MPLAB IDE Tutorial

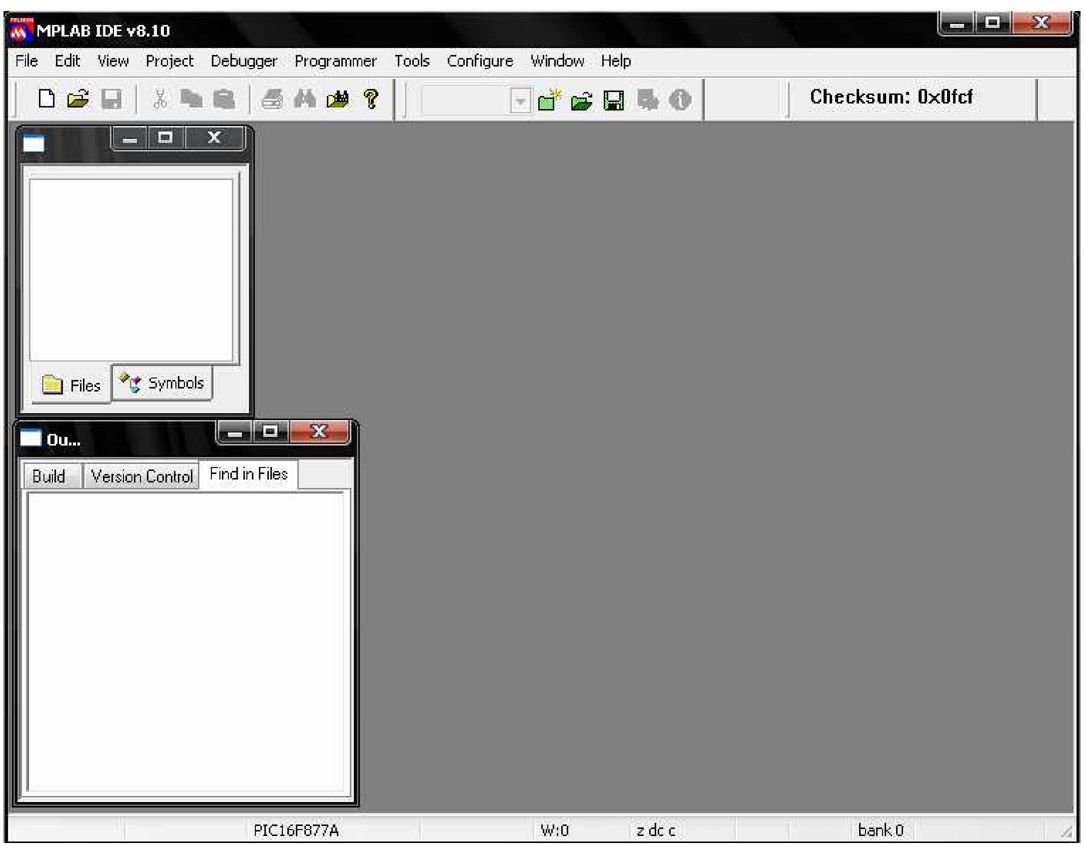
MPLAB IDE (Integrated Development Environment) is used for editing, compiling codes as well as simulating them using an inbuilt simulator.
In this tutorial, we will cover the following steps (specifically to program the PIC16F877A)

1) Setting up a new project
    - Selecting the device
    - Setting up Language Tools
    - Naming the project
    - Adding Required files to the Project

2) Editing & Compiling the Code
    - Viewing Windows
    - Checking the Configuration Bits
    - Locating the main code area
    - Locating the variable definition area
    - Locating the ISR (Interrupt Service Routine)
    - Building the code

3) Testing the code using the Simulator

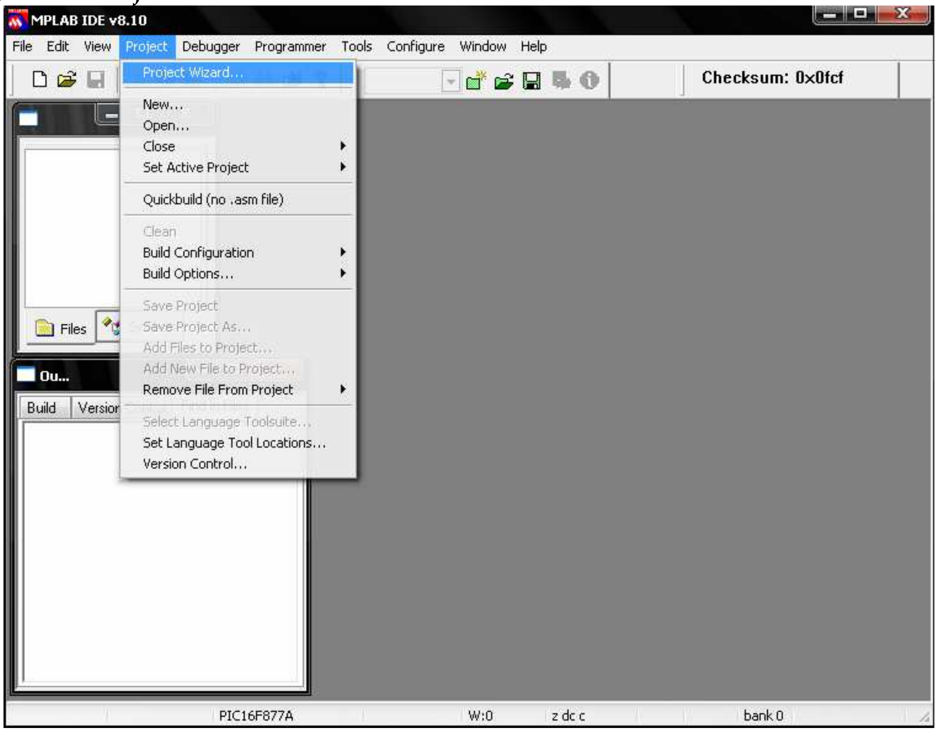
MPLAB IDE when opened, looks like this:



Step 1) Setting up a new project

a) Selecting the Device

Go to: ***Project -> Project Wizard*** as shown

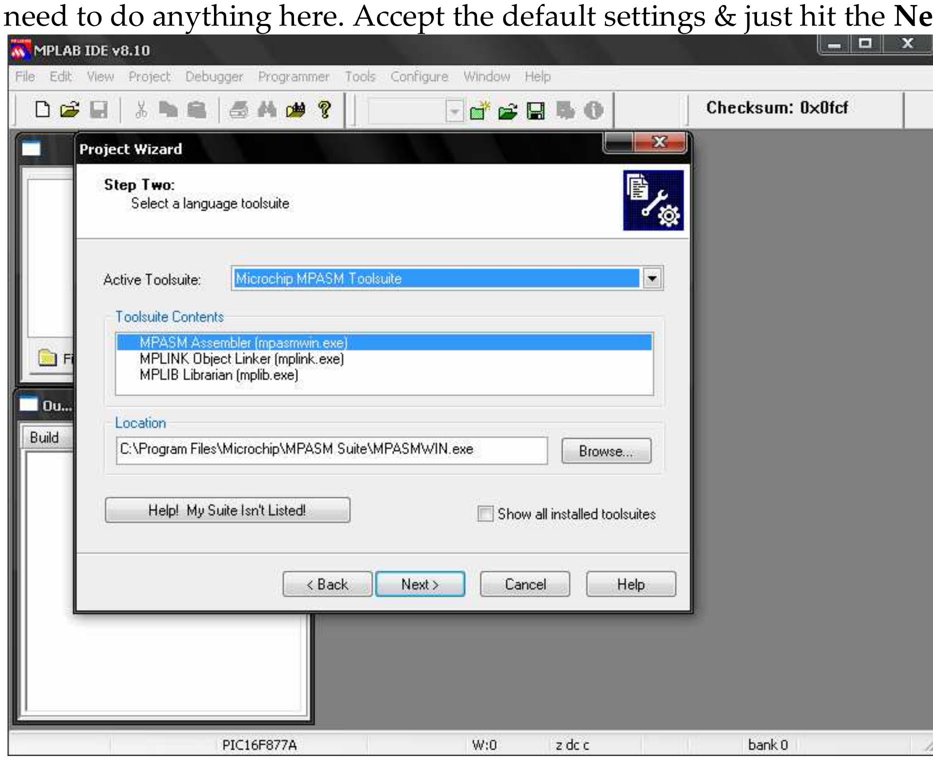


- Click **Next** on the Welcome Screen.
- You arrive at a device selection window. Make sure that the Device selected is '**PIC16F877A**' .

  Click on the **Next** Button.

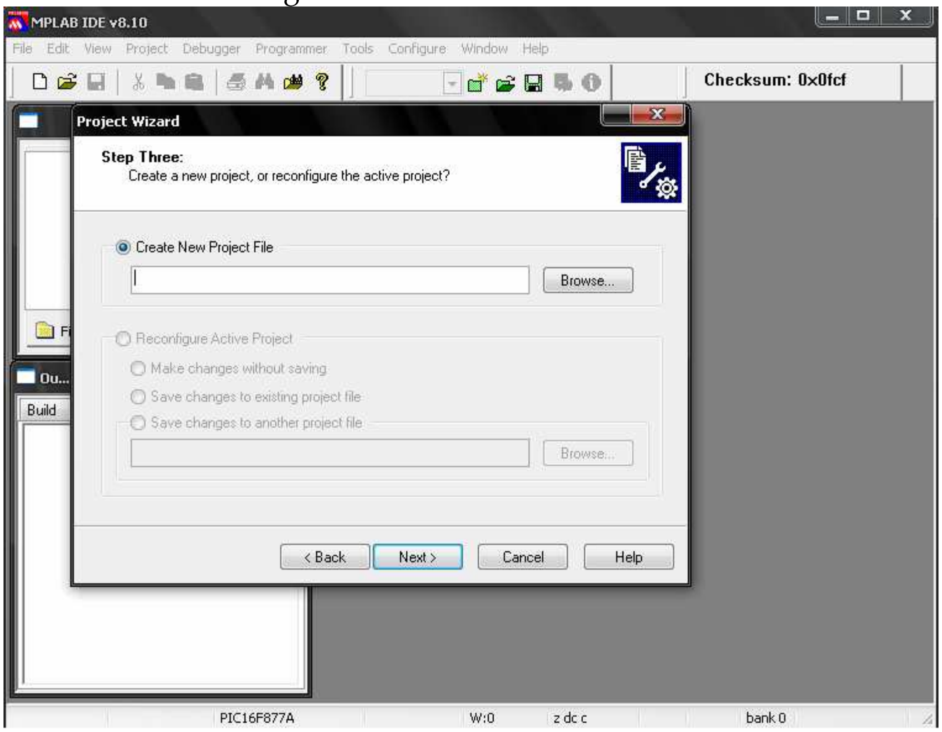
b) Setting up the Language Tools

You don't need to do anything here. Accept the default settings & just hit the **Next** button.



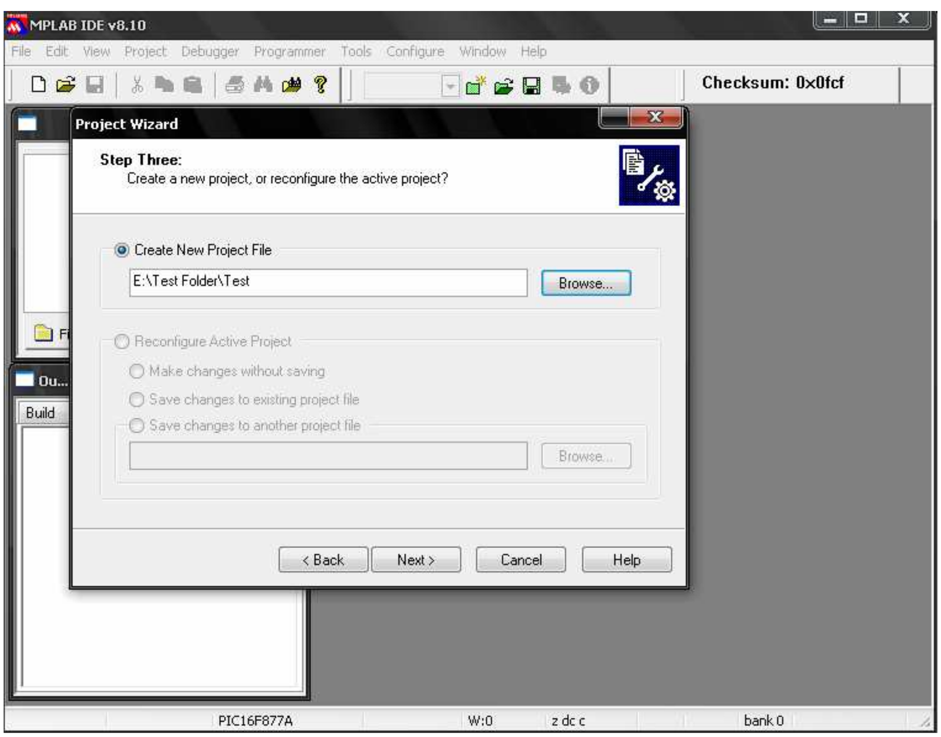
c) Naming the Project

You will encounter the following window:



- Click on the **Browse** Button.
- While inside the Browse Window:
  - Initially **Create a new folder** (with any name) in some location. For e.g. ' Test Folder '.
  - Then click on this folder to go inside it, type any desirable name that you want your program to have (for e.g. Test), and then **click on Save** Button.

- Your window will now look like this. Hit the Next button.

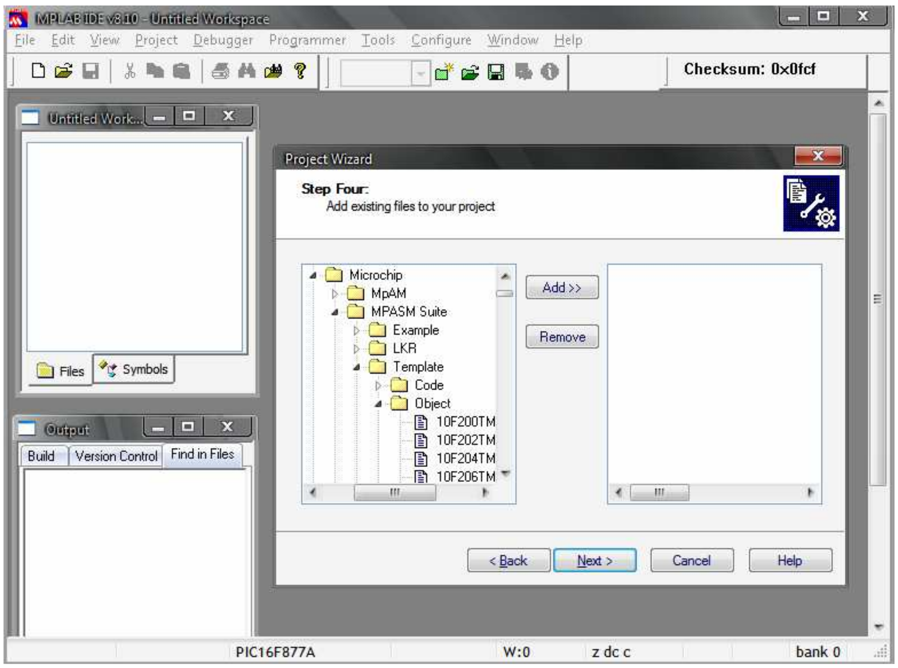


d) Adding required files to the Project

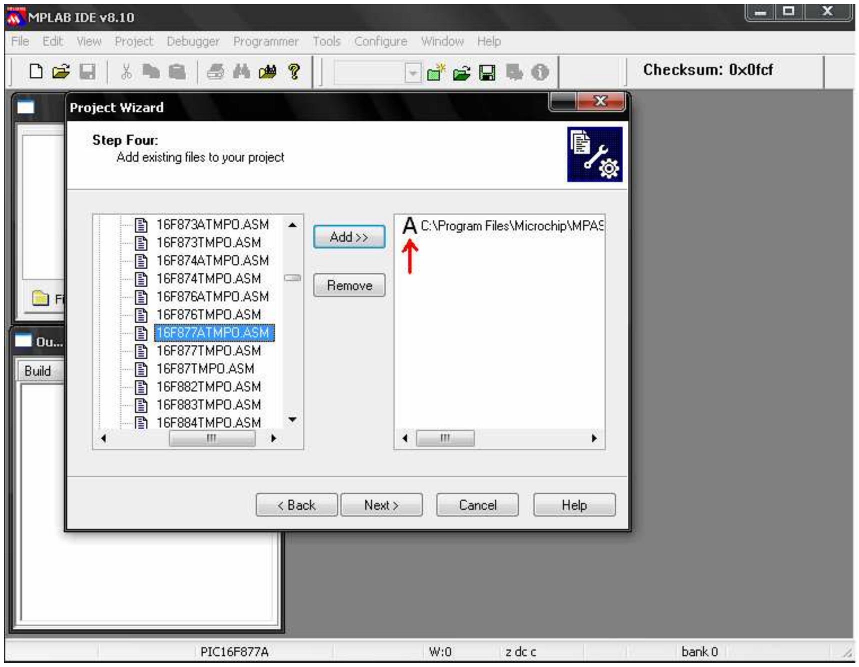
We will need to copy two files to our project.

- You will have arrived at a window that shows a tree type structure of folders on the computer.
  Using the arrows, migrate to the folder:
  - ➢ *C:\Program Files\Microchip\MPASM Suite*

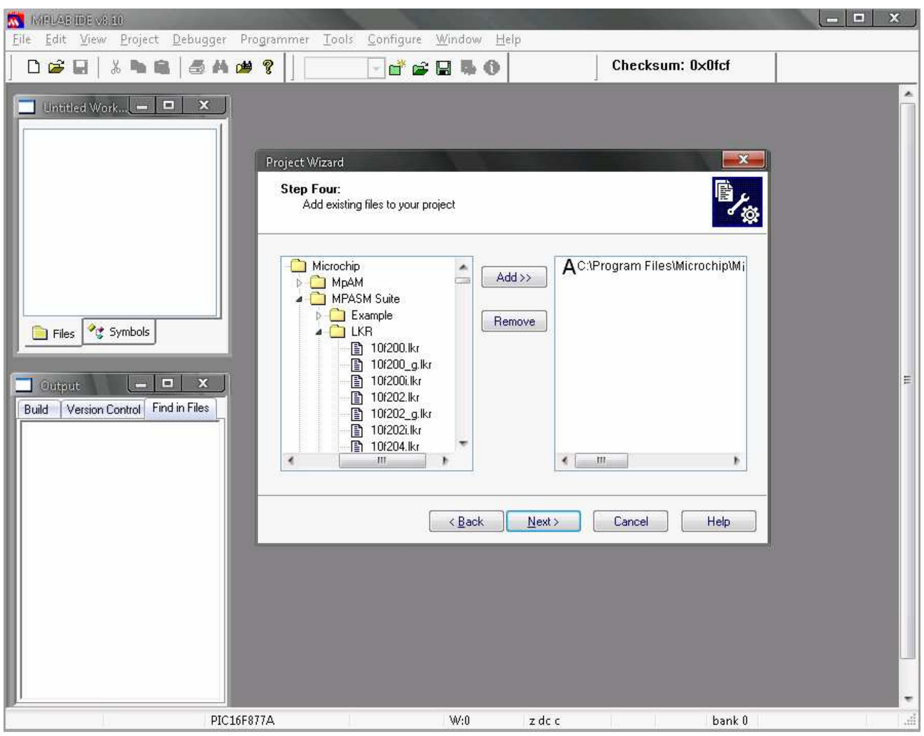
- You will see the following folder structure:



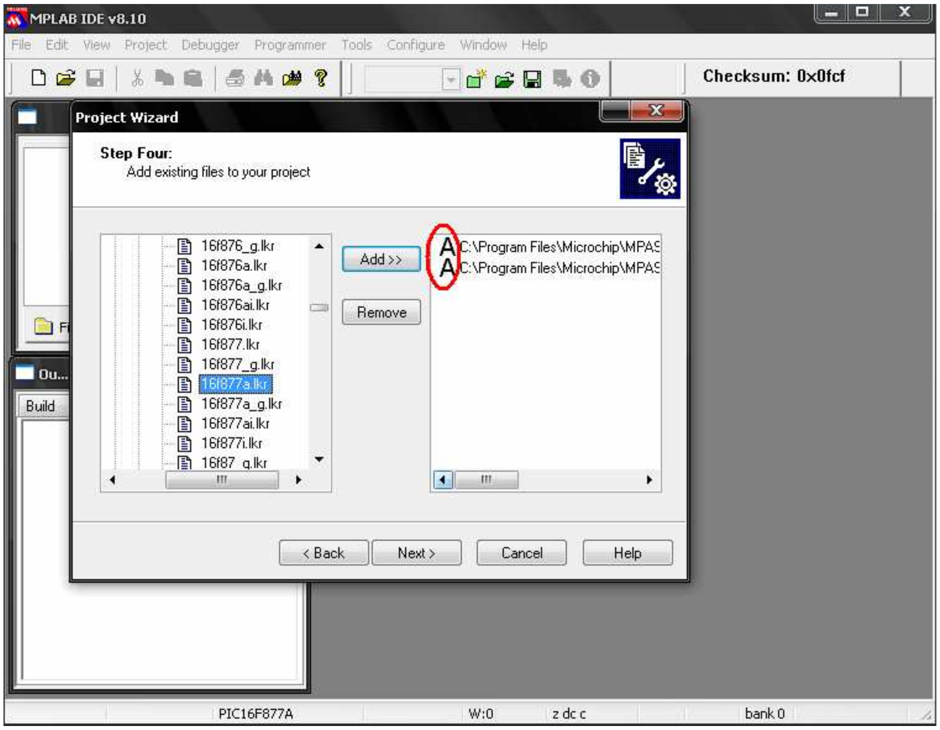
- Firstly, add the template file to your project as follows
  - ➢ Using arrows, enter the folder:
    C:\Program Files\Microchip\MPASM Suite\Template\Object
  - ➢ Find the file : **16F877ATMPO.ASM** ------ (be careful to choose the right file)
  - ➢ Select this file and click on 'Add >>' so that the file enters the right column (with an **A**)
  - ➢ **DO NOT** click on Next.



- Secondly, add a linker file to your project as follows (in the same window):
    - Using arrows, enter the folder:
      C:\Program Files\Microchip\MPASM Suite\LKR



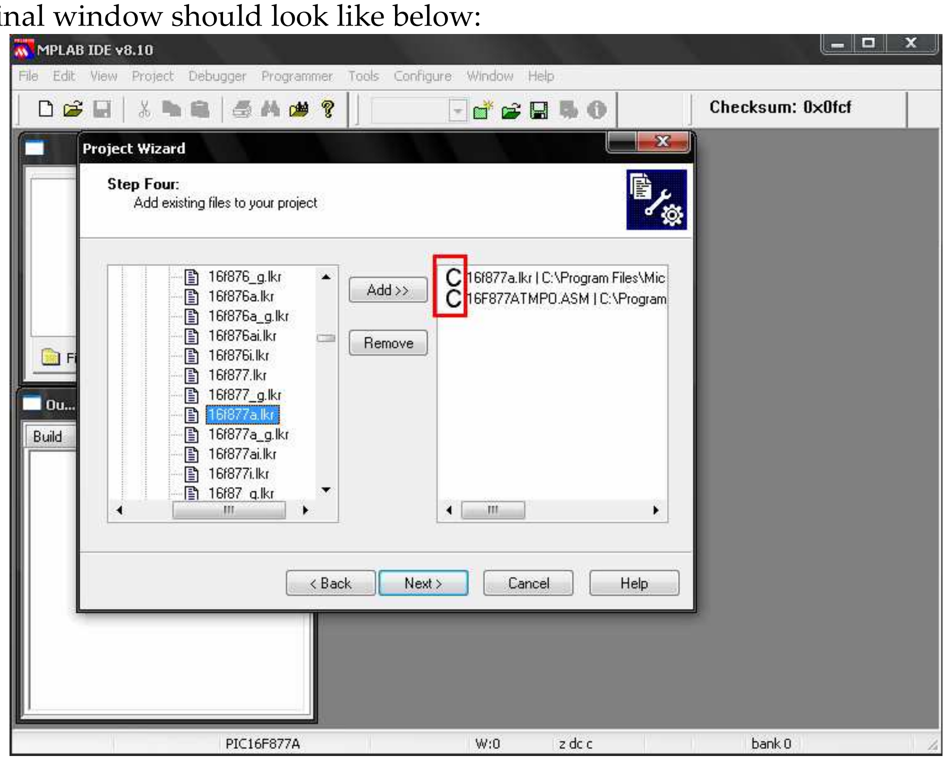
-
    - Find the file: **16f877a.lkr**
      (be careful and **DO NOT CONFUSE** with other similar files like 16f877.lkr, 16f877_g.lkr, 16f877ai.lkr, 16f877a_g.lkr, etc.)
    - Select this file and again click on '**Add >>**' so that it enters the right column (with an **A**)
    - **DO NOT** click on Next.



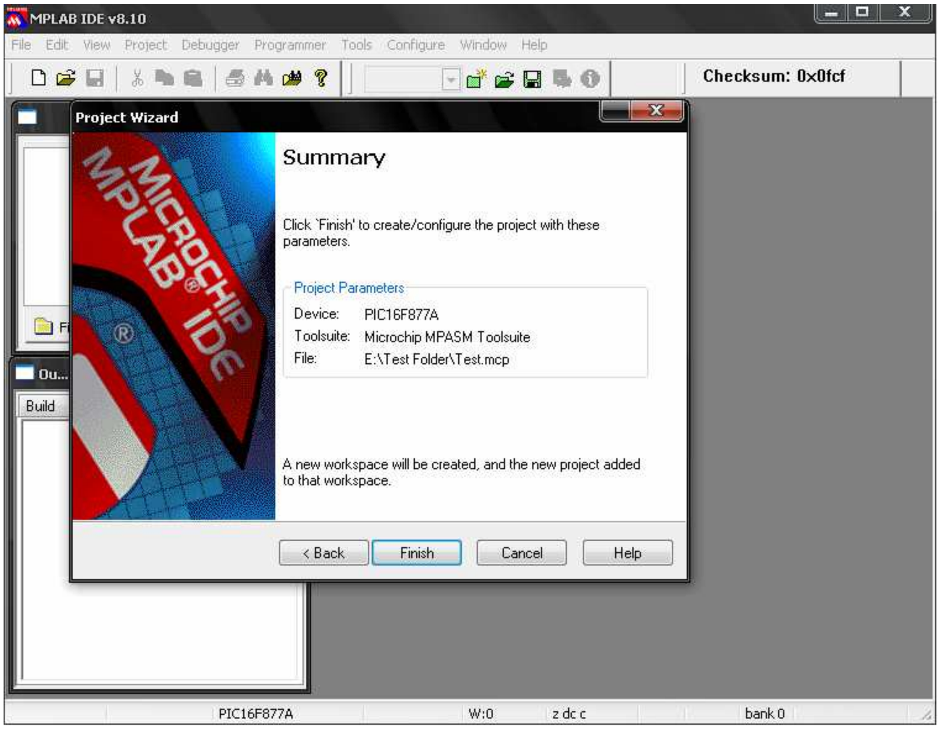
- Move your mouse Pointer over the '**A**'s in the right column, and keep on clicking on them until you see a '**C**' in their place. (as you click on the **A**, you will go through **U**, **S**, and then **C**)

  C --- stands for **copy**. (since we want to copy the files to our project)

- Your final window should look like below:



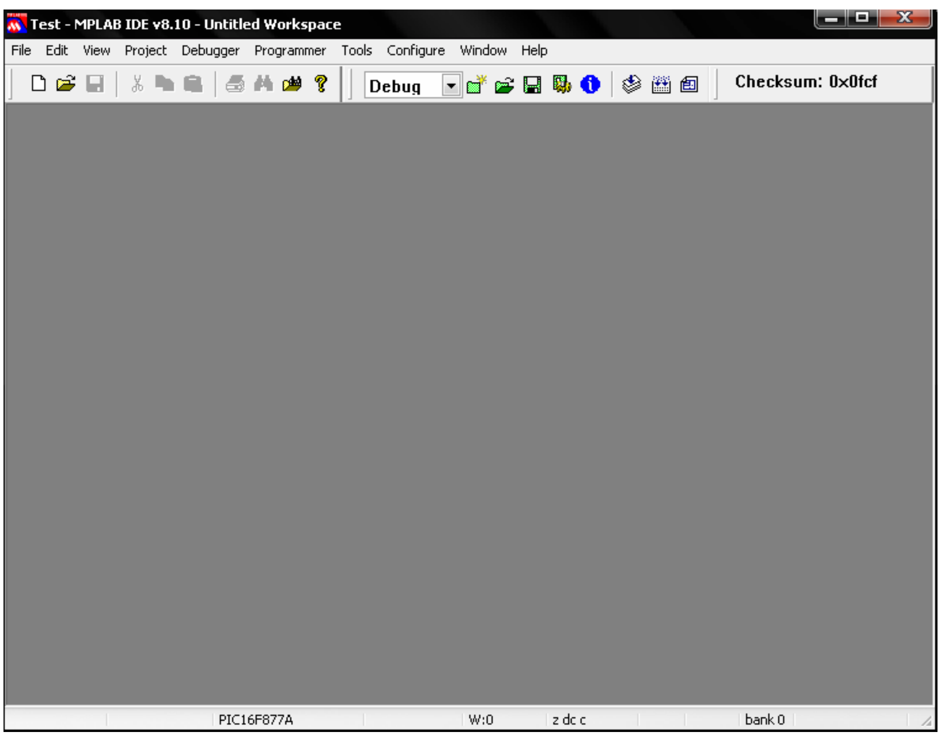
- Now, hit the **Next** Button
- You will arrive at the **Summary** window. Click on '**Finish**'



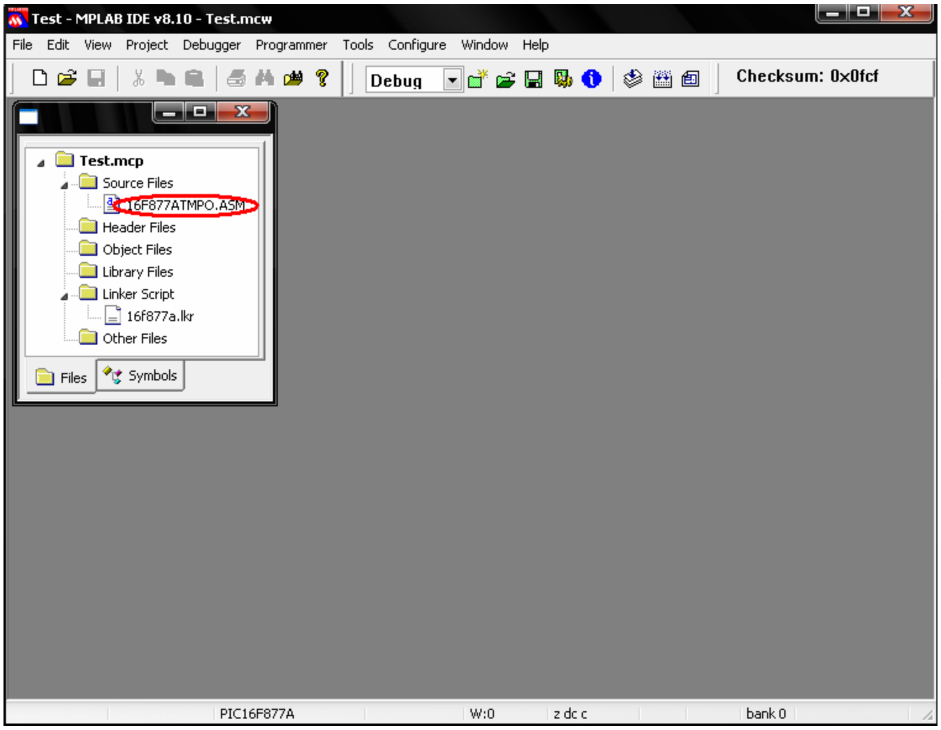
Step 2) Editing & Compiling the code

a) Viewing Windows

You will arrive at seemingly blank window as follows:



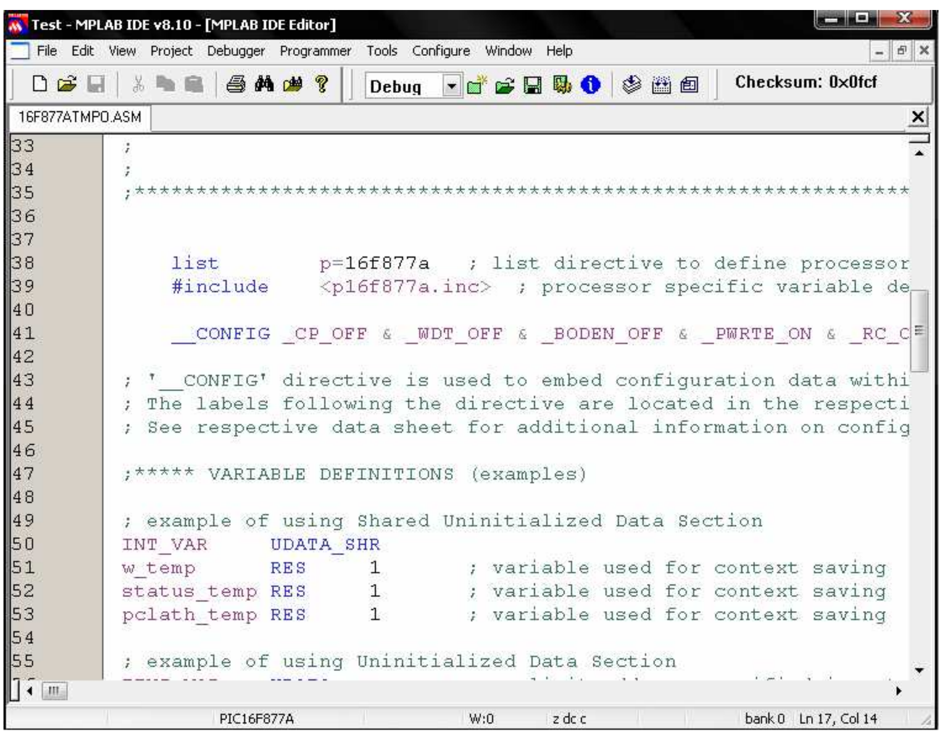
- From the Menu bar above, Go to : ***View -> Project***.
- The project window looks as follows:



- In this small window, you will see the file **16F877ATMPO.ASM**.
- Double-Click on this file to open the Editor window, where you will write your codes.

b) Checking the Configuration Bit-String

- Maximise the Editor window to view the default code in the template.
- Its a pretty long code. Scroll up/down to locate the script that is shown as follows:



- Locate these lines in the code

```
list        p=16f877a        ; list directive to define processor
#include    <p16f877a.inc>   ; processor specific variable definitions
```

__CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _**HS**_OSC & _WRT_OFF & _LVP_**OFF** & _CPD_OFF

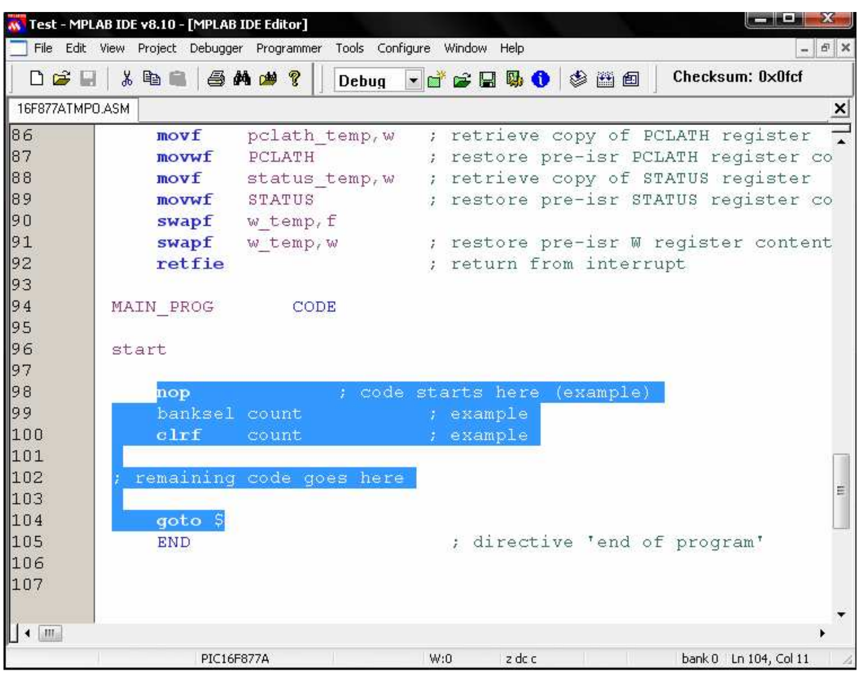
Ensure that this Configuration String in the code is exactly as it is written above. Especially look-out for : **HS**_OSC ... and LVP_**OFF** .

If not, then correct it.

c) Locating the Main code Area:

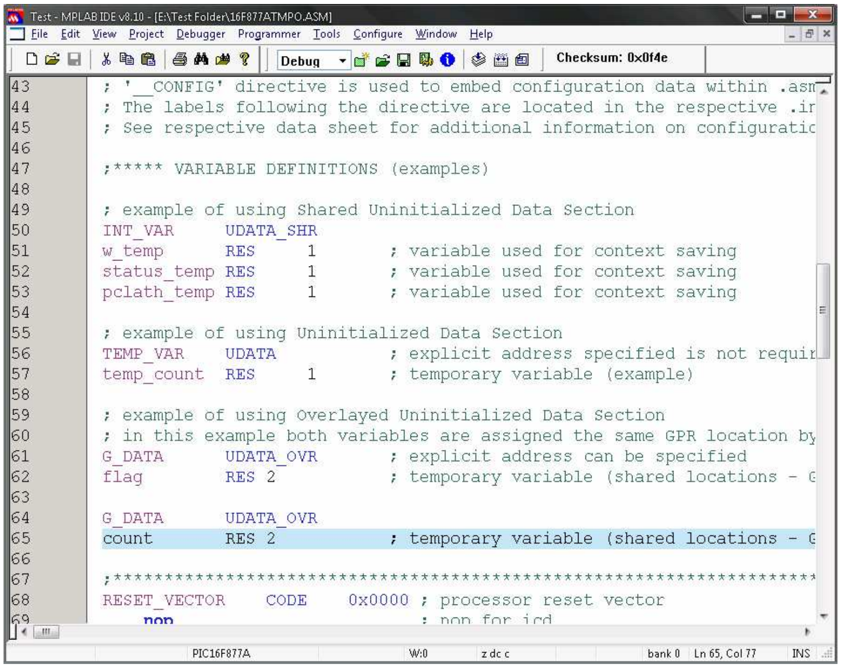
- Scroll up/down in the code to find the line that contains

```
MAIN_PROG        CODE

start

; Whatever code that you wish to run for the experiment, should be written in this place
;entirely (except the Interrupt code).

END
```



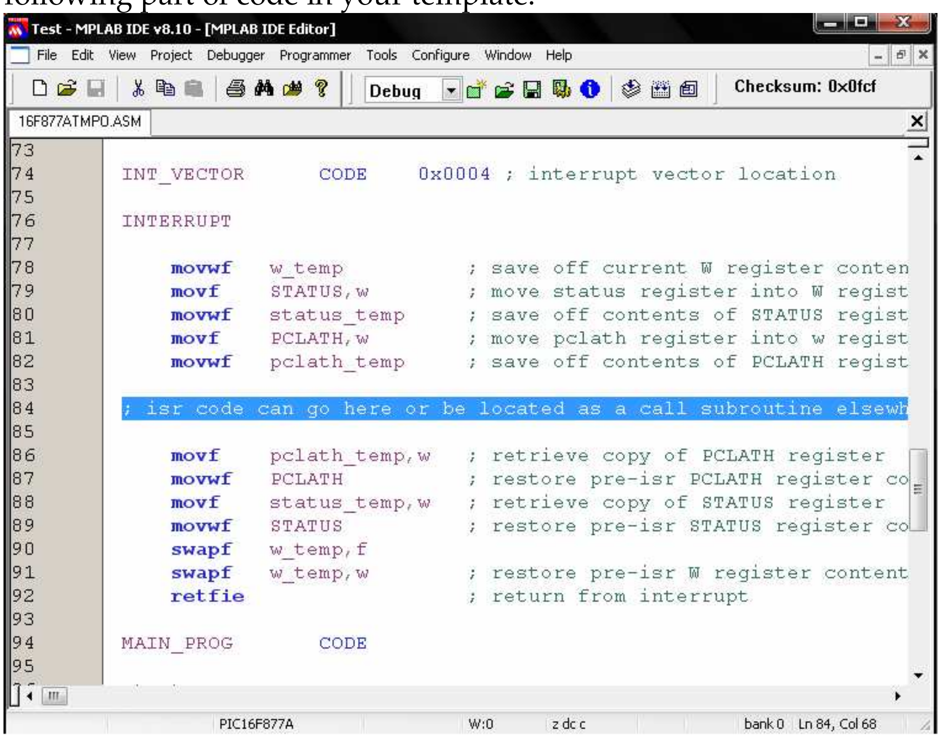
d) Locating the Variable Definition Area

- Pay attention to the part highlighted in this frame. Locate this line in your code.
- The variable (register) '**count**' is defined here. Similarly you can define more variables below it.



e) Locating the ISR (Interrupt Service Routine)

Locate the following part of code in your template.




```
INT_VECTOR    CODE    0x0004     ; interrupt vector location

INTERRUPT

    movwf        w_temp                    ; save off current W register contents
    movf         STATUS,w                  ; move status register into W register
```

```
    movwf     status_temp         ; save off contents of STATUS register
    movf      PCLATH,w                  ; move pclath register into w register
    movwf     pclath_temp               ; save off contents of PCLATH register

; isr code can go here or be located as a call subroutine elsewhere

    movf      pclath_temp,w       ; retrieve copy of PCLATH register
    movwf     PCLATH                    ; restore pre-isr PCLATH register contents
    movf      status_temp,w       ; retrieve copy of STATUS register
    movwf     STATUS              ; restore pre-isr STATUS register contents
    swapf     w_temp,f
    swapf     w_temp,w                  ; restore pre-isr W register contents
    retfie                        ; return from interrupt
```

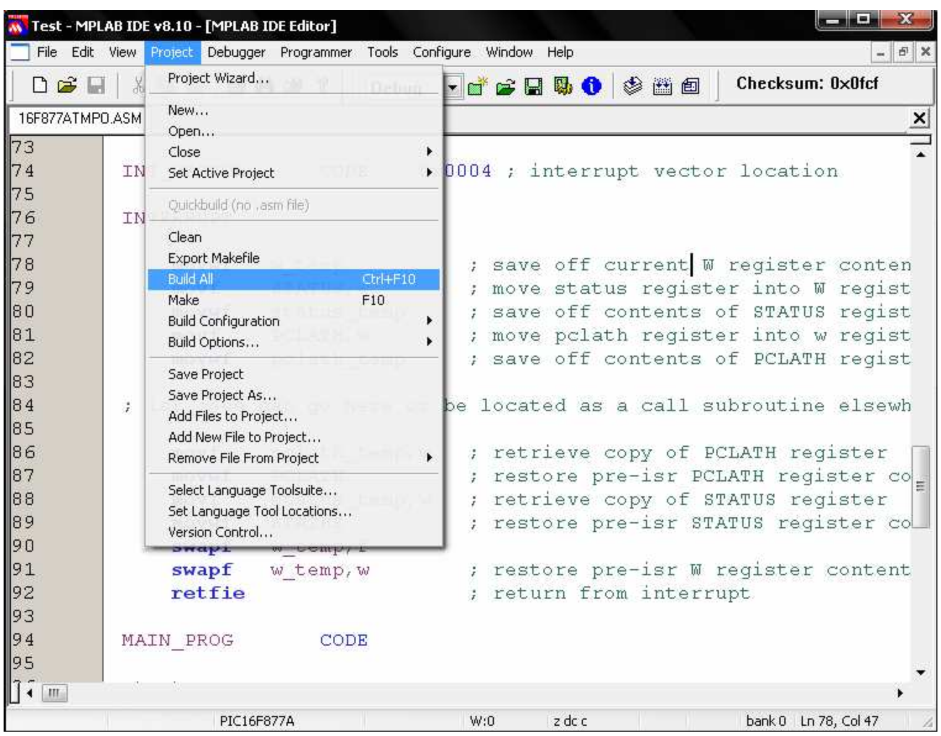
- Whatever code you wish to run when any interrupt is called, needs to be written in the place of the commented line:

  ; isr code can go here or be located as a call subroutine elsewhere

f) Building (Compiling) the Code

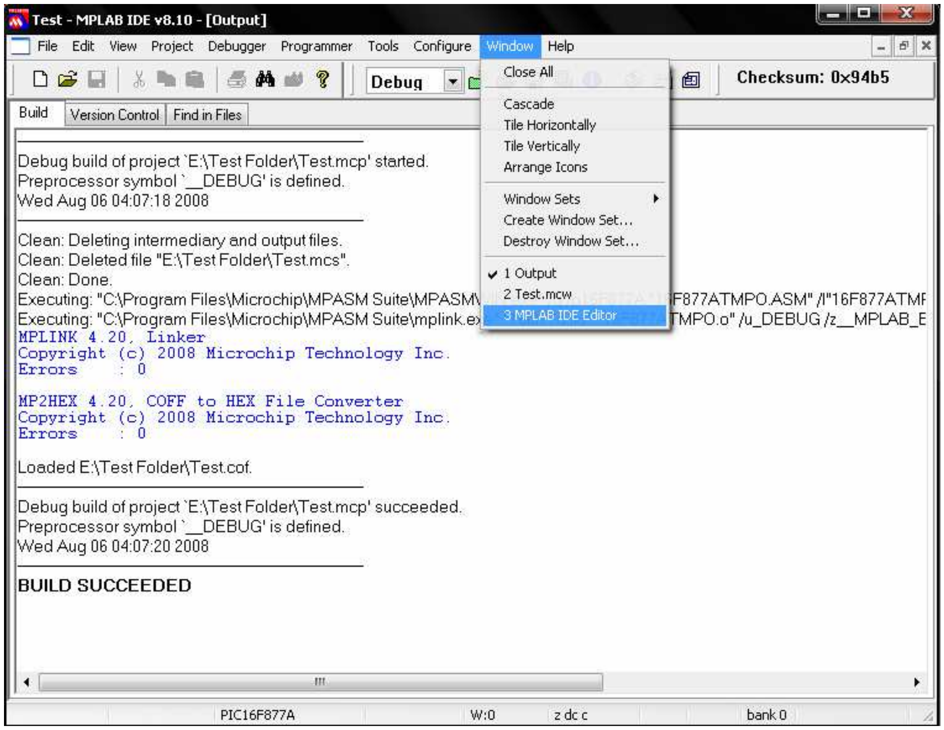
After you finish writing the desired code for a particular experiment, you need to compile the code for errors.

- On the Menu Bar, go to: ***Project -> Build All***



- If your code has no errors in it, you will get the message '**BUILD SUCCEEDED**', in the Output Window.
- To return to the editor window, Go to: ***Window -> MPLAB IDE Editor***.

- From this point onwards, it is convenient to resize the windows, so that you can view all of them simultaneously.

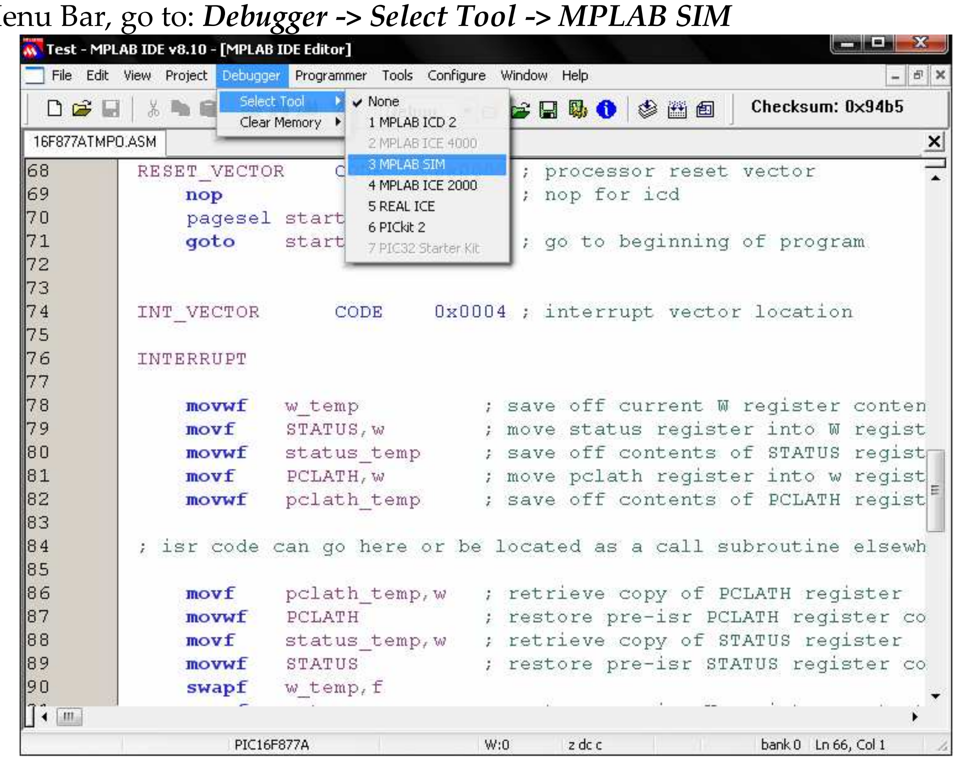


Step 3) Testing the code using Simulator

This is the most important step before you burn your program into the microcontroller IC. The simulator enables you to simulate and check your code if it is working fine, rather than feeding it into the IC and then wondering why the program doesn't work.

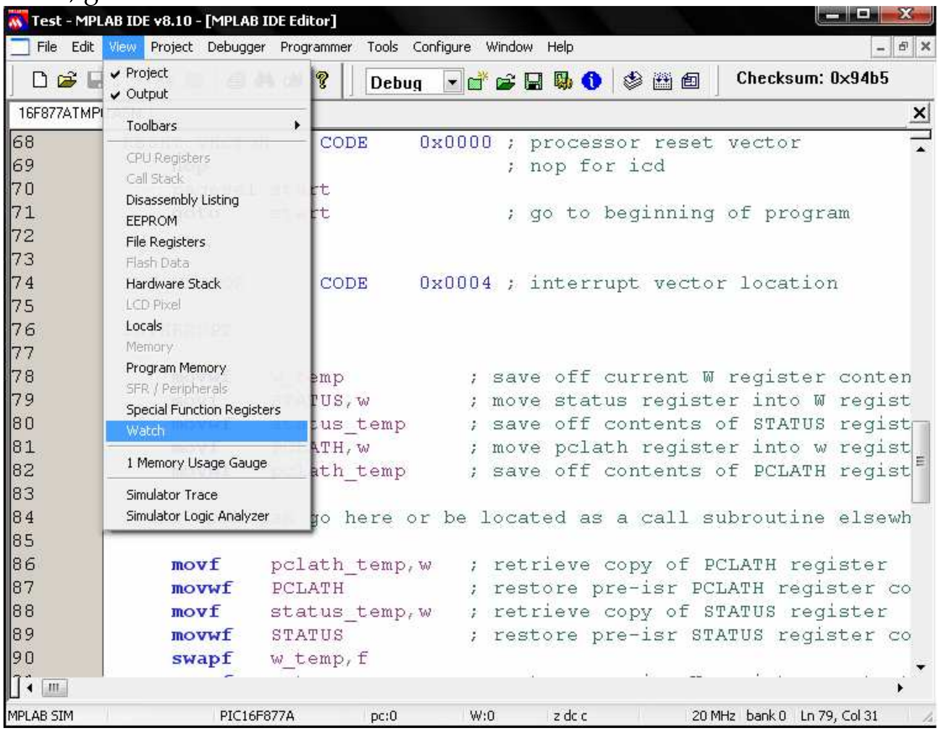
a) Selecting the Debugger (Simulator)

On the Menu Bar, go to: ***Debugger -> Select Tool -> MPLAB SIM***

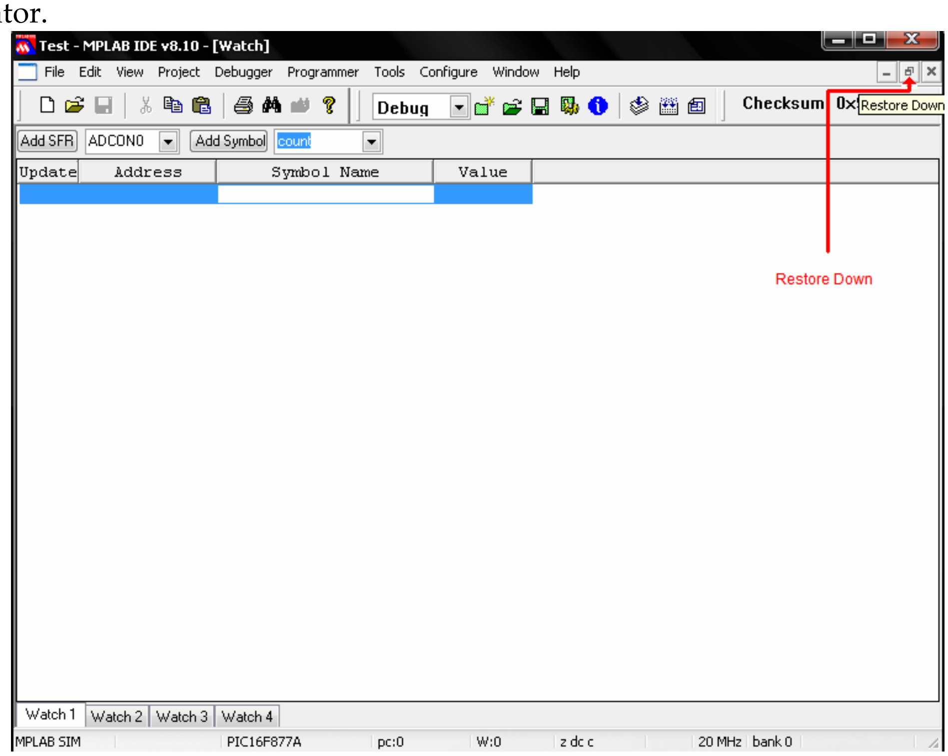


b) Viewing the Watch Window

On the Menu Bar, goto : *View -> Watch*



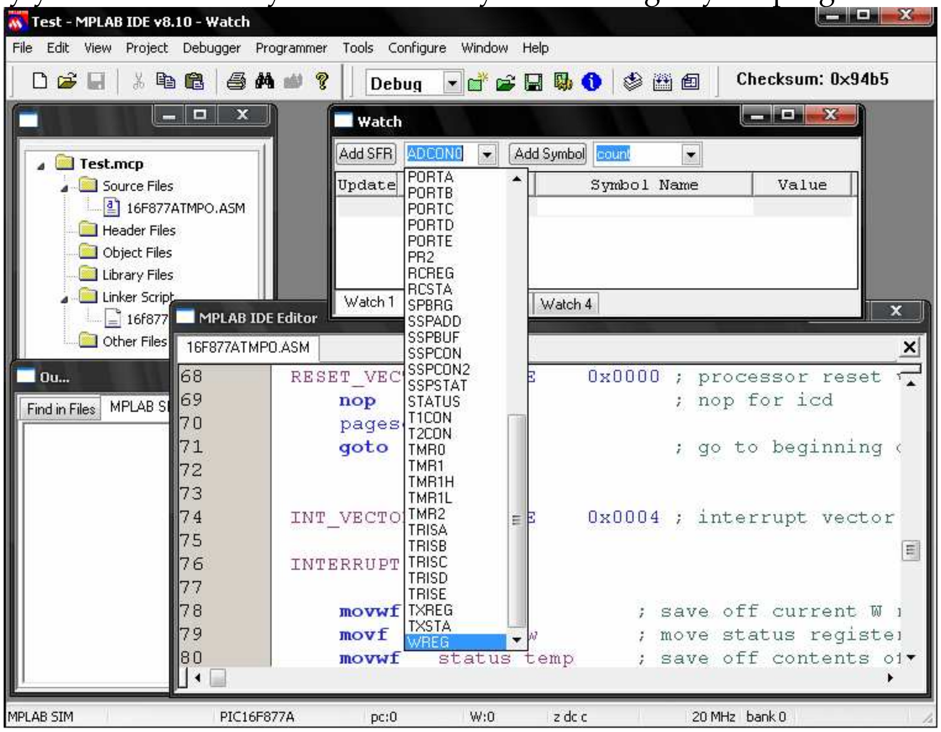
- Resize (**Restore Down**) the Watch window, in order to see all the windows simultaneously. This is necessary in order to watch the program run & output simultaneously on the simulator.



c) Selecting the Registers for a Watch

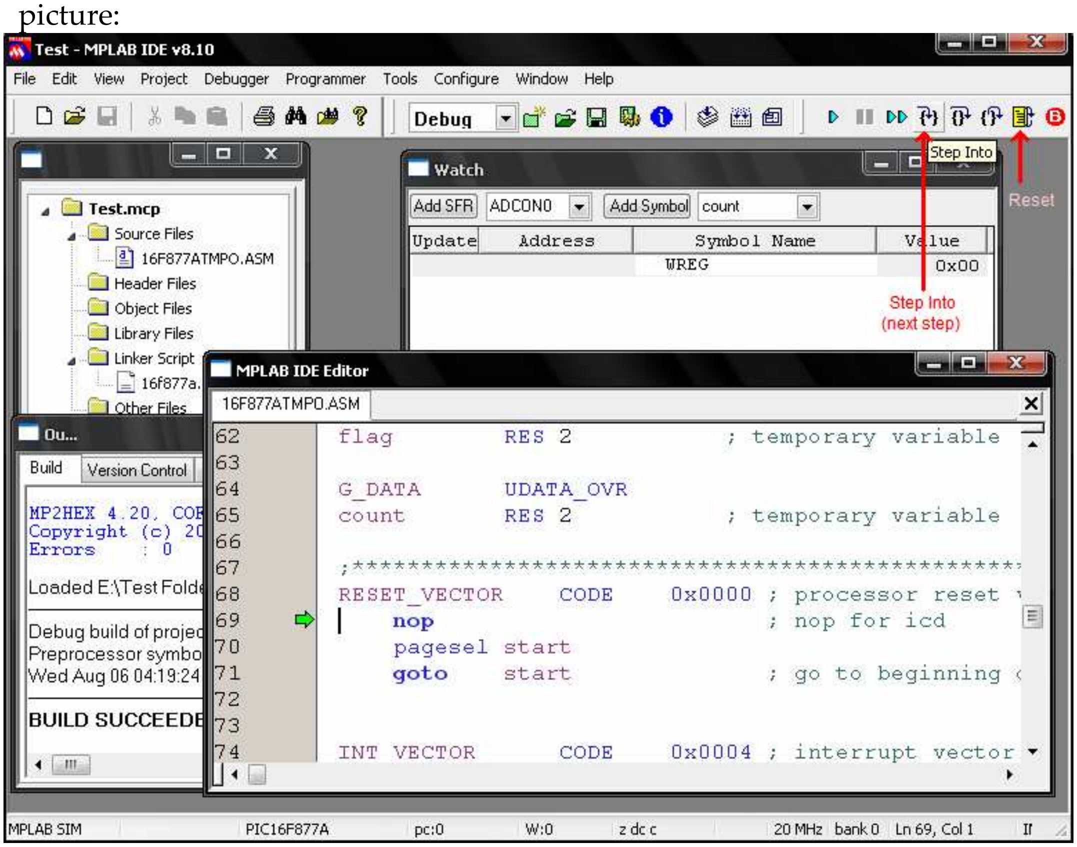
- There are 2 **pull-down lists** in the Watch window. One contains all the SFR's (Special Function Registers)
- Select all those registers that you are using in your program (e.g. TRIS, PORT, WREG, STATUS)

- For e.g. to watch the value of WREG (Accumulator), select WREG from the drop-down list and then click on '**Add SFR**'. Similarly for others.
- Similarly you can watch any variables that you are using in your program. For e.g. count.



d) Checking the Program Sequence

- Your resized windows will look like this. Try to locate the 2 buttons highlited in the picture:



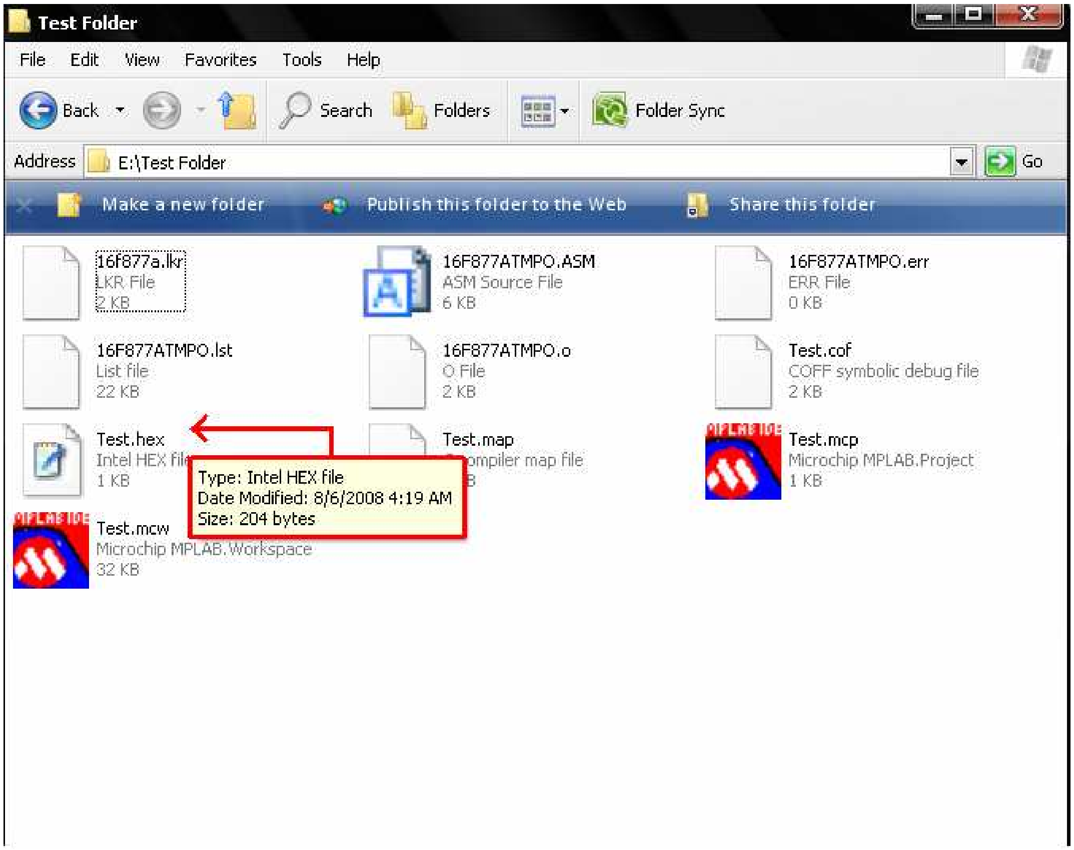
- Also locate the green arrow pointing to some line of the program.

- Make sure you have compiled your program (using **Build All** Option)
- Now, keeping all the registers on watch into view, along with some visible part of the code; press the **'Step Into' Button**. You will see the green arrow shift below.
- Keep on pressing the '**Step Into**' Button to see how each instruction shifts the green arrow (Program Counter) to some position; and at the same time watch if the values of the Registers in the Watch window change as desired.
- Press the '**Reset**' Button to restart the program from the beginning.
- If you have introduced any delays in your program, you will need to reduce the delay cycles in order to see the simulation; and then again increase the delay cycles (time) while feeding the program into the actual microcontroller.

Step 4) Feeding the Program into the Microcontroller

If you are sure that your program is working fine in the simulator, then proceed as follows

- Change the values of Delays if required.
- Compile the program for one final time (in order to ensure that all recent changes have been taken into account)
- Now, in '*My Computer'* locate the actual folder where you have saved your project.
- Inside this folder, you will see many files as follows:



- Locate the file with your program name & a **.hex** extension.
- This is the actual file that needs to be fed to the Microcontroller software. Use this .hex file in the software used for programming the PIC IC.