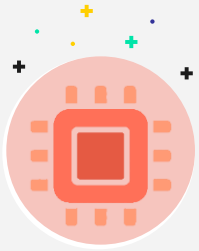
1404/02



Microcontrollers

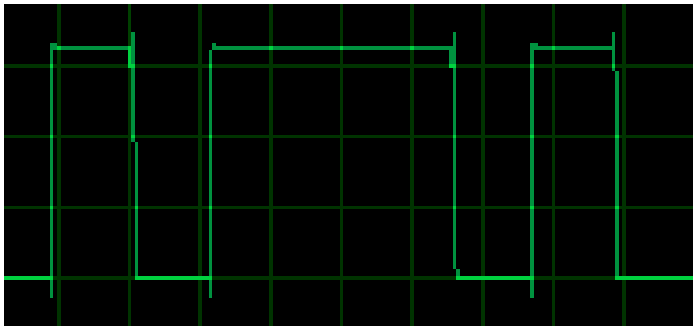
Lecture 9:

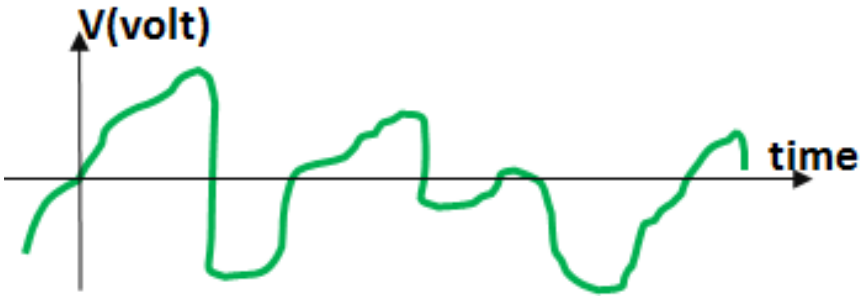
Analog to Digital Converter (ADC)

By: M.Razeghizadeh

Start now!

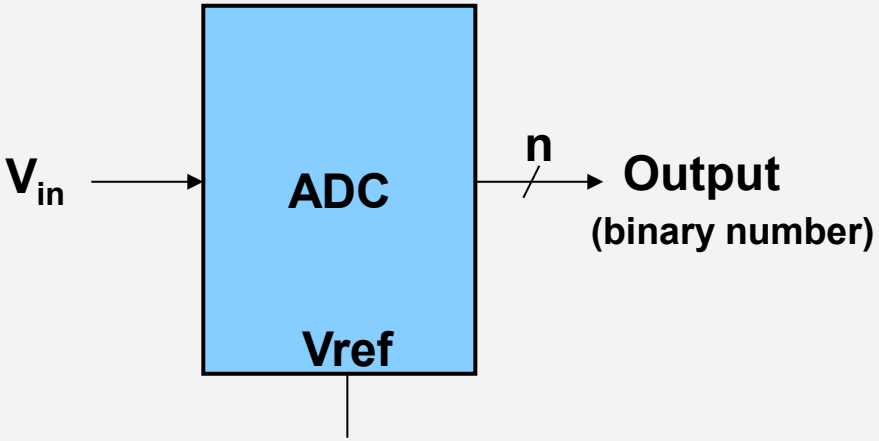
Analog vs. Digital Signals

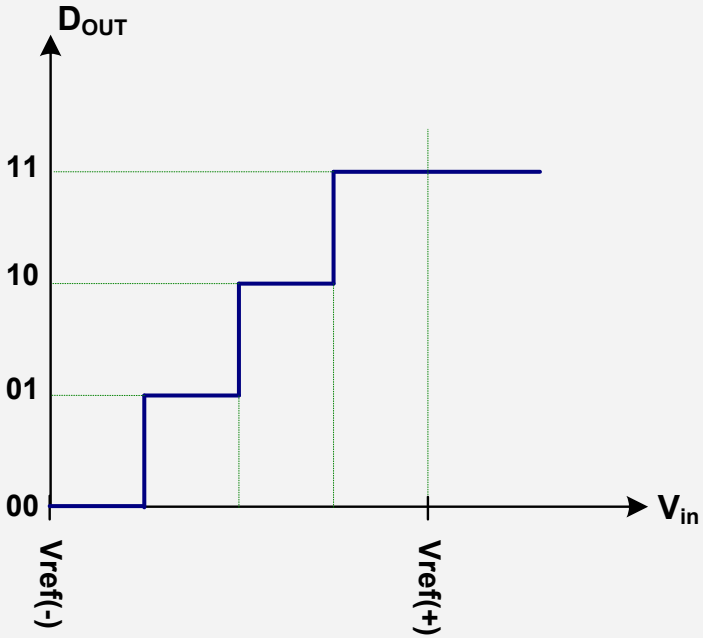


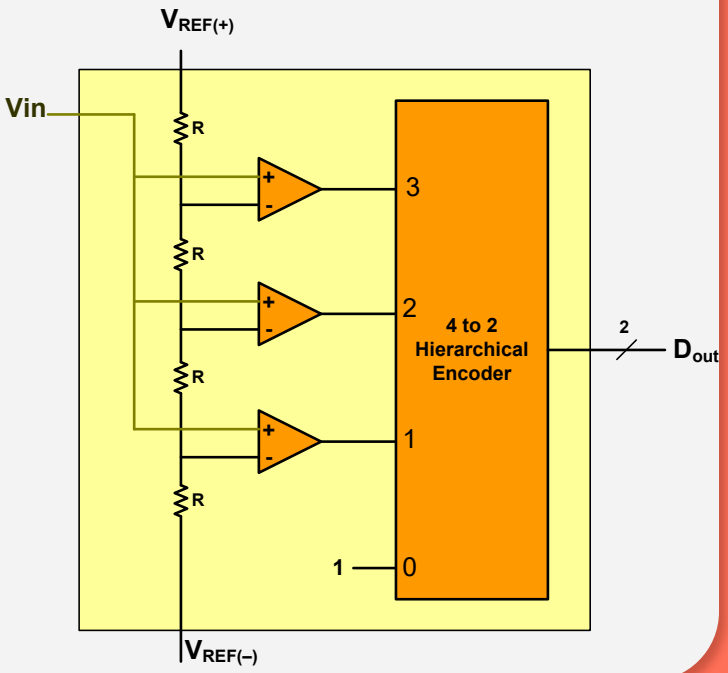


$$stepSize = \frac{Vref}{numofsteps}$$

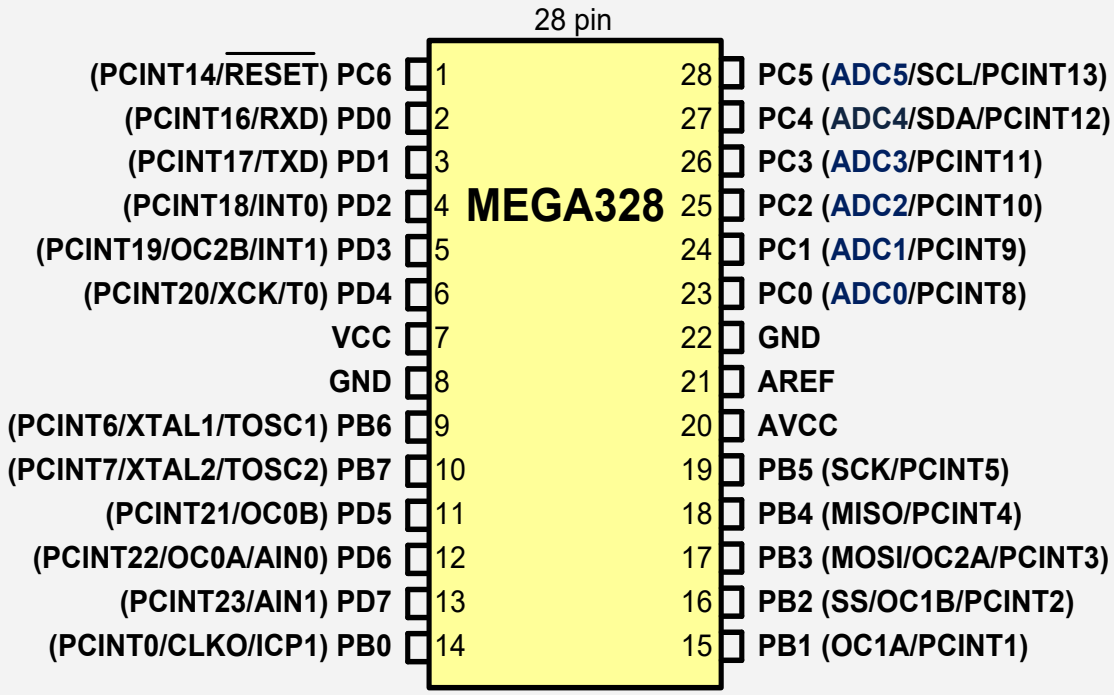
$$output = \left\lfloor \frac{Vin}{stepSize} \right\rfloor$$



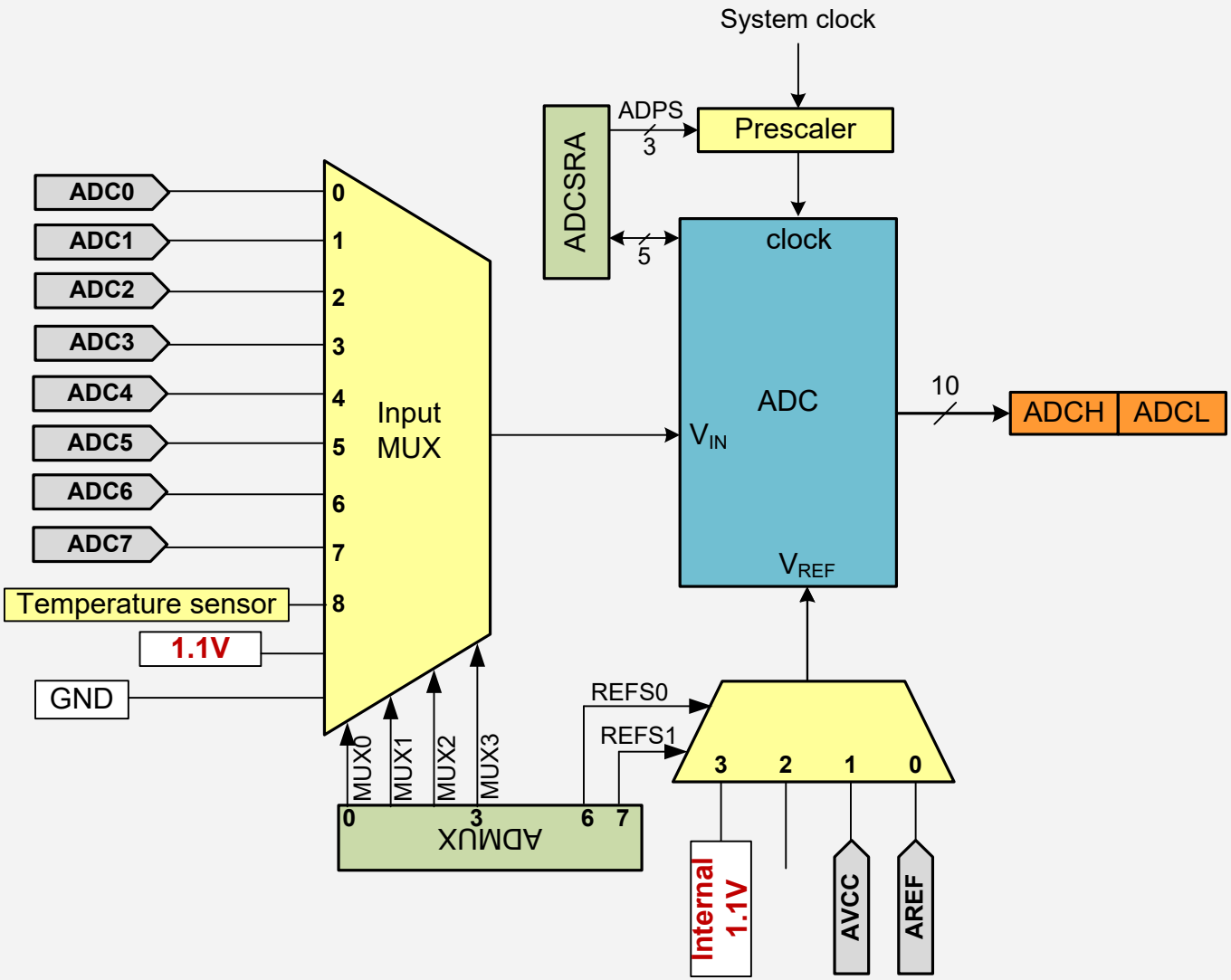




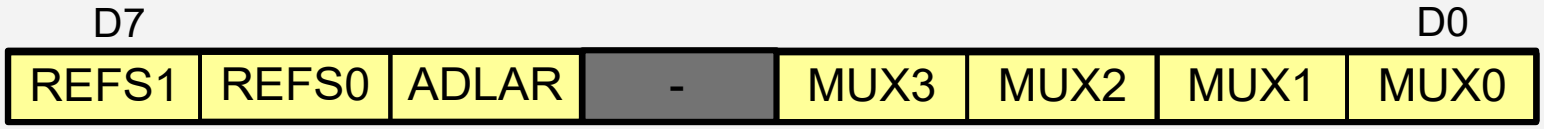
ADC in AVR


28 pin
MEGA328
(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

ADC in AVR


System clock
ADPS
3
Prescaler
ADCSRA
5
clock
ADC
V_{IN}
V_{REF}
10
ADCH
ADCL
ADC0
ADC1
ADC2
ADC3
ADC4
ADC5
ADC6
ADC7
Temperature sensor
1.1V
GND
Input
MUX
0
1
2
3
4
5
6
7
8
MUX0
MUX1
MUX2
MUX3
REFS0
REFS1
3
2
1
0
ADMUX
0
3
6
7
Internal
1.1V
AVCC
AREF

ADMUX

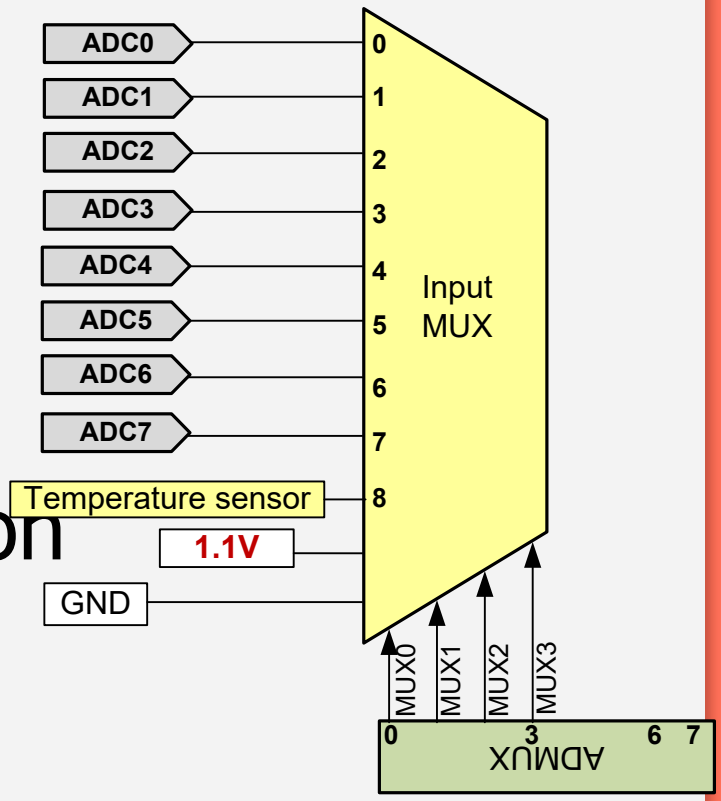


- MUX0-MUX1: input select
- ADLAR:
  - 0: right adjust the result
  - 1: left adjust the result
- REFS1-REFS0: Vref selection

ADMUX

D7							D0
REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0

- MUX0-MUX1: input select
- ADLAR:
  - 0: right adjust the result
  - 1: left adjust the result
- REFS1-REFS0: Vref selection

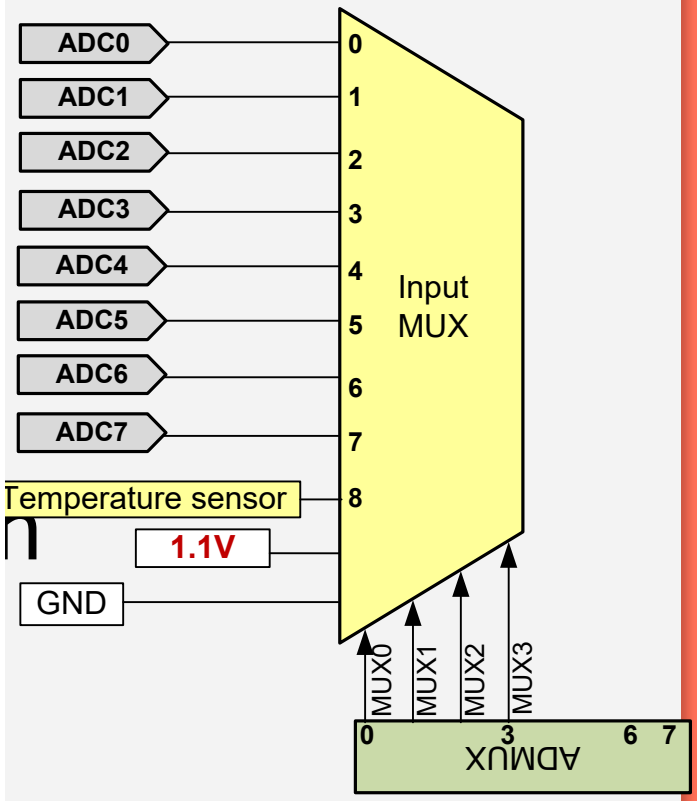


ADMUX

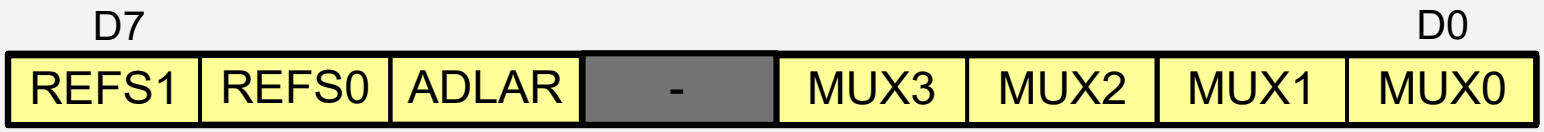
D7							D0
REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0

Table 24-4. Input Channel Selections

MUX3...0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8[1]
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

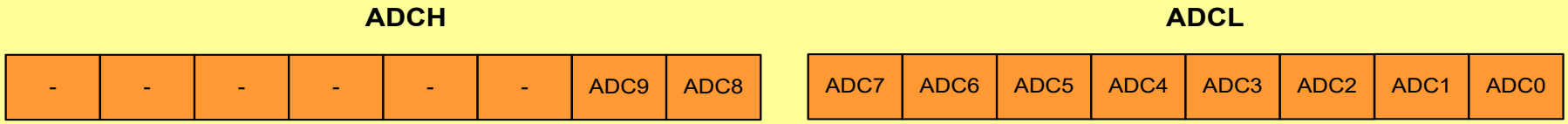


ADMUX

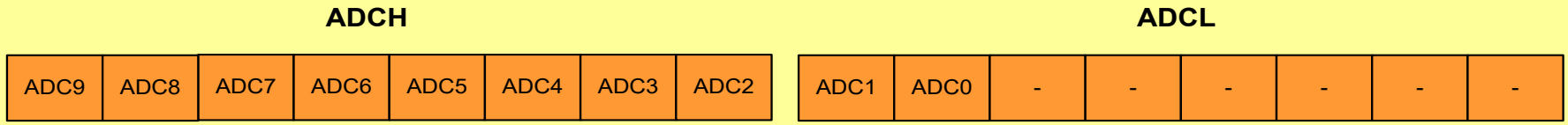


- MUX0-MUX1: input select
- ADLAR:

ADLAR = 0



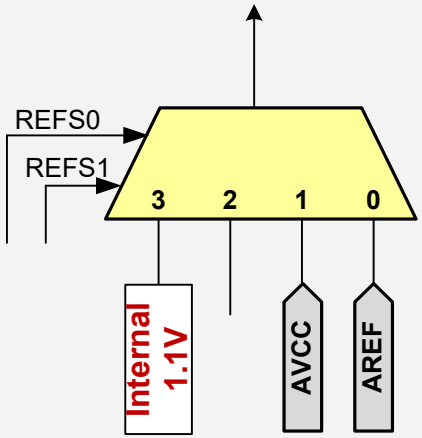
ADLAR =1



ADMUX

D7							D0
REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0

- MUX0-MUX1: input select
- ADLAR:
  - 0: right adjust the result
  - 1: left adjust the result
- REFS1-REFS0: Vref selection



ADMUX

D7							D0
REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0

- MUX0-MUX1: input select
- ADLAR:
  - 0: right adjust the result
  - 1: left adjust the result
- REFS1-REFS0: Vref selection

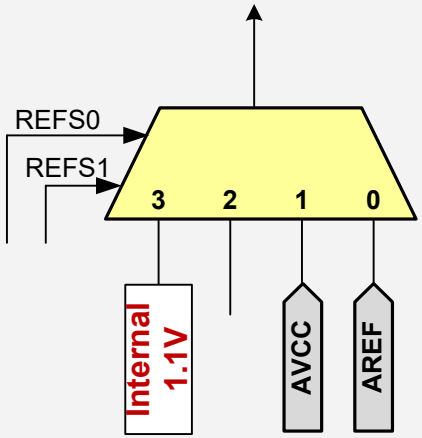


Table 24-3. Voltage Reference Selections for ADC

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

ADCSA

D7							D0
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.

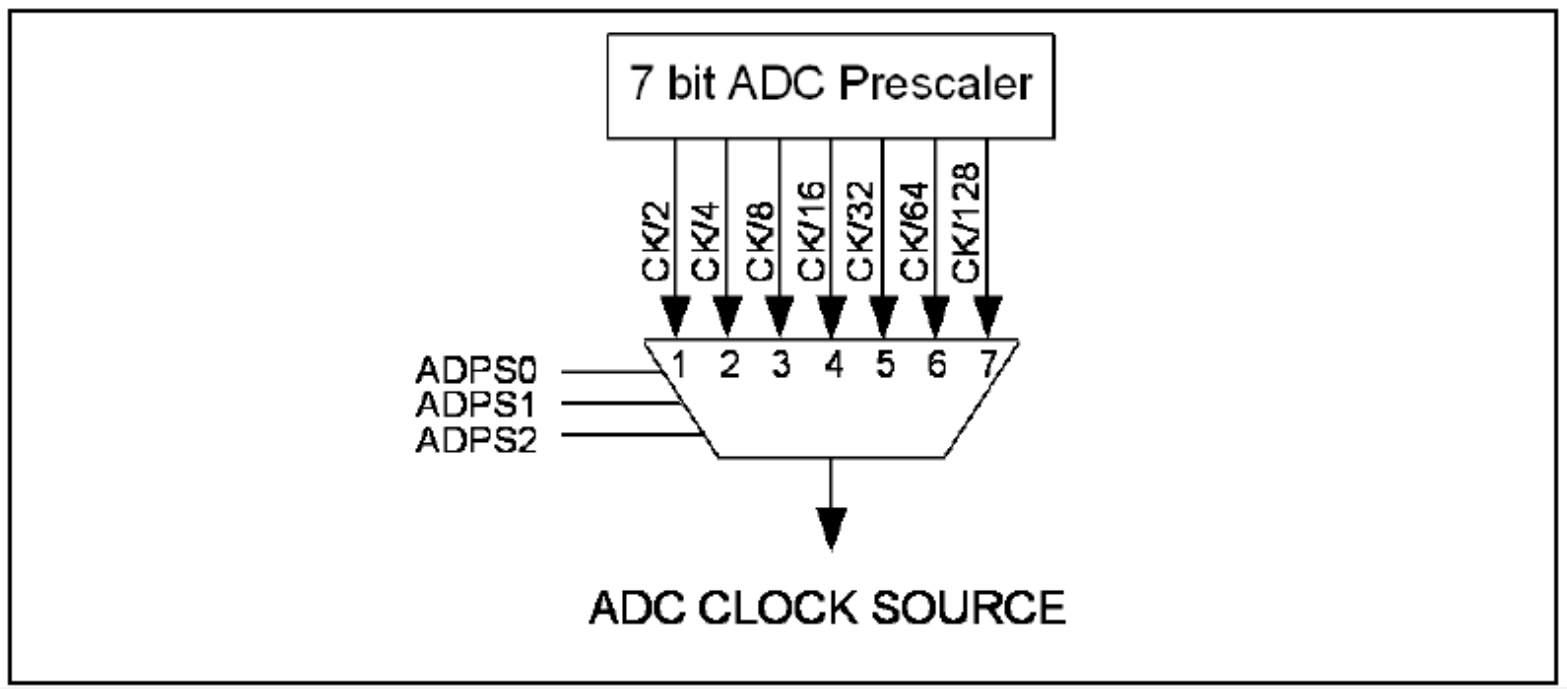
ADIF- Bit4 ADC Interrupt Flag
This bit is set when an ADC conversion completes and the Data Registers are updated

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 Bits let us change the clock frequency of ADC
- The frequency of ADC should not be more than 200 KHz
- Conversion time is longer in the first conversion



ADCSA

D7							D0
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

ADEN- Bit7 ADC Enable

This b… this bit to zero

ADSC

To sta…

ADAT

Auto …

ADIF

This b… dated

Table 24-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

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.

ADCSA

D7 D0

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

To sta

ADAT

Auto T

ADIF

This b dated

ADIE

Writing this bit to one enables the ADC Conversion Complete Interrupt.

AD

The

clo

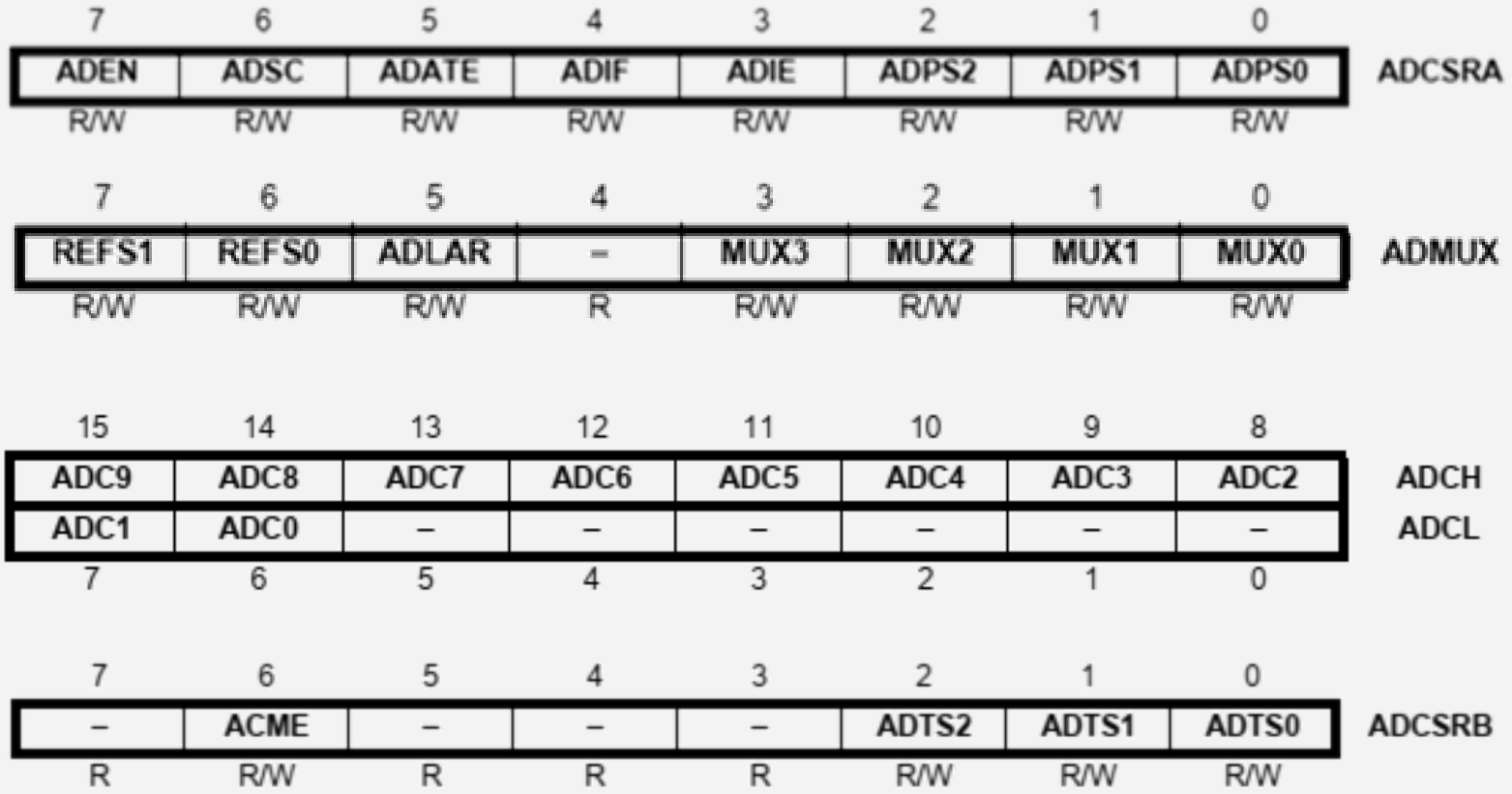
Table 24-6. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match A
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

ADCSRB

7	6	5	4	3	2	1	0
–	ACME	–	–	–	ADTS2	ADTS1	ADTS0
R	R/W	R	R	R	R/W	R/W	R/W
0	0	0	0	0	0	0	0

ADCSA


7 6 5 4 3 2 1 0
ADEN ADSC ADATE ADIF ADIE ADPS2 ADPS1 ADPS0
R/W R/W R/W R/W R/W R/W R/W R/W
ADCSRA
7 6 5 4 3 2 1 0
REFS1 REFS0 ADLAR – MUX3 MUX2 MUX1 MUX0
R/W R/W R/W R R/W R/W R/W R/W
ADMUX
15 14 13 12 11 10 9 8
ADC9 ADC8 ADC7 ADC6 ADC5 ADC4 ADC3 ADC2
ADC1 ADC0 – – – – – –
7 6 5 4 3 2 1 0
ADCH
ADCL
7 6 5 4 3 2 1 0
– ACME – – – ADTS2 ADTS1 ADTS0
R R/W R R R R/W R/W R/W
ADCSRB

Steps in programming ADC

1. Make the pin for the selected ADC channel an input pin.
2. Turn on the ADC module
3. Select the conversion speed
4. Select voltage reference and ADC input channels.
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 registers to get the digital data output.
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.

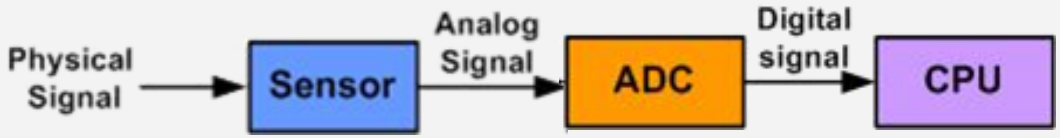
A program with ADC

- This program gets data from channel 0 (ADC0) of ;ADC and displays the result on Port B and Port D.

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>

int main (void)
{
 DDRB = 0xFF;//make Port B an output
 DDRD = 0xFF; //make Port D an output

 ADCSRA= 0x87;//make ADC enable and select ck/128
 ADMUX= 0xC8;//1.1V Vref, temp. sensor, right-justified

 while(1)
{
  ADCSRA |= (1<<ADSC);//start conversion

  while((ADCSRA&(1<<ADIF))==0);//wait for conversion to finish

  ADCSRA |= (1<<ADIF);

  PORTD = ADCL;//give the low byte to PORTD
  PORTB = ADCH;//give the high byte to PORTB

  _delay_ms(100);
 }
}
```

Sensors

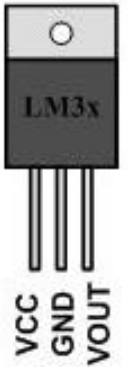
- Sensor: Converts a physical signal (e.g. light, temperature, humidity, etc.) to an electrical signal (e.g. resistance, voltage, current, capacitance, etc)

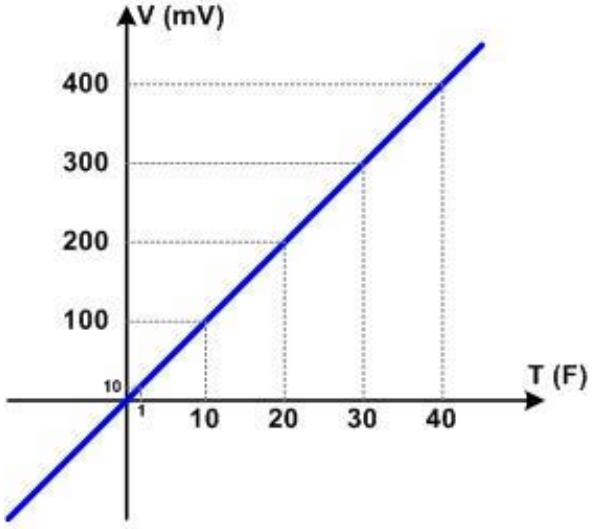


- LM35 and LM34:
  - convert temp. to voltage
  - 10mV for each degree

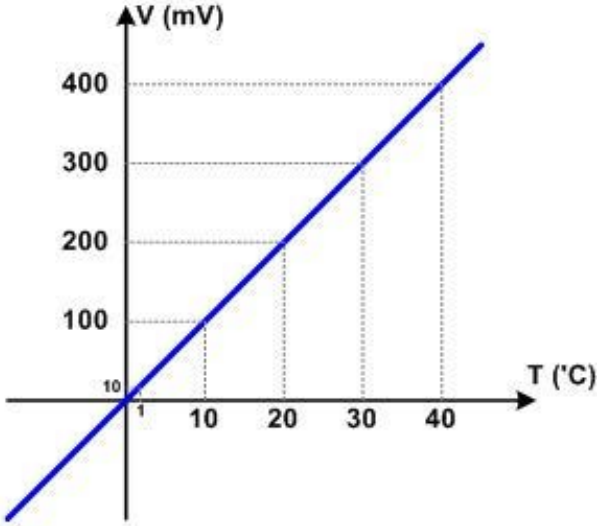
Bottom view
TO92 Package

Top view
TO220 Package





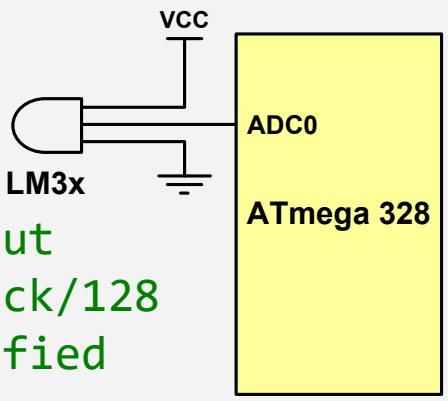
(a) LM34



(b) LM35

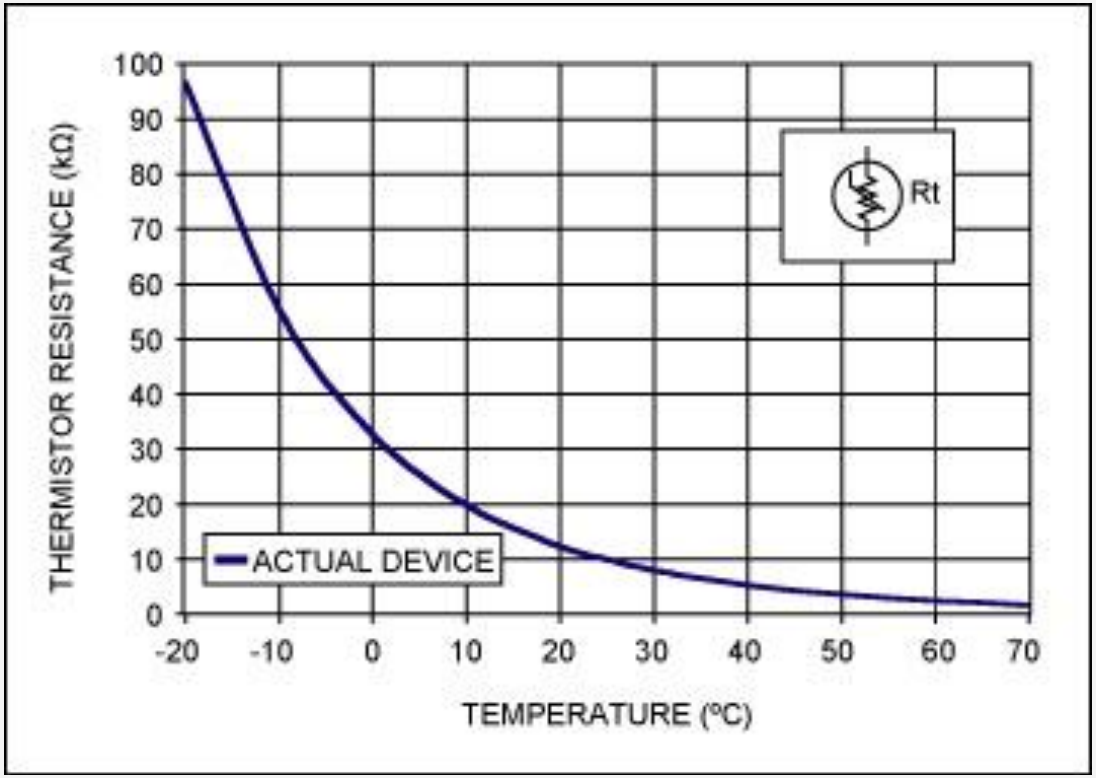
Using LM35

```
//this program reads the sensor and displays it on Port D
#include <avr/io.h> //standard AVR header
int main (void)
{
 DDRB = 0xFF; //make Port B an output
 DDRC = 0;//make Port C an input for ADC input
 ADCSRA = 0x87;//make ADC enable and select ck/128
 ADMUX = 0xC0;//1.1V Vref, ADC0, right-justified

 while (1){
  ADCSRA |= (1<<ADSC);//start conversion
  while((ADCSRA&(1<<ADIF))==0); //wait for end of conversion
  ADCSRA |= (1<<ADIF); //clear the ADIF flag
  PORTB = (ADCL|(ADCH<<8))*10/93;//PORTB = adc value/9.3
 }
}
```



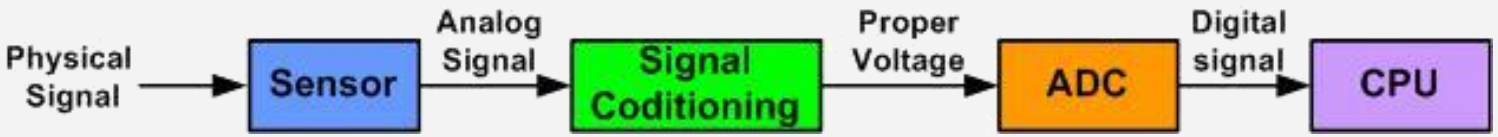
Thermistor (a temperature sensor)

- Converts temperature to resistance
- It is not linear

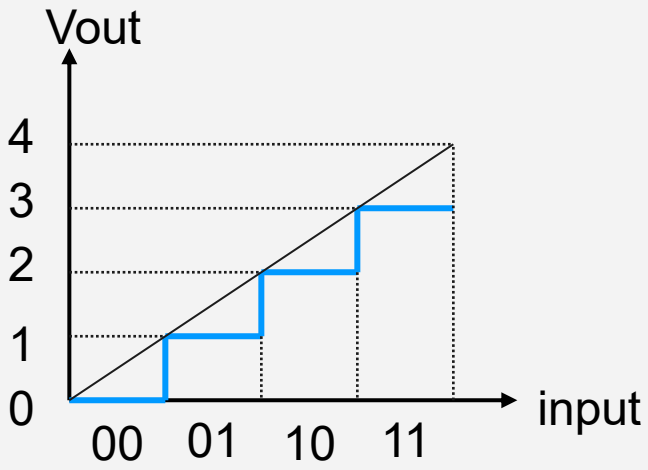


Signal conditioning

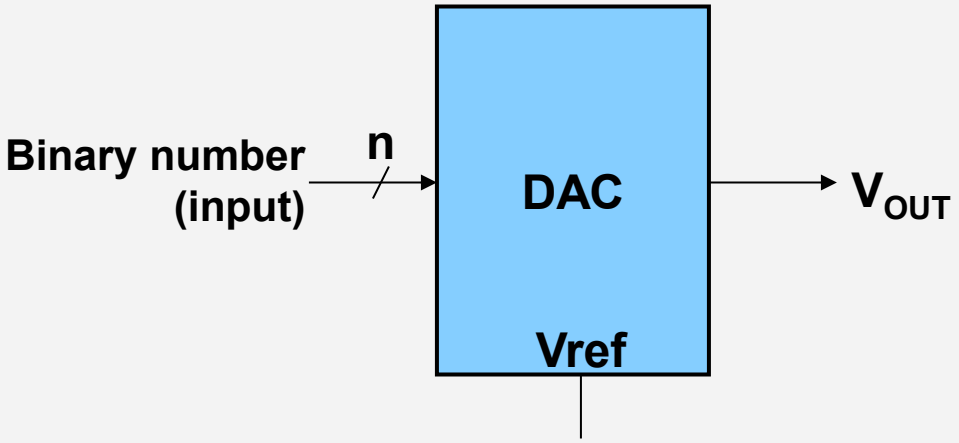
- The output of some sensors (e.g. PT100) is in form of resistance
- Some humidity sensor provide the result in form of Capacitance
- We need to convert these signals to voltage, however, in order to send input to an ADC. This conversion is called signal conditioning.



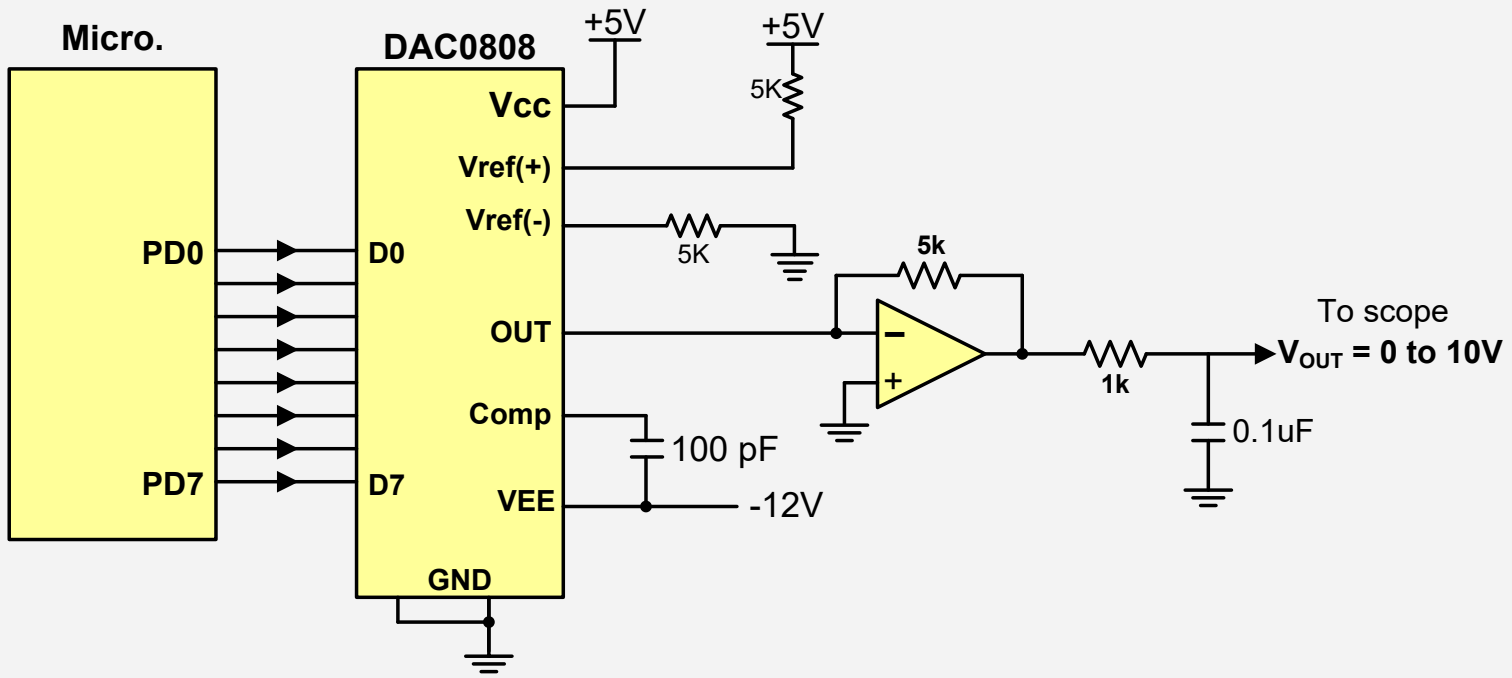
$$\text{Step size} = \frac{V_{REF}}{\text{Num of steps}}$$



$V_{OUT} = \text{num} \times \text{step size}$

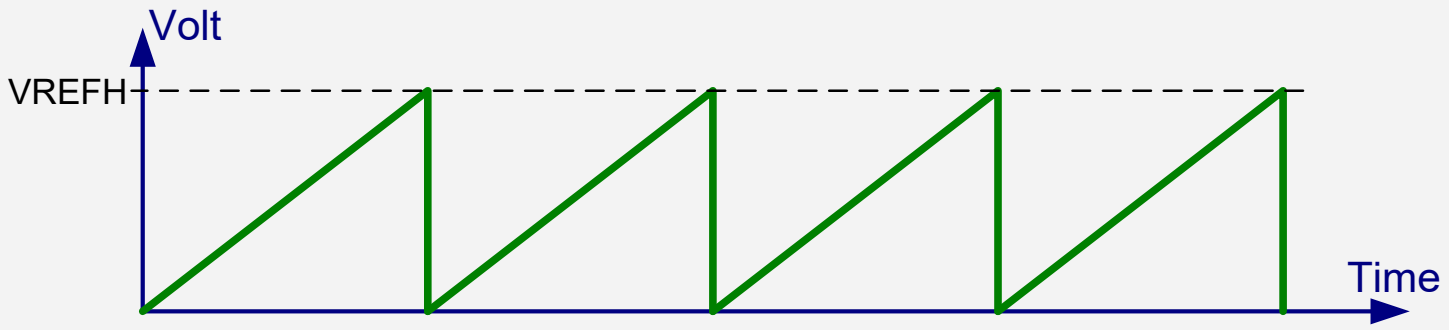


Connecting a DAC to the microcontroller



Generating a saw-tooth wave using DAC

```
#include <avr/io.h>

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



Thanks!

Do you have any questions?

razeghizade@gmail.com

www.pudica.ir

CREADITS: This presentation was created by M.Razeghizadeh
Please keep this slide for attribution