EIE3105: ADC, DAC, and Sensor (Chapter 14)

Dr. Lawrence Cheung

Semester 2, 2021/22

Topics

- ADC
  - Introduction to ADC
  - ADC major characteristics
  - ADC in AVR
    - ADMUX
    - ADCH and ADCL
    - ADCSRA
  - ADC programming

Topics

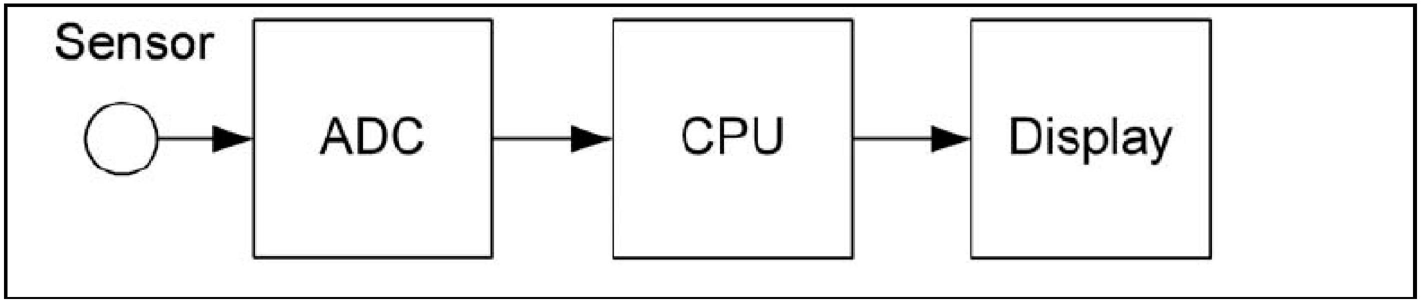
- Sensor
  - Signal conditioning
  - Sensor interfacing
- DAC

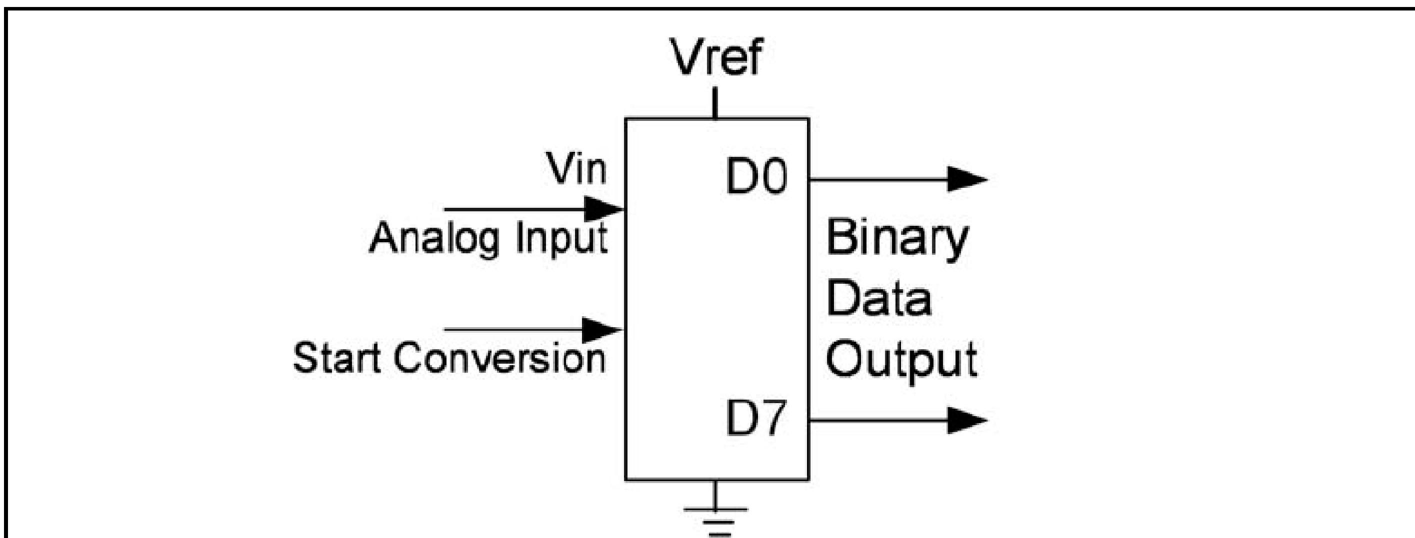
Introduction to ADC

- ADC = Analog-to-Digital Converter
- Computer system: Binary
- Physical world: Analog
  - Temperature
  - Pressure
  - Humidity
  - Velocity

Introduction to ADC

- Analogue vs. digital signal





ADC major characteristics

- Resolution (Step size)
  - The smallest change that can be discerned by an ADC
  - The higher resolution ADC provides a smaller step size.
- Conversion time
  - The time it takes the ADC to convert the analog input to a digital (binary) number.

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	65536	5/65536 = 0.076

Notes: $V_{CC} = 5$ V

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

ADC major characteristics

- V_{ref}
  - It is an input voltage used for reference voltage.
  - It controls the step size of the ADC conversion.
  - 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*

ADC major characteristics

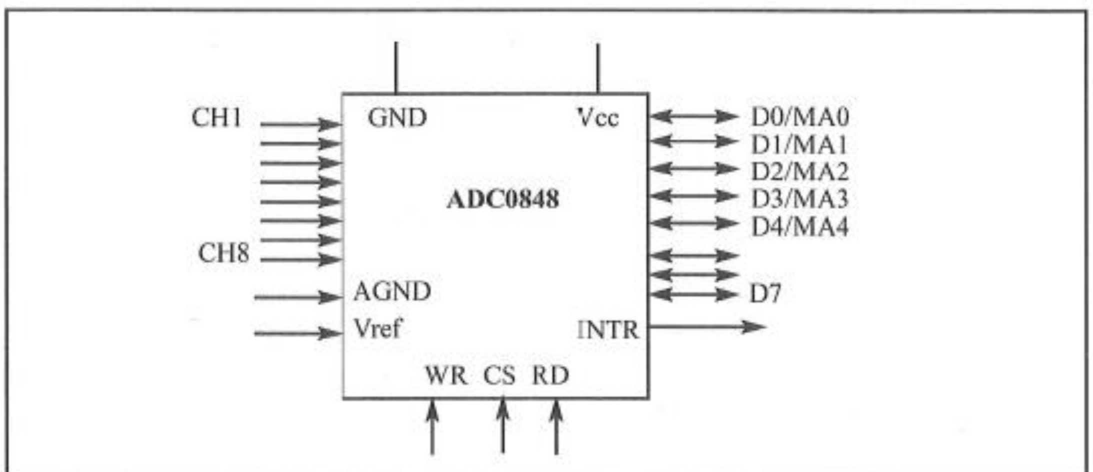
- D0 – D7 (D7 is MSB)
  - The digital data output
  - The result of conversion (rounding will be applied)
  - D = Vin / step size
  - Example: step size = 19.53 mV, V_{in} = 1.953 V, D = 100 = 64H.

Another Example

- For an 8-bit ADC, we have V_{ref} = 2.56 V. Calculate the digital data output if the analog input is 1.7 V.
- Step size = 2.56 V / 256 = 10 mV
- D = 1.7 V / 10 mV = 170 (decimal)
- Digital data output: 10101010 (D7 – D0)

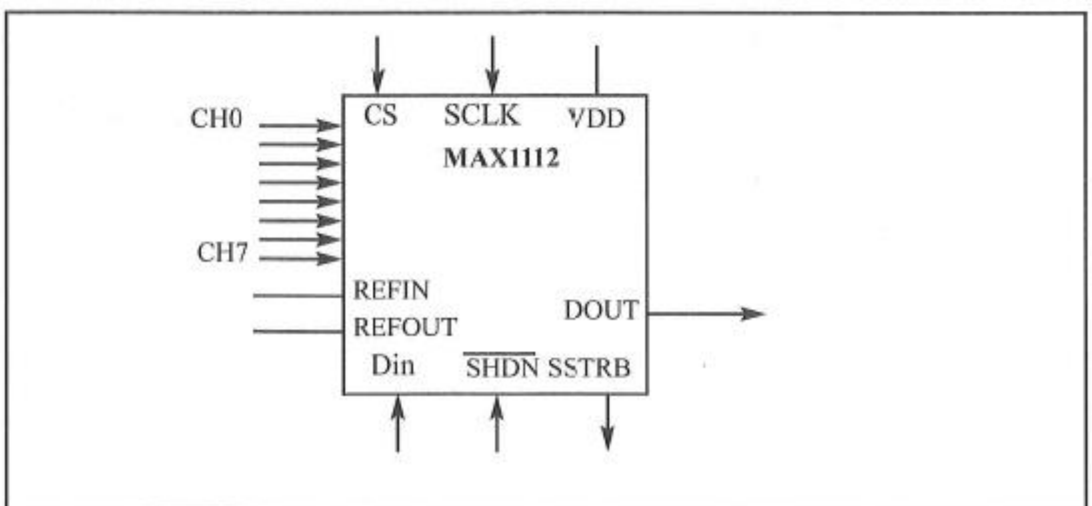
ADC major characteristics

- Parallel ADC
  - The D0-D7 data pins of the 8-bit ADC provide an 8-bit parallel data path between the ADC chip and the CPU.
  - To save pins, many 12- and 16-bits ADCs use pins D0-D7 to send out the upper and lower bytes of the binary data.



ADC major characteristics

- Serial ADC
  - One pin for data out. We need a parallel-in-serial-out shift register for sending data out the binary data one bit at a time.
  - In recent years, space is a critical issue. Thus serial ADCs are becoming widely used.



ADC major characteristics

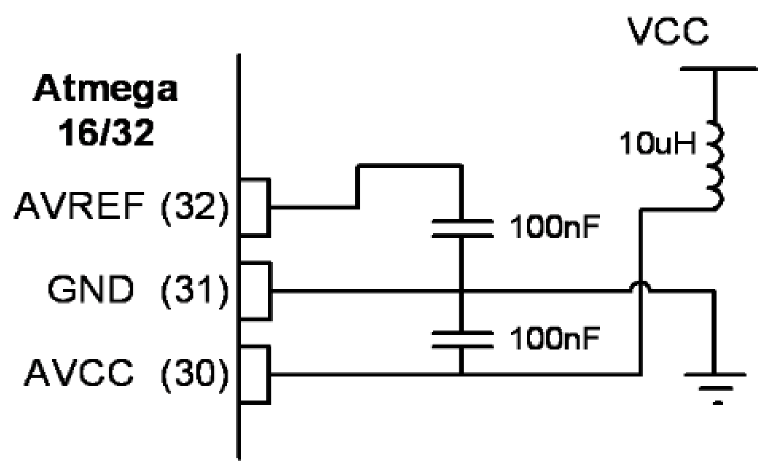
- Analog input channels
  - Many data acquisition applications need more than one ADC, thus multiple channels are provided in an ADC chip.
  - Monitor multiple quantities such as temperature, pressure, heat, and so on.
  - AVR: up to 16 channels

ADC major characteristics

- Start conversion (SC) and end-of-conversion (EOC) signals
  - When the SC signal is activated, the ADC starts converting the analog input to a digital output.
  - When the data conversion is complete, the EOC signal notifies the CPU that the converted data is ready to be picked up.

ADC in AVR

- Atmega 16/32 have an internal ADC.
  - 8 analogue input channels
  - 7 differential input channels
  - 2 differential input channels with 10x or 200x gain
  - 3 sources of V_{ref}
  - Internal 2.56 V V_{ref} generator



ADC in AVR

- ADMUX = ADC multiplexer selection register
- ADCH = High data
- ADCL = Low data
- ADCSRA = ADC Control and Status Register
- SFIOR = Special Function I/O Register

ADMUX

REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

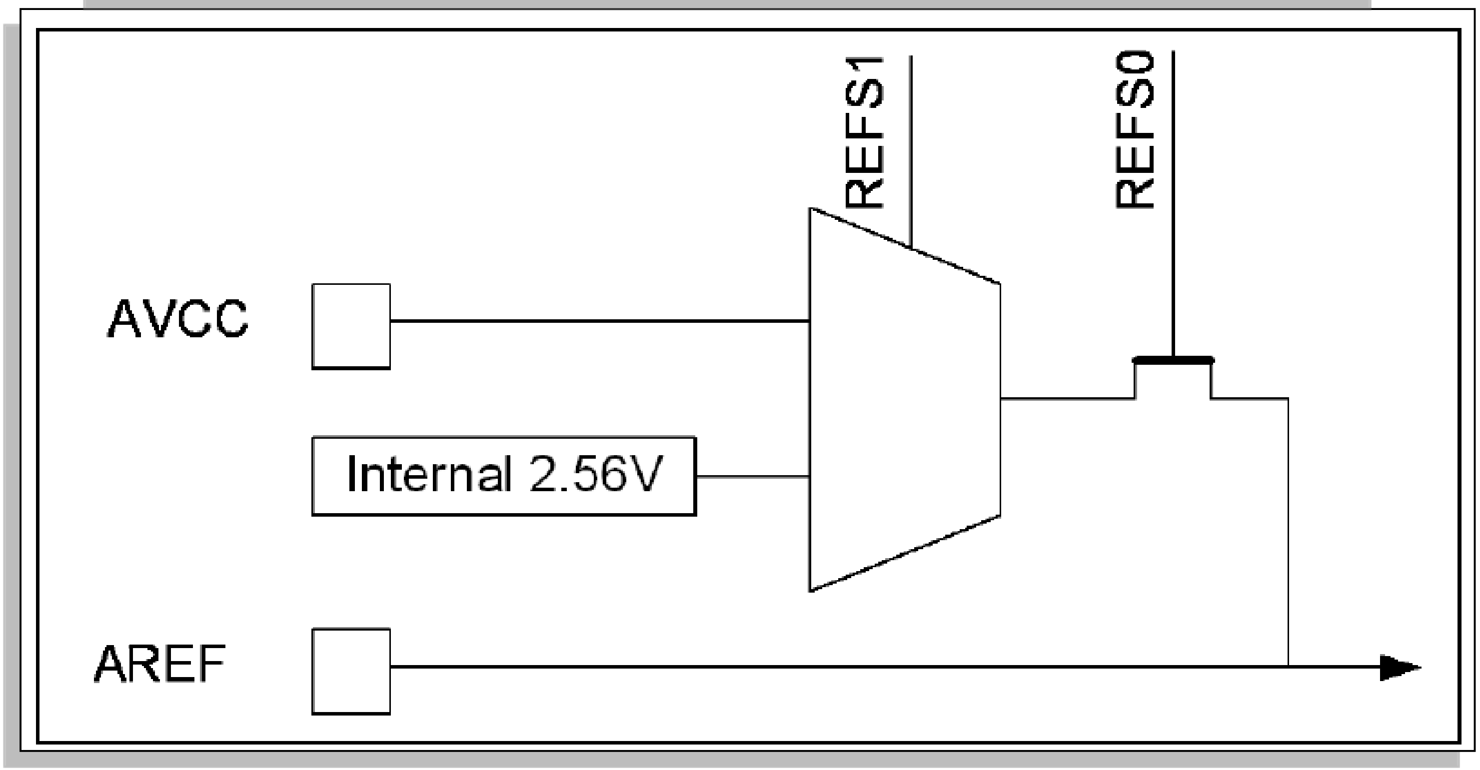
REFS1:0- Bit7:6 Reference Selection Bits
These bits select the voltage reference for the ADC.

ADLAR- Bit5 ADC Left Adjust Results
This bit dictate either the left bits or the right bits of the result registers ADCH:ADCL are used to store the result. If we write ADLAR to one the result will left adjusted otherwise the result is right adjusted.

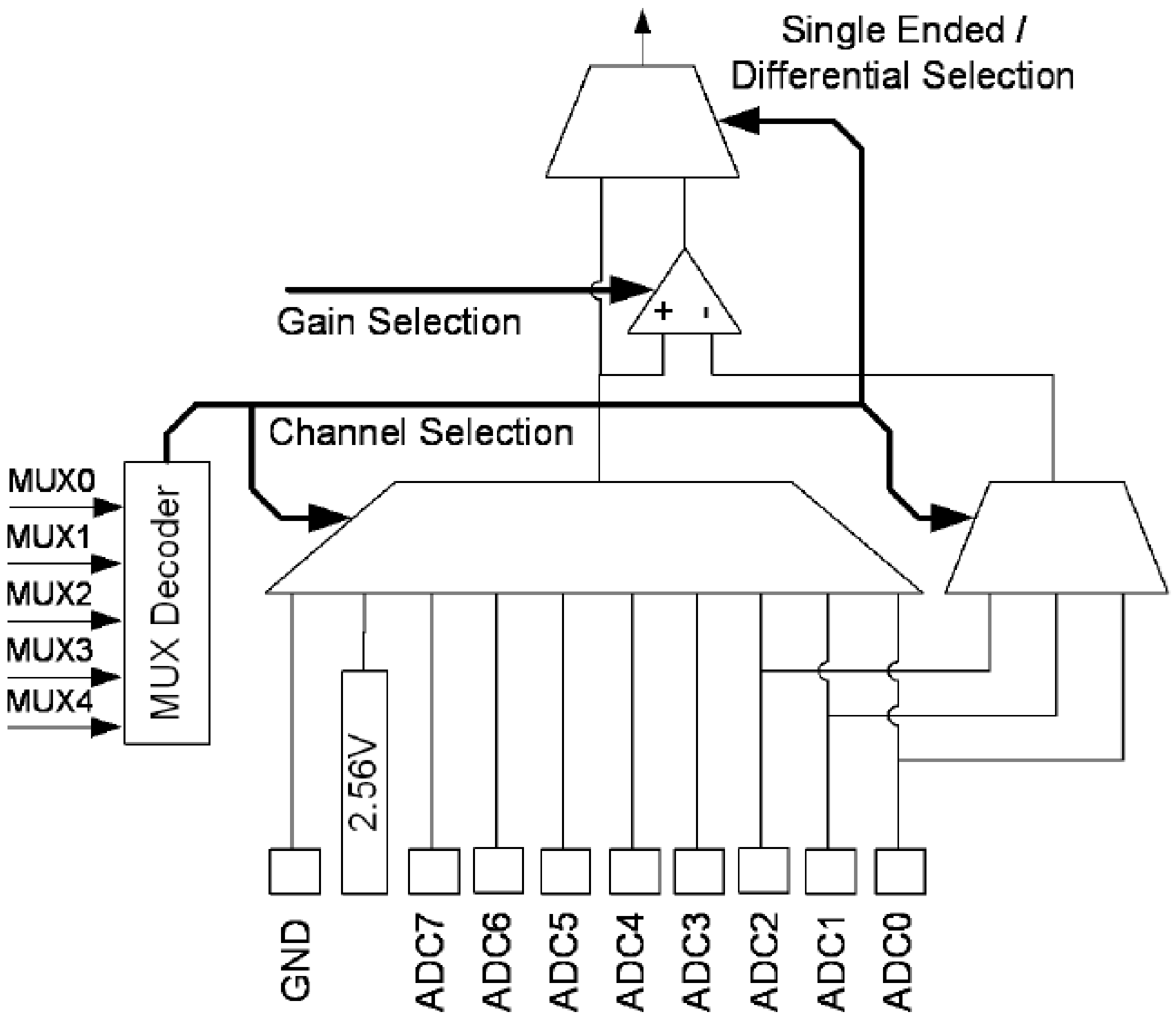
MUX4:0- Bit4:0 Analog Channel and gain selection bits
The value of these bits selects the gain for the differential channels and also which combination of analog inputs are connected to the
ADC.

ADMUX

REFS1	REFS0	V Reference
0	0	AREF pin
0	1	AVCC pin
1	0	Reserved
1	1	Internal 2.56 V



ADC input channel source



ADC input channel source

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

ADC input channel source

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

ADCH and ADCL

- ADCH:ADCL stores the results of conversion.
- The 10-bit result can be right or left justified:

ADLAR = 0

ADCH							
-	-	-	-	-	-	ADC9	ADC8

ADCL							
ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0

ADLAR =1

ADCH							
ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2

ADCL							
ADC1	ADC0	-	-	-	-	-	-

ADCSRA

ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

ADEN- Bit7 ADC Enable
This bit enables or disables the ADC. Writing this bit to one will enable and writing this bit to zero will disable the ADC even while a conversion is in progress.

ADSC- Bit6 ADC Start Conversion
To start each coversion you have to write this bit to one.

ADATE- Bit5 ADC Auto Trigger Enable
Auto Triggering of the ADC is enabled when you write this bit to one.

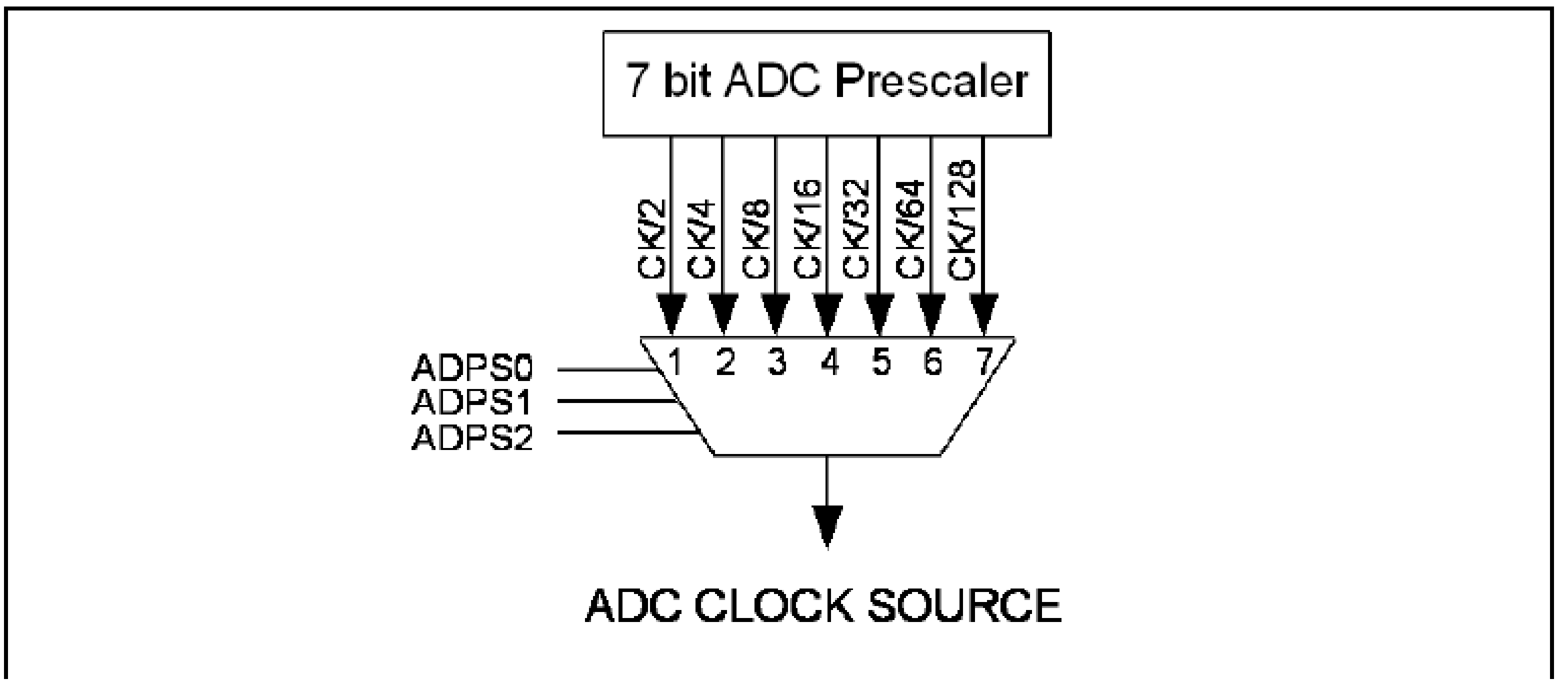
ADIF- Bit4 ADC Interrupt Flag
This bit is set when an ADC conversion completes and the Data Registers are updated

ADIE- Bit3 ADC Interrupt Enable
Writing this bit to one enables the ADC Conversion Complete Interrupt.

ADPS2:0- Bit2:0 ADC Prescaler Select Bits
These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

ADC Prescaler

- PreScaler Select Bits changes the clock frequency of ADC
- The frequency of ADC should not be more than 200 KHz.
- Conversion time is longer in the first conversion.



ADC Prescaler

Table 13-3: V_{ref} source selection table

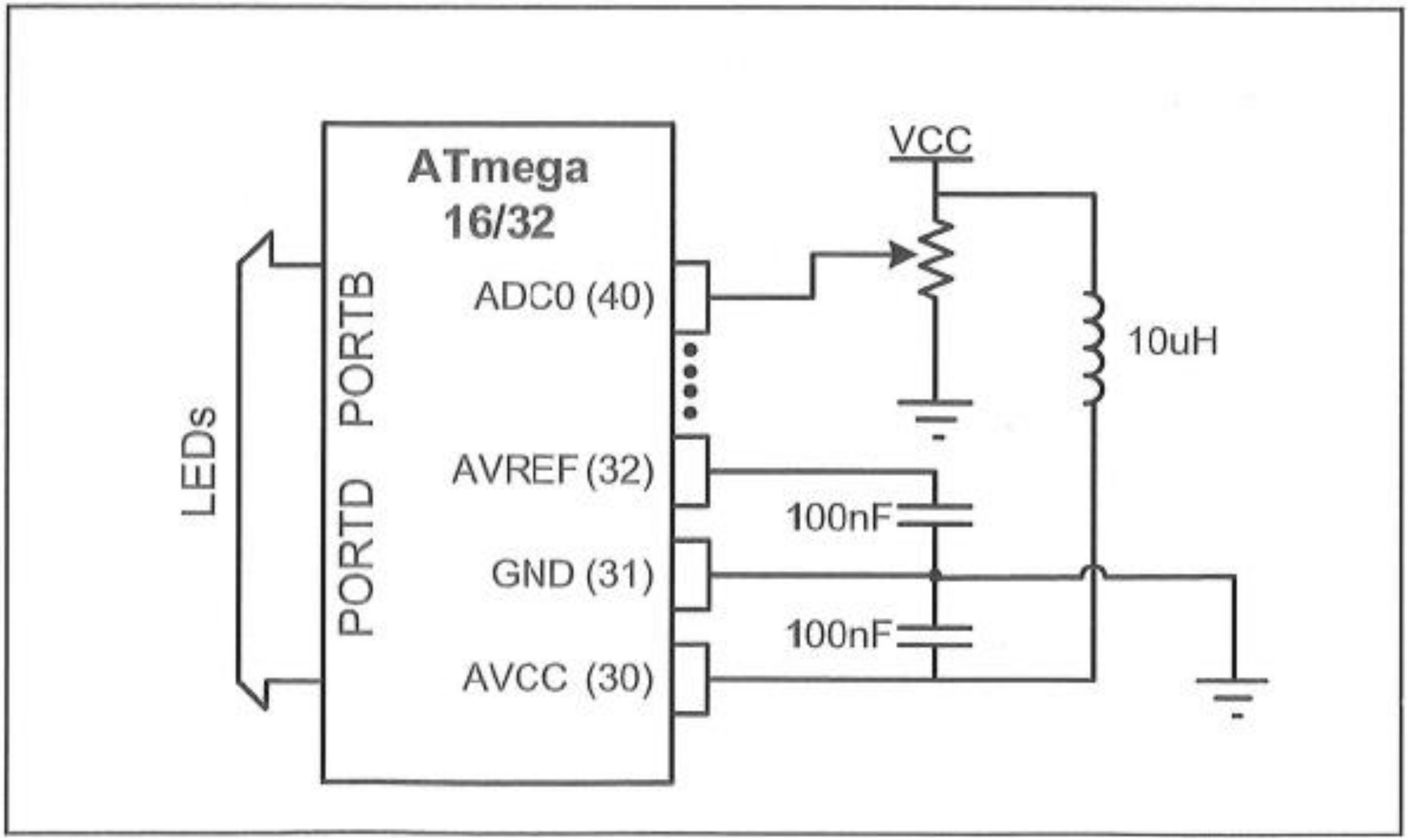
Condition	Sample and Hold Time (Cycles)	Conversion Time (Cycles)
First Conversion	14.5	25
Normal Conversion, Single ended	1.5	13
Normal Conversion, Differential	2	13.5
Auto trigger conversion	1.5 / 2.5	13/14

ADC programming

- Steps:
  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 A/C input channels. We use REFS0 amd REFS1 bits in ADMUX register to select voltage reference and MUX4:0 bits in ADMUX to select ADC input channel.
  5. Activate the start conversion bit by writing ADSC bit of ADCSRA to one.
  6. Wait for the conversion to be completed by polling the ADIF bit in ADCSRA register.
  7. After the ADIF bit has gone one read the ADCL and ADCH registers to get the digital data output. Note that you have to read ADCL before ADCH otherwise the result may 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 Vref source or input channel go back to step 4.

ADC programming

- Connection



ADC programming

- Polling (C programming)

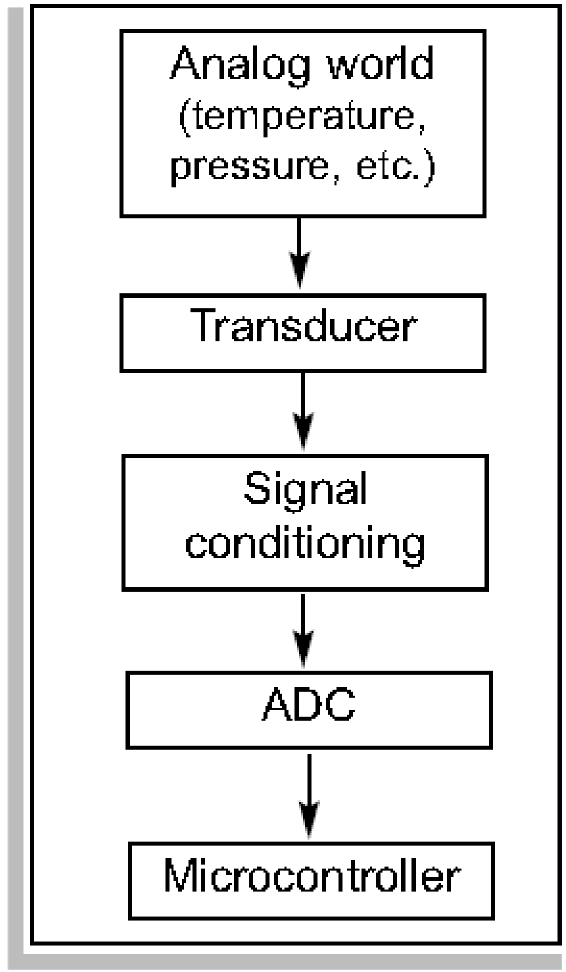
```
#include <avr/io.h>          //standard AVR header
int main (void)
{
  DDRB = 0xFF;               //make Port B an output
  DDRD = 0xFF;               //make Port D an output
  DDRA = 0;                  //make Port A an input for ADC input
  ADCSRA= 0x87;              //make ADC enable and select 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;
}
```

ADC programming

- Interrupt (C programming)

```c
#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;
}
```

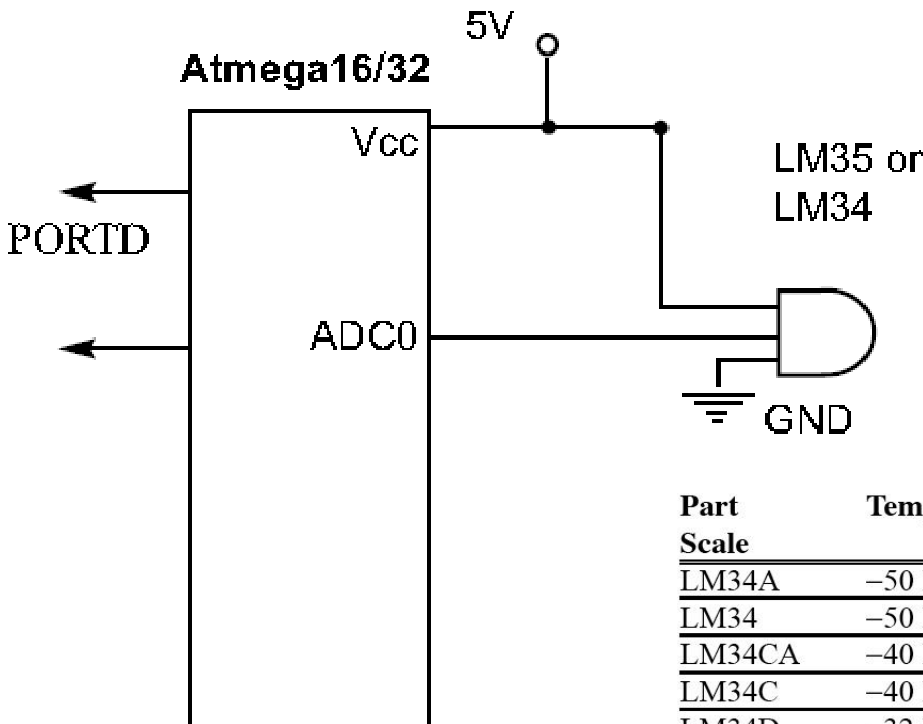
Signal conditioning



Signal conditioning

- The most common transducer produce an output in the form of voltage, current, charge, capacitance, and resistance.
- We need to convert these signals to voltage.
- This conversion is called signal conditioning.

Sensor Interfacing



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.

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.

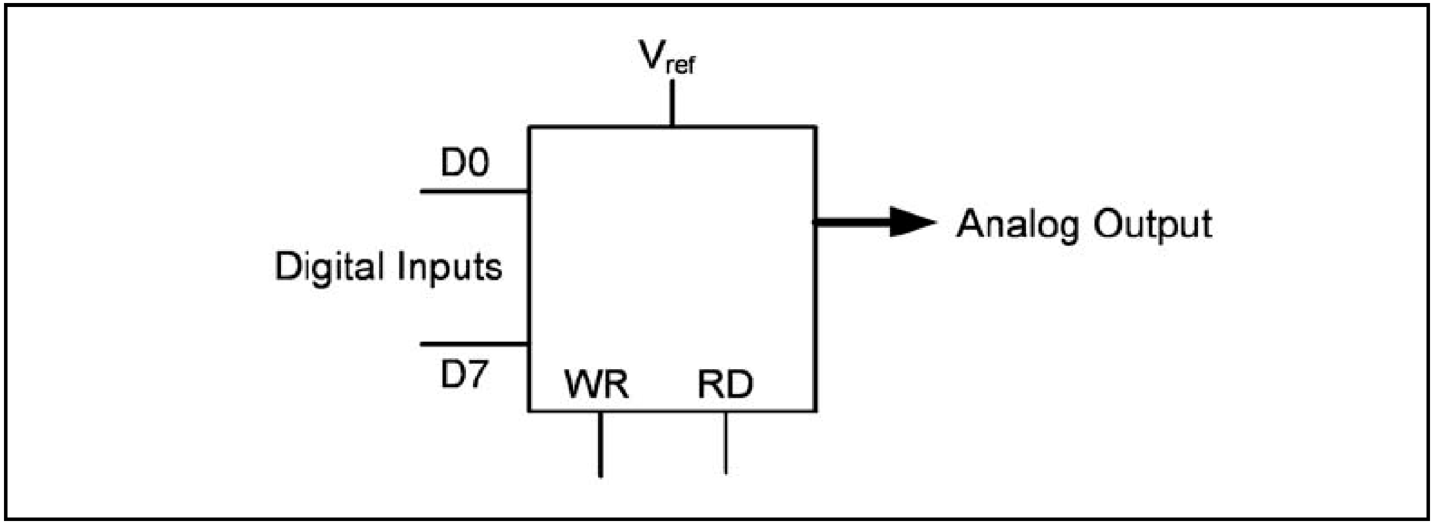
Sensor Interfacing

- 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

DAC

- DAC = Digital-to-Analog Converter
- How to connect an DAC to AVR?



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

DAC

- Convert digital pulses to analog signals.
- Resolution of DAC
  - Decided by the number of data bit inputs
  - Example:
    - 8-bit DAC provides 256 discrete voltage levels.
    - 12-bit DAC provides 4096 discrete voltage levels.

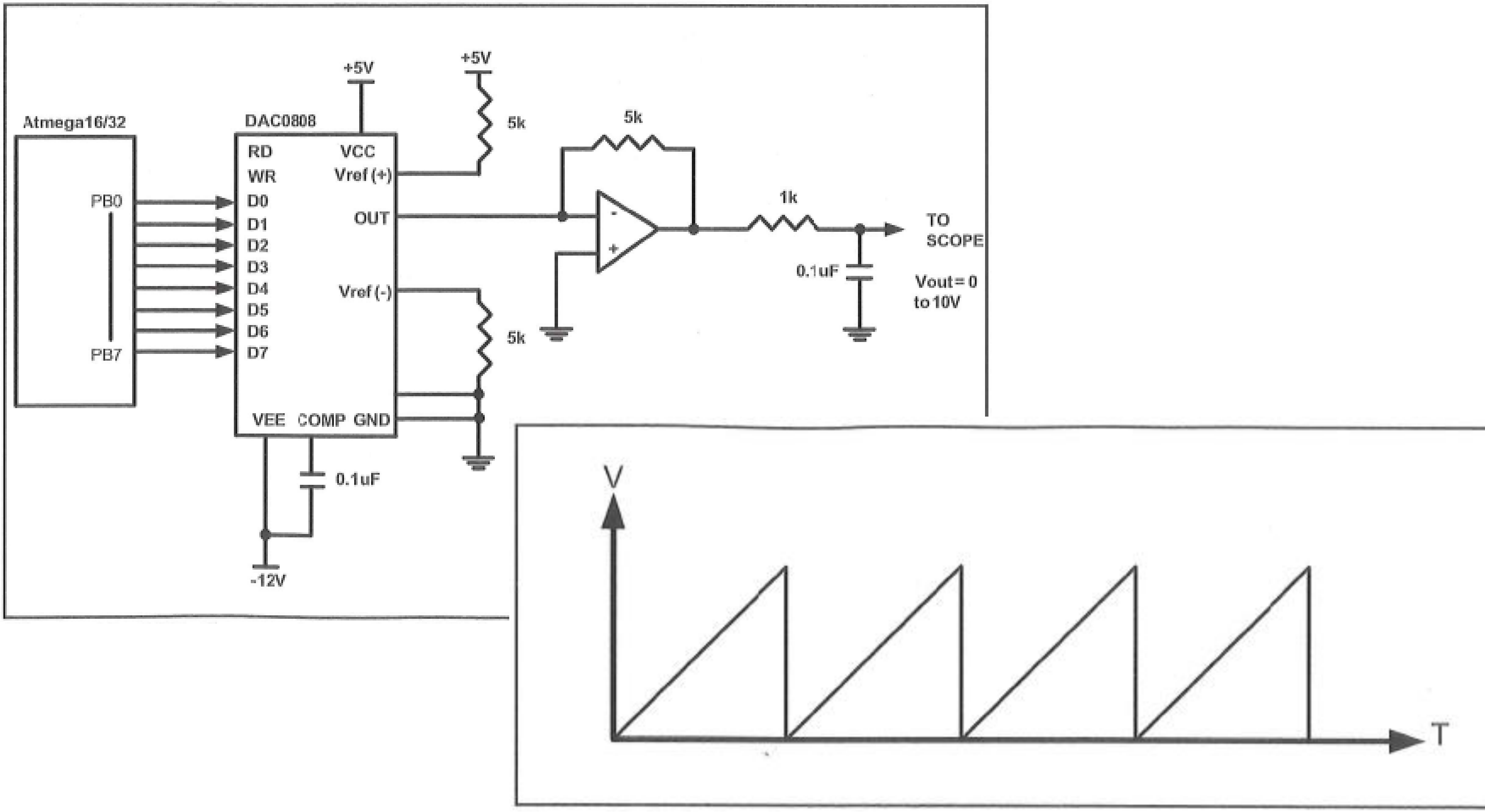
DAC

- Assuming that R = 5 kΩ and I_{ref} = 2 mA, calculate the V_{out} for the following binary inputs:
  - 10011001 (99H)
- Solution:
  - I_{out} = 2 mA (153 / 256) = 1.195 mA
  - V_{out} = 1.195 mA x 5kΩ = 5.975 V

Another Example

- Assuming that R = 5 kΩ and I_{ref} = 2 mA, calculate the V_{out} for the following binary inputs:
  - 11001000 (C8H)
- Solution:
  - I_{out} = 2 mA (200 / 256) = 1.562 mA
  - V_{out} = 1.562 mA x 5kΩ = 7.8125 V

DAC



Atmega16/32
PB0
PB7
DAC0808
RD
WR
D0
D1
D2
D3
D4
D5
D6
D7
VCC
Vref (+)
OUT
Vref (-)
VEE
COMP
GND
+5V
+5V
5k
5k
5k
1k
0.1uF
0.1uF
-12V
TO
SCOPE
Vout= 0
to 10V
V
T

DAC

- C

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

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

Reference Readings

- Chapter 14 – *The AVR Microcontroller and Embedded Systems : Using Assembly and C*, M. A. Mazidi, S. Naimi, and S. Naimi, Pearson, 2014.

End