34 to the AVR

The A/D has 10-bit resolution with a maximum of 1024 steps, and the LM34 (or LM35) produces 10 mV for every degree of temperature change. Now, if we use the step size of 10 mV, the V_{out} will be 10,240 mV (10.24 V) for fullscale output. This is not acceptable even though the maximum temperthe A/D ature sensed by the LM34 is 300 degrees F, and the highest output we will get for is 3000 mV (3.00 V).

Now if we use the internal 2.56 V reference voltage, the step size would be 2.56V/1024 = 2.5 mV. This makes the binary output number for the ADC four times the real temperature because the sensor produces 10 mV for each degree of temperature change and the step size is 2.5 mV (10 mV/2.5 mV = 4). We can scale it by dividing it by 4 to get the real number for temperature. See Table 13-11.

mashhoun@iust.ac.ir Iran Univ of Science & Tech

13.3 SENSOR INTERFACING AND SIGNAL CONDITIONING

Table 13-11: Temperature vs. V_{out} for AVR with V_{ref} = 2.56 V

Temp. (F)	V_{in} (mV)	# of steps	Binary V_{out} (b9–b0)	Temp. in Binary
0	0	0	00 00000000	00000000
1	10	4	00 00000100	00000001
2	20	8	00 00001000	00000010
3	30	12	00 00001100	00000011
10	100	20	00 00101000	00001010
20	200	80	00 01010000	00010100
30	300	120	00 01111000	00011110
40	400	160	00 10100000	00101000
50	500	200	00 11001000	00110010
60	600	240	00 11110000	00111100
70	700	300	01 00011000	01000110
80	800	320	01 01000000	01010000
90	900	360	01 01101000	01011010
100	1000	400	01 10010000	01100100

13.3 SENSOR INTERFACING AND SIGNAL CONDITIONING

Figure 13-15 shows the pin configuration of the LM34/LM35 temperature sensor and the connection of the temperature sensor to the ATmega32.

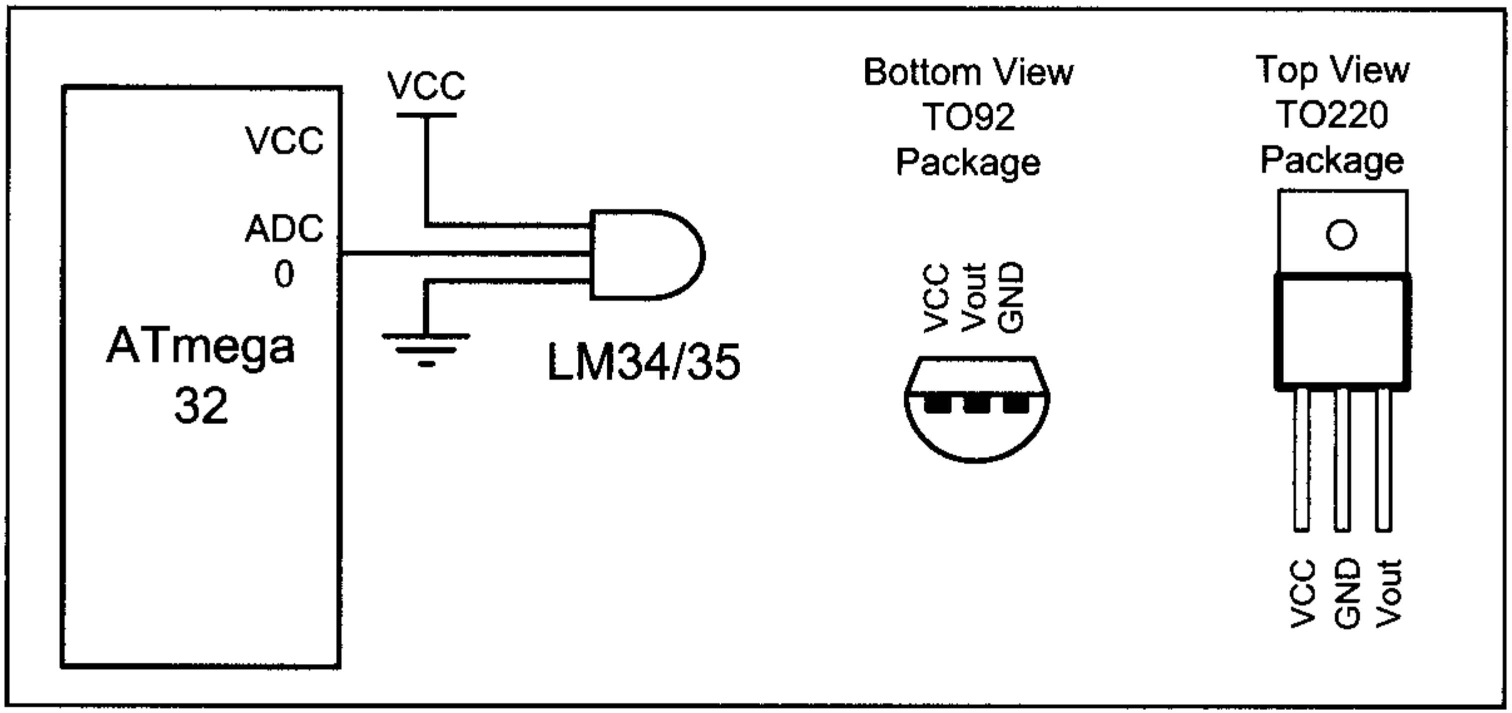


Figure 13-15. LM34/35 Connection to AVR and Its Pin Configuration

13.3 SENSOR INTERFACING AND SIGNAL CONDITIONING

Reading and displaying temperature

Programs 13-4 and 13-4C show code for reading and displaying temperature in both Assembly and C, respectively.

The programs correspond to Figure 13-15. Regarding these two programs, the following points must be noted:

1. The LM34 (or LM35) is connected to channel 0 (ADC0 pin).
2. The 10-bit output of the AID is divided by 4 to get the real temperature.
3. To divide the 10-bit output of the A/D by 4 we choose the left-justified option and only read the ADCH register. It is same as shifting the result two bits right. See Example 13-4.

mashhoun@iust.ac.ir Iran Univ of Science & Tech

ADC, DAC AND SENSOR INTERFACING
13.3 SENSOR INTERFACING AND SIGNAL CONDITIONING

```
;this program reads the sensor and displays it on Port D
.INCLUDE "M32DEF.INC"
      LDI   R16,0xFF
      OUT   DDRD, R16          ;make Port D an output
      LDI   R16,0
      OUT   DDRA, R16          ;make Port A an input for ADC
      LDI   R16,0x87           ;enable ADC and select ck/128
      OUT   ADCSRA, R16
      LDI   R16,0xE0           ;2.56 V Vref, ADC0 single-ended
      OUT   ADMUX, R16         ;left-justified data
READ_ADC:
      SBI   ADCSRA,ADSC        ;start conversion
KEEP_POLING:                   ;wait for end of conversion
      SBIS  ADCSRA,ADIF        ;is it end of conversion?
      RJMP  KEEP_POLING        ;keep polling end of conversion
      SBI   ADCSRA,ADIF        ;write 1 to clear ADIF flag
      IN    R16,ADCH           ;read only ADCH for 8 MSB of
      OUT   PORTD,R16          ;result and give it to PORTD
      RJMP  READ_ADC           ;keep repeating
```

Program 13-3: Reading Temperature Sensor in Assembly

13.3 SENSOR INTERFACING AND SIGNAL CONDITIONING

```
;this program reads the sensor and displays it on Port D
#include <avr/io.h>          //standard AVR header
int main (void)
{
  DDRD = 0xFF;               //make Port D an output
  DDRA = 0;                  //make Port A an input for ADC input
  ADCSRA = 0x87;             //make ADC enable and select ck/128
  ADMUX = 0xE0;              //2.56 V Vref and ADC0 single-ended
                             //data will be left-justified
  while (1){
      ADCSRA |= (1<<ADSC);   //start conversion
      while((ADCSRA&(1<<ADIF))==0); //wait for end of conversion
      PORTB = ADCH;          //give the high byte to PORTB
  }
  return 0;
}
```

Program 13-3C: Reading Temperature Sensor in C

13.3 SENSOR INTERFACING AND SIGNAL CONDITIONING

Example 13-4

In Table 13-11, verify the AVR output for a temperature of 70 degrees. Find values in the AVR A/D registers of ADCH and ADCL for left-justified.

Solution:

The step size is 2.56/1024 = 2.5 mV because Vref = 2.56 V.

For the 70 degrees temperature we have 700 mV output because the LM34 provides 10 mV output for every degree. Now, the number of steps are 700 mV/2.5 mV = 280 in decimal. Now 280 = 0100011000 in binary and the AVR A/D output registers have ADCH = 01000110 and ADCL = 00000000 for left-justified. To get the proper result we must divide the result by 4. To do that, we simply read the ADCH register, which has the value 70 (01000110) in it.

13.4 DAC INTERFACING

Digital-to-analog converter (DAC)

The digital-to-analog converter (DAC) is a device widely used to convert digital pulses to analog signals.

Recall from your digital electronics course the two methods of creating a DAC: binary weighted and R/2R ladder. The vast majority of integrated circuit DACs, including the MC1408 (DAC0808) used in this section, use the R/2R method because it can achieve a much higher degree of precision.

The first criterion for judging a DAC is its resolution, which is a function of the number of binary inputs. The common ones are 8, 10, and 12 bits. The number of data bit inputs decides the resolution of the DAC because the number of analog output levels is equal to 2^n, where n is the number of data bit inputs. Therefore, an 8-input DAC such as the DAC0808 provides 256 discrete voltage (or current) levels of output.

ADC, DAC AND SENSOR INTERFACING
13.4 DAC INTERFACING

See Figure 13-16. Similarly, the 12-bit DAC provides 4096 discrete voltage levels. There are also 16-bit DACs, but they are more expensive.

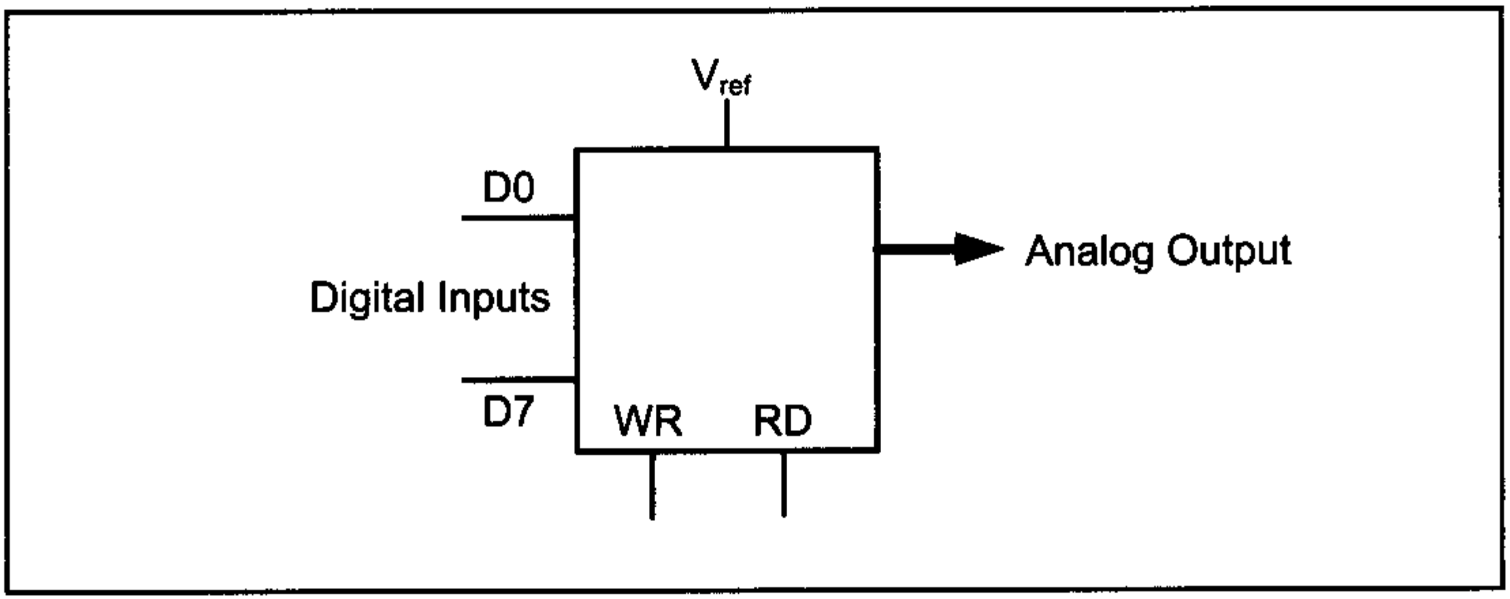


Figure 13-16. DAC Block Diagram

13.4 DAC INTERFACING

MC1408 DAC (or DAC0808)

In the MC1408 (DAC0808), the digital inputs are converted to current (I_{out}), and by connecting a resistor to the I_{out} pin, we convert the result to voltage. The total current provided by the I_{out} pin is a function of the binary numbers at the D0-D7 inputs of the DAC0808 and the reference current (I_{ref}), and is as follows:

$$I_{out} = I_{ref}\left(\frac{D7}{2} + \frac{D6}{4} + \frac{D5}{8} + \frac{D4}{16} + \frac{D3}{32} + \frac{D2}{64} + \frac{D1}{128} + \frac{D0}{256}\right)$$

where D0 is the LSB, D7 is the MSB for the inputs, and If is the input current that must be applied to pin 14. The I_{ref} current is generally set to 2.0 mA. Figure 13-17 shows the generation of current reference (setting I_{ref} = 2 mA) by using the standard 5 V power supply. Now assuming that I_{ref} = 2mA, if all the inputs to the DAC are high, the maximum output current is 1.99 mA.

13.4 DAC INTERFACING

Ideally we connect the output pin I_{out} to a resistor, convert this current to voltage, and monitor the output on the scope. In real life, however, this can cause inaccuracy because the input resistance of the load where it is connected will also affect the output voltage. For this reason, the I_{ref} current output is isolated by connecting it to an op-amp such as the 741 with R_f = 5 kilohms for the feedback resistor. Assuming that R = 5 kilohrns, by changing the binary input, the output voltage changes as shown in Example 13-5.

Example 13-5

Assuming that R = 5 kilohms and I_{ref} = 2 mA, calculate V_{out} for the following binary inputs:

(a) 10011001 binary (99H)
(b) 11001000 (C8H)

Solution:

(a) I_{out} = 2 mA (153/256) = 1.195 mA and V_{out} = 1.195 mA × 5K = 5.975 V
(b) I_{out} = 2 mA (200/256) = 1.562 mA and V_{out} = 1.562 mA × 5K = 7.8125 V

ADC, DAC AND SENSOR INTERFACING
13.4 DAC INTERFACING

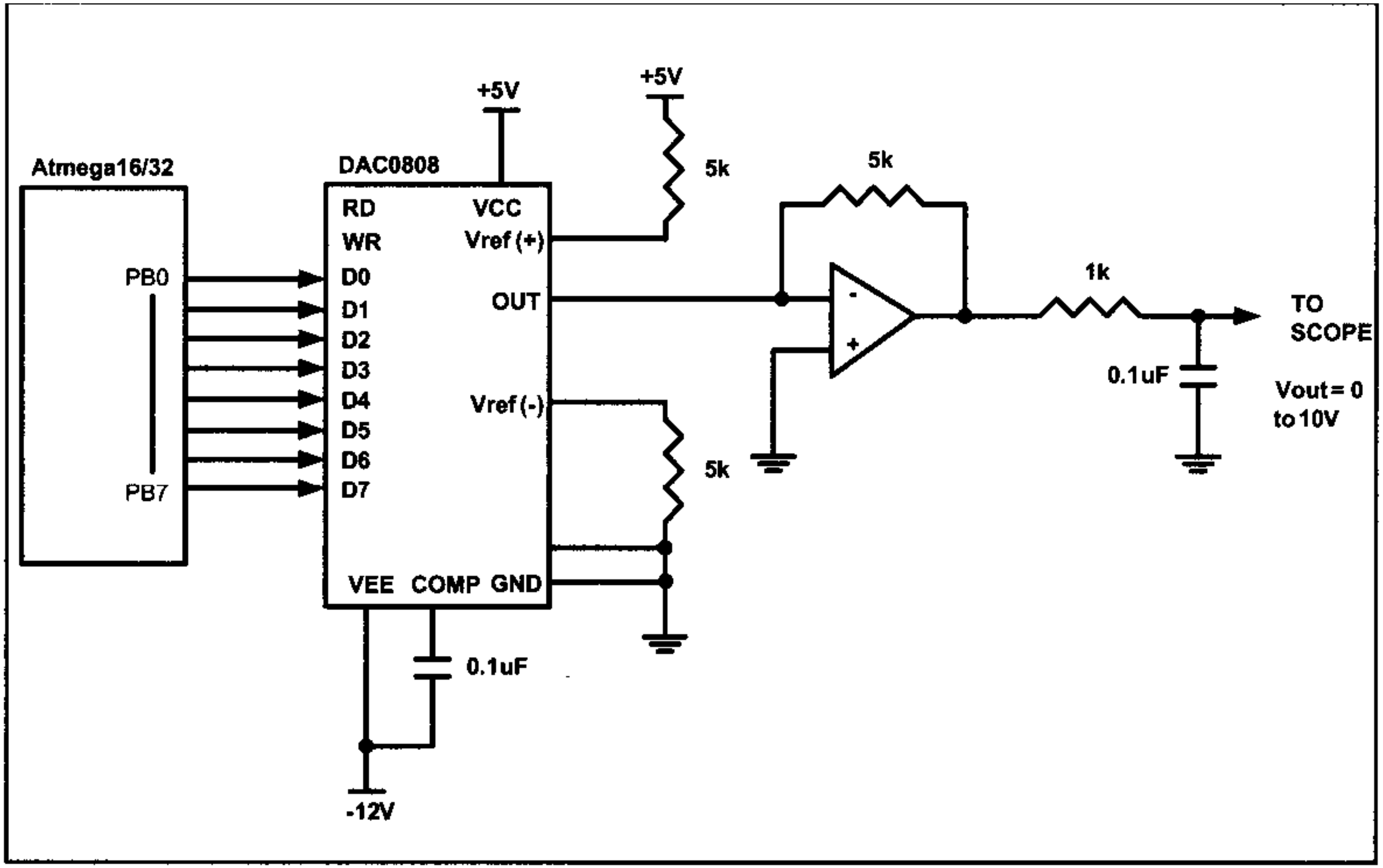


Figure 13-17. AVR Connection to DAC0808

13.4 DAC INTERFACING

Generating a stair-step ramp

In order to generate a stair-step ramp, you can set up the circuit in Figure 13-17 and load Program 13-4 on the AVR chip. To see the result wave, connect the output to an oscilloscope. Figure 13-18 shows the output.

```
LDI   R16,0xFF
      OUT   DDRB, R16          ;make Port B an output
AGAIN:
      INC   R16                ;increment R16
      OUT   PORTB,R16          ;sent R16 to PORTB
      NOP                      ;let DAC recover
      NOP
      RJMP  AGAIN
```

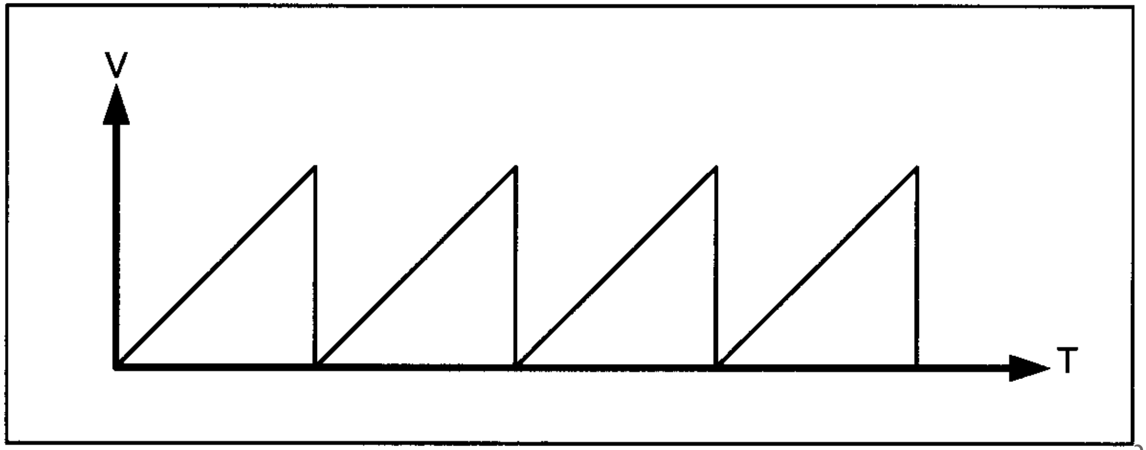


Figure 13-18. Stair Step Ramp Output

13.4 DAC INTERFACING

Programming DAC in C

Program 13-4C shows how to program the DAC in C.

```
#include <avr/io.h>            //standard AVR header

int main (void)
{
  unsigned char i = 0;         //define a counter
  DDRB = 0xFF;                 //make Port B an output
  while (1){                   //do forever
     PORTB = i;                //copy i into PORTB to be converted
     i++;                      //increment the counter
  }
  return 0;
}
```

Program 13-4C: DAC Programming in C