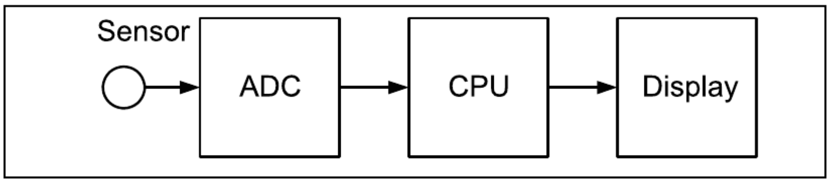
SENSOR INTERFACING

A sensor (also called transducer) accepts analog data (such as temperature) and convert it into electrical form (such as voltage, current, capacitance, resistance etc.). Analog to Digital Convertor (ADC) is used to convert these in to digital data. ADC accepts inputs in the form of voltage. So the electrical signals(current, capacitance, resistance etc) are first converted to voltage before giving to ADC. This is called signal conditioning. Then the digital data produced by ADC is processed by CPU as shown in the figure:



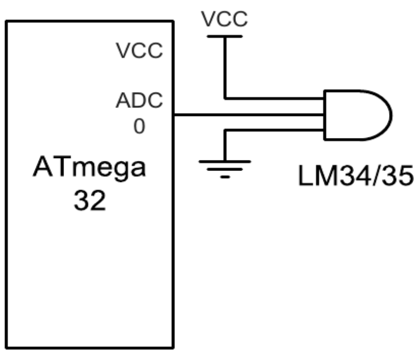
Temperature Sensors:

A temperature sensor (also called thermistor) accepts temperature as input and generates electrical signal as output.

- LM34/ LM35 series are widely used of temperature sensors developed by National Semiconductor Corp.
- It output 10mV for each degree of input temperature
- The output voltage produced by LM34 is linearly proportional to the Fahrenheit temperature.
- The output voltage produced by LM35 is linearly proportional to the Celsius temperature.

Interfacing LM34/35 to AVR

- The LM34 (or LM35) produces 10 mV for every degree of temperature change.
- The AVR ADC has 10-bit resolution with a maximum of 1024 steps
- If we use the internal 2.56 V reference voltage, the step size would be 2.56 V/1024 = 2.5mV
- This will make the output of ADC four times the input temperature (2.5mVx4=10mV)
- To get the real temperature in binary, we divide the output by 4. (Using the left-justified option and reading only the ADCH register gives the real temperature)
- Following figure shows connection of the temperature sensor (LM34/LM35) to the ATmega32, where the sensor is connected to channel 0 (ADC0 pin).

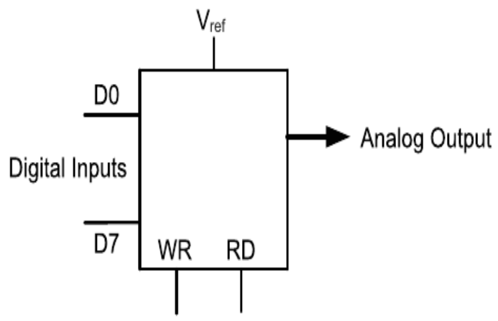


Qn: Write an AVR C program to interface LM35 temperature sensor with ATmega32 and send the digital data to PortB. Use ADC0 input channel, 2.56V internal reference voltage CK/128 prescaler.

```
#include <avr/io.h>          //standard AVR header
int main (void)
{
  DDRB = 0xFF                //make Port B  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;
}
```

DAC INTERFACING

- The digital-to-analog converter (DAC) is a device used to convert digital pulses to analog signals.
- Following figure shows block diagram of DAC

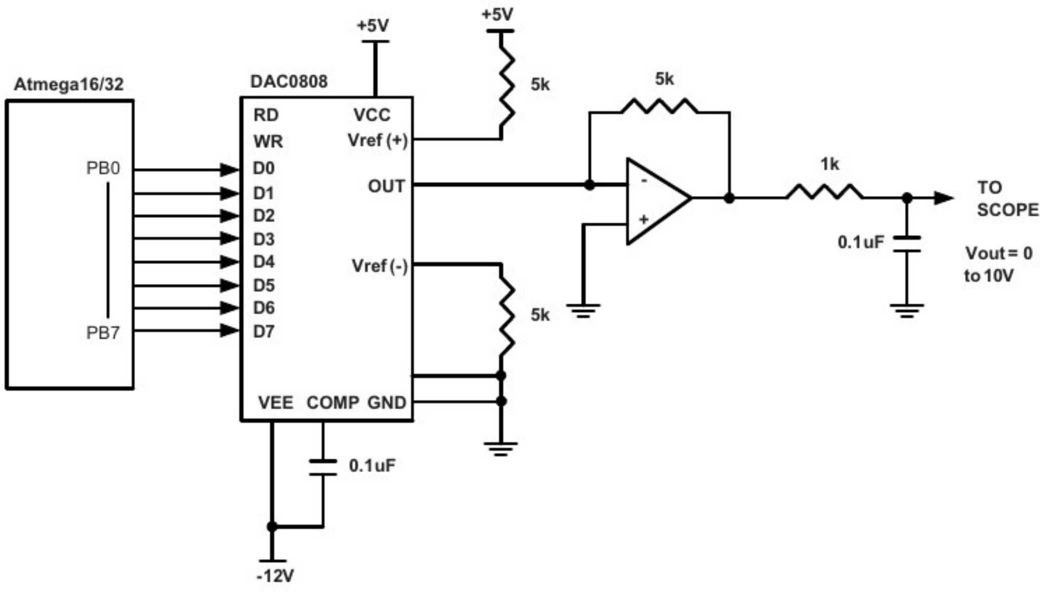


- It accepts digital data as inputs. The number of input bits is the resolution of DAC
- The number of analog output levels is equal to 2^n, where n is the number of data bit inputs
- For example, an 8 bit resolution DAC has 256 output voltage levels.
- A commonly used DAC is MC1408 DAC (or DAC0808).
- In the MC1408 (DAC0808), the digital inputs are converted to current (Iout), and by connecting a resistor to the Iout pin, we convert the result to voltage.
- The total current provided by the Iout pin is a function of the binary numbers at the D0–D7 inputs of the DAC0808 and the reference current (Iref), and is as follows:

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

Interfacing AVR with DAC

- Following figure shows the connection diagram of AVR to DAC0808



- The output form DAC0808 is connected to a resistor to convert the output current to voltage and monitor the output on a oscilloscope.
- AVR PORTB is connected to D0 -D7 pins of DAC0808

Example: AVR C program to generate a stair-step ramp

```
#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;
}
```

In the above program, the PORTB register will be incremented from 0 to 255 repeatedly which causes to generate a stair-step wave form as shown below: