Zagazig University
Faculty of Engineering
Computer and Systems Eng. Dept.

جامعة الزقازيق
كلية الهندسة
قسم هندسة الحاسبات والمنظومات

.0

Lab(2): Interrupt handling with PIC Microcontroller

1. Introduction:

Digital interrupts represent one of the main concept used in modern computers and embedded systems. In this lab, we'll explore close the usage of this concept in a real-life situation using one of the most common chips in Pic productions which is PIC16F877A.

2. Objectives

- Applying the concept of interrupts in HW project using PIC16F877A.
- Advancing the Simulation skills using Proteus as a simulator.
- Using PIC microC Pro as a compiler for C.
- Using MPLAB IPE as a compiler.

3. Requirements

SW requirements:

- Proteus 8 Professional (or higher compatible versions)
- microC PRO for PIC v6.6 (or higher compatible versions)
- MPLAB IPE v3.50 (or any other burning software)

http://ww1.microchip.com/downloads/en/DeviceDoc/MPLABX-v3.50-windows-installer.exe

HW requirements:

- PIC16F877A
- PICkit 3 Programmer (or any other working programmer)
- Crystal 4.000 MHz
- 12V DC Fan
- Relay 5V Coil
- 9V DC Battery
- UA741CN "General Purpose Single Operational Amplifier"
- Character LCD 2x16
- LM35dz "Temperature Sensor"
- L7805CV "Positive Voltage Regulator 5V"
- 1 kΩ - 10 kΩ resistor

Prerequisites:

- Basic understanding of C programming.
- Basic understanding of the Proteus schematic design.
- Basic knowledge of HW testing in debugging.

4. Procedure

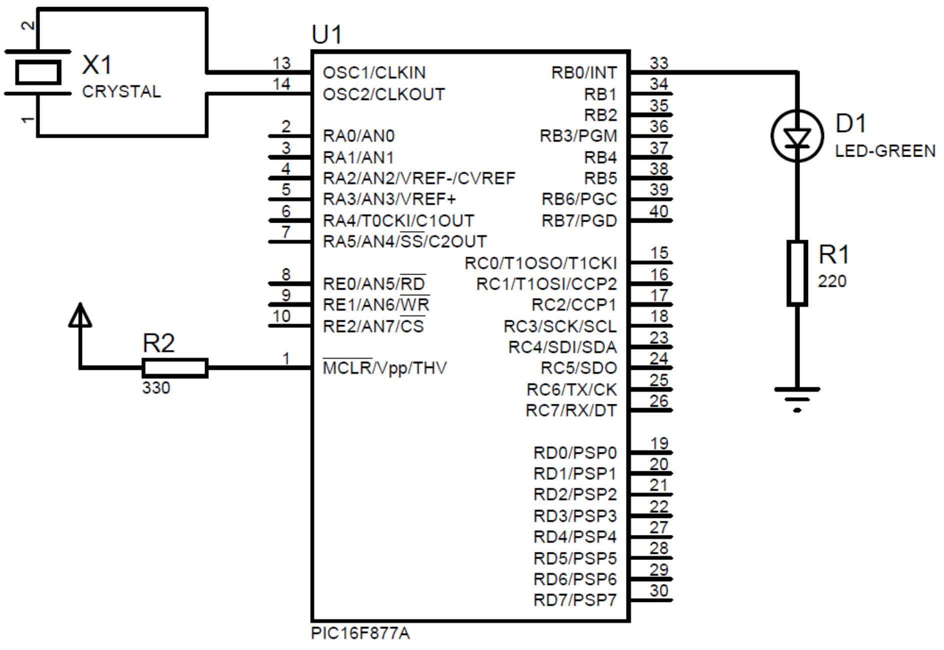
Review:

Interrupts are special events that requires immediate attention, it stops a microcontroller/microprocessor from the running task and to serve a special task known as Interrupt Service Routine (ISR) or Interrupt Handler.

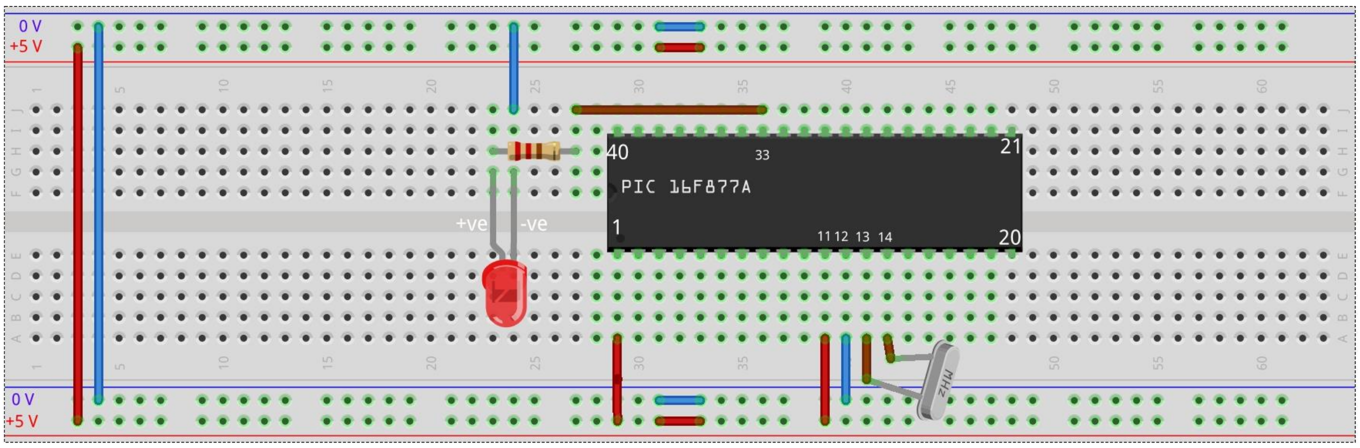
Interrupt is the one of the most powerful features in embedded applications. Almost all the real-time applications are implemented using Interrupts.

Before starting to explore the interrupts option in the (PIC16F877a) IC. A simple example should be implemented as a reminder on how to connect and use the IC.

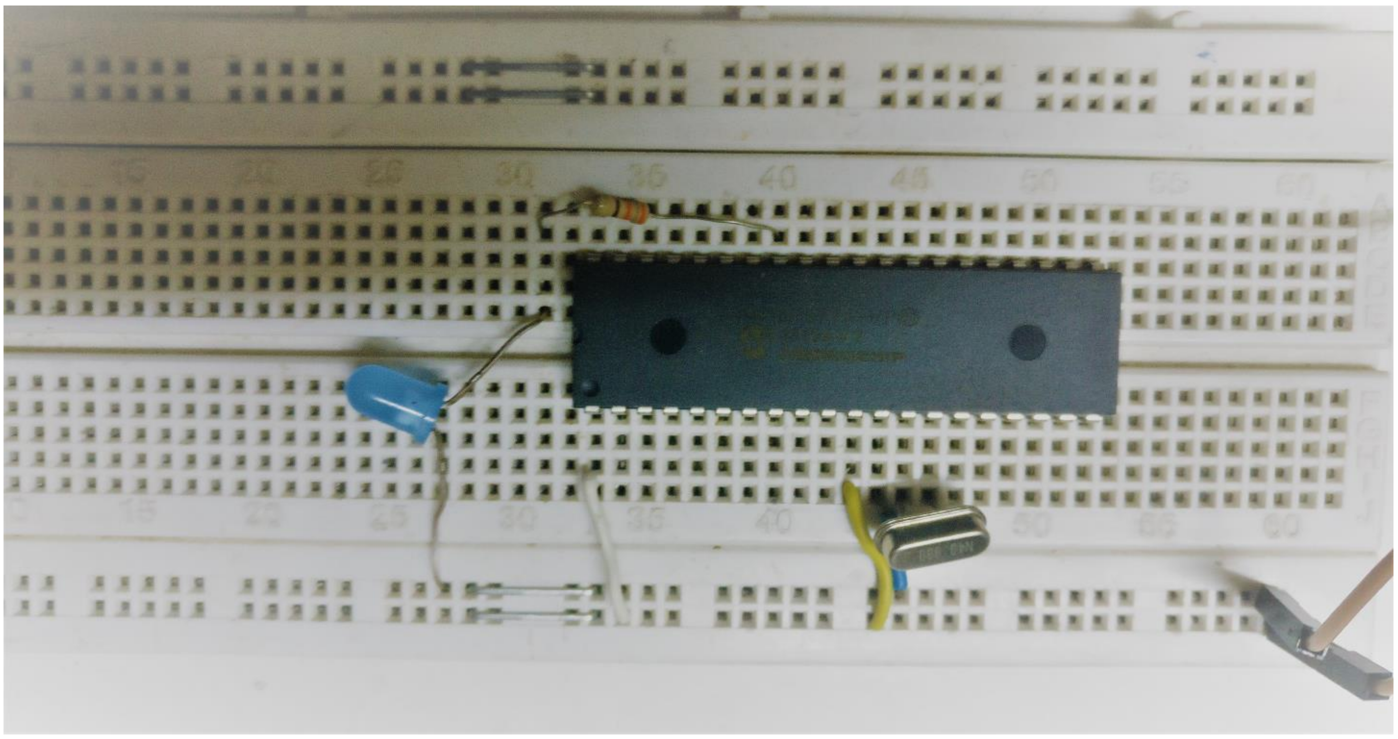
Example 0: Design a simple blinking led circuit that switches states between high and low every 1 second.



Board Connection:



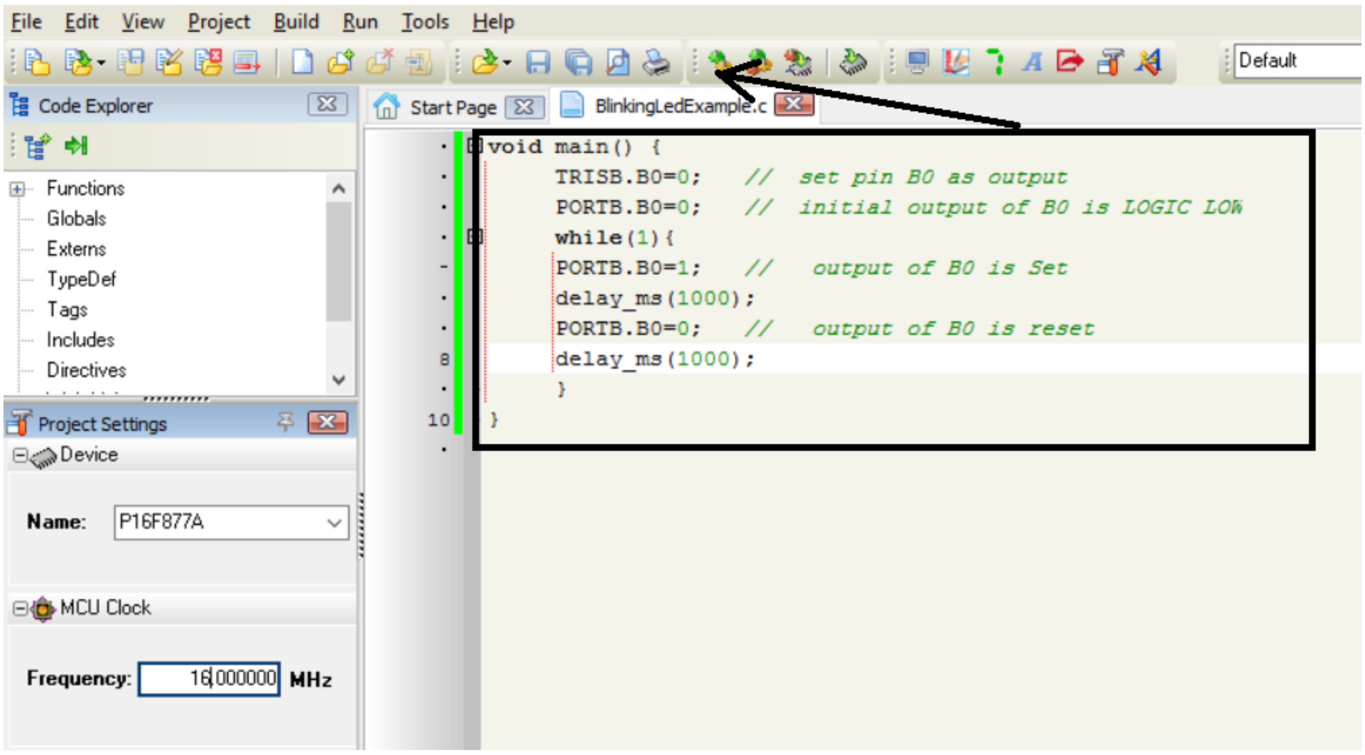
Real life view:



https://youtu.be/049JmdmIOdo

But how is the code burnt the PIC 16f8777a Chip?

First write the blinking led code in your MicroC PRO for PIC and press compile as the following pic.

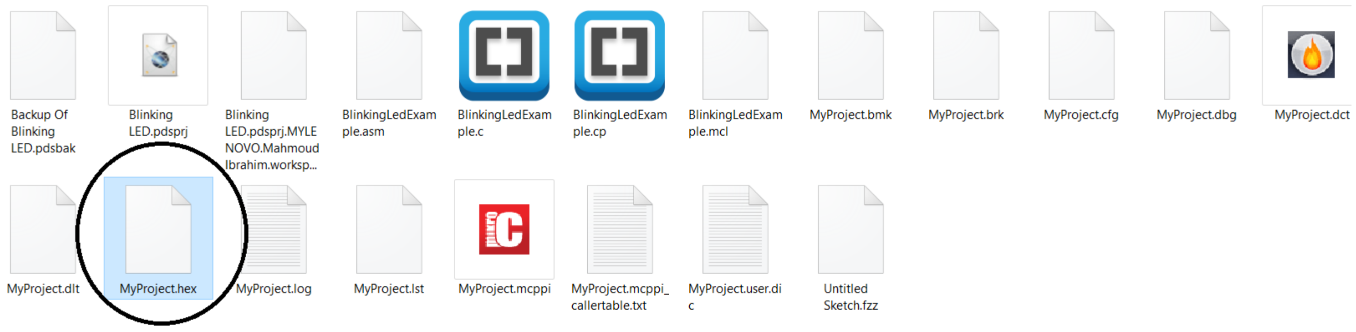


```
void main() {
     TRISB.B0=0;    //  set pin B0 as output
     PORTB.B0=0;    //  initial output of B0 is LOGIC LOW
     while(1){
     PORTB.B0=1;    //   output of B0 is Set
     delay_ms(1000);
     PORTB.B0=0;    //   output of B0 is reset
     delay_ms(1000);
     }
}
```

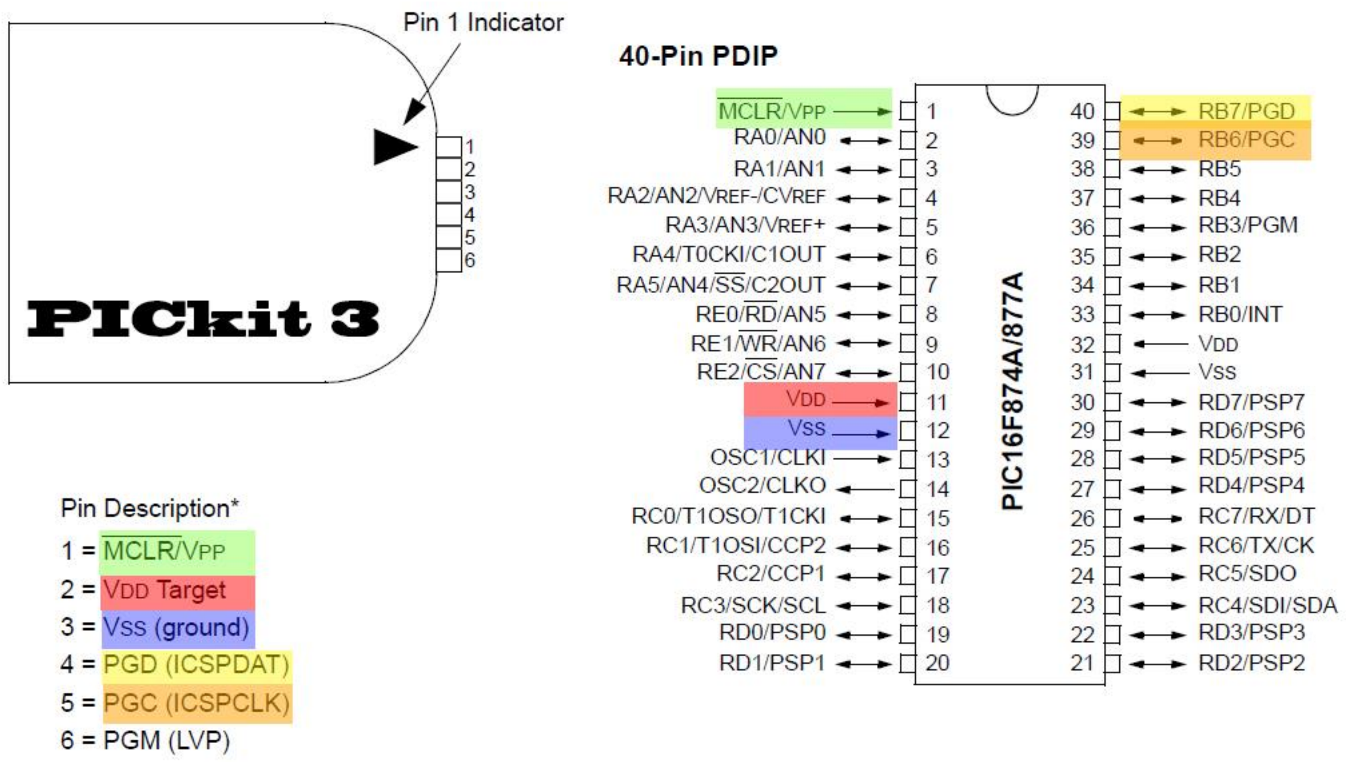
Device Name: P16F877A

MCU Clock Frequency: 16.000000 MHz

This will generate a hex file in your project path.

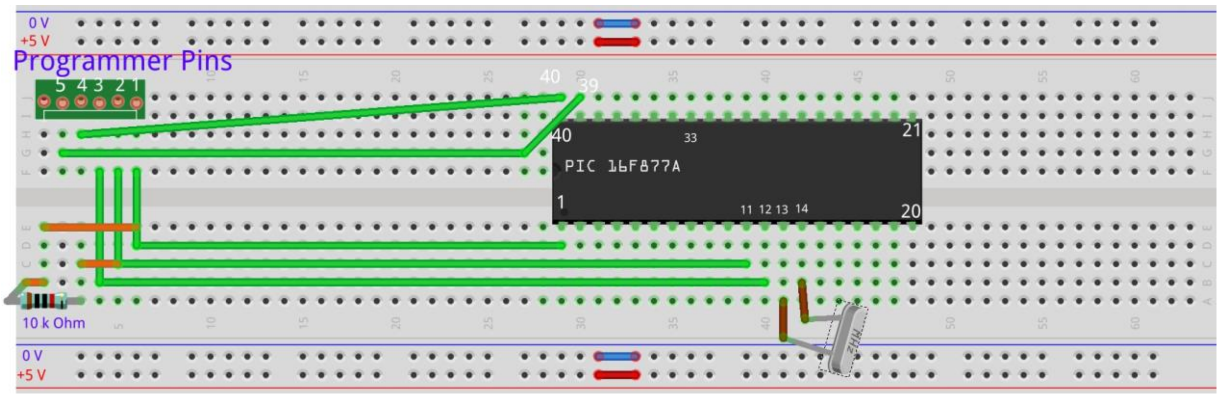


Then connect the PICkit 3 to the pic as the following figure suggests.

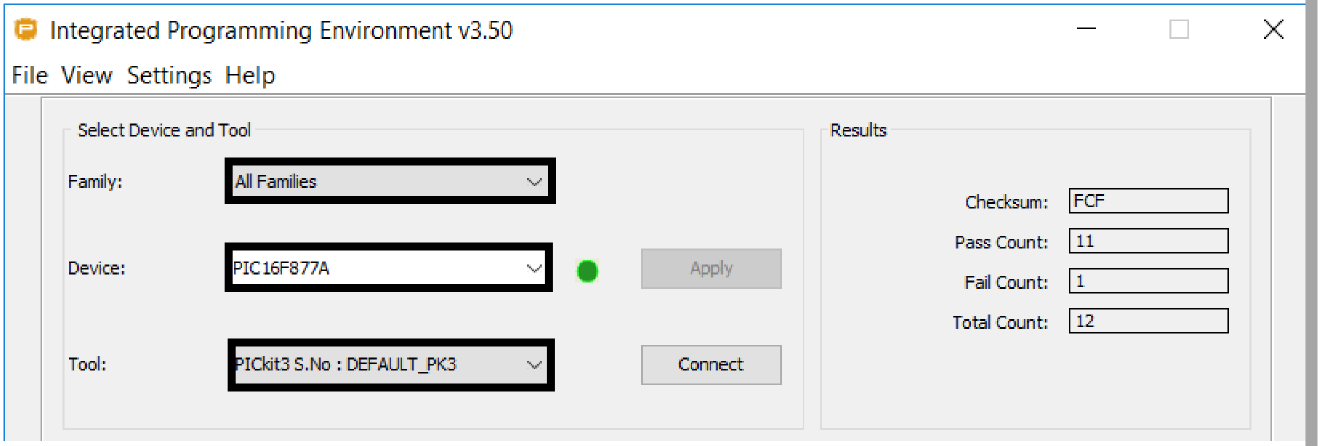


Connect matching colours

- **Please remember to connect the Crystal to pins 13 and 14.**
- **Add a 10k resistor between PICkit 3 pins (1 and 2).**
- **Pin 6 is not connected.**
- **After you make sure that the connections are right. Connect the USB to your Desktop and the programmer.**



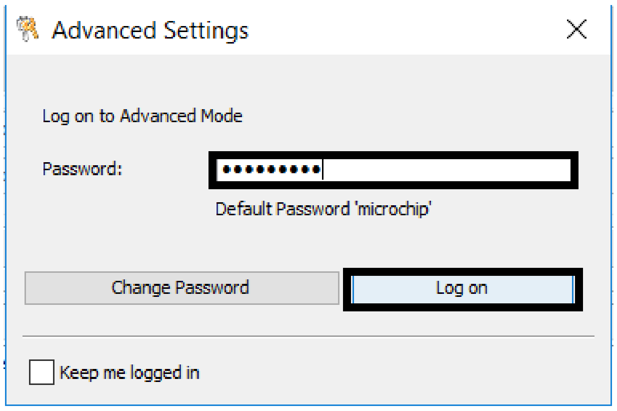
- **Open the MPLAB IPE v3.50 and Select the device and tool.**



Family: All Families
Device: PIC16F877A
Tool: PICkit3 S.No : DEFAULT_PK3

Checksum: FCF
Pass Count: 11
Fail Count: 1
Total Count: 12

- **From Settings menu choose Advanced Mode.**
- **Type the default password is: <u>microchip</u> then login.**



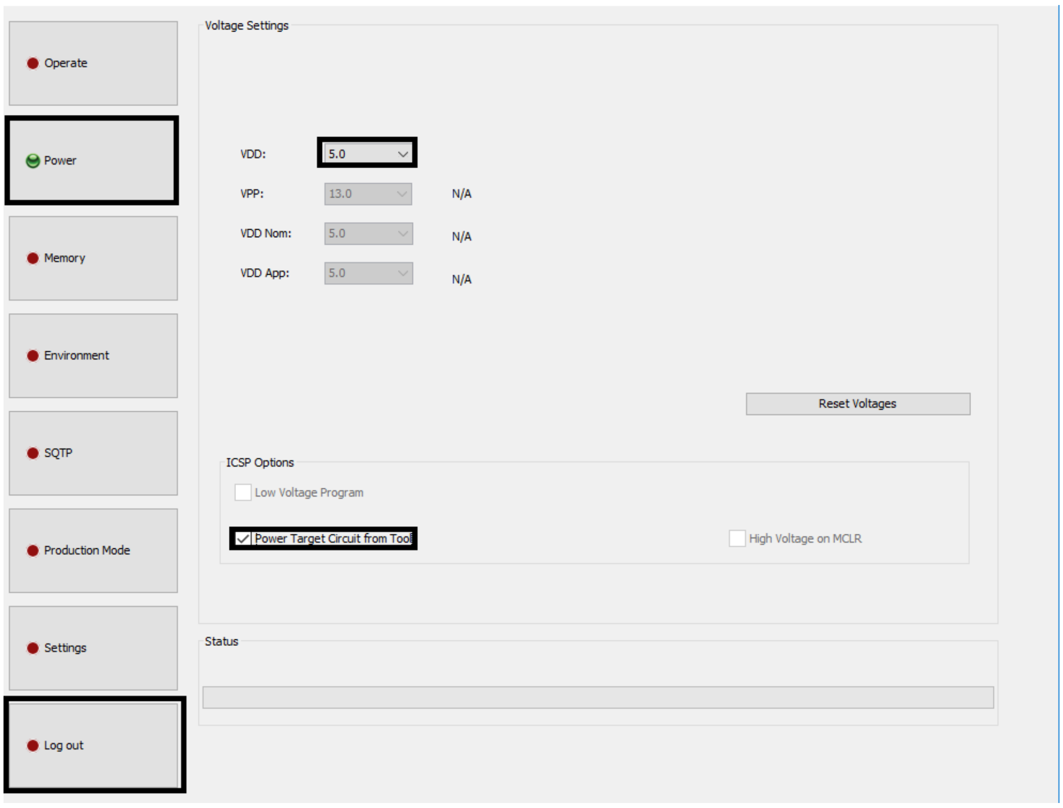
Log on to Advanced Mode

Password: ••••••••••

Default Password 'microchip'

Keep me logged in

- **Press the power button then choose VDD = 5 and mark the check box in the figure below then log out using the log out button on the left side.**



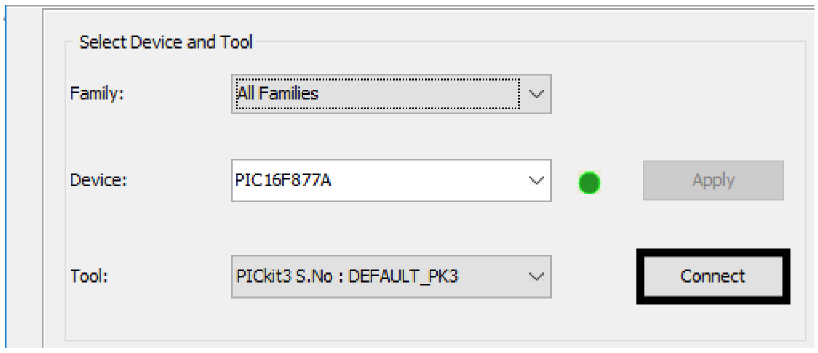
Voltage Settings

VDD: 5.0
VPP: 13.0 N/A
VDD Nom: 5.0 N/A
VDD App: 5.0 N/A

ICSP Options

☐ Low Voltage Program
✓ Power Target Circuit from Tool
☐ High Voltage on MCLR

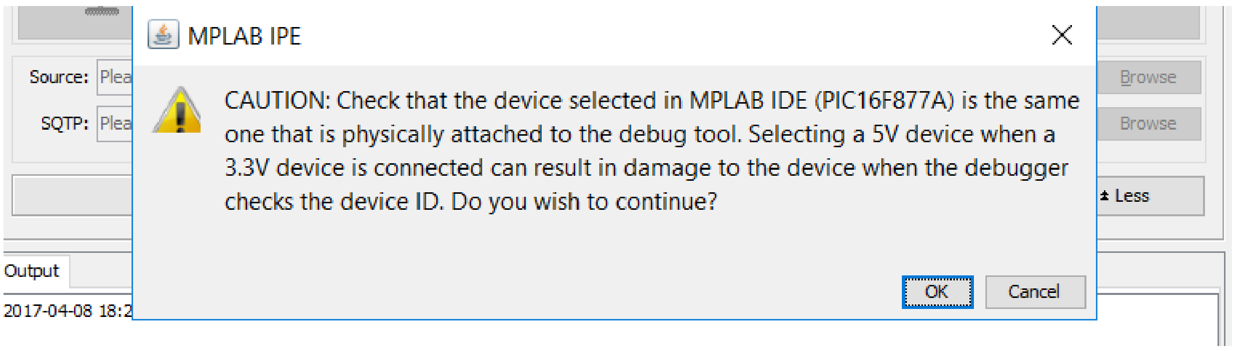
- **After making sure all the circuit and PICKit connections are right press connect.**



Select Device and Tool

Family: All Families
Device: PIC16F877A
Tool: PICkit3 S.No : DEFAULT_PK3

Press ok for this warning.



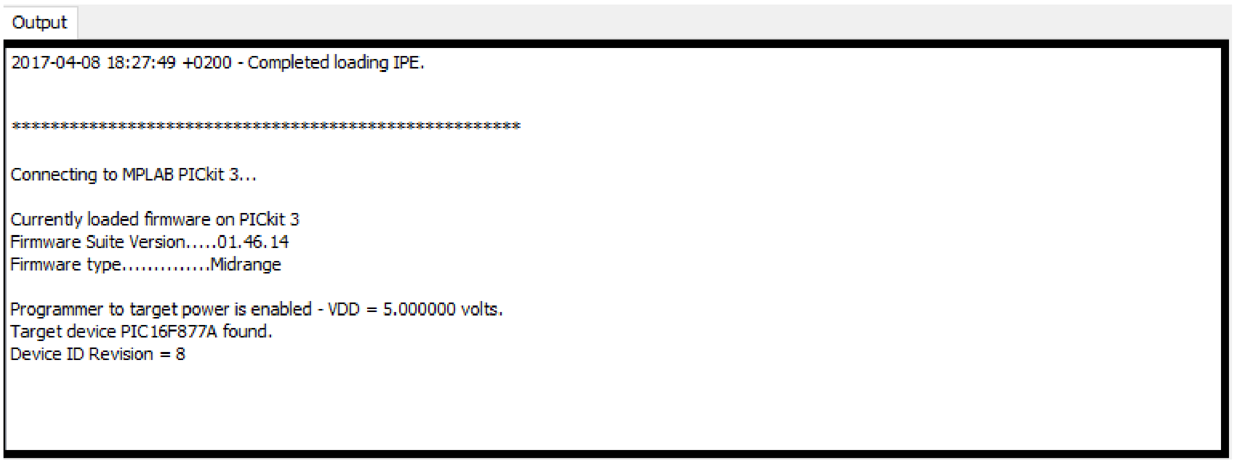
MPLAB IPE

CAUTION: Check that the device selected in MPLAB IDE (PIC16F877A) is the same one that is physically attached to the debug tool. Selecting a 5V device when a 3.3V device is connected can result in damage to the device when the debugger checks the device ID. Do you wish to continue?

2017-04-08 18:2

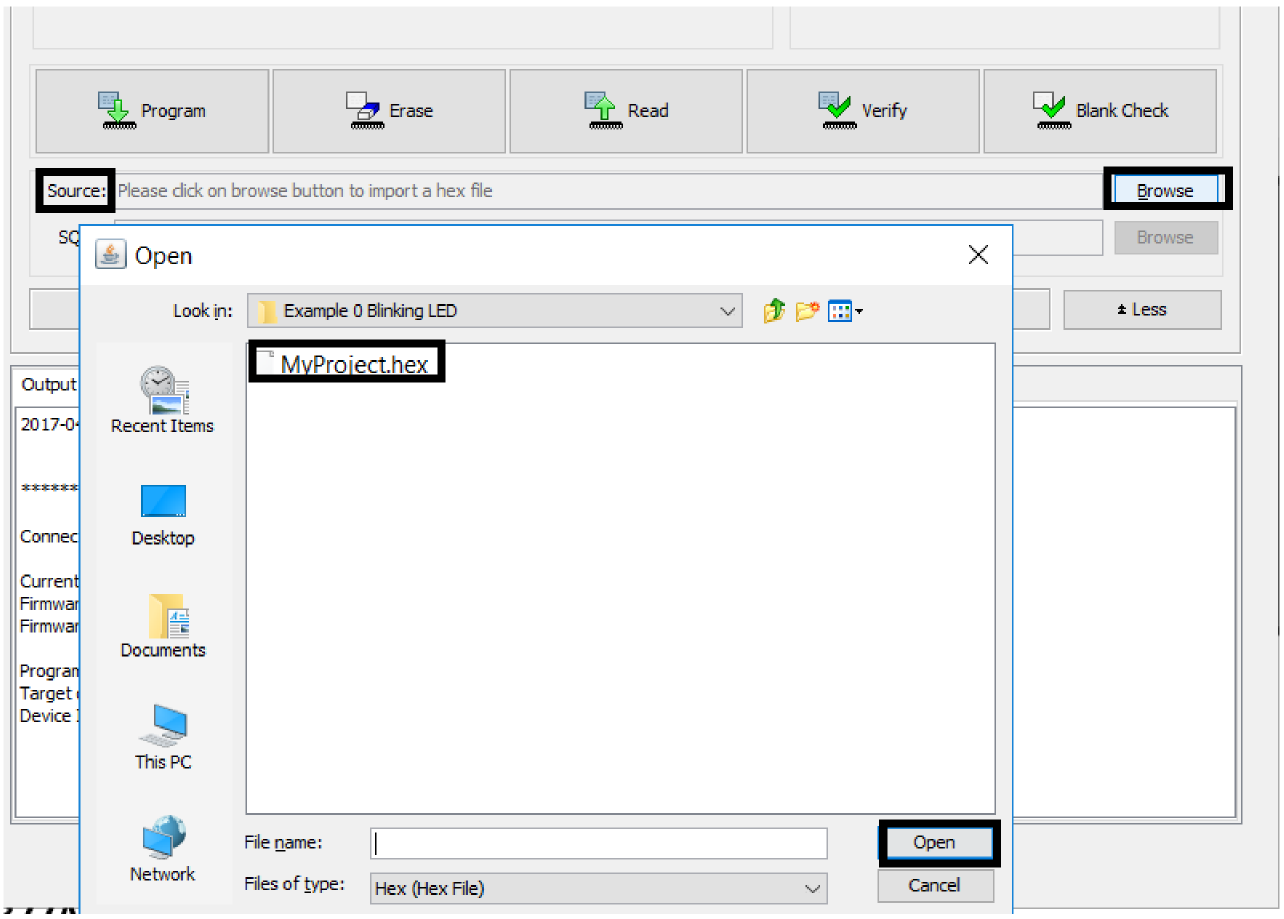
If the output displays:

Target device PIC16F877A found. Then your connection is right.



```
Output
2017-04-08 18:27:49 +0200 - Completed loading IPE.


*****************************************************


Connecting to MPLAB PICkit 3...

Currently loaded firmware on PICkit 3
Firmware Suite Version.....01.46.14
Firmware type.............Midrange

Programmer to target power is enabled - VDD = 5.000000 volts.
Target device PIC16F877A found.
Device ID Revision = 8
```

- **To start programming (writing) the hex file to the Chip. Press source then navigate to the path of the hex file you'd like to burn and choose it.**



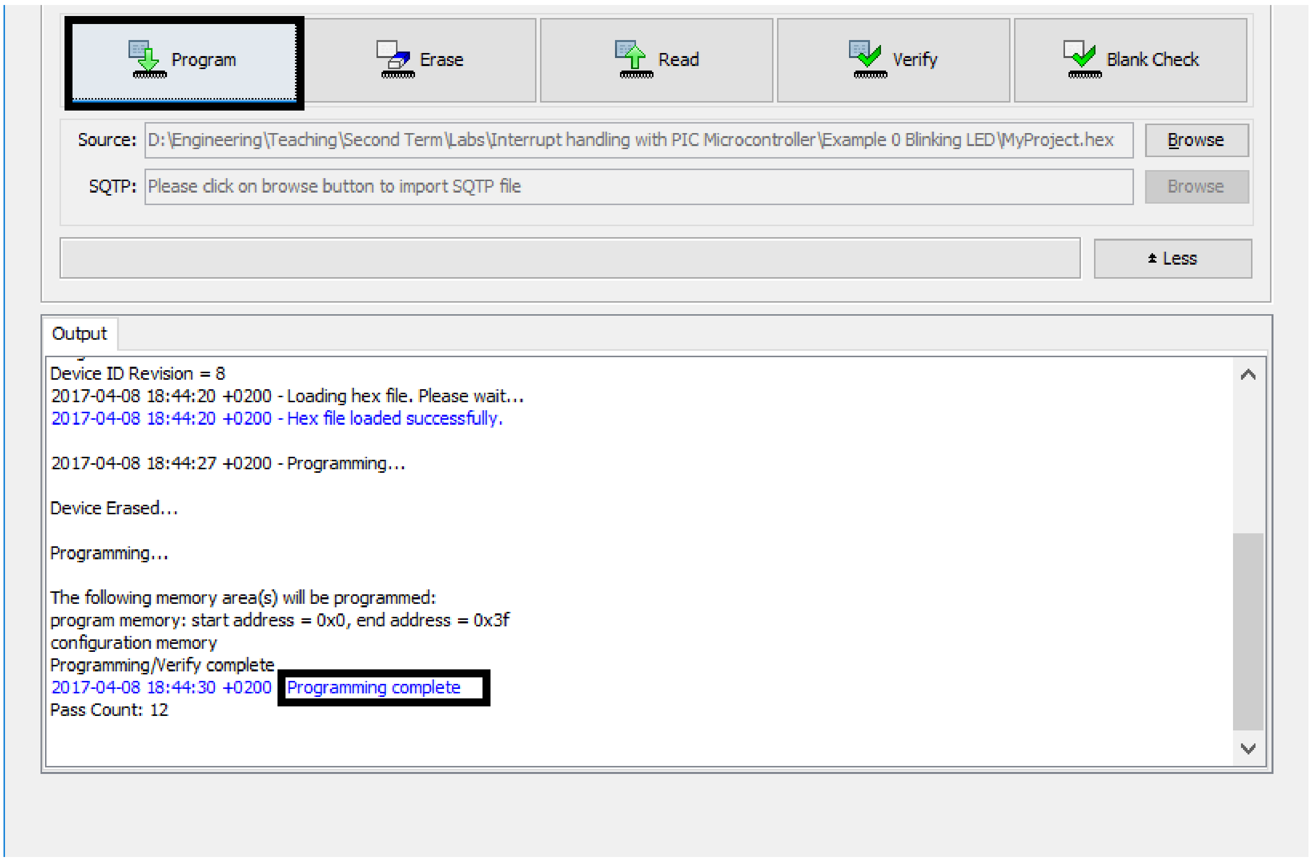
Source: Please click on browse button to import a hex file

Open — Look in: Example 0 Blinking LED — MyProject.hex

File name:

Files of type: Hex (Hex File)

After that press program and wait until the programming is complete.



Source: D:\Engineering\Teaching\Second Term\Labs\Interrupt handling with PIC Microcontroller\Example 0 Blinking LED\MyProject.hex

SQTP: Please click on browse button to import SQTP file

```
Device ID Revision = 8
2017-04-08 18:44:20 +0200 - Loading hex file. Please wait...
2017-04-08 18:44:20 +0200 - Hex file loaded successfully.

2017-04-08 18:44:27 +0200 - Programming...

Device Erased...

Programming...

The following memory area(s) will be programmed:
program memory: start address = 0x0, end address = 0x3f
configuration memory
Programming/Verify complete
2017-04-08 18:44:30 +0200 Programming complete
Pass Count: 12
```

The previous steps could be repeated whenever required.

In the following pages, we shall explore in some detail the interrupts available in the 16f877a chip and make use of them in a real-life scenario.

PIC 16F877A has the following 15 interrupt sources:

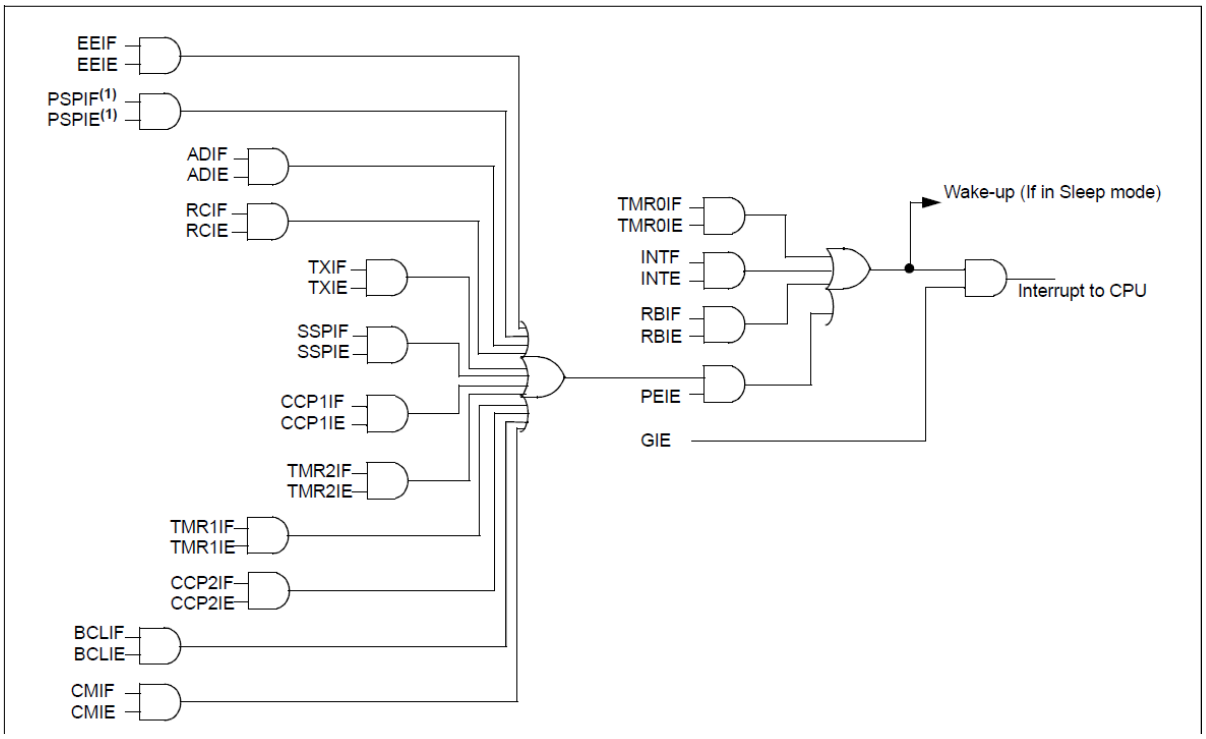
INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

- **External (Enabled/disabled by setting bit <4> bit in Register INTCON) which is named INTE. Occurs when an input changes on PORTB<pin 0>. It sets flag bit, INTF which is no 1 in register INTCON.**
- **Timer 0 (Enabled by setting bit TMR0IE no <5> in Register INTCON)**
- **RB Port Change (Enabled/disabled by setting bit <3> bit in Register INTCON) which is named RBIE. Occurs when an input changes on PORTB<pin 4 to 7>. It sets flag bit, RBIF which is no 0 in register INTCON.**
- Timer 1 (Enabled by setting bit 0 in Register PIE1)
- Parallel Slave Port Read/Write (used when interfacing with another µp Enabled by setting bit <7> in Register PIE1)
- A/D Converter (Enabled by setting bit <6> in Register PIE1)
- USART Receive (Enabled by setting bit <5>) in Register PIE1)
- USART Transmit (Enabled <4> by setting bit in Register PIE1)
- Synchronous Serial Port (Enabled by setting bit <3> in Register PIE1)
- CCP1 (Capture, Compare, PWM) (Enabled by setting bit <2> in Register PIE1)
- CCP2 (Capture, Compare, PWM) (Enabled by setting bit <0> in Register PIE2)
- TMR2 to PR2 Match (Enabled by setting bit 2 named TMR2ON in Register T2CON)
- Comparator (Enabled by setting bit in Register)
- EEPROM Write Operation (Enabled by setting bit <3> in Register PIE2)
- Bus Collision (Enabled by setting bit <3> in Register PIE2)

In this Lab, the main focus would be on the 1st and the 2nd interrupt. The 3rd one , which is also in Bold, has similar working concept as the first one.

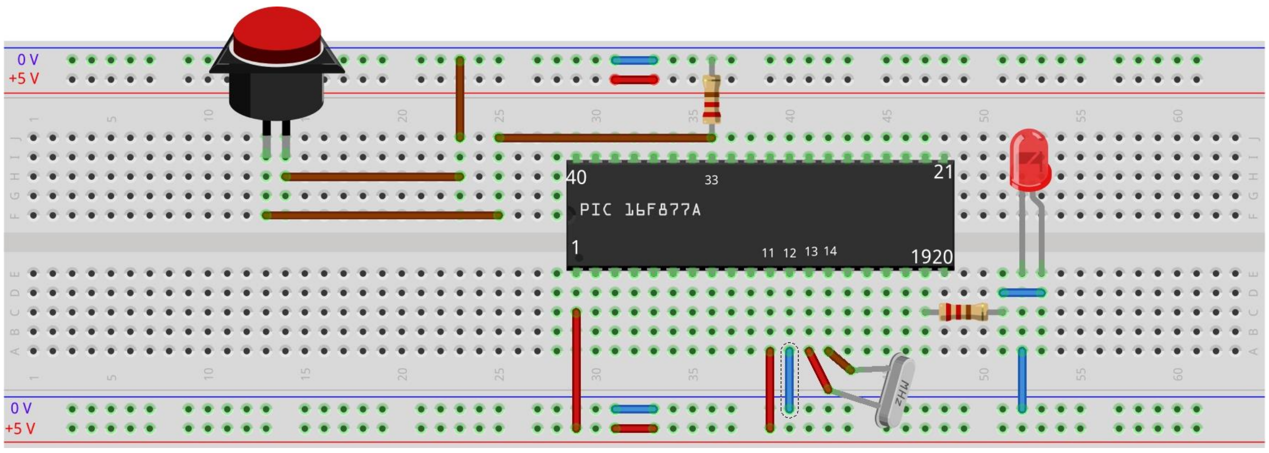
Here’s a Full view of how an interrupt occurs in a PIC16F877A IC:



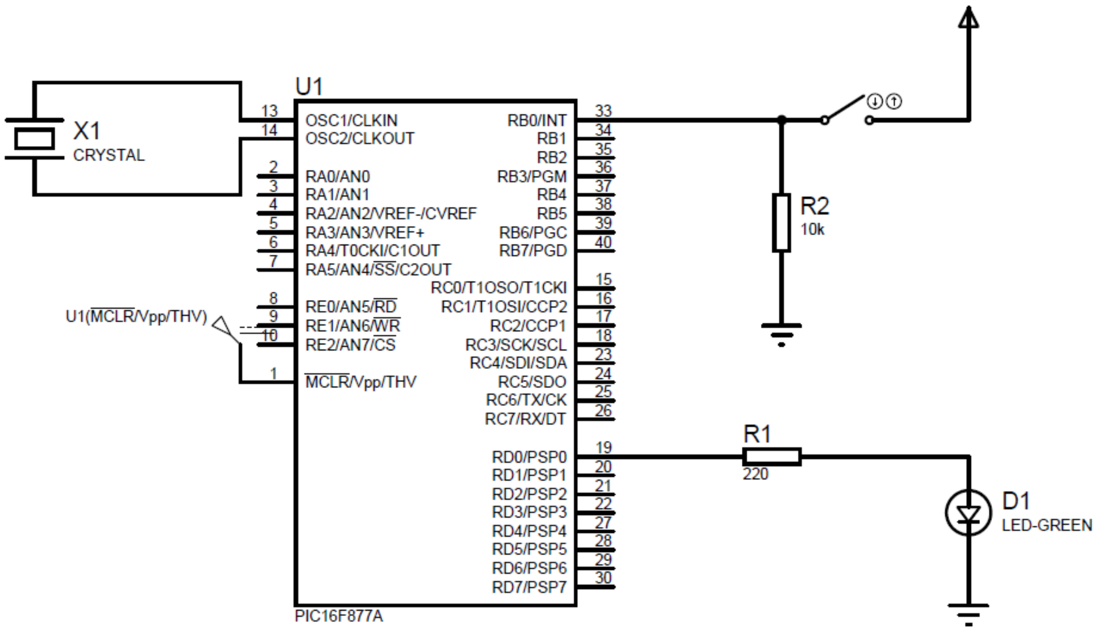
Lab 8 1st figure

Example 1: Implementing an external interrupt

Design a system that toggles a led based when an External Rising edge interrupt occurs. (Hint use External Interrupt)

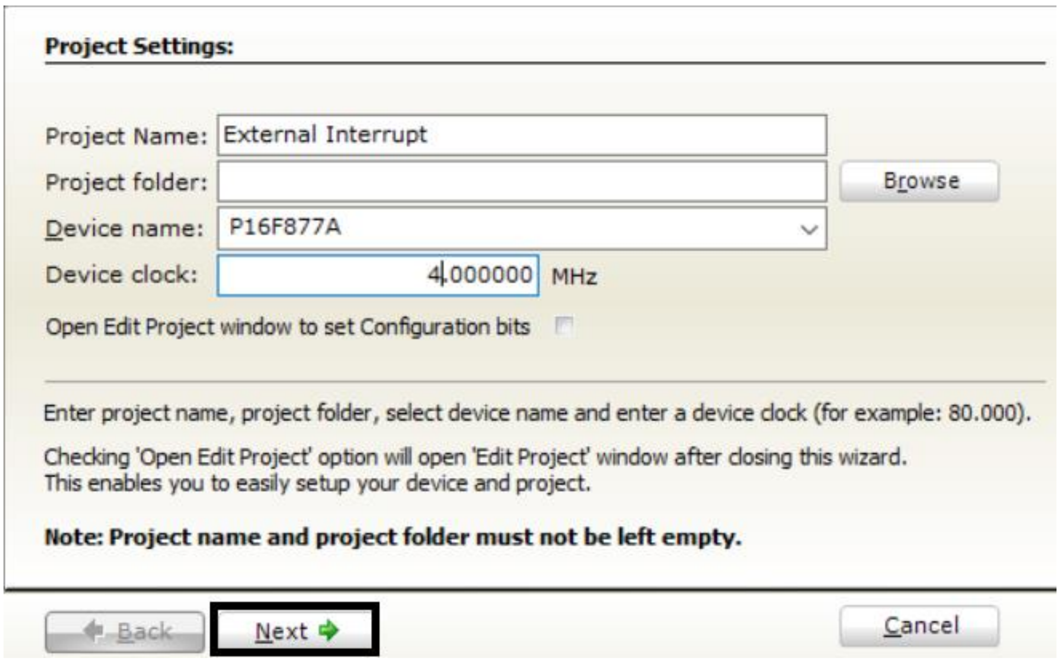


https://youtu.be/I50_92KFlvQ



Lab 8 2nd figure

Open mikroC PRO and set the Project settings with appropriate values:



Project Settings:

Project Name: External Interrupt

Project folder:

Device name: P16F877A

Device clock: 4.000000 MHz

Open Edit Project window to set Configuration bits ☐

Enter project name, project folder, select device name and enter a device clock (for example: 80.000).

Checking 'Open Edit Project' option will open 'Edit Project' window after closing this wizard.
This enables you to easily setup your device and project.

Note: Project name and project folder must not be left empty.

Continue with default settings. Write the following code:

```
void main() {
  TRISB.B0=1;   //  set pin B0 as input
  TRISD.B0=0;   //  set pin D0 as output
  PORTD.B0=0;   //  initial output of D0 is LOGIC LOW
  GIE_bit=1;    // This bit Enables Global Interrupts
  INTE_bit=1;   // This bit enables the external interrupt
}
void interrupt(void)
{
        if(INTF_bit==1){
                INTF_bit=0;   // To stop recursive interrupt execution
                PORTD.B0=~PORTD.B0;  // Toggle execution
        }
}
```

Let’s discuss the Execution of this code. The program starts Execution from the main function. The first line is

TRISB.B0=1

It literally means tristate of pin B0 is enabled. As know from logic the tristate (third state) is the Hi-Z state which is needed when the pin B0 works as an input to avoid the loading effect and read the input accurately.

TRISD.B0=0

This line means that Pin D0 doesn’t have a third state which means it can only be either high or low (output).

PORTD.B0=0

Initial output state of D0 is output LOW.

GIE_bit=1

This bit value in the INTCON register is responsible for enabling (when set with 1) or disabling (when reset) all the other interrupts (lab8 1st figure).

INTE_bit=1

This bit value is responsible for enabling external interrupts.

void interrupt(void)

The default interrupt handler and is executed whenever an interrupt occurs regardless of its type. That’s why we need an if statement to check for its type when there are multiple interrupts.

if(INTF_bit==1)

This checks if the flag INTF_bit is set (with 1) which means that an external interrupt at pin B0 has occurred (in this case a rising edge).

INTF_bit=0

If we forget to write this line the interrupt function would keep executing indefinitely.

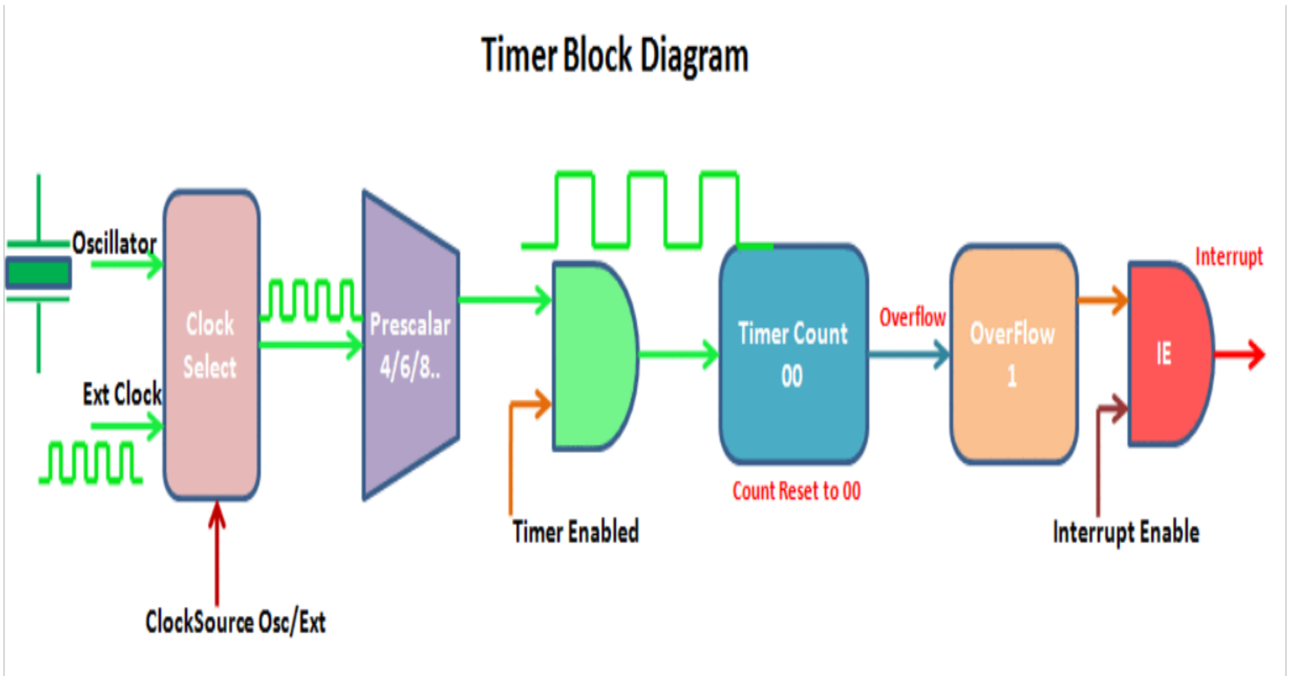
PORTD.B0=~PORTD

This line is to toggle the led (change the state of the led either on to off or off to on).

Example 2: Implementing a timer interrupt

Design a system that displays the number of seconds elapsed since it turns on using a 2*16 LCD screen. (Hint use a TMR0 interrupt).

First we need to understand How this kind of interrupt operates:



Lab 8 3rd figure source: exploreembedded.com

The main idea behind any timer is to wait for a number of ticks that the user needs in order to perform an action there's a built-in oscillator which is T0 clock quarter the external oscillator frequency and (of course four times the time period), Which is a 4 MHZ/4 in this case.

The timer counts from a specific value decided by the user in the TMR0 (8-bit register) to 255 (FF) and the next tick makes an overflow where the interrupt occurs if the GIE is set (the General interrupt register is set to 1 which means that interrupts are activated.

Now all we need to know is how much time does a single tick take and then multiply it by the number of ticks from let's assume 0 to 255.

Tic time = Number of available counts * oscillator time period

In our example the timer frequency is 1 MHZ so the timer OSCilator time period is 1 µs.

This is a very small value and we only have at max 256 tics to enable the interrupt. We need to figure out a way to enlarge the tick time. Luckily for us it's implemented in the PIC16F877A a Pre-scaler value which can increase the tic time which is either 1,2,4,8,16,16,32,64,128,256. Now the tic time could be increased up to 256 Times. To sum up Interrupt occurrence delay can be calculated from the following equation:

Interrupt delay = Oscillator time period*number of tics*prescaler

And since the oscillator here is the external 4 MHZ crystal so the

Timer0 time period = 4 / FOSC

FOSC: frequency of external Crystal oscillator

number of tics = 256 – TMR0 stating value

prescaler value depends on the PSA0 PSA1 PSA2 values which is according to the datasheet

Bit Value	TMR0 Rate
000	1 : 2
001	1 : 4
010	1 : 8
011	1 : 16
100	1 : 32
101	1 : 64
110	1 : 128
111	1 : 256

Bit Value: PS2 PS1 PS0

The general formula is

Timer interrupt delay Time =

$$\frac{Prescalar * (256 - TMR0)}{(FOSC/4)}$$

Now. We want to determine a suitable value for interrupt execution (Assume 8 ms)

So, every 125 interrupts the clock should increment the second time by 1.

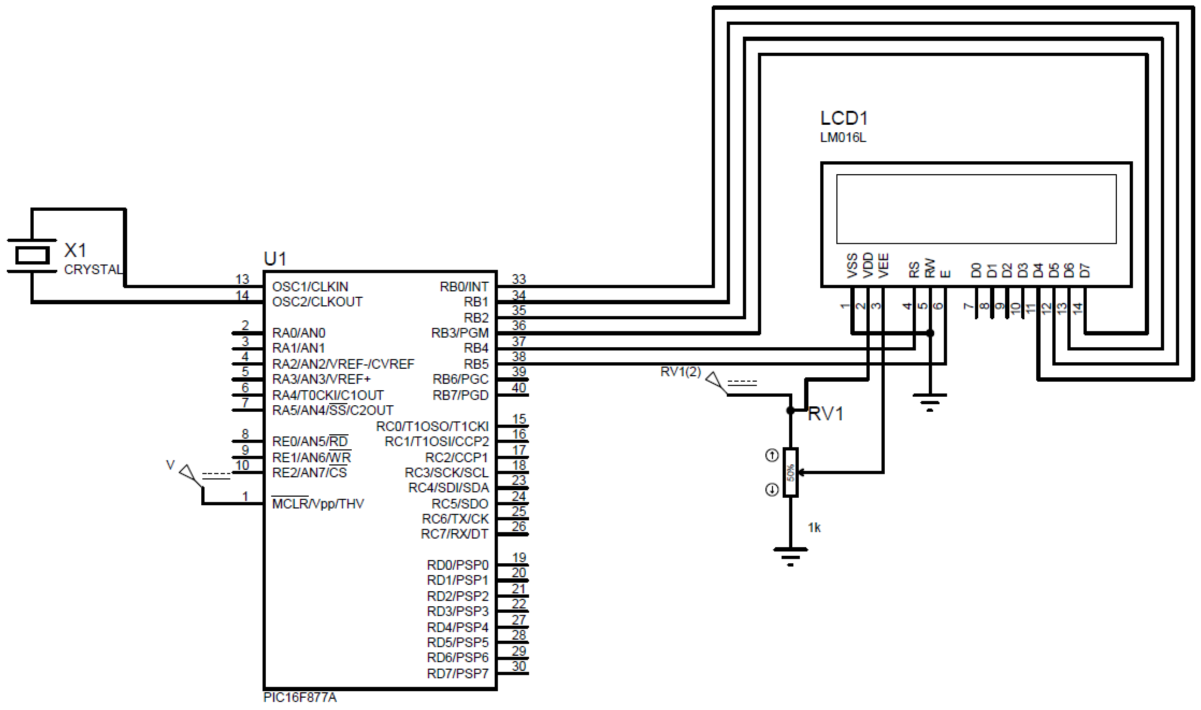
Assume a prescaler of (32) and the Crystal frequency is 4 MHZ. Now solve for TMR0 starting value which is 6 in this case.

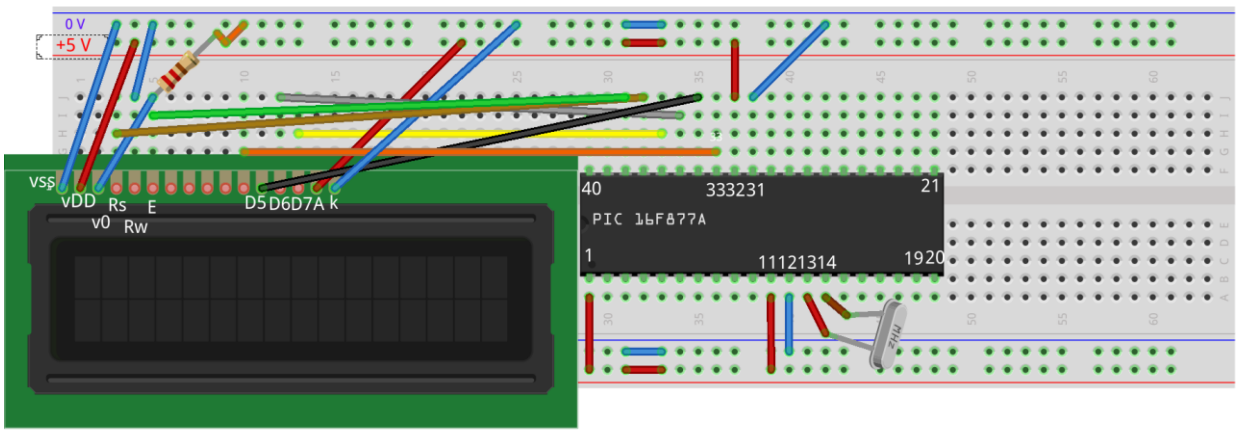
Before the example code, first there are a few register bits we need to know how they work.

TMR0: counts from a specific value (determined by software or zero by default).
T0CS: when reset (Equals 0) the Internal instruction cycle clock where f = FOSC /4
PSA: when reset, the prescaler is assigned to the timer0 module.
PS2, PS1, PS0: (ddd) take values from binary (000) to binary (111) and determine the prescaler value.

Example 2: Implementing a timer interrupt

The schematic design is easy enough just a 2*16 LCD connected to the B pins (for further detail check the MicroC PRO Help menu)



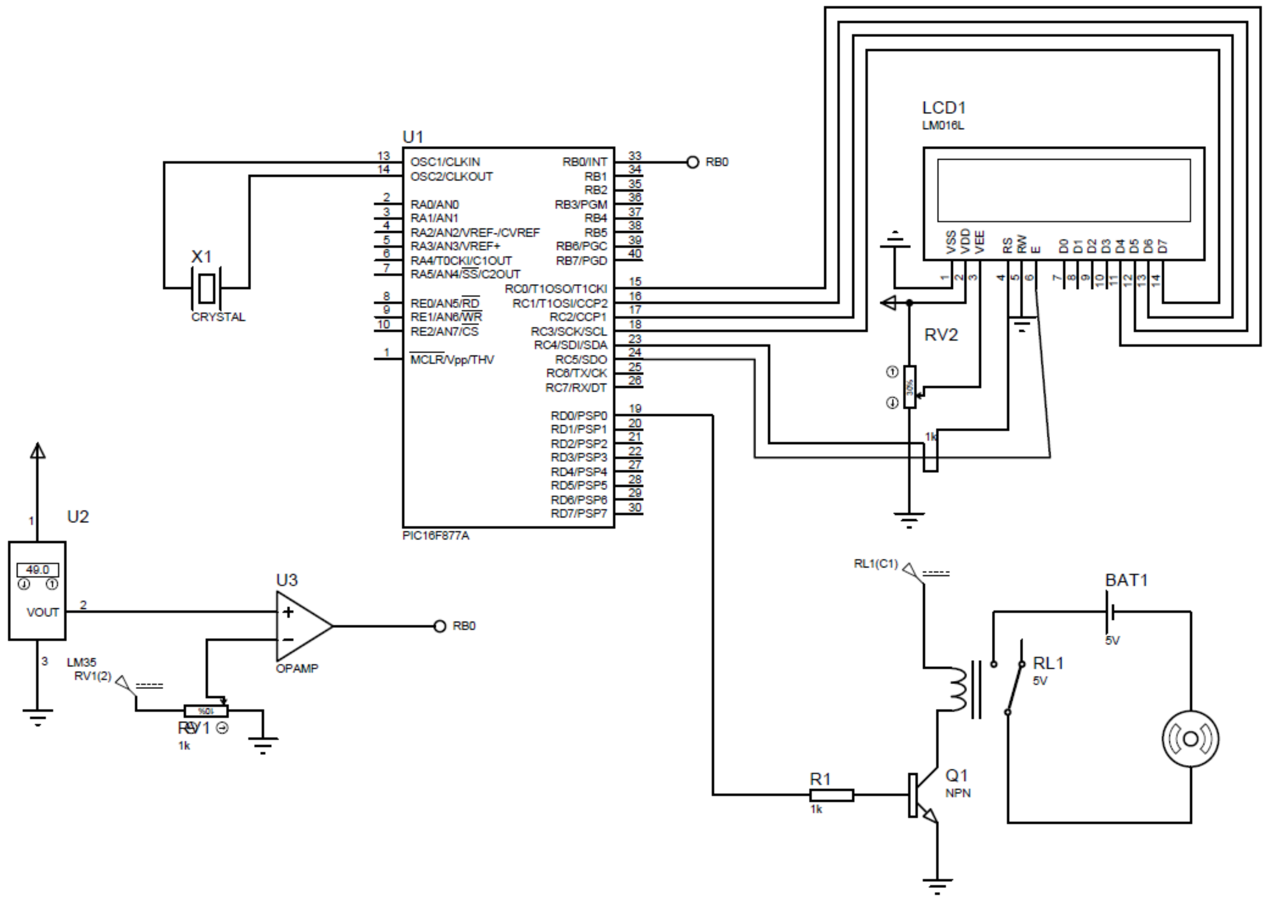


https://youtu.be/HQTTesUVABI

```
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D7 at RB3_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D7_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB0_bit;
 char txt[7];
 int time,counter=0;
void initialize_timer_Interrupt(void){
   TMR0IE_bit=1;         // Enable timer 0 interrupt
   GIE_bit=1;            //Enable Global Interrupt
   T0CS_bit=0;           // Select f/4 clock for the TMR0
   PSA_bit=0;            // Prescaler is assigned to the Timer0 module
   PS0_bit=0;            // Set pre-scaler to 8
   PS1_bit=1;            // PS2,PS1,PS0 = 010
   PS2_bit=0;            //
   TMR0=6;               //counter starting value
}
void main() {
     Lcd_Init();   // Initiate the LCD
     LCD_Cmd(_LCD_CURSOR_OFF);    // Stop the cursor
     Lcd_Out(1,1,"Elapesed Time:");  // Show elapsed time message
     initialize_timer_Interrupt();   // invoke timer interrupt initialization function
     while(1){
       if(counter==500){  // check if the counter reaches 125
         counter=0;     // start counter from the beginning
         time=time++;    // increase time one second
         IntToStrWithZeros(time, txt);  // convert the number of seconds to string
         Lcd_Out(2,1,txt);            // display the number of seconds
        }
     }
}
void interrupt() {       // Interrupt handler
  if (INTCON.TMR0IF==1) {     // check for timer 0 interrupt flag
   counter++;             // increment 1 every interrupt
   INTCON.TMR0IF=0;        // reset the TMR0IF flag
   TMR0=6;                // store 6 in the TMR0 register
  }
}
```

5. Task1 (MONDAY FEB,26,2018)

- Apply the skills you gained in this Lab to make a system that uses the PIC16F877a. An interrupt is executed when the temperature reaches 50 degrees or higher. The ISR sends a signal that operates the fan and keeps working until the temperature is back below 50 degrees. Also, include a timer interrupt that is used to display the elapsed time since the last time the fan cooling system was operating. (Hint you can use the TMR0 interrupt with the INTE at pin B0 or RB interrupt at any pin in B4:7) The schematic should look something similar to this.



- In case you have multiple interrupts explain the priority by which your interrupts are executed in the PIC16F877A.

Task2 (Thursday March,1,2017)

Design a simple irrigation system that irrigates some plant every 3 days for 15 Secs. Use Timer interrupts to adjust the time. Display on the 7-seg the total time spent since the last irrigation.

Task3 (Thursday March,1,2017)

Design a model for a garage security system. The garage motor runs for 4 seconds to open after the right password is entered. Use timer interrupts to adjust the period.

6. Evaluation Criteria:

Lab Attendance:

★

Deliver a working simulation of the solution and fail to answer in the Evaluation Oral Exam questions:

★★

Deliver a working simulation of the solution and answer to the Evaluation Oral Exam questions:

★★★

Deliver a working HW model of the solution and fail to answer in the Evaluation Oral Exam questions:

★★★★

Deliver a working HW model of the solution and answer to the Evaluation Oral Exam questions:

★★★★★

Notes:

* Max number of team members is 5.

** Only members who attend the Evaluation Exam would be graded.

7. Additional material

PIC16F877A datasheet:

microchip.com/downloads/en/DeviceDoc/39582b.pdf

Recommended readings:

Interrupts with PIC µc:	electrosome.com/interrupt-pic-microcontroller
timer interrupt Calculate:	microcontrollerboard.com/pic-timer0-tutorial.html