COVENTRY UNIVERSITY

Faculty of Engineering & Computing

Introduction to the MPLAB Integrated Development Environment (IDE)

Using Assembler files and the MPLAB Simulator

Objectives: To introduce the student to the MPLAB Integrated Development Environment (IDE) and the MPLAB simulator part of the IDE. A small assembly code program will be entered and assembled. The generated .HEX file will loaded into the simulator and various operational aspects of our code will be observed by using the available tools of the IDE simulator.

Introduction: There is a number of Development platforms supporting the PIC family of microcontrollers. The MPLAB is a free software development tool supplied by the company that makes the PIC micro Microchip. MPLAB is a software program that runs on the PC and provides the tools that allow you to produce the code and program your microcontroller (with the appropriate programming device). It consist of a text editor, a cross assembler supporting a wide range of Microchip micros, a linker, a simulator and supporting a variety of device drivers for a range of device programmers. It also provides support for integrating a wide choice of high level language tools from third parties.

MPLAB IDE provides the necessary tools to do the following:

- Create the source code using it's build-in editor
- Assemble, compile and link source code using a variety of language tools such as assembly, C (from a variety of suppliers), PICBasic and PICBasic Pro.
- Debug your code using the build-in simulator which provides a variety of debugging options or by using the in-circuit debugger.
- Interface to ICD (In Circuit Development) hardware tools that allow you to debug and program your target device.

The tutorial that follows demonstrates some of the functions of the MPLAB development tools. A step by step instruction will help the user in getting to know the procedures involved in the development of PIC based embedded coding.
The MPLAB software used for this tutorial is free to download from the following URL:

http://datainfo.coventry.ac.uk/Panos/Info/Microcontrollers/PIC/MPLAB/

The following steps are required in the software development using the MPLAB IDE:

- Project creation
- Enter the source files (assembler or high level)
- Add source files to the project
- Assemble or compile your source files(s)

- Debug your project using the simulator or ICD

In the documentation that follows we will discuss these steps and simulate/debug our project file.

<u>Procedure: Project creation.</u>

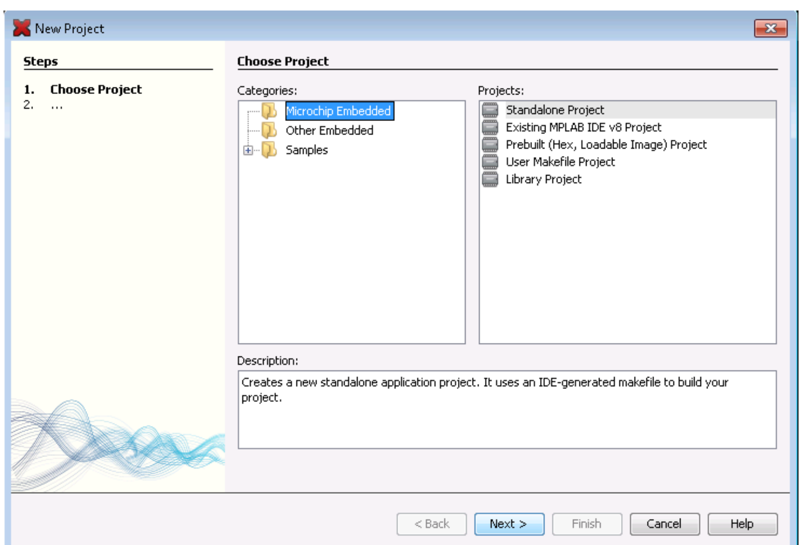
Use Apps Anywhere, or from the desk top to launch MPLABX

MPLAB will be launched showing the welcome screen. Your screen may differ from the one shown depending on the previous use of the software environment**. Click on Create New**

1.Close any open projects in MPLAB® X by right clicking on the Project name and selecting "Close"**.**

2.Click the "New Project" icon to start the project creation process 

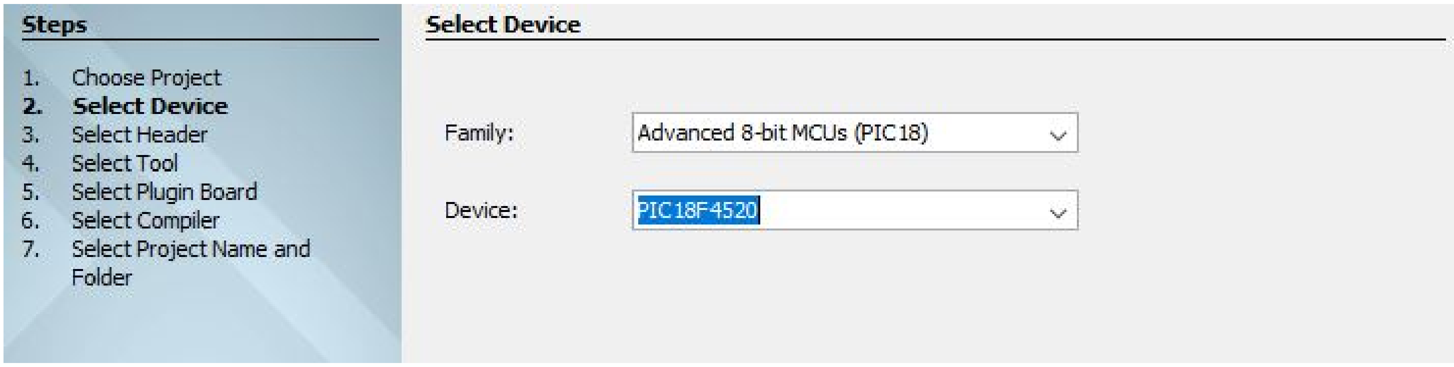
3.Select "Microchip Embedded" then "Standalone project**"**.



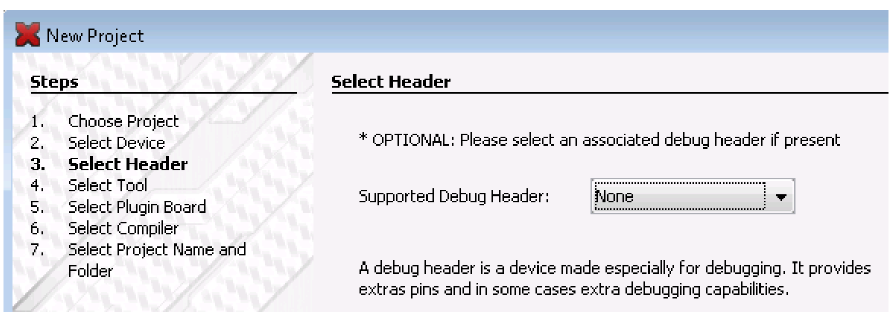
Click 

Select the Processor

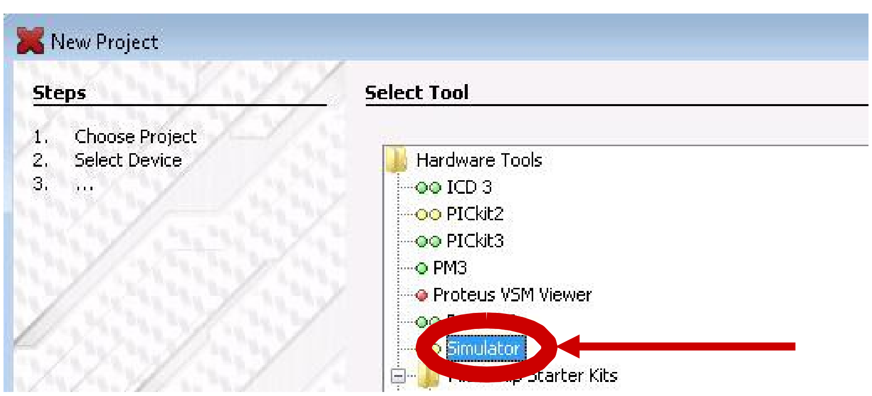
- Select “8-bit MCUs (PIC18)” from the ‘Family’ pull down menu, then select “PIC18F4520” from the ‘Device’ menu.



If the Header option comes up needed: select “None” click 



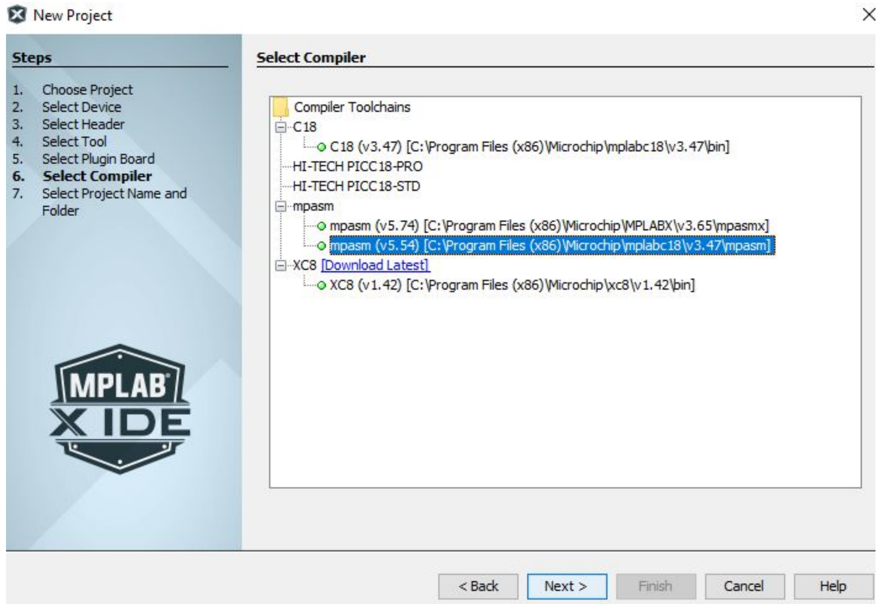
Select “Simulator” under Hardware tools when asked to select a tool.
(You will use other debugging hardware options in future labs)



Click 

Make sure that you have :
mpasm selected.

Click Next >



We will need to supply a name and a directory for our project.

Select Project Name and Folder

Click on the Browse button and navigate through the directory structure and select your H network drive. On the line marked “Project Name” H:\PIC Projects\Tutorial 1.

Notice MPLAB® X filling in the Project Folder line with H:\PIC Projects\Tutorial 1\Tutorial 1.X.X

Alternatively you can enter the path/filename as shown:

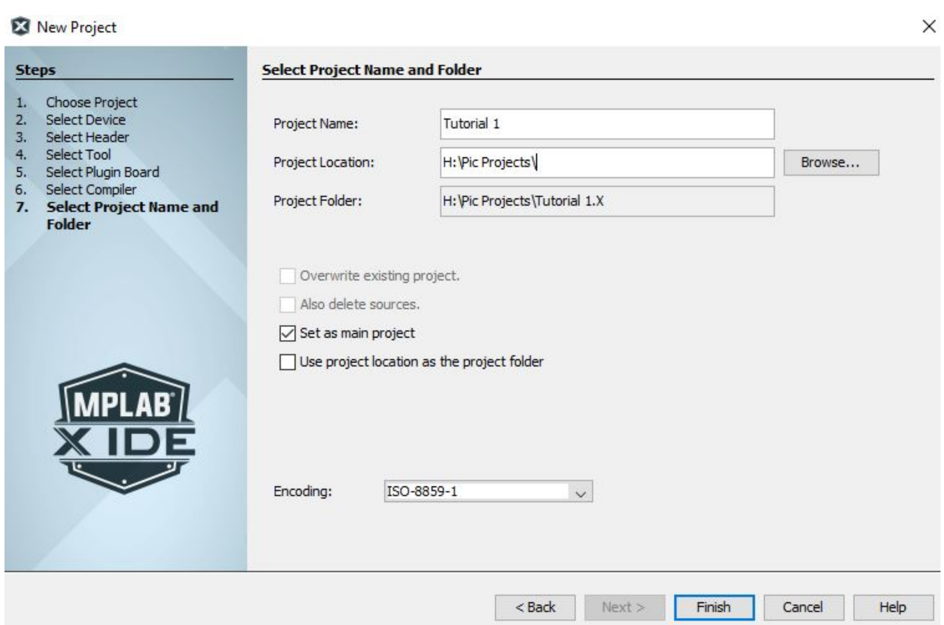
H:\PIC Projects\Tutorial 1

(Or you can use your own storage media)

This will set the project name to: Tutorial 1 and your project directory to **H: PIC Projects.** The Lab PCs have restrictions in using drive C for creating student folders.

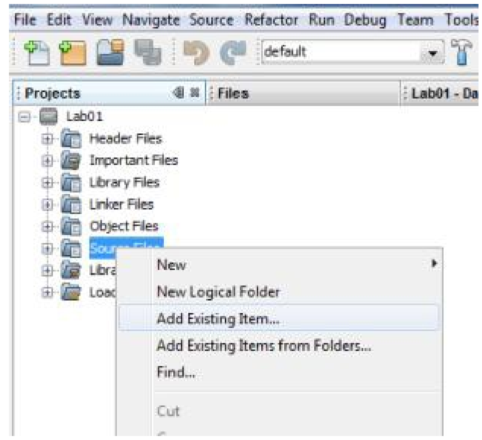
You might find that the PIC Projects folder already exist on your system. If not create a new folder by clicking on the Create New Folder icon as shown

Enter a file name for your project. We will call this Tutorial 1



Click Finish

Step four asks us to add files to the project. We will create a new Source file by using an Assembler MPLAB templates.
Templates are simple files that can be used to start a project. They have all the necessary sections for any source file and provide for good programming practices.



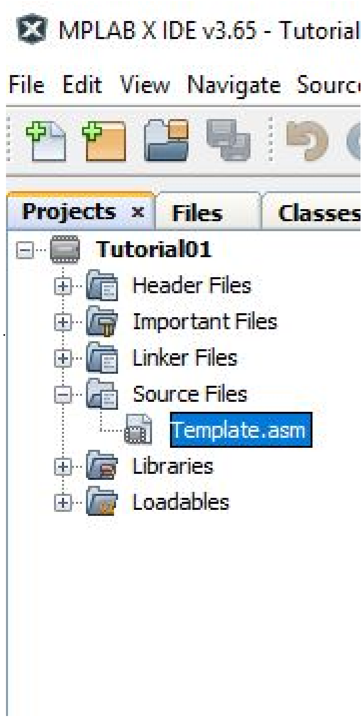
These templates are located at:
http://datainfo.coventry.ac.uk/Panos/Info/Experiments/Year%201/102SE/Examples%20Assembler/Assembler%20Template.asm

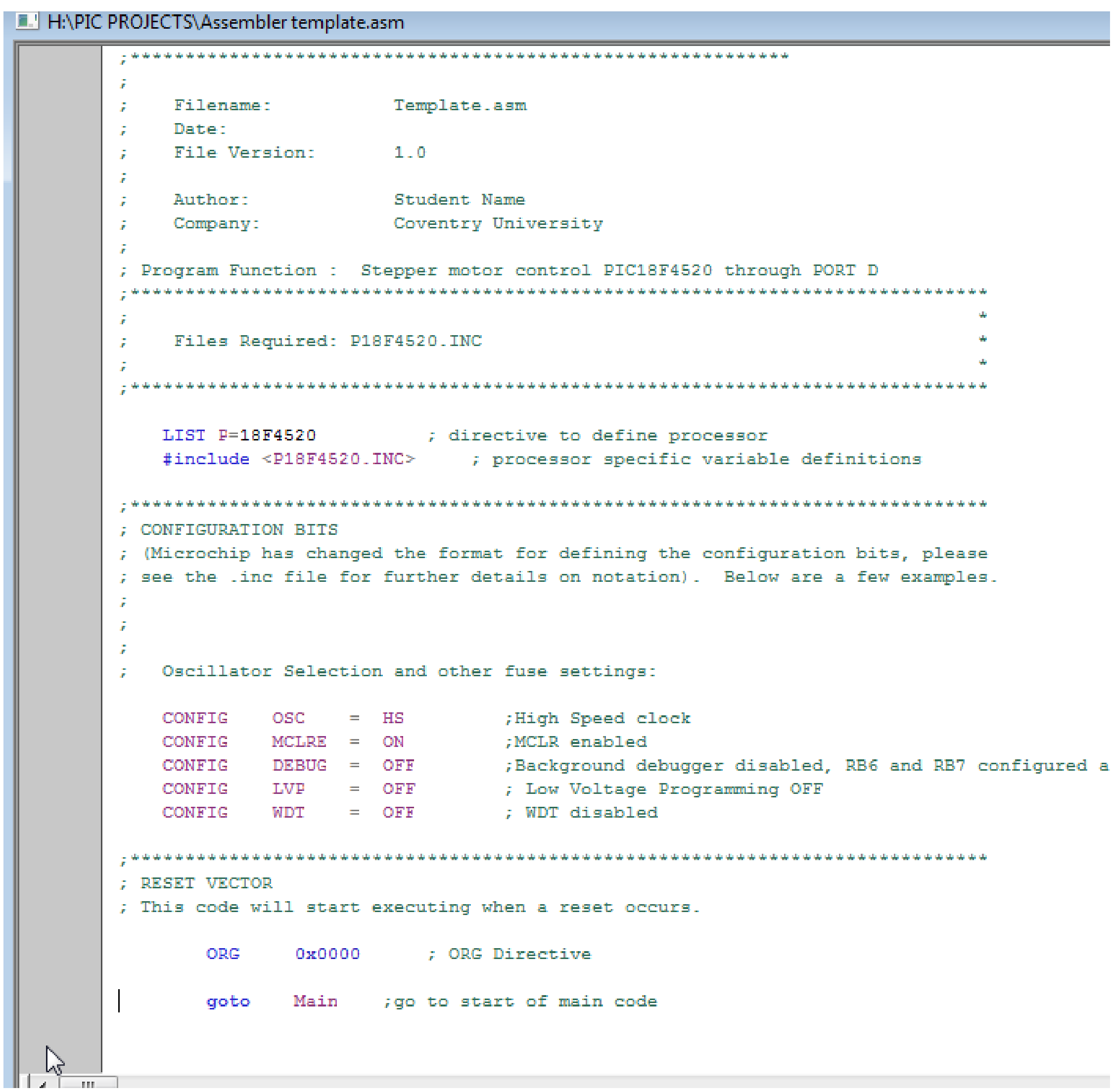
Save the template file in your H network drive PIC Projects directory. (Open Notepad Highlight text and copy into Notepad and save it as Template.asm).

Right click on the "**Source Files** "folder in the project window Select "Add Existing Item" Highlight the 'Template.asm' file in the folder

H:\PIC Projects\Tutorial 1

Ensure the radio button labeled "**Relative**" in the lower right border of the dialog box is checked Click "**Select**". The template file (that you saved) it should now appear on your right side window. Double click the " **Template.asm**" in the project window and the contents for you to edit will appear in the right.




H:\PIC PROJECTS\Assembler template.asm

```
;************************************************************
;
;    Filename:           Template.asm
;    Date:
;    File Version:       1.0
;
;    Author:             Student Name
;    Company:            Coventry University
;
; Program Function :  Stepper motor control PIC18F4520 through PORT D
;******************************************************************************
;                                                                             *
;    Files Required: P18F4520.INC                                             *
;                                                                             *
;******************************************************************************

    LIST P=18F4520          ; directive to define processor
    #include <P18F4520.INC>     ; processor specific variable definitions

;******************************************************************************
; CONFIGURATION BITS
; (Microchip has changed the format for defining the configuration bits, please
; see the .inc file for further details on notation).  Below are a few examples.
;
;
;
;   Oscillator Selection and other fuse settings:

    CONFIG    OSC    =  HS          ;High Speed clock
    CONFIG    MCLRE  =  ON          ;MCLR enabled
    CONFIG    DEBUG  =  OFF         ;Background debugger disabled, RB6 and RB7 configured a
    CONFIG    LVP    =  OFF         ; Low Voltage Programming OFF
    CONFIG    WDT    =  OFF         ; WDT disabled

;******************************************************************************
; RESET VECTOR
; This code will start executing when a reset occurs.

        ORG     0x0000      ; ORG Directive

        goto    Main    ;go to start of main code
```

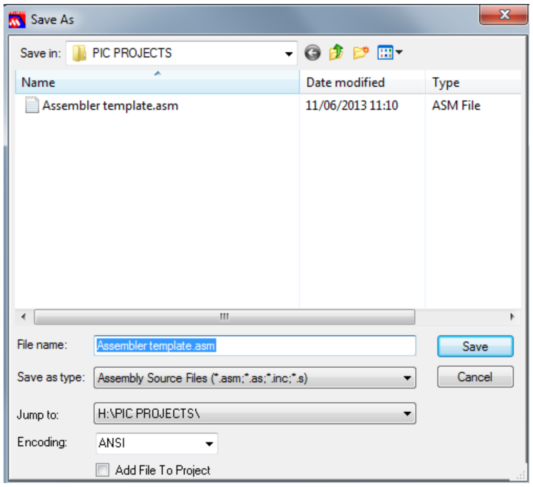
We will now make some changes to our source template file by adding our own program and therefore tailoring it to our own application.

Please enter the additional code to your template. The full listing of your source file should be as shown:

```
;******************************************************************************
;
;    Filename:           Assembler Tutorial 1.asm
;    Date:               20/10/10
;    File Version:       1.0
;
;    Author:            Your Name Goes Here
;    Company:           Coventry University
;
; Program Function:  Simple Port D output counting hex values, no delay routi
;******************************************************************************
;
;    Files Required: P18F4520.INC
; *****************************************************************************

    LIST      P=18F4520          ; directive to define processor
    #include <P18F4520.INC>      ; processor specific variable definitions

;******************************************************************************
; CONFIGURATION BITS
;   Oscillator Selection and other fuse settings:

    CONFIG    OSC     =  HS         ;High Speed clock
    CONFIG    MCLRE   =  ON         ;MCLR enabled
    CONFIG    DEBUG   =  OFF        ;Background debugger disabled, RB6 and RB7
    CONFIG    LVP     =  OFF        ; Low Voltage Programming OFF
    CONFIG    WDT     =  OFF        ; WDT disabled

;******************************************************************************
; RESET VECTOR
; This code will start executing when a reset occurs.

        ORG 0x0000      ; ORG Directive

        goto    Main    ;go to start of main code

Main:
        MOVLW   0x00    ; move literal 00 into W register
        MOVWF   TRISD   ; copy content of W into register TRISD
                        ; above instruction sets  PORTD as Output
Loop
        MOVLW    0x55   ; Move Literal hex 55 into W
        MOVWF   PORTD   ; Move contents of W into PortD
        NOP             ; No Operation
        MOVLW   0xAA    ; Move Literal hex AA into W
        MOVWF   PORTD   ; Move contents of W into PortD
        NOP             ; No Operation
        GOTO    Loop                ; Go to location Loop and repeat process

;******************************************************************************
; End of program

        END             ; End Directive
```

Having made the changes to our program we will need to **save** the modified file. This **file will be saved in our project folder under a new file** name **leaving the original template file intact.**

From the file menu of MPLAB select **File** followed by **Save As**



In response to the **Save As** display, type in the new file name for your project as **Tutorial 1.asm** (.asm is the assembler extension). The destination folder should be **PIC Projects**.

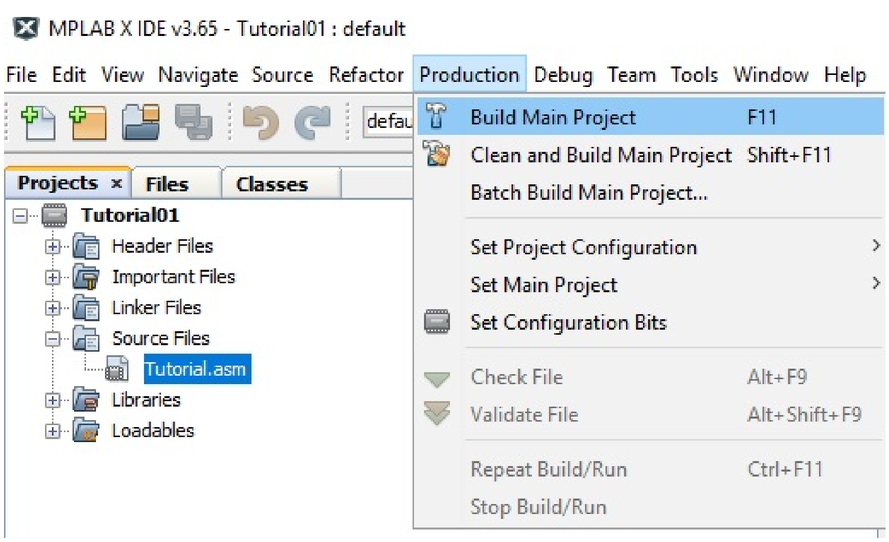
Save your source file by clicking on Save the button.

A copy of the source code can be copied from

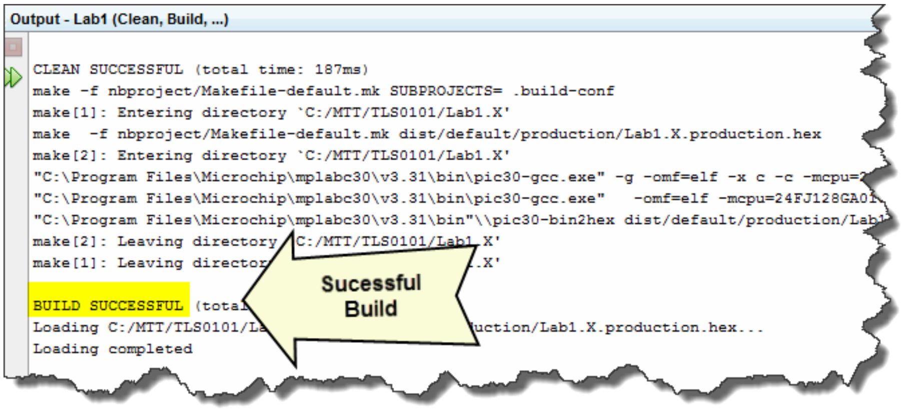
http://datainfo.coventry.ac.uk/Panos/Info/Experiments/Year%201/102SE/Examples%20Assembler/Assembler%20Example%201.asm

Building the Project

The next step to "translate" our source file into the appropriate files that we can use for programming our target Microcontroller. This is done by using the Build command of MPLAB which in effect is the Assembly process.. The following procedure shows the steps required for this:



“**Clean and Build Project**” icon  to build the project



A successful build process creates four additional files. These retain the source file name but they have the following extensions **.cof**, **.o** **.map**, and .hex. These files are used for debugging and programming our PIC device.
If any errors appear on your output window these will be flagged**. Double- clicking on the error line** will take you the corresponding offending line of your editor window. All we need to do in this case is to correct the errors and repeat the Build process.

To verify the code we will use the MPLAB® X simulator.

Objective

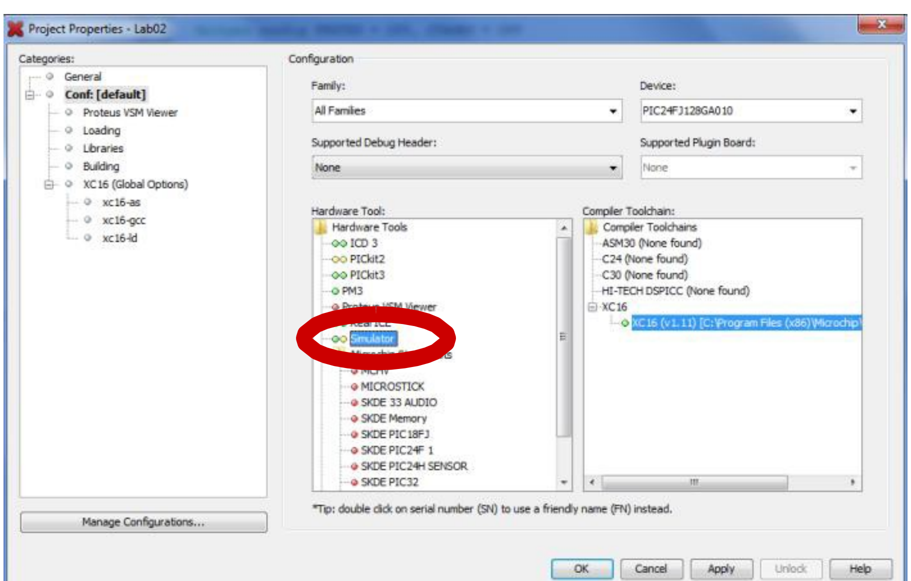
This lab reviews how to build a debug version of a project and send it to the MPLAB® X simulator. Once the simulation session has started, this lab reviews how to control the program execution. This lab provides the basics of observing the program to ensure the correct lines of code are executed. This lab also reviews how to monitor and control program variables and the PIC Special Function Registers (SFRs)

Select the Simulator as the “Hardware tool”

- Select Tutorial 1 from the project list
- Right click
- Select “Properties”
- Click Conf:[default]
- Under “Hardware Tools”, verify that Simulator is selected

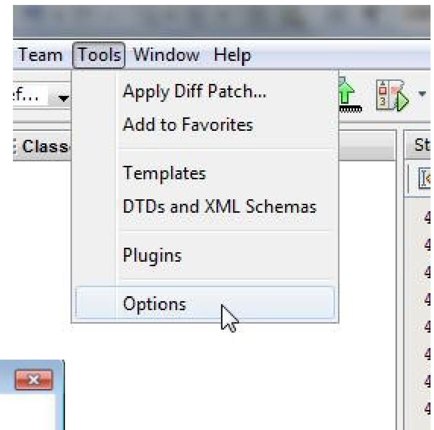
Information

The MPLAB® X simulator is a soft- ware simulator even though it is listed under the “Hardware Tools”. As you will learn later in this lab the simulator has most of the functionality of a hardware de- bugger

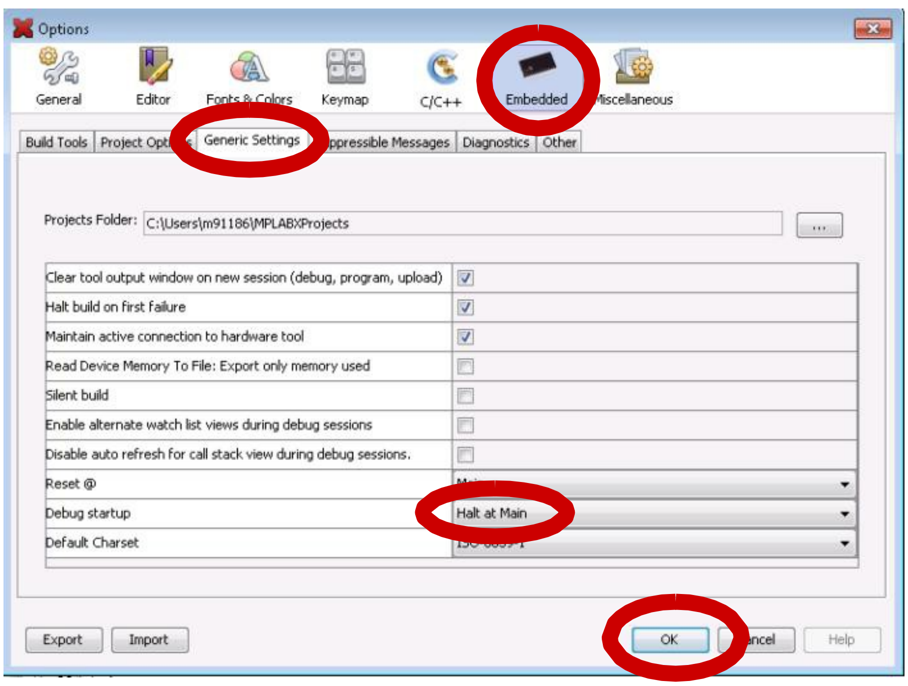


Selecting the Simulator as the Hard- ware tool for a project
Click “OK”

After building the debug version of a project, MPLAB® X will launch a debug session and begin a simulation. This auto-start feature can save time, however the user may not wish for the debug session to proceed into the main program until directed to do so. To prevent the debug session from automatically run-ning after we build we will set an option in MPLAB® X requesting the debug session halt at the beginning main();



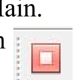
- From the “Tools” pull down menu select “Options”
- Select the “Embedded icon”
- Select the “Generic Settings” tab
- Ensure the “Debug startup” is set to ‘Halt at Main’

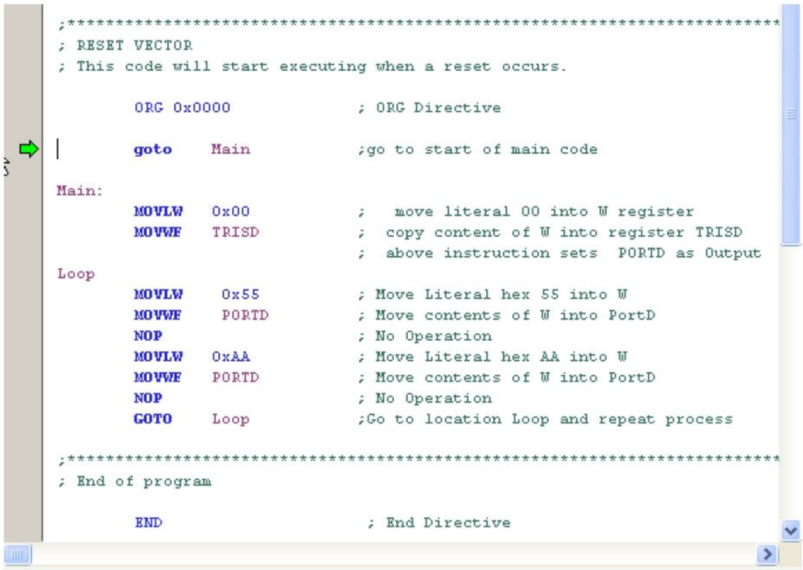


Starting the Debug Session

Build a debug version of the open project by clicking on the “Debug Project” icon.

You can run the program by clicking on the contiue icon green arrow .

On selecting Reset  a listing of our assembly code appears with a green arrow that shows where the program execution will begin. This first instruction goto Main instructs the processor to jump to the label called Main.
You can pause the program using pause icon  or stop the program using the stop icon 



You will notice that clicking the "Debug Project" icon will build the project, download it to the simulator, start the simulator session, and run until the program enters the main() function.

Debug Toolbar

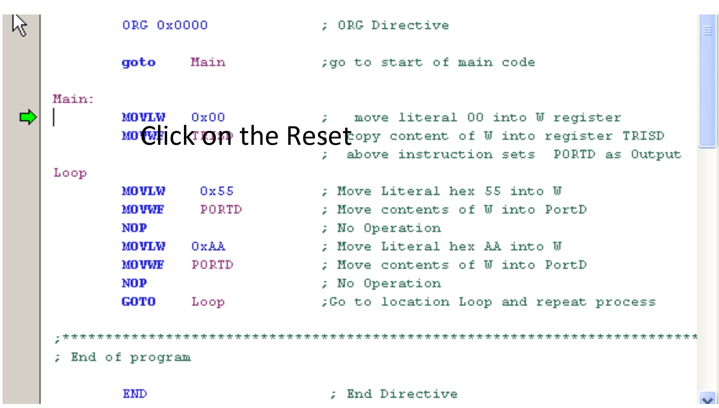
Notice the addition of the Debug Toolbar

On selecting Reset a listing of our assembly code appears with a green arrow that shows where the program execution will begin. This first instruction goto Main instructs the processor to jump to the label called Main.

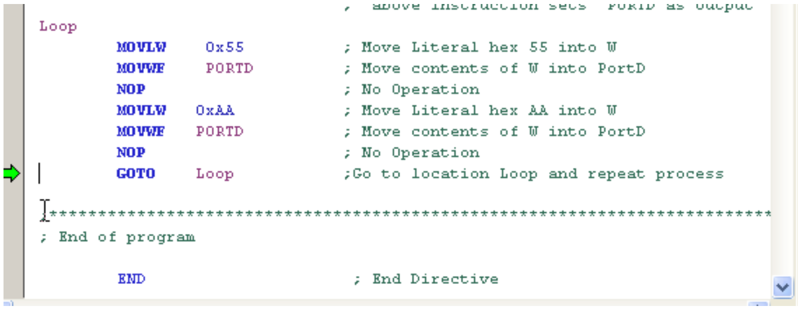
Stepping through the Code

From the debugger Toolbar select **the Step Into** icon to execute one line of source code.

The green arrow now points to **MOVLW 0x00** instruction. This will point to the instruction to be executed next. As we click on the **Step Into** icon the program will continue to step to next instruction of our code. In other words we are simply executing one instruction at a time through our code.



Continue stepping through your code till you reach the GOTO Loop instruction.



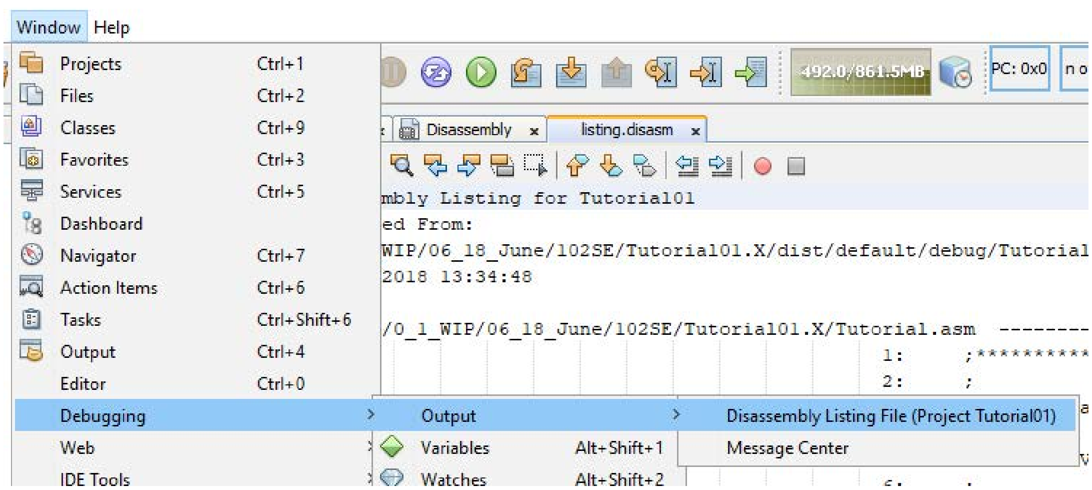
What happens if we continue stepping from this location? Explain

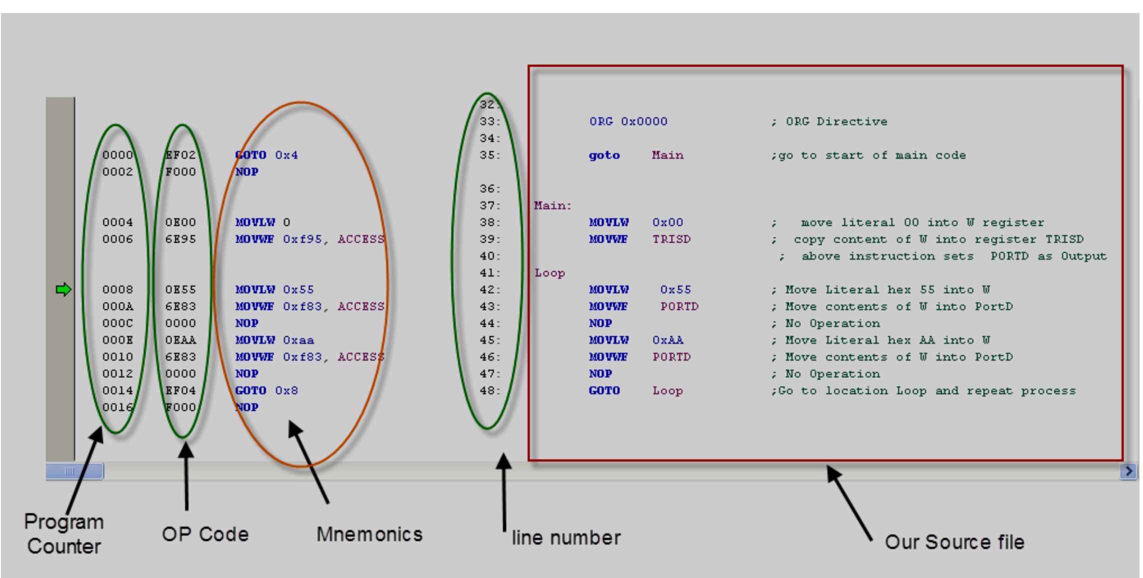
Disassembly Listing

The Disassembly listing file gives a detailed view of our code. It includes our original source file but also adds a useful selection of other options such as line numbers, Program counter location of each of the instructions as well as the Op Code for the Mnemonics used. It could be very useful when we need to debug our code.

A disassembly list of our assembly code can be obtained by using the following procedure:

Select **Window** from the Toolbar followed by **Debugging Output Disassembly Listing**.

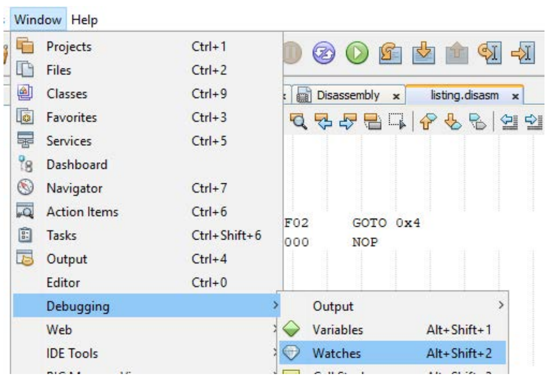


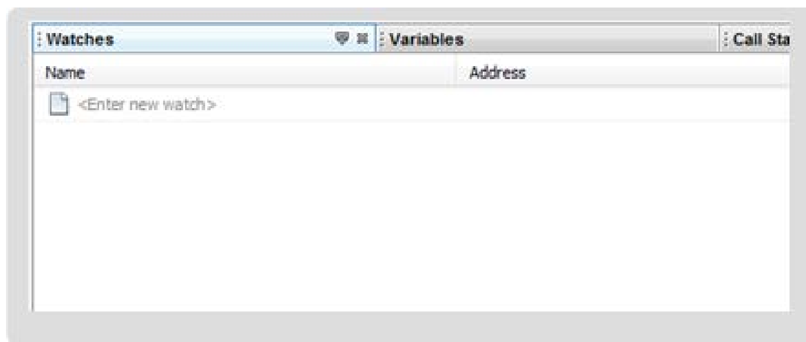


Watch Window.

In order to see if our code executes properly we need to see what's happening at various strategic parts of our code. In our example we are using one of the PIC18 ports namely Port D as output. The program initially configures the function of the port and then writes alternating values of ones and zeros into it. These values are hex 55, (binary 01010101) followed by the compliment of this i.e. hex AA , (binary 10101010). One of the most important aspects of our code is to observe the writing of this data to the Port.
To be able to achieve this we will need to set a Watch Window on Port D.
The following procedure applies:

- From the **Window** menu select **Debugging** ,then **Watches**





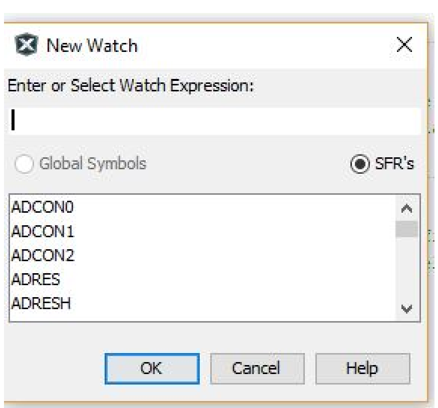
A Watch Window with no variables or SFRs being monitored

> Are there any variables in the Watch window when you opened it up?
> MPLAB® X keeps a history of the last Watches used. Keeping this persistent data can speed up debug session by not requiring the programmer to reconfigure watches.
> Today we are going to *delete* everything in the Watch window before we start.
> To erase the Watch Window "Right Click the mouse in the Watch Window and select "Delete All"

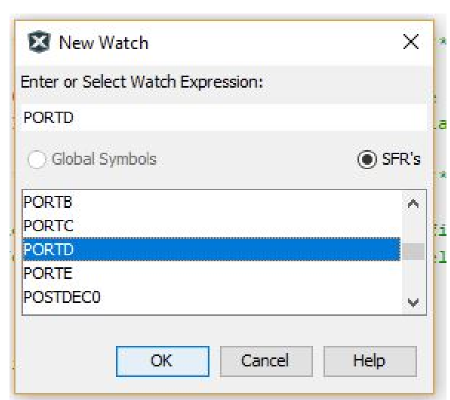
Adding an item to the Watch Window

- Right click in the Watches Window and select "**New Watch…**"
- Highlight the radio button "**SFR**'s"
- Select PORTD from the scrolling menu of Special Function Registers (SFR)

- Click "OK" 



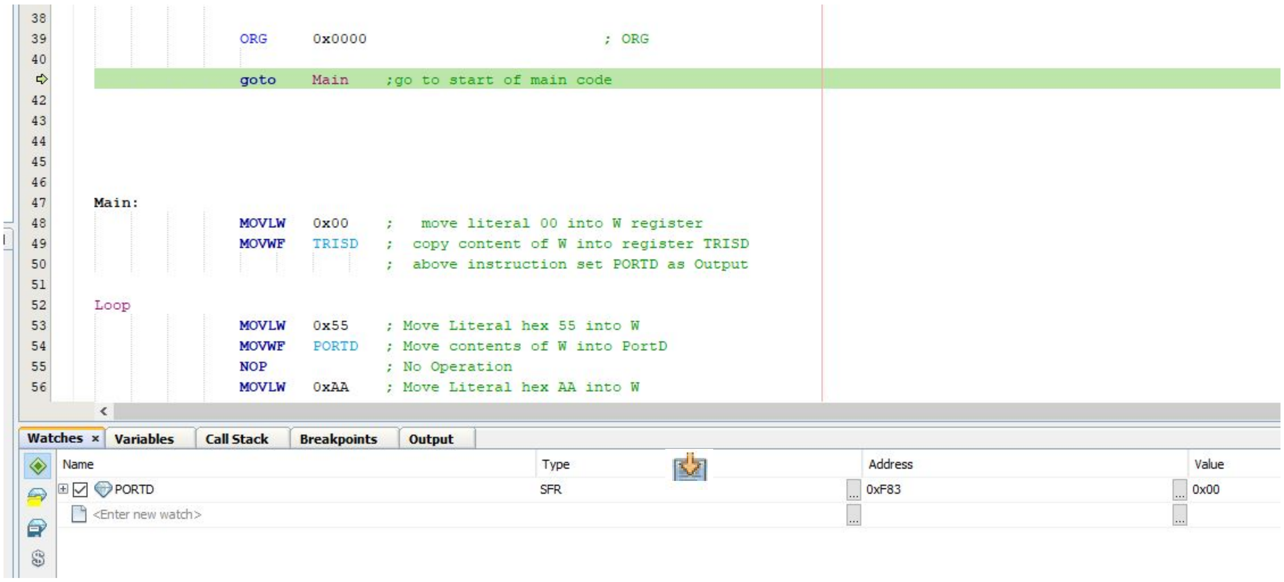
New window appears



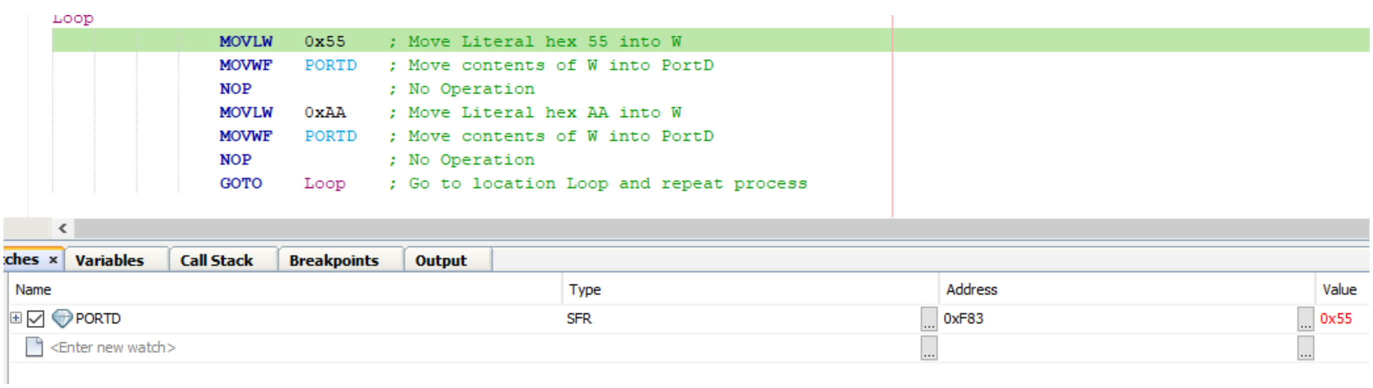
Let us now go back and repeat the stepping through the code excercise but this time we will be able to observe the changes on Port D through the watch window on Port D.

Click on the Reset  icon on the Debugger toolbar.

The Reset action will set the program counter to the Reset vector location 0x00 which is the start of of code. The green arrow on our list points to this location as shown in the screenshot that follows.
At this point the content of PORTD is 00 which is the default reset value for the ports.

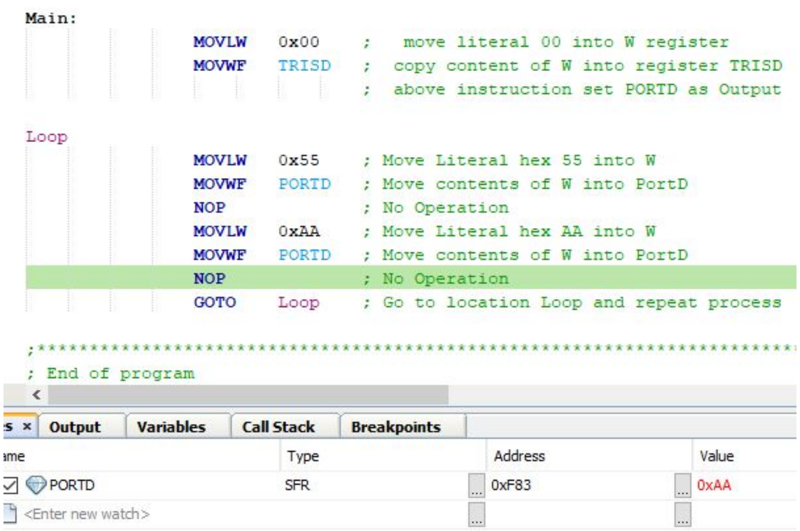


Continue stepping () through the code and observe the changes of Port D in the watch Port D window.



At the program counter location indicated by the green arrow the contents of Port D is hex 55 i.e. instruction MOVWF has transferred the contents of W register (0x55) into Port D

Continue stepping through the program till you reach the second NOP instruction.



At the program counter location indicated by the green arrow the contents of Port D has changed to Hex AA i.e. instruction MOVWF has transferred the contents of W register (0xAA) into Port D

Continuing looping through the program we will see that Port D alternates between the two hex values of 0x55 and 0xAA.

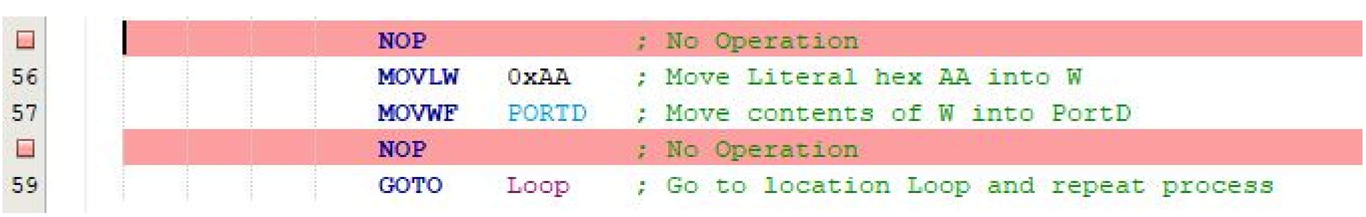
Now select the **Continue** icon file  from the debugger toolbar and observe the results on the source display and the watch window.

Breakpoints

Stepping through a large program could be quite tedious and time consuming. A tool that is very useful in such a situation is the use of Breakpoints. To set breakpoint you simply click on the instruction at which you need to set a breakpoint. Breakpoints are simply halt commands that stop the program at the selected strategic points of your code. When a breakpoint is encountered the program halts. From this point we can either single step or continue execution till we reach another breakpoint.

On your program code double click on the two **NOP** instructions. By doing so a red  appears on the selected code location indicating that the breakpoints have been set. To clear a breakpoint you simply double click on the corresponding breakpoint location.

Having selected our two breakpoint locations our code displayed should be as shown below:



Now let us run the code at full simulator speed by using the Run icon  of the debugger toolbar. We should now be able to see the skipping of the in-between code. The program halts at the two breakpoint locations.

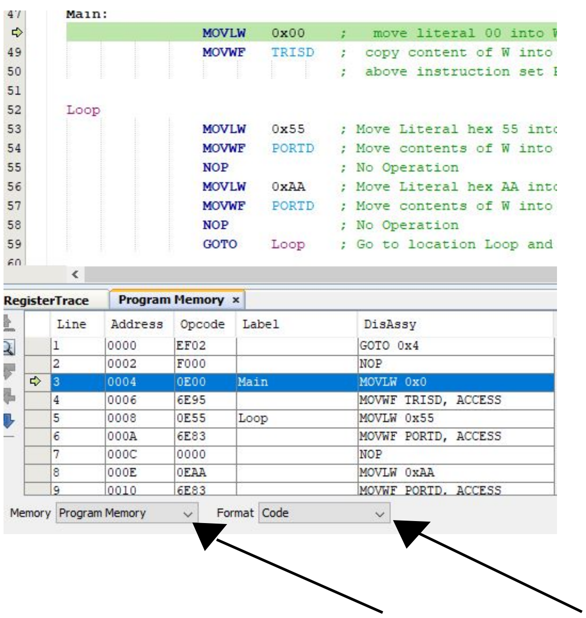
<u>Other useful tools</u>

Program Memory: provides a split information panel of your program together with an informative display of other useful aspects the that user needs to know for debugging purposes. These include pre / current program counter locations, addresses of opcode, opcodes, Mnemonic instructions, source and destination of data and number of machine cycles taken for the instructions.

To launch the **Program Memory** simulator select **Window** from you View tab, and select **PIC Memory Views,** Then select **Program Memeory** from the drop-down menu as shown in the following screen shot.

By launching the **Program Memory** the following screen is displayed. Reset the processor by clicking on the **Reset** icon 

Start the program and continue execution by clicking on to the **Step Into** 

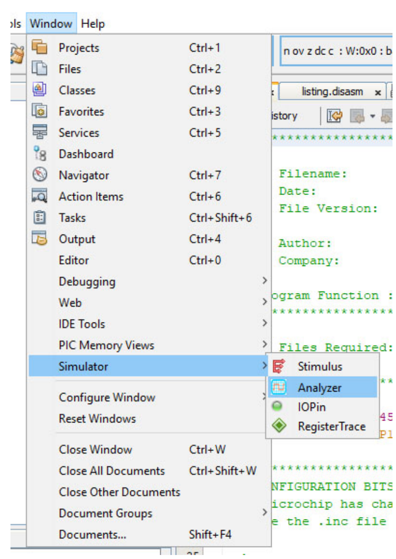


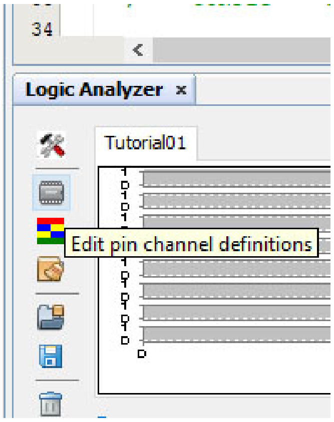
Screenshot providing useful debug information. This includes line numbers, Program locations, OP-Code, Labels, Mnemonics, etc.

Click on this down arrow to view more program memory information

<u>The Simulator Logic Analyser</u>: provides a logic analyser timing display of various pins and bus signals of the micro as we will see.

To launch the Logic analysis tool select **Window ->Simulator-> Analyzer** from the tool bar followed by the Simulator Logic Analyser tab from the drop-down menu as shown below.



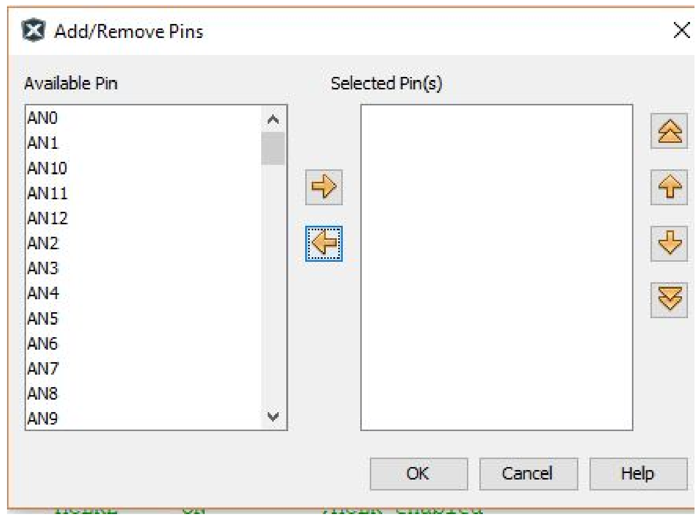


Logic Analyser:

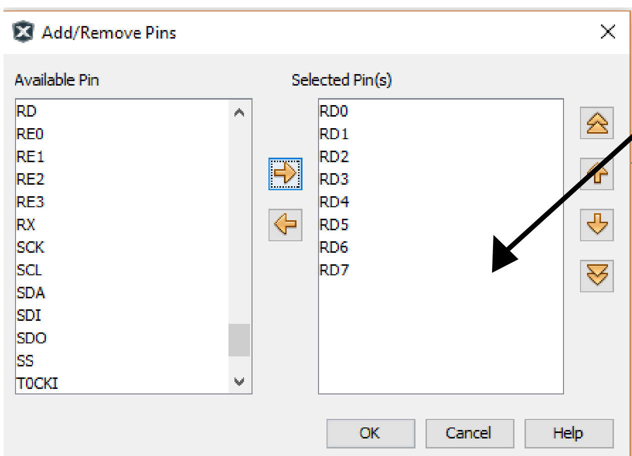
The logic analyser display window will open.

You can select diferent pins and bus signals of the micro as we will see.

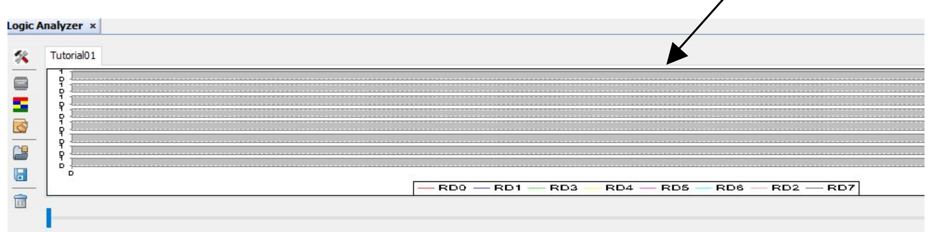
To launch the Logic analysis tool Click on the **Edit Pin**  Icon



Scroll down and select all of PORTD pins, RD0 to RD7



Initial non configured Logic Analyser display. This may vary depending on the previous user settings



To observe the timing activity of the selected Port D signals RD0 – RD7 all we need to do is to reset  the debugger and single step through our code using the Step Into  icon as before.

1
0
1
0
1
0
1
0
1

Simulator Logic Analyser display showing the timing diagram for PortD bits 0-7. These values toggle between 0x55 and 0xAA

X055

We have briefly explained some of the basic function of the MPLABX IDE and its various components **Exercises**

1. Rewrite the example code so that it displays three different hex values on port C. Use the trace and logic analysis tools to display your values.

2. Write an assembly code program similar to the example shown on page 3 of the Introduction to Assembly Language PIC18 tutorial. Build the program and provide simulation results showing important areas of your code.

For the above exercises consult the Instruction set information documentation supplied previously.
This is also available on datainfo.coventry.ac.uk @
http://datainfo.coventry.ac.uk/Panos/Info/Experiments/Year%201/102SE/

Useful links
http://www.microchip.com/

http://datainfo.coventry.ac.uk/Panos/Info/Microcontrollers/PIC/MPLAB/MPLAB%20Full%20Tutorial.pdf

Lab Sheet by PDA Modified and updated by SWY 19/06/18