COMP551 – Interfacing

Fall 2017

Lab 2

Using MPLAB and C18 to Compile a Simple Program

(Note: "Project Basics and MPLAB IDE Configuration" supplement included with this lab.)

Students:		
Student ID's:		
Section:	**01** **02** **03** **04**	

NOTE: Labs are due at the start of the next lab period. Only submit one lab per group of two students.

Lab 2 – Using MPLAB and C18 to Compile a Simple Program

2.1 PROGRAM: "HELLO, WORLD!"

2.1.1 Write the Source Code

The typical "Hello, world!" function contains this C statement to print out a message:

```
printf ("Hello, world!\n");
```

The function main() is written like Example 2-1:

EXAMPLE 2-1: HELLO, WORLD! main() CODE

```
void main (void)
{
        printf ("Hello, world!\n");
        while (1)
          ;
}
```

Note: Often, the final "while (1)" statement is not used for the "Hello, world!" program because the example compiles on a PC, executes then returns back to the operating system and to other tasks. In the case of an embedded controller, however, the target microcontroller continues running and must do something, so in this example, an infinite loop keeps the microcontroller busy after doing its single task of printing out "Hello, world!".

For this to compile using MPLAB C18, the code is shown in Example 2-2.

EXAMPLE 2-2: HELLO WORLD CODE

```
#include <stdio.h>

#pragma config WDT = OFF

void main (void)

{

        printf ("Hello, world!\n");

        while (1);

}
```

The first line includes the header file, stdio.h, which has prototypes for the printf() function. The #pragma statement is unique to MPLAB C18. The #pragma statement controls the Watchdog Timer of the target microcontroller, disabling it so it won't interfere with these programs.

Note: The Watchdog Timer is a peripheral on the PIC18 MCUs that is enabled by default. When it's enabled, the program will eventually time-out and reset. In a finished application, the Watchdog Timer can be enabled and used as a check to ensure that the firmware is running correctly.

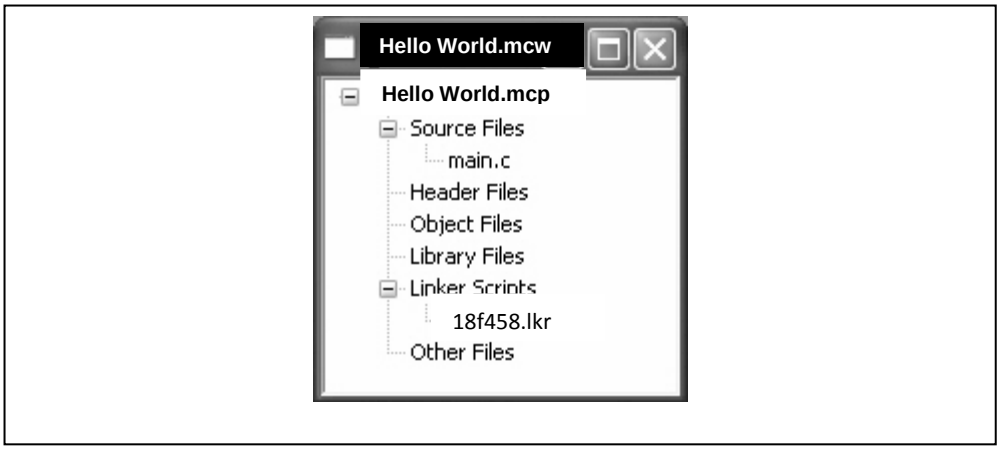
2.1.2 Make Program

Create a new project (refer to the MPLAB handout, Project Basics) called **Hello World** in a new folder named **Lab 2**. Create a new file, type the code in Example 2-2 into it and save it as a file named main.c. Then, add the file main.c as the source file in this folder and add the 18f458.lkr linker script.

NOTE: The 18f458.lkr file is located in the C:\mcc18\bin\lkr directory.

The final project should look like Figure 2-1:

FIGURE 2-1: FINAL PROJECT WINDOW



Note: For this example, the PIC device you select is not important. Select the PIC18f458 as the current device with ***Configure->Select Device***.

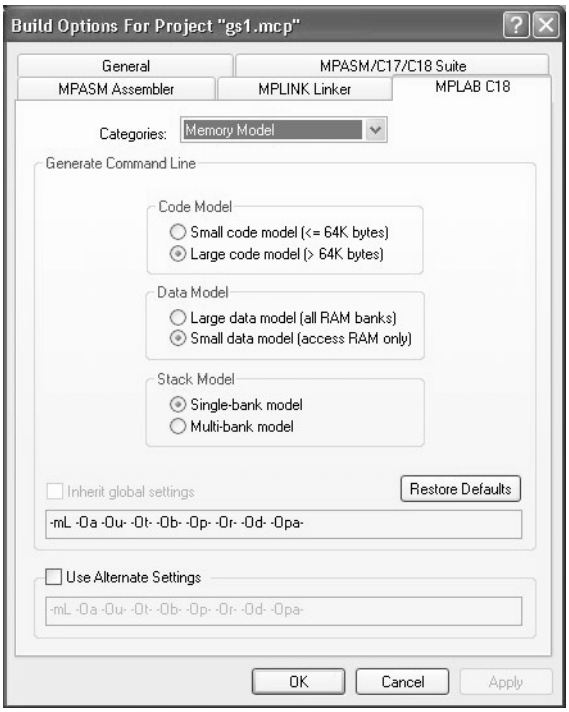
2.1.3 Set Memory Model

When using the standard libraries by including stdio.h, the large code model should be selected for the project.

Note: The standard libraries are built with the large code model, and if the large code model is not checked, a warning will be issued about a type qualifier mismatch.

Go to ***Project>Build Options>Project*** and select the **MPLAB C18** tab, then select **Categories: Memory Model** and check the **Large code model (> 64K bytes)**.

FIGURE 2-2: SELECT LARGE CODE MODEL



2.1.4 Test Program

Use ***Project>Build All*** or the equivalent icon to build the project. After a successful build, the **Run** icon becomes blue, indicating that the program is halted and ready to run. Select the **Run** icon and it turns gray, indicating it is running. The **Halt** icon turns blue, indicating the program is running and can be halted. In addition, on the status bar at the bottom is a “Running...” indicator. Select the **Halt** icon and open the Output window if it is not already open (Figure 2-3).

FIGURE 2-3: OUTPUT WINDOW: "HELLO, WORLD!"

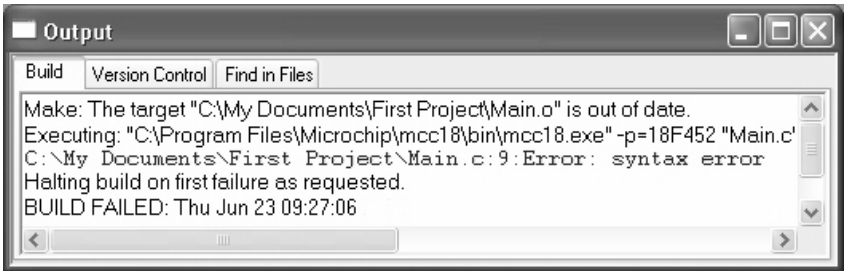
The text, "Hello, world!" should appear in the **SIM Uart1** tab of the Output window. Select the **Reset** icon to reset the program, and then select the **Run** icon again to print the message a second time in the Output window.

Note: After "Hello, world!" prints out, the program continues executing, running in an endless while (1) loop until it is halted. If **Run** is executed immediately after halting, the endless loop resumes running. In order to re-execute the program from the beginning, select the **Reset** icon after the program is halted.

2.1.5 Resolving Problems

If a mistype caused an error when building the project, the last lines in the Output window may look like Figure 2-4:

FIGURE 2-4: OUTPUT WINDOW SYNTAX ERROR



Double click on the line with “syntax error” to bring up the MPLAB Editor window with the cursor on the line with the syntax error (Figure 2-4).

- An error that reads “could not find stdio.h” usually means that the include path is not set up.
- A warning that reads “type qualifier mismatch in assignment” may mean that the large memory model is not selected when using standard I/O.
- An error that “c018i.o is not found” could mean that the library path is not set up correctly.
- An error that reads “could not find definition of symbol...” is usually caused by using the wrong linker script:
- Make sure that the 18f458.lkr file in the mcc18\bin\lkr directory is used. MPLAB IDE also has a linker script for assembler-only projects in one of its subdirectories. Always use the mcc18\bin\lkr linker scripts for all projects using the MPLAB C18 compiler.

If “Hello, world!” does not appear in the Output window, try these steps:

1. Make sure that the simulator is selected (***Debugger>Select Tool>MPLAB SIM***).
2. Make sure the **Uart1** is enabled to send printf() text to the MPLAB IDE Output window as shown in Figure 3-15 of the Project Basics handout.
3. Select the **Halt** icon (Figure 3-16).
4. **Build All** again. There should be no errors in the Output window, and the message “Build Succeeded” should appear as the last line (Figure 3-13).
5. Select the **Reset** icon on the Debug Toolbar (Figure 3-16).
6. Select the **Run** icon on the Debug Toolbar (Figure 3-16).

2.2 PROGRAM 2: BASIC CALCULATIONS

2.2.1 Make Program

Create a new project called **Basic Calculations** in the folder named **Lab 2**. Create a new file, type the code in Example 2-3 into it and save it as a file named main.c. Then, add the file main.c as the source file in this folder and add the 18f458.lkr linker script.

EXAMPLE 2-3: BASIC CALCULATIONS CODE

```
#include <stdio.h>

#pragma config WDT = OFF

void main (void)

{

   int a, b, c;

   a = 5;

   b = 7;

   c = a * b;

   printf("%d x %d = %d\n", a, b, c);

}
```

After you compiled and executed the program, what was the result? How many times was the output displayed in the output window? How would you modify the program to have the program display the output only once?

2.2.2 Program Explanation

Explain what each line of code does in the Basic Calculations example;

```
#include <stdio.h>
```

_____

```
#pragma config WDT = OFF
```

_____

```
void main (void)
```

_____

```
{
```

_____

```
int a, b, c;
```

_____

```
a = 5;
```

_____

```
b = 7;
```

_____

```
c = a * b;
```

_____

```
printf("%d x %d = %d\n", a, b, c);
```

_____

_____

```
}
```

_____

Exercises:

1. Modify the Hello World program to use a **for** loop to print the phrase five times.
2. Modify the Hello World program to use a **while** loop to print the phrase ten times.
3. Modify the Hello World program to use a **do/while** loop to print the phrase fifteen times.
4. Write a program to convert pounds to kilograms, from 0 pounds to 200 pounds in increments of 10 pounds. The program should display the table of results in the output window only once.

Submit a hardcopy of the program output and the source code stapled to the lab assignment cover sheet, for each modified program, in exercise 1, 2 and 3 and for the temperature conversion program as well as the answers to the questions. Be sure to document your code properly with plenty of meaningful comments.

Chapter 3. Project Basics and MPLAB IDE Configuration

3.1 INTRODUCTION

This section covers the basics of MPLAB projects and configuration options for testing the examples and applications in this guide with MPLAB SIM. This is intended as an overview and covers a generic application. Details on such things as device selection and linker scripts will vary with applications. This chapter can be skipped if these basic operations are known.

> **Note:** This is not a step-by-step procedure to create and build a project, but an overview and a checklist to ensure that MPLAB IDE is set up correctly. The *MPLAB IDE User's Guide* has a tutorial for creating projects.

Topics covered in this chapter are:

- Project Overview
- Creating a File
- Creating Projects
- Using the Project Window
- Configuring Language Tool Locations
- Verify Installation and Build Options
- Building and Testing

3.2 PROJECT OVERVIEW

Projects are groups of files associated with language tools, such as MPLAB C18, in the MPLAB IDE. A project consists of source files, header files, object files, library files and a linker script. Every project should have one or more source files and one linker script.

Typically, at least one header file is required to identify the register names of the target microcontroller. Header files are typically included by source files and are not explicitly added to the project.

The project's output files consist of executable code to be loaded into the target microcontroller as firmware. Debugging files are generated to help MPLAB IDE correlate the symbols and function names from the source files with the executable code and memory used for variable storage.

Most examples and applications in this guide consist of a project with only one source file and one linker script.

For additional information, refer to the *MPLAB® IDE Quick Start Guide* (DS51281).

3.3 CREATING A FILE

Start MPLAB IDE and select *File>New* to bring up a new empty source file. The examples and applications in this guide list source code that can be typed in, or copied and pasted into a text file using the MPLAB editor. Find example source in `mcc18\example\getting started`.

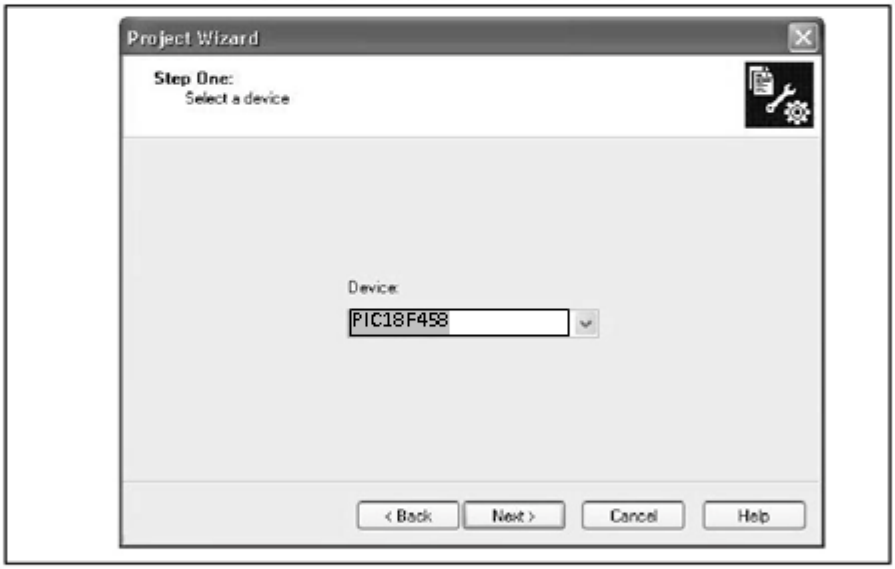
Type or copy the source text (as listed in each example in this manual) into this new file. (Text copied from examples in this document may not preserve white space.) Use *File>Save As* to save this file. Browse to or create a new folder location to store projects. Click **Save**.

> **Note:** Creating a new source file can be done either before or after creating a new project. The order is not important. Creating a new file does not automatically add that file to the currently open project.

3.4 CREATING PROJECTS

1. Select *Project>Project Wizard* to create a new project. When the Welcome screen displays, click **Next>** to continue.
2. At "Step One: Select a device", use the pull-down menu to select the device.

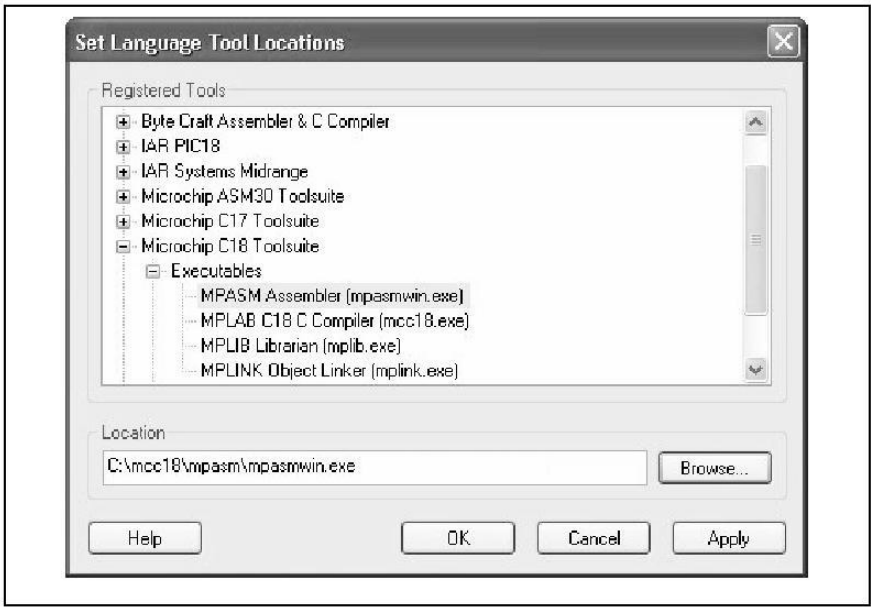
FIGURE 3-1: PROJECT WIZARD – SELECT DEVICE



Click **Next>** to continue.

3. At “Step Two: Select a language toolsuite”, choose “Microchip C18 Toolsuite” as the “Active Toolsuite”. Then click on each language tool in the toolsuite (under “Toolsuite Contents”) and check or set up its associated executable location (Figure 3-2).

FIGURE 3-2: PROJECT WIZARD – SELECT LANGUAGE TOOLSUITE



MPASM Assembler should point to the assembler executable, `MPASMWIN.exe`, under “Location”. If it does not, enter or browse to the executable location, which is by default:

`C:\mcc18\mpasm\MPASMWIN.exe`

MPLAB C18 C Compiler should point to the compiler executable, `mcc18.exe`, under “Location”. If it does not, enter or browse to the executable location, which is by default:

`C:\mcc18\bin\mcc18.exe`

MPLINK Object Linker should point to the linker executable, `MPLink.exe`, under “Location”. If it does not, enter or browse to the executable location, which is by default:

`C:\mcc18\bin\MPLink.exe`

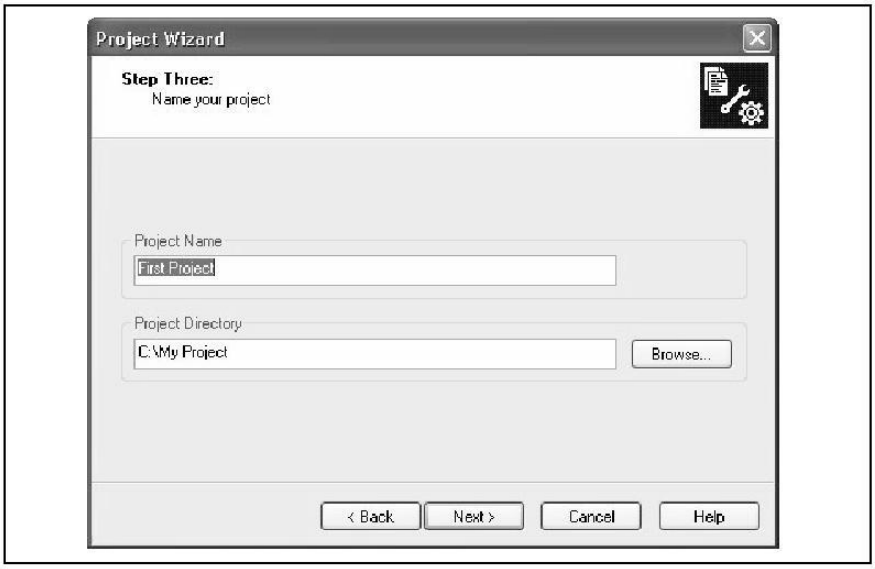
MPLIB Librarian should point to the library executable, `MPLib.exe`, under “Location”. If it does not, enter or browse to the executable location, which is by default:

`C:\mcc18\bin\MPLib.exe`

Click **Next>** to continue.

4. At “Step Three: Name your project” (Figure 3-3), enter the name of the project and use **Browse** to select the folder where the project will be saved. Then click **Next>** to continue.

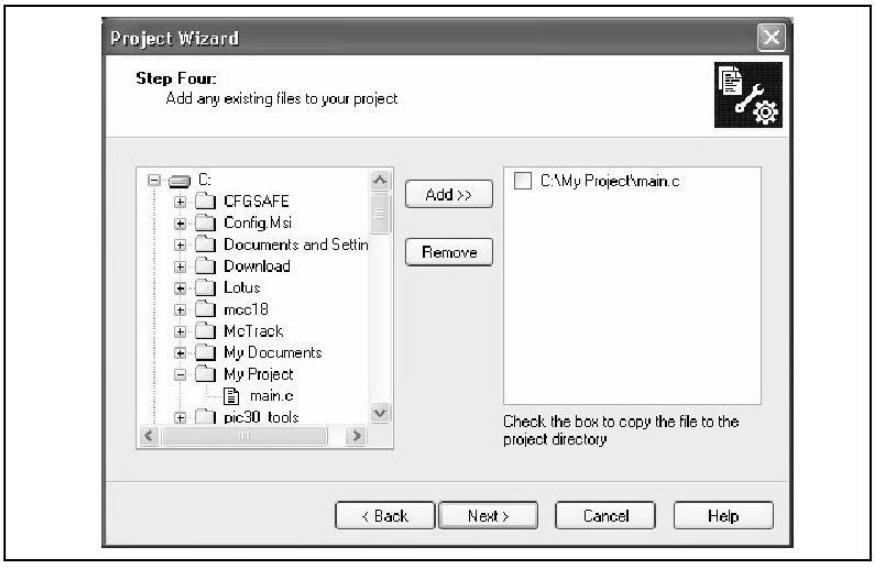
FIGURE 3-3: PROJECT WIZARD – PROJECT NAME AND DIRECTORY



5. At “Step Four: Add any existing files to your project”, navigate to the source file to be added to the project.

   First, select the source file created earlier. If source files have not yet been created, they can be added later (see Figure 3-4). Click **ADD>>** to add it to the list of files to be used for this project (on the right).

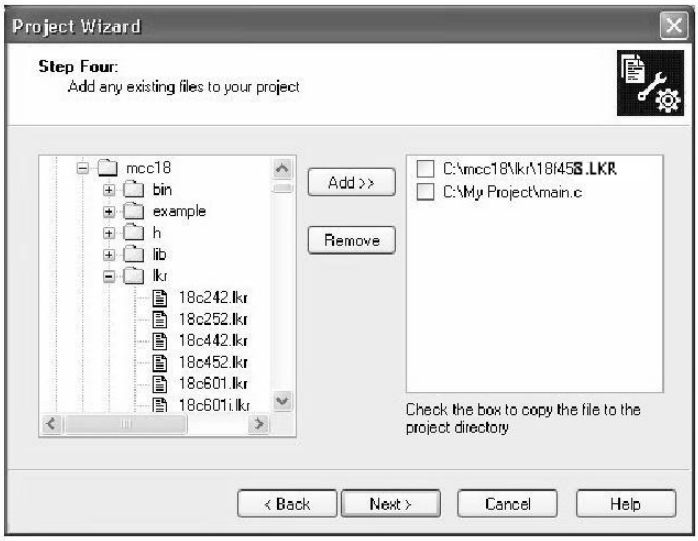
FIGURE 3-4: PROJECT WIZARD – ADD C SOURCE FILE



Second, a linker script file must be added to tell the linker about the memory organization of the selected device. Linker scripts are located in the `lkr` subfolder of the installation directory for MPLAB C18. Scroll down to the `.lkr` file for the selected device, click on it to highlight and click **ADD>>** to add the file to the project. See example in Figure 3-5. Select **Next>** to continue.

Note: There are also linker scripts delivered with MPASM when it is installed with MPLAB IDE. Make sure to use the linker scripts in the `\mcc18\lkr` folder.

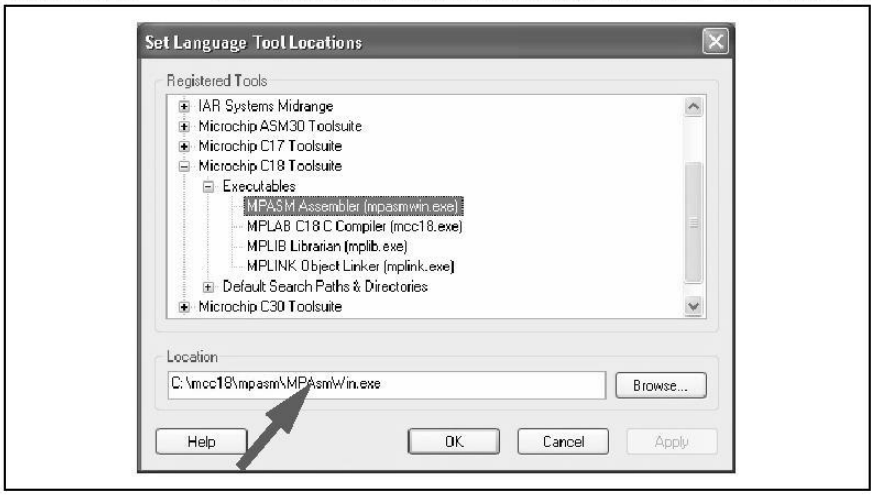
FIGURE 3-5: PROJECT WIZARD – ADD LINKER SCRIPT



6. At the Summary screen, review the "Project Parameters" to verify that the device, toolsuite and project file location are correct. Use **<Back** to return to a previous wizard dialog. Click **Finish** to create the new project and workspace. Click **OK** to exit.

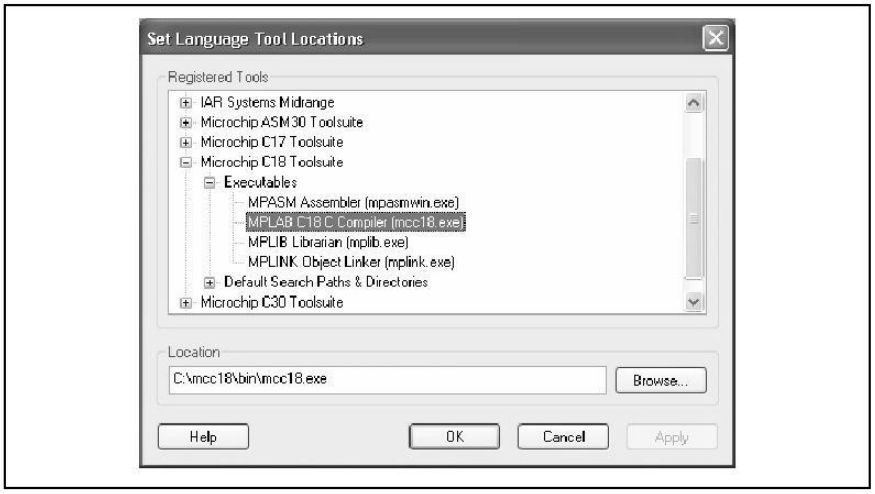
For MPASM Assembler, verify its location is `C:\mcc18\mpasm\MPASMWIN.exe` as shown in Figure 3-7.

FIGURE 3-7: SET LANGUAGE TOOL LOCATIONS: MPASM™ ASSEMBLER



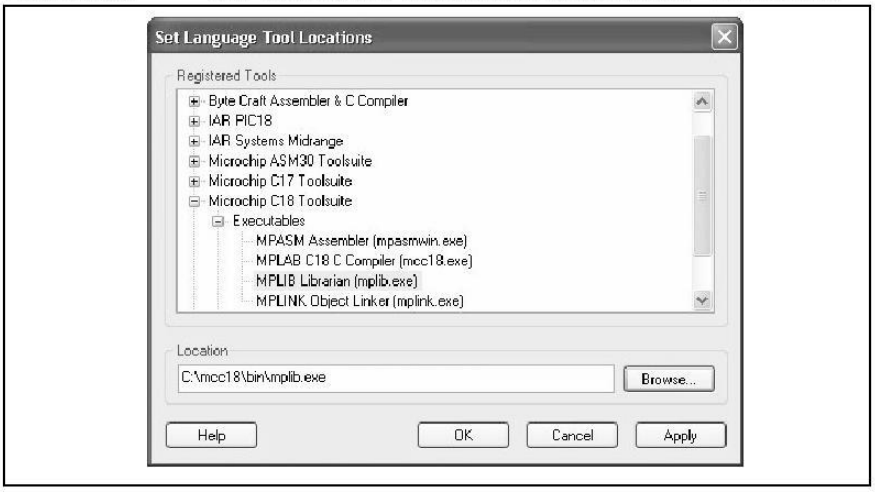
For MPLAB C18 compiler executable, verify its location is `C:\mcc18\bin\mcc18.exe` as shown in Figure 3-8.

FIGURE 3-8: SET LANGUAGE TOOL LOCATIONS: MPLAB® C18



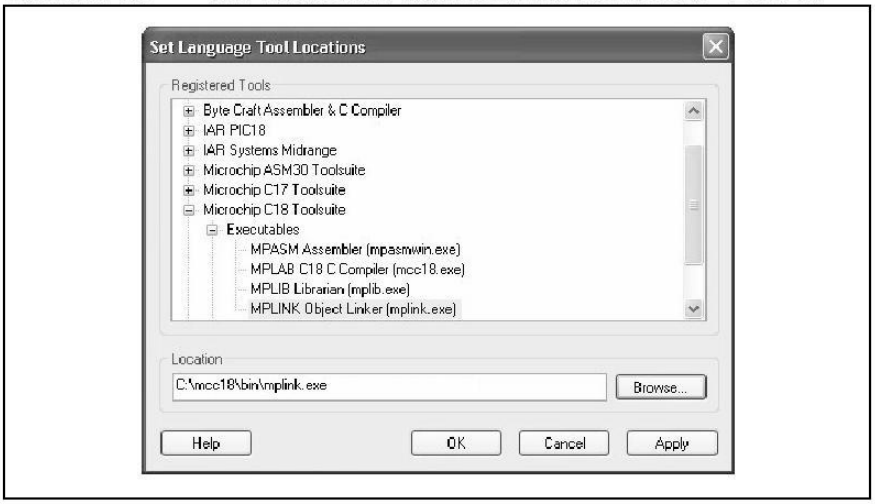
For MPLIB Librarian (part of the compiler package executables), verify its location is `C:\mcc18\bin\MPLib.exe` as shown in Figure 3-9.

FIGURE 3-9: SET LANGUAGE TOOL LOCATIONS: MPLIB™ LIBRARIAN



And for the MPLINK Linker, ensure that its location is `C:\mcc18\bin\MPLink.exe` as shown in Figure 3-10.

FIGURE 3-10: SET LANGUAGE TOOL LOCATIONS: MPLINK™ LINKER



Click **OK** to save these settings and close this dialog.

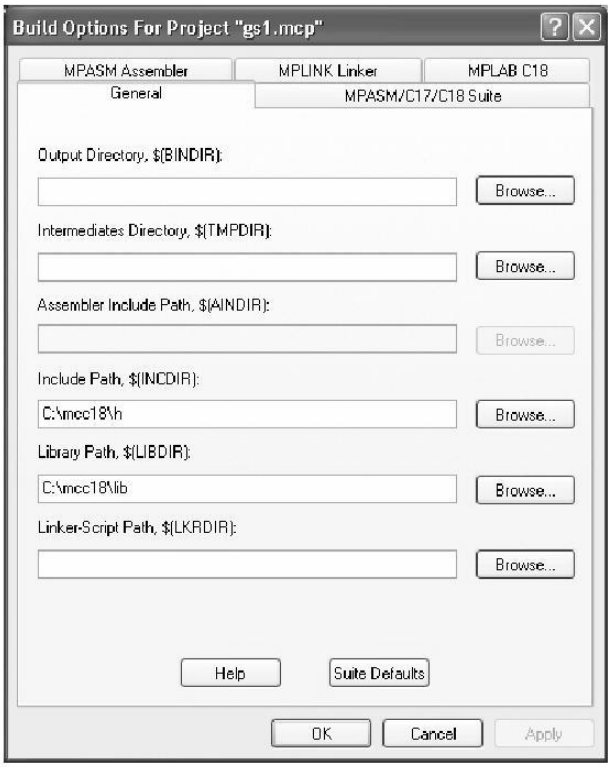
3.7 VERIFY INSTALLATION AND BUILD OPTIONS

Before proceeding with compiling and testing programs, the installation and project settings should be verified.

The language tools should be installed correctly and the settings should be appropriate for these first examples of code, otherwise errors may result. Follow through with these checks:

1. Select the *Project>Build Options...>Project*, and click on the **General** tab. If the **Include Path** and the **Library Path** are not set as shown in Figure 3-11, use the **Browse** button to locate these folders in the MPLAB C18 installation folder.

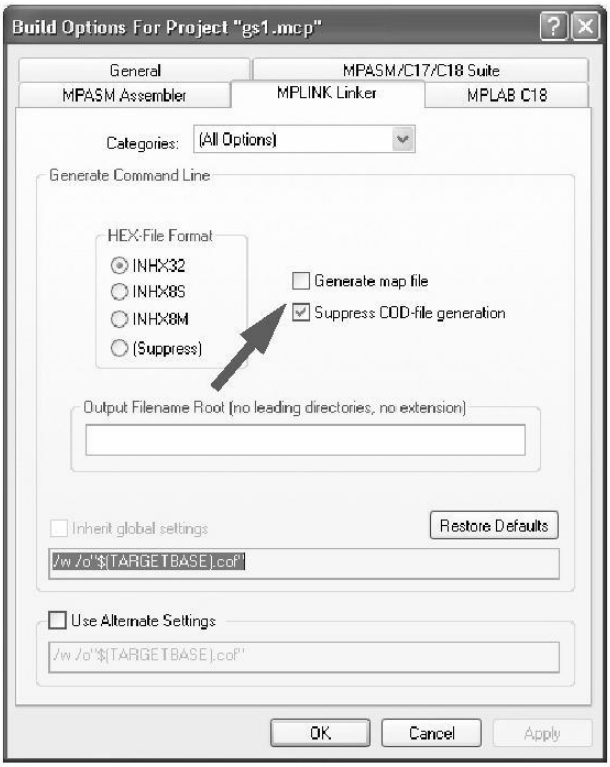
FIGURE 3-11: BUILD OPTIONS: GENERAL



Note: Multiple paths can be entered for a single include or library search path by separating them with a semicolon:
`c:\myprojects\h;c:\mcc18\h`.

2. One option may need to be changed from the default. Click on the **MPLINK Linker** tab. If it is not checked, click on the box labeled **Suppress COD-file generation**:

FIGURE 3-12: BUILD OPTIONS: MPLINK™ LINKER



Note: If this box is not checked, the linker will also generate an older `.cod` file type, which is no longer used by MPLAB IDE. This file format has a file/path length limitation of 62 characters which will cause this error: "`name exceeds file format maximum of 62 characters`".

Click **OK** to close this dialog.

3.8 BUILDING AND TESTING

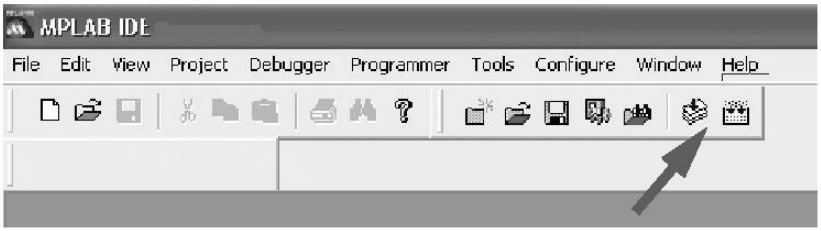
3.8.1 Build Project

If everything is installed as instructed, the project can be built using the menu selection *Project>Build All* or *Project>Make*.

> **Note:** Compiling and linking all the files in a project is called a "make" or a "build". **Build All** will recompile all source files in a project, while **Make** will only recompile those that have changed since the last build, usually resulting in a faster build, especially if projects have many source files.

Shortcut keys, **Ctrl+F10** and **F10**, can be used instead of selecting items from the menu. There are icons on the toolbar for these functions as well, so either one function key or one mouse click will build the project:

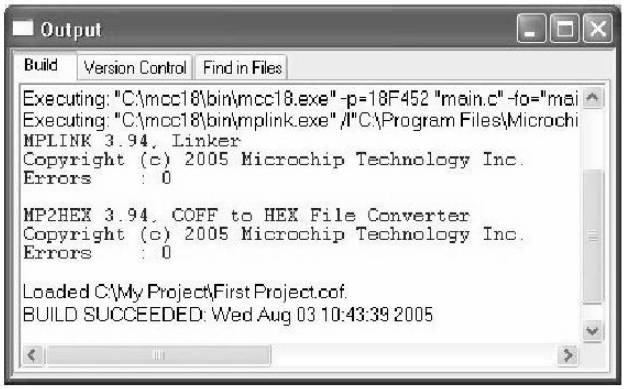
FIGURE 3-13: BUILD ALL AND MAKE ICONS



> **Note:** By moving the cursor over these icons, a pop-up will identify their function.

The project should build correctly as seen in the Output window:

FIGURE 3-14: OUTPUT WINDOW AFTER SUCCESSFUL BUILD



If the message "`Errors : 0`" is not shown from both MPLINK (the Linker) and MP2HEX (the `.hex` file converter), something may have been mistyped. Expand the Output window and look for the first error. If it was a mistype, then double click on that error line in the Output window to edit the error in the file `main.c`. If there was some other error, see **Chapter 7. "Troubleshooting"**.

3.8.2 Testing with MPLAB® SIM

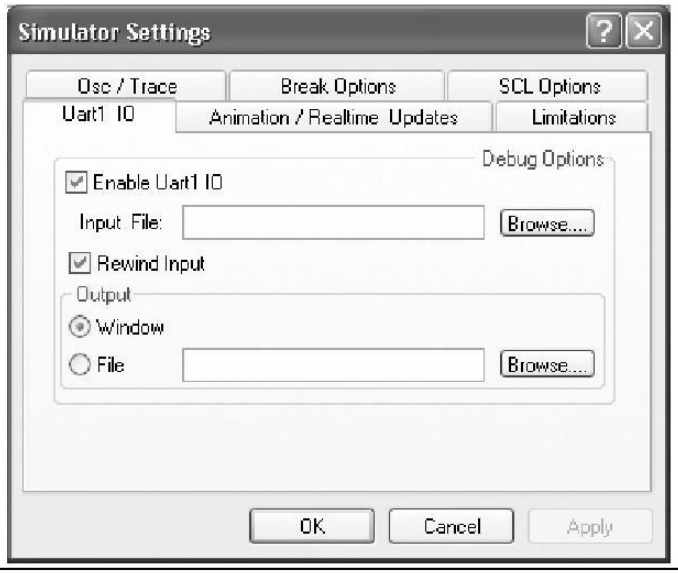
To test these programs in MPLAB IDE, use the built-in simulator, MPLAB SIM.

1. To enable the simulator, select *Debugger>Select Tool>MPLAB SIM*.

   The project should be rebuilt when a debug tool is changed because program memory may be cleared.

2. Select *Debugger>Settings* and click on the **Uart1 IO** tab. The box marked **Enable Uart1 IO** should be checked, and the **Output** should be set to **Window** as shown in Figure 3-15:

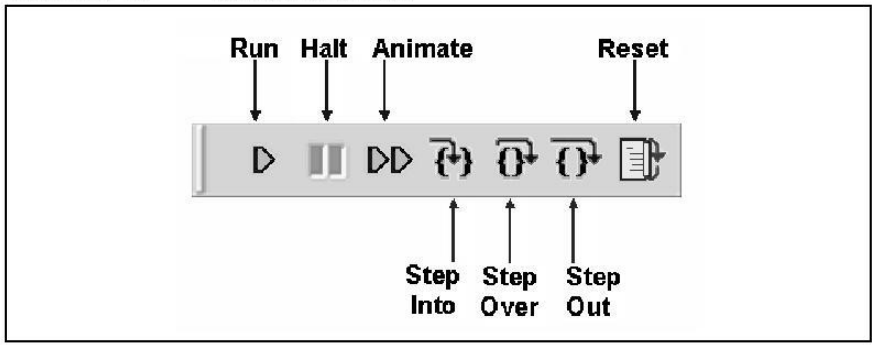
FIGURE 3-15: SIMULATOR SETTINGS: UART1



Note: This dialog box allows the text from the `printf()` function to go to the simulator's UART (a serial I/O peripheral), and then to the MPLAB IDE Output window.

After the simulator is selected, the **Debug Toolbar** (Figure 3-16) appears under the MPLAB menus.

FIGURE 3-16: DEBUG TOOLBAR



Icon	Function
Run	Run program
Halt	Halt program execution
Animate	Continually step into instructions. To Halt, use *Debugger>Halt* or press the Halt icon.
Step Into	Step into the next instruction
Step Over	Step over the next instruction
Step Out	Step out of the subroutine
Reset	Perform a $\overline{\text{MCLR}}$ Reset

See the *MPLAB® IDE User's Guide* for more information on projects, MPLAB configuration and extended debugging techniques.

NOTES: