LS
C
C++
DS

ES

HW + SW + SPECIFIC TASK

Automatic gate opening system (Hospital entrance gate) - Pure Hardware

-ES ? not ES ,

1. Sensors
2. Motor -Relay -gate
3. Comparator
4. timer

Standalone ES

Slave Standalone ES

-Blindly follows intructions
-Ex Semi automatic washing machine

Independent Standalone ES

-Take decision by its own
-EX-Google car , Tesla car

Real Time ES

-Time bound system
- Within specificied amt of time

Ex.Air bag system , Pacemaker

Networked ES

-Connection
-Wifi based sys
-Bluetooth based

-Home automation sys

Mobile ES

-Movable

Ex. Drones ,Google car ,Robots

Hybrid ES

- if an ES belongs to more than one catagory

Ex: Google cars ,Robots

1. Automatic gate opening system - Hospital entrance gate

1.Sensors
2.Motor -Relay -gate
3.Comparator
4.timer
5.mc

ES ? yES

To whom ? to everyone

2 Automatic gate opening system - Jewellery Manf company

1.Sensors
2.Motor -Relay -gate
3.Comparator
4.timer
5.mc

ES ? yES

To whom

-employee
9 to 6 -
-Biometric
-id
-Face recog - database - image processing

Security - 24/7

2 shifts

Owner - any time

SOC -system on chip -uc or up -Heart
memory - brain

Types of Memory

1.ROM - Read only memory - read - non - volatile
2.RAM -Random access Memory - read and write - volatile
3.Hybrid memory

ROM

PROM / - ProgrammableRead only memory

- programmable - only one
-cant reprogram
-OTP -One time programmable

Ex: Kids toys industry

EPROM - Erasable Programmable Read only memory -

-Erasable
-uv rays
-UVROM

mobile - selfie - EPROM- delete -

Ex: Labs ,R&D

Mask ROM

-Hide / Protect

-manuf date ,slot id

Ex - Bootloader

RAM - Random access Memory
-read and write operation

-SRAM
-DRAM

SRAM - static Ram

-power is lost ,data is lost
-less dense
-low power

ex . cache memory

DRAM - Dynamic Random Access Memory

Ex.Ghajini - short term memory loss

-short term memory loss
-capacitor - no charge - data is lost
-charge refreshment is needed
-more power
-more dense

Ex:Main memory

Hybrid memory

-properties of ROM and RAM

mobile - selfie - PROM - delete ? no

-SRAM - baterry low - sw off - restart - photo ?no

EEPROM - Electrically erasable programmable read only memory

- read and write
- byte access
-read fast
-write slow
-few bytes

1 byte read and write

Flash Memory

-read and write

-block access - 4 byte , 8,64,256

- large amt of data

1 byte

ex

1 block = 8 byte

5 th byte rewrite

NOR Flash

-less dense
-more reliable
-code
-XIP support

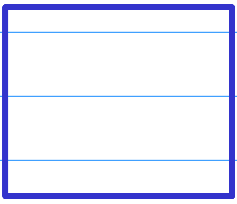
execute in place

NAND FLASH

-more dense
-less reliable
-data

code

data



1 cm2

less dense - MBs

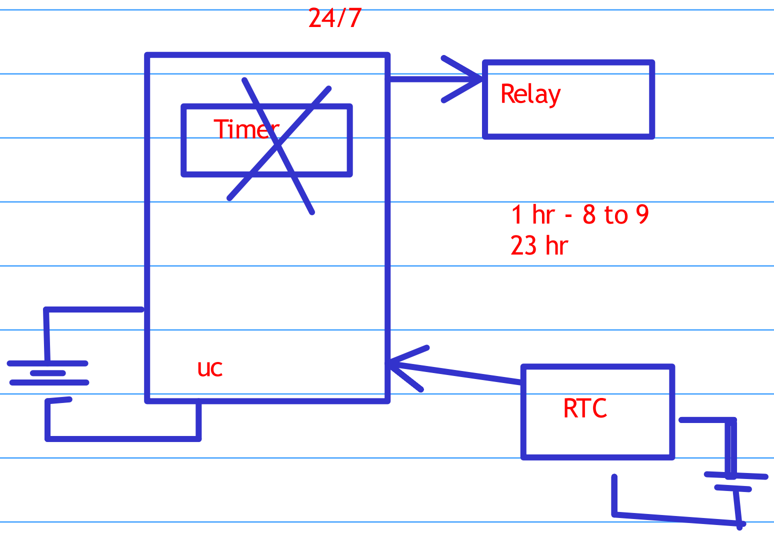
more dense - GBs

main.c -> HDD
gcc main.c -> a.out
./a.out - memory segments - RAM

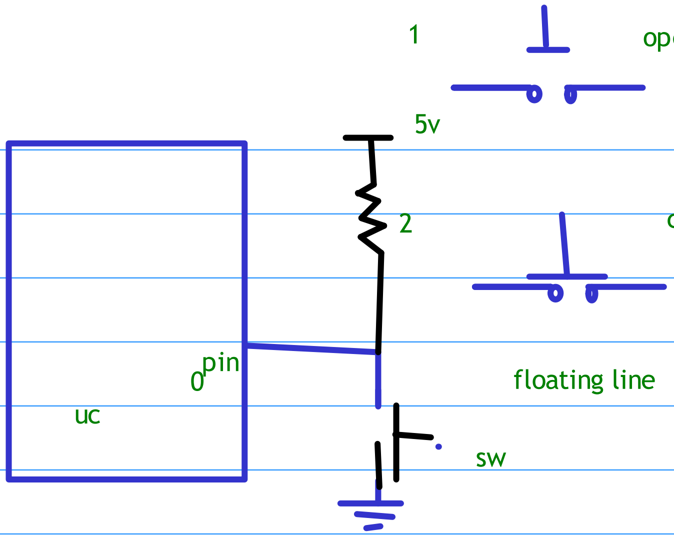
4 - 8 kbs -
12 kb

Automatic Water pump system

Req - start pump at 8 am -9 am everyday



gnd/0v = 0
vcc/5v/vdd = 1



open state/not press

close state/pressed

floating line

Best piece of code written for given req

space
Time

Samsung note 7 -
iphone 8
tata nano -

few byte - eeprom
code - nor flash

3 yrs

Basic prototype- 2 yrs

Non Recurring enginerring cost

xyz

x+y+z - xyz - xy

ES-HW+SW+SPECIIFC task
GPS - General purpose system
-system designed for general purpose

EX-Computer

mobile phone -

A system using which if yoiu are able to design an ES ,its not ES ,its GPS

ex.can i desigin an ES using automatic gate opening system? no - ES

can i desigin an ES using google car? no -ES
can i desigin an ES using MOBILE? YES -ES not GPS

Screen
keyboard
mouse

Real time ES

Asystem capable of taking action in given instance within specified amt of time

Hard real time

-time strictly followed

Ex.Air bag system , pace maker

Soft real time 40 min - 45 mins

- few delay is acceptable

autumatic water pump sys - 8.05 - >

Firm real time

-time strictly followed - loss in property

Rover - soft landing
crash landing - loss

Rover+human - hard

Fire alarm - office -human - > hard real time ->
, stock room - > hard real time

main.c -hdd
gcc main.c - a.out - hdd
./a.out - memory segments

stack
heap
code
data

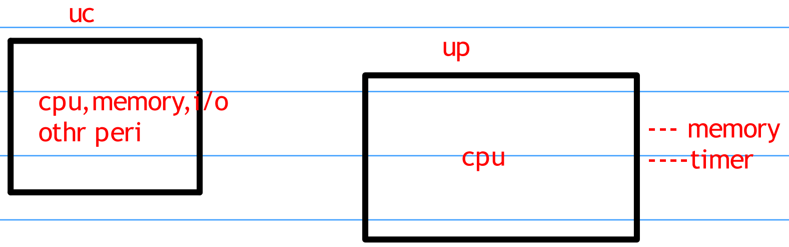
- RAM

same

Ex> Von-Neuman arch

RISC CISC

uC - ?



uc

cpu,memory,i/o
othr peri

up

cpu

--- memory
----timer

soc

3 up ,5 timer, memory

3 years

1 st 2 year 60%

3-

EOL - END OF LIFE - DATE

Host
- Sys which is used to develop a specific app

ex:computer

Target

sys developed for a specific app

ex. online ,simulation tool - PICsimlab board -PIC genious - PIC16F877A

Cross compiler

compiler - app - converts source code to machine code

where is code - >computer
where is it compiled -> computer
running ? target

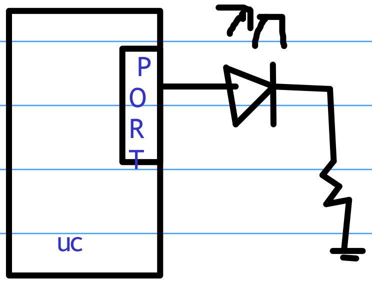
cross compiler

ex, xc8 ,AVR gcc

```
#include <stdio.h>

void  main(void )
{
        int x = 20;

        printf(“%d\n”, x);


}
```

Req -write a program to turn on LEDs

```
//led = 1 // turn on led
 // led = 0 //turn off led

void  main(void )
{
        int led;
        led = 1;
                                  -?
}
```



PORT -interface between uc and ext peril

need to know about pic16f877a - > Datasheet

need to know info about borad - board schematic daigram

PIC16F877A

Ports - 5 PORTS

PORTA	- 6
PORTB	- 8
PORTC	-8
PORTD	-8
PORTE	-3

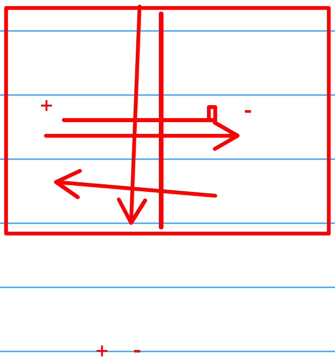
BiDirectional

size of uc = size of data bus = 8 bit

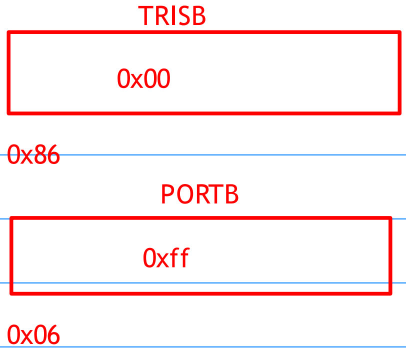
//led = 1 // turn on led
// led = 0 //turn off led
// to turn on leds on PORTB
// direction is output for led

PORTB
-8 BIT
-bidirectional

```
void main(void)
 {
     int TRISB,PORTB;
     TRISB = 0x00;                ?
     PORTB = 0xFF;

}
```



```
void main(void)
{
    unsigned char *trisb = (unsigned char*) 0x86;
     *trisb = 0x00;
    unsigned char *portb = (unisgned char *) 0x06;
    *portb= 0xFF;
}
```



main.c

```
#include <xc.h>
init_config()
{
   //make all portb pins as output
   TRISB = 0x00;
    //turn off all leds
  PORTB = 0x00;
}

Void main()
{
   init_config();
    while(1)
   {
    PORTB = 0xFF;
   }

}
```

.hex

target - exe - compiler - write

mplab xIDE
xc8
picsimlab

current sourcing ckt
to turn on led ,uc = 1
to turn off led, uc = 0

current sinking ckt

to turn on led ,uc = 0
to turn off led, uc = 1

PORTA	-TRISA
PORTB	-TRISB
PORTC	-TRISC
PORTD	-TRISD
PORTE	-TRISE

POTRB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
	1	1	1	1	1	1	1	1
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0

TRISB bit (= 1) will make the corresponding PORTB
pin an input

TRISB7 = 1 // RB7 act as an input

Clearing a TRISB bit (= 0)
will make the corresponding PORTB pin an output (

TRISB3 = 0 // RB3 is output

TRISB = 0xFF ; /1111 1111 //ALL PINS of PORTB is input

TRISB = 0x00 ; //0000 0000 -all are output

TRISB = 0xF0 //1111 0000 -> 0b1111 0000 ,

RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
0	0	0	0	0	0	0	0
DELAY							
0	0	0	0	0	0	0	1
DELAT							
0	0	0	0	0	0	1	1
DELAY							
0	0	0	0	0	1	1	1
1	1	1	1	1	1	1	1

RD0

1111 1111 << 1
1111 1110 << 1

1111 1100

0000 0000 << 1 | 1
0000 0001 << 1 | 1
0000 0011<<1
0000 1100|1

Pull up ckt -low level or falling edge

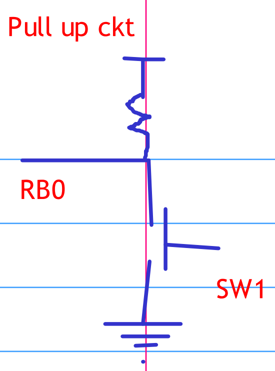
when sw is open - uc = 1
when sw is closed -uc = 0

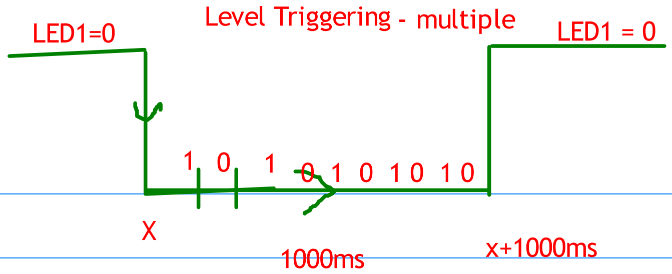
Pull down ckt -High level or rising edge

when sw is open - uc = 0
when sw is closed - uc = 1

In a circuit , if the connection is not complete or if its not connected to ant potential then its called Floating line .

i need to take action based on low level or falling edge



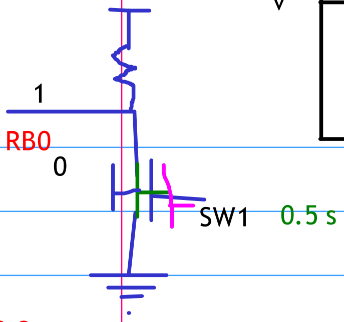


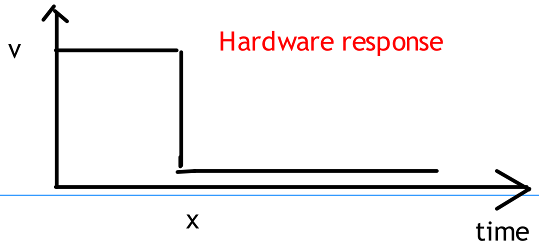
TV remote
vol++ -

```
LED1 = 0;
 main()
 {
    while(1)
     { //100 ms to execute
         if ( RB0 ==0)   //Check sw is pressed or not
            LED1 = !LED1;  //toggling led

      }
 }
```

Pull up

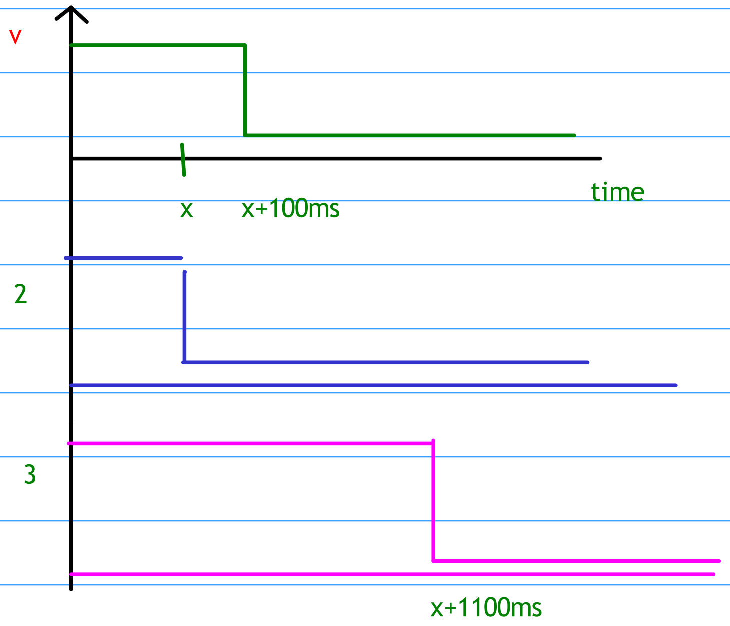
Polling method





Hardware response

software response



```
main.c
void main()
{
 while(1)
 {
 /*code which executes for 100 ms*/
 if (RB0 == 0)
    //turn on fan
 /*code which executes for 1000 ms*/
 }
}
```

Disadvantage of polling method

- Bad response
- Event loss
- poor power managemnet

Interrupt

Interrupt is a signal ,it wil make the processor to stop its current exexution proceed to execute the interrupt handler set for interrupting source

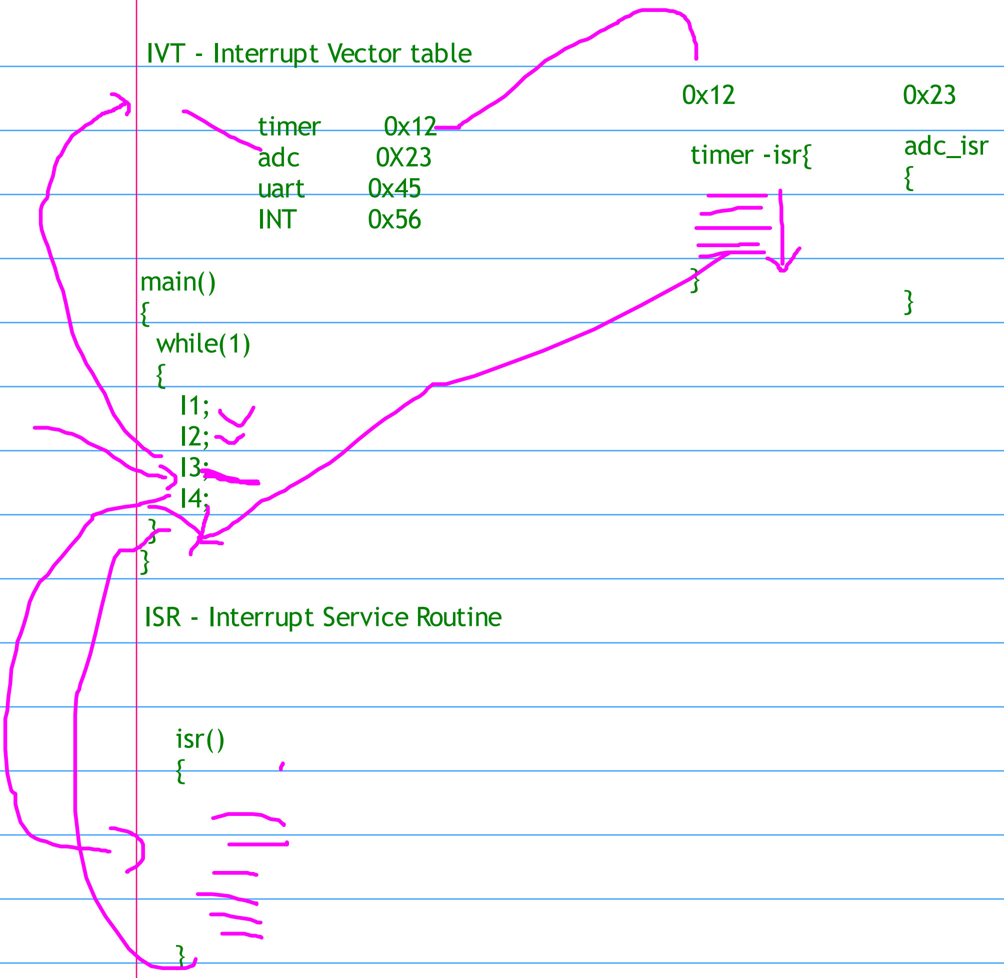
Exceptional error

INT ,TRAP

LI
system calls -software interrupt

IVT
ISR

IVT - Interrupt Vector table



timer	0x12
adc	0X23
uart	0x45
INT	0x56

```
0x12                0x23

timer -isr{         adc_isr
                    {

}
                    }
```

```
main()
{
 while(1)
 {
   I1;
   I2;
   I3;
   I4;
 }
}
```

ISR - Interrupt Service Routine

```
isr()
{

}
```

isr

- simple and short

-no blocking stst
  loop

Latencey is the time taken from the point when interrupt occurs to the execution of 1st instruction in thr interrupt handle

Air bag system - crash /acc - interrupt - opens air bag(1st inst)

timer
uart
adc

PIC16F877A
- 15 sources of interrupt
- 15 interrupt flag bit
  -It also has individual interrupt enable bits.

- Global interrupt enable

GIE = 1 //enable all interrupt
GIE = 0 //disables all interrupt

```
main()
{
  while(1)
  {

  ..
  }
}
```

```
isr()
{
   if( INTF == 1)
  {
  //take action
  INTF =0;
  }




}
```

External Interrupt (INT)

- INT
- RB0

-INTEDG = 0 //selecting falling edge
INTEDG = 1 //Selecting rising edge

INTE = 1 //enables external interrupt
INTE = 0 //disables external interrupt

Toggle LED whenever ext interrupt occures

-LED
- External interrupt

Togglr LED using polling and interrupt

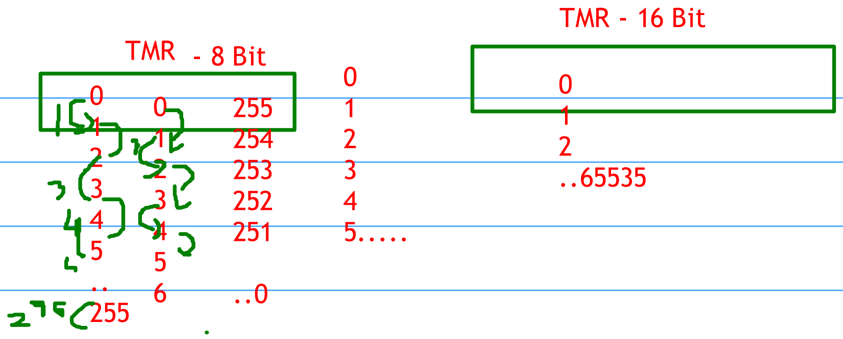
SLEEP(3)
xc sleep

LED1 - interrupt method
LED2 - polling method

-LED
-External interrupt
-DKP

off
on-off-on-off-on
off-on-off-0n-off

Resolution -Width of timer register



Tick

change of timer register value from one valur to other

Total ticks = 255

The period of tick

1 Tick duration = Quantum

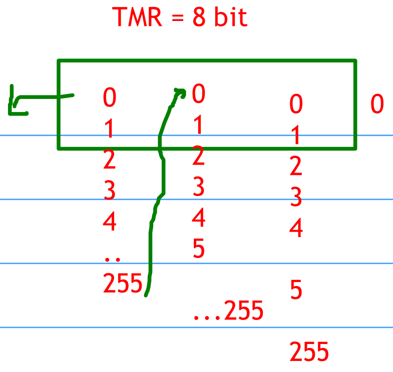
System clock settings

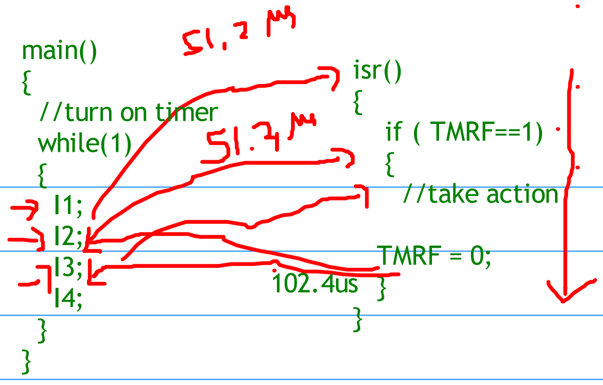
Oscilator freq =20 Mhz

Quantum / 1 tick duration = 1 IC
=4 pulses
= 4 (1/20 MHz)

Quantum = 200ns

Total ticks = 255
Total time for ticks = 255 * 200ns
= 51 us





```
main()
{
  //turn on timer
  while(1)
  {
    I1;
    I2;
    I3;
    I4;
  }
}

isr()
{
  if ( TMRF==1)
  {
    //take action

   TMRF = 0;
  }
}
```

How many ticks are needed for interrupt = 256

Time for interrupt = 256*200ns
= 51.2us

Scale

measurement

1:1

1: 2
1:10
1:100

Scale is a method where we can delay the process of going into isr if req.

Prescale
Postscale

Prescale - Instrustion cycle

1:1 , 1 tick = 1 IC

1:2 , 1 tick = 2 IC ,400 ns , time for interupt = 256 * 400ns = 51.2us * 2 =102.4 us

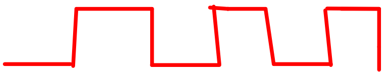
1:4 , 1 tick = 4 IC , 200ns*4 ,800ns ,Time for innterupt =51.2 us * 4= 204 .8 us

1:8 , 1 tick = 8 IC
...
....

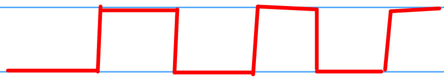
Post scale - overflow

1:1 - how many overflow for interrupt = 1 overflow - 51.2us
1:2 - 2 overflow - 2 * 51.2 us = 102.4us
1:4 - 4 overflow - 4 * 51.2 us = 204.8 us

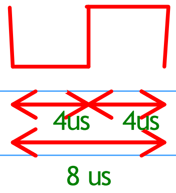
```
main.c
#define LED1 RD0
void main()
{
LED1 = 0;
while(1)
{
  LED1 = !LED1;
  for (delay=100000;delay--;);
}
```



RDO - CRO

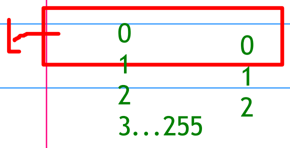


Req
5 pulses of 8us



Resultion - 8 bit
Q = 1 us

TMR- 8 bit



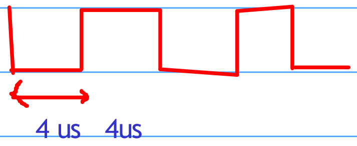
```
void main()
{
  cp = 0;
  //turn on timer
     TMR = 252;
  while(1)
  {
     ;
  }
}
```

4us

```
isr()
{
 if (TMRF == 1)
 {
  TMR= 252;
   cp = !cp;
   TMRF = 0;
 }
}
```

252 252
253
254
255 0

0
1
2
3
4
5

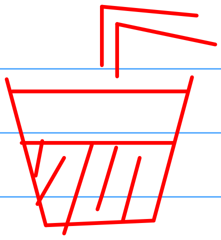


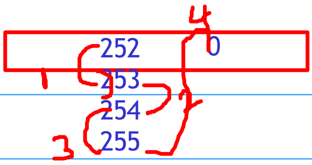
Total ticks for interrupt = 256
Total time for interrupt = 256 * 1 us
=256 us

Need to tooglr for every 4us
4 us
TIMER should interrupt = 4 us

= 4us
=only 4 tickd
- 4 * 1 us = 4us

time = 10 mins
time = 5 mins ✓




4
252
0
1
253
2
254
3
255