Revision: 21 Jan 2014 (JFF)
Dr. Richard Wall, University of Idaho, rwall@uidaho.edu

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Project 0: MPLAB ® X – Integrated Development Environment

Table of Contents

Purpose

It is the purpose of this project to teach some of the aspects of an important tool that can assist in software development. This project introduces you to the synthesis and analysis tools for producing microprocessor C code using the MPLAB ® X integrated development environment (IDE) on the Cerebot MX7cK processor board. It is not the purpose of this project to teach the art and science of software development nor is it to teach C programming.

Minimum Knowledge and Programming Skills

1. Knowledge of C or C++ programming
2. Fundamental understanding of computer architecture

Equipment List

1. Digilent Cerebot 32MX7cK processor board with USB cable
2. Microchip MPLAB ® X IDE
3. Microchip MPLAB ® XC32 Compiler

Software Resources

1. XC32 C/C++ Compiler Users Guide
2. PIC32 Peripheral Libraries for MBLAB C32 Compiler
3. PIC32 Family Hardware Reference Manual Section 16 Output Compare
4. Cerebot MX7cK Board Reference Manual
5. MPLAB X Integrated Development Environment (IDE)
6. C Programming Reference

Overview

Regardless of previous experience, all MPLAB X users should be aware of the MPLAB X Integrated Development Environment (IDE) web site has documentation available under Help on the MPLAB ® tool bar. The primary software development tool is the integrated development environment (IDE) that provides an interface between the development platform and the target embedded system. The target system for this project is the Digilent Cerebot MX7cK processor board. Conventional IDE systems consist of software that runs on the PC or Linux computer and special hardware that manages the microcontroller on the target system. A separate PIC microcontroller on the Cerebot 32MXcK processor board provides the special hardware needed to interface with the PC that is running the MPLAB X IDE. The IDE assists the program developer in converting one or more text files written using C or assembly language instructions into a file that contains a set of binary instructions that is programmed into a microprocessor.

The MPLAB X IDE does this transformation in four steps. First, the editor is where the programmer writes his or her C language programs. A project consists of one or more text files that the compiler converts into one or more object files. The object files are combined using a linker that generates a single file that the loader then programs into the microprocessor.

Source files with the ".c" extension contain C statements that define the functionality of the file such as the interface to an LCD or the implementation of a software time delay. Files with the ".h" extension are called header files that contain C instructions and directives that allow the source code to be reused in different application. Generally each ".c" file has a companion ".h" file that contains function prototypes and definitions.

The MPLAB X editor has many features to help you to write C code that the compiler can convert into an object file. The compiler checks each source file for proper C syntax and that all references are resolved. Compiler errors are identified with the file name, the line number in the file where the error was detected, and a description of the error. The link and load processes will be completed only if the compiler completes its task without finding any errors. There may be times that the compiler generates warnings. In such cases, the link and load operations will still be completed. Compiler warnings are generated if the source code attempts to give the compiler an instruction that is ambiguous or confusing but the syntax is correct. In these cases, the compiler makes a reasonable but not necessarily correct interpretation of your instruction. Clean code is code that compiles with no errors or warnings when using the setting for the highest level of error and warning generation.

Project_0 consists of five text files: Project0.c, Project0.h, CerebotMX7cK.c, CerebotMX7cK.h, and config_bits.h. Project0.c contains the C source code for this application that contains the function "main". Every project must have one and only one source file containing the function "main". Project0.h contains the function prototypes as well as "#define" statements used in Project0.c.

The *config_bits.h* file contains pragma directives that configures the operating characteristics of the PIC32 processor. The reader should refer to the Cerebot MX7cK Board Reference Manual for additional information concerning the directives in *config_bits.h*. Files *CerebotMX7cK.c* and *CerebotMX7cK.h* contain functions that initialize the PIC32 processor for the interface to hardware on the Digilent Cerebot MX7cK processor board. These three files will be used in this project as well as all future projects.

A fundamental skill to learn in this project is how to use the MPLAB IDE to develop software that runs on the PIC32 processor. The steps in this exercise are designed to show you how to create a new stand-alone project and how to use the software instrumentation to verify that the program is executing as designed. You will use "cursor over variable" and "watch window" to allow you to observe the values of data variables. You will use single stepping and break points to examine the sequence of program execution. You will also use the compiler error messages to assist you in finding syntactical errors.

Another commonly overlooked software development tool is the common handheld calculator. Computer algorithms can become convoluted and complex. For the synthesis phase of code development, it is good practice to write out the algorithm as a series of expressions that be implemented using a series of steps with a calculator. Then, convert these expressions into C program statements. For the code development test phase, generate a set of input parameters and work through your series of expressions using the calculator and record the results for each step. Finally run the C program using the same set of parameters. If the results do not match, then there is an error in your expressions or C code, or possibly both. Thorough algorithm testing requires that the input parameters vary over the possible range of values.

MPLAB ® X IDE Tutorial

The following tutorial is specific to the Cerebot MX7cK processor board that using the "licensed debugger" provided by Digilent. Microchip provides additional tutorials that the reader should also use to become familiar with the tools and features of MPLAB X IDE.

Connecting the Cerebot MX7cK Processor Board

Figure 1 is a picture of the Cerebot MX7cK processor board. Before connecting the debugging USB cable to the PC, verify that the Board Power select jumpers shown at the upper left are positioned correctly. As shown, the processor board is powered using the debugging PC USB connection. The board power switch is located in the lower left corner. Once the board power is turned on, the RED power indication LED should be on. It is helpful to connect the processor board with the debugging USB cable and power it on before launching a new MPLAB X project.

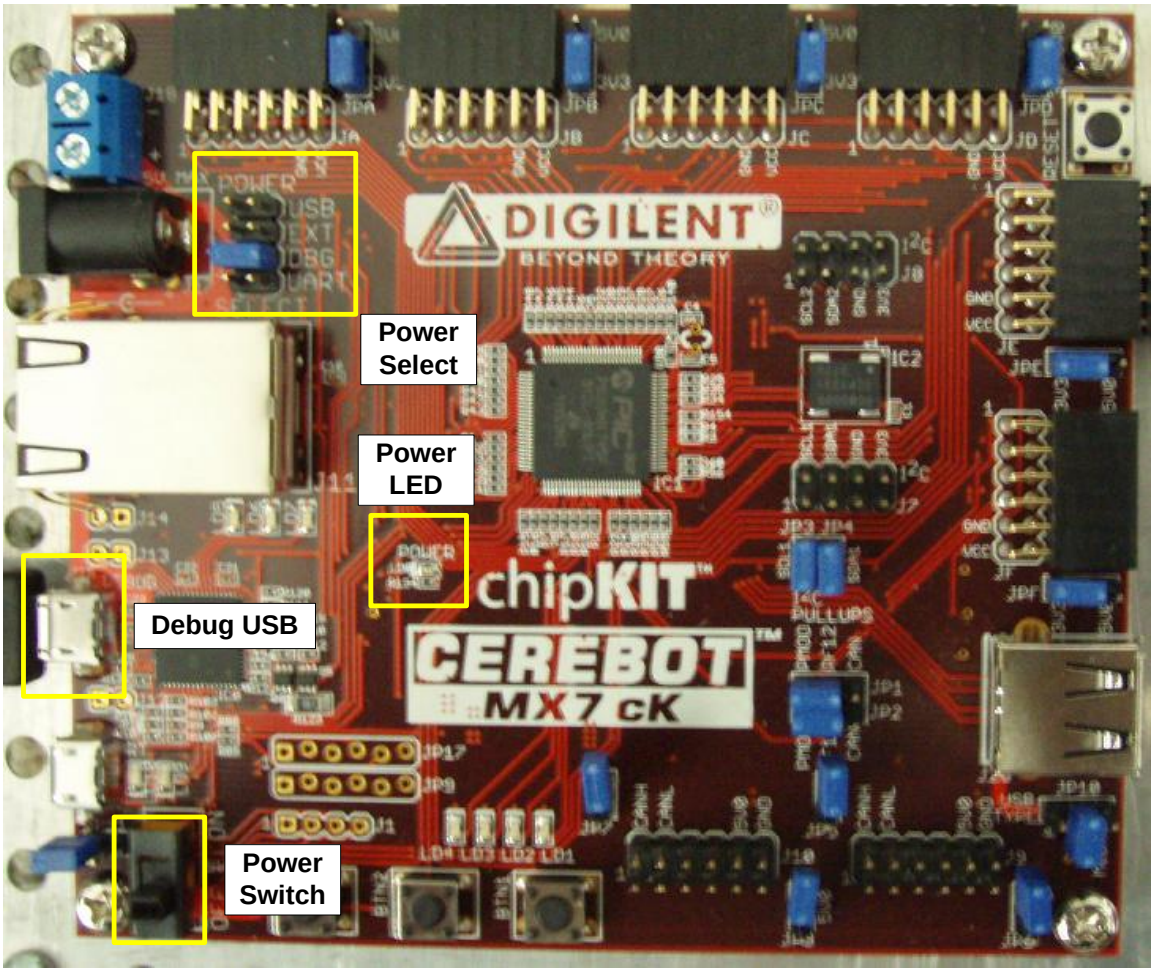

Figure 1. Cerebot MX7cK processor board

Launching a New MPLAB ® X Project

A. Launch the MPLAB X IDE by selecting the MPLAB X icon on the PC desktop.
B. Initially, a window for a blank project page as shown in Figure 2 will appear.

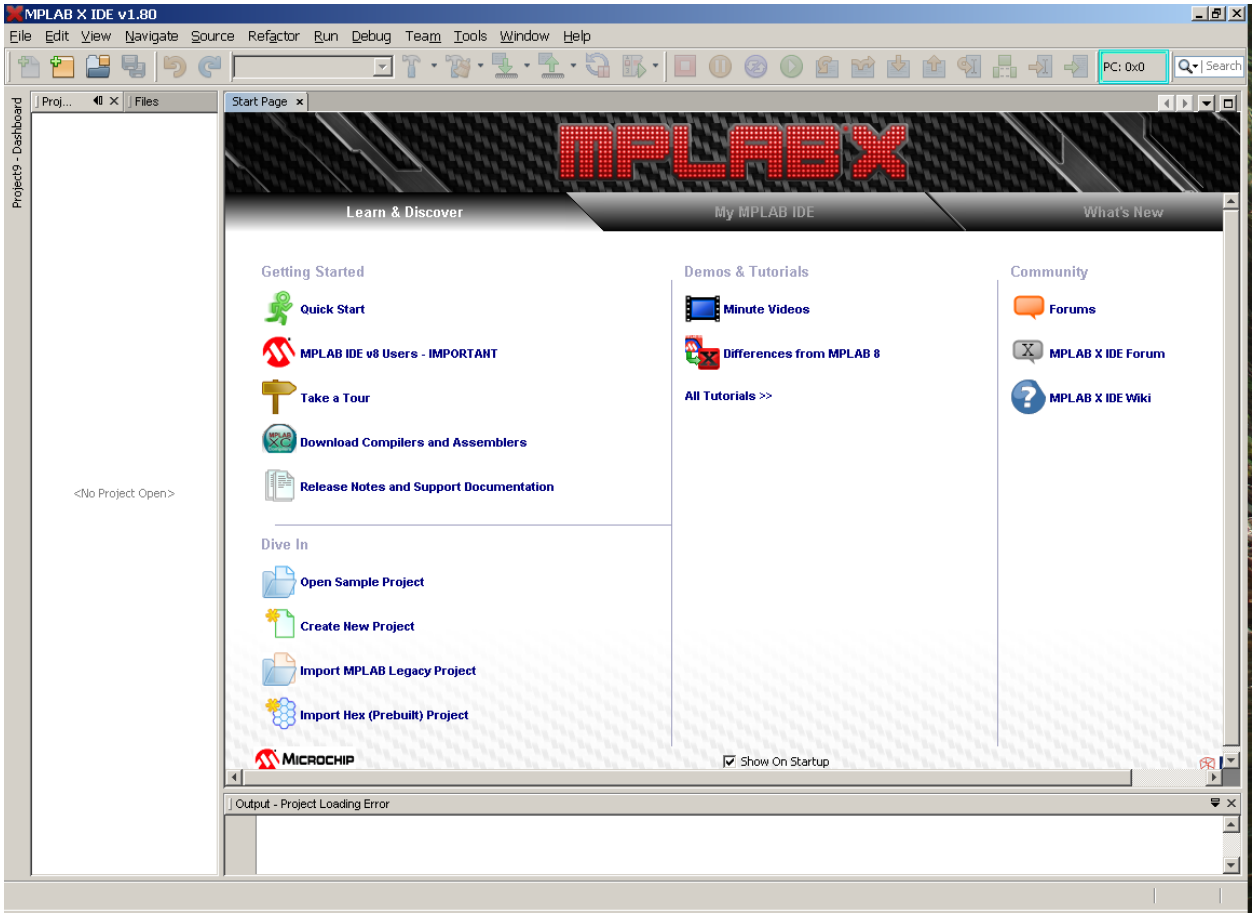

Figure 2. MPLAB X IDE window

C. First time users are encouraged to watch the *Minute Videos* by selecting the movie icon in the *Demos and Tutorials* column. The available topics are displayed in a window as shown in Figure 3. In particular, new users should watch the video titled, *Creating a New Project*. Users who have projects developed using MPLAB 8x and earlier versions should also view the video titled *Converting an MPLAB 8.xx Project to MPLAB X*. These videos target different PIC products but provide a good overview of the process that will be presented in the following steps for the Cerebot MX7cK processor board.

+

Figure 3. MPLAB X *Movie* window

D. After viewing the appropriate tutorial(s), start the process of creating a new MPLAB X project by clicking of the "Create a New Project" link in the *Drive In* box shown on the MPLAB X Start Page. A new window will be launched that walks you through the process described in the *New Project* minute video. Appropriate step 1 selections for Project 0 are shown in Figure 4. A standalone project package all source files into a single project. The selections shown in Figure 4 are the MPLAB X 1.80 default selections.

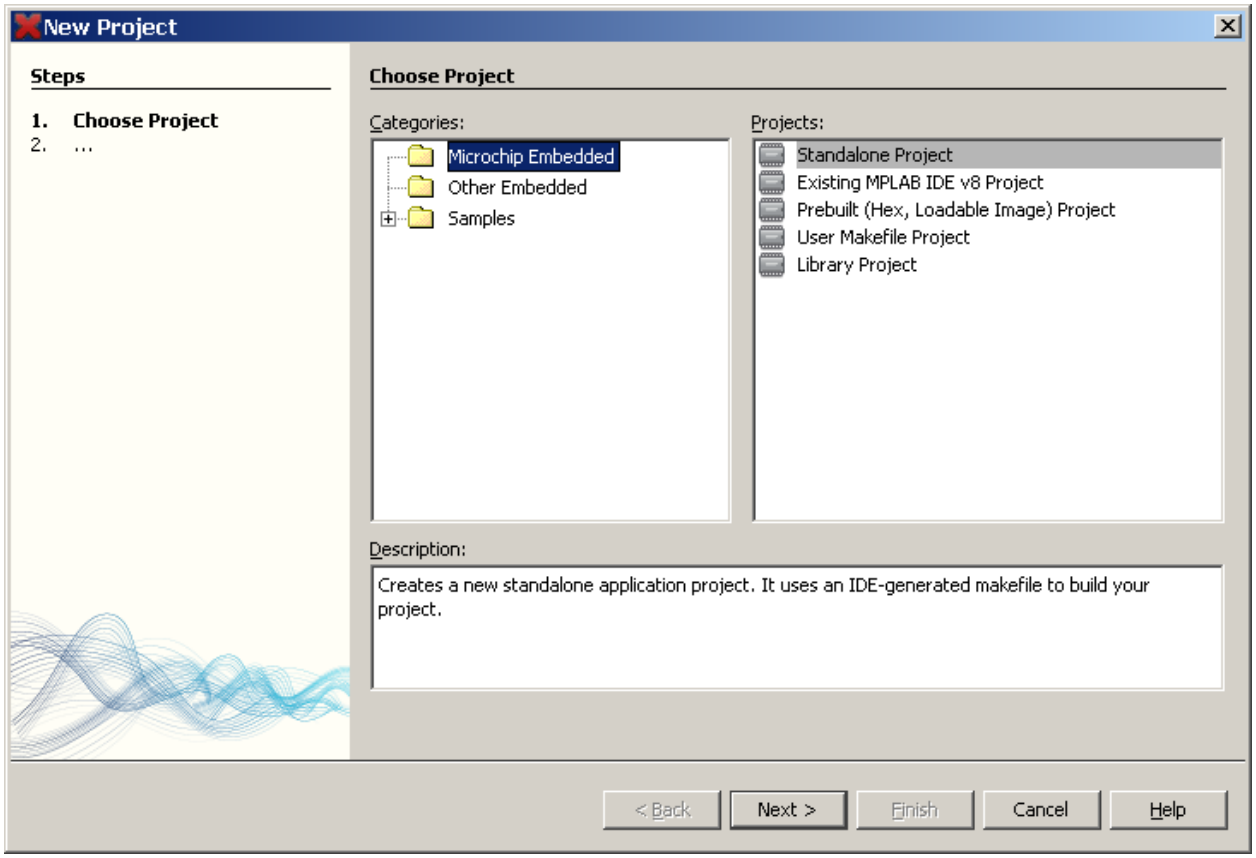

Figure 4. Step 1 of creating a new MPLAB X project

E. After making the selections for step 1, you will need to select the processor that is on the Cerebot MX7cK processor board. Step 2 is divided into two parts for simplicity: Figure 5 shows the selection of the device family as *32-bit MCUs(PIC32)*. Figure 6 shows that the processor device we are using is the *PIC32MX795F512L*. Figure 7 shows the step 2 window after selecting the Cerebot MX7cK processor board. Click on the **Next>** control to continue to step 4. (Note that we skip step 3.)

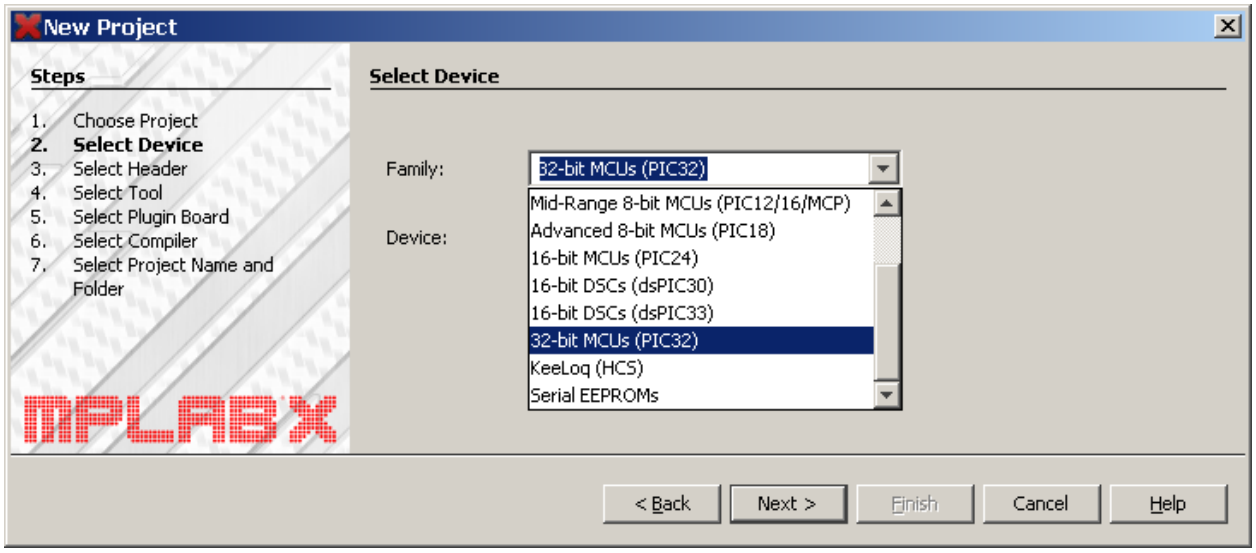

Figure 5. Step 2a of creating a new MPLAB X project

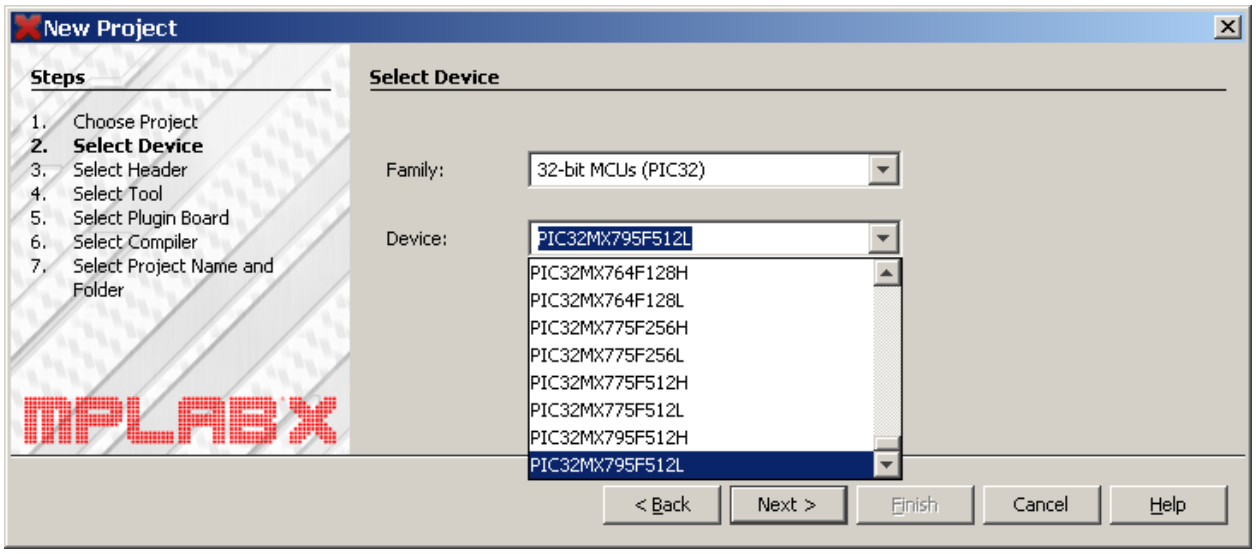

Figure 6. Step 2b of creating a new MPLAB X project

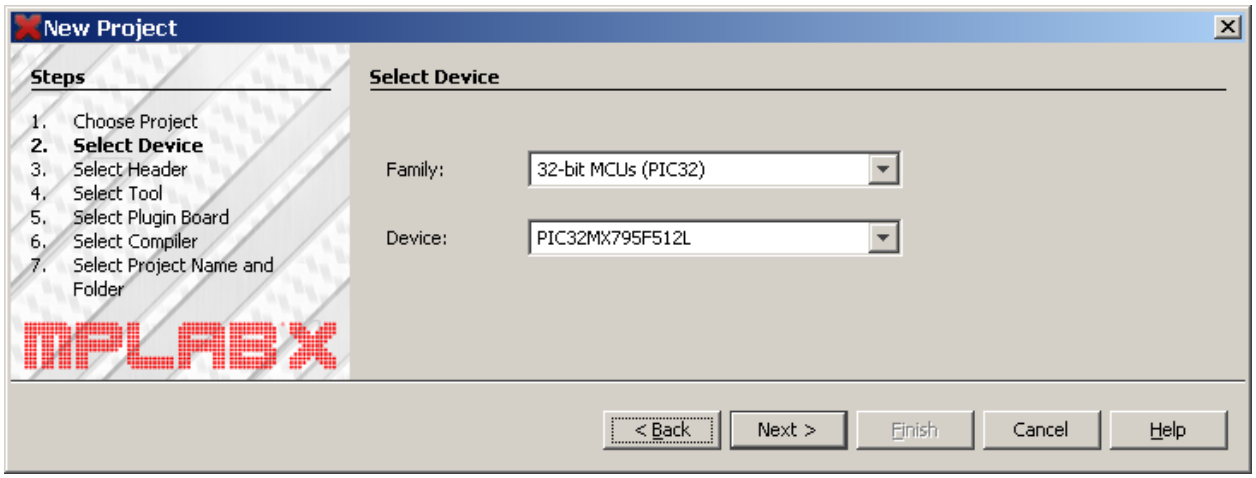

Figure 7. Step 2 after making device selection for the Cerebot MX7cK processor board

F. In step 4, we select the debugging tool using the window shown in Figure 8. The debugging tool provided by Digilent for the Cerebot MX7cK processor board appears under the Other Tools folder. If a Cerebot MX7cK processor is connected to the PC via the USB cable and the power to the board is turned on, the board's serial number will show on the list. Note: After the project has been built, if the project launched when the PC is connected to a different Cerebot MX7cK processor board, MPLAB X will prompt you to approve the connection to the new board.

   Click on the **Next>** control to continue to step 6. (Note that we skip step 5.)

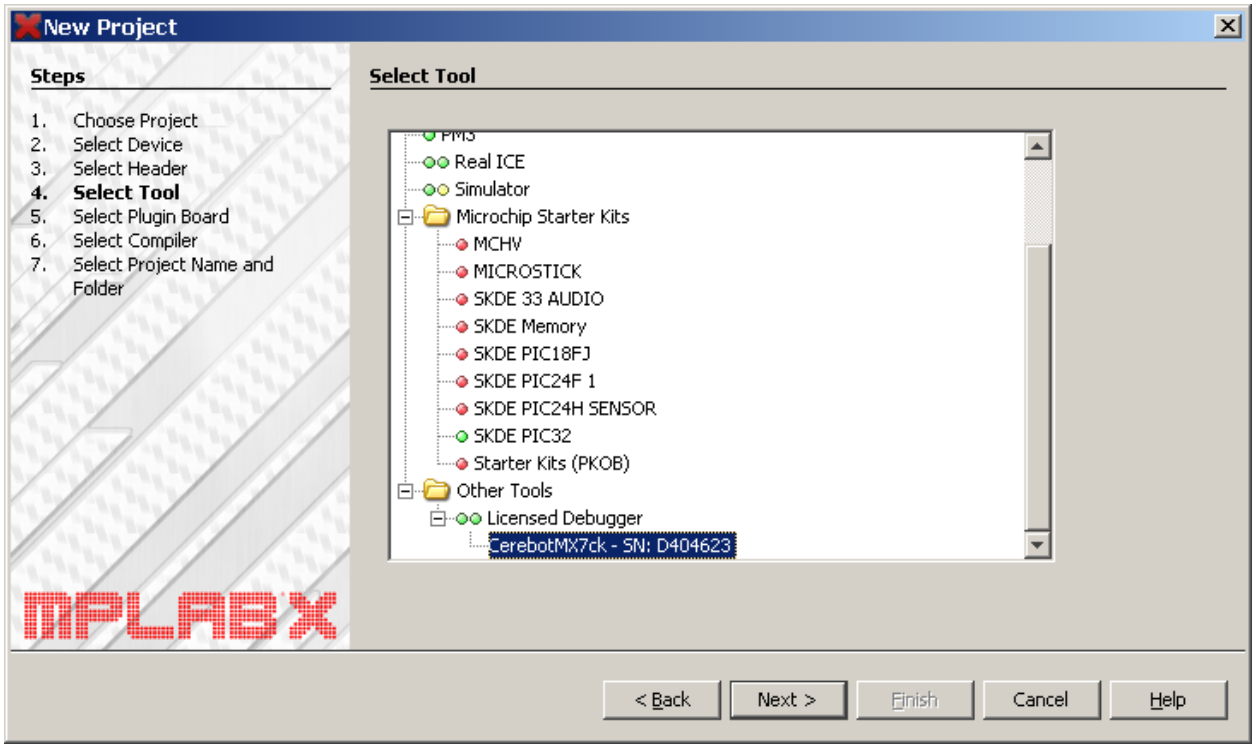

Figure 8. Step 4 window that selects the debugging hardware

G. Step 6 enables the selection of the C compiler to use from the list of C compilers found on the PC that you are currently working on. Figure 9 shows that this particular PC has both the C32(v2.02) complier and the XC32(v1.21). This project will use the XC32 compiler. Click on the **Next>** control to continue to the final step.

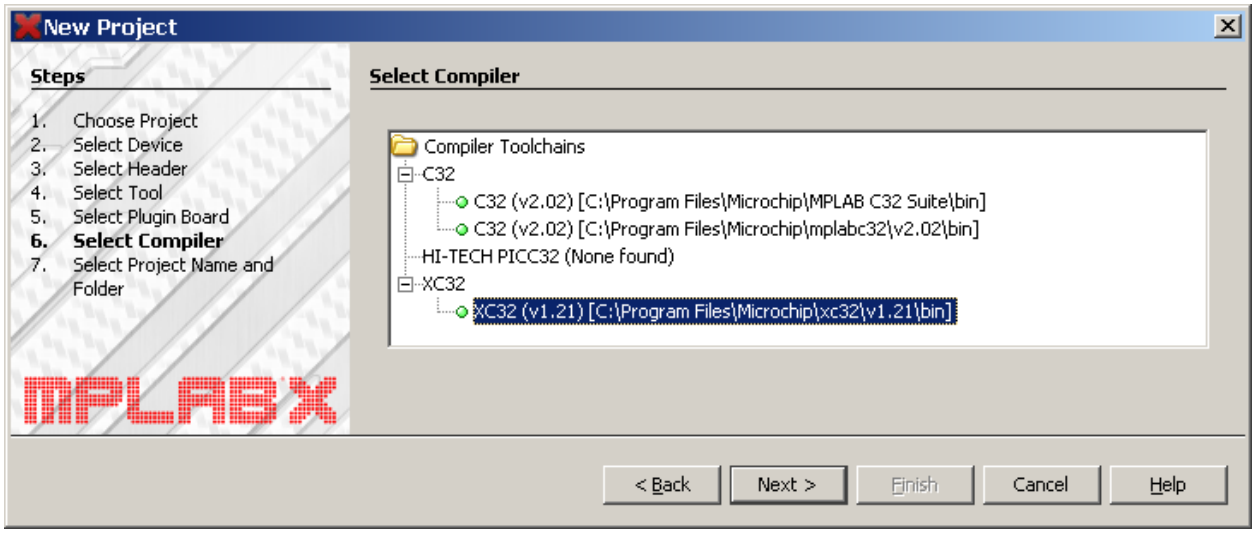

Figure 9. Step 6 for selecting the C compiler

H. Copy the five source files listed above into the Project 0 project folder. As shown in Figure 10, the last step in creating a project, allows you to specify where the project will be stored. This example shows that the source files are located in a folder named C:/code. You will specify a file location of your own choosing. The project Location is where we will put the five

source files for Project 0. Use the default setting for the remaining options. Click on **Finish** to complete the process of creating a project. The MPLAB X IDE window should now appear as shown in Figure 11. (The Encoding selection shown in Figure 10 refers to the keyboard character set.)

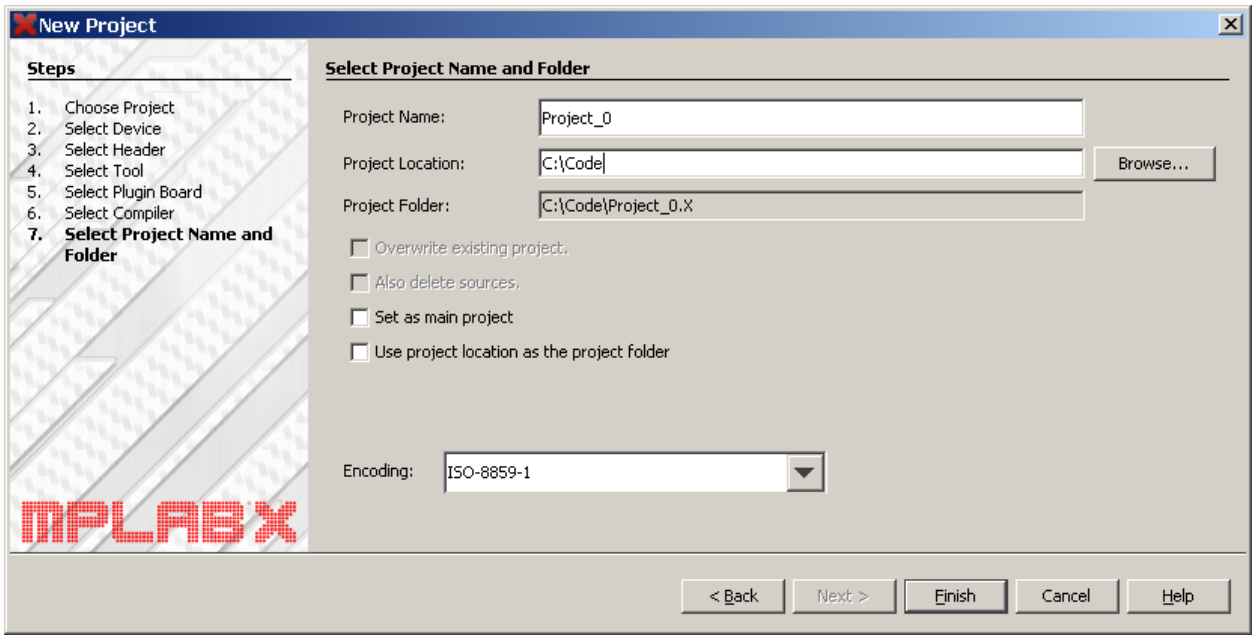

Figure 10. Project location

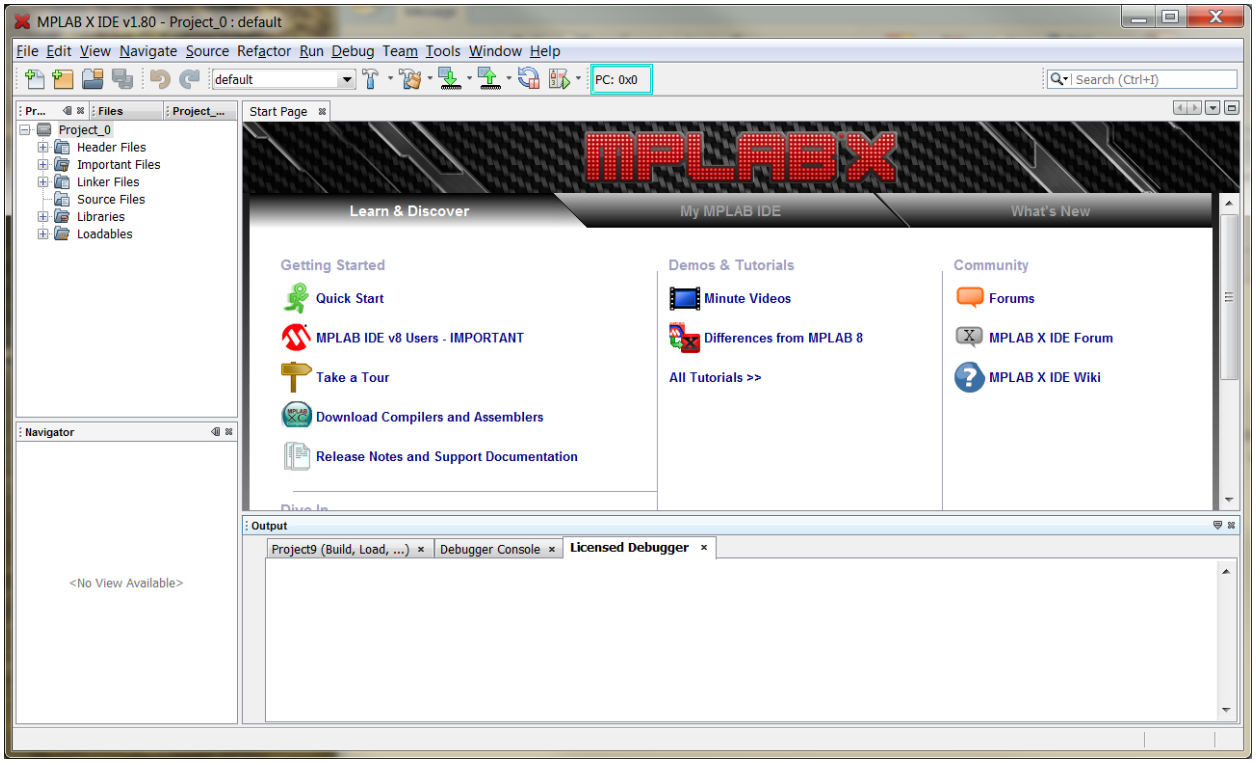

Figure 11. MPLAB X screen after creating Project 0

I. In the Project box, right click on the Headers Files folder and select the option to "*Add existing item…*". Select the three files with the ".h" extension shown in Figure 12 then click on the Select button. Using the *Relative* store path option allows projects to be easily move from one development PC to another where the directory structure may be different.

Repeat this for the Source Files folder selecting the files with the ".c" extension as shown in Figure 13. Expanding the Header Files folder and the Source Files folder should result in the MPLAB X window looking as shown in Figure 14.

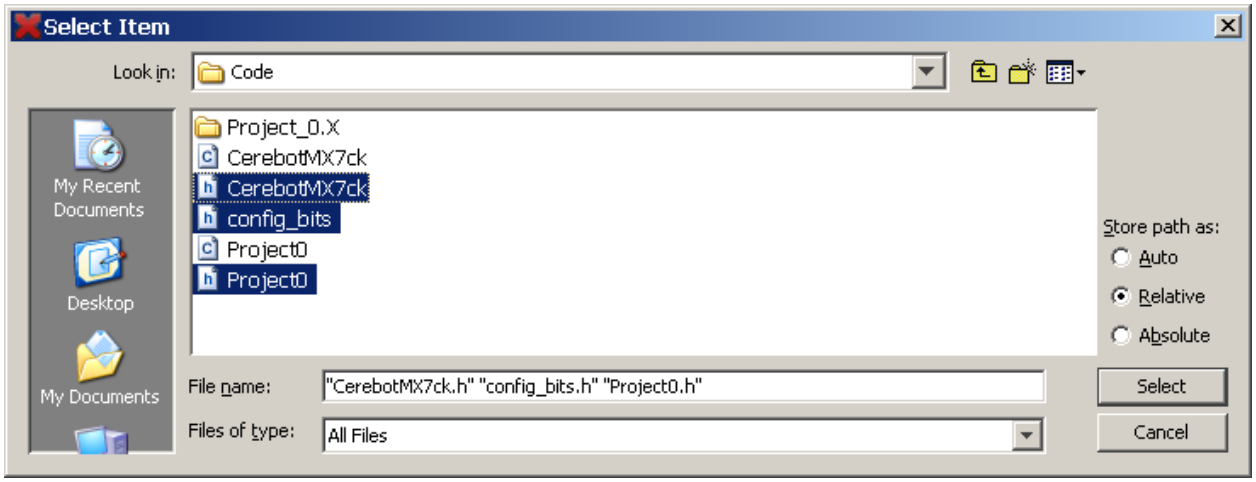

Figure 12. Selection of header files for Project 0

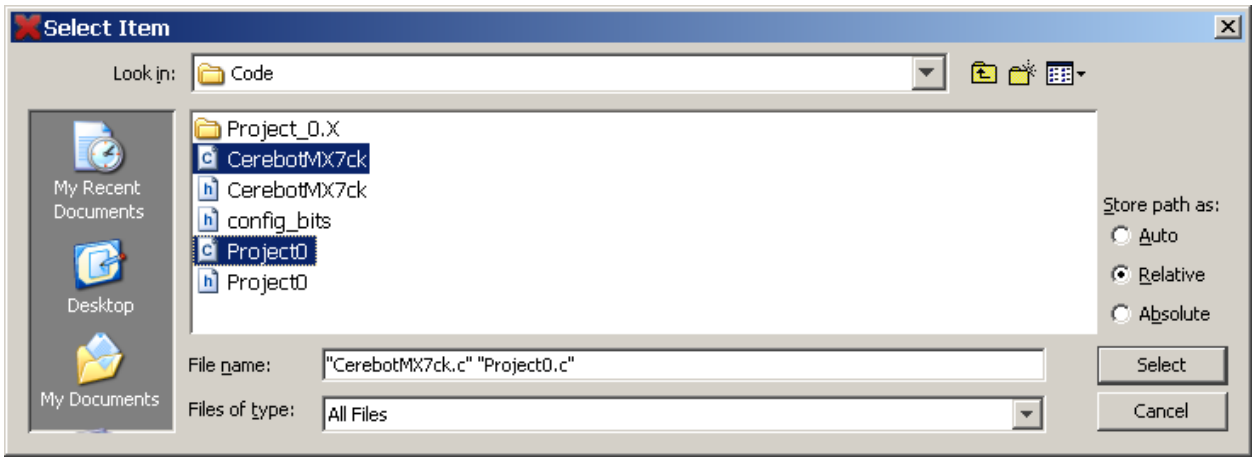

Figure 13. Selection of C source files for Project 0

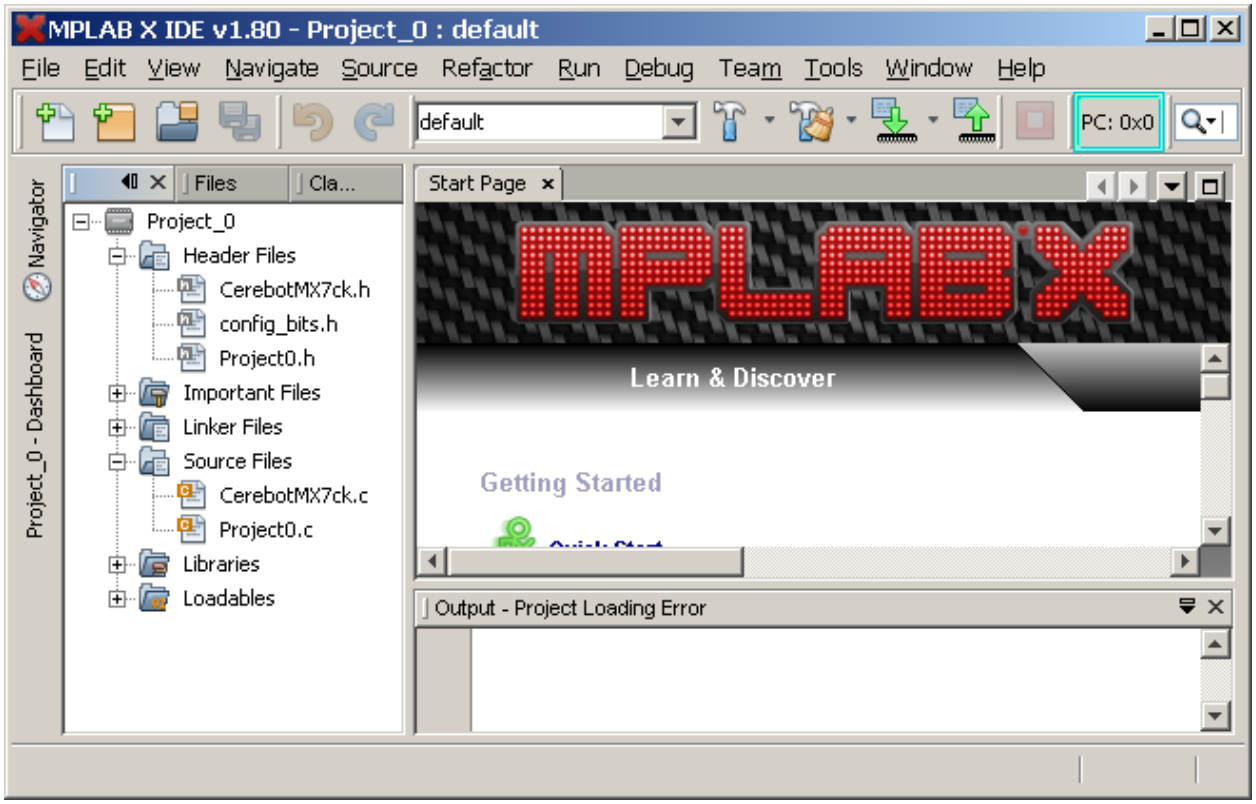

Figure 14. MPLAB X IDE window after Project 0 has been created and project files added

Software Generation - Program Synthesis

All of the text files for Project 0 are provided on the web. After completing the steps to create a project, you can either create additional source and header files or add existing files by clicking on either the source files folder icon or the headers files folder icon in the Project window as shown in Figure 15. This process is also described in the *Minute Video* discussed in paragraph C of Launching a New Project. Click on *Add existing item…* or *Add existing item from Folders…* to add additional existing files to the project. The process is the same for adding and creating header files. It is possible to include precompiled files in projects. However, the use of files like linker files and library files are beyond the scope of this project.

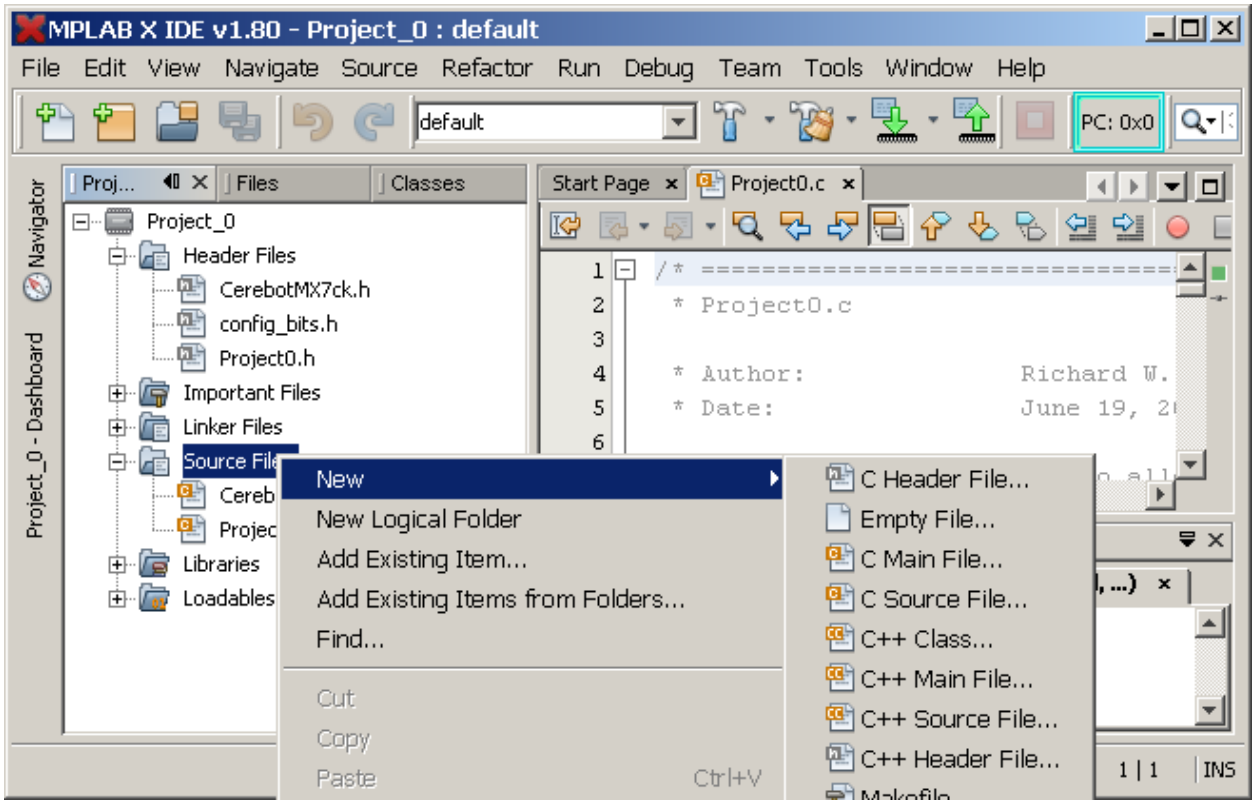

Figure 15. Creating a new source file

The C source file named Project0.c contains the function “main”. The execution of the user generated code starts here. The *main* function, such as the one in Project0.c, contains three common elements: program statements that are executed only once such as the initialization routines, an infinite loop that conditionally repeats the execution of statements that implements the application, and a *return* statement that should never be executed but is there to satisfy the fact that the main function has a non-void return data type.

Software Testing - Program Analysis

There are two major debugging resources provided by the MPLABX IDE: observing code execution and observing the values of variables. Code execution is observed by using two execution controls: breakpoints and single stepping. Break points allow the program to run to a specific line of code and halting execution before executing the line of code where the breakpoint is placed. Single-stepping executes the current line of code only. The controls along the top of the screen shown in Figure 17 control the flow of program execution. After generating the needed source files, the user clicks on the *Debug* button. The code in the project is processed through the compile, link and load operations and the program begins execution. The program runs until the execution encounters a breakpoint or the *Program Pause* control is selected. The processor resumes running if the *Run Program* control is selected.

By default, MPLAB X begins execution as soon as the program is loaded into the processor. During program development, it is useful to halt the program execution on the first executable line of code in the *main* function. To change the default setting, select: Tools -> Options -> Embedded -> Generic Settings. Change the selection for ***Debug startup*** from ***Run*** to ***Halt at Main*** as shown in the figure below.

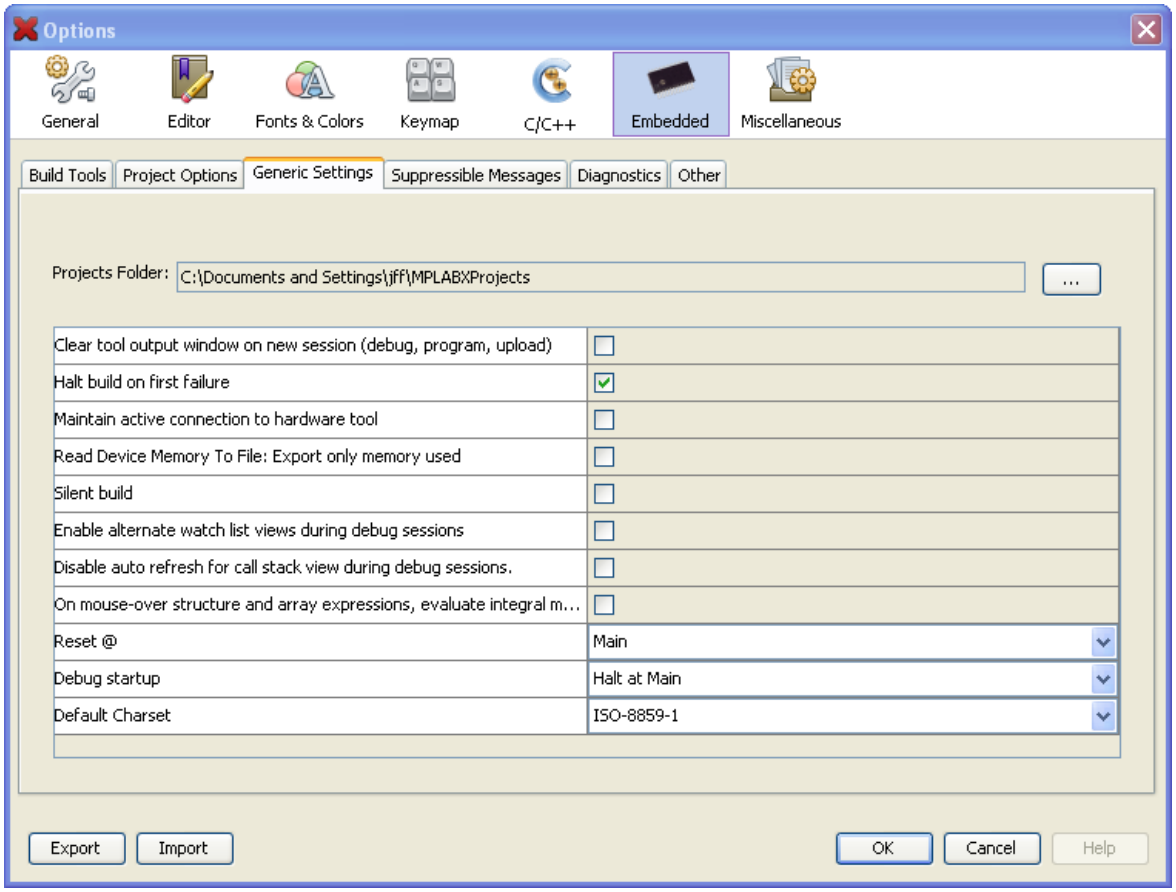

Figure 16. Setting debug run option

Breakpoints are enabled by clicking of the line number beside the source code where you want the program to halt. The two enabled breakpoints shown in Figure 17 are at lines with the pink box that replaces the line number. If the line of source code is shaded green, the processor has halted *before* executing that line of code. The maximum number of breakpoints available on the Cerebot MX7cK is six.

Figure 17 identifies two types of single stepping: step-over and step-into. The difference in single-step operation is observed only if the next line of code to be executed is a call to a function. Step-over executes all of the lines of code in the function call instruction as if is a single instruction. The step-into halts code execution at the first line of code in the function that is called.

There are two types of windows used for observing variable data: Watch windows and Variables windows. Those not familiar MPLAB X are directed to the Microchip web based tutorials that cover the Watch and Variable windows. Data displayed in the Watch window are for global variables and variables declared as static local variables. Watch window variables must be individually added to the window.

Data displayed in the Variables window are function local variables and automatically displayed when executing a particular function. Figure 17 shows the screen when the Variables window is selected for display. Take note of the three control icons on the left. The user should place the mouse pointer over each of those icons to have a description of control functionality displayed. Clicking of the middle icon (the blue diamond gem with an arrow) toggles the display between

local variables only to local variables and global variables as shown in Figure 17. Note that values shown for static local variables declared outside the function where program execution is halted are marked as “Out of Scope”. These are variables that are declared as static in functions other than the function where execution is halted.

Complete descriptions of the Watch window and the Variables window is available using MPLABX Help.

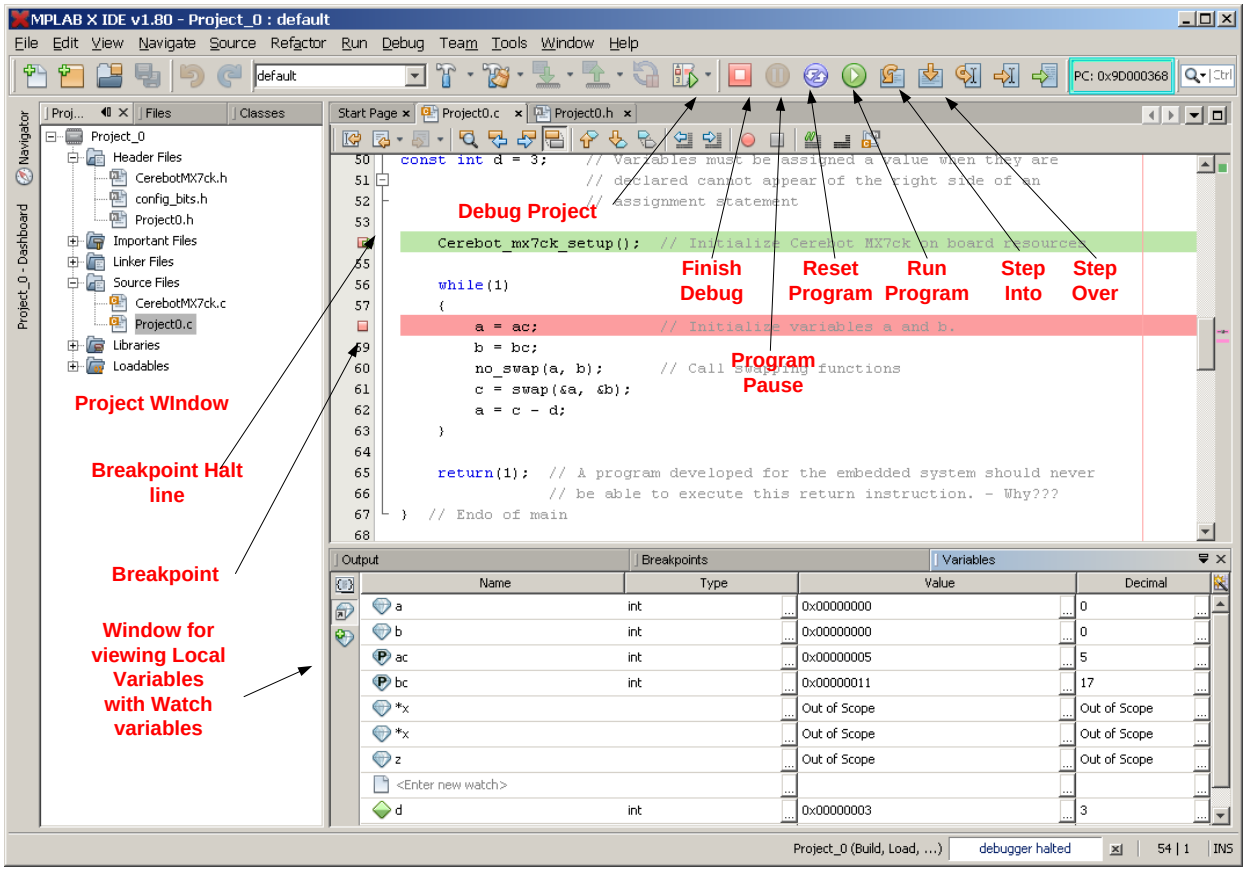

Figure 17. MPLAB X debugging controls and displays

Another method of observing the value assigned to a variable is by placing the cursor on the variable. This only works if working in the debug mode and the processor is halted.