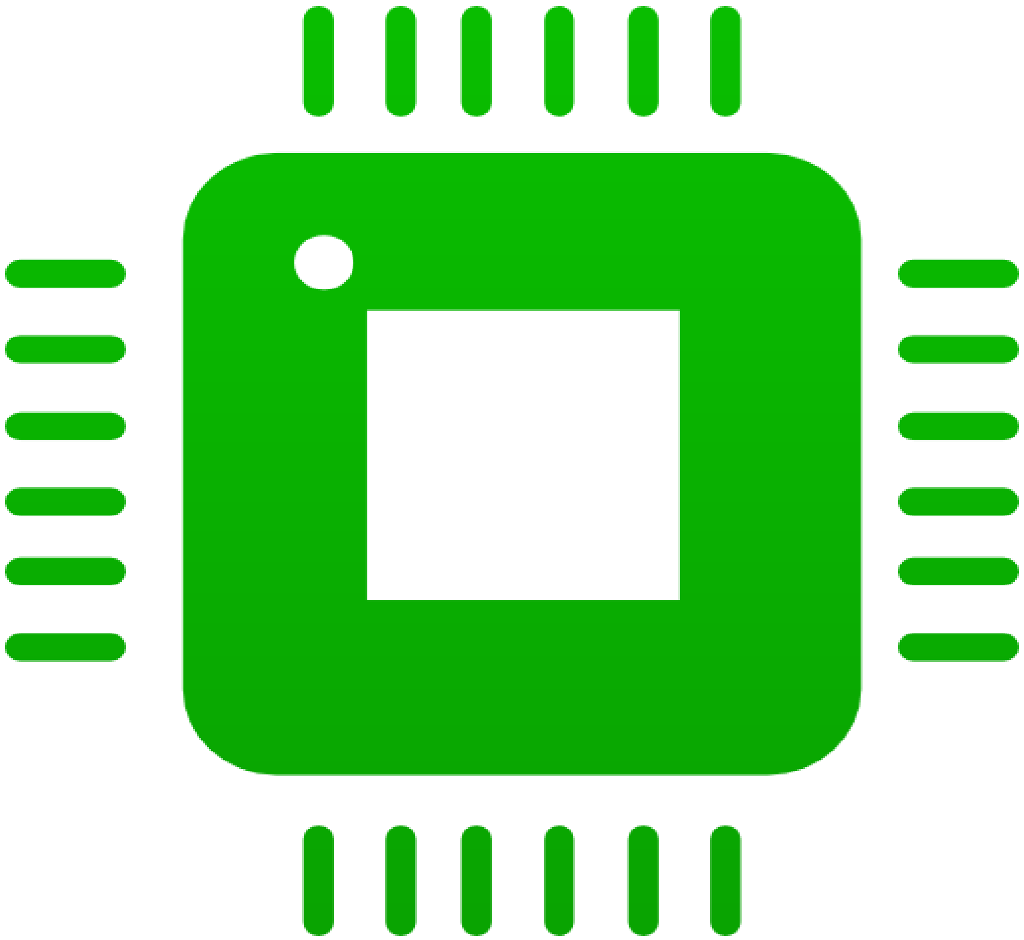
Fab Lab Ismailia represent :
Embedded Systems Workshop
by : Mohammed hemed



5- ADC
Analog to Digital Converter

- uC use binary (discrete) values , but in physical world everything is analog(continuous) .
- Temperature – humidity – velocity – pressure (wind or liquid) are examples of physical quantities we deal with every day .
- A physical quantity is converted to electrical (voltage , current) signals using a device called a transducer or (sensor).
- To make uC interact with the physical world we need analog to digital converter to translate the analog signals to digital numbers to make uC able to read and process them

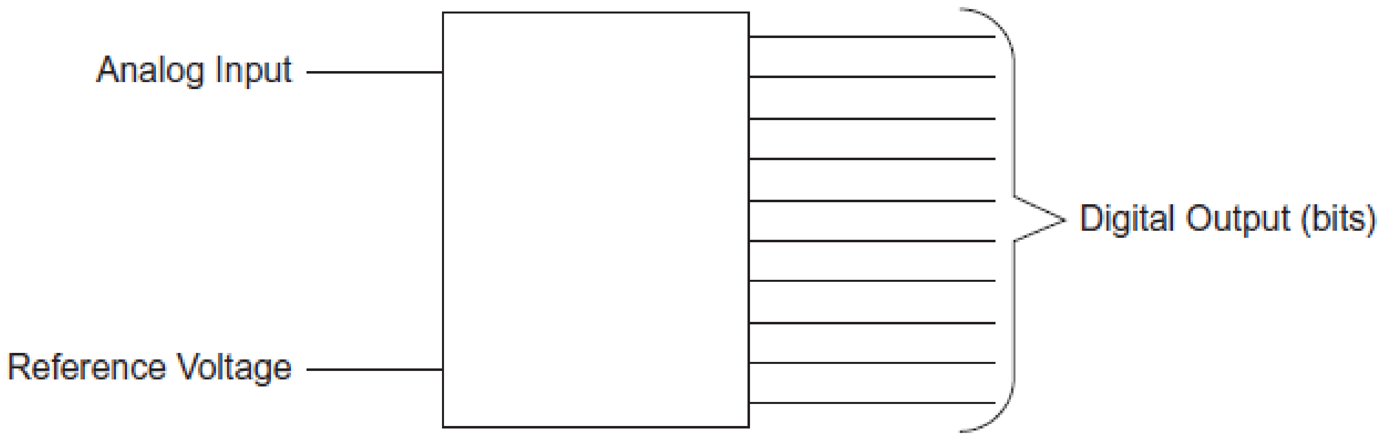


Figure 6.1 Simple ADC.

Characteristics of ADC

- Resolution : ADC has n-bit resolution , where n can be 8 , 10 , 12 , 16 or even 24 bits , we can't change the resolution of ADC as it decided at the time of its design .
- Resolution = VLSB = Vref / 2^n-bit
- Vref : an input voltage used for the reference voltage
- Step Size : The smallest change could ADC sense

  step size = Vref / resoulution

  ex1 : if we have 8-bit ADC , and Vref = 5 so step size = 5/256 =19.53 mV

  ex2 : if we have 10-bit ADC , and Vref = 5 so step size = 5/1024 = 4.88 mV

the higher resolution provide smaller step size .

- Conversion time : The time it takes ADC to convert analog input to a digital (binary) number it depend on the ADC clock – technology used in fabrication of ADC chip , the method used in conversion A2D

Resolution - step size - Vref

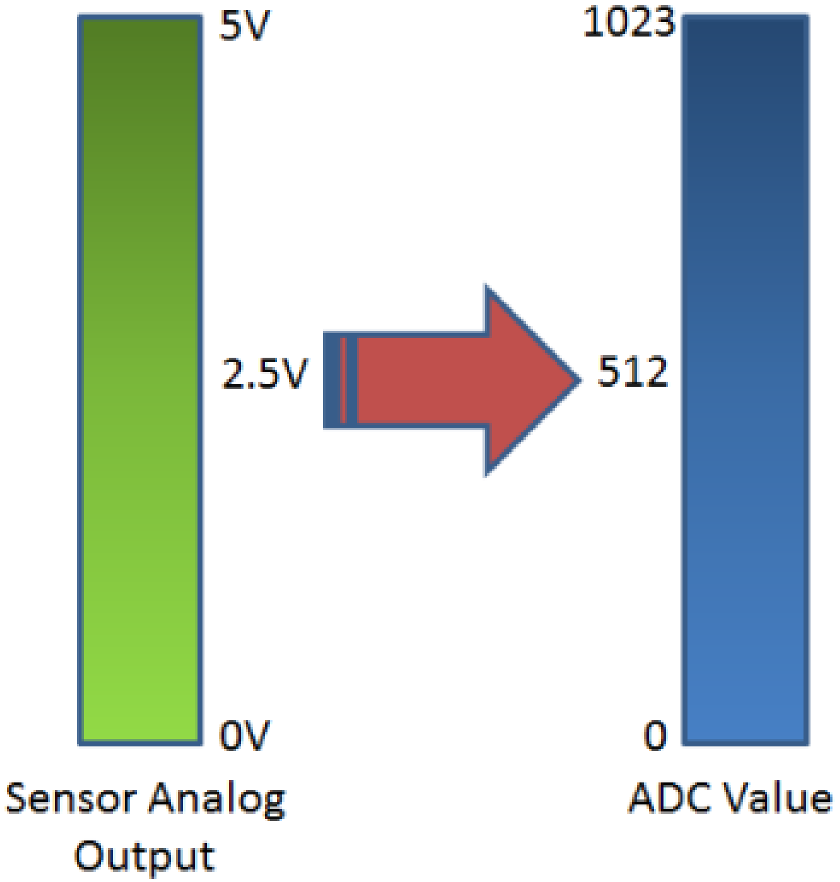
Vref (V)	Vin Range (V)	Step Size (mV)
5.00	0 to 5	5 / 1024 = 4.88
4.96	0 to 4.096	4.096 / 1024 = 4
3.00	0 to 3	3 / 1024 = 2.93
2.56	0 to 2.56	2.56 / 1024 = 2.5
2.00	0 to 2	2 / 1024 = 2
1.28	0 to 1.28	1.28 / 1024 = 1.25
1.024	0 to 1.024	1.024 / 1024 = 1
Note: In a 10-bit ADC, step size is V_{ref}/1024		

Table 7-3: Vref Relation to Vin Range for an 10-bit ADC

V_{ref} (V)	Vin Range (V)	Step Size (mV)
5.00	0 to 5	5 / 256 = 19.53
4.00	0 to 4	4 / 256 = 15.62
3.00	0 to 3	3 / 256 = 11.71
2.56	0 to 2.56	2.56 / 256 = 10
2.00	0 to 2	2 / 256 = 7.81
1.28	0 to 1.28	1.28 / 256 = 5
1.00	0 to 1	1 / 256 = 3.90
Note: In an 8-bit ADC, step size is V_{ref}/256		

Table 7-2: Vref Relation to Vin Range for an 8-bit ADC

8-bit vs 10-bit



ATMEGA16/32

- 8 channels » 8 pins
- 10 bit resolution
- 2^{10} = 1024 steps

ADC calculation

- Digital data output : in 8-bit ADC we have an 8-bit digital data output of D0-D7

While in 10-bit ADC the output D0-D9 , to calculate the output voltage :

Dout =(Vin/step size) ,

where Dout is digital data output (in decimal) , vin : analog input voltage ,

step size is the smallest change ADC could sense ex stepSize = 5/256 = 19mV

ex : find the Dout voltage if you know the Vin which is analog = 2.7 , and ADC is 10-bit resolution , vref = 3 :

Solution :

- First -> step size = Vref / resolution = 3 / 1024 = 2.93 mV
- Second -> Dout = Vin / step size = 2.7 / 2.93mV = 922

Successive approximation ADC

- Is a widely used method of converting an analog input to digital output , it has three main component :

- Sample and hold circuit (S/H) :

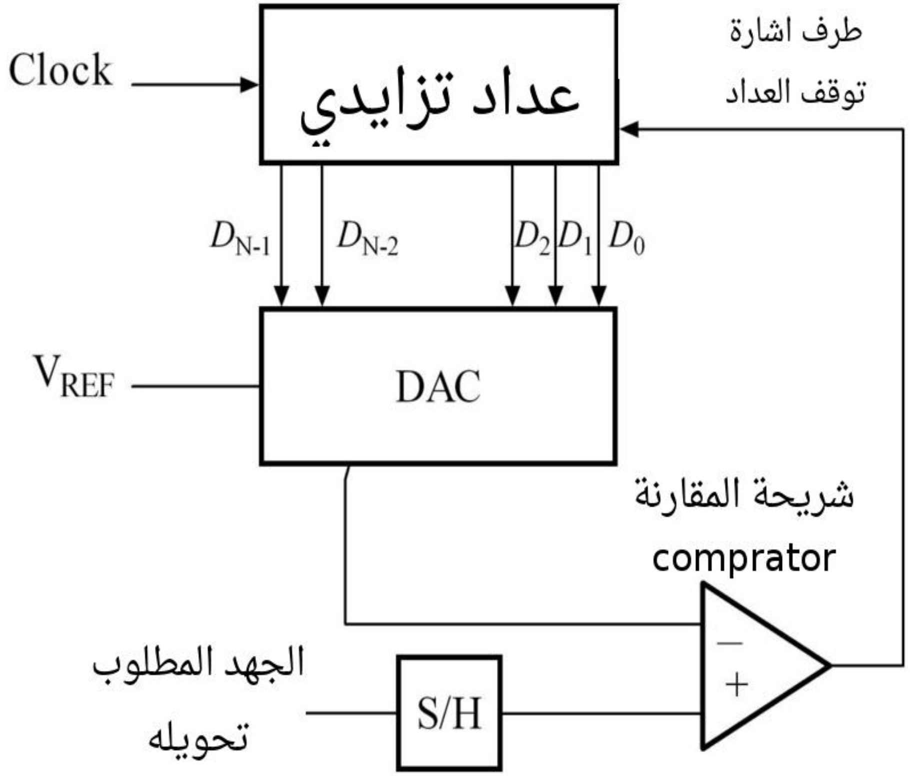
it take a sample from the input voltage and catch it .

- Analog Comparator :

an electronic chip like a scale have two sides to compare between them the first is sample which come from (S/H) and second is :

- (Digital counter + DAC) :

to generate small voltage and send it to the analog comparator , if two voltage aren't not equal the they increment the voltage again until it's be more than or equal the sample , after that analog comparator will tell the control unit to stop , then the reading will store in two registers : ADCH - ADCL



Sample and hold

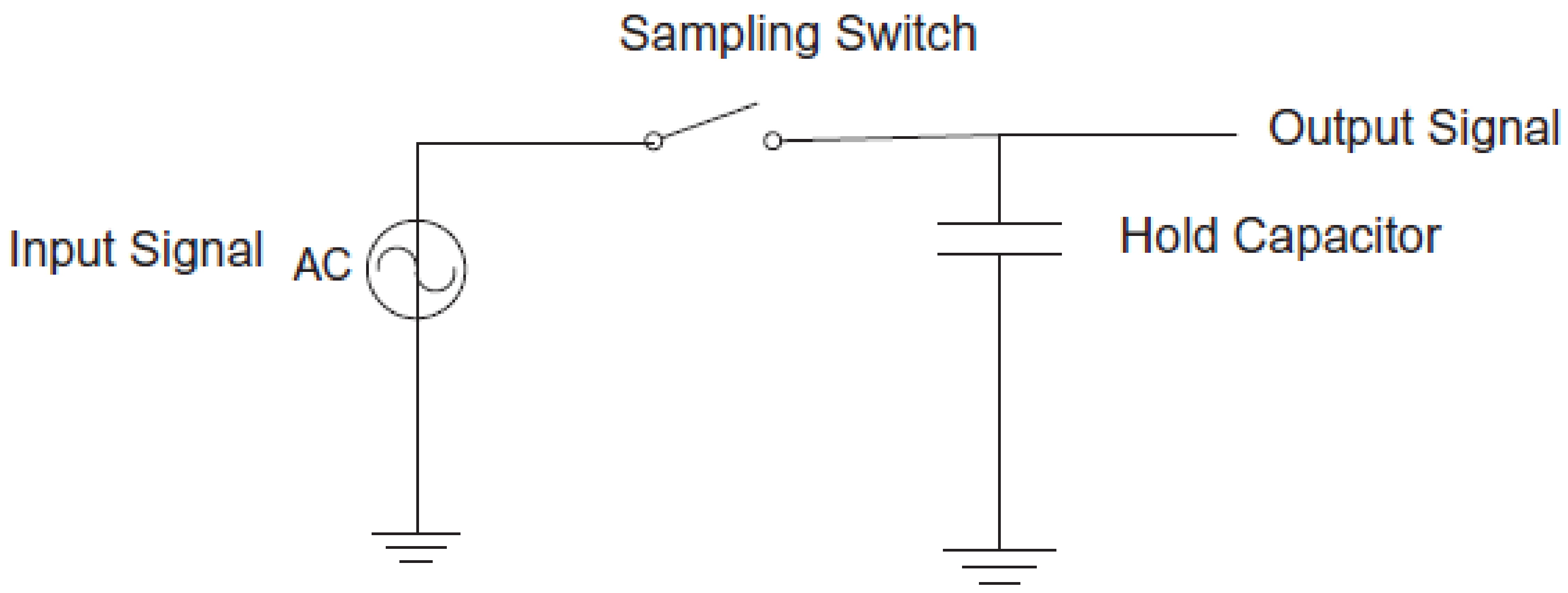


Figure 6.2 Sample and hold circuit.

Atmega32 ADC features we need to know

- Resolutoin could be 8-bit or 10-bit
- 8 Multiplexed Single Ended Input Channels PORTA ->
- ADC has a separate analog supply voltage pin,

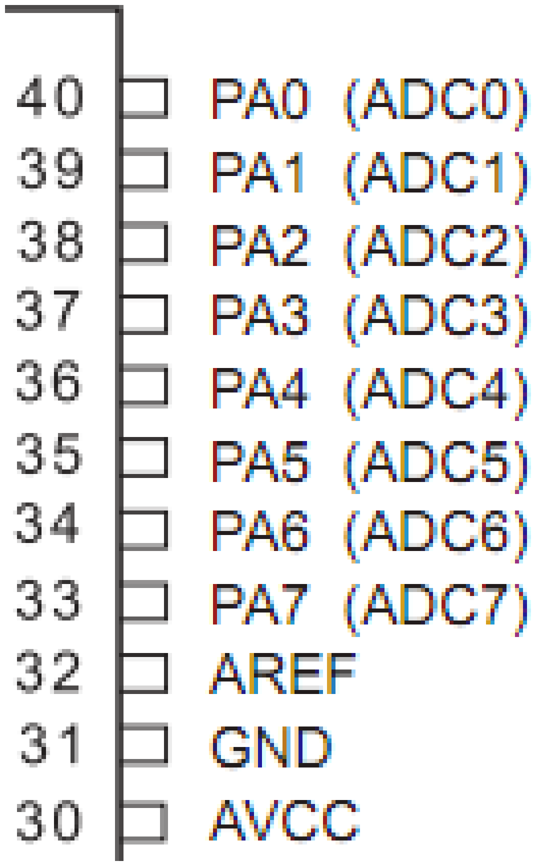
AVCC. AVCC must not differ more than ±0.3V from

- Selectable 2.56V ADC Reference Voltage

: Internal reference voltages of nominally 2.56V or

AVCC are provided On-chip.

- Interrupt on ADC Conversion Complete



ADC Registers

- **ADMUX** : is responsible for choose the pin which will be connected to ADC , also contains some other important bits to set analog reference .
- **ADCSRA** : is responsible for on-off ADC, and control ADC speed (clock).
- **ADCL + ADCH** : two registers to store the ADC readinas

Bit	7	6	5	4	3	2	1	0
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

ADCL

Bit	7	6	5	4	3	2	1	0
							ADC9	ADC8
Access							R	R
Reset							0	0

ADCH

ADC Control and Status Register A – ADCSRA

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

ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

How to use Atmega32 ADC ?

1- Enable ADC :

- ADC by default is disabled to save power we enable it form Enable bit in ADCSRA register :

ADCSRA |= (1<<ADEN) ;

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

2- set ADC clock :

- We said in the architecture of ADC that we have a counter to increase the compare voltage , this counter need a clock to run .
- This counter problem is he want to run on a slow speed in compare to the uC speed (>=1MHz) so we use what is called prescaler register : it's a chip responsible for dividing the clock on specific number of 2's multipliers (2 – 4 – 8 16 – 32 – 64 - 128) , called division factor we choose the division factor from bits ADPS0 – ADPS1 – ADPS2

ADCSRA |= (1<<ADPS0) | (1<<ADPS1) ; // prescaller = 8

Table 85. 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

- From the data sheet we know that ADC clock mustn't be more than 128 kHz

but if it less it's ok . For example : if clock = 1 MHz what's the appropriate division factor ? 1M / 8 = 125 KHz

- The effect of each division factor is the speed of converting A2D .

So we set prescaler = 8 as our clock = 1MHz

ADCSRA |= (1 << ADPS0) | (1 << ADPS1);

- Then we choose 8-bit or 10-bit mode via ADLAR bit if set we use 8-bit

ADMUX |= (1<<ADLAR);

- When working on 8bit mode the reading is saved into ADCH

- When working on 10bit mode the reading will need another register

This mode called left adjusted mode

3- select the wanted input channel via bits MUX0:MUX4

```
ADMUX |=(1<<MUX0); // select A0 to be analog input
```

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

- We let REFS1 – REFS0 bits as default which mean Analog Ref = VCC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

4- start Conversion :

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

5- wait ADC to finish :

```
While (ADCSRA & (1<<ADSC)) ;
```

Then send the reading to PORTC :

```
PORTC = ADCH ;
```

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADC data Register :

Bit	15	14	13	12	11	10	9	8	
	ADC9	**ADC8**	**ADC7**	**ADC6**	**ADC5**	**ADC4**	**ADC3**	**ADC2**	**ADCH**
	ADC1	**ADC0**	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	**ADC9**	**ADC8**	**ADCH**
	ADC7	**ADC6**	**ADC5**	**ADC4**	**ADC3**	**ADC2**	**ADC1**	**ADC0**	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 0

- In some situations we need to save the reading in a variable to use it again :

But if we declare a normal variable the compiler do what is called

code optimization which mean if we write

```
uint8 adcRead ;
adcRead = ADCH ;
PORTC = adcRead ;
```

-The compiler decide to delete the variable as it seems unusable

and put the reading into the port directly : PORTC = ADCH ;

We avoid this by using what is called volatile variables : which enforce the compiler to avoid this variable whatever its content .

So use "volatile" keyword if you have a variable will change during uC working and you want the compiler to delete it :

```
volatile adcRead ;
```

Put all together

- Write a code to read analog input from a variable resistor via input channel PA1 if you know that

- clock frequency = 1MHz , prescaller = 8 , Vref = VCC , resolution = 8 bit

then display the result on PORTD .

References :

- Books :
- Simply AVR - > Abdallah Ali
- The AVR microcontroller & Embedded Systems using Assembly & c -- > Mazidi
- ATMEGA 32A Datasheet
- PIC microcontroller -- > Milan Verle
- Websites :
- https://www.sparkfun.com
- http://maxembedded.com
- https://www.tutorialspoint.com/cprogramming
- https://stackoverflow.com
- https://www.quora.com
- https://www.lucidchart.com

Any questions ?

- Instructor : Mohammed Hemed

- Embedded Systems developer at fab lab Ismailia

Repository link of Embedded workshop Material:

https://github.com/FabLab-Ismailia/Embedded-Systems-Workshop

Contact me :

- Gmail :

mohammedhemed23@gmail.com

- Linkedin :

https://www.linkedin.com/in/mohammedhemed