Practical Activity # 1
Installation and Familiarization of MPLAB IDE
(syllabus reference: Unit 2)

Exercise Objectives:
1. Installing MPLAB IDE
2. Installing MPLAB XC8 C Compiler
3. Creating workspace and projects in MPLAB IDE
4. Compiling projects in MPLAB IDE
5. Simulating embedded systems program using Proteus PIC16F877A VSM

Student Outcomes:
At the end of the exercise, students must be able to:
1. install and configure the software required;
2. create workspaces and projects in MPLAB IDE;
3. compile source codes in MPLAB IDE using MPLAB XC8 compiler and
4. simulate embedded systems program using Proteus.

Tools Required:
The following will be available for the students (click link to download):
1. MPLAB IDE v8.2 (free)
2. MPLAB XC8 v1.33 (free)
3. Proteus v8.11 (online license)

Delivery Process:
The instructor will conduct a one-time step by step demonstration on the installation and setting up of the starter kit as well as using the IDE and the debugger software. All inquiries pertaining to the latter must be referred to this manual.

Part I. Installation of MPLAB IDE and MPLAB XC8

Installing MPLAB IDE

The MPLAB Integrated Development Environment (IDE) is a supplemental software developed by Microchip to aid the development in their range of microcontroller products or MCU. It manages projects and is able to compile C codes. It also has an assembler for development using the Microchip MCU native instruction set. It also supports a wide-variety of MCU models.

> **Note:** You will be installing a full version of MPLAB Integrated Development Environment (IDE).

1. Download the MPLAB IDE v8.92 installer.
2. After downloading, unzip the archive file then double click on "setup.exe" to start the installation process.
3. Follow the installation instructions. Just click “Next” to proceed. You have to agree to the License Agreement by clicking “yes”, otherwise the installation would be terminated. Use the default installation path (C:\Program Files…).
4. Validate the installation by running MPLAB. Once it successfully opens, close the application and proceed.

> **Warning:** Make sure that MPLAB is closed before proceeding.

Installing MPLAB XC8

Since MPLAB does not have a pre-packaged compiler for C programs, a separate compiler will be installed. The MPLAB XC8 C compiler is a proprietary software by Microchip and supports only 8-bit MCUs such as the PIC16F877A.

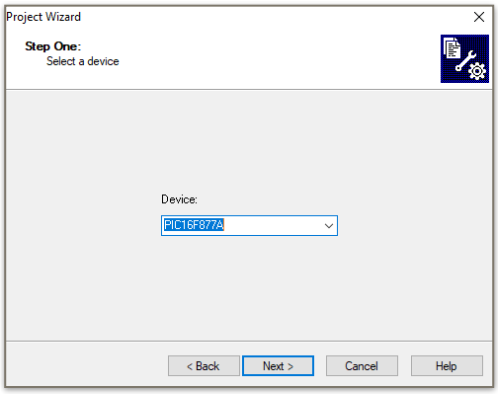
1. Download the MPLAB XC8 v1.33 installer. Run the installer.
2. After the license agreement, make sure to tick "Install Compiler" then click Next. Then click Next in the "Installation Type".
3. Use the default installation path then click Next. Then click Next in the "Compiler Settings".
4. After the installation process is finished, in the "Licensing Information" just click Next then Finish.

> **Note:** The version of XC8 you have installed is not the PRO edition. There is no optimization during compilation.

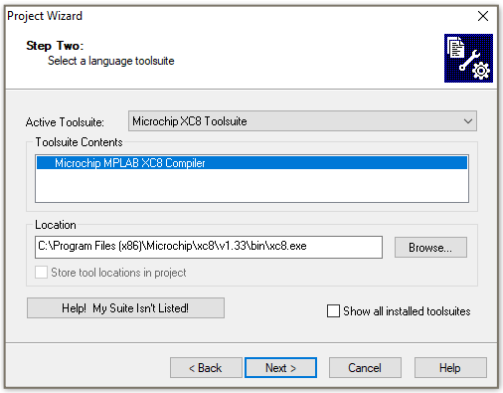
Part II. Creating Projects in MPLAB IDE

Before proceeding, make sure both MPLAB IDE and MPLAB XC8 are installed.

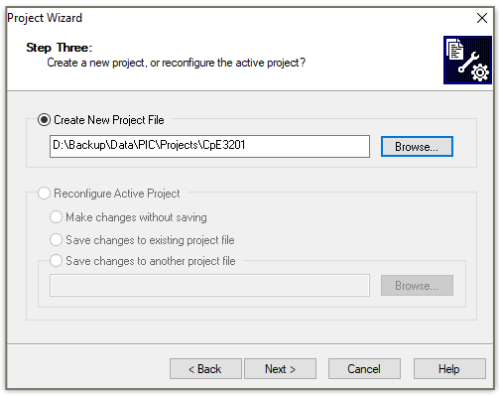
1. Open MPLAB IDE. Then click on "Project" on the main menu then select "Project Wizard". When the project wizard window opens, click Next.



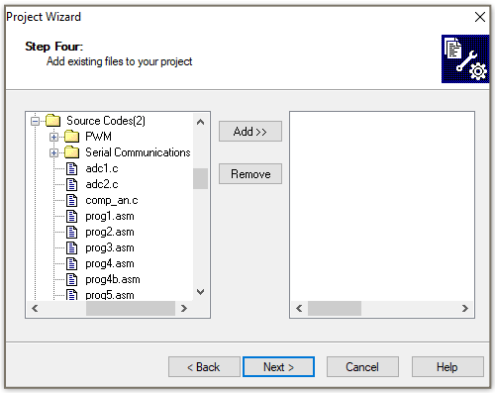
2. Select PIC16F877A from the device drop down list then click Next.



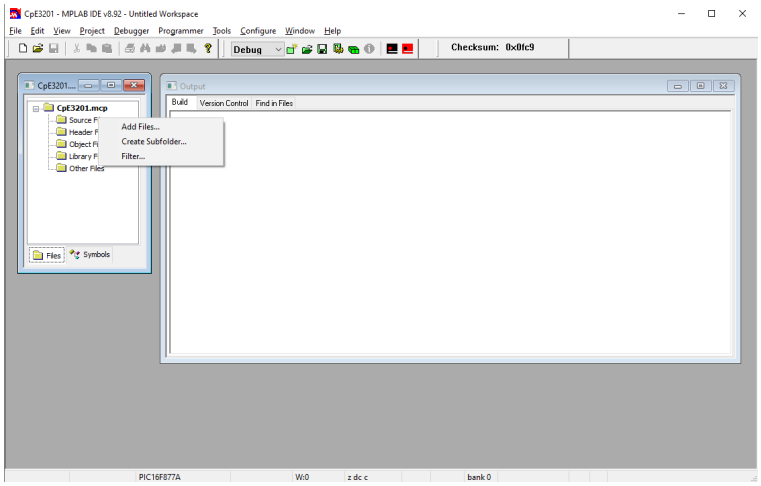
3. On the "Active Toolsuite", select "Microchip XC8 Toolsuite". You can change the location of the application otherwise use the default installation path then click Next.



4. Then click on browse then locate the folder where you want to save your MPLAB projects then create a name of the project. You can name the project "CpE3201". Click Next to proceed.
5. If you have available source code, you can add it in your project by selecting the files from the left. The files added to the project will appear on the right. If you don't have any source code to add yet, just click Next. Click Finish on the summary window to end the Project Wizard.



6. As of this point, you have created a project a project as well as a workspace. MPLAB project files have an extension of ".mcp" and workspace files have an extension of ".mcw". Workspace save whatever the state of the MPLAB IDE. If you will open a workspace, it will open the project associated with the workspace and restore any open windows like source code and build status.



7. To add a source file in your project, right click on the "Source Files" folder in the workspace window then click "Add Files". Locate the file(s) you want to add. You can add multiple source files if you made your program as such. Only files with ".c" extension can be added.

8. The added files will appear below "Source Files" folder. You can remove the files by right-clicking the file then click "Remove".
9. To create a new source file, go to "File" the click "New" or simply ctrl+N. A source file window will open. Write something say "#include<xc.h>" then save it as a C source file. Saving the file as a C source file will turn on the code editing features.
10. Add the created source file to the project as in procedure #7.

Part III. Programming the PIC16F877A MCU

It is important to configure the PIC16F877A to enable or disable some features such as code protection, power-up timer, brown-out reset, oscillator options etc. Details of the 13-bit configuration word can be found on page120 of the PIC16F87X data sheet. The following are procedures to set the configuration word via the MPLAB IDE and through the source code.

> **Warning:** Unchecking "Configuration bits set in code" will override any configuration set the in the source code. Do this only when no configuration are on the source code.

1. To set the configuration word via MPLAB IDE, click on "Configure" on the menu bar then click "Configuration Bits". Once the configuration windows opens, uncheck the "Configuration bits set in code".
2. Set the following bit(s) by clicking the items in the "Setting" column. A drop down menu will appear and choose a configuration. For this exercise, follow the MCU configuration below:
   - FOSC: *XT oscillator* (crystal oscillator)
   - WDTE: *WDT disable*d
   - PWRTE: *PWRT enabled*
   - BOREN: *BOR enabled*
   - LVP: *RB3 is digital I/O...*
   - CPD: *Data EEPROM code protection off*
   - WRT: *Write protection*
   - CP: *Code protection off*

   The hexadecimal value of the configuration word should be 3F71H.
3. If you want to put the configuration word on the source code itself, make sure to check the "Configuration bits set in code" in the configuration window. This will allow the compiler to use settings in the source code.
4. In your source code, add the following pre-processor directives just below the include files.

```
#include<xc.h>        // include file for the XC8 compiler

#pragma config FOSC = XT
#pragma config WDTE = OFF
#pragma config PWRTE = ON
#pragma config BOREN = ON
#pragma config LVP = OFF
#pragma config CPD = OFF
#pragma config WRT = OFF
#pragma config CP = OFF

...
```

5. The following is an example program that will blink an LED connected to RB0 (PORTB, bit 0) with a loop delay.

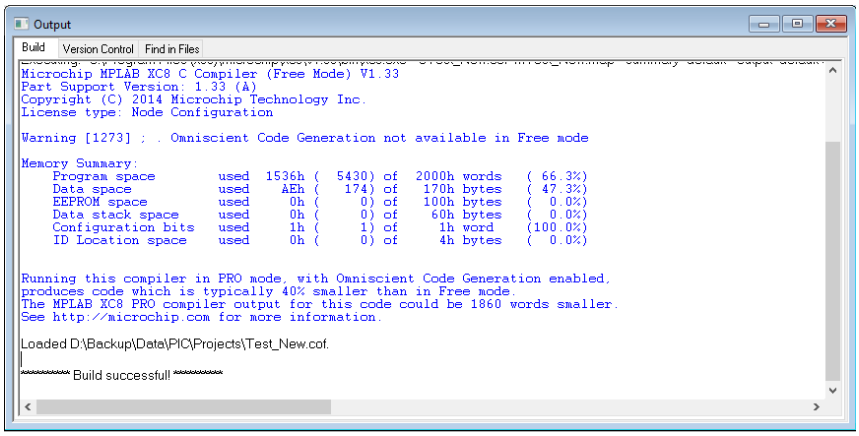
```
void main()
{
        int cnt;

        TRISB = 0x00;  // set all bits (port) in PORTB as output

        for(;;)
        {
                for(cnt=0;cnt<10000;cnt++);    // delay
                PORTB = 0x01;                  // set RB0 to 1 (LED ON)
                for(cnt=0;cnt<10000;cnt++);    // delay
                PORTB = 0x00;                  // set RB0 to 0 (LED OFF)
        }

}
```

6. Save your source code and make sure it is added to the "Source Files" in the project.
7. Build the project by going to "Project" in the main menu and click Build or use the keyboard shortcut F10. Should the build is successful, a “Build successful!” message shall be seen on the build status window (Output). If error(s) are encountered, debug the code and build again.



8. After a successful build process, the compiler will generate a hex file that will be used to flash the program to the MCU. The file is located in the same folder as the project file with the filename "<name of project>.hex". Take note of the latter as it will be used in the next part.

Part IV. Simulating Program using Proteus

After building, the generated hex file, it must be embedded into the PIC16F877A virtual simulation

> **Note:** We will not compile the source code in Proteus.

model in Proteus. Then simulation can be performed.

1. Open Proteus.
2. Create a new Firmware Project with the filename "LE1.pdsprj" and select PIC16 and PIC16F877A as "Family" and "Controller" respectively. Select MPLAB XC8 as the compiler.
3. Construct the circuit by adding power/ground (VDD, VSS) and a crystal oscillator to the MCU at OSC1 and OSC2. Make sure the oscillator frequency is 4 MHz. Connect MCLR to VCC. Connect an LED (properly biased) to RB0.
4. Embed the hex file (program file) created in Part III to the MCU by right-clicking on the device then select "Edit Properties". Upload the program file in the "Program File" by clicking browse and locate the .hex file.
5. Run the simulation and observe the output. If the output is not correct, check your program, edit and then build. You don't have to perform #4. Just stop the simulation, then start again to update the firmware (program file).

Part VI. Hands-on Exercise

1. Modify the Proteus project "LE1.pdsprj" by connecting a normally open, active-high push button switch to RA0 (PORTA, bit 0).
2. Close the CpE3201 project by going to "Project" then click "Close". Create a new project using the Project Wizard with the name "<your last name>_CpE3201".
3. Create a new source file in MPLAB with filename "LE1.c". Add the created source file to the project. Include the configuration word of the MCU in the source code.
4. Write a program that will blink the LED three (3) times when the switch is pressed. After blinking three times, the program will for a switch press again.
5. Build the project and verify if it builds successfully.
6. Embed the firmware in the PIC16F877A VSM in Proteus ("LE1.pdsprj").
7. Run the simulation and observe the output. If the output is incorrect, check your program, edit and then build and repeat the simulation.

Submission Instructions:

Submit the following in Canvas in the correct order:

- LE1.c
- LE1.pdsprj*

** make sure that the firmware has already been embedded.*

Criteria	Outstanding (10 pts)	Competent (8.5 pts)	Marginal (7.5 pts)	Not Acceptable (5 pt)	None (0 pt)
Creating and Configuring a Project	Successfully created and configured a project.	-	-	Project was not successfully created.	No project created.
Adding source files to Project	Successfully added source files to a project.	-	-	Failed to add source files in a project.	No project created.
Building a Project	Project build was successful.	-	-	Project build failed.	No project created.
Simulation using Proteus	Successfully performed the simulation.	-	-	Simulation not successful.	No project created.

Assessment

Copyright Information

Author: Van B. Patiluna (vbpatiluna@usc.edu.ph)
Contributors: none
Date of Release: February 8, 2021
Version: 1.0.2

References

PIC16F87X Data Sheet. Microchip Technology Inc., 2001.

Change log:

Date	Version	Author	Changes
February 8, 2021	1.0	Van B. Patiluna	Initial draft.
February 10, 2021	1.0.1	Van B. Patiluna	Revised one of the rubric criteria.
February 11, 2021	1.0.2	Van B. Patiluna	Corrected factual errors.