IEEE
CAIRO UNIVERSITY STUDENT BRANCH

Embedded Systems Workshop

Session 4

IEEE

Hello!

Instructor: Mohamed Magdy

Cairo University, Electrical Engineering (EPE) Department.

Contact me: mohamed.naem04@eng-st.cu.edu.eg

LinkedIn: https://shorturl.at/hgslP

Attendance

place qr code

AGENDA :

Input VS Output

Pull Up VS Pull down

Input VS Output

In our course we will use Microcontroller AVR Atmega32.
It has 32 DIO pins grouped as following:

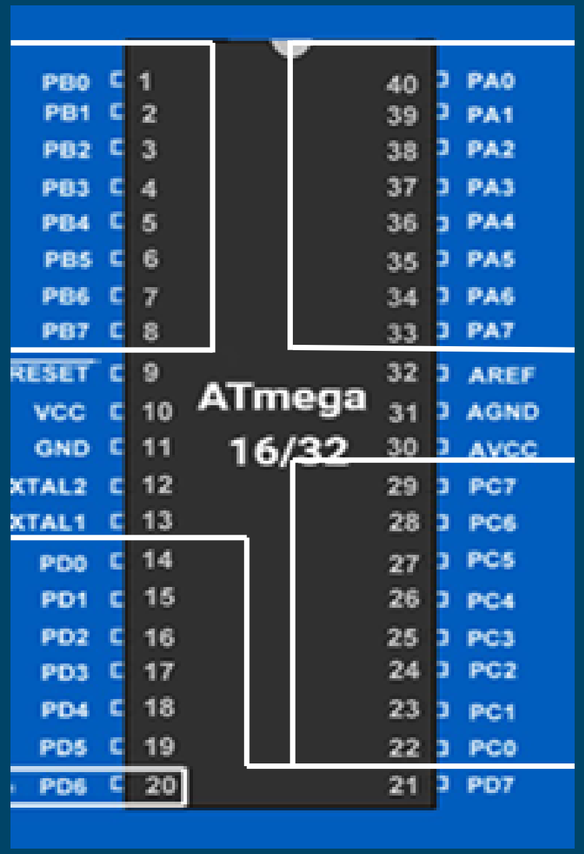
1- PORTA has 8 DIO Pins from A0 to A7

2- PORTB has 8 DIO Pins from B0 to B8

3- PORTC has 8 DIO Pins from C0 to C8

4- PORTD has 8 DIO Pins from D0 to D8

PORT B



PORT A

PORT D

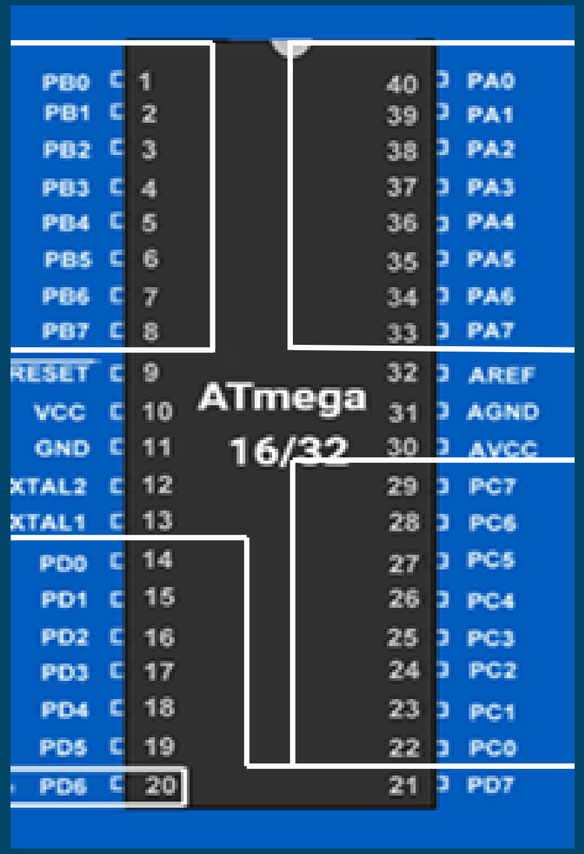
PORT C

Each port has 8 pins the pins of these four ports can be used as general purpose input/output (-GPIO-)

Each I/O Could be input or output (multiplexed direction).This could be controlled through the I/O port direction register

PORT B

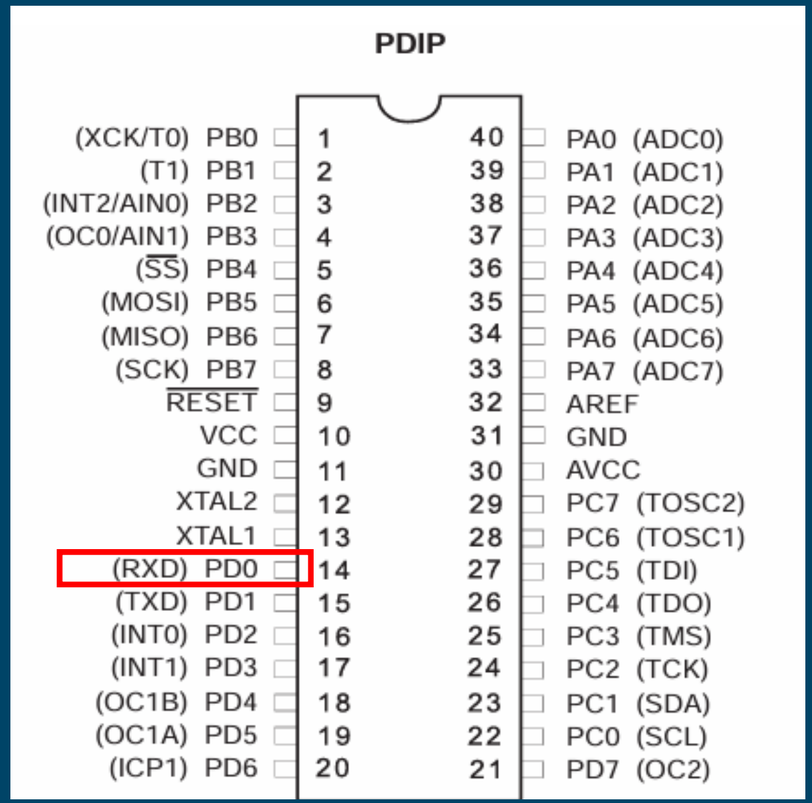
PORT A



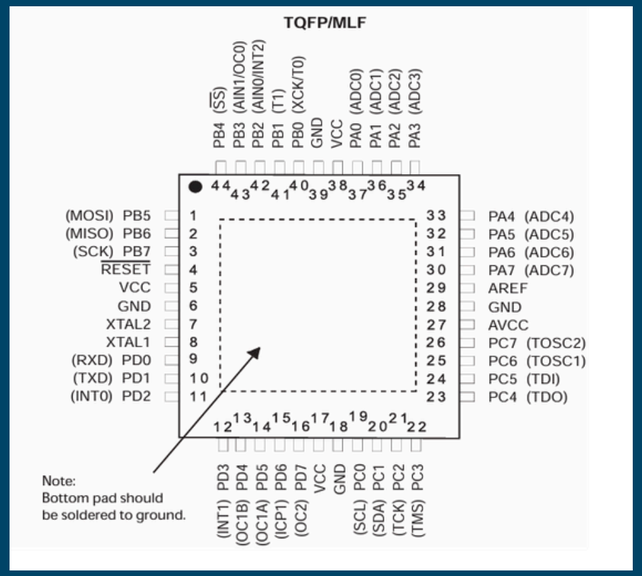
PORT D

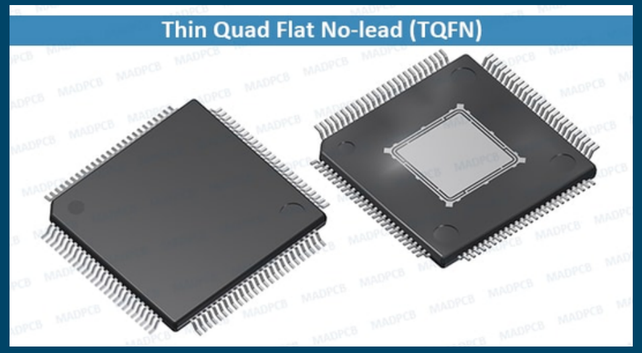
PORT C

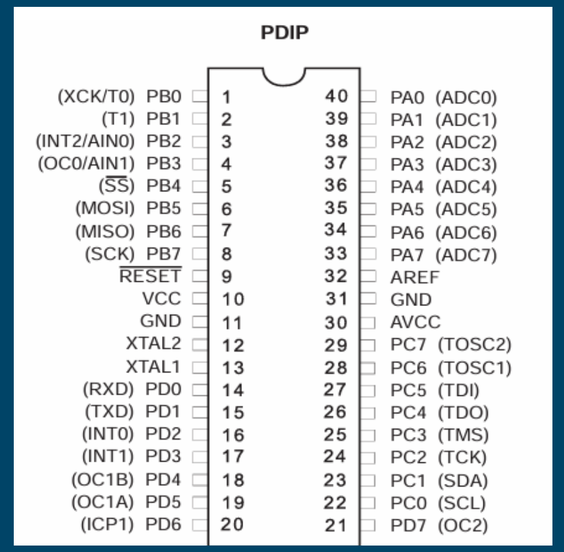
Each MC pin could be assigned to different I/O functions. Pin Could be configured for digital I/O, analog input, Timer I/O, Serial I/O, For example PD0 could be digital I/O or UART serial input (Multiplexed Functionality)



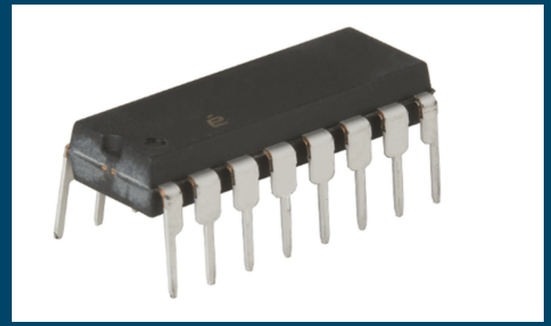
Packaging







PDIP PLASTIC DUAL IN LINE PACAKGE



Each Port is controlled by 3 registers

DDRx(Data Direction Register)

This register used to decide the direction of the pins , i.e Whether the pins will act as input pins or as output pins

PORTx(Output Register)

This register is used to set the logic on the output pins HIGH OR LOW

PINx (Input Register)

This register is used to read the logic level on the port input pins

Note : x could be A,B,C, OR D depending on which port registers are being addressed

Each Port is controlled by 3 registers

Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
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	

Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
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	

Port A Input Pins Address – PINA

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

I/O Pin Programming Steps In AVR MC

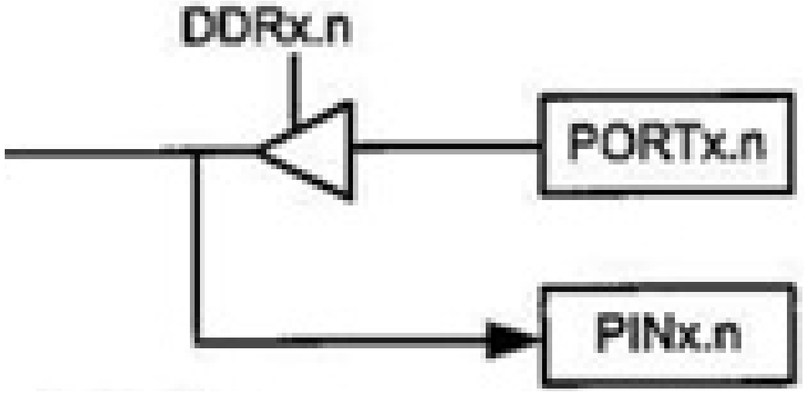
1-Decide which pin is input and which pin is output

2-Configure the port direction use register DDRx

1 for Output. 0 for Input.

3- In Read (input case) Use register PINx

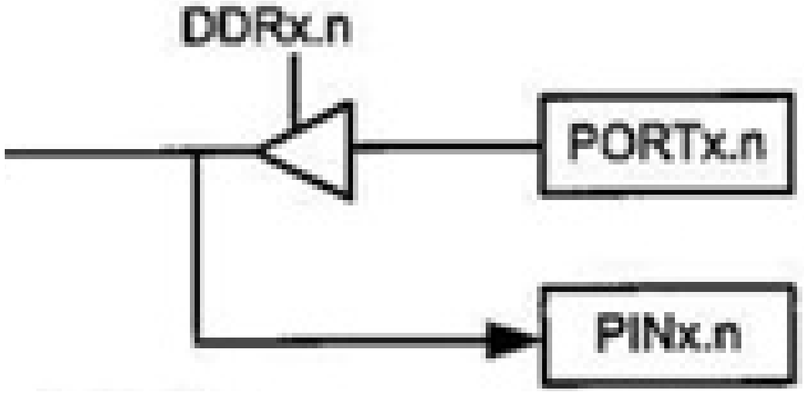
4-In write (Output case) Use register PORTx



I/O Pin Programming Steps In AVR MC

For each output pin we can perform read/write operation on this pin .We can write on it using PORTx register and read it through PINx or PORTx register .

For each Input pin we can perform read operation only through the PINx register



Pull Up VS Pull Down

Pull Up VS Pull Down

What is Pull-up resistor ?

Why we need it ?

How to use this feature ?

What is Pull-down resistor ?

Why we need it ?

How to use this feature ?

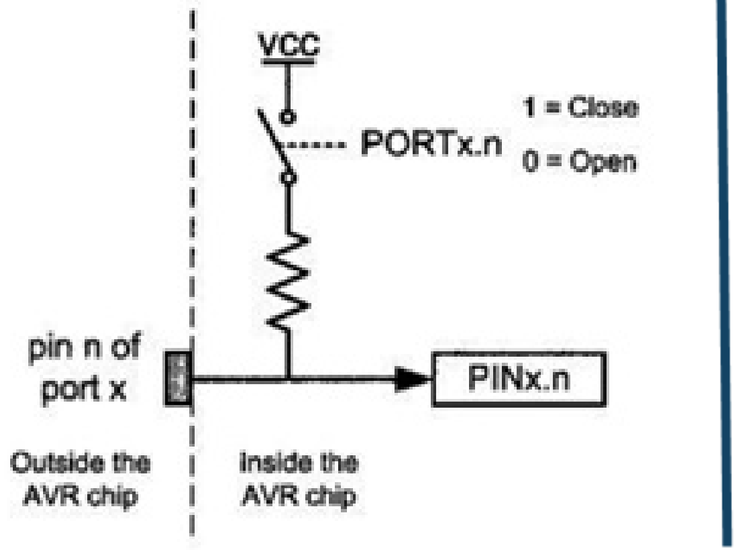
Internal Pull Up resistor

In case you set any PIN as input you can activate the internal pull up resistor by setting the corresponding bit in Portx register.

For each Input pin we can perform read operation only through the PINx register.

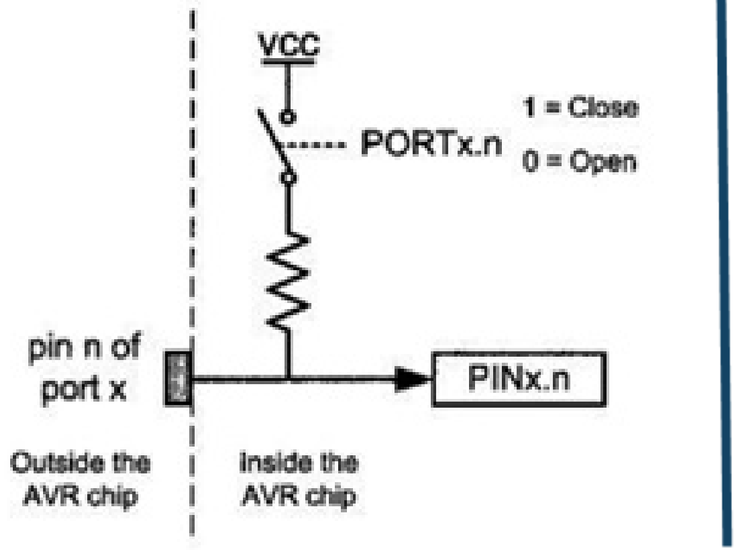
This feature is supported by all pins in the Atmega16/32 MC.

There is no internal Pull-down resistor in Atmega16/32 MC pins but it could be supported in other MC



Pull-up Resistor

Internal Pull Up resistor



Pull-up Resistor

Example :

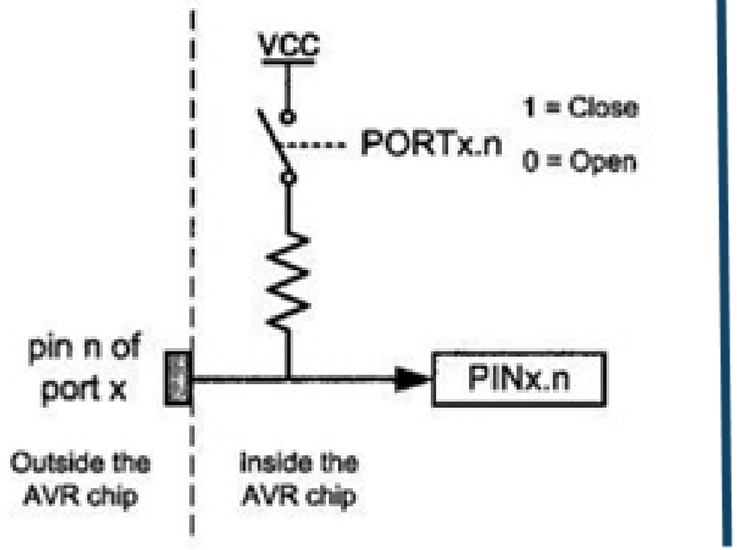
Setup pin 2 in PORTB as input and activate the internal pull up resistor

```
DDRB&=~(1<<2);
PORTB|=(1<<2);
```

Internal Pull Up resistor

Lets Summarize our cases :

A PORT can have four different states depending upon the state of the bits corresponding to this pin in the data direction register (DDRx) and a data register (PORTx) of that specific port.



Pull-up Resistor

4 Cases

DDxn	PORTxn	I/O	PULL-UP	Comment
0	0	I	NO	TRI-STATE (HIGH-Z)
0	1	I	YES	Pxn will source current if pulled low externally
1	0	0	NO	output low (sink)
1	1	0	NO	output high (source)

Internal Pull Up resistor

This is internal PULL-UP resistor to all the pins can be disabled by writing one to the PUD BIT IN THE SFIOR register When this is done the pins are in tri state(high impedance states) if DDRxn =0 and PORTXN=1

Special FunctionIO Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit-2 (PUD) : Pull up disable

when this bit is written to one , the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the PUll-ups

Basics

How to set values in registers ?

Set all pins of Port A AS OUTPUT

DDRA = 255;

DDRA= 0xFF;

DDRA=0b11111111;

How to set values in registers ?

set all Pins in PORTA

PORTA=255;

PORTA= 0xFF;

PORTA=0b11111111;

How to set values in registers ?

Clear all pins in PORTA

PORTA=0;

PORTA= 0x00;

PORTA=0b00000000;

How to deal with a specific pin and conserving other pins ?

Set Specified bit in register

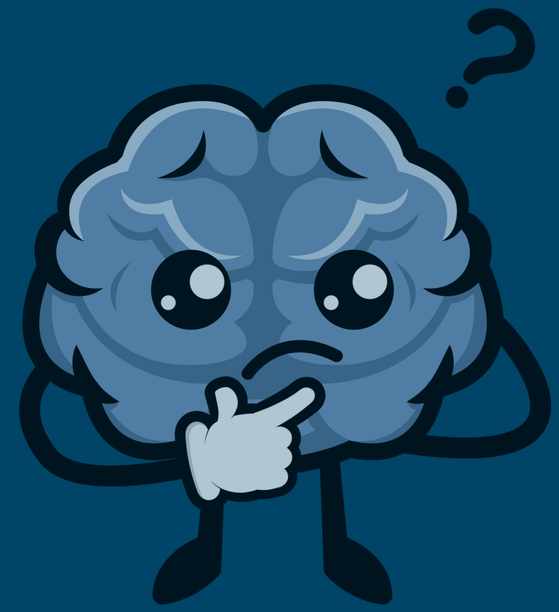
Make OR operation on the old register value with one shift left by the pin number For example if we want to set pin number 5 in PORTA

```
PORTA|= (1<<5);
```

Clear(reset) specified bit in register

Make AND operation on the old register value with the 1 ‘s complement value of one shift left bt the pin number For example if we want to clear pin number 3 in PORTB

```
PORTB&=~ (1<<3);
```



How to deal with a specific pin and conserving other pins ?

Toggle or Flip specified bit in register

Make XOR operation on the register old value with one shift left by the pin number for example if we want to toggle pin number 2 in PORTC

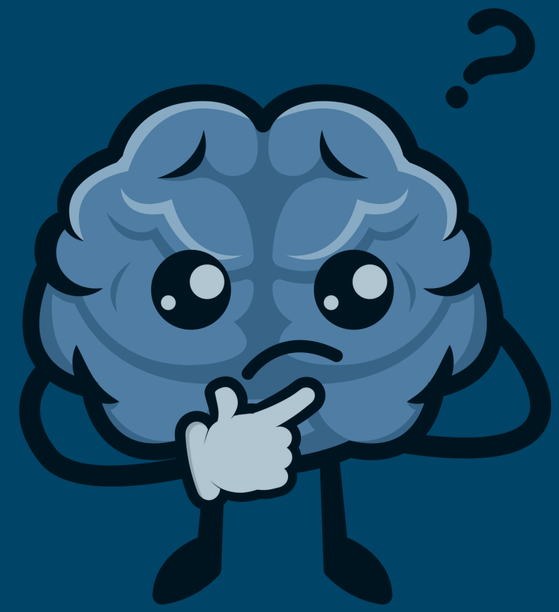
```
PORTC^=(1<<2);
```

How to deal with a specific pin and conserving other pins ?

Check if a specified bit is set (one)

Make AND operation on the register value with one shift left by the pin number For examlpe check if pin 2 in PORTD Is set

```
if( PIND&(1<<2) )
{
/*CODE*/

}
```



How to deal with a specific pin and conserving other pins ?

Check if a specified bit is cleared (reset or zero)

Just revert the previous condition For example check if pin 2 IN PORTD is cleared

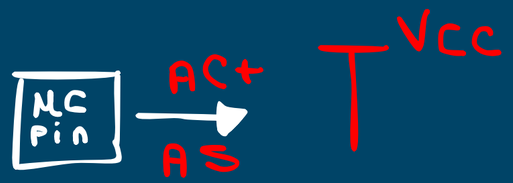
```
if( ! ( PIND &(1<<2) ) ) {

/*CODE*/

}
```

Know More about your MC (DATA SHEET)

Sink current vs Source current

Sink current : PIN LOW

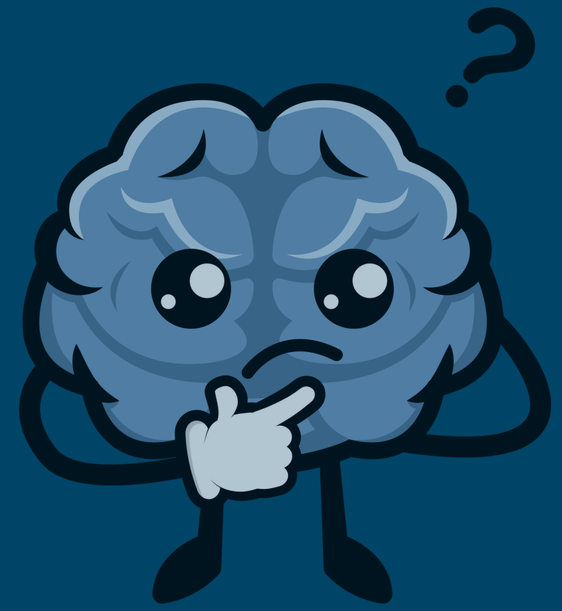
the sink value is the amount of current that can flow into the pin (and therefore into the MC) safely.



Source current : PIN HIGH

The SOURCE current is the amount of the current that can be pulled out of the pin safely.

It is very important that you try to limit the amount of current flowing into and out of your pin (typically using series resistance)



Applications

Output pins

LED AND 7-Segemnt

LCD display

Motors

Buzzer

Signal to other MC

Output to PC through PC Serial Port

Input pins

Switches (Push button ,keypad and etc..)

analog/digital sensors

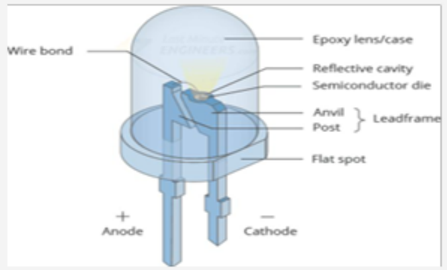
Signal from another MC

input from PC through PC Serial Port

Interfacing

LED :

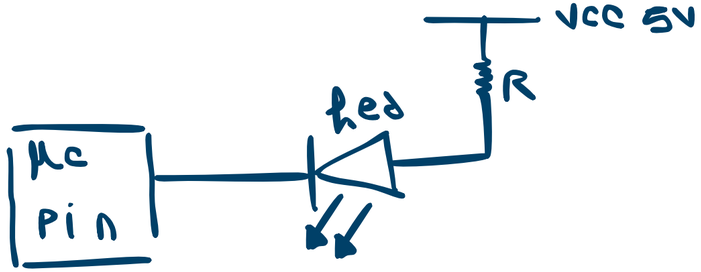
Light Emitting Diode is an electrical element that emits light by supplying a voltage difference between its terminals



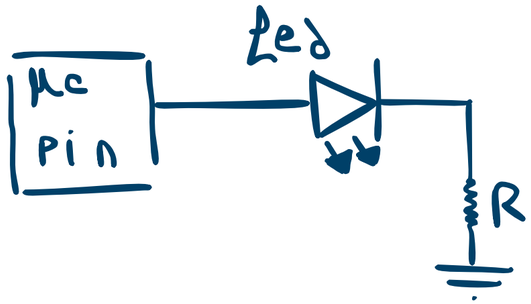
Interfacing

LED :

Negative Logic



Positive Logic



R= 220 : 350 Ω

Interfacing

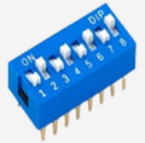
Mechanical Switch :

Mechanical Switch is an electrical component that can connect or break an electrical circuit.

Tactile switch

Paddle switch

DIP switch



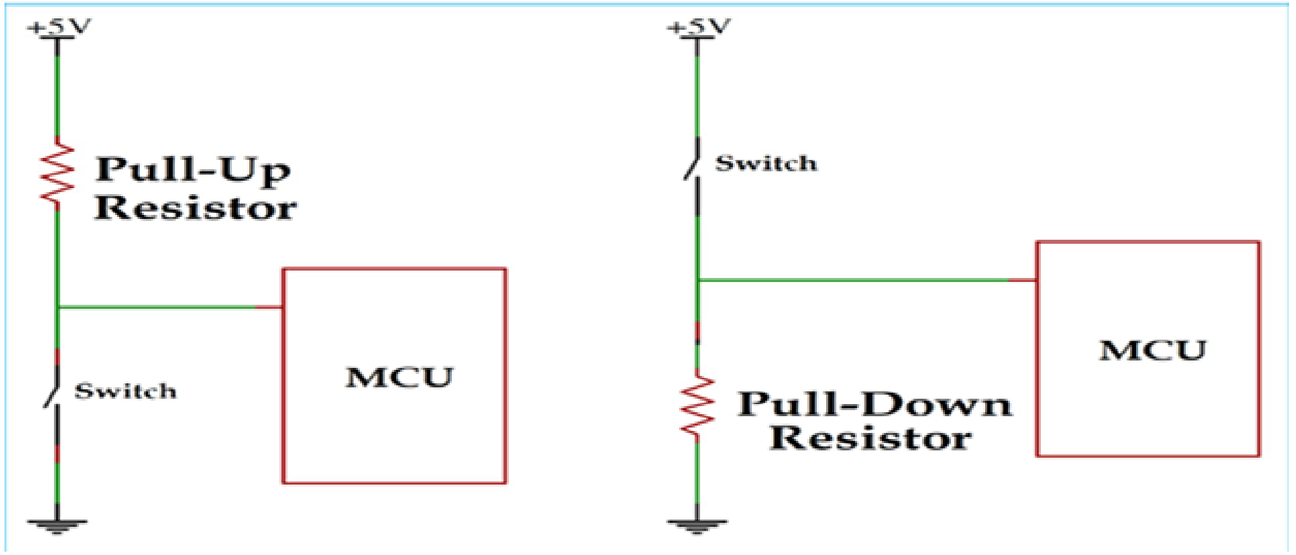
Push Button

Rocker switch

Interfacing

Mechanical Switch :

Switch shall be connected by pull up or pull down resistor to avoid short circuits.

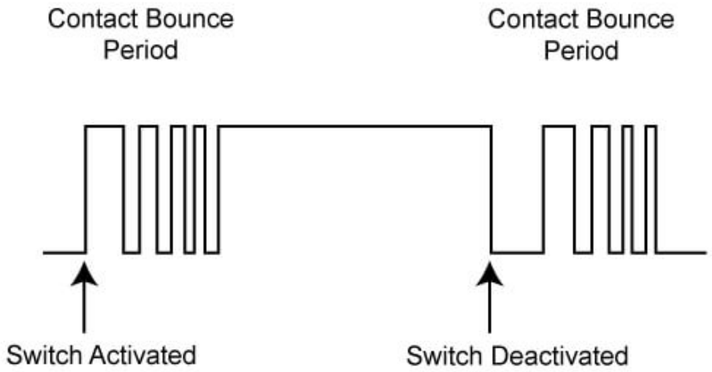


Interfacing

De-bonce Issue

When the Push Button is pressed the switch does not provide a clean edge
The input value is bouncing for 10-30 ms

If this signal was used as an input to a digital counter for example you would get multiple counts rather than the expected single count



Sol:

1-Software
2-hardawre

infinite Loop

Any C project in Embedded Systems application shall have an infinite loop called Super loop . This loop is a must even if you will leave it empty!!

Super loop used as here we have an embedded application used for a certain purpose

This loop prevents the program counter (PC) from continues incrementing over the flash memory and execute a garbage code.

```
while(1) {
/*code*/
}
```

Most Common

```
for(;;){
/*code*/
}
```

Better performance

```
loop:
/*code*/
gotoloop;
```

Not Good !

Busy loop Delay

Software Technique to make the processor halt it's activity for a certain time using loop.

Delay Library "util/delay.h" In AVR provides Two basic functions:

_delay_ms (value in ms);
_delay_us (value in us);

Before Using the delay library we have to define our system frequency through the F_CPU macro used by the delay library:

#define F_CPU 8000000UL 8MHZ

If not defined it will use its default frequency value which is 1Mhz

TASK