MPLAB® X IDE User's Guide

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

Printed on recycled paper.

ISBN: 978-1-62077-799-2

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Table of Contents

Preface

INTRODUCTION

This chapter contains general information that will be useful to know before using MPLAB® X IDE. Items discussed include:

- Document Layout
- Conventions Used
- Recommended Reading

DOCUMENT LAYOUT

This document describes how to use the MPLAB X IDE. The layout of the manual is as follows:

- **Chapter 1. "What is MPLAB X IDE?"** – an overview of what the MPLAB X IDE is and where to find help.
- **Chapter 2. "Before You Begin"** – describes how to install USB drivers for the hardware tools and language toolsuites for compiling/assembling code.
- **Chapter 3. "Tutorial"** – provides step-by-step descriptions of features for using MPLAB X IDE.
- **Chapter 4. "Basic Tasks"** – describes how to use the basic features of MPLAB X IDE. It is similar to the Tutorial chapter but with more detail.
- **Chapter 5. "Additional Tasks"** – describes how to use additional features of MPLAB X IDE, e.g., importing MPLAB IDE v8 projects or using the stopwatch.
- **Chapter 6. "Advanced Tasks"** – describes how to use the advanced features of MPLAB X IDE, e.g., working with multiple projects and project configurations.
- **Chapter 9. "Troubleshooting"** – discusses troubleshooting techniques.
- **Chapter 10. "MPLAB X IDE vs. MPLAB IDE v8"** – explains the major feature, menu, and tool support differences between MPLAB X IDE and MPLAB IDE v8.
- **Chapter 11. "Desktop Reference"** – provides a reference to MPLAB X IDE desktop items, including menus, toolbars, and the status bar.
- **Chapter 12. "MPLAB X IDE Windows and Dialogs"** – references NetBeans™ windows and dialogs and discusses the windows and dialogs that are unique to MPLAB X IDE.
- **Chapter 8. "Project Files and Folders"** – explains the folder structure and locations of project files.
- **Appendix A. "Configuration Settings Summary"** – shows how to set Configuration bits in code for supported language tools. This is required in MPLAB X IDE as the Configurations Settings window only temporarily sets the bits for debug.

CONVENTIONS USED

The following conventions may appear in this documentation:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic	Referenced books	*MPLAB® IDE User's Guide*
	Emphasized text	...is the *only* compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog, or an individual menu item	"Save project before build"
Underlined, italic with right angle bracket between text	A path in text	*File>Save*
Right angle bracket between text	A path in a table cell	File>Save
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain	Sample source code	`#define START`
	Filenames	`autoexec.bat`
	File paths	`c:\mcc18\h`
	Keywords	`_asm, _endasm, static`
	Command-line options	`-Opa+, -Opa-`
	Bit values	`0, 1`
	Constants	`0xFF,'A'`
Italic	A variable argument	*`file`*`.o`, where *`file`* can be any valid filename
Square brackets [ ]	Optional arguments	`mpasmwin [options] file [options]`
Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments; an OR selection	`errorlevel {0|1}`
Ellipses...	Replaces repeated text	`var_name [, var_name...]`
	Represents code supplied by user	`void main (void) { ... }`

RECOMMENDED READING

This user's guide describes how to use MPLAB X IDE. Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

Readme for MPLAB IDE

For the latest information on using MPLAB X IDE, read the release notes under the "Release Notes and Support Documentation" heading on the Start page. The release notes contain update information and known issues that may not be included in this user's guide.

Readme Files

For the latest information on using other tools, read the tool-specific Readme files in the Readmes subdirectory of the MPLAB X IDE installation directory. The Readme files contain update information and known issues that may not be included in this user's guide.

Online Help Files

Comprehensive help files are available for MPLAB X IDE, MPLAB Editor and MPLAB SIM simulator. Tutorials, functional descriptions and reference material are included.

Device Data Sheets and Family Reference Manuals

See the Microchip web site, http://www.microchip.com, for complete and updated versions of PIC® MCU and dsPIC® DSC data sheets and related device family reference manuals.

NOTES:

Chapter 1. What is MPLAB X IDE?

1.1 INTRODUCTION

MPLAB® X IDE is a software program that is used to develop applications for Microchip microcontrollers and digital signal controllers. (Experienced embedded-systems designers may want to skip to the next chapter.)

This development tool is called an Integrated Development Environment, or IDE, because it provides a single integrated "environment" to develop code for embedded microcontrollers.

This chapter describes the development of an embedded system and briefly explains how MPLAB X IDE from Microchip is used in the process.

Topics discussed here include the following:

- An Overview of Embedded Systems
- The Development Cycle
- Project Manager
- Language Tools
- Target Debugging
- Device Programming
- Components of MPLAB X IDE
- MPLAB X IDE Online Help
- Other MPLAB X IDE Documentation
- Web Site
- MPLAB X IDE Updates

1.2 AN OVERVIEW OF EMBEDDED SYSTEMS

An embedded system is typically a design that uses the power of a small microcontroller, like the Microchip PIC® microcontroller (MCU) or dsPIC® digital signal controller (DSC). These microcontrollers combine a microprocessor unit (like the CPU in a personal computer) with some additional circuits called peripherals, plus some additional circuits on the same chip to make a small control module requiring few other external devices. This single device can then be embedded into other electronic and mechanical devices for low-cost digital control.

1.2.1 Differences Between an Embedded Controller and a Personal Computer

The main difference between an embedded controller and a personal computer is that the embedded controller is dedicated to one specific task or set of tasks. A personal computer is designed to run many different types of programs and to connect to many different external devices. An embedded controller has a single program and, as a result, can be made cheaply to include just enough computing power and hardware to perform that dedicated task.

A personal computer has a relatively expensive generalized central processing unit (CPU) at its heart with many other external devices (memory, disk drives, video controllers, network interface circuits, etc.). An embedded system has a low-cost microcontroller unit (MCU) for its intelligence, with many peripheral circuits on the same chip, and with relatively few external devices.

Often, an embedded system is an invisible part, or sub-module of another product, such as a cordless drill, refrigerator or garage door opener. The controller in these products does a tiny portion of the function of the whole device. The controller adds low-cost intelligence to some of the critical sub-systems in these devices.

An example of an embedded system is a smoke detector. Its function is to evaluate signals from a sensor and sound an alarm if the signals indicate the presence of smoke. A small program in the smoke detector either runs in an infinite loop, sampling the signal from the smoke sensor, or lies dormant in a low-power "Sleep" mode, being awakened by a signal from the sensor. The program then sounds the alarm. The program would possibly have a few other functions, such as a user test function, and a low battery alert.

While a personal computer with a sensor and audio output could be programmed to do the same function, it would not be a cost-effective solution (nor would it run on a nine-volt battery, unattended for years!). Embedded designs use inexpensive microcontrollers to put intelligence into the everyday things in our environment, such as smoke detectors, cameras, cell phones, appliances, automobiles, smart cards and security systems.

1.2.2 Components of a Microcontroller

The PIC MCU has on-chip program memory (Figure 1-1) for the firmware, or coded instructions, to run a program (Figure 1-2). A Program Counter (PC) is used to address program memory, including Reset and interrupt addresses. A hardware stack is used with call and return instructions in code, so it works with, but is not part of, program memory. Device data sheets describe the details of program memory operation, vectors and the stack.

FIGURE 1-1: PIC® MCU DATA SHEET – PROGRAM MEMORY AND STACK

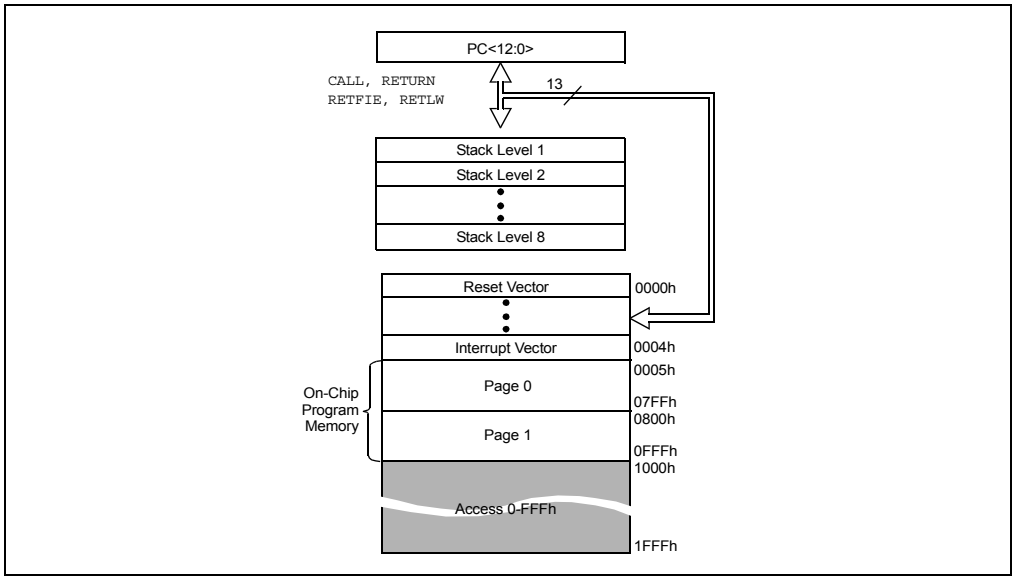
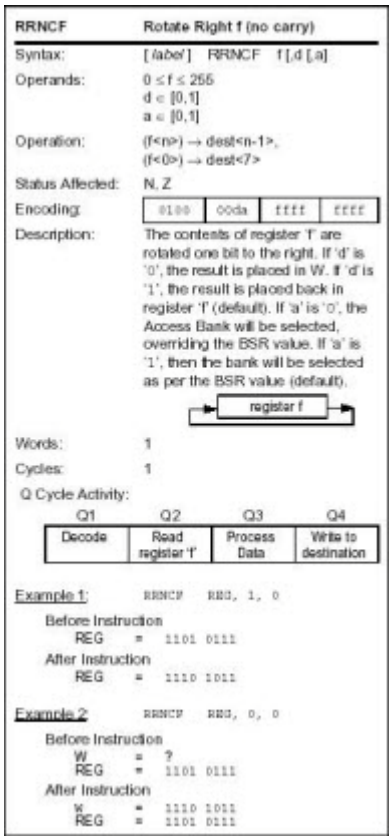


FIGURE 1-2: PIC® MCU DATA SHEET – INSTRUCTIONS (EXCERPT)



The microcontroller also has data or “file register” memory. This memory consists of Special Function Registers (SFRs) and General Purpose Registers (GPRs) as shown in Figure 1-4. SFRs are registers used by the CPU and peripheral functions for controlling the desired operation of the device. GPRs are for storage of variables that the program will need for computation or temporary storage. Some microcontrollers have additional data EEPROM memory. As with program memory, device data sheets describe the details of data memory use and operation.

FIGURE 1-3: PIC® MCU DATA SHEET – FILE REGISTERS

	File Address		File Address		File Address		File Address
Indirect addr. (1)	00h	Indirect addr. (1)	80h	Indirect addr. (1)	100h	Indirect addr. (1)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	PORTA	105h	TRISA	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h	PORTC	107h	TRISC	187h
	08h		88h		108h		188h
	09h		89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDAT	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2(1)	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh		18Eh
TMR1H	0Fh	OSCCON	8Fh	EEADRH	10Fh		18Fh
T1CON	10h	OSCTUNE	90h		110h		190h
TMR2	11h		91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD(2)	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h	WPUA	95h	WPUB	115h		195h
CCPR1H	16h	IOCA	96h	IOCB	116h		196h
CCP1CON	17h	WDTCON	97h		117h		197h
RCSTA	18h	TXSTA	98h	VRCON	118h		198h
TXREG	19h	SPBRG	99h	CM1CON0	119h		199h
RCREG	1Ah	SPBRGH	9Ah	CM2CON0	11Ah		19Ah
	1Bh	BAUDCTL	9Bh	CM2CON1	11Bh		19Bh
PWM1CON	1Ch		9Ch		11Ch		19Ch
ECCPAS	1Dh		9Dh		11Dh	PSTRCON	19Dh
ADRESH	1Eh	ADRESL	9Eh	ANSEL	11Eh	SRCON	19Eh
ADCON0	1Fh	ADCON1	9Fh	ANSELH	11Fh		19Fh
General Purpose Register 96 Bytes	20h – 7Fh	General Purpose Register 80 Bytes	A0h – EFh	General Purpose Register 80 Bytes	120h – 16Fh		1A0h
		accesses 70h-7Fh	F0h – FFh	accesses 70h-7Fh	170h – 17Fh	accesses 70h-7Fh	1F0h – 1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

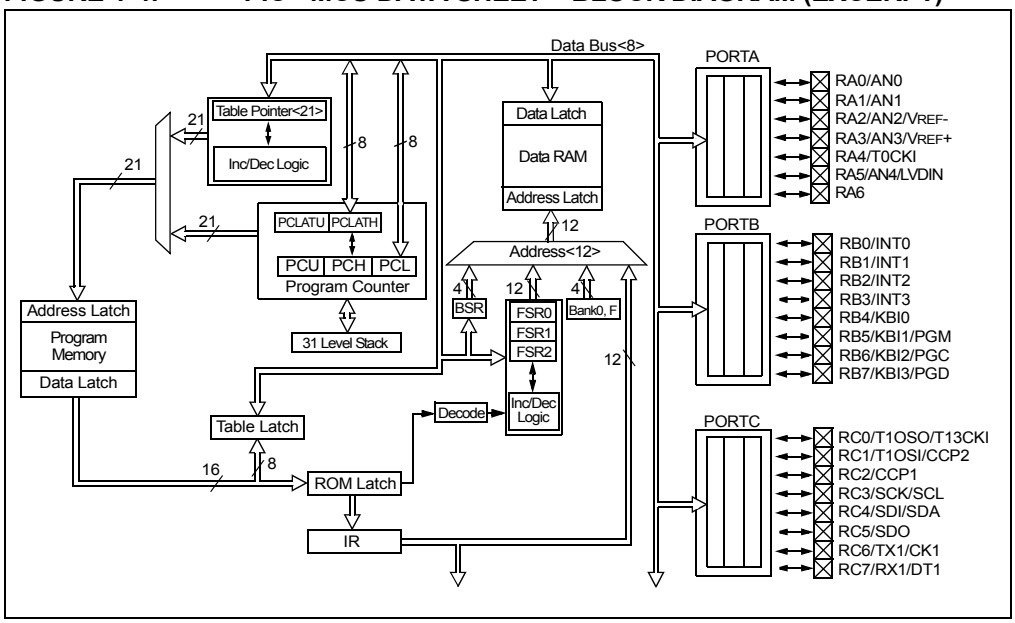
Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

Note 2: Address 93h also accesses the SSP Mask (SSPMSK) register under certain conditions.

In addition to memory, the microcontroller has a number of peripheral device circuits on the same chip (Figure 1-4). Some peripheral devices are called input/output (I/O) ports. I/O ports are pins on the microcontroller that can be used as outputs and driven high or low to send signals, blink lights, drive speakers – just about anything that can be sent through a wire. Often these pins are bidirectional and can also be configured as inputs, allowing the program to respond to an external switch, sensor, or to communicate with some external device.

FIGURE 1-4: PIC® MCU DATA SHEET – BLOCK DIAGRAM (EXCERPT)



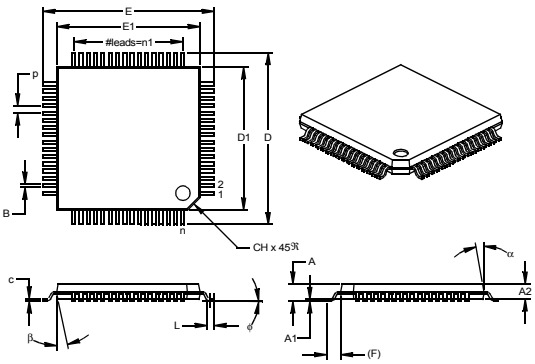
To design such a system, choose which peripherals are necessary for the application.

The following is a list of common peripherals:

- Analog-to-Digital Converters (ADCs) allow microcontrollers to connect to sensors and receive changing voltage levels.
- Serial communication peripherals that allow streaming communications over a few wires to another microcontroller, to a local network, or to the Internet.
- Peripherals on the PIC MCU called “timers” accurately measure signal events and generate and capture communications signals, produce precise waveforms, even automatically reset the microcontroller if it gets “hung” or lost due to a power glitch or hardware malfunction.
- Other peripherals detect when the external power is dipping below dangerous levels, so that the microcontroller can store critical information and safely shut down before power is completely lost.

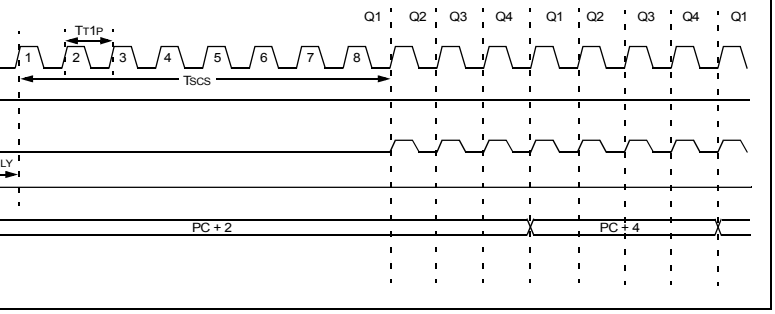
The peripherals, and the amount of memory an application needs to run a program, largely determine which PIC MCU to use. Other factors could include the power consumed by the microcontroller and its “form factor,” i.e., the size and characteristics of the physical package that must reside on the target design (Figure 1-5).

FIGURE 1-5: EXAMPLE PIC® MCU DEVICE PACKAGE



A microcontroller becomes active when power is applied and an oscillator begins generating a clocking signal (Figure 1-6). Depending on the microcontroller, there may be several internal and external oscillator operational modes.

FIGURE 1-6: PIC® MCU DATA SHEET – TIMING (EXCERPT)



1.2.3 Implementing an Embedded System Design with MPLAB X IDE

A development system for embedded controllers is a system of programs running on a computer to help write, edit, debug and program code – the intelligence of embedded systems applications – into a microcontroller. MPLAB X IDE is such a system; it contains all the components needed to design and deploy embedded systems applications.

The typical tasks for developing an embedded controller application are:

1. Create the high level design. From the features and performance desired, decide which PIC MCU or dsPIC DSC device is best suited to the application, then design the associated hardware circuitry. After determining which peripherals and pins control the hardware, write the firmware – the software that will control the hardware aspects of the embedded application. A language tool such as an assembler, which is directly translatable into machine code, or a compiler that allows a more natural language for creating programs, should be used to write and edit code. Assemblers and compilers help make the code understandable, allowing function labels to identify code routines with variables that have names associated with their use, and with constructs that help organize the code in a maintainable structure.
2. Compile, assemble and link the software using the assembler and/or compiler and linker to convert your code into “ones and zeros” – machine code for the PIC MCUs. This machine code will eventually become the firmware (the code programmed into the microcontroller).
3. Test your code. Usually a complex program does not work exactly the way imagined, and “bugs” need to be removed from the design to get proper results. The debugger allows you to see the “ones and zeros” execute, related to the source code you wrote, with the symbols and function names from your program. Debugging allows you to experiment with your code to see the value of variables at various points in the program, and to do “what if” checks, changing variable values and stepping through routines.
4. “Burn” the code into a microcontroller and verify that it executes correctly in the finished application.

Of course, each of these steps can be quite complex. The important thing is to concentrate on the details of your own design, while relying upon MPLAB X IDE and its components to get through each step without continuously encountering new learning curves.

Step 1 is driven by the designer, although MPLAB X IDE can help in modeling circuits and code so that crucial design decisions can be made.

MPLAB X IDE really helps with steps 2 through 4. Its Programmer's Editor helps write correct code with the language tools of choice. The editor is aware of the assembler and compiler programming constructs and automatically "color-keys" the source code to help ensure it is syntactically correct. The Project Manager enables you to organize the various files used in your application: source files, processor description header files and library files. When the code is built, you can control how rigorously code will be optimized for size or speed by the compiler and where individual variables and program data will be programmed into the device. You can also specify a "memory model" in order to make the best use of the microcontroller's memory for your application. If the language tools run into errors when building the application, the offending line is shown and can be double clicked to go to the corresponding source file for immediate editing. After editing, you will rebuild and try your application again. Often this write-compile-fix loop is done many times for complex code as the sub-sections are written and tested. MPLAB X IDE goes through this loop with maximum speed, allowing you to get on to the next step.

When the code builds with no errors, it needs to be tested. MPLAB X IDE has components called "debuggers" and free software simulators for all PIC MCU and dsPIC DSC devices to help test the code. Even if the hardware is not yet finished, you can begin testing the code with the simulator, a software program that simulates the execution of the microcontroller. The simulator can accept a simulated input (stimulus), in order to model how the firmware responds to external signals. The simulator can measure code execution time, single step through code to watch variables and peripherals, and trace the code to generate a detailed record of how the program ran.

When the hardware is in a prototype stage, a hardware debugger, such as an in-circuit emulator or an in-circuit debugger, can be used. These debug tools run the code in real time on your actual application by using special circuitry built into many devices with Flash program memory. They can "see into" the target microcontroller's program and data memory, and stop and start program execution, allowing you to test the code with the microcontroller in place on the application.

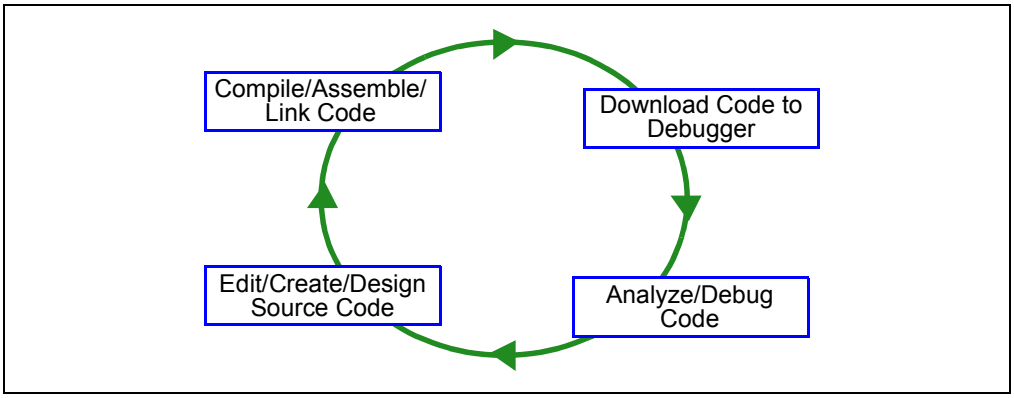
After the application is running correctly, you can program a microcontroller with one of Microchip's devices or development programmers. These programmers verify that the finished code will run as designed. MPLAB X IDE supports most PIC MCUs and all dsPIC DSCs.

1.3 THE DEVELOPMENT CYCLE

The process for writing an application is often described as a development cycle, since it is rare that all the steps from design to implementation can be done flawlessly the first time. More often code is written, tested and then modified in order to produce an application that performs correctly.

The Integrated Development Environment allows the embedded systems design engineer to progress through this cycle without the distraction of switching among an array of tools. By using MPLAB X IDE, all the functions are integrated, allowing the engineer to concentrate on completing the application without the interruption of separate tools and different modes of operation.

FIGURE 1-7: THE DESIGN CYCLE



MPLAB X IDE is a “wrapper” that coordinates all the tools from a single graphical user interface, usually automatically. For instance, once code is written, it can be converted to executable instructions and downloaded into a microcontroller to see how it works. In this process multiple tools are needed: an editor to write the code, a project manager to organize files and settings, a compiler or assembler to convert the source code to machine code and some sort of hardware or software that either connects to a target microcontroller or simulates the operation of a microcontroller.

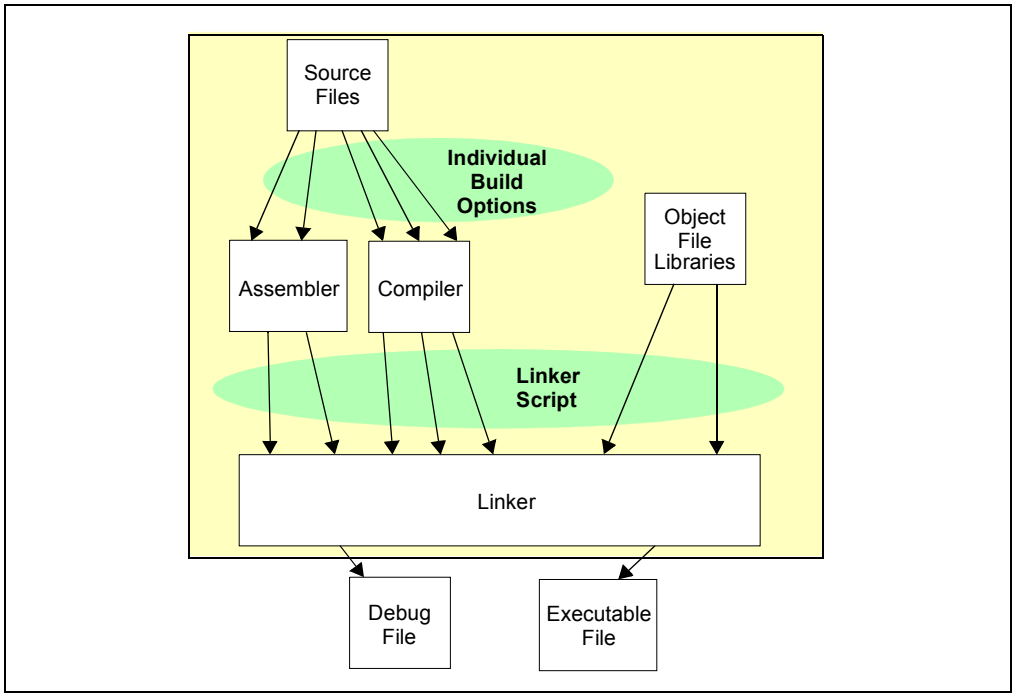
1.4 PROJECT MANAGER

The project manager organizes the files to be edited and other associated files so they can be sent to the language tools for assembly or compilation, and ultimately to a linker.

The linker has the task of placing the object code fragments from the assembler, compiler and libraries into the proper memory areas of the embedded controller, and ensure that the modules function with each other (or are "linked").

This entire operation from assembly and compilation through the link process is called a project "build". Properties specified for the language tools can be invoked differently for each file, if desired, and a build process integrates all of the language tools' operations.

FIGURE 1-8: MPLAB® X IDE PROJECT MANAGER



The source files are text files that are written conforming to the rules of the assembler or compiler. The assembler and compiler convert them into intermediate modules of machine code and placeholders for references to functions and data storage.

The linker resolves these placeholders and combines all the modules into a file of executable machine code. The linker also produces a debug file which allows MPLAB X IDE to relate the executing machine codes back to the source files.

A text editor is used to write the code. It recognizes the constructs in the text and uses color coding to identify various elements, such as instruction mnemonics, C language constructs and comments. The editor supports operations commonly used in writing source code. After the code is written, the editor works with the other tools to display code execution in the debugger. Breakpoints (which stop or "break" the execution of code) can be set in the editor, and the values of variables can be inspected by hovering the mouse pointer over the variable name. Names of variables can be dragged from source text windows and then dropped into a Watch window where their changing values can be watched after each breakpoint or during code execution.

1.5 LANGUAGE TOOLS

Language tools are programs such as cross-assemblers and cross-compilers. Most people are familiar with some of the language tools that run on a computer, e.g., Visual Basic or C compilers.

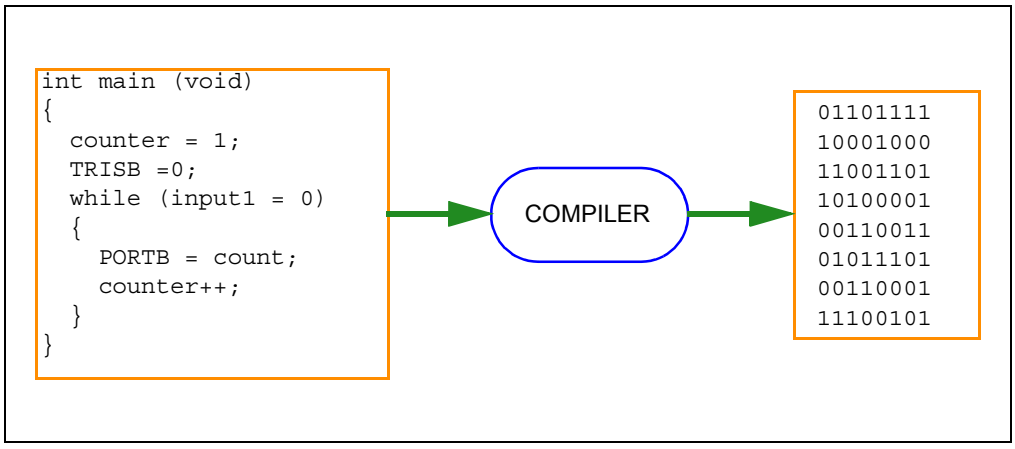
When using language tools for embedded systems, a “cross-assembler” or “cross-compiler” is used. These tools differ from typical compilers in that they run on a computer, but they produce code to run on another microprocessor (or microcontroller).

Language tools also produce a debug file that MPLAB X IDE uses to correlate the machine instructions and memory locations with the source code. This bit of integration allows the MPLAB X IDE editor to set breakpoints, allows Watches windows to view variable contents, and lets you single step through the source code, while watching the application execute.

Embedded system language tools also differ somewhat from compilers that run and execute on a computer because they must be very space conscious. The smaller the code produced, the better, because that provides the smallest possible memory usage for the target, which reduces cost. This means that techniques to optimize and enhance the code, using machine-specific knowledge, are desirable.

The size of programs for computers typically extends into the megabytes for moderately complex programs. The size of simple embedded systems programs may be as small as a thousand bytes or less. A medium size embedded system might need 32K or 64K of code for relatively complex functions. Some embedded systems use megabytes of storage for large tables, user text messages or data logging.

FIGURE 1-9: A COMPILER CONVERTS SOURCE CODE INTO MACHINE INSTRUCTIONS



1.6 TARGET DEBUGGING

In a development environment, the execution of the code is tested on a debugger. The debugger can be a software program that simulates the operation of the microcontroller for testing, or it can be a hardware instrument to analyze the program as it executes in the application.

1.6.1 Software Debuggers

Simulators are built into MPLAB X IDE so a program can be tested without any additional hardware. A simulator is a software debugger, and the debugger functions for the simulator are almost identical to the hardware debuggers, allowing a new tool to be learned with ease. Usually, a simulator runs somewhat slower than an actual microcontroller, since the CPU in the computer is being used to simulate the operations of the microcontroller.

1.6.2 Hardware Debuggers

There are two types of hardware that can be used with MPLAB X IDE: programmers and hardware debuggers. A programmer simply burns the machine code from the PC into the internal memory of the target microcontroller. The microcontroller can then be plugged into the application and, hopefully, it will run as designed.

Usually, however, the code does not function exactly as anticipated, and the engineer is tasked with reviewing the code and its operation in the application to determine how to modify the original source code to make it execute as desired. This process is called debugging. As noted previously, the simulator can be used to test how the code will operate, but once a microcontroller is programmed with the firmware, many things outside the scope of the simulator come into play. Using just a programmer, the code could be changed, reprogrammed into the microcontroller and plugged into the target for retest, but this could be a long, laborious cycle if the code is complex, and it is difficult to understand exactly what is going wrong in the hardware.

This is where a hardware debugger is useful. Hardware debuggers can be in-circuit emulators or in-circuit debuggers, which use microcontrollers that have special built-in debugging features. A hardware debugger, like a simulator, allows the engineer to inspect variables at various points in the code, and single step to follow instructions as the hardware interacts with its specialized circuitry.

1.6.3 Integrated Development Environment

Debugging usually becomes urgent near the end of the project design cycle. As deadlines loom, getting the application to function as originally designed is the last step before going into deployment of the product, and often has the most influence on producing delays in getting a product out. That's where an integrated development environment is most important. Doing fine "tweaks" to the code, recompiling, downloading and testing all require time. Using all tools within a single environment will reduce the time around the "cycle." These last steps, where critical bugs are worked out, are a test for the embedded systems designer. The right tool can save time. With MPLAB X IDE many tools can be selected, but they all will have a similar interface, and the learning curve from simulator to low-cost in-circuit debugger to powerful in-circuit emulator is small.

1.7 DEVICE PROGRAMMING

After the application has been debugged and is running in the development environment, it needs to be tested on its own. A device can be programmed with an in-circuit emulator, an in-circuit debugger, a development programmer, or a device programmer. MPLAB X IDE can be set to the programmer function, and the part can be “burned”. The target application can now be observed in its nearly final state. Engineering prototype programmers allow quick prototypes to be made and evaluated. Some applications can be programmed after the device is soldered on the target PC board. Using In-Circuit Serial Programming™ (ICSP™) programming capability, the firmware can be programmed into the application at the time of manufacture, allowing updated revisions to be programmed into an embedded application later in its life cycle. Devices that support in-circuit debugging can even be plugged back into an in-circuit debugger after manufacturing for quality tests and development of next generation firmware.

Production programming can be accomplished using a production programmer and the MPLAB IPE, which is installed with MPLAB X IDE.

1.8 COMPONENTS OF MPLAB X IDE

MPLAB X IDE includes:

- a full-featured programmer's text editor that also serves as a window into the debugger.
- a project manager (visible as the Projects window) that provides integration and communication between the IDE and the language tools.
- a number of assembler/linker suites for the development of firmware for your project's device.
- a debugger engine that provides breakpoints, single stepping, Watch windows and all the features of a modern debugger. The debugger works in conjunction with debug tools, both software and hardware.
- a software simulator for all PIC MCU and dsPIC DSC devices. The simulator is actually composed of several device-specific simulator executables. MPLAB X IDE decides which one to use based on your project's device.

Optional components can be acquired or purchased to work with the MPLAB X IDE:

- **Compiler Language Tools**

MPLAB XC C compilers from Microchip provide fully integrated, optimized code for PIC MCUs and dsPIC DSCs. Along with compilers from microEngineering Labs, CCS and SDCC, they are invoked by the MPLAB X IDE project manager to compile code that is automatically loaded into the target debugger for instant testing and verification.

- **Programmers**

MPLAB ICD 3 in-circuit debugger, MPLAB REAL ICE™ in-circuit emulator, and MPLAB PM3 programmer are capable of the production programming of code into target devices. PICkit™ 3 in-circuit debugger is capable of the development programming of code into target devices.

All of these tools may be used with MPLAB X IDE to control programming of both code and data, as well as the Configuration bits to set the various operating modes of the target microcontrollers or digital signal controllers.

In addition, all of these tools may be used with MPLAB IPE to program code, data and configuration bits. MPLAB IPE is designed more for production programming, and its interface is simplified to do just that.

- **In-Circuit Debuggers and Emulators**

PICkit 3 and MPLAB ICD 3 in-circuit debuggers, and MPLAB REAL ICE in-circuit emulator can be used to debug application code on target devices. By using some of the on-chip resources, these can download code into a target microcontroller inserted in the application, set breakpoints, single step and monitor registers and variables. The emulator includes additional debug features, such as trace.

- **Plug-In Tools**

Several plug-ins are available to add to the capabilities for MPLAB X IDE. For example, the Data Monitor and Control Interface (DMCI) provides a mechanism to view and control variables in code and change their values real-time. It also allows you to view output data in a graphical format.

For more on the plug-in tools supported, see **Section 5.21 "Add Plug-In Tools"**.

1.9 MPLAB X IDE ONLINE HELP

MPLAB X IDE is built upon the NetBeans platform. Therefore, many of the NetBeans functions are now MPLAB X IDE functions.

Please refer to all help files for a complete understanding of MPLAB X IDE behavior. To launch help, select *Help>Help Contents*. This merges all help files into one, so it may take a little more time to open.

For NetBeans information, see the online help files under “NetBeans Help” in the table of contents. For all MPLAB X IDE development tool information, see the online help files under “MPLAB X IDE Help” in the table of contents.

You can also view individual help files for selected tools under *Help>Tool Help Contents*. Launching an individual help file will be faster and provide a smaller search of topics.

For a comparison of MPLAB X IDE and MPLAB IDE v8, see **Chapter 10. “MPLAB X IDE vs. MPLAB IDE v8”**.

1.10 OTHER MPLAB X IDE DOCUMENTATION

In addition to help, links to other documentation, videos, forums, and wikis are featured on the Start Page (Figure 1-10).

The MPLAB X IDE Wiki is a good place to look for tips on advance features (Figure 1-11).

FIGURE 1-10: MPLAB® X IDE START PAGE

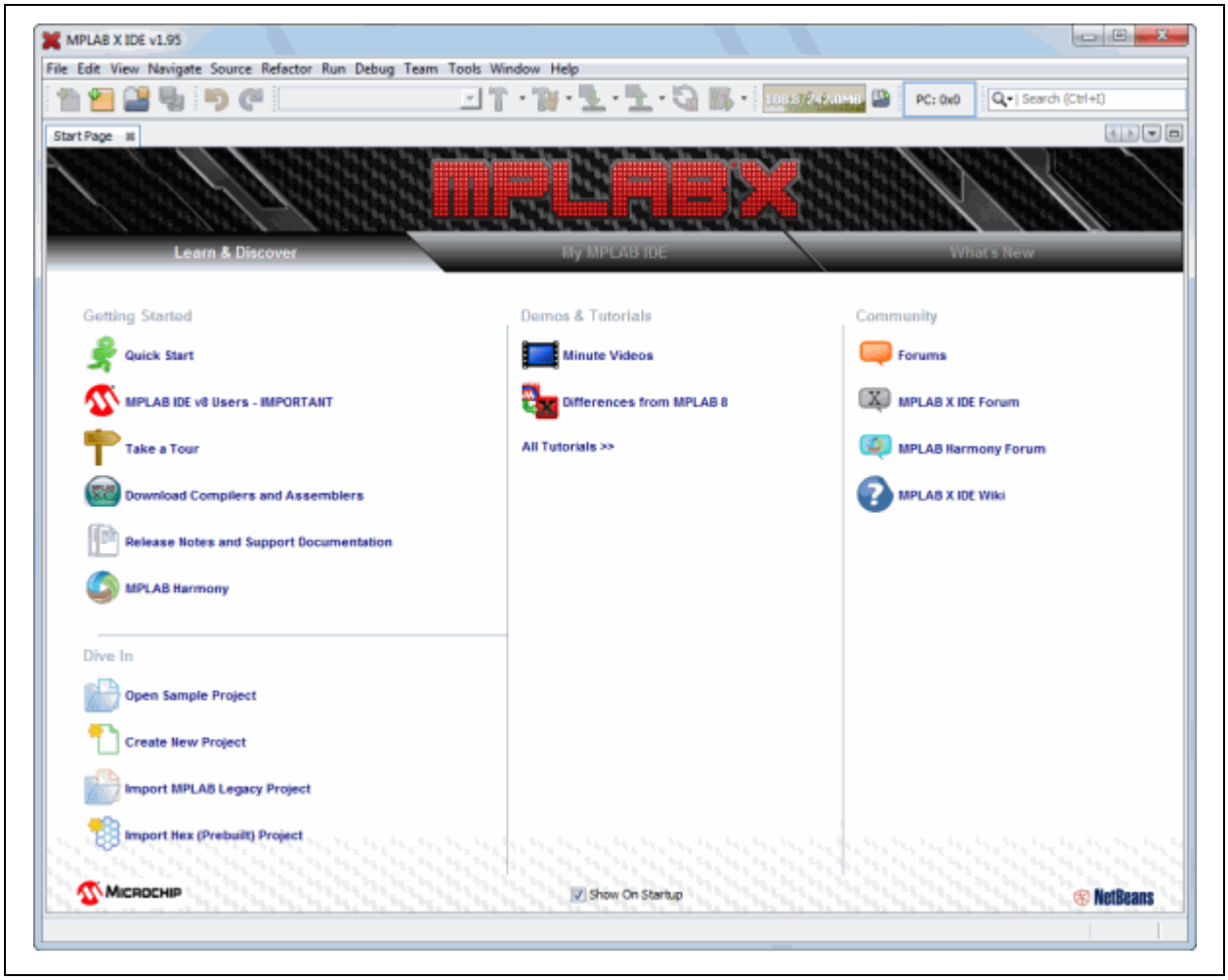


FIGURE 1-11: MPLAB® X IDE WIKI

1.11 WEB SITE

Microchip provides online support via our web site at:

http://www.microchip.com/devtools

This web site make files and information easily available to customers. For more details, see **Support**.

1.12 MPLAB X IDE UPDATES

MPLAB X IDE is an evolving program with thousands of users. Microchip Technology is continually designing new microcontrollers with new features. Many new MPLAB X IDE features come from customer requests and from internal usage. Continued new designs and the release of new microcontrollers ensure that MPLAB X IDE will continue to evolve.

MPLAB X IDE is scheduled for a version update approximately every few months to add new device support and new features.

For projects that are midway through development when a new version of MPLAB X IDE is released, it is considered "best practice" to not update to the new release unless there is a compelling reason to do so, such as a bug fix on a bug that inhibits the current efforts. The start of a new project is the best time to update to a new release.

Each new release of the MPLAB X IDE software has new features implemented, so the printed documentation will inevitably "lag" the online help. The online help is the best source for any questions about MPLAB X IDE.

To be notified of updates to MPLAB X IDE and its components, subscribe to the Development Tools section of myMICROCHIP Personalized Notification Service on:

http://www.microchip.com/pcn

For more details, see **Support**.

Chapter 2. Before You Begin

2.1 INTRODUCTION

Before you can use MPLAB X IDE you must do the following:

- Install JRE and MPLAB X IDE
- Install the USB Device Drivers (For Hardware Tools)
- Connect to a Target (For Hardware Tools)
- Install the Language Tools
- Launch the IDE
- Launch Multiple Instances of the IDE

2.2 INSTALL JRE AND MPLAB X IDE

When you install MPLAB X IDE (based on the NetBeans platform), the correct Java Runtime Environment (JRE) for Windows operating system (OS) or Linux OS will be installed. For Mac OS, if the correct JRE is already installed, the MPLAB X IDE install will proceed. If not, you will be prompted by a dialog to acquire the correct version. Follow the instructions, install the JRE, and then install MPLAB X IDE.

2.3 INSTALL THE USB DEVICE DRIVERS (FOR HARDWARE TOOLS)

For correct tool operation, you might need to install USB drivers.

2.3.1 USB Driver Installation for Mac or Linux Operating Systems

When you install MPLAB X IDE on a Mac or Linux box, the installer will place the USB drivers for you. You do not need to do anything.

2.3.2 USB Driver Installation for Windows® XP/7/8 Operating Systems

If you install MPLAB X IDE on a personal computer uses Windows OS, follow the instructions below to correctly install the USB drivers. (The USB hardware tool drivers for MPLAB IDE v8.xx are not the same as those for MPLAB X IDE.)

These instructions apply to the following tools:

- MPLAB REAL ICE in-circuit emulator
- MPLAB ICD 3 in-circuit debugger
- MPLAB PM3 device programmer
- PIC32 Starter Kit

You do not need to do anything for PICkit 2, PICkit 3 or other MPLAB Starter Kits.

Follow the instructions below to determine your installation method.

2.3.2.1 BEFORE YOU INSTALL THE DRIVERS

Whether you use the Switcher utility or activate the preinstaller to install your drivers (both discussed in following sections), be aware that your system's version of WinUSB drivers will be replaced if they are older than the Switcher or preinstaller version. If you want to keep your version of WinUSB drivers, rename these files before installing any Microchip device driver.

The WinUSB driver files are located at the following locations:

32-bit OS

`C:\Windows\system32\WinUSB.dll` (32-bit)

`C:\Windows\system32\drivers\WinUSB.sys` (64-bit)

64-bit OS

`C:\Windows\SysWOW64\WinUSB.dll` (32-bit)

`C:\Windows\system32\WinUSB.dll` (64-bit)

`C:\Windows\system32\drivers\WinUSB.sys` (64-bit)

2.3.2.2 IF YOU HAVE WINDOWS XP 64, MANUALLY SWITCH

If using the Windows XP 64-bit OS, you will have to switch the device drivers manually. See **Section 2.3.2.6 "If You Need to Manually Install the Drivers"**.

2.3.2.3 IF YOU HAVE WINDOWS 7 OR 8, USE ADMINISTRATOR MODE

If you will use the Switcher executable to install your device drivers, you must be in Administrator mode to run this program on Windows 7 or 8.

To run the Device Driver Switcher GUI application as administrator, right click on the executable – `MPDDSwitch.exe` or `MPDDSwitch64.exe` – and select 'Run as Administrator'.

It is recommended that you use the GUI application first to switch the drivers. If this is problematic, you may switch the drivers using command-line applications.

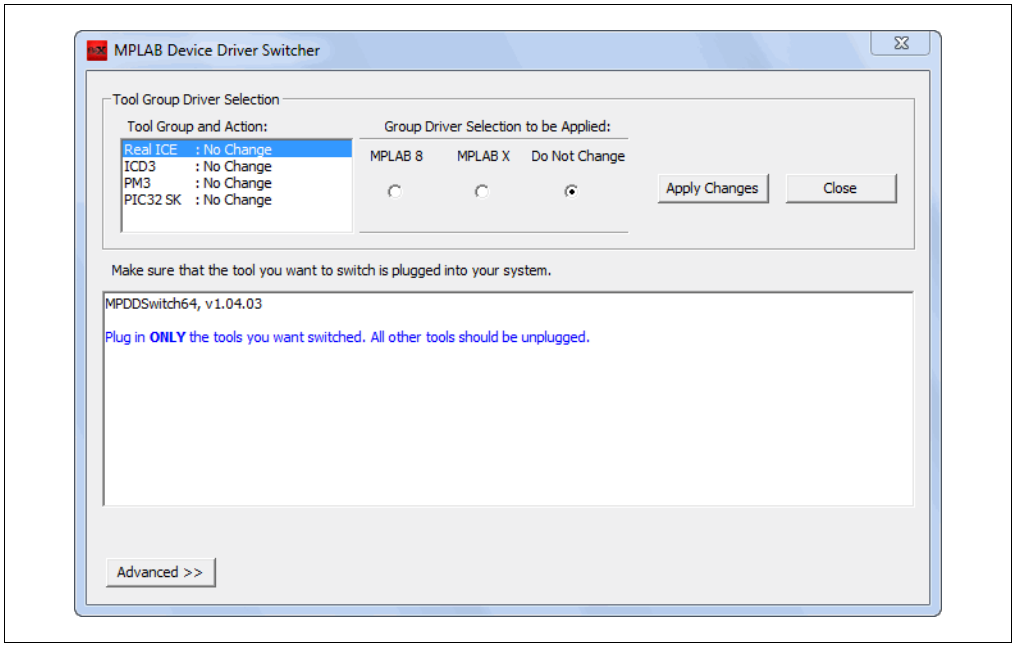
To run the command-line application – `mchpdds32.exe` or `mchpdds64.exe` – as administrator, first open the command prompt in Admin mode: *Start>All Programs>Accessories>Command Prompt*, right click and select 'Run as Administrator'. This will open up the Administrator: Command Prompt. After this, the instructions provided in the `ReadMe32.txt` or `ReadMe64.txt` file may be followed to complete the driver switching.

2.3.2.4 IF MPLAB IDE V8.XX IS ALREADY INSTALLED ON YOUR SYSTEM

If MPLAB IDE v8.xx or earlier is already installed on your computer, you will run the Switcher program to switch from MPLAB IDE v8 drivers to MPLAB X IDE drivers, which are installed the first time your switch.

The Switcher program is a GUI application called MPDDSwitch (32-bit OS) or MPDDSwitch64 (64-bit OS). This should be available as the desktop icon MPLAB Driver Switcher.

FIGURE 2-1: SWITCHER UTILITY



1. Plug your desired tool into USB connector on your personal computer.
2. Open the computer's Device Manager window. For example, on a personal computer running Windows XP, right click on the My Computer icon and select "Properties". In the System Properties dialog, click the **Hardware** tab and then click the **Device Manager** button.
3. Expand the "Microchip Tools" section to view the current driver for your tool. The name should be of the form: Microchip *Tool Name*.
4. Go to the MPLAB X IDE install folder and find the Switcher folder, by default:
   a) 32-bit OS: `C:\Program Files\Microchip\MPLABX\Switcher`
   b) 64-bit OS: `C:\Program Files (x86)\Microchip\MPLABX\Switcher`
5. Under the `Switcher` folder, go to the folder for your operating system - `32Bit` or `64Bit`.
6. Launch `MPDDSwitch.exe` (32-bit OS) or `MPDDSwitch64.exe` (64-bit OS).
7. If your MPLAB IDE v8 or MPLAB X IDE installation is not in the default directory, click **Advanced** to specify the location of the driver files.
8. To install or switch USB drivers:
   a) Click to select the *connected* tool for which you wish to switch drivers under "Tool Group and Action".
   b) Click the radio button for either "MPLAB 8" or "MPLAB X".
   c) Click **Apply All**. Switcher progress will be shown in the large text window. This may take some time.

   Note: If the tool(s) are not connected when Switcher is run, the driver(s) will not be installed or switched for those particular tools.
9. If the GUI fails to install the drivers, check the paths to the driver files by clicking **Advanced**. Run the Switcher again.
10. If the GUI still fails to install the drivers, you will need to install the drivers manually. For instructions and driver locations, see **Section 2.3.2.6 "If You Need to Manually Install the Drivers"**.
11. Once the program/batch completes, view the name of the drivers in the Device Manager window. It should say "Microchip WinUSB Device".

Once your MPLAB X IDE drivers are installed, you can switch your drivers back and forth between MPLAB IDE v8.xx and MPLAB X IDE.

2.3.2.5 IF MPLAB IDE V8.XX IS NOT INSTALLED ON YOUR SYSTEM

You do not need to do anything; the USB drivers will be preinstalled when MPLAB X IDE is installed. Once you plug your tool into a computer USB port, a "New Hardware Found" notification should appear. Then either the install will proceed automatically or your will have to follow a wizard and choose to "Automatically select driver". However, if either procedure fails to install the drivers, you will need to install the drivers manually. For instructions and driver locations, see **Section 2.3.2.6 "If You Need to Manually Install the Drivers"**.

2.3.2.6 IF YOU NEED TO MANUALLY INSTALL THE DRIVERS

If you need to install the drivers manually:

1. Open the Device Manager (under Control Panel). Look under “Microchip Tools” for your tool or, if you cannot find it there, under “Other Devices” for “Unknown Device”.
2. Right-click on your tool name or “Unknown Device” and select “Update Driver Software”.
3. In the Update Driver Software dialog, select “Browse my computer for driver software”.

   Note: DO NOT select “Search automatically for updated driver software”. This will install the wrong device driver. If you accidentally select this, back out or exit and repeat these steps to install the correct driver.

4. Locate the correct device driver for your system. The default locations for the device drivers are:

   ```
   C:\Program Files\Microchip\MPLABX\Switcher\32Bit\winusb\
      x86\MCHPWinUSBDevice.inf
   ```

   or

   ```
   C:\Program Files\Microchip\MPLABX\Switcher\64Bit\winusb\
      amd64\MCHPWinUSBDevice.inf
   ```

   For Windows 8, the drivers are in either the `x86_Windows8` or in the `amd64_Windows8` subfolder.

5. If a Windows Security dialog pops up, select “Install this driver software anyway” to proceed to install the drivers.

2.3.2.7 TOOL COMMUNICATION ISSUES

1. If you are using a docking station or hub and have issues after plugging in the tool, you may need to plug the tool directly into a USB port on your computer. This is a known issue with the WinUSB driver.
2. If you need to reinstall a driver manually, you will need to point to the INF file in the `32Bit` or `64Bit` folder. See **Section 2.3.2.6 “If You Need to Manually Install the Drivers”** for details.

2.4 CONNECT TO A TARGET (FOR HARDWARE TOOLS)

For in-circuit debuggers and emulators, refer to the following to determine how to connect your hardware tool to a target:

- the Development Tools Design Advisory
- the Header Specification (if you are using a header)
- your tool documentation

For dedicated programmers, refer to your tool documentation for connection information.

If you are using a Microchip demonstration board, evaluation kit or reference design as your target, please refer to the accompanying documentation for set up information.

2.5 INSTALL THE LANGUAGE TOOLS

When you install MPLAB X IDE, the following language tools are installed as well: MPASM toolchain (ASM30 toolchain no longer included).

Currently there are several C compiler toolsuites (compiler, assembler, linker, etc.) that can be used with MPLAB X IDE. Go to the Microchip web site:

http://www.microchip.com/xc

where you can find free compilers (Free, Evaluation) and full-featured, code-optimized compilers (Standard, Pro).

To select a compiler toolsuite, consider which device you wish to use and then choose a toolsuite that supports that device.

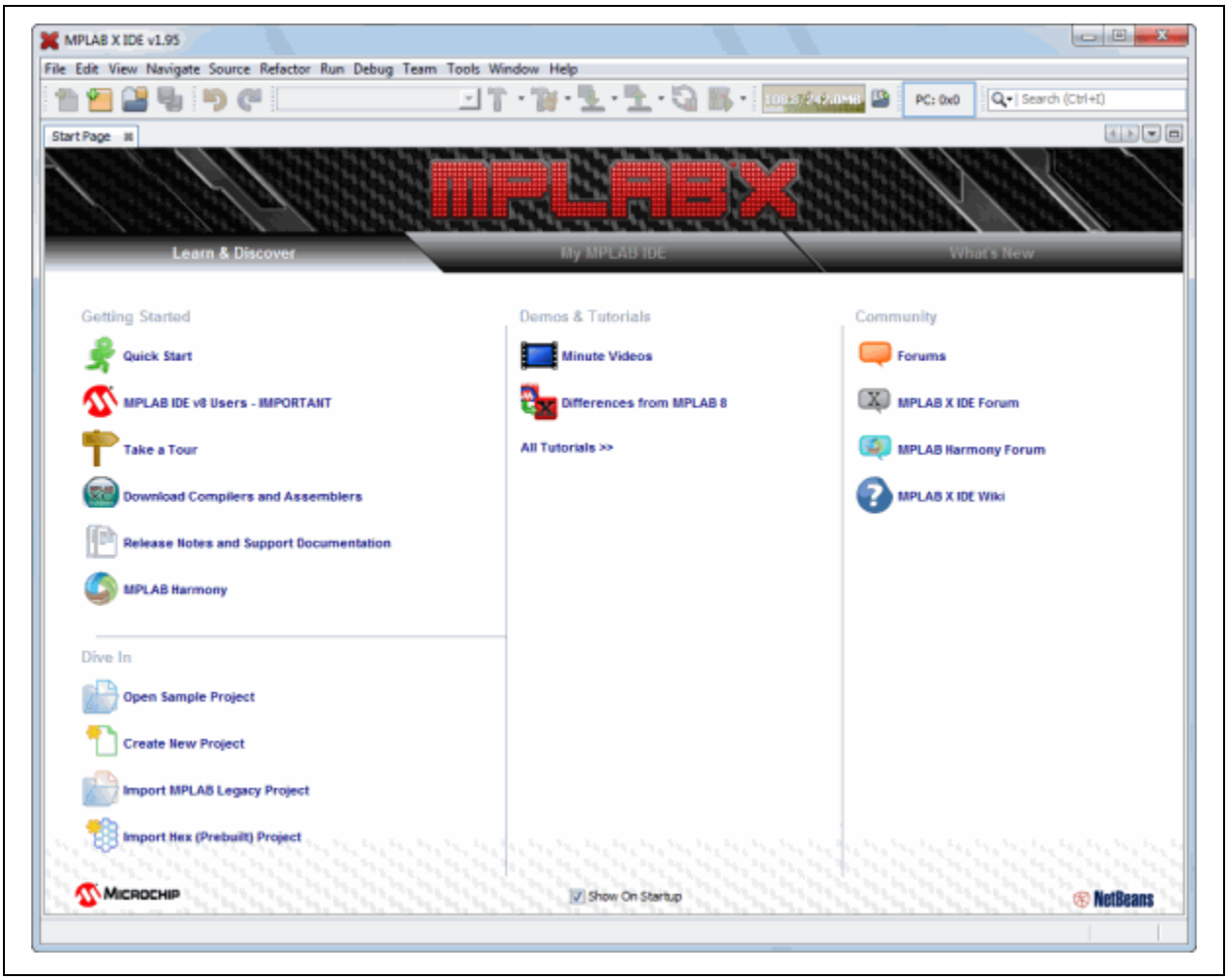
To install and license the compiler you want, view the document *Installing and Licensing MPLAB XC C Compilers* (DS50002059). MPLAB X IDE includes the option to license your installed compiler and roam in and out network licenses. See the "Licenses" option under **Section 11.2.10 "Tools Menu"**.

2.6 LAUNCH THE IDE

Double click on the MPLAB X IDE icon to launch the program.

MPLAB X IDE is built upon the NetBeans platform. If you are familiar with the NetBeans IDE, then the MPLAB X IDE desktop will look familiar.

FIGURE 2-2: MPLAB® X IDE DESKTOP



On the **Start Page**, there are 3 tabs with links. The items on each tab are defined below.

TABLE 2-1: LEARN AND DISCOVER

Getting Started	
Quick Start	Get started quickly by creating and setting up a project.
MPLAB® IDE v8 Users – IMPORTANT	Install WinUSB device drivers for your hardware tools.
Take a Tour	Take a tour of MPLAB X IDE operation.
MPLAB® X IDE Limitations	What is and is not currently supported on this version.
Dive In	
Open Sample Project	Open a functional project as an example.
Create New Project	Create a new MPLAB X IDE project. It is recommended that you view the "Quick Start" before creating your first project.
Import Legacy Project	Import your existing MPLAB IDE v8 project. It is recommended that you view the "Quick Start" before working with an imported project.
New (Hex) Prebuilt Project	Import the hex file from a prebuilt project for another tool.
Demos and Tutorials	
Minute Videos	View short videos explaining MPLAB X IDE operation.
Difference from MPLAB v8	View differences between MPLAB X IDE and MPLAB IDE v8.
All Tutorials	View all available tutorials.
Community	
Forums	Go to the Microchip forums web page.
MPLAB X IDE Forum	Register for the MPLAB X IDE forum.
MPLAB X IDE Wiki	Open the MPLAB X IDE developer's help center.

TABLE 2-2: MY MPLAB X IDE

Recent Projects	
MyProject.c:	List of recently opened projects.
Extend MPLAB	
Selecting Simple or Full Featured Menus	On initial start-up, MPLAB X IDE displays simple menus. For more features, follow these instructions.
Install More Plug-Ins	Open the plug-ins dialog.
Notes and Newsletters	
ANxxxx, TBxxxx	Featured application notes and technical briefs.
microSOLUTIONS E-newsletter	Featured newsletters.
All App Notes/ Newsletters	View all available.
References and Featured Links	
Data Sheets, etc.	Click a link to go to the item described.

TABLE 2-3: WHAT'S NEW

Data Sheets and Errata	
Data Sheet, Silicon Errata	List of featured data sheets and errata. To see a list of all these documents, all click "All Data Sheets" or "All Errata".
Reference Manuals and Programming Spec	
Family Reference Manual, Device Programming Spec	List of featured reference manuals and programming specifications. To see a list of all these documents, all click "All Reference Manuals" or "All Programming Specs".
Recently Released Software	
Source code	List of recent software supporting Microchip device development. To see a list of all these documents, all click "All Recently Released Software".
Product and Corporate News	
New stuff, new news	List of featured Microchip products and news. To see a list of all these documents, all click "All News".

2.7 LAUNCH MULTIPLE INSTANCES OF THE IDE

Some setup is required before using hardware tools (PICkit 3, etc) with an instance of MPLAB X IDE. After any hardware tool setup, an instance of the IDE may be invoked from its own directory.

- Setting Up Hardware Tools to Work with Multiple Instances
- Invoking Instances of the IDE

2.7.1 Setting Up Hardware Tools to Work with Multiple Instances

By default, you can work with up to five (5) instance of the IDE. If you want to have more instances, you will need to manually modify the "mchpdefport" file.

2.7.1.1 USE AND FORMAT OF "MCHPDEFPORT" FILE

The "mchpdefport" file provides the information necessary for tool hot-plug use to both the IDE and to the low-level USB library (DLL, so, or dylib file). The format for this file is as follows:

```
localhost
30000
30002
30004
30006
30008
```

The first line indicates the host name on which the IDE is running.

The other lines represent port or socket numbers through which the low-level library communicates with the upper-level IDE. Each instance of the IDE will be assigned to a different port or socket. All communications between the instances of the IDE should be hidden from the user.

For up to five (5) instances of MPLAB X IDE, you do not need to alter this file. If you want more than five instances, you can edit the file to add more port or socket numbers.

2.7.1.2 LOCATION OF THE "MCHPDEFPORT" FILE

Within a default installation of the IDE, the "mchpdefport" file can be found in the following places:

OS	Location
Windows (64-bit)	`C:\Windows\system32` and `C:\Windows\SysWOW64` (Both occurrences of "mchpdefport" must be modified.)
Windows (32-bit)	`C:\Windows\system32`
Linux	`/etc/.mplab_ide`
Mac (OSX)	`/etc/.mplab_ide`

2.7.2 Invoking Instances of the IDE

MPLAB X IDE requires each instance to have its own user directory. Therefore, preferences set or plug-ins added to one instance will not be reflected in another.

In order to invoke multiple instances launch the IDE with the `--userdir` option and specify a directory.

2.7.2.1 WINDOWS OS

Create a shortcut with the `--userdir` option. For example:

1. Right click on the desktop and select *New>Shortcut*.
2. Browse to the installed MPLAB X IDE executable, by default at:
   ```
   "C:\Program Files\Microchip\MPLABX\mplab_ide\bin\
    mplab_ide.exe"
   ```
3. At the end of the line, enter:
   ```
   --userdir "C:\Documents and Settings\MyFiles\
     ApplicationData\.mplab_ide\dev\beta7Instance2"
   ```
4. Click **OK**.

2.7.2.2 LINUX OS

The installed version run without any parameters (clicking on the desktop icon) will run with a user directory of `$(HOME)/.mplab_ide`. To change the user directory, run the `$InstallationDir/mplab_ide/bin/mplab_ide` shell script passing the argument `--userid anydir`. For example, to run MPLAB X IDE in two different instances:

```
$ /opt/microchip/mplabx/mplab_ide/bin/mplab_ide --userdir ~/.anydir1 &
$ /opt/microchip/mplabx/mplab_ide/bin/mplab_ide --userdir ~/.anydir2 &
```

You can create desktop icons that have the user ID embedded too.

2.7.2.3 MAC OS

Open a Shell window and type the following command line to execute your installation of MPLAB X IDE (example: Beta 7.12) in the alternate user directory:

```
$/bin/sh /Applications/microchip/mplabx/712/mplab_ide.app/Contents/
 Resources/mplab_ide/bin/mplab_ide --userdir "${HOME}/Library/
 Application Support/mplab_ide/dev/beta7.12”
```

NOTES:

Chapter 3. Tutorial

3.1 INTRODUCTION

This tutorial provides a guided example for working with an MPLAB X IDE project.

- Setting Up the Hardware and Software
  - Tutorial Equipment
  - Installation and Set Up
- Creating and Setting Up a Project
  - Create a New Project
  - View Changes to the Desktop
  - View or Make Changes to Project Properties
  - Set Options for Debugger, Programmer or Language Tools
  - Set Language Tool Locations
  - Add An Existing File to the Project
  - Editor Usage
  - Configuration Bits
- Running and Debugging Code
  - Build a Project
  - Run Code
  - Debug Run Code
  - Control Program Execution with Breakpoints
  - Step Through Code
  - Watch Symbol Values Change
  - View Device Memory (including Configuration Bits)
  - Program a Device

3.2 SETTING UP THE HARDWARE AND SOFTWARE

The following information discusses preparation to begin using the MPLAB X IDE.

3.2.1 Tutorial Equipment

The products used in this tutorial are:

Tool	Web Page	Order Number
MPLAB® X IDE	http://www.microchip.com/mplabx	Free
MPLAB® XC32 C Compiler*	http://www.microchip.com/xc	SW006023-1 (Standard Version)
MPLAB® REAL ICE™ in-circuit emulator	http://www.microchip.com/realice	DV244005
Explorer 16 Development Board	http://www.microchip.com/explorer16	DM240001
PIC32MX360F512L PIM	http://www.microchipdirect.com/product-search.aspx?Keywords=MA320001	MA320001

* You may acquire a free or evaluation edition of this compiler from the microchip website. Download the compiler and, when installing, do not enter a license number.

3.2.2 Installation and Set Up

See **Chapter 2. "Before You Begin"** to install MPLAB X IDE, set up the emulator (install USB drivers and properly connect to your target), and install the 32-bit language tools. Then launch MPLAB X IDE and begin this tutorial.

3.3 CREATING AND SETTING UP A PROJECT

The following information discusses setting up projects in MPLAB X IDE, which are required to develop your application code.

3.3.1 Create a New Project

MPLAB X IDE is project-based, so you must set up a project to work on your application.

New projects can be created by selecting either:

- **Start** page, **Learn & Discover** tab, “Dive In” section, “Create New Project” link
- *File>New Project* (or Ctrl+Shift+N)

The New Project Wizard will launch to guide you through new project set up.

STEP 1

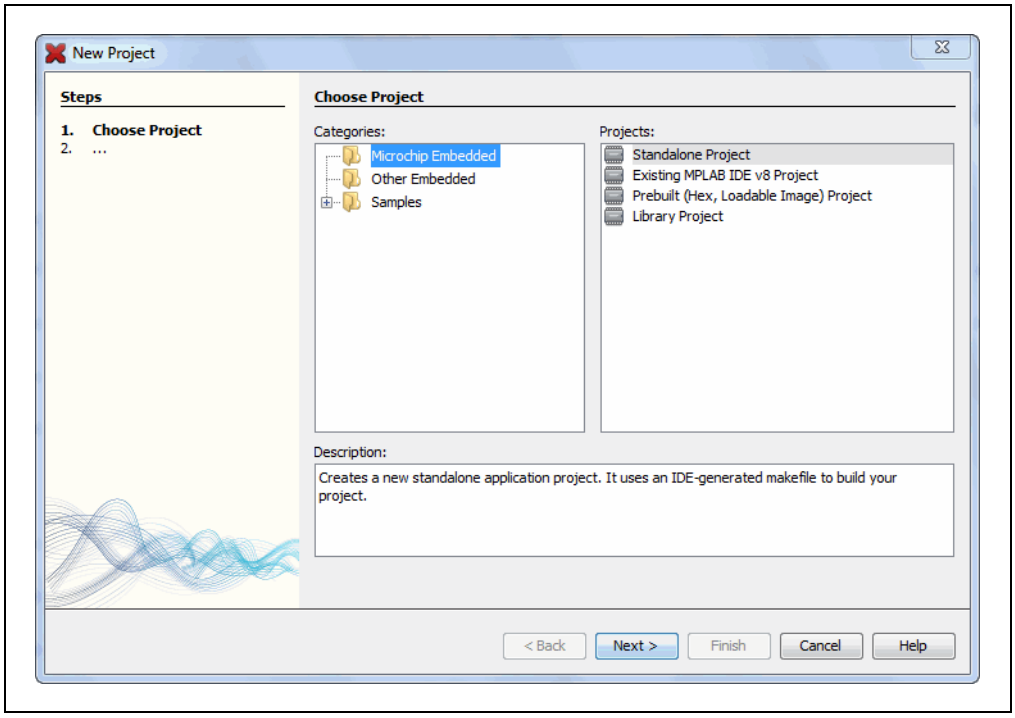
Step 1 asks you to choose a project category.

This is a NetBeans dialog. To work with Microchip products, choose “Microchip Embedded”.

Secondly, choose a project type. For this tutorial, choose “Stand-alone Project”.

Click **Next>** to move to the next dialog.

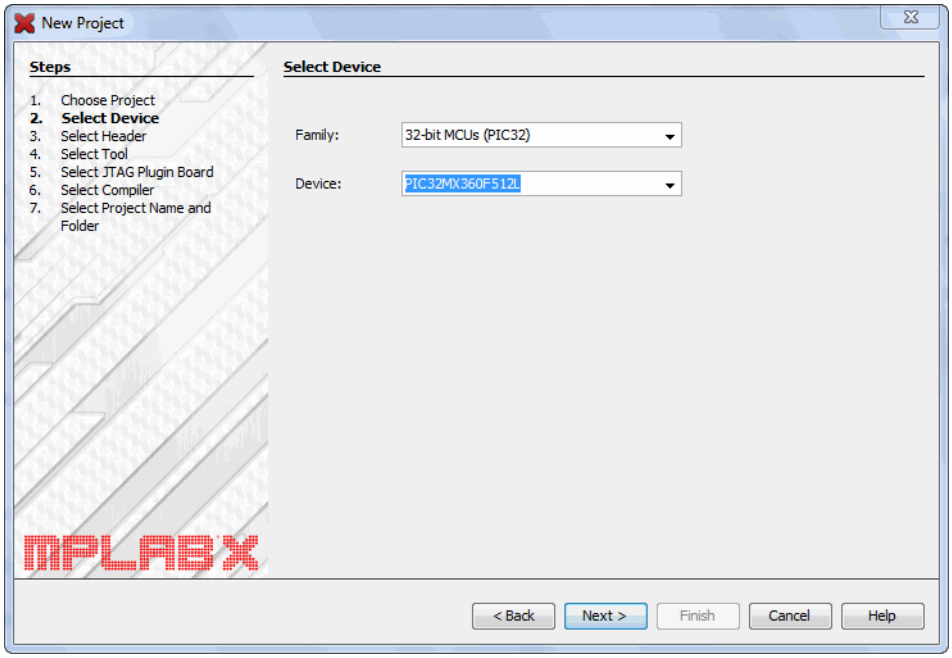
FIGURE 3-1: PROJECT WIZARD – CHOOSE PROJECT



STEP 2

Step 2 is for choosing your device, in this case PIC32MX360F512L. When you are done, click **Next>**.

FIGURE 3-2: PROJECT WIZARD – SELECT DEVICE



STEP 3

Step 3 only appears if a header is available for your selected device. Since there is no header for the PIC32MX360F512L device, MPLAB X IDE knows to skip this step.

STEP 4

Step 4 selects the tool.

Tool support for the selected device is signified by the colored circles (lights) in front of the tool name. If you cannot see the colors, mouse over a light to pop up text about support.

Light	Color	Support
●	Green	Full (Implemented and fully tested)
●	Yellow	Beta (Implemented but not fully tested)
●	Red	None

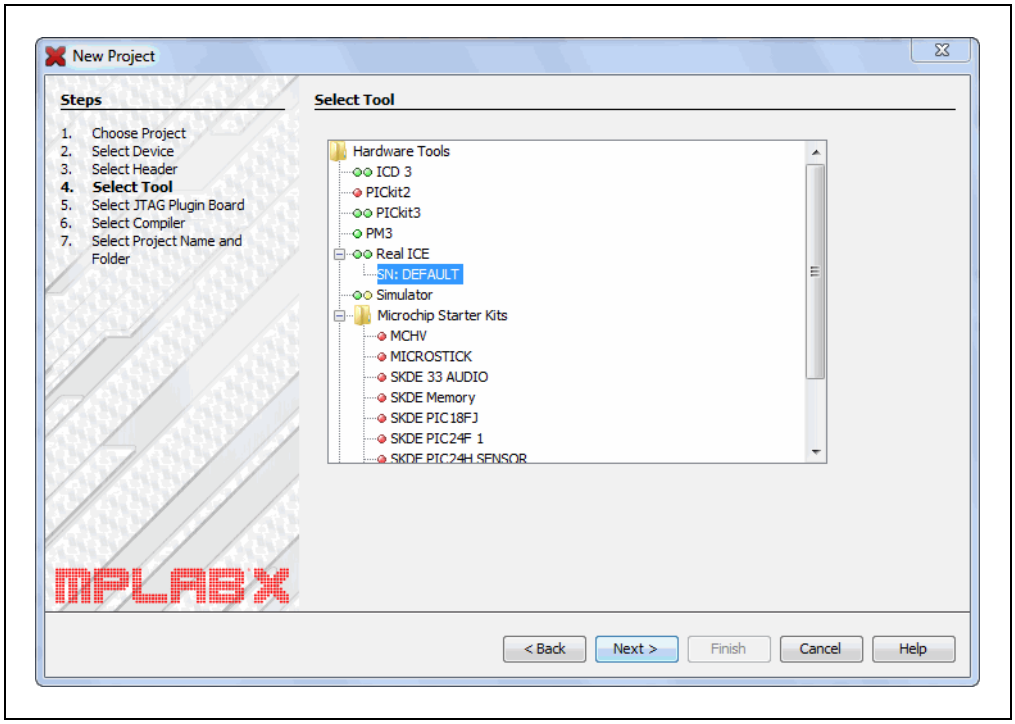
For some tools, there are two lights next to the tool name, where the *first light* is the left-most light and the *second light* is to the right of the first.

Light No.	Debug Tools	Simulator
1	Debugger Support	Core (Instruction Set) Support
2	Programmer Support	Peripheral Support

For the hardware tools, you will notice that a serial number (SN) is specified below any tool that is connected to your computer. This allows you to select from several connected hardware tools.

Select your tool and then click **Next>**.

FIGURE 3-3: PROJECT WIZARD – SELECT TOOL



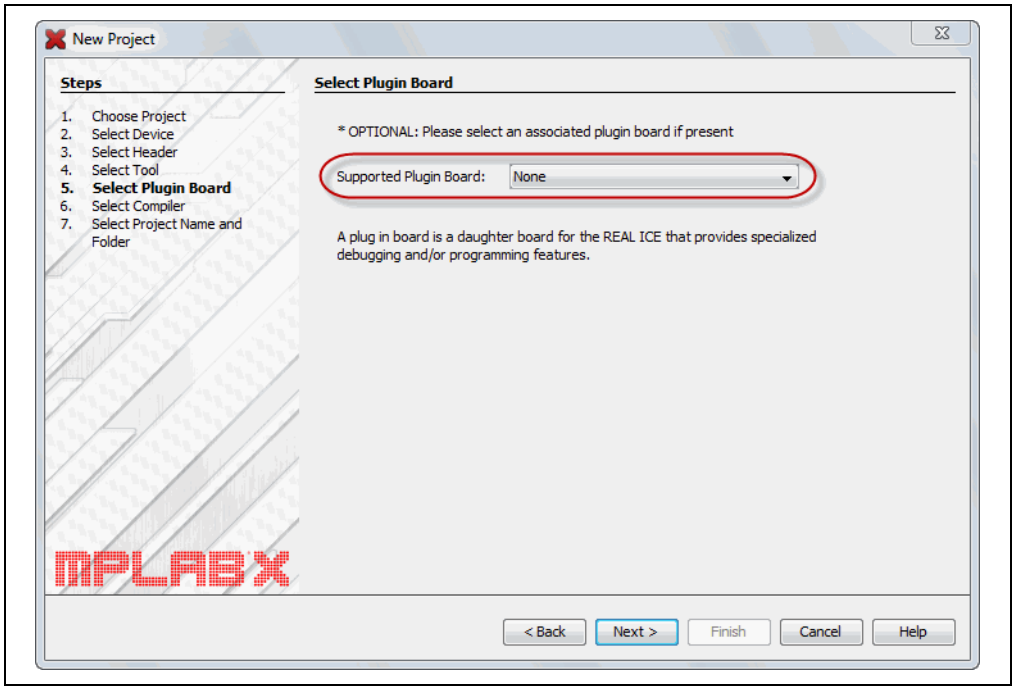
STEP 5

Step 5 only appears if MPLAB REAL ICE in-circuit emulator is selected as the tool.

For the MPLAB REAL ICE in-circuit emulator, you may specify a plug-in board to use. A plug-in board is the circuit board that is inserted into the emulator's driver board slot. Since the Explorer 16 board works with either the Standard or High-Speed Communications driver boards, leave the "Supported Plugin Board" as "None".

Select your tool and then click **Next>**.

FIGURE 3-4: PROJECT WIZARD – SELECT PLUGIN



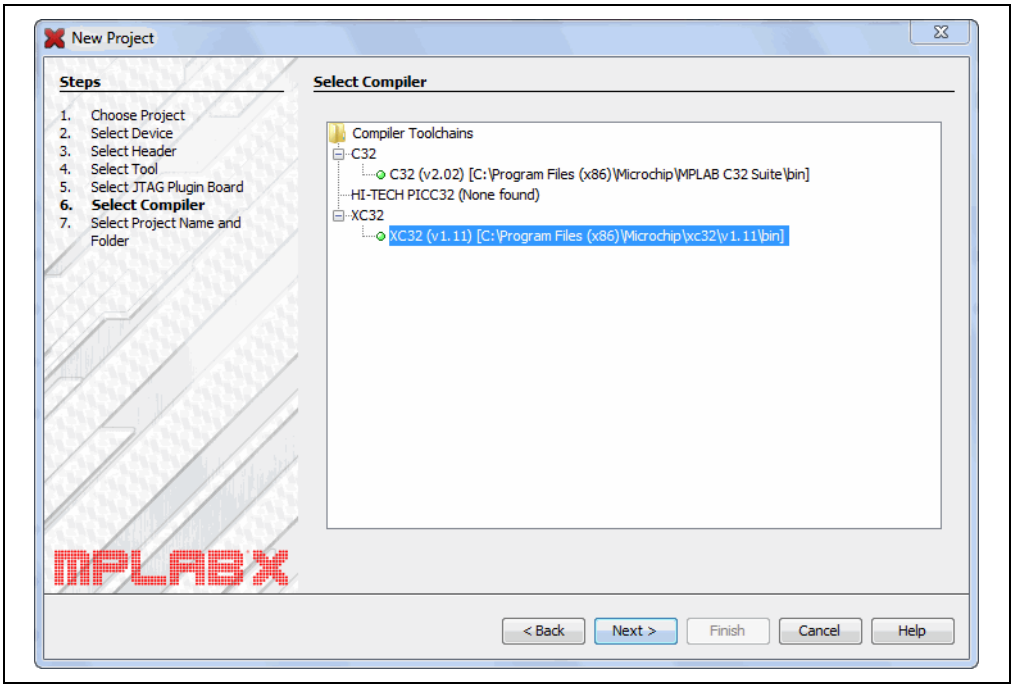
STEP 6

Step 6 selects the language tool, either a C compiler or assembler. Again, the colored circle (light) in front of the compiler name signifies the device support level. Mouse over for text.

The version and installation location of a language tool is displayed beneath that tool. This allows you to select from several installed language tools.

Select your tool and then click **Next>**.

FIGURE 3-5: PROJECT WIZARD – SELECT LANGUAGE TOOL



STEP 7

Step 7 selects the project name, location and other project options.

Enter the project name `MyProject`.

By default, projects will be placed in:

- Windows XP – `C:\Documents and Settings\`*`UserName`*`\MPLABXProject`
- Windows 7/8 – `C:\Users\`*`UserName`*`\MPLABXProjects`
- Linux – `/home/`*`UserName`*`/MPLABXProjects`
- Mac – `/Users/`*`UserName`*`/MPLABXProjects`

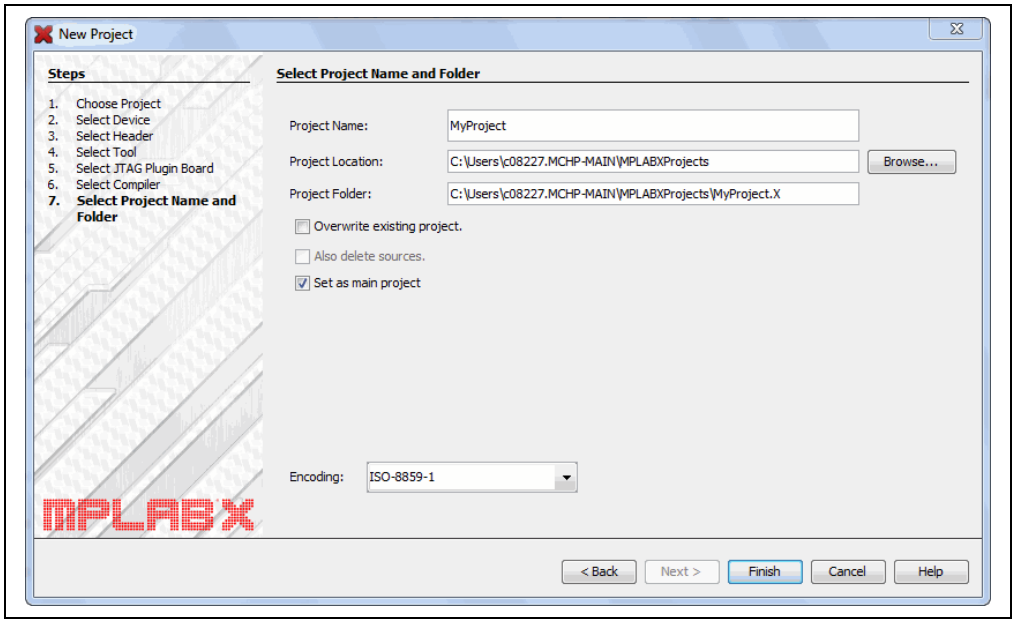
If the Project Location does not point here, browse to the appropriate location.

Check "Set as main project" to make this your main project.

This tutorial was produced with the encoding set to ISO-8859-1 (Latin 1) so you do not need to change this setting.

When you are done, select **Finish** to complete new project creation.

FIGURE 3-6: PROJECT WIZARD – SELECT PROJECT NAME AND FOLDER

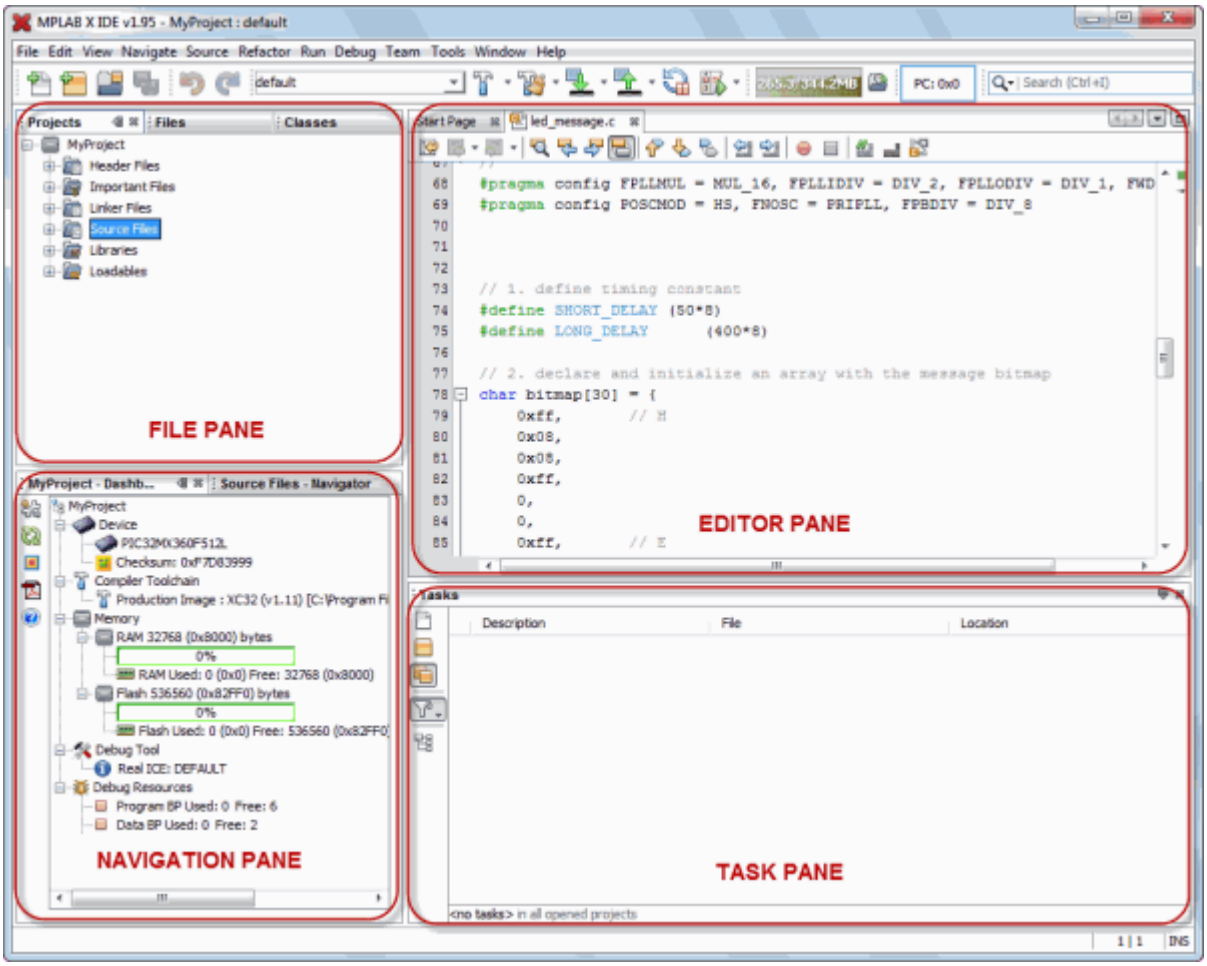


3.3.2 View Changes to the Desktop

Once you have created your project, several panes will open in the IDE.

- **File pane** – A pane with four tabbed windows - Projects, Files, Classes and Services (not shown automatically) windows. In this tutorial, we will focus on the Projects window, which displays the project tree with files grouped by category.
- **Navigation pane** – A pane that displays information on the file or project selected. For a project, the project environment shows details about the project and for a file, symbols and variables are shown.
- **Editor pane** – A pane for viewing and editing project files. The Start Page also is visible here.
- **Task pane** – A pane that displays task output from building, debugging or running an application.

FIGURE 3-7: MPLAB X IDE DESKTOP



If you double click on any file name in the File pane, the related file will open in the Editor pane under a tab next to the Start Page. To close the tab, click on the “x” next to the file name.

Right click on the project name in the File pane, Projects window, to view the pop-up (context) menu. Do the same for the project’s subfolders.

3.3.3 View or Make Changes to Project Properties

Once a project has been created, you can view or change the project properties in the Project Properties dialog.

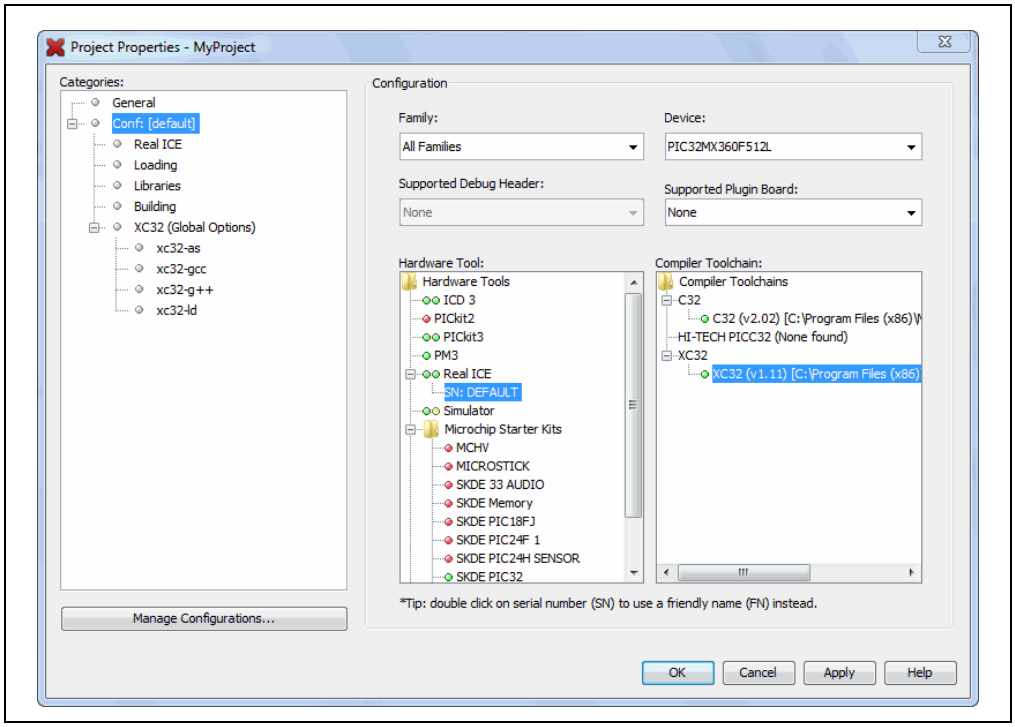
Access this dialog by either:

- right clicking on the project name in the Projects window and selecting "Properties".
- clicking on the project name in the Projects window and then selecting *File>Project Properties*.

Click the "Conf:[default]" category to reveal the general project configuration, such as the project device, related debug/programmer tool, and language tool. Do not change any of these items for this tutorial unless you have made a mistake in previous sections.

Then update in this dialog and click **Apply**.

FIGURE 3-8: PROJECT PROPERTIES DIALOG



3.3.4 Set Options for Debugger, Programmer or Language Tools

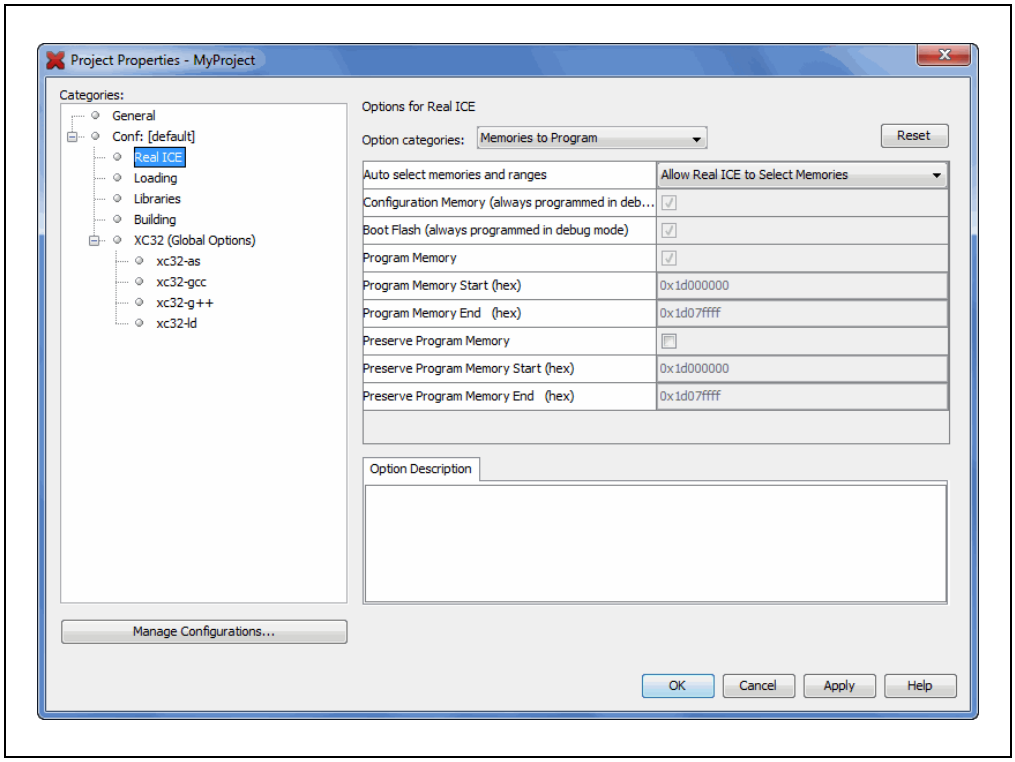
Additionally, you set options for your tools in the Project Properties dialog.

To set up or change debugger/programmer tool options:

- Click on REAL ICE to see related setup options. For more on what these options mean, see the emulator documentation.

Do not make any changes for this tutorial.

FIGURE 3-9: TOOL SETUP PAGE

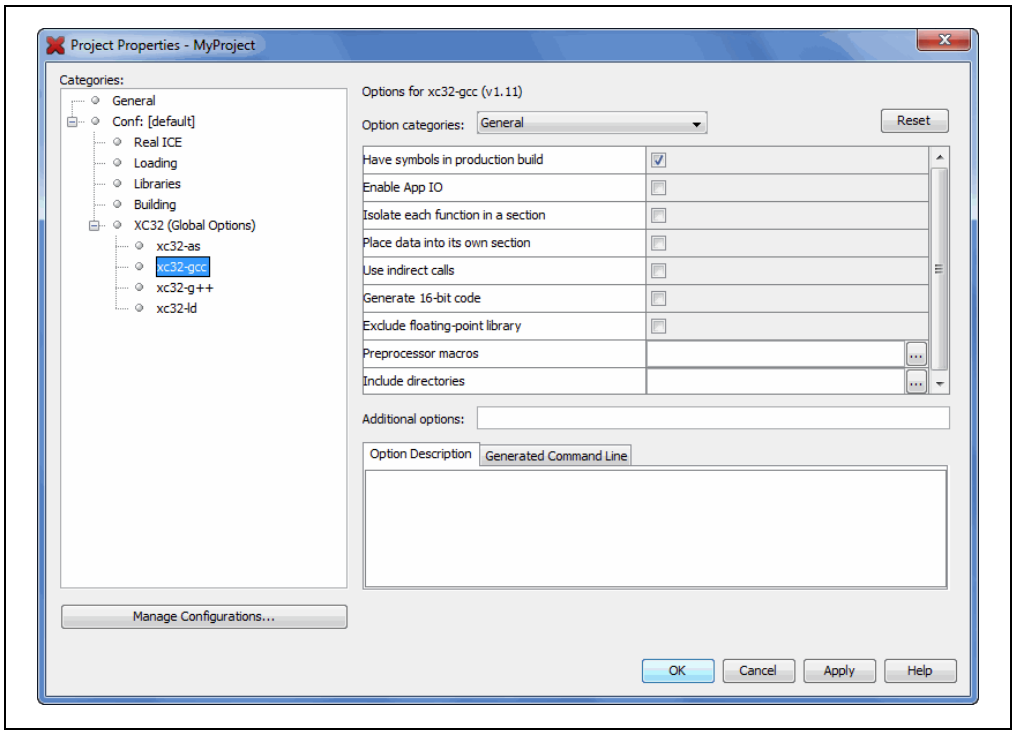


To set up or change language tool options:

- Click on your language tool to see related setup options. For more on what these options mean, see your language tool documentation.

Do not make any changes for this tutorial.

FIGURE 3-10: LANGUAGE TOOL SET UP PAGE



3.3.5 Set Language Tool Locations

To see what language tools are available to MPLAB X IDE and view or change their paths:

- **For Mac OS**:
  Access the build tools from *mplab_ide>Preferences>Embedded>Build Tools* from the main menu bar.
- **For Other OSs:**
  Access the build tools from *Tools>Options>Embedded>Build Tools*.

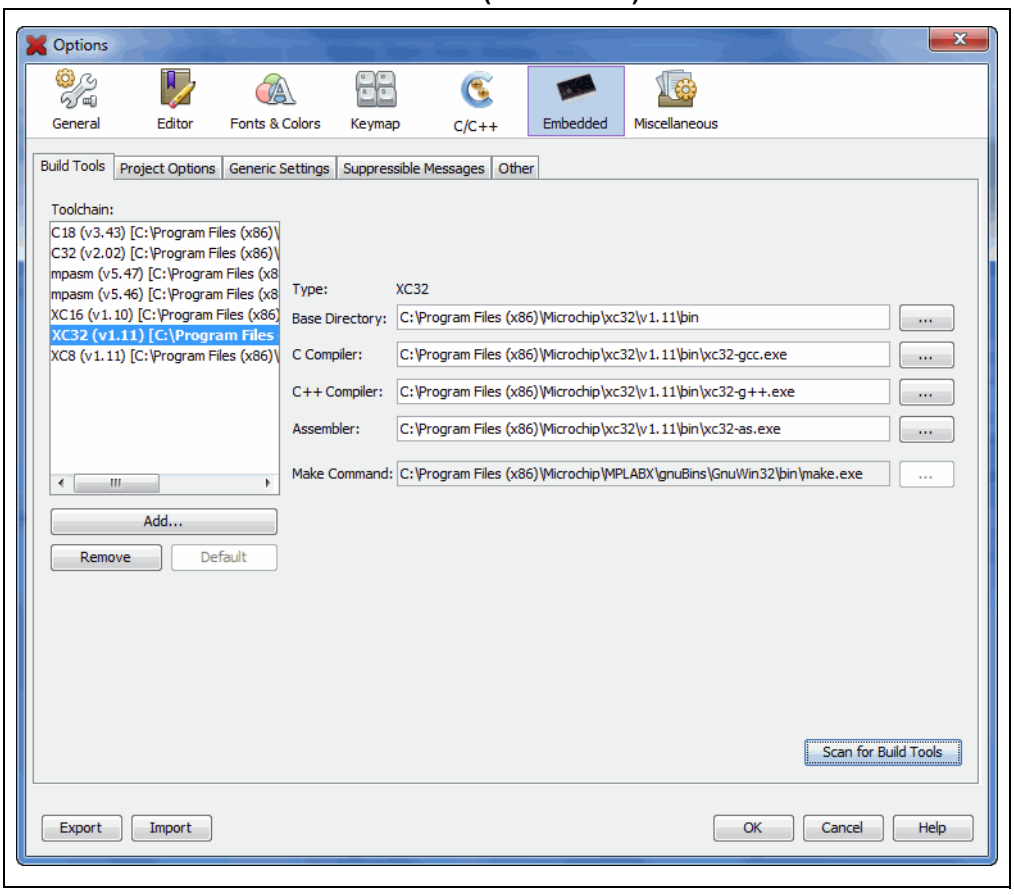
The window should automatically populate with all installed toolchains. If you do not see your tool listed, try the following:

- **Scan for Build Tools** – Scans the environment path and lists the language tools installed on the computer.
- **Add** – Manually add the tool to the list by entering the path to the directory containing the tool executable(s), i.e., base directory. Typically, this is the `bin` subdirectory in the tool installation directory.

If you have more than one version of a compiler available, select one from the list.

Ensure that the XC32 toolchain is selected for this tutorial.

FIGURE 3-11: LANGUAGE TOOL (COMPILER) LOCATIONS



3.3.6 Add An Existing File to the Project

For this tutorial, you will use existing example code:

- Go to the Explorer 16 Development Board web page on the Microchip web site: http://www.microchip.com/explorer16
- Click "PIC32 Explorer 16 LED Example Application" to download the ZIP file containing the example code.
- Once downloaded, unzip the project.
- Move the file `led_message.c` to the project directory (`MyXProject.X`).

Existing files can be added to a project by doing one of the following:

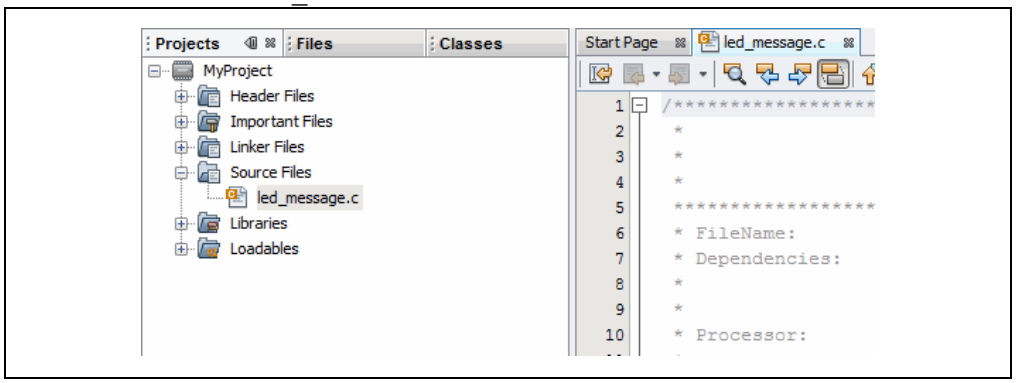
- Right clicking on the project in the Project/File window and selecting "Add Existing Item"
- Right clicking on a logical folder (e.g., Source Files) in the Project/File window and selecting "Add Existing Item"

When adding a file, you can choose whether to add it as:

- Auto – Let MPLAB X IDE decide how best to locate the file.
- Relative – Specify the file location relative to the project. (Recommended: Relative paths will help portability when moving a project to another computer.)
- Absolute – Specify the file location by an absolute path.
- Copy – Copy the specified file to the project folder.

The file will appear in the File pane under the project specified and a tab with the file's name will appear in the Editor pane.

FIGURE 3-12: `LED_MESSAGE.C` IN FILE PANE



Code Listing for `led_message.c`:

```
/********************************************************************
 *
 *                  Message via 8 LEDs
 *
 ********************************************************************
 * FileName:        led_message.c
 * Dependencies:p32xxxx.h
 *
 *
 * Processor:       PIC32
 *
 * Complier:        MPLAB C32
 *                  MPLAB IDE v8.0+
 * Company:         Microchip Technology, Inc.
 *
 * Software License Agreement
 *
 * The software supplied herewith by Microchip Technology Incorporated
 * (the "Company") for its PIC32 Microcontroller is intended
 * and supplied to you, the Company's customer, for use solely and
 * exclusively on Microchip PIC32 Microcontroller products.
 * The software is owned by the Company and/or its supplier, and is
 * protected under applicable copyright laws. All rights are reserved.
 * Any use in violation of the foregoing restrictions may subject the
 * user to criminal sanctions under applicable laws, as well as to
 * civil liability for the breach of the terms and conditions of this
 * license.
 *
 * THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES,
 * WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED
 * TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 * PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT,
 * IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
 * CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 *
 * $Id: led_message.c 5898 2007-10-23 19:39:48Z rajbhartin $
 *
 ********************************************************************/

/*
** Message in a bottle
**
** Explorer16 version (long delays)
**
** Thanks to Lucio DiJasio for letting us use this example.
**
** Run this example on Explorer-16 board with PIC32MX PIM.
** Hold the board vertically from the PICtail connector size
** and wave the board back-and-forth to see message "HELLO" on LEDs
*/

#include <p32xxxx.h>

// Config settings
// POSCMOD = HS, FNOSC = PRIPLL, FWDTEN = OFF
// PLLIDIV = DIV_2, PLLMUL = MUL_16
// PBDIV = 8 (default)
// Main clock = 8MHz /2 * 16    = 64MHz
// Peripheral clock = 64MHz /8  =  8MHz
```

```
// Configuration Bit settings
// SYSCLK = 64 MHz (8MHz Crystal/ FPLLIDIV * FPLLMUL / FPLLODIV)
// PBCLK = 8 MHz
// Primary Osc w/PLL (XT+,HS+,EC+PLL)
// WDT OFF
// Other options are don't care
//
#pragma config FPLLMUL = MUL_16, FPLLIDIV = DIV_2, FPLLODIV = DIV_1,
FWDTEN = OFF
#pragma config POSCMOD = HS, FNOSC = PRIPLL, FPBDIV = DIV_8



// 1. define timing constant
#define SHORT_DELAY (50*8)
#define LONG_DELAY(400*8)

// 2. declare and initialize an array with the message bitmap
char bitmap[30] = {
    0xff,// H
    0x08,
    0x08,
    0xff,
    0,
    0,
    0xff,// E
    0x89,
    0x89,
    0x81,
    0,
    0,
    0xff,// L
    0x80,
    0x80,
    0x80,
    0,
    0,
    0xff,// L
    0x80,
    0x80,
    0x80,
    0,
    0,
    0x7e,// O
    0x81,
    0x81,
    0x7e,
    0,
    0
    };

// 3. the main program
main()
{
    // disable JTAG port
    DDPCONbits.JTAGEN = 0;

    // 3.1 variable declarations
    int i;           // i will serve as the index
```

```
    // 3.2 initialization
    TRISA = 0;      // all PORTA as output
    T1CON = 0x8030; // TMR1 on, prescale 1:256 PB

    // 3.3 the main loop
    while( 1)
    {
        // 3.3.1 display loop, hand moving to the right
        for( i=0; i<30; i++)
        {// 3.3.1.1 update the LEDs
            PORTA = bitmap[i];

            // 3.3.1.2 short pause
            TMR1 = 0;
            while ( TMR1 < SHORT_DELAY)
            {
            }
        } // for i

        // 3.3.2 long pause, hand moving back to the left
        PORTA = 0;      // turn LEDs off
        TMR1 = 0;
        while ( TMR1 < LONG_DELAY)
        {
        }
    } // main loop
} // main
```

3.3.7 Editor Usage

The sample code should not need to be edited. However, when you need to edit your code, you will be using the NetBeans editor. General information on this editor is available from the table of contents under the NetBeans help topic *IDE Basics>Basic File Features*. C compiler information regarding the editor is available from the table of contents under the NetBeans help topic *C/C++/Fortran Development>C/C++/Fortran Project Basics>Navigating and Editing C/C++/Fortran Source Files*. A list of features is available under **Chapter 7. "Editor"**.

To use editor features, go to the:

- Edit menu (see **Section 11.2.2 "Edit Menu"**.)
- Editor toolbar located at the top of each file's Editor window.
- Window right click (context) menu.

FIGURE 3-13: EDITOR TOOLBAR

3.3.8 Configuration Bits

In the sample code, Configuration bits are already set. For your own application, you must set up the Configuration bits.

You must set Configuration bits in code. However, you can temporarily change Configuration bits during a debug session in the Configuration Bits window (*Window>PIC Memory Views>Configuration Bits*). Then you may export the setting by right clicking in the window and selecting "Generate Source Code to Output". You can copy this code from the Output window into your code.

For a summary of Configuration bits settings for different devices, see **Appendix A. "Configuration Settings Summary"**.

3.4 RUNNING AND DEBUGGING CODE

The following information discusses using the MPLAB X IDE to run or debug your code.

3.4.1 Build a Project

For MPLAB X IDE, it is not necessary to build the project first and then run or debug. Building is part of the run and debug processes. For initial development or major changes, however, you may want to make sure that the project builds before attempting to run or debug.

To build a project:

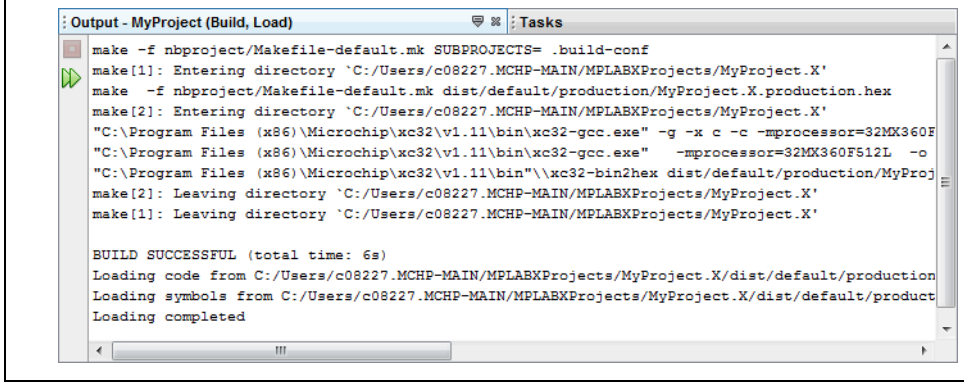
- In the Projects window, right click on the project name and select “Build”. You may also select “Clean and Build” to remove intermediary files before building.
- Click on the “Build Project” or “Clean and Build Project” toolbar icon.

Build Icon

Clean and Build Icon

Build progress will be visible in the Output window (lower right hand corner of the desktop.) For this tutorial, the code should build successfully.

FIGURE 3-14: OUTPUT SUCCESSFUL BUILD



To view checksum information:

Open the Dashboard window (*Window>Dashboard*) if it is not already open to see the checksum after a build.

3.4.2 Run Code

Once the code builds successfully, you can attempt to run the application. Click on the "Make and Program Device Project" icon (or select *Run>Run Project*) to run your program.

Make and Program Device Project Icon

The lights on the demo board should be flickering. Wave the board back and forth to see the word, "Hello".

Run progress will be visible in the Output window as well.

Use the **Hold in Reset** button to toggle between device Reset and running.

Hold in Reset Icon

You can add a "Run Project" icon to the toolbar if you wish (*View>Toolbars>Customize*).

Run Icon

3.4.3 Debug Run Code

For this tutorial, the code used has been tested and runs. However, your own code may need to be debugged as you develop your application.

To Debug Run the tutorial code, click on the "Debug Project" icon (or select *Debug>Debug Project* or *Debug>Step Into*) to begin a debug session.

Debug Run Icon

Debug Run progress will be visible in the Output window.

To halt your application code:

Click on the "Pause" icon (or select *Debug>Pause*) to halt your program execution.

To run your code again:

Click on the "Continue" icon (or select *Debug>Continue*) to start your program execution again.

To end execution of your code:

Click on the "Finish Debugger Session" icon (or select *Debug>Finish Debugger Session*) to end your program execution.

For more details on debugging C code projects, see the NetBeans help topic *C/C++/Fortran Development>Debugging C/C++/Fortran Applications with gdb*.

The difference between Run and Debug Run will become apparent when working with debug features, beginning with **Section 3.4.4 "Control Program Execution with Breakpoints"**.

3.4.4 Control Program Execution with Breakpoints

When debugging code, it can be useful to suspend execution at a specific location in code so that variable values can be examined. To do this, use breakpoints.

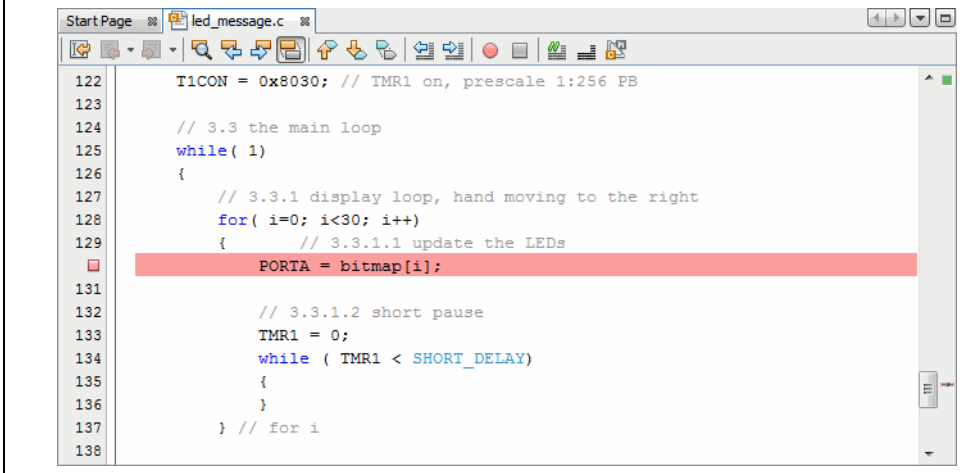
Set a breakpoint on the following line of code:

```
PORTA = bitmap[i];
```

To set a breakpoint on a line do one of the following:

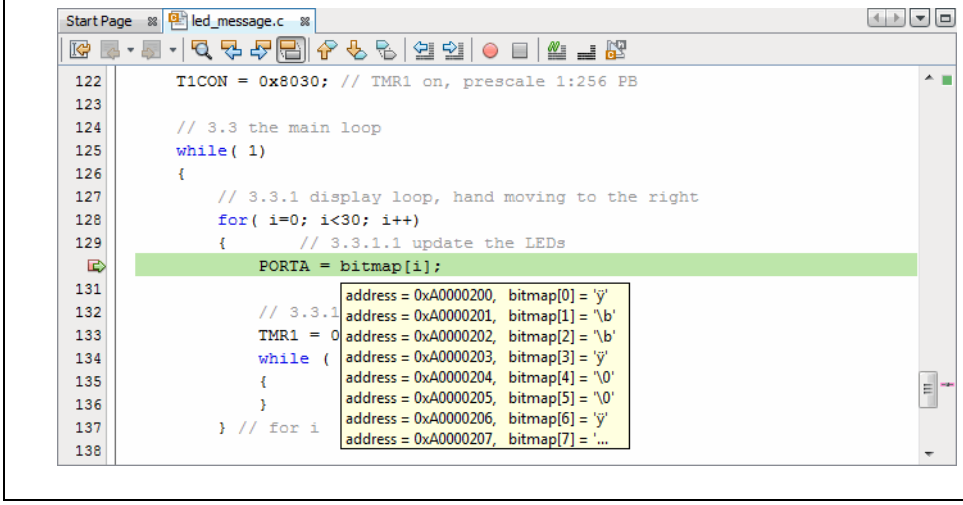
- Click the left margin of the line in the Source Editor
- Press Ctrl-F8

FIGURE 3-15: BREAKPOINT SET IN CODE



Debug Run the tutorial program again. The program will halt at the breakpoint. Hover over the `bitmap[]` variable to see its values.

FIGURE 3-16: PROGRAM EXECUTION HALTED AT BREAKPOINT



To clear the breakpoint do one of the following:

- Repeat the step to set a breakpoint
- Select *Debug>Toggle Breakpoint*

For more on breakpoints, see the NetBeans help topics under *C/C++/Fortran Development>Debugging C/C++/Fortran Applications with gdb>Setting C/C++/Fortran Breakpoints*.

3.4.5 Step Through Code

Use one of the stepping functions on the Debug menu and Debug toolbar to move through code either from the beginning of code or after a breakpoint halt. Examine changes in variable values (see next section) or determine if the program flow is correct.

There are several ways to step through code:

- Step Over – Executes one source line of a program. If the line is a function call, executes the entire function then stops.
- Step Into – Executes one source line of a program. If the line is a function call, executes the program up to the function's first statement and stops.
- Step Out – Executes one source line of a program. If the line is a function call, executes the functions and returns control to the caller.
- Run to Cursor – Runs the current project to the cursor's location in the file and stop program execution.

For more on stepping, see the NetBeans help topics under *C/C++/Fortran Development>Debugging C/C++/Fortran Applications with gdb>C/C++/Fortran Debugging Sessions>Stepping Through Your C/C++/Fortran Program*.

3.4.6 Watch Symbol Values Change

Watch the values of symbols that you select change in the Watches window. Determining if these values are as expected during program execution will help you to debug your code.

To create a new watch:

1. Select *Debug>New Watch*. The New Watch dialog will open (see below).
2. Enter a Watch expression, in this case `PORTA`, and then click **OK**. The Watches window will now appear on the desktop with the symbol.

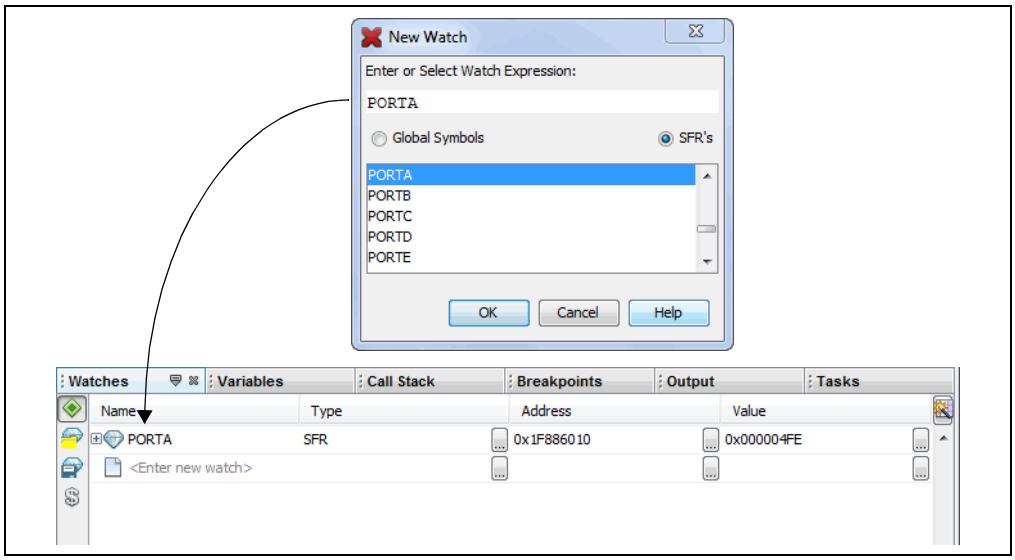
OR

1. Right click on PORTA in code and select “New Watch”.
2. The Watches window will open with PORTA in it.

OR

1. Select *Window>Debugging>Watches*.
2. Drag and drop PORTA from the editor window into the Watches window.

FIGURE 3-17: WATCHES WINDOW WITH SYMBOL



To view symbol changes:

1. Debug Run and then Pause your program.
2. Click the **Watches** tab to view the window and see the symbol value. (Red text means a change.)

For more on watches, see the NetBeans help topics under *C/C++/Fortran Development>Debugging C/C++/Fortran Applications with gdb> Viewing C/C++/Fortran Program Information>Creating a C/C++/Fortran Watch*.

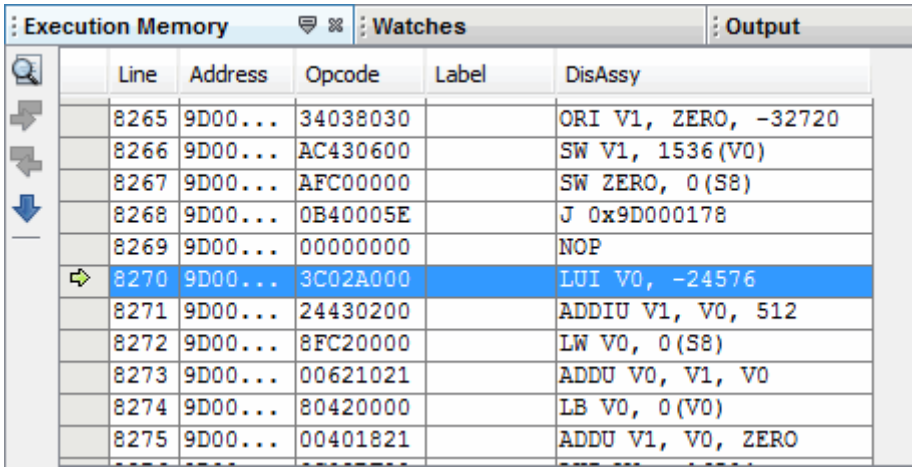
3.4.7 View Device Memory (including Configuration Bits)

MPLAB X IDE has flexible, abstracted memory windows that provide a customizable view of differing types of device memory. From a Memory window, you select the type of memory and memory format in drop-down boxes.

Begin by viewing Flash memory:

1. Select *Window>PIC Memory Views>Execution Memory*.
2. The Execution Memory window will open showing the last halt location.

FIGURE 3-18: MEMORY WINDOW CONTENT



Change the memory view by:

- Selecting another window from the *Window>PIC Memory Views* list.
- Using the drop-down "Memory" menu on the window.

To set Memory window options:

Right clicking in the Memory window will pop up a menu with various options such as display options, fill memory, table import/export and output to file. For more information on this menu, see **Section 12.8.13 "Memory Window Menu"**.

To refresh the Flash Memory window:

1. Halt your program (Finish Debugger Session).
2. Click on the icon named "Read Device Memory".

Read Device Memory Icon

3.4.8 Program a Device

Once your code is debugged, you can program it onto a target device.

First, you check the programming options in the Project Properties window. For this tutorial, no options will be changed.

Second, you program the device in one of two ways:

- Click **Run**: The project is built (if necessary) and device is programmed. The program will immediately begin execution on completion of programming.
- Click **Make and Program Device**: The project is built (if necessary) and device is programmed. The program will NOT immediately begin execution on completion of programming. You will need to disconnect the hardware tool from the target board before the application will run.

Other programming-related functions are:

- **Hold in Reset**: Toggle the device between Reset and Run.
- **Read Device Memory**: Transfer what is in target memory to MPLAB X IDE.

Programming-related icons are:


Run Icon


Hold In Reset Icon


Make and Program Device Icon


Read Device Memory Icon

Note:	Not all programming functions are in the MPLAB X IDE. For additional programming support, see the MPLAB IPE included with the MPLAB X IDE installation.

NOTES:

Chapter 4. Basic Tasks

4.1 WORKING WITH MPLAB X IDE PROJECTS

The following steps show how to work with projects in MPLAB X IDE

Preliminaries

1. Before You Begin, install MPLAB X IDE, set up any hardware tools (install USB drivers and properly connect to your target) and install language tools for your selected device. Then launch MPLAB X IDE and begin this tutorial.

Create and Build a Project

1. Create a New Project by using the New Project wizard. Then View Changes to the Desktop.
2. View or Make Changes to Project Properties in the Project Properties dialog. Also Set Options for Debugger, Programmer or Language Tools in the same dialog.
3. Set Language Tool Locations and Set Other Tool Options in the Tools Options dialog.
4. Create a New File to add to your project or Add Existing Files to a Project. Enter or edit your application code to the File window.
5. Discover other features for Editor Usage.
6. Add and Set Up Library and Object Files.
7. Set File and Folder Properties to keep or exclude individual files or entire folders from the build.
8. Set Build Properties for pre- and post-build steps and loading an alternative hex file on build.
9. Build a Project.

Execute Code

1. Run Code with the Run menu.
2. Debug Run Code with the Debug menu.

Debug Code

1. Control Program Execution with Breakpoints. Set breakpoints in-line or via the Breakpoint window.
2. Step Through Code as the program executes.
3. Watch Symbol Values Change in the Watches and Variables windows.
4. View/Change Device Memory (including Configuration Bits). Memory types are dependent on the device selected.
5. Use View The Call Stack to navigate function calls.

Program a Device

1. Program a Device using simple toolbar buttons.

4.2 CREATE A NEW PROJECT

MPLAB X IDE is project-based, so you must set up a project to work on your application.

New projects can be created by selecting either:

- Start Page, **Learn and Discover** tab, "Dive In", "Create New Project"
- *File>New Project* (or Ctrl+Shift+N)

The New Project wizard will launch to guide you through new project set up.

4.2.1 Step 1: Choose Project

Step 1 will first ask you to choose a project category. In most cases you will choose a project type from "Microchip Embedded":

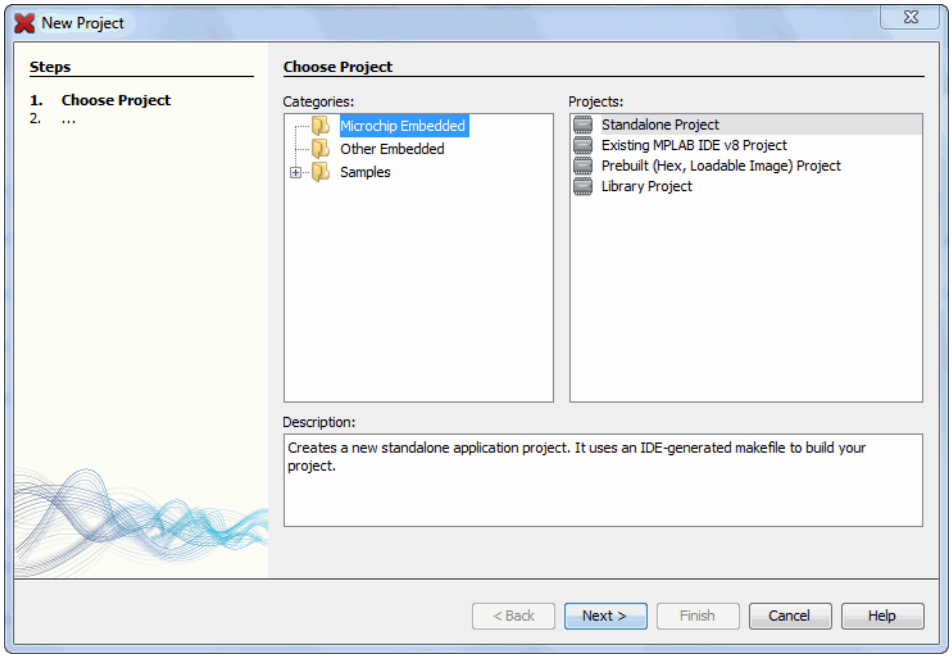
- Standalone Project – Create a new C and/or assembly code project. This type of project is shown in this section.
- Existing MPLAB IDE v8 Project – Convert your existing MPLAB IDE v8 project into an MPLAB X IDE project. For details see **Section 5.2 "Import MPLAB Legacy Project"**.
- Prebuilt (Hex, Loadable Image) Project – Load an existing project image into MPLAB X IDE. For details, see **Section 5.3 "Prebuilt Projects"**.
- Library Project – Create a new C and/or assembly code project that will build into a library instead of executable hex file. For details, see **Section 5.6 "Library Projects"**.

Other options are:

- Other Embedded Projects – Projects from other vendors.
- Sample Projects – Includes ready-to-use projects for different device families and project templates for different device families.

Once you have made your selections, click **Next>** to move to the next dialog.

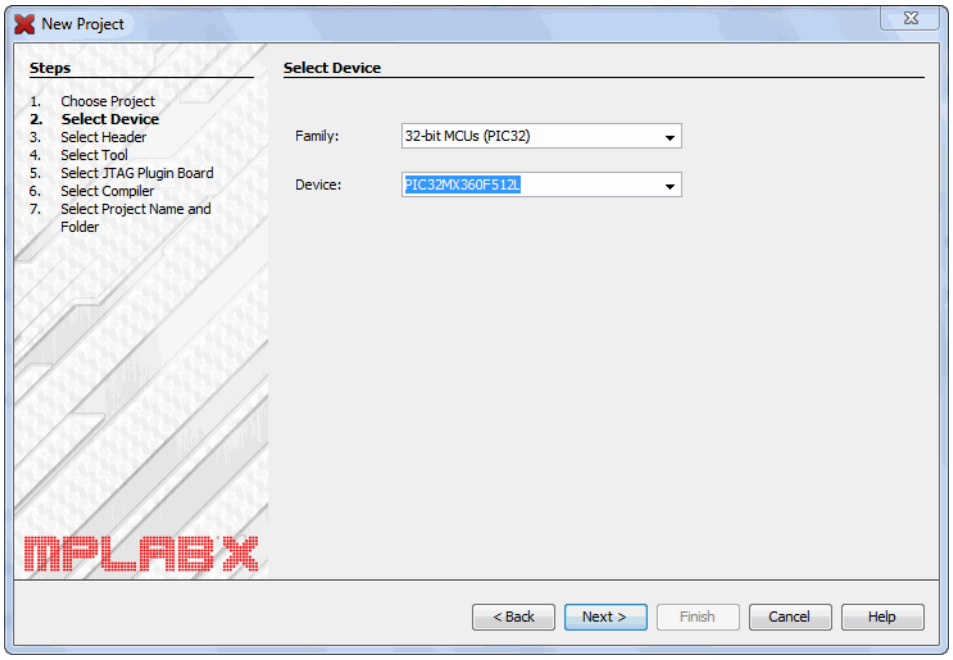
FIGURE 4-1: PROJECT WIZARD – CHOOSE PROJECT



4.2.2 Step 2: Select Device

Step 2 is where you select the device you will be using in your application from the “Device” drop-down list. To narrow your selection list, chose a Family first. Click **Next>**.

FIGURE 4-2: PROJECT WIZARD – SELECT DEVICE

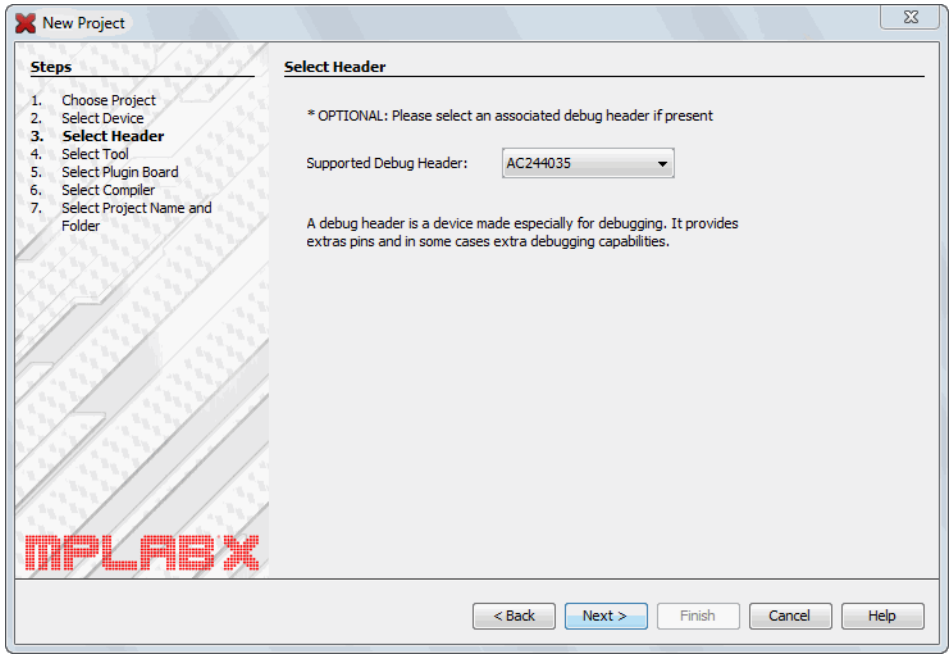


4.2.3 Step 3: Select Header

Step 3 will appear if a header is available for your selected device. To determine if a header is required for debug or if your device has on-board debug circuitry, consult the "*Processor Extension Pak and Header Specification*" (DS51292 or online help). Then choose whether or not to use a header. Click **Next>** when done.

Note:	You can select a header later (if one is available) using the Project Properties window.

FIGURE 4-3: PROJECT WIZARD – SELECT HEADER



4.2.4 Step 4: Select Tool

Step 4 involves selecting the tool.

Tool support for the selected device is signified by the colored circles (lights) in front of the tool name. If you cannot see the colors, mouse over a light to pop up text about support.

Light	Color	Support
●	Green	Full (Implemented and fully tested)
●	Yellow	Beta (Implemented but not fully tested)
●	Red	None

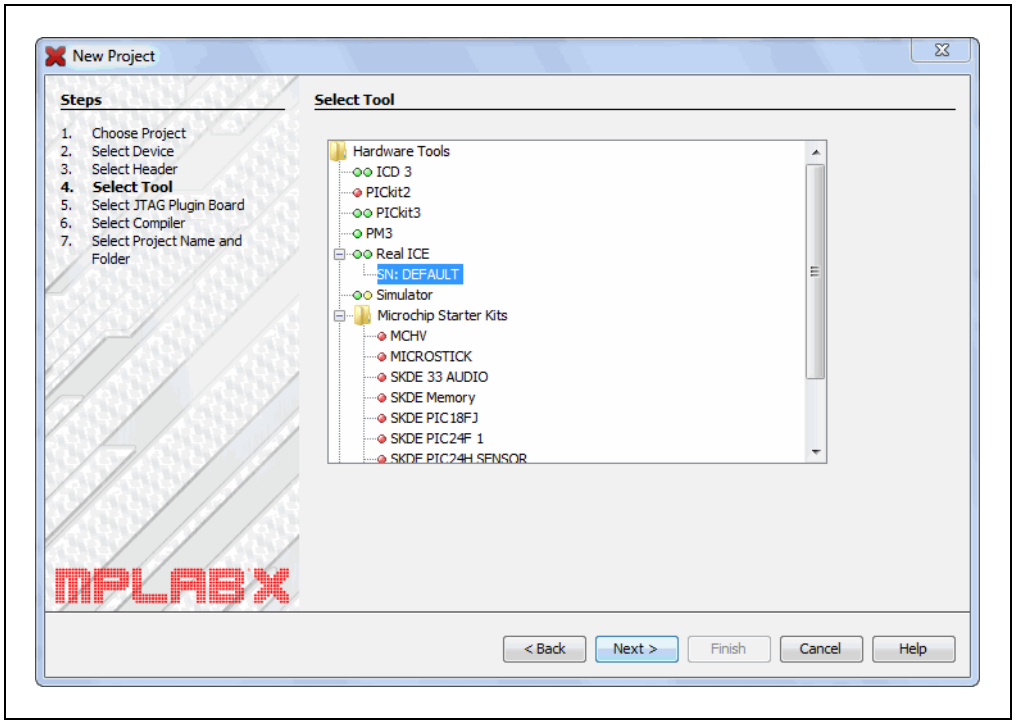
For some tools, there are two lights next to the tool name, where the *first light* is the left-most light and the *second light* is to the right of the first.

Light No.	Debug Tools	Simulator
1	Debugger Support	Core (Instruction Set) Support
2	Programmer Support	Peripheral Support

For the hardware tools, you will notice that a serial number (SN) is specified below any tool that is connected to your computer. This allows you to select from several connected hardware tools.

Select your tool and then click **Next>**.

FIGURE 4-4: PROJECT WIZARD – SELECT TOOL



4.2.5 Step 5: Select Plug-In Board

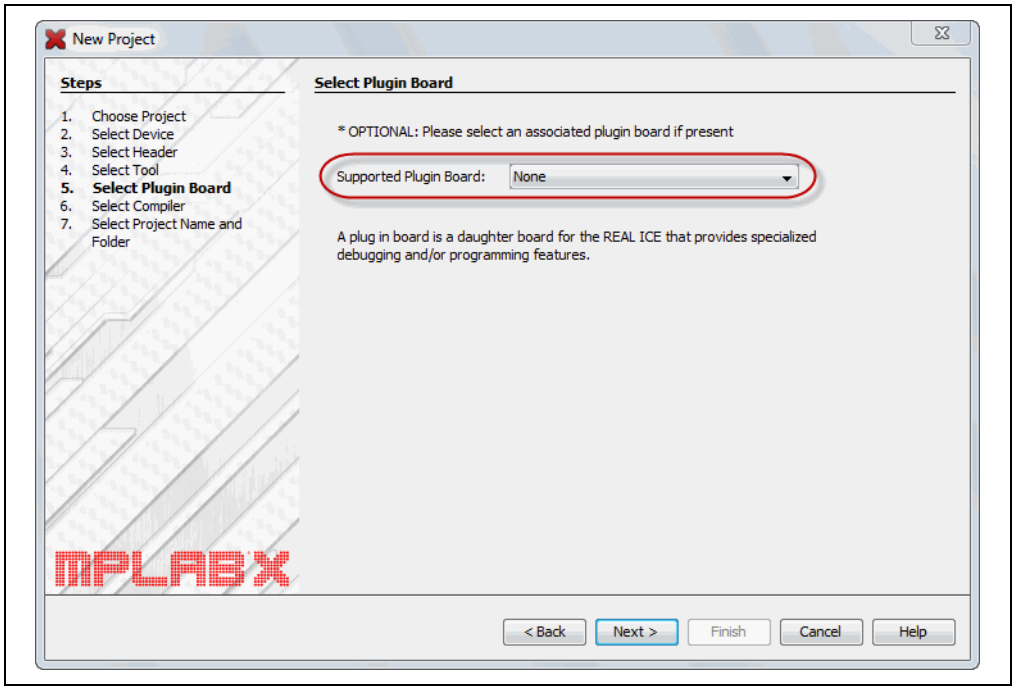
For the MPLAB REAL ICE in-circuit emulator, you may specify a plug-in board to use. A plug-in board is the circuit board that is inserted into the emulator's driver board slot.

TABLE 4-1: EMULATOR PLUGIN BOARDS

Supported Plugin Board	Board Description
None	Standard Communications driver board
None	High-Speed Communications driver board
JTAG Driver Board	JTAG Adapter board
Power Monitor Board	Power Monitor board (also inserts into logic probe connector)

Select your tool and then click **Next>**.

FIGURE 4-5: PROJECT WIZARD – SELECT PLUGIN



4.2.6 Step 6: Select Compiler

Step 6 involves selecting the language tool, either a C compiler or assembler. Again, the colored circle (light) in front of the compiler name signifies the device support level. Mouse over for text.

> **Note:** If you do not see your language tool listed, look under *Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Embedded** button, **Build Tools** tab, to ensure MPLAB X IDE can find the tool. If your tool is listed, then your project device may not be supported by your tool. Consider selecting or installing another language tool that supports the device.

TABLE 4-2: MICROCHIP LANGUAGE TOOLS

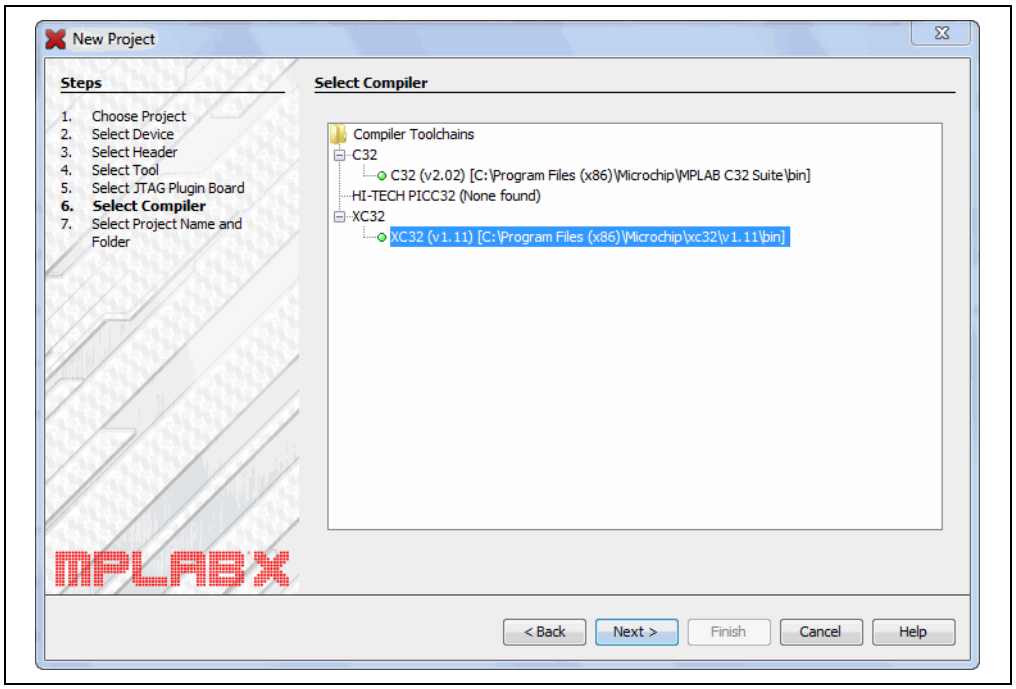
Toolchain	Full Name
8-Bit Device Language Tools	
MPASM*	MPASM Assembler, MPLINK Object Linker and Utilities
C18*	MPLAB C Compiler for PIC18 MCUs
HI-TECH PICC	HI-TECH C Compiler for PIC10/12/16 MCUs
HI-TECH PICC18	HI-TECH C Compiler for PIC18 MCUs
XC8	MPLAB XC8 C Compiler
16-Bit Device Language Tools	
ASM30**	MPLAB Assembler, Object Linker and Utilities for PIC24 MCUs and dsPIC DSCs
C30	MPLAB C Compiler for PIC24 MCUs and dsPIC DSCs
C24	MPLAB C Compiler for PIC24 MCUs (subset of C30)
dsPIC	MPLAB C Compiler for dsPIC DSCs (subset of C30)
HI-TECH DSPICC	HI-TECH C Compiler for PIC24 MCUs and dsPICDSCs
XC16	MPLAB XC16 C Compiler
32-Bit Device Language Tools	
C32	MPLAB C Compiler for PIC32 MCUs
HI-TECH PICC32	HI-TECH C Compiler for PIC32 MCUs
XC32	MPLAB XC32 C Compiler
* Most compilers come with an assembler, linker and utilities. MPLAB C18 is supported by MPASM. ** No longer included with MPLAB X IDE in v1.30. Please use an assembler that comes with one of the 16-bit compilers.	

For more on each language tool, consult the language tool documentation.

For third party language toolchains (CCS, etc.), see the “Readme for Third Party Tools.htm” file on the Start Page, “Release Notes and Support Documentation”.

Select your tool and then click **Next>**.

FIGURE 4-6: PROJECT WIZARD – SELECT LANGUAGE TOOL



4.2.7 Step 7: Select Project Name and Folder

Step 7 involves selecting the project name, location and other project features. When you are done, select **Finish** to complete new project creation.

Project Name, Location and Folder

Enter the project name. By default, the name will be appended with `.X`. This is not required and simply a convention. You can delete this if you wish in the “Project Folder” text box.

Browse to a folder location. You can create a new project folder if you need one. By default, projects will be placed in:

- Windows XP – `C:\Documents and Settings\`*`UserName`*`\MPLABXProject`
- Windows 7/8 – `C:\Users\`*`UserName`*`\MPLABXProjects`
- Linux – `/home/`*`UserName`*`/MPLABXProjects`
- Mac – `/Users/`*`UserName`*`/MPLABXProjects`

However, you may choose to put them in your own location.

Main Project

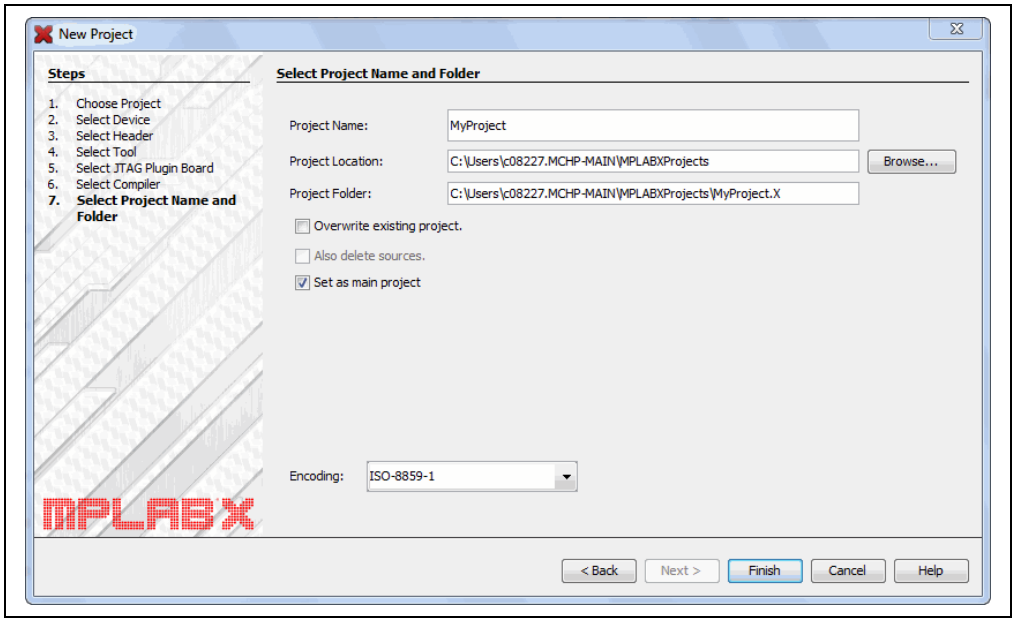
Check “Set as main project” to specify this project as your main project.

Encoding

Select the encoding for the project. The default is ISO-8859-1 (Latin 1) character set.

This selection will specify the code syntax coloring, which can be edited under *Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Fonts and Colors** button, **Syntax** tab.

FIGURE 4-7: PROJECT WIZARD – SELECT PROJECT NAME AND FOLDER

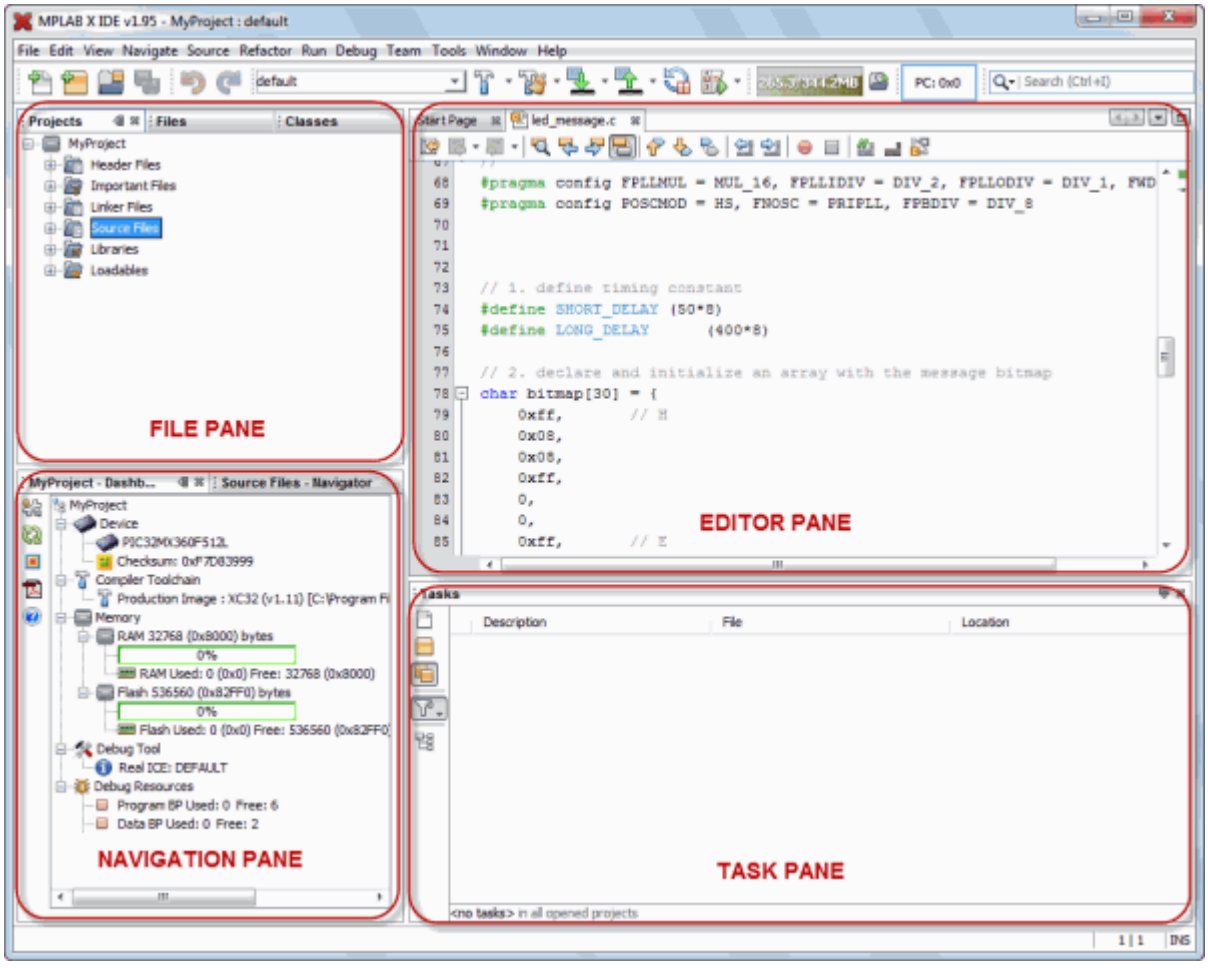


4.3 VIEW CHANGES TO THE DESKTOP

Once you have created your project, several panes will open in the IDE.

- **File pane** – pane with four tabbed windows:
  - The Projects window displays the project tree with files grouped by category.
  - The Files window displays the project files according to the folder organization on your computer.
  - The Classes window displays any classes and their functions, variables and constants in the code. Double click on an item to see its declaration.
  - The Services window displays any services available to use for code development.
- **Navigator pane** – displays information on the symbols and variables in the file selected in the File pane.
- **Editor pane** – for viewing and editing project files. The Start Page also is visible here.
- **Task pane** – displays task output from building, debugging, or running an application.

FIGURE 4-8: MPLAB X IDE DESKTOP



If you double click on any file name in the File pane, the related file will open in the Editor pane under a tab next to the Start Page. To close the tab, click on the “x” next to the file name.

Right click on the project name in the File pane, Projects window, to view the pop-up (context) menu. Do the same for the project’s subfolders.

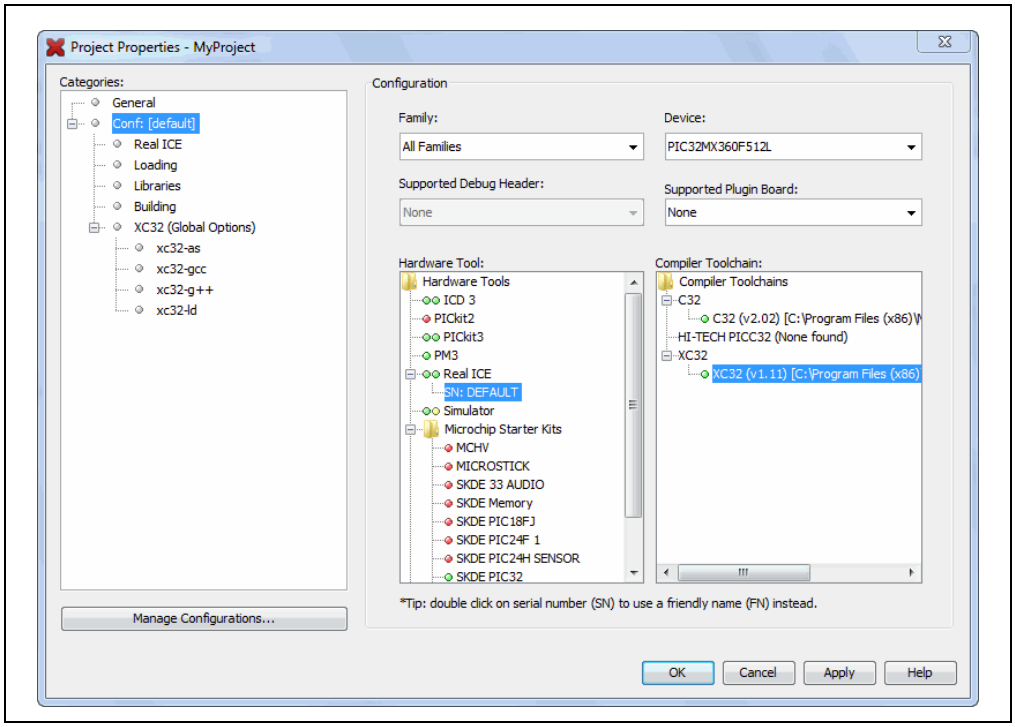
4.4 VIEW OR MAKE CHANGES TO PROJECT PROPERTIES

Once a project has been created, you can view or change the project properties in the Project Properties dialog. Access this dialog by either:

- Right clicking on the project name in the Projects window and selecting “Properties”.
- Clicking on the project name in the Projects window and then selecting *File>Project Properties*.

Click the “Conf: [default]” category to reveal the general project configuration, such as the project device, related debug/programmer tool, and language tool. To change any of these items, refer to steps 4 and 5 of **Section 4.2 “Create a New Project”**.

FIGURE 4-9: PROJECT PROPERTIES DIALOG



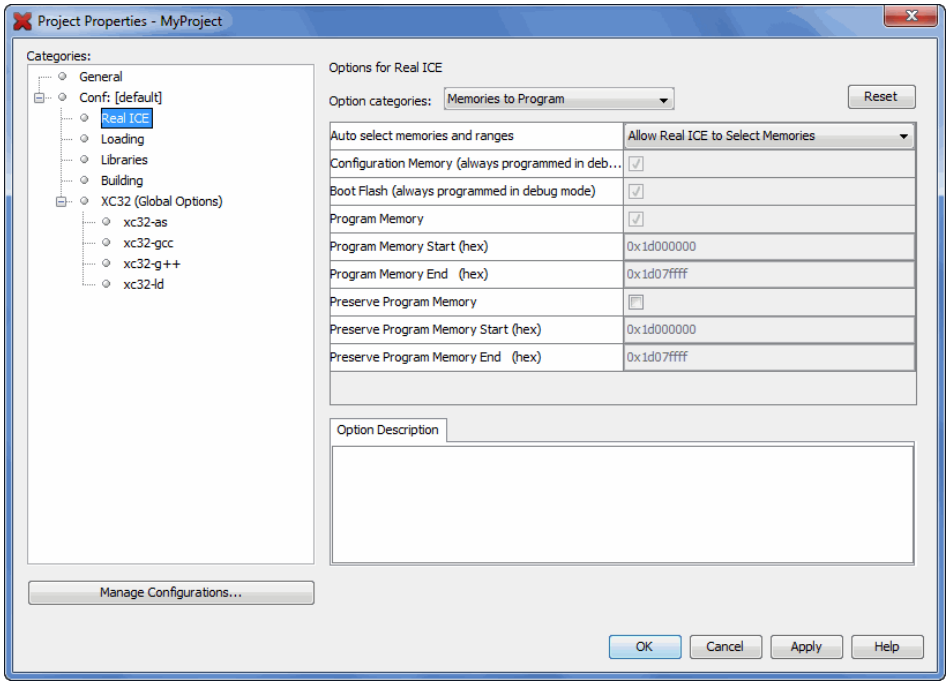
4.5 SET OPTIONS FOR DEBUGGER, PROGRAMMER OR LANGUAGE TOOLS

Additionally, you set options for your tools in the Project Properties dialog.

To set up or change debugger/programmer tool options:

Click on your hardware tool or simulator (beneath Conf:[default]) to see related set up options. For more on what these options mean, see your tool documentation.

FIGURE 4-10: TOOL SETUP PAGE

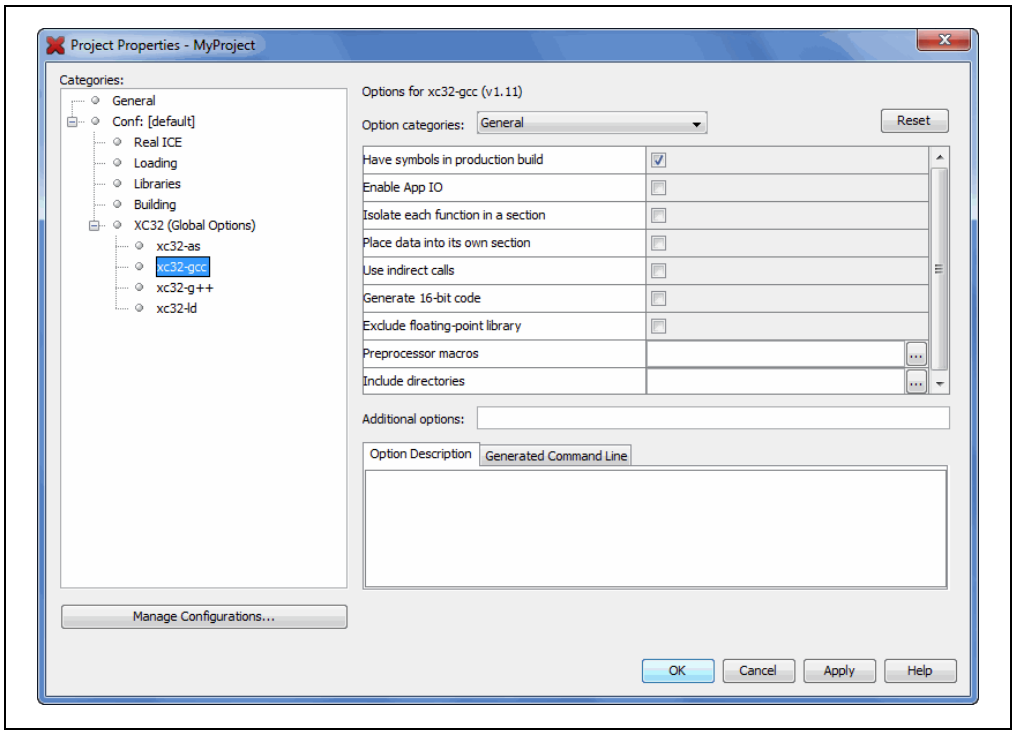


To set up or change language tool options:

Click on your language tool to see related setup options. For more on what these options mean, see your language tool documentation.

See the NetBeans help topic *C/C++/Fortran Development>Working with C/C++/Fortran Projects>Setting Project Properties* for additional help.

FIGURE 4-11: LANGUAGE TOOL SET UP PAGE



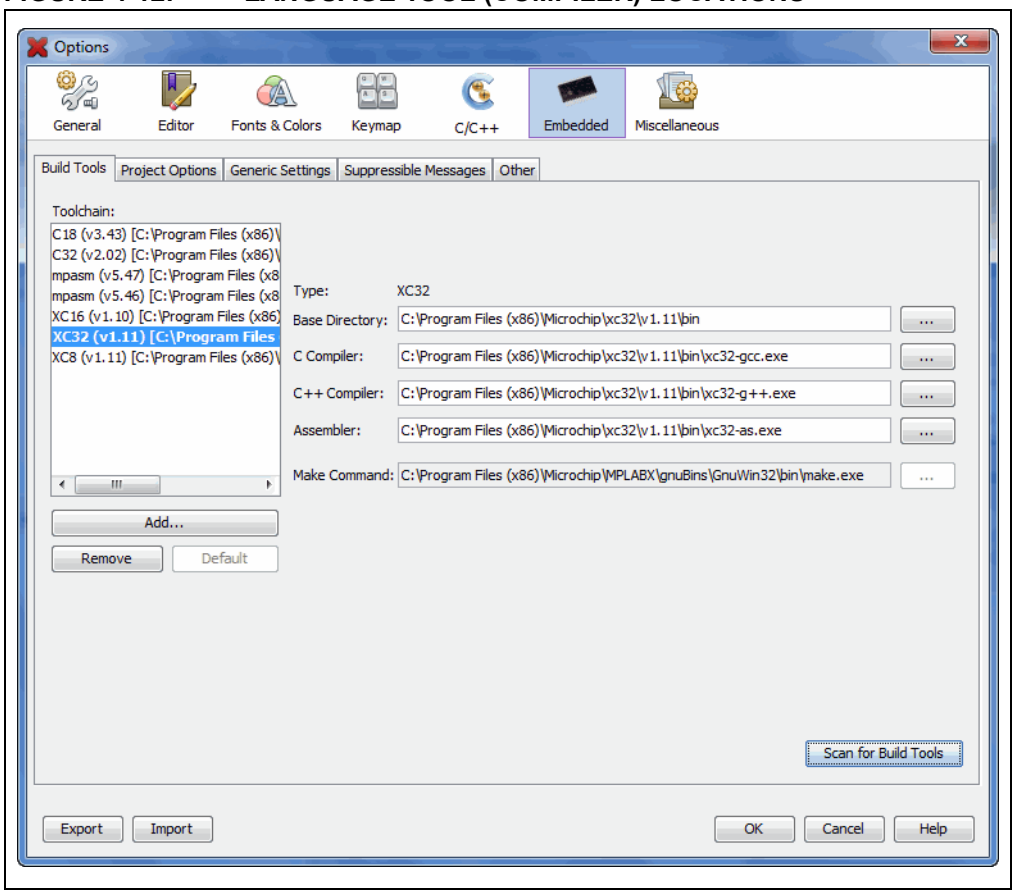
4.6 SET LANGUAGE TOOL LOCATIONS

To see what language tools are available to MPLAB X IDE and view or change their paths:

- **For Mac OS X** – Access the build tools from: *mplab_ide>Preferences>Embedded>Build Tools* from the main menu bar.
- **For Other OS –** Access the build tools from: *Tools>Options>Embedded>Build Tools*.

The window should automatically populate with all installed toolchains.

FIGURE 4-12: LANGUAGE TOOL (COMPILER) LOCATIONS



4.6.1 Add or Change a Toolchain

If you do not see your tool listed under Toolchains, try the following:

- **Scan for Build Tools** – Scan the environment path and list the language tools installed on the computer.
- **Add** – Manually add the tool to the list by entering the path to the directory containing the tool executable(s), i.e., base directory. Typically, this is the `bin` subdirectory in the tool installation directory.

If you have more than one version of a compiler available, select one from the list.

To change the path of a tool, enter the new path or browse for it.

4.6.2 About Toolchain Paths

MPLAB X IDE searches for toolchains under default installation paths and the PATH environmental variable. An example default path for MPLAB XC16 under a Windows 64-bit operating system is:

```
C:\Program Files (x86)\Microchip\xc16\bin
```

When you install a compiler, you have the choice to:

1. install at the default location or at a different location
2. add the location to the PATH

If you are installing at a location different from the default, you should add that location to the PATH.

If you do not choose to have the installer add a new location to the PATH, you can point MPLAB X IDE to the compiler location in *Tools>Options*, **Embedded** button, **Build Tools** tab, by clicking **Add**.

Finally, you can manually add the new location to the PATH variable.

4.7 SET OTHER TOOL OPTIONS

In addition to the build paths, you may set up other options. Select from the following tabs in the Options window, Embedded category:

- Project Options – Set project-related options, such as make options and whether paths are defaulted to relative or absolute (**Section 12.12.2 “Project Options Tab”**).
- Generic Settings – Set up the log file and other project features (**Section 12.12.3 “Generic Settings Tab”**).
- Suppressible Messages – Select messages to be suppressed (**Section 12.12.4 “Suppressible Messages Tab”**)
- Other – Edit the lists of accepted file extensions for C source files and header files (**Section 12.12.6 “Other Tab”**).

4.8 CREATE A NEW FILE

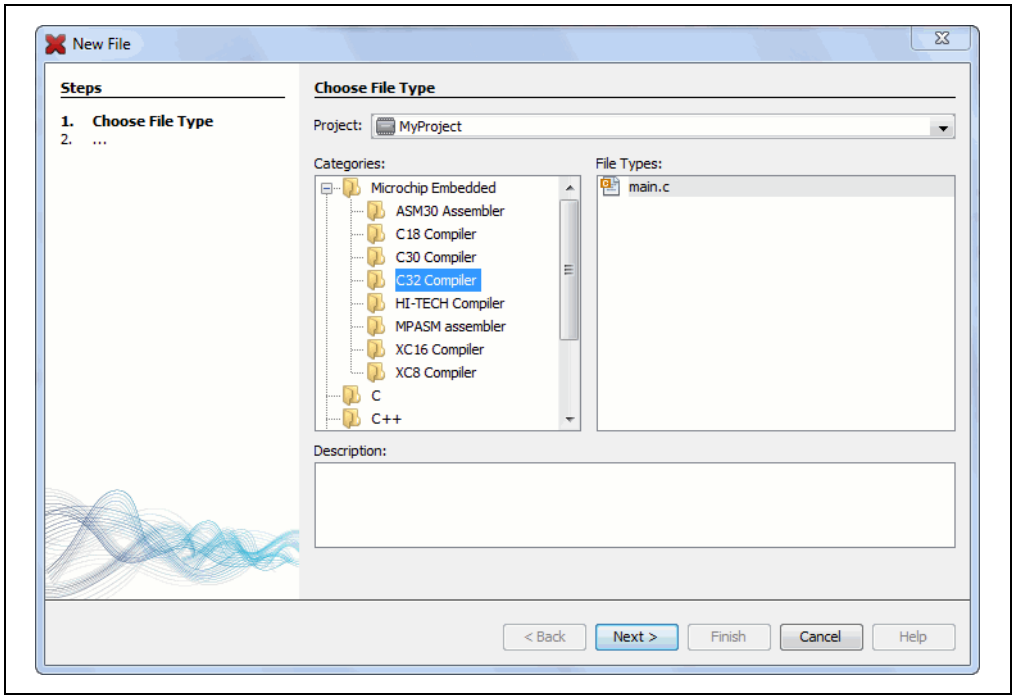
New files can be created by doing one of the following:

- Selecting *File>New File* (or Ctrl+N)
- Right clicking on the project in the Project/File window and selecting *New>Other*
- Right clicking on a logical folder (e.g., Source Files) in the Project/File window and selecting *New>Other*

A New File Wizard with launch to guide you through new file set up.

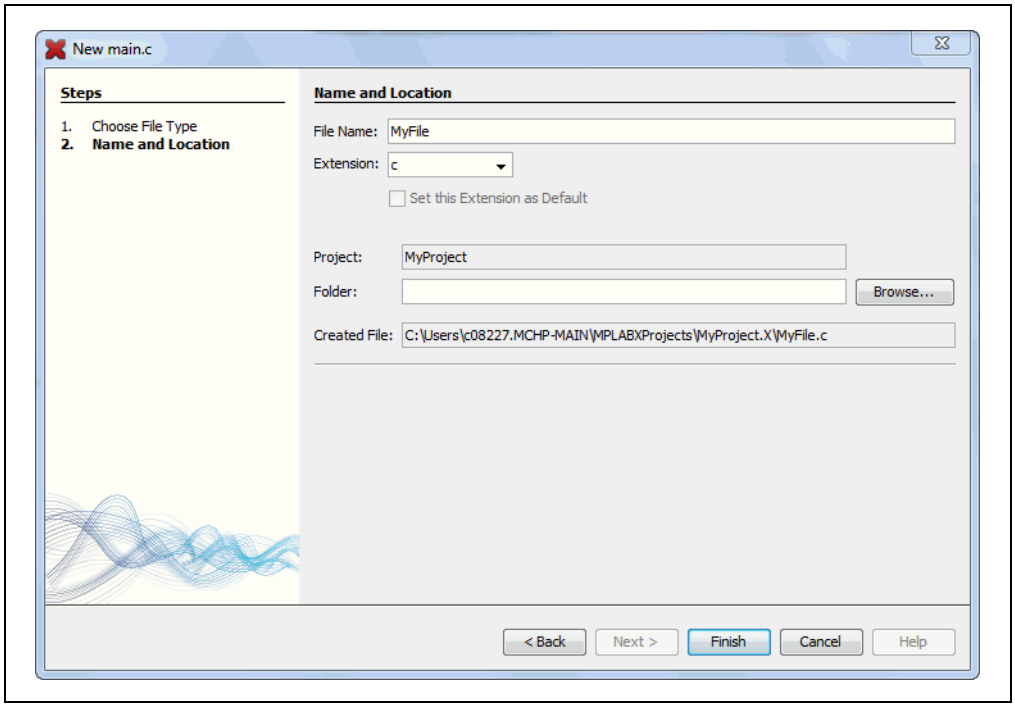
Step 1. Choose File Type: Choose the file category by expanding "Microchip Embedded" to find an appropriate selection. Then select a file type.

FIGURE 4-13: FILE WIZARD – CHOOSE CATEGORY AND TYPE



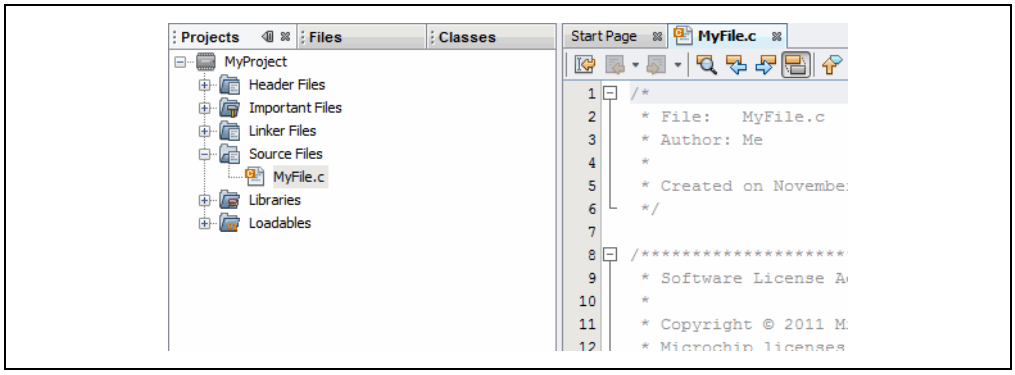
Step 2. Name and Location: Name the file and place it in the project folder.

FIGURE 4-14: FILE WIZARD – CHOOSE ASSOCIATED PROJECT



The file will appear in the File pane under the project specified and a tab with the file's name will appear in the Editor pane. Enter your file content under this tab. The text in the file will have syntax color based on its file type.

FIGURE 4-15: NEW FILE – MYFILE.C



4.9 ADD EXISTING FILES TO A PROJECT

Existing files can be added to a project by doing one of the following:

- Right clicking on the project in the Project/File window and selecting "Add Existing Item"
- Right clicking on a logical folder (e.g., Source Files) in the Project/File window and selecting "Add Existing Item"

4.9.1 Files in Project Folder

When adding a file, you can choose whether to add it as:

- Auto – Let MPLAB X IDE decide how best to locate the file.
- Relative – Specify the file location relative to the project. (Most portable project.)
- Absolute – Specify the file location by an absolute path.
- Copy – Copy the specified file to the project folder.

The file will appear in the File pane under the project specified and a tab with the file's name will appear in the Editor pane.

4.9.2 Files Outside of Project Folder

When adding a source file that is not in the project folder, add the file as "Relative". This will create an external folder (`_ext`) so the project can find the file when building.

To make use of navigation features such as `//TODO` and file context menus, you must tell the project where the files are located. To do this:

1. In the Projects window, right click on the project name and select "Properties".
2. Under "Categories", click "General".
3. Next to "Source Folders", click **Add**.
4. Browse to the path to the external file(s) you have added to the project. Select **Select**.
5. Click **Apply** or **OK**. Then rebuild the project.

When you import an MPLAB IDE v8 project, the source files are a not in the project folder. Use *File>Import>MPLAB IDE v8 Project* to automatically import the files.

4.10 EDITOR USAGE

To create new code or change existing code, use the NetBeans editor. General information on this editor is available from the table of contents under the NetBeans help topic *IDE Basics>Basic File Features*. C compiler information regarding the editor is available from the table of contents under the NetBeans help topic *C/C++/Fortran Development>C/C++/Fortran Project Basics>Navigating and Editing C/C++/Fortran Source Files*. A list of features is available under **Chapter 7. "Editor"**.

To use editor features, go to the:

- Edit menu (see **Section 11.2.2 "Edit Menu"**)
- Editor toolbar located at the top of each file's Editor window

FIGURE 4-16: EDITOR TOOLBAR

- Window right click (context) menu.

To set editor properties:

1. Select *Tools>Options* (*mplab_ide>Preferences* for Mac OS X) to open the Options dialog.
2. Click on the **Editor** button. Then click on a tab to set up Editor features.
3. Click on the **Fonts and Colors** button. Then click on a tab to set up color options.

To navigate through files and format your code:

1. Select items from the Navigate menu to navigate inside and outside of files.
2. Select items from the Source menu to format, comment, and auto-complete code.

To use code folding:

For details on code folding, see **Section 7.4 "Code Folding"**.

4.11 ADD AND SET UP LIBRARY AND OBJECT FILES

You can reference library files to be used by the linker in the following locations:

- the Libraries folder in the Projects window
- the Project Properties dialog

Additional library file set up may be done in the language tool librarian.

4.11.1 Libraries Folder

In the Projects window, right click on the Libraries folder to see these options:

- **Add Library Project**

  Make a project that produces a library as its output required for your project, i.e., establish a dependency. For more on this, see the NetBeans help topic *C/C++/Fortran Development>C/C++/Fortran Project Basics>Building C/C++/Fortran Projects>Creating Dependencies Between C/C++/Fortran Projects*.

- **Add Library/Object File**

  Add an existing library file or pre-built (object) file to the project.

- **Properties**

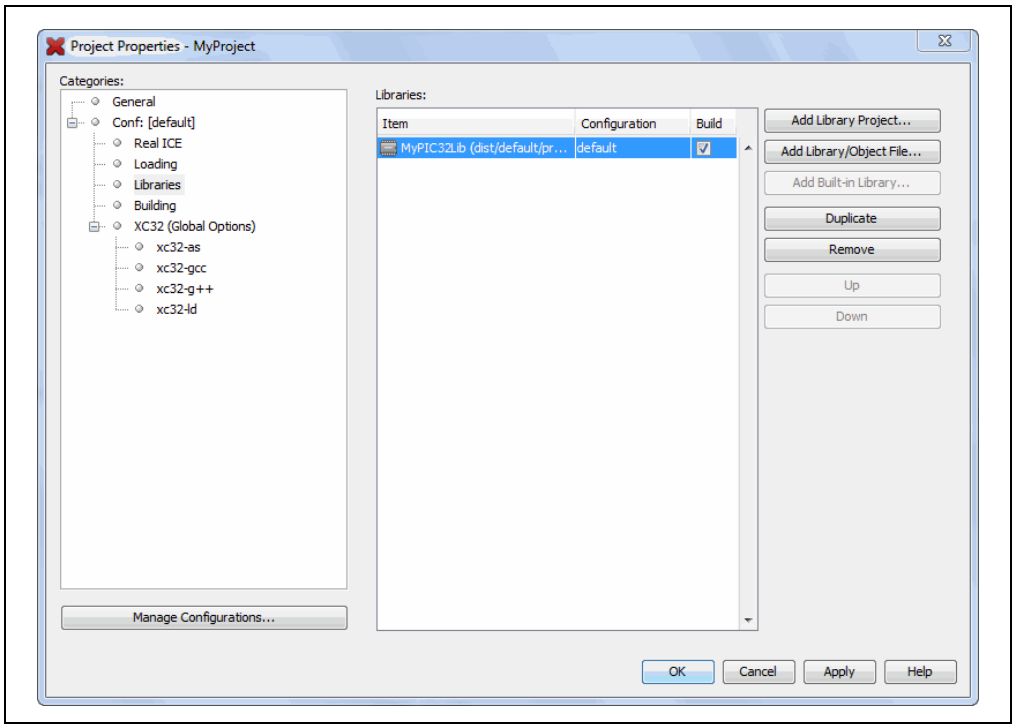
  Open the project's Properties window to select additional options (see next section).

4.11.2 Project Properties Window: Libraries Category

Open the Project Properties window and click on the "Libraries" category. Any files added to the Libraries folder in the Projects window will be visible here. You may add additional files and manage these files using the buttons on the right.

The order of the files here determines their link order.

FIGURE 4-17: MANAGE LIBRARY/OBJECT FILES AND SET UP LINK ORDER

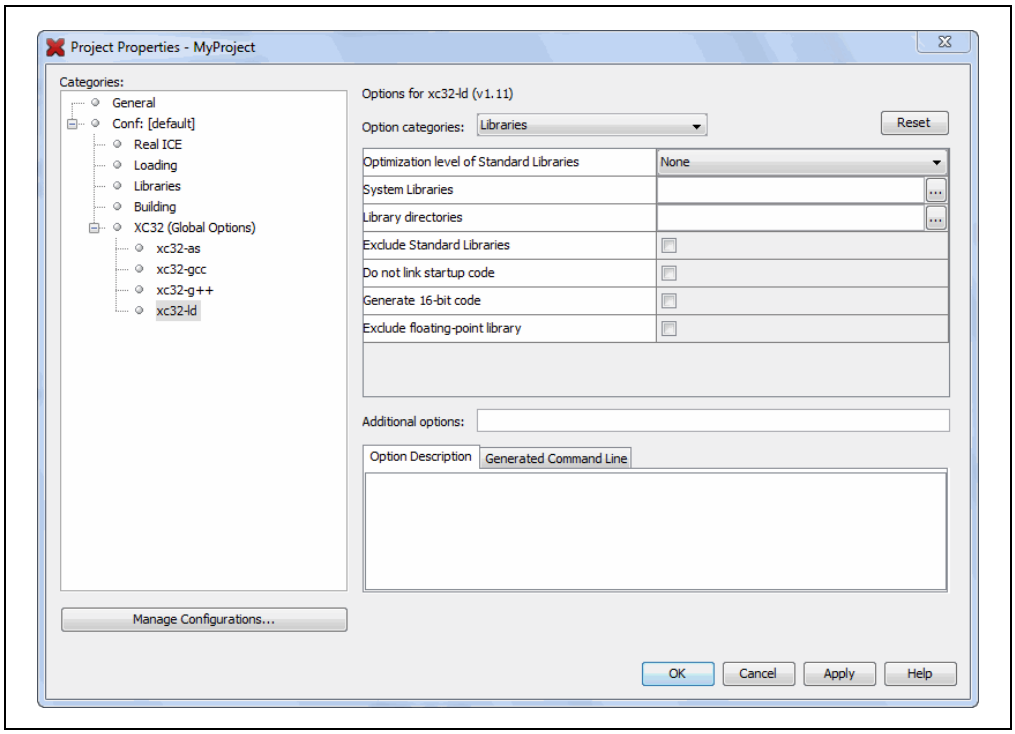


4.11.3 Project Properties Window: Librarian Category

Open the Project Properties window and click on the librarian under your language tool category. Consult your language tool documentation to determine the executable name of your librarian.

In the left pane, click on the name of the linker; then in the right pane, select the option “Libraries”. Choose library options from this list.

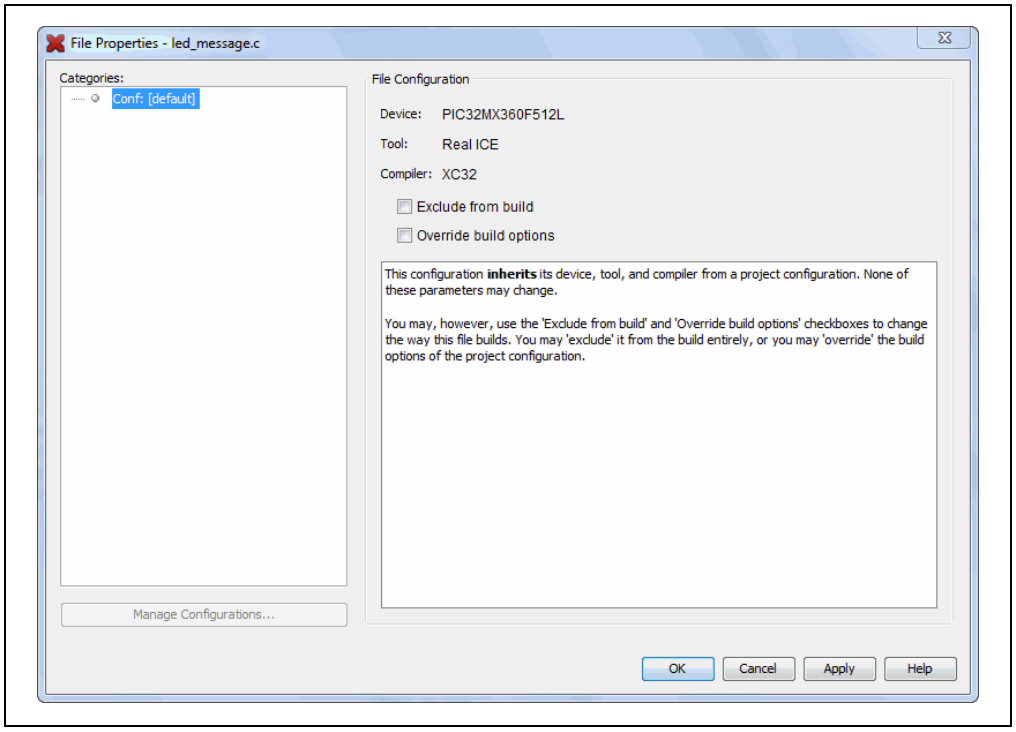
FIGURE 4-18: SET UP LIBRARIES OPTIONS IN THE LIBRARIAN



4.12 SET FILE AND FOLDER PROPERTIES

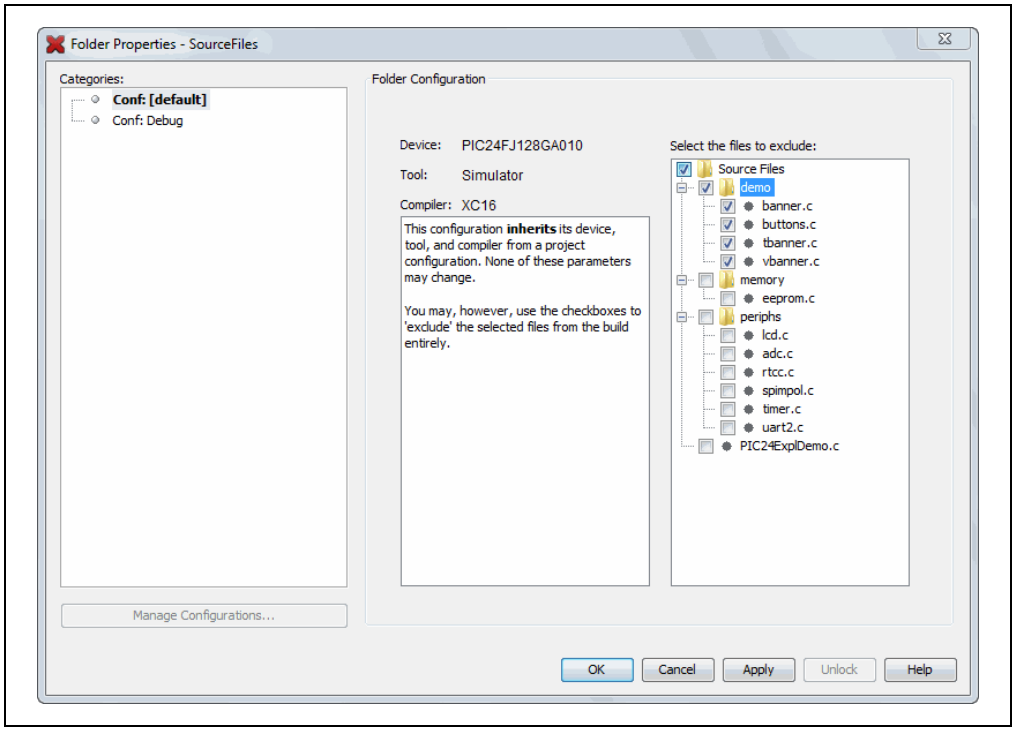
A project file can be built differently from other files in the project using the File Properties dialog. Access this dialog by right clicking on the file name in the Projects window and selecting "Properties". Select to exclude the file from the project build or override the project build options with the ones selected here. To override build options, check the checkbox and then click "Apply" to see selection options appear under "Categories".

FIGURE 4-19: FILE PROPERTIES



An entire (virtual) folder can be excluded from the build process by right clicking on the folder name in the Projects window and selecting “Properties”. Select to exclude folders or individual files from the build. Selecting a folder that is above other folders will select the entire contents of that folder. You may then deselect files or other folders as you wish.

FIGURE 4-20: FOLDER PROPERTIES

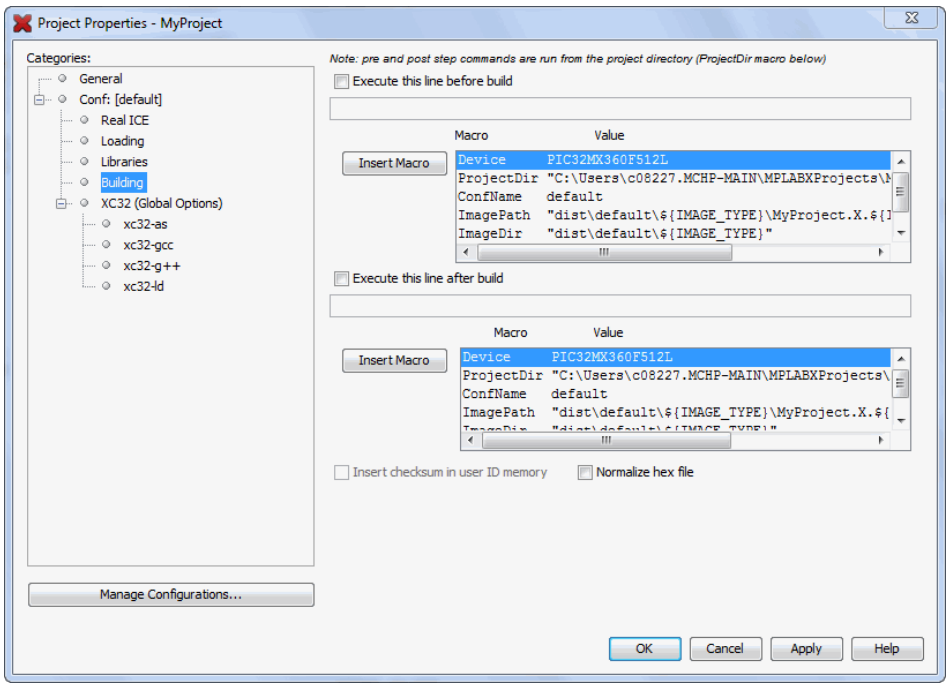


4.13 SET BUILD PROPERTIES

Before building the project, you may want to set up additional build properties:

- Execute this line before/after the build
- Insert checksum in user ID memory
- Normalize hex file

FIGURE 4-21: MAKE OPTIONS BUILD PROPERTIES



4.13.1 Execute this line before/after the build

Type in a command to be executed at the very beginning or at the very end of the build process. These commands are inserted into the `nbproject/Makefile-$CONF.mk` file and allow you to customize the build process. If you need to refer to some of the project-related items (like the image name) in the script or program you are calling, use the supplied macros.

You can type the macros yourself or click **Insert Macro** to copy the macro name into the current position in the edit box. Commands are run in the make process with the current directory being set to the MPLAB X IDE project directory. The project directory is defined as the project that contains the `nbproject` folder

TABLE 4-3: MACROS

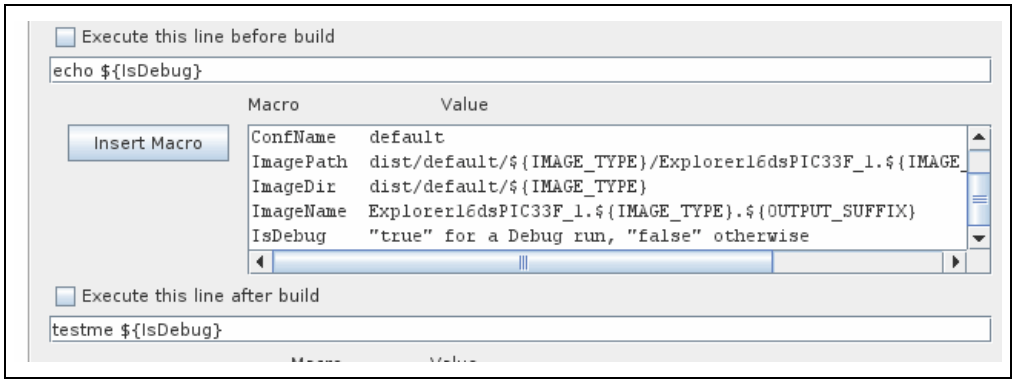
Name*	Function
`Device`	Device for the current-selected project configuration.
`IsDebug`	“true” for a Debug Run; “false” otherwise.
`ProjectDir`	Location of the project files on the PC.
`ConfName`	Name of the currently-selected project configuration.
`ImagePath`	Path to the build image.
`ImageDir`	Directory containing the build image.
`ImageName`	Name of the build image.

* Additional macros may be available, depending on the compiler used.

EXAMPLE 4-1: EXECUTE A PROCESS ONLY DURING DEBUG

On the window shown below, click the `IsDebug` macro to select it, then click **Insert Macro** to insert it into a script.

FIGURE 4-22: BUILD PROPERTIES - PRE/POST BUILD



Check the value of `$1` (Linux or Mac OS) or `%1` (Windows OS) in the script being run.

Bash Code Example

```
#!/bin/bash
echo is $1
if [ "$1" == "true" ]; then
  echo We are in debug
else
  echo We are in production
fi
```

Batch Code Example

```
@echo off
if "%1" == "true" goto debug
:production
@echo We are in production
goto end
:debug
@echo We are in debug
:end
```

4.13.2 Insert checksum in user ID memory

If supported by the device, check to use the checksum number generated after a build as the user ID.

4.13.3 Normalize hex file

A hex file is normalized when line addresses are in incremental order and data is buffered to be one size (16 bytes) where possible.

For example, the following lines (records) are from the file `myfile.hex`, output from the MPLAB XC16 C compiler/linker:

```
:0801f800361e0000361e000057
:020000040000fa
:10020800361e0000361e0000361e0000361e000096
```

When this file is normalized, the file data looks like this:

```
:10022800361E0000361E0000361E0000361E000076
:10023800361E0000361E0000361E0000361E000066
:10024800361E0000361E0000361E0000361E000056
```

which is of the format:

```
:bbaaaarrdd...ddcc
```

where:

`:`	Start record
`bb`	Byte count
`aaaa`	Address
`rr`	Record type
`dd`	Data
`cc`	Checksum

In MPLAB X IDE, the application HEXMATE is used to normalize Intel hex files. When the "Normalize hex file" option is checked, the following is called after a build:

`hexmate` *`myfile`*`.hex -o`*`myfile`*`.hex`

Essentially HEXMATE unpacks the entire hex file and arranges the data at the addresses specified by the hex file. It then repackages the resulting memory image into a new hex file. In the resulting file, all the data is ascending order and is contiguous. If there is a gap in the addresses, then there will also be a gap in the output file (there is no filling of unused addresses).

For more information on using HEXMATE, see the `manual.pdf` in the `docs` folder of the installed MPLAB XC8 C compiler.

A normalized hex file is useful for programming code (such as a bootloader) over a serial connection, as the variation of record bytes is minimized.

4.14 BUILD A PROJECT

For MPLAB X IDE, it is not necessary to build the project first and then run or debug. Building is part of the run and debug processes. For initial development or major changes, however, you may want to make sure that the project builds before attempting to run or debug.

To build a project:

- In the Projects window, right click on the project name and select “Build”. You may also select “Clean and Build” to remove intermediary files before building.
- Click on the “Build Project” or “Clean and Build Project” toolbar icon.

Build progress will be visible in the Output window.

Available build functions are:

TABLE 4-4: BUILD OPTIONS ON TOOLBAR BUTTONS

Button Icon	Function	Description
	Build Project	Make all the files in the project.
	Build for Debugging	Make all the files in the project and add a debug executive to the built image.
	Clean and Build	Remove previous build files and make all the files in the project.
	Clean and Build for Debugging	Remove previous build files and make all the files in the project. Add a debug executive to the built image.

To view errors in the Output window:

1. Right click in the Output window and select “Filter”.
2. In the Filter dialog, check “Match” and enter “: error” to show only errors in the Output window that stopped the build.
3. Toggle the filter on and off using <Ctrl>-<G>.

See you language tool documentation for a discussion of errors.

To view checksum information:

- Open the Dashboard window (see **Section 5.17 “View the Dashboard Display”**) to see the checksum after a build.

For more information on building your project, also see the NetBeans help topic *C/C++/Fortran Development>C/C++/Fortran Project Basics>Building C/C++/Fortran Projects*.

4.15 RUN CODE

Once the code builds successfully, you can attempt to run the application.

4.15.1 How Run Works

On clicking the **Run Project** toolbar button:

1. A build if the make process determines it is necessary.
2. For in-circuit debuggers/emulators and programmers, the target device will automatically be programmed with the image (no debug executive) and the device will then be released to run, i.e., no breakpoints or other debug features will be enabled. To hold a device in Reset after programming, click "Hold in Reset" instead of "Run Project".

Hold in Reset

3. For simulators, the application will simply execute with no debugging capability.

Run progress will be visible in the Output window.

4.15.2 Run Considerations

When running your program, consider:

1. MPLAB X IDE operation connects to the hardware tool at runtime (Run or Debug Run). This means that settings made in MPLAB X IDE will only be passed to the tool at runtime. Settings changed during a halt will be updated only when runtime is again initiated.

   To always be connected to the hardware tool (like MPLAB IDE v8), see *Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Embedded** button, **Generic Settings** tab, "Maintain active connection to hardware tool" checkbox.
2. See **Section 4.23 "Program a Device"**.

4.15.3 Running Your Application Code

1. In the Projects window, select your project or make it the main project (right click on the project and select "Set as main project".)
2. Click on the "Make and Program Device" icon (or select *Run>Run Project*) to run your program.

Make and Program Device

Run progress will be visible in the Output window.

4.16 DEBUG RUN CODE

If your code does not run successfully, you will need to debug it. Running the code in debug mode is called a Debug Run.

4.16.1 How Debug Run Works

On clicking the “Debug Project” toolbar button:

1. A build will occur if the make process determines it is necessary.
2. For in-circuit debuggers/emulators, the target device or header will automatically be programmed with the image (including the debug executive) and a debug session will be started.
3. For simulators, a debug session will be started.

Debug Run progress will be visible in the Output window.

4.16.2 Debug Macros Generated

MPLAB X IDE generates debug macros for use with Microchip language tools. Macros passed to Microchip compilers and assemblers are shown in Table 4-5.

TABLE 4-5: MICROCHIP TOOLS DEBUG MACROS

Macro Name	Assoc. Tool	Function
`__DEBUG`	All	Specifies that this is a debug build.

`__MPLAB_REAL_ICE__`
`__MPLAB_ICD3__`
`__MPLAB_PK3__`
`__MPLAB_PICKIT2__`
	XC8	Specifies the hardware debug tool in use. The format is `__MPLAB_`*xxx*`__`, where *xxx* represents the hardware tool specifier.

`__MPLAB_DEBUGGER_REAL_ICE`
`__MPLAB_DEBUGGER_ICD3`
`__MPLAB_DEBUGGER_PK3`
`__MPLAB_DEBUGGER_PICKIT2`
	XC16, XC32, MPASM	Specifies the hardware debug tool in use. The format is `__MPLAB_DEBUGGER_`*xxx*, where *xxx* represents the hardware tool specifier.

`__MPLAB_DEBUGGER_PIC32MXSK`	XC32	Specifies the starter kit in use.

You may use these macros in your own code. For example:

```
#ifdef __DEBUG
  fprintf(stderr,"This is a debugging message\n");
#endif
```

4.16.3 Debug Considerations

When debugging your code, consider:

1. You will need to be in a debug session (debug mode) to activate many debugging features. For example, to view variable values in watch or memory windows.
2. MPLAB X IDE operation connects to the hardware tool at runtime (Run or Debug Run). This means that settings made in MPLAB X IDE will be passed to the tool only at runtime. Settings changed during a halt will only be updated when runtime is again initiated.

   To always be connected to the hardware tool (like MPLAB IDE v8), see *Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Embedded** button, **Generic Settings** tab, “Maintain active connection to hardware tool” checkbox.
3. For some applications you may need to break down the debug steps for independent execution. To do this, use the steps under *Debug>Discrete Debugger Operation*.

4.16.4 Debugging Your Application Code

To debug your application code:

1. In the Projects window, select your project or make it the main project (right click on the project and select "Set as main".)
2. Click on the "Debug Project" icon (or select *Debug>Debug Project* or *Debug>Step Into*) to begin a Debug Run.

Debug Project

To halt your application code:

- Click on the "Pause" icon (or select *Debug>Pause*) to halt your program execution.

To run your code again:

- Click on the "Continue" icon (or select *Debug>Continue*) to start your program execution again.

To end execution of your code:

- Click on the "Finish Debugger Session" icon (or select *Debug>Finish Debugger Session*) to end your program execution.

The difference between Run and Debug Run will become apparent when working with debug features, beginning with **Section 4.17 "Control Program Execution with Breakpoints"**.

To launch the debugger:

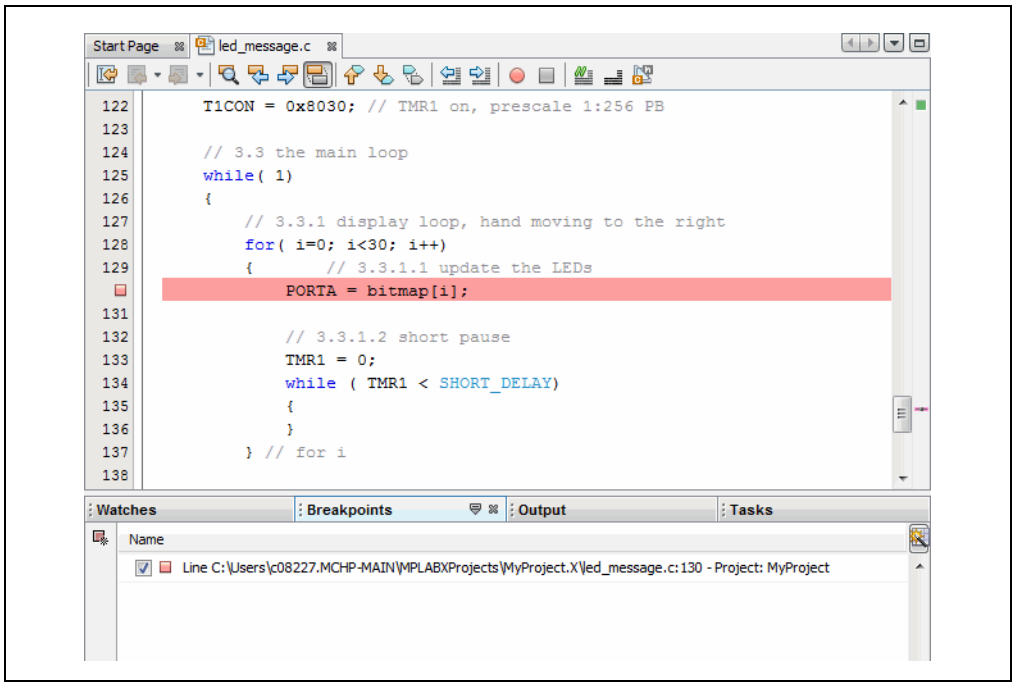
If your code is built for debugging and you simply want to launch the debug tool, you can do so by selecting the down arrow next to the "Debug Project" icon and selecting "Launch Debugger".

4.17 CONTROL PROGRAM EXECUTION WITH BREAKPOINTS

When debugging code, it can be useful to suspend execution at a specific location in code so that variable values can be examined. To do this, use breakpoints.

For more on breakpoints, see the NetBeans help topics under *C/C++/Fortran Development>Debugging C/C++/Fortran Applications with gdb>Setting C/C++/Fortran Breakpoints*.

FIGURE 4-23: BREAKPOINT AND BREAKPOINTS WINDOW



4.17.1 Set/Clear a Simple Breakpoint

To set a breakpoint on a line do one of the following:

- Click the left margin of the line in the Source Editor
- Press Ctrl+F8

To clear the breakpoint do one of the following:

- Repeat the step to set a breakpoint
- Select *Debug>Toggle Breakpoint*.

4.17.2 Set Breakpoints with the Breakpoint Dialog

To set a breakpoint with the New Breakpoint dialog:

1. Select *Debug>New Breakpoint*.
2. Select the breakpoint type and other options in the New Breakpoint dialog.

Options available for each breakpoint type as discussed in detail under **Section 12.4.1 "New Breakpoint Dialog"**.

4.17.3 Set/Clear Breakpoints in the Breakpoints Window

To view and toggle the breakpoint in the Breakpoints window:

1. Select *Window>Debugging>Breakpoints*.
2. Toggle the breakpoint by checking/unchecking the checkbox.

To set a breakpoint with the New Breakpoint dialog:

1. Click on the icon in the top left of the window.

4.17.4 Set a Breakpoint Sequence (Device Dependent)

A breakpoint sequence is a list of breakpoints that execute but do not halt until the last breakpoint is executed. Sequenced breakpoints can be useful when there is more than one execution path leading to a certain instruction and you only want to exercise one specific path.

To create a Breakpoint Sequence:

1. Right click on an existing breakpoint or shift click to select a group of existing breakpoints and right click on the group.
2. From the pop-up menu, go to "Complex Breakpoint" and select "Add a New Sequence".
3. Enter a name for your sequence in the dialog box and click **OK**.
4. The breakpoint(s) will appear under the new sequence.
5. To add additional existing breakpoints to the sequence, right click on the breakpoint and select *Complex Breakpoint>Add to Name*, where *Name* is the name of the sequence.
6. To add new breakpoints to the sequence, right click on the sequence and select "New Breakpoint".

To select the Sequence Order:

1. Expand on a sequence to see all items.
2. Right click on an item and select *Complex Breakpoints>Move Up* or *Complex Breakpoints>Move Down*. Sequence execution of breakpoints is bottom-up; the last breakpoint in the sequence occurs first.

To Remove a Sequence or Breakpoint:

1. Right click on the item and select "Disable" to temporarily remove the item.
2. Right click on the item and select "Delete" to permanently remove the item.

4.17.5 Set a Breakpoint Tuple (Device Dependent)

For MPLAB X IDE, a tuple represents an ANDed list of breakpoints. ANDed breakpoints can be useful when a variable is modified in more than one location and you need to break only when that variable is modified in one particular location.

Only two breakpoints can be ANDed and these must consist of one program memory breakpoint and one data memory breakpoint. Breakpoint 1 AND breakpoint 2 must occur at the same time before a program halt.

To create a Breakpoint Tuple:

1. Click on the icon in the upper left of the Breakpoints window to open the New Breakpoint dialog.
2. Create an address breakpoint. Click **OK** to add it to the Breakpoints window.
3. Repeat steps 1 and 2 to create a data breakpoint.
4. Right click on one breakpoint and select *Complex Breakpoint>Add to New Tuple*.
5. Enter a name for your tuple in the dialog box and click **OK**.
6. The breakpoint will appear under the new tuple.
7. Right click on the other breakpoint and select *Complex Breakpoint>Move to Name*, where *Name* is the name of the tuple.

To Remove a Tuple or Breakpoint:

1. Right click on the item and select “Disable” to temporarily remove the item.
2. Right click on the item and select “Delete” to permanently remove the item.

4.17.6 Breakpoint Applications

To determine the timing between breakpoints:

- Use the stopwatch (see **Section 5.14 “Use the Stopwatch”**).

To determine breakpoint resources:

- Open the Dashboard window (see **Section 5.17 “View the Dashboard Display”**) to see the number of available and used breakpoints and whether software breakpoints are supported.

4.17.7 Breakpoint Usage Considerations

Starter kits, in-circuit debuggers (including PICkit 2 and 3) and the MPLAB REAL ICE in-circuit emulator support a limited number of breakpoints. The number of breakpoints available is dependent on the device selected. To see the number of breakpoints available and keep track of the number you have used, see the Dashboard window (**5.17 “View the Dashboard Display”**).

The following MPLAB X IDE features use breakpoints to accomplish their functions:

- Step Over
- Step Out
- Run to Cursor
- Reset to Main

If you attempt to use one of these features when no breakpoints are available, a dialog will be displayed telling you that all resources are used.

4.18 STEP THROUGH CODE

Use one of the stepping functions on the Debug menu and Debug toolbar to move through code either from the beginning of code or after a breakpoint halt. Examine changes in variable values (see next section) or determine if the program flow is correct.

There are several ways to step through code:

- Step Over – Executes one source line of a program. If the line is a function call, executes the entire function then stops.
- Step Into – Executes one source line of a program. If the line is a function call, executes the program up to the function's first statement and stops.
- Step Out – Executes one source line of a program. If the line is a function call, executes the functions and returns control to the caller.
- Run to Cursor – Runs the current project to the cursor's location in the file and stop program execution.

In addition to the Editor window, you can single-step through code in the Disassembly window (**Section 5.15 "View the Disassembly Window"**) and program memory in a Memory window.

For more on stepping, see the NetBeans help topics under *C/C++/Fortran Development>Debugging C/C++/Fortran Applications with gdb>C/C++/Fortran Debugging Sessions>Stepping Through Your C/C++/Fortran Program*.

4.19 WATCH SYMBOL VALUES CHANGE

Watch the values of symbols that you select change in the Watches window. Determining if these values are as expected during program execution will help you to debug your code.

Symbols you may add to a Watches window are:

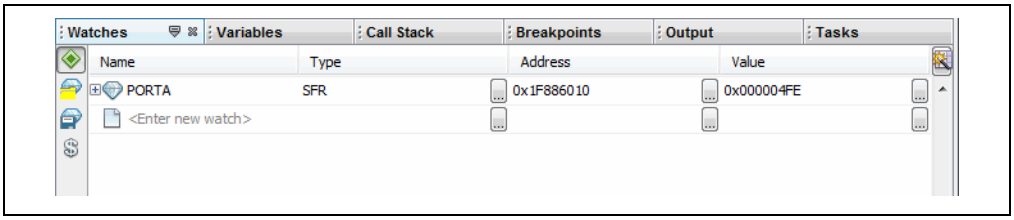
- Global symbols - visible after a build
- SFRs - Special Function Registers (device dependent)
- Absolute addresses

In general, you must Pause from a Debug Run to be able to see updated values. However, some tools allow runtime updates (you can see the values change as your program is running). Check your tool documentation to see if it supports this feature.

For all devices except PIC32 MCUs, symbols used in a runtime watch must be sized to match the device memory. That is, you need 8-bit symbols when using an 8-bit device.

For C-language enumerated types, you may enter either the enum label (text) or integer value in the window. For labels, be aware that they are case sensitive.

FIGURE 4-24: WATCHES WINDOW – PROGRAM PAUSE



To view the Watches window do one of the following:

- Select *Window>Debugging>Watches* to open the window.
- Click the **Watches** tab in the Output window if the window is already open.

To create a new watch directly:

You can add a symbol to the Watches window directly by doing one of the following:

- Double click in the name column and type in a global symbol, SFR, or absolute address (0x300).
- Right click on a global symbol or SFR in the Editor window and select “New Watch”.
- Select the global symbol or SFR in the Editor window and drag-and-drop it into the Watches window.

To create a new watch using the New Watch dialog:

You can add a symbol or SFR to the Watches window by using the New Watch dialog:

- Right click in the Watches window and select “New Watch” or select *Tools>New Watch*. Click the selection buttons to see either Global Symbols or SFRs. Click on a name from the list and then click **OK**.
- Select the symbol or SFR name in the Editor window and then select “New Watch” from the right click menu. The name will be populated in the window. Click **OK**.

To create a new runtime watch:

Before you add a runtime watch to the Watches window, you need to set up the clock:

1. Right click on the project name and select "Properties".
2. Click on the debug tool name (e.g., Real ICE) and select the option category "Clock".
3. Set the runtime instruction speed.

To add a global symbol or SFR as a runtime watch, follow the instructions under "To create a new watch using the New Watches dialog", except select "New Runtime Watch" instead of "New Watch".

For all devices except PIC32 MCUs, symbols used in a runtime watch must be sized to match the device memory. That is, you need 8-bit symbols when using an 8-bit device.

To view symbol changes:

1. Debug Run and then Pause your program.
2. Click the **Watches** tab to make the window active.
3. For watch symbols, continue to Debug Run and Pause to see changing values. For runtime watch symbols, continue Debug Run and watch the values change as the program executes.

You must be in a debug session to see the values of symbols - global symbols, SFRs, arrays, register bitfields, etc.

To change the radix of a watch symbol:

- Right click in the line of the symbol and select "Display Value As".

To perform other tasks:

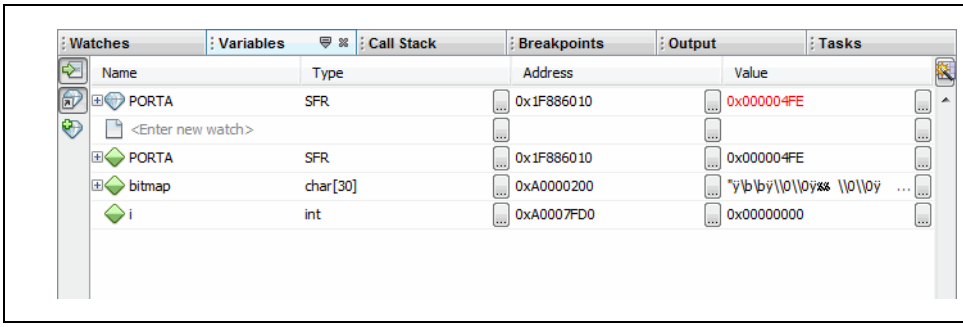
For more on watches, see **Section 12.14 "Watches Window"**.

4.20 WATCH LOCAL VARIABLE VALUES CHANGE

Watch the values of local variables change in the Variables window. Determining if these values are as expected during program execution will help you to debug your code.

In general, you must Pause from a Debug Run to be able to see updated values. However, some tools allow runtime updates. Check your tool documentation to see if it supports this feature.

FIGURE 4-25: VARIABLES WINDOW – PROGRAM PAUSE



To view the Variables window do one of the following:

- Select *Window>Debugging>Variables* to open the window.
- Click the **Variables** tab in the Output window if the window is already open.

To view variable changes:

1. Debug Run and then Pause your program.
2. Click the **Variables** tab to view the window and see the local variable value.

To change the radix of a variable:

- Right click in the line of the variable and select “Display Value As”.

To perform other tasks:

For more on variables, see the NetBeans help topics under *C/C++/Fortran Development>Debugging C/C++/Fortran Applications with gdb>Viewing C/C++/Fortran Program Information>C and C++ Variables and Expressions in the IDE*.

4.21 VIEW/CHANGE DEVICE MEMORY (INCLUDING CONFIGURATION BITS)

MPLAB X IDE has flexible, abstracted memory windows that provide a more customized view of the different types of device memory during debug. You must Pause from a Debug Run to be able to see updated values in this window.

4.21.1 View Device Memory

1. Click a window in a pane to make the pane active. The memory window will open in this pane.
2. Select a memory view from *Window>PIC Memory Views*. The available choices are explained in the following tables:

TABLE 4-6: MEMORY VIEWS – 8- AND 16-BIT DEVICES

Type	Description
Program Memory	All program memory (ROM) on the device
File Registers	All file register (RAM) memory on the device
SFRs	All Special Function Registers (SFRs)
Peripherals	All SFRs by Peripheral
Configuration Bits	All Configuration registers
EE Data Memory	All EE Data memory on the device
User ID Memory	User ID memory

TABLE 4-7: MEMORY VIEWS – 32-BIT DEVICES

Type	Description
Execution Memory	All Flash memory on the device
Data Memory	All RAM memory on the device
Peripherals	All Special Function Registers (SFRs)
Configuration Bits	All Configuration registers
CPU Memory	All CPU memory
User ID Memory	User ID memory

3. Once a Memory window is open, you may further modify your view by selecting the type of memory and memory format in drop-down boxes.

FIGURE 4-26: MEMORY AND MEMORY FORMAT SELECTION

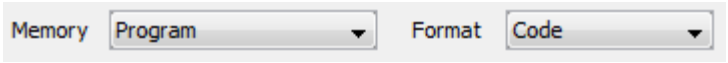


TABLE 4-8: MEMORY WINDOW OPTIONS – 8- AND 16-BIT DEVICES

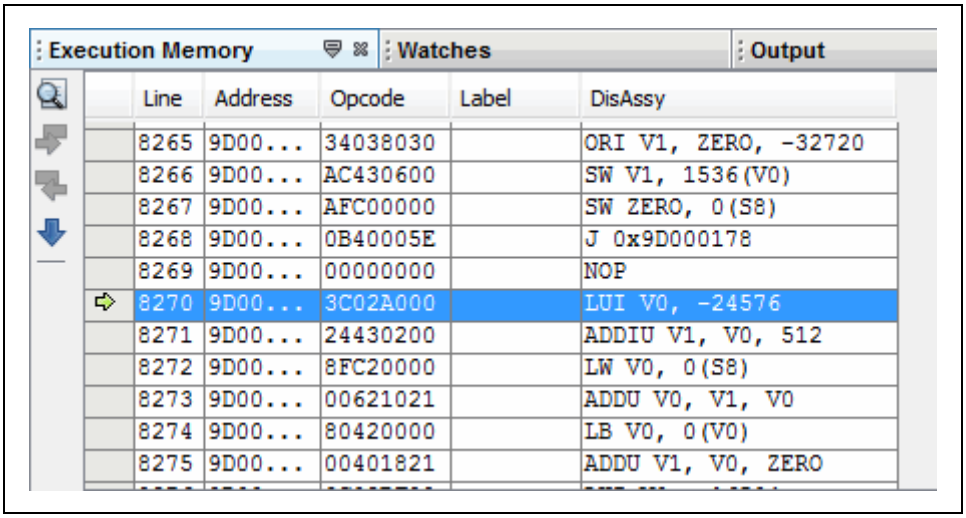
Option	Value	Description
Memory	File Registers	All file register memory on the device
	Program	All program memory on the device
	SFR	All Special Function Registers (SFRs)
	Configuration Bits	All Configuration registers
	EEPROM	All EEPROM memory
	User ID	User ID memory
Format	Data	Data Memory (RAM)
	Code	Program Memory (ROM)

TABLE 4-9: MEMORY WINDOW OPTIONS – 32-BIT DEVICES

Option	Value	Description
Memory	RAM Memory Flash Memory Peripheral Configuration Bits CPU Memory Memory User ID	All RAM memory on the device All Flash memory on the device All Special Function Registers (SFRs) All Configuration registers All CPU memory All memory User ID memory
Format	Data Code	Data Memory (RAM) Program Memory (ROM)

4. After a Debug Run and then Pause, the window will populate with the memory chosen.
5. Close the window by clicking the “x” on that window’s tab.

FIGURE 4-27: MEMORY WINDOW CONTENT



4.21.2 Change Device Memory

You must Debug Run your code to change memory values. You cannot change memory during a Run.

> **Note:** The data will change only during the Debug Run. Your application code is not changed.

To change memory values:

- Change a value in the Memory window by clicking in the appropriate column and selecting or entering new data. For some windows, the text will be red to show a change.
- The Fill memory feature is found on the context (right click) menu of most Memory windows.
- For program memory, you must rebuild to see the changes. Use *Debug>Discrete Debugger Operation* to program the target and launch the debugger with the changed data.

4.21.3 Set Memory Window Options with the Context Menu

Right clicking in the Memory window will pop up a context menu with various options such as display options, fill memory, table import/export and output to file. For more information on this menu, see **Section 12.8.13 "Memory Window Menu"**.

4.21.4 Set Configuration Bits

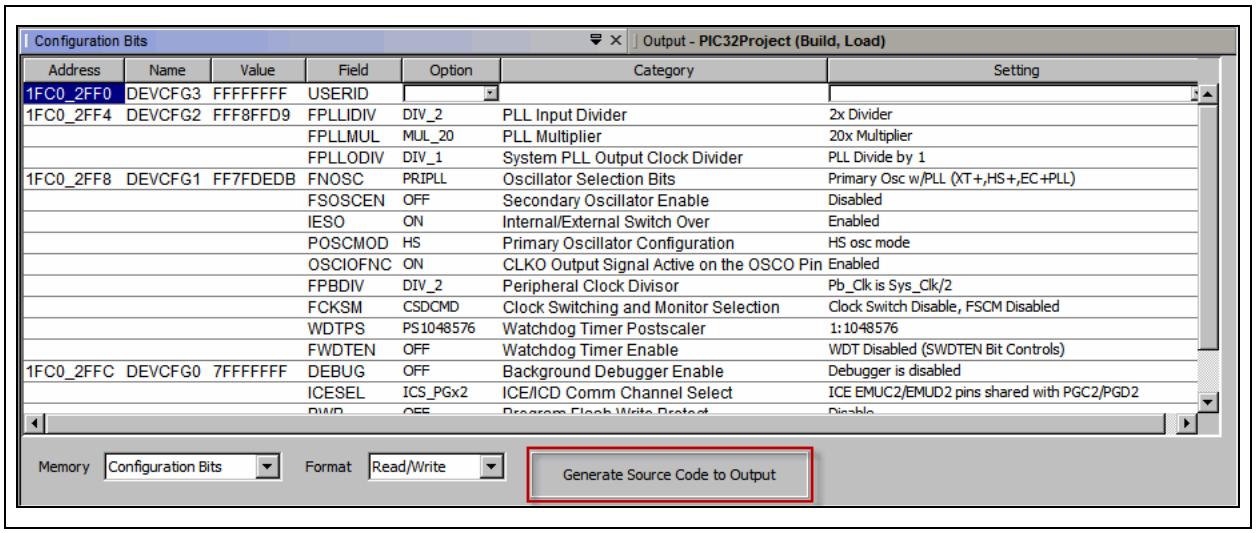
You must set Configuration bits in code. To aid you in developing your Configuration bit settings, you can use the Configuration Bits window (see **Section 4.21.1 "View Device Memory"**.)

You can temporarily change Configuration bits during a debug session in the Configuration Bits window (see **Section 4.21.2 "Change Device Memory"**.) Once you have the settings you want, select "Generate Source Code to Output". You can then copy the code from the Output window into your code.

You cannot edit the Configuration bits if you are not in a debug session. Also, if you rebuild, all changes in the Configuration Bits window will be lost.

For a summary of Configuration bits settings for different devices, see **Appendix A. "Configuration Settings Summary"**.

FIGURE 4-28: CONFIGURATION BITS WINDOW



4.21.5 Refresh Selected Memory Windows

For Memory windows showing program, EEPROM, user ID, or configuration bits memory, you can refresh the view by doing the following:

1. If you are debugging, halt your program unless your tool and device support Debug Reads (see below).
2. Click on the icon named “Read Device Memory”.

Read Device Memory Icon

Debug Reads

For most devices, you must halt your program (Finish Debugger Session) before you can read device memory. For some devices, you can read while in debug mode (Debug Read). You will know that this is available as the “Read Device Memory” icon will not be grayed out when you are debugging.

Currently, MPLAB REAL ICE in-circuit emulator and MPLAB ICD 3 support Debug Reads.

Debug Reads are done at target oscillator speeds so if the target is running very slow, a read may take a long time. You can force a fast ICSP read by finishing the debug session and then doing a read since ICSP reads will always be done when not in a debug session.

4.22 VIEW THE CALL STACK

For 16- and 32-bit devices, a software Call Stack window is available to view CALLs and GOTOs in executing C code. This window is not applicable for assembly code. (It is recommended that code optimization be turned off when using the call stack.)

The Call Stack window displays functions and their arguments listed in the order in which they were called in the executing program.

To view the call stack:

1. Debug Run and then Pause your program.
2. Select *Window>Debugging>Call Stack*. A Call Stack window will open.

For more on the call stack, see the NetBeans help topic *C/C++/Fortran Development>Debugging C/C++/Fortran Applications with gdb>Using the C/C++/Fortran Call Stack*.

4.23 PROGRAM A DEVICE

Once your code is debugged, you can program it onto a target device.

4.23.1 Set Project Programming Properties

Set up programming options in the Project Properties window:

1. Right click on the project name in the Projects window and select "Properties".
2. Under "Categories", Click on the hardware tool you will use to program your code, e.g., PM3.
3. Review the settings under the "Memories to Program" options category.
   If you wish to use a Preserve Memory option, ensure that your code is ***not*** code protected. Code is preserved when the programmer reads the section it needs to save, performs a bulk erase of the device, reprograms the device and then rewrites the area that is preserved with what was saved.
4. Review the settings under the "Program Options" options category.
5. Depending on your hardware tool, there may be other programming option categories. Review each one to ensure the settings are correct for your project.

For more information about programming options, please consult your hardware tool documentation.

After the programming options are set up as you desire, you may proceed to program the device.

4.23.2 Perform Programming

To program your target device with debugged code, click the toolbar button **Make and Program Device Project**.

Other programming-related functions are shown in the Table 4-10. The first function is activated by clicking on the button. For other functions, click on the down arrow next to the button icon.

TABLE 4-10: PROGRAMMING FUNCTIONS ON TOOLBAR BUTTONS

Button Icon	Function	Details
	Make and Program Device	The project is built (if necessary) and the device is programmed. The program will immediately begin execution on completion of programming.
	Program Device for Debugging	The device is programmed from a debug image. The program will immediately begin execution on completion of programming.
	Program Device for Production	The device is programmed from a production image. The program will immediately begin execution on completion of programming.
	Programmer to Go PICkit 3	Use the Programmer to Go feature of PICkit 3.
	Read Device Memory	Transfer what is in target memory to MPLAB X IDE.
	Read Device Memory to File	Transfer what is in target memory to the specified file.
	Read EE/Flash Data Memory to a File	Transfer what is in target data memory to the specified file.
	Hold In Reset	Toggle the device between Reset and Run.

Note: Not all programming functions are in the MPLAB X IDE. For additional programming support, see the MPLAB IPE included with the MPLAB X IDE installation.

NOTES:

Chapter 5. Additional Tasks

5.1 PERFORMING ADDITIONAL TASKS

The following steps shows how to perform more tasks in MPLAB X IDE

Work with Projects

1. Open an MPLAB IDE v8 project in MPLAB X IDE by using the Import MPLAB Legacy Project wizard.
2. Open a prebuilt image (Hex, COF or ELF) using the Prebuilt Projects import wizard.
3. Use Loadable Projects and Files to combine or replace project hex files. A common application is using loadables for combining bootloaders and application code, as in Loadable Projects and Files: Bootloaders.
4. Create Library Projects to build their output as a library.
5. Create Projects from Other Embedded Projects or Sample Projects.
6. Work with Other Types of Files, not just Microchip ones. Also Modify or Create Code Templates to change the default file templates that you use in your project.
7. Switch Hardware or Language Tools used in your project.
8. Modify Project Folders and Encoding of an existing project.
9. Speed Up Build Times using parallel make.

Debug Code

1. Use the Stopwatch to determine the timing between breakpoints.
2. View the Disassembly Window to see disassembled code.
3. Use View The Call Graph to navigate function calls.
4. View the Dashboard Display to see project information such as breakpoint resources, checksums and memory usage.

Manage Code

1. Improve Your Code* by using refactoring and profiling tools.
2. Control Source Code by using built-in file history or a version control system.
3. Collaborate on Code Development and Error Tracking* by using a team server and an issue tracking system.

Add Functionality

1. Add Plug-In Tools to aid code development.

* To see this feature, refer to the **Start Page**, **My MPLAB X IDE** tab, "Extend MPLAB" section, "Selecting Simple or Full-Featured Menus" topic.

5.2 IMPORT MPLAB LEGACY PROJECT

The Import Legacy Project wizard will import an MPLAB IDE v8 project into an MPLAB X IDE project with the following considerations:

- Settings that were saved in a workspace in MPLAB IDE v8 (such as tool settings) will not be transferred to the new MPLAB X IDE project. Refer to the MPLAB IDE v8 help for what is stored in a workspace (*MPLAB IDE Reference>Operational Reference>Saved Information*.)
  Project settings, such as compiler, linker, and assembler options, will be transferred to the new MPLAB X IDE project.
- For an MPLAB IDE v8 project using the MPLAB C Compiler for PIC24 MCUs and dsPIC DSCs (aka MPLAB C30) and a COFF debug file format:
  - The MPLAB X IDE project will be converted to the ELF/DWARF debug file format unless the project uses COFF libraries, in which case the project format will continue to be COFF.
  - MPLAB IDE v8 is case insensitive to file extensions such as `.c` and `.C`. However, MPLAB X IDE is case sensitive and associates `.c` to C code files and `.C` to C++ code files. Therefore, if you import an MPLAB IDE v8 project with C code specified as `.C`, MPAB X IDE will rename the `.C` files to `.c` to avoid incorrect compiler behavior.

5.2.1 Open the Wizard

There are two ways to open this wizard - the Start Page option and the New Project option.

5.2.1.1 START PAGE OPTION

- On the **Start Page**, click on the **Learn & Discover** tab, "Dive In" section, "Import MPLAB Legacy Project" link. Or Select *File>Import>MPLAB IDE v8 Project*.
- The "Import Legacy Project" wizard opens.

5.2.1.2 NEW PROJECT OPTION

To open the New Project wizard, do one of the following:

- On the **Start Page**, click on the **Learn & Discover** tab, "Dive In" section, "Create New Project" link.
- *File>New Project* (or Ctrl+Shift+N)

A wizard with launch to guide you through new project set up.

- **Step 1. Choose Project:** Select the "Microchip Embedded" category and choose from the project type "Existing MPLAB IDE v8 Project".
- The "Import Legacy Project" wizard opens.

5.2.2 Import Legacy Project Wizard

Follow the steps below to import your MPLAB IDE v8 project. Click **Next>** to move to the next step.

- **Step 1 or 2. Import Legacy Project:** Enter or browse to the legacy project.
- **Step 2 or 3. Select Device:** Select the device you will be using in your application from the "Device" drop-down list. To narrow your selection list, chose a Family first.

- **Step 3 or 4. Select Header:** This step will appear if a header is available for your selected device. To determine if a header is required for debug or if your device has on-board debug circuitry, consult the *Processor Extension Pak and Debug Header Specification* (DS51292 or online help). Then choose whether or not to use a header.
- **Step 4 or 5. Select Tool:** Select the development tool you will be using to develop your application from the list.

  The level of tool support for the selected device will be displayed next to the tool name. That is, green for full support, yellow for beta support and red for no support yet.
- **Step 5 or 6. Select Compiler:** Select the language tool (compiler) you will be using to develop your application from the list.

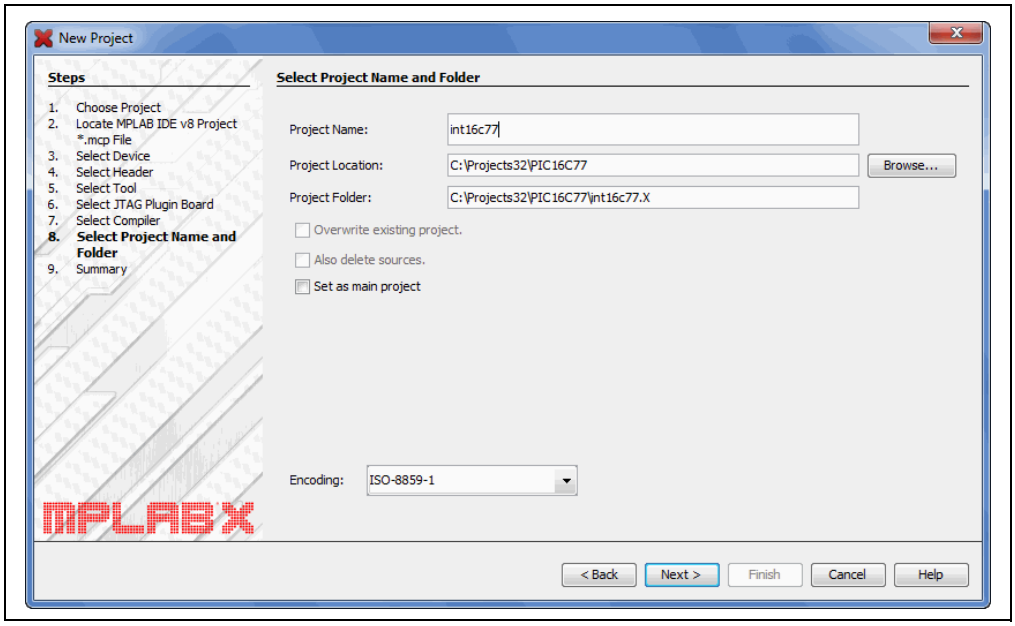
  The level of tool support for the selected device will be displayed next to the tool name. That is, green for full support, yellow for beta support and red for no support.
- **Step 6 or 7. Select Project Name and Folder:** It is recommended that you do not change the default name and location to preserve maintainability of both projects.

  File Locations: The new project will not copy the source files into its folder, but instead will reference the location of the files in the v8 folder. To create an independent MPLAB X IDE project, create a new project and copy the MPLAB IDE v8 source files to it.

  Main Project: Check the checkbox to make this project the main project on import.

  File Formatting: “ISO-8859-1” is the default character encoding used when importing a project from MPLAB IDE v8. You should select the encoding that matches the one that is used in the imported project. For example, if the MPLAB IDE v8 format is “950 (ANSI/OEM – Traditional Chinese Big5)”, then select “Big5” from the drop-down list.

FIGURE 5-1: IMPORT LEGACY – SELECT PROJECT NAME AND FOLDER



- **Step 7 or 8. Summary:** Review the summary before clicking **Finish**. If anything is incorrect, use the **Back** button to go back and change it.

The legacy project will open in the Projects window.

5.3 PREBUILT PROJECTS

Create a project from a prebuilt loadable image (Hex, COF, or ELF files) by using the Import Image File wizard.

> **Note:** To program a prebuilt image into a device, you will click the **Make and Program Device** button even though it will only program the device (no make).

There are two ways to open this wizard – the Start Page option and the New Project option.

5.3.1 Start Page Option

- On the **Start Page**, click on the **Learn & Discover** tab, "Dive In" section, "Import Hex (Prebuilt) Project" link. Or Select *File>Import>Hex/ELF (Prebuilt) File*.
- The "Import Image File" wizard opens.

5.3.2 New Project Option

To open the New Project wizard, do one of the following:

- On the **Start Page**, click on the **Learn & Discover** tab, "Dive In" section, "Create New Project" link.
- *File>New Project* (or Ctrl+Shift+N)

A wizard with launch to guide you through new project set up.

- **Step 1. Choose Project:** Select the "Microchip Embedded" category and choose from the project type "Prebuilt (Hex, Loadable Image) Project".
- The "Import Image File" wizard opens.

5.3.3 Import Image File Wizard

Follow the steps below to import your image file. Click **Next>** to move to the next step.

- **Step 1 or 2. Import Image File:** Select the name and location of your image file. You may browse to a location.
- **Step 2 or 3. Select Device:** Select the device you will be using in your application from the "Device" drop-down list. To narrow your selection list, choose a Family first.
- **Step 3 or 4. Select Header:** This step will appear if a header is available for your selected device. To determine if a header is required for debug or if your device has on-board debug circuitry, consult the *Processor Extension Pak and Debug Header Specification* (DS51292 or online help). Then choose whether or not to use a header.
- **Step 4 or 5. Select Tool:** Select the development tool you will be using to develop your application from the list.

  The level of tool support for the selected device will be displayed next to the tool name. That is, green for full support, yellow for beta support and red for no support yet.
- **Step 6 or 7. Select Project Name and Folder:** Select a name and location for your new project. You may browse to a location.
- **Step 7 or 8. Summary:** Review the summary before clicking **Finish**. If anything is incorrect, use the **Back** button to go back and change it.

The new project will open in the Projects window.

For information on exporting a project as hex, see **Section 12.11.2 "Projects Window – Project Menu"**.

5.4 LOADABLE PROJECTS AND FILES

Use loadable projects and files to combine projects, combine hex files, combine projects and hex files or replace the project hex file. The hexmate application is used to merge project or loaded hex files into one file. (For details on this application, see support documents in the `docs` folder of the installed MPLAB XC8 compiler.)

Loadable projects or files are useful for creating combined bootloader and application code. See **Section 5.5 “Loadable Projects and Files: Bootloaders”**.

The combinations of current projects and loadables are listed below.

TABLE 5-1: LOADABLE COMBINATIONS

Current Project	Loadable	Caveat
Stand-Alone Existing MPLAB IDE v8 Library	Stand-Alone	None
	Hex file	None
	COF/ELF file	• Can debug, but not build. An error will be displayed in the Output window.
Prebuilt (Hex)	Hex file	• No auto checking for overlapping memory areas. • Can debug but not build. **Build** button will be disabled.
	COF/ELF file	
Prebuilt (COF/ELF)	Hex file	
	COF/ELF file	

The options are listed below.

TABLE 5-2: LOADABLE OPTIONS

Add Loadable Project(s)	Load one or more existing projects into your current project. When you build your current project, all projects will be built and the hex files will be combined into one. All debug files will be combined as well (COFF or ELF).
Add Loadable File(s)	Load one or more existing hex files into your current project. When you build your current project, the hex file will be combined with the other hex files into one file. **Note:** You will no longer be able to debug the project that contains hex file(s). Use Loadable Projects for debugging.
Add Alternate File	Load an alternate hex file to be used. This options provides a post-build step where you may copy or move your project hex file to another location, use a tool such as hexmate to merge your file with another hex file, and then load the file back into the IDE.

To set up and use loadables, see:

- Projects Window - Loadables Set Up
- Project Properties Window - Loading Set Up
- The Preferred Method to Use Loadables

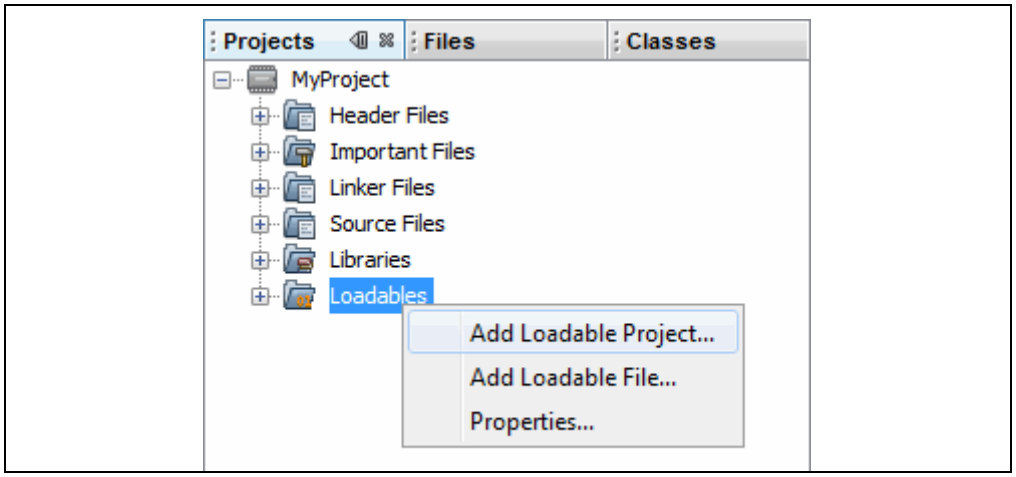
5.4.1 Projects Window - Loadables Set Up

Right Click on the "Loadables" folder in the Projects window (Figure 5-2) and select an option:

- Add Loadable Project - Select to add an existing project to your current project. Repeat to add additional projects.
- Add Loadable Files - Select to add an existing hex file to your current project. Repeat to add additional hex files.
- Properties - Open the Project Properties window for Loading. See **Section 5.4.2 "Project Properties Window - Loading Set Up"**.

Build your current project to build all projects and combine hex files into one. Any debug files will also be combined.

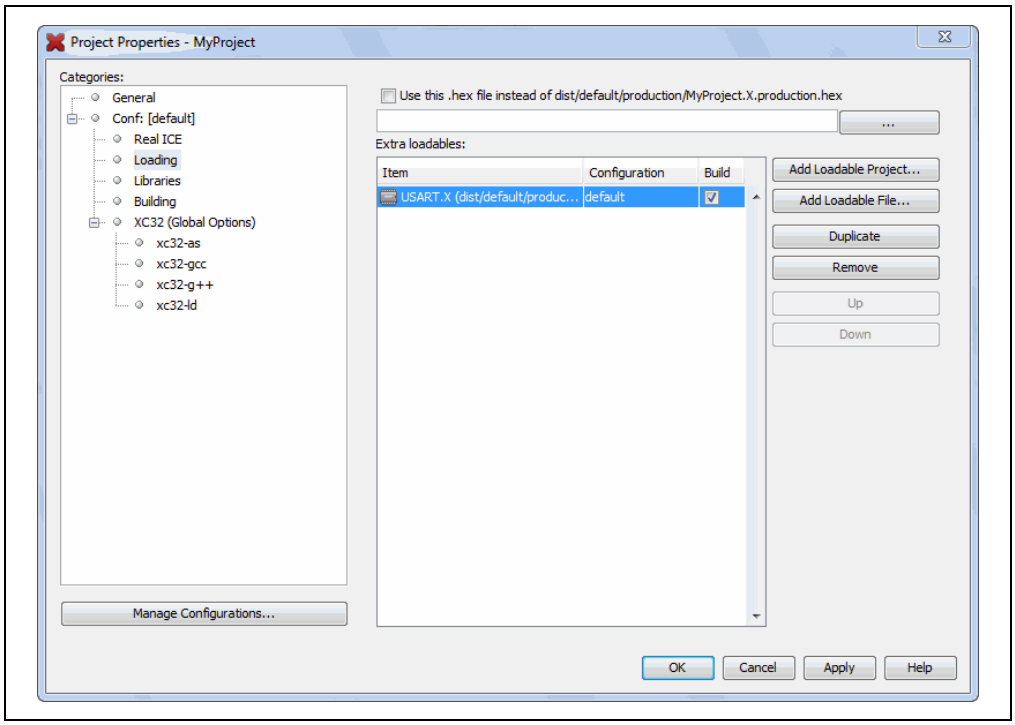
FIGURE 5-2: PROJECTS WINDOW - LOADABLES FOLDER



5.4.2 Project Properties Window - Loading Set Up

Open the Project Properties window (*File>Project Properties*) and click on “Loading”.

FIGURE 5-3: PROJECT PROPERTIES - LOADING



To combine the current project with other projects

1. Click **Add Loadable Project**. Browse to a project, select it and click **Add**.
2. Select a build configuration to use for this project in the drop-down box under “Configuration”. If you have not added different configurations to this project, you will only see “default”.
3. Ensure the “Build” checkbox is checked if you want to build this project when you build the current project.
4. If you have added more than one project, the order shown here will determine the order in which the hex files will be added to the current project’s hex file. Use **Up** and **Down** to change the order.
5. Click **Apply** or **OK** to accept the changes.

The next time you build the current project, the projects listed here will also be built (if “Build” is checked) and their hex files will be combined with the current project’s hex file to create a single output hex file. Any debug files will also be combined.

To combine the current project hex file with other hex files

1. Click **Add Loadable File**. Browse to a hex file, select it and click **Add**.
2. If you have added more than one file, the order shown here will determine the order in which the hex files will be added to the current project’s hex file. Use **Up** and **Down** to change the order.
3. Click **Apply** or **OK** to accept the changes.

The next time you build the current project, the hex files listed here will be combined with the current project’s hex file to create a single output hex file.

To load an alternative hex file:

1. Click to check "Load this hex file on run or program builds only (debug not affected)".
2. Browse to the desired hex file and select it. Select how you want the file referenced (Auto, Relative, Absolute) and then click **Add**.

The next time you build the current project, the alternative hex file will load on build complete.

5.4.3 The Preferred Method to Use Loadables

The recommended way to use loadables is:

1. Select the project that sets the device configuration bits and initialization as the main project (right click and select "Set as Main Project"). The loadable should not have device configuration settings as this will conflict with the main project.
2. Add the other projects as loadables to this main project. If the project being loaded has more than one project configuration, be sure to specify that when loading.

Specifying the project containing the loadables as the main project ensures that a change in any loadable will be picked up by the build when the **Build** button is pressed.

5.5 LOADABLE PROJECTS AND FILES: BOOTLOADERS

To combine a bootloader with application code:

1. Create one project for your application and one project for your bootloader.
2. Load the bootloader project or hex file into the application project. See **Section 5.4.1 "Projects Window - Loadables Set Up"** or **Section 5.4.2 "Project Properties Window - Loading Set Up"** for how to do this.

The next time you build your application project, the resulting hex file will be a combined bootloader/application hex file. Any debug files will also be combined.

For build errors, see **Section 5.4.3 "The Preferred Method to Use Loadables"** or the sections below.

Consideration #1: MPLAB C Compiler for PIC18 MCUs (MPLAB C18)

This compiler provides application start-up code (`c018x.o`) that begins at the reset vector (address 0) for use in initializing the software stack, optionally initializing the `idata` section, and jumping to `main()`. If this start-up code is left in an application, there will always be a conflict with the bootloader code reset and you will get a linker error message about a data conflict.

A resolution would be to edit the start-up code to begin at an address other than 0.

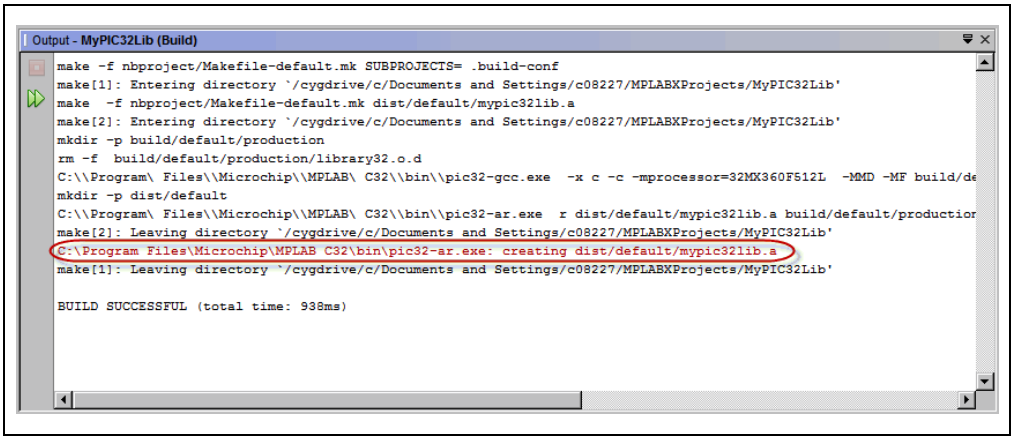
Consideration #2: MPLAB XC8 - PIC18 MCU Example

The following Microchip webinar details how to combine a bootloader with application code for a PIC18 MCU using MPLAB XC8 and MPLAB X IDE:
"Linking PIC18 Bootloaders & Applications"

5.6 LIBRARY PROJECTS

Create a new library project that uses an IDE-generated makefile to build your project as a library file instead of an executable.

FIGURE 5-4: LIBRARY PROJECT EXAMPLE



To begin, open the New Project wizard by doing one of the following:

- On the **Start Page**, click on the **Learn & Discover** tab, “Dive In” section, “Create New Project” link.
- *File>New Project* (or Ctrl+Shift+N)

A wizard will launch to guide you through new project set up. Click **Next>** to move to the next step.

- **Step 1. Choose Project:** Select the “Microchip Embedded” category and choose from the project type “Library Project”.
- **Step 2. Select Device:** Select the device you will be using in your application from the “Device” drop-down list. To narrow your selection list, chose a Family first.
- **Step 3. Select Header:** This step will appear if a header is available for your selected device. To determine if a header is required for debug or if your device has on-board debug circuitry, consult the *Processor Extension Pak and Debug Header Specification* (DS51292 or online help). Then choose whether or not to use a header.
- **Step 4. Select Tool:** Select the development tool you will be using to develop your application from the list.

  The level of tool support for the selected device will be displayed next to the tool name. That is, green for full support, yellow for beta support and red for no support yet.
- **Step 5. Select Compiler:** Select the language tool (compiler) you will be using to develop your application from the list.

  The level of tool support for the selected device will be displayed next to the tool name. That is, green for full support, yellow for beta support and red for no support.
- **Step 6. Select Project Name and Folder:** Select a name and location for your new project. You may browse to a location.

The new project will open in the Projects window.

5.7 OTHER EMBEDDED PROJECTS

MPLAB X IDE can create a project from selected, other embedded projects.

1. Select *File>New Project*.
2. Click on "Other Embedded" under "Categories" and select from a list of available embedded projects.
3. Continue to create an MPLAB X IDE project.

This feature imports your existing files into an MPLAB X IDE project. Conversion of other embedded project settings or code is not yet available.

For information on how to work with MPLAB X IDE, see:

- **Chapter 3. "Tutorial"**
- **Chapter 4. "Basic Tasks"**

For information on available compilers, see:

http://www.microchip.com/xc

5.8 SAMPLE PROJECTS

Create a sample project to help you learn about Microchip devices, tools and MPLAB X IDE.

1. Select *File>New Project*.
2. Click on "Samples>Microchip Embedded" under "Categories" and select from a list of available embedded projects (that blink demo board lights) or template projects. Read the Description for more information.

Part numbers for demo boards are as follows:

Explorer 16 Demo Board: DM240001

PICDEM 2 Plus: DM163022-1

5.9 WORK WITH OTHER TYPES OF FILES

When selecting *File>New*, you are presented with many types of files. Using Microchip compiler files has been explored previously. However, there are other types of files you can select depending on your project language tool or need to create a specific file.

TABLE 5-3: FILE TYPES

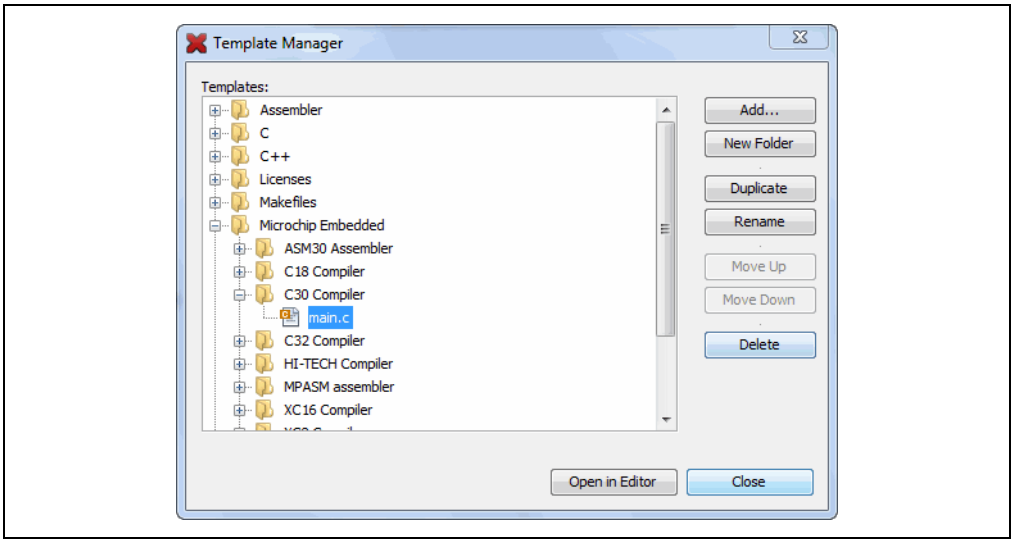
Category	File Type
Microchip Embedded	Files for language tools that are supported in MPLAB X IDE Select your compiler folder to see the available file types.
C	Generic C files
C++	Generic C++ files
Assembler	Generic assembly files
Shell Scripts	Shell script files: Bash, C, Korn, etc.
Makefiles	Makefile files
XML	XML files
Other	Other types of files, such as HTML, JavaScript, etc. If you do not see the type of file you want listed here, select "Empty File". In the next window, name the file with the desired extension.

5.10 MODIFY OR CREATE CODE TEMPLATES

When you create a file to add to your project (**Section 4.8 "Create a New File"**), a template is used for the new file. To change this template, select *Tools>Templates* and then "Open in Editor" to edit a template. You may also use "Add" or "Duplicate" in this dialog to create new templates.

"New Folder" can be used to create a new folder to hold templates. Be aware that MPLAB X IDE filters out all but Microchip Embedded, Shell Scripts, Makefiles and Other, so files or folders should be created under those folders.

FIGURE 5-5: TEMPLATE MANAGER



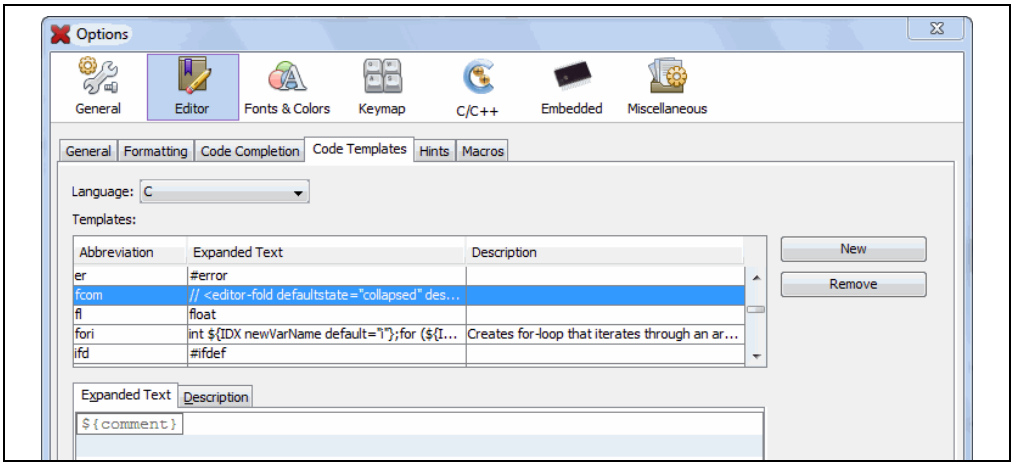
You may set template options by selection *Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Editor** button, **Code Templates** tab (as shown below).

An option of note for C code is the "fcom" option. In an Editor window, type "fcom" and then press "Tab" to insert the following text into the source code:

```
// <editor-fold defaultstate="collapsed" desc="comment">
// </editor-fold>
```

This option allows you to hide/view sections of code.

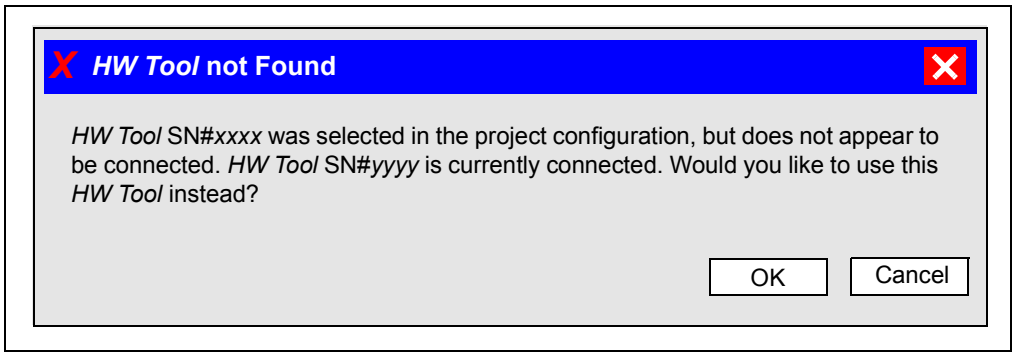
FIGURE 5-6: TEMPLATE OPTIONS



5.11 SWITCH HARDWARE OR LANGUAGE TOOLS

When you open an existing project and you have connected a hardware tool that is different from the one you specified in your project, MPLAB X IDE will pop up a dialog asking if you would like to make the new hardware tool the project tool.

FIGURE 5-7: SWITCH HARDWARE TOOL DIALOG



You may also plug in two or more hardware tools and switch between them in the Project Properties dialog (*File>Project Properties*).

To switch between different versions of compiler toolchains (language tools), again use the Project Properties dialog.

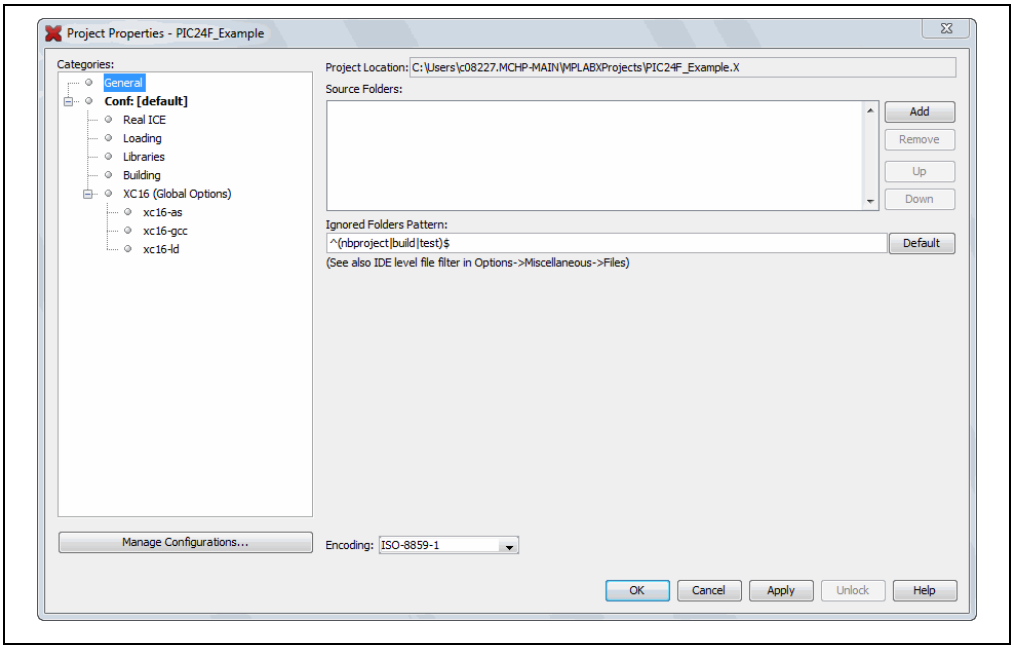
5.12 MODIFY PROJECT FOLDERS AND ENCODING

When you created your project, you specified the project folder and encoding. Once the project is created, you may add or ignore project folders and change the project encoding using the "General" category in the Project Properties window.

TABLE 5-4: PROJECT PROPERTIES - GENERAL CATEGORY

Option	Description
Project Location	View the current project location. See **Section 8.8 "Moving, Copying or Renaming a Project"** to change the project location.
Source Folders	Add folders for MPLAB X IDE to search to look for project files. **Note:** Adding files outside the project folder may make the project less portable.
Ignored Folders Pattern	Ignore folders in the project folder according to the regular expression pattern specified.
Encoding	Change the project encoding. This selection will specify the code syntax coloring, which can be edited under *Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Fonts and Colors** button, **Syntax** tab.

FIGURE 5-8: PROJECT PROPERTIES - GENERAL



5.13 SPEED UP BUILD TIMES

Depending on the configuration of your computer, you may be able to use parallel make (see **Section 12.12.2 "Project Options Tab"**) to speed up your project build times. Not all language tools support parallel make.

Another option is to consider your operating system (OS). Some OS's have faster file accessing. MPLAB X IDE supports Windows, Linux and Mac OS's. Research which one might be right for you.

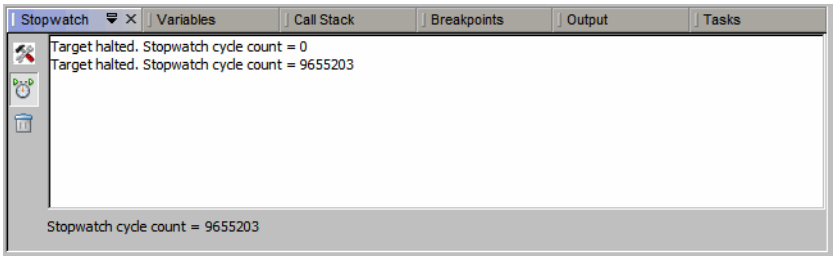
5.14 USE THE STOPWATCH

Use the stopwatch to determine the timing between two breakpoints.

To use the Stopwatch:

1. Add a breakpoint where you want to start the stopwatch.
2. Add another breakpoint where you want to stop the stopwatch.
3. Select *Window>Debugging>Stopwatch*. Click on the Properties icon on the left of the window and select the start and stop breakpoints.
4. Debug Run the program again to get the stopwatch timing result.

FIGURE 5-9: STOPWATCH WINDOW WITH CONTENT



The stopwatch has the following icons on the left side of the window:

Icon	Description
Properties	Set stopwatch properties. Select one current breakpoint or trigger to start the stopwatch and one to stop the stopwatch.
Reset Stopwatch on Run	Reset the stopwatch time to zero on the start of a run.
Clear History	Clear the stopwatch window.
Clear Stopwatch	(Simulator Only) Reset the stopwatch after you reset the device.

5.15 VIEW THE DISASSEMBLY WINDOW

View disassembled code in this window. Select *Window>Output>Disassembly Listing File* to open the window.

This information may also be found in the listing file produced by the linker. Open this file by selecting *File>Open File* and browsing for the `ProjectName.lst` file.

A quick way to view the entire disassembly file is to right click in the disassembly window and select "Disassembly Listing File".

Note: The disassembly window will disassemble each instruction but has no history of banking associated with the instruction. Therefore, SFR names displayed in the window will be for Bank 0.

5.16 VIEW THE CALL GRAPH

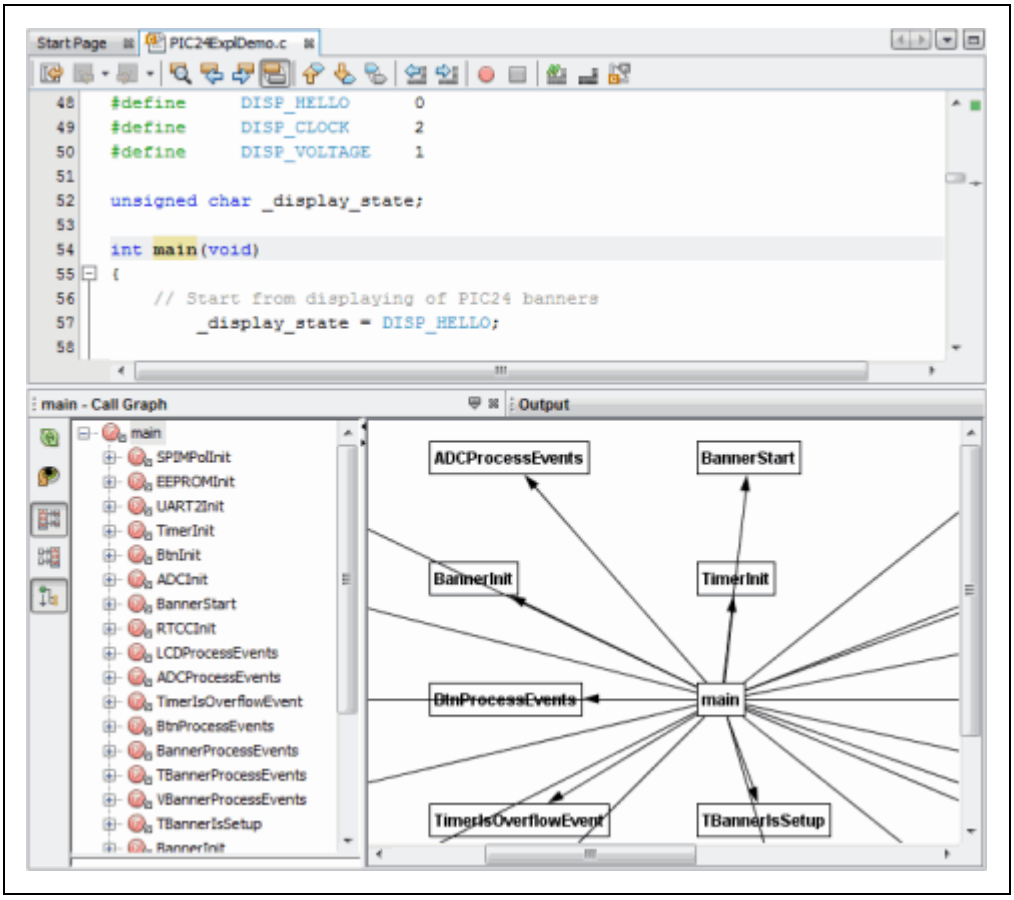
The Call Graph window displays a tree view of either the functions called from a selected function, or the functions that call that function.

To view the call graph:

Right click on a function and select “Show Call Graph” from the drop-down menu.

For more information, see the NetBeans help topic *C/C++/Fortran Development>Working With C/C++/Fortran Projects>Navigating Source Files and Projects>Using the Call Graph*.

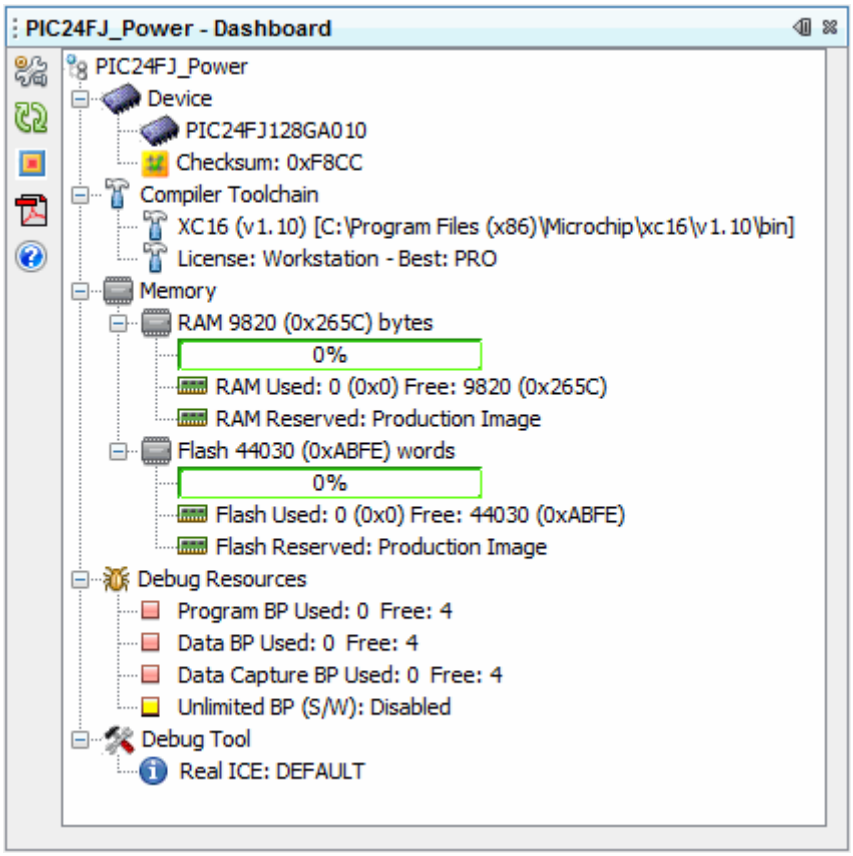
FIGURE 5-10: CALL GRAPH OF MAIN FUNCTION



5.17 VIEW THE DASHBOARD DISPLAY

Select *Window>Dashboard* to display project information.

FIGURE 5-11: DASHBOARD DISPLAY



The project displayed will either be:

- The active project in the Projects window if no main project is selected (*Run>Set Main Project>None*.) Click on a project in the Projects window to make it active.
- The main project. No other project, active or inactive, will be displayed.

Features of the Dashboard window are listed in the tables below.

TABLE 5-5: DASHBOARD GROUPS

Group	Definition and Content
Device	• The project device • Any status flags generated during a Run or Debug Run • The checksum for the built program, i.e., you must build to see this
Compiler Toolchain	• The name of the project toolchain (compiler, assembler, linker, etc.) Includes the tool version number and the path to the executable files. • Toolchain license type: PRO, STD, or Free For more on these types, see your language tool documentation.

TABLE 5-5: DASHBOARD GROUPS (CONTINUED)

Group	Definition and Content
Memory	The type and amount of memory (data and program) used by the project, as well as memory reserved for debugging. Memory Used should be a guide to the amount of memory remaining. The compilers output a Memory Summary that details usage for Program Space, Configuration Bits, ID location, and EEPROM (if on the device). The sum of these memory spaces, allowing for word sizes, should agree with the Dashboard Flash Used. The Map file should be examined when memory is at a premium.
Debug Resources (Breakpoints)	The number of hardware breakpoints available for the project device. How many breakpoints are in use at the current time. Are software breakpoints supported on the project device.
Debug Tool	Connection status of the debug tool. **Hardware Debug Tools** The connection is only active during a Debug Run, Run, or programming. Otherwise it is inactive. To keep the tool connection active at all times, go to *Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Embedded** button, **Generic Settings** tab, and check "Maintain active connection to hardware tool". Click the **Refresh Debug Tool Status** button to see hardware debug tool firmware versions and current voltage levels. **Simulator** When the simulator is selected as your project debug tool, there will be an extra entry called "Click for Simulated Peripherals". Click on this item to see what device peripherals are supported.

TABLE 5-6: SIDEBAR ICONS

Icon	Function
	**Project Properties** Display the Project Properties dialog.
	**Refresh Debug Tool Status** Click this to see hardware debug tool details.
	**Toggle Software Breakpoint - Enabled/Disabled** Click to alternately enable or disable software breakpoints. The state of this feature is shown in the icon: Red center: Disabled Green center: Enabled **Black** center: Not supported - combination of debug tool and device capabilities
	**Open Device Data sheets** Get a device data sheet from the Microchip web site (http://www.microchip.com/). Click to either open a saved, local data sheet or open a browser to go to the Microchip web site to search for a data sheet.
	**Compiler Help** Click to open the Master Index (if available) for documents in the `docs` folder of the compiler installation directory.

5.18 IMPROVE YOUR CODE

Improve your code by using code refactoring and/or profiling.

> **Note:** To see this feature, refer to the **Start Page**, **My MPLAB X IDE** tab, "Extend MPLAB" section, "Selecting Simple or Full-Featured Menus" topic.

Refactoring code is a method of making code simpler without changing its functionality. Currently you can do the following with C code:

- Find function usages throughout files
- Rename functions and parameters throughout files

For more information, see **Section 7.5 "C Code Refactoring"**.

Profiling code is the examination of CPU Usage, Memory Usage, and Thread Usage tools, all while the program is running. The profiling tools run automatically whenever you run your C project.

5.19 CONTROL SOURCE CODE

MPLAB X IDE has a built-in local file history feature, complements of the NetBeans platform. This feature provides built-in versioning support for local projects and files, similar to conventional version control systems. Available tools include a local DIFF and file restoration. Right click on a file in the Project or File window to see Local History options.

To see local history for a file:

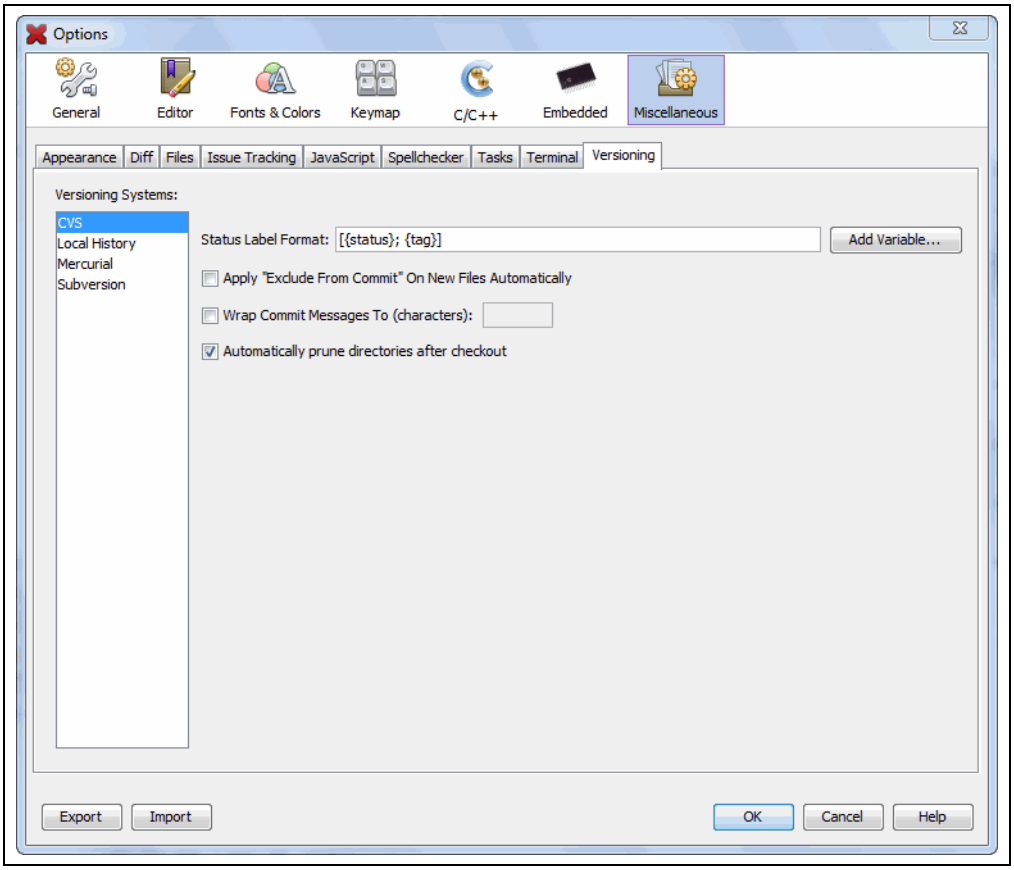
- Right click on the file in the Project or File window and select *Local History>Show Local History*. Any past changes to the file should be listed here.
- Right click on the file in the Project or File window and select *Local History>Revert to*. The Revert to dialog opens with any previous versions of the document. Select one and click **OK** to revert to that version.

If you prefer to use a full version control system, support is available for CVS, Subversion, and Mercurial.

To use source/version control:

1. Team menu – Select a version control program from the submenus and set up that version control program.
2. *Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Miscellaneous**, **Versioning** – Set up version control options (see Figure 5-12).
3. *Window>Versioning* – Open version control windows.

FIGURE 5-12: VERSION CONTROL OPTIONS



Project files that need to be saved into a repository:

The following table lists project files that either need or do not need to be committed to a version control repository.

TABLE 5-7: PROJECT FILES SAVED TO REPOSITORY

Directory or File(s)	Commit?
Project directory	
`Makefile`	Yes
Source files	Yes
build directory	No
dist directory	No
nbproject directory	
`configurations.xml`	Yes
`project.properties`	Yes
`project.xml`	Yes
`Makefile-*`	No
`Package-*`	No
private directory	No
Yes: Required to generate the project image. No: These directories/files are regenerated and therefore do not need to be saved.	

See **Section 8.3 "Files Window View"** for more on the project structure.

For more on using local file history and/or source control, see the NetBeans help topics under *IDE Basics>Version Control and File History*.

For further information on the source control programs above, see:

- CVS – http://www.nongnu.org/cvs/
- Subversion – http://subversion.tigris.org/
- Mercurial – http://mercurial.selenic.com/

5.20 COLLABORATE ON CODE DEVELOPMENT AND ERROR TRACKING

Collaborate on code development with your group using a team server (such as Kenai.com) supported inside MPLAB X IDE.

> **Note:** To see this feature, refer to the **Start Page**, **My MPLAB X IDE** tab, "Extend MPLAB" section, "Selecting Simple or Full-Featured Menus" topic.

Supporting menu items are:

- *Team* – The main team server menu. Log into your account, create or open your project, share your project, get resources, send a chat message or show your contact list.
- *File>Open Team Project* – Open an existing team project.

Collaborate on tracking bugs by using issue tracking systems, namely Bugzilla™ and JIRA® (plug-in required). Supporting menu items are:

- *Windows>Services* – Right click on "Issue Tracker" to add an issue tracker.
- *Team>Find Issues* – In the Issue Tracker window, select the project's issue tracker, select criteria, and click Search.
- *Team>Report Issues* – In the Issue Tracker window, select the project's issue tracker, specify the issue details, and click Submit.

For more on team projects and issue tracking, see the NetBeans help topic *IDE Basics>Collaborative Development*.

To find out more about these tools, see the following:

- Kenai – http://kenai.com/
- Bugzilla – http://www.bugzilla.org/
- JIRA – http://www.atlassian.com/software/jira/

5.21 ADD PLUG-IN TOOLS

MPLAB IDE v8 plug-in tools like DMCI and MATLAB will be available for MPLAB X IDE from the Plugin Manager (*Tools>Plugins*). The Macro controls which appear under the MPLAB IDE v8 Tools menu are really an editor feature and so exist under the Edit menu for MPLAB X IDE.

- Add Plug-Ins
- Upgrade Plug-Ins
- Configure Update Centers
- Plug-In Code Location

5.21.1 Add Plug-Ins

To view and add plug-ins to MPLAB X IDE:

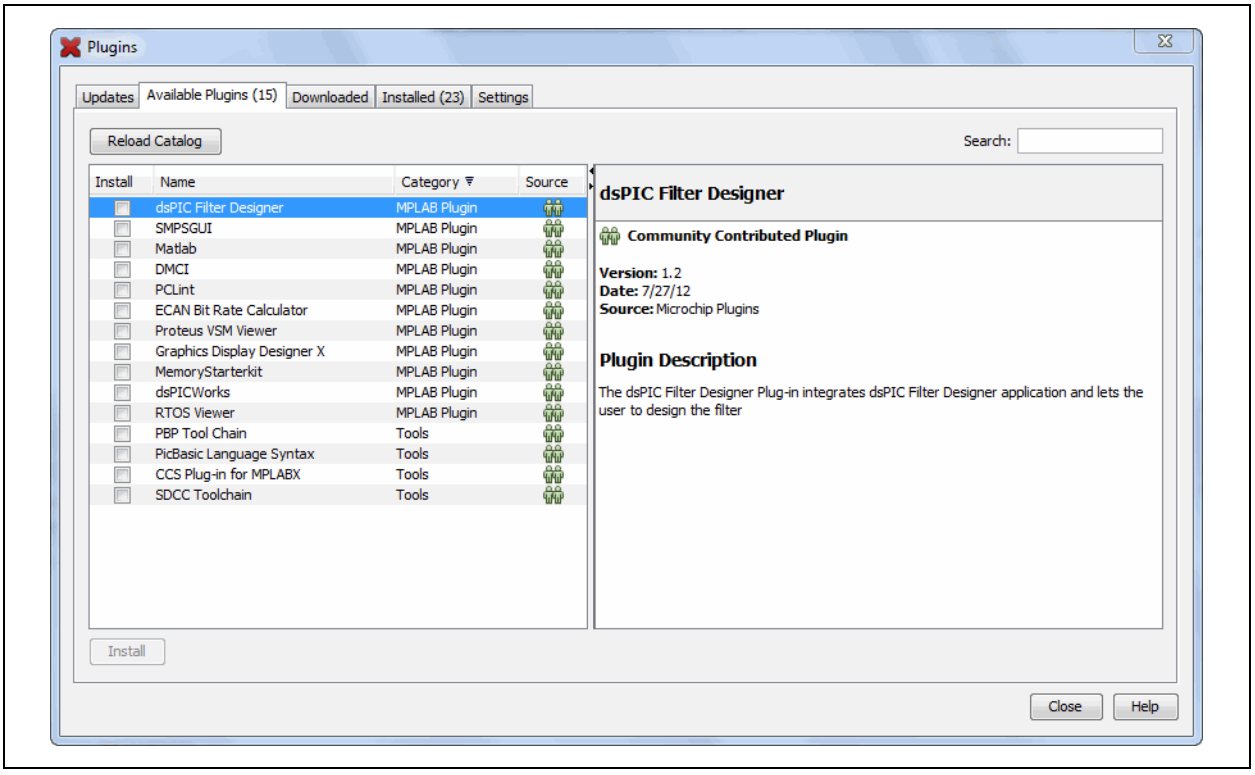
1. Click on the **Available Plugins** tab.
2. Select your plug-in by checking its checkbox.
3. Click **Install**.
4. Follow the on-screen instructions to download and install your plug-in.

> **Note:** Some plugins may be dependent on modules in other plugins in order for the functionality to be implemented. The Plugins Manager warns you when this is the case.

5. Look for your tool under *Tools>Embedded*. If you do not see it, you may need to close and re-open MPLAB X IDE.

Click the **Help** button to read more about installing plug-ins.

FIGURE 5-13: AVAILABLE MICROCHIP PLUG-IN TOOLS



5.21.2 Upgrade Plug-Ins

Installing a new version of MPLAB X IDE will NOT update your installed plug-ins. Plug-ins are tested against a versioned interface of a release. Not all plug-ins can be migrated between versions so they are not carried over. This is true also for NetBeans.

To upgrade your plug-in to a newer version or add it for a newer version of MPLAB X IDE, please us the Plugins dialog in **Section 5.21.1 “Add Plug-Ins”**.

5.21.3 Configure Update Centers

To see available plugins, you must have one or more update centers configured.

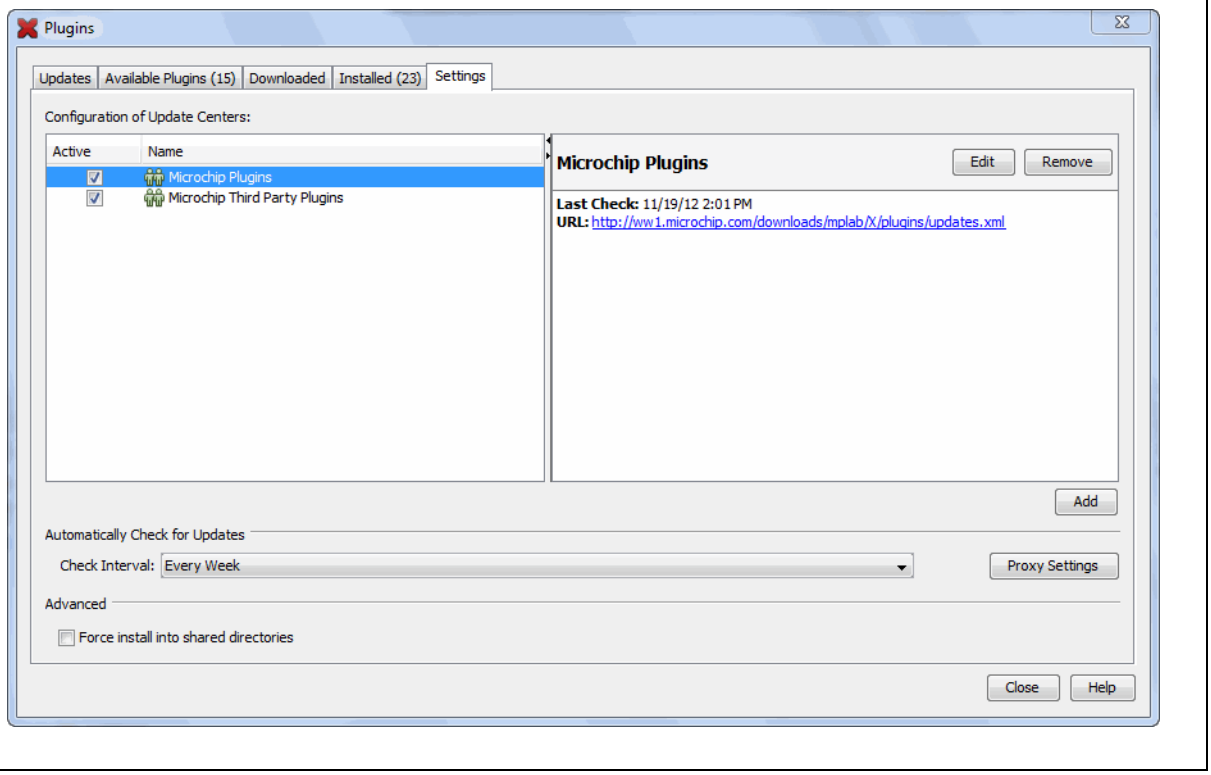
MPLAB X IDE comes with two Microchip Update Centers already configured:

- http://ww1.microchip.com/downloads/mplab/X/plugins/updates.xml
- http://ww1.microchip.com/downloads/mplab/X/thirdpartyplugins/updates.xml

To configure another Update Center in the Plugin Manager:

1. Select *Tools>Plugins* and click the **Settings** tab.
2. Click the **Add** button to open the Update Center Customizer dialog.
3. Enter a name for the update center.
4. Enter a URL for the update center.
5. Click **OK**.

FIGURE 5-14: CONFIGURE MICROCHIP UPDATE CENTER



5.21.4 Plug-In Code Location

Plug-In code is stored with MPLAB X IDE user configuration data. See **Section 8.6 “Viewing User Configuration Data”**.

NOTES:

Chapter 6. Advanced Tasks

6.1 INTRODUCTION

This chapter provides a guide for performing advanced tasks in MPLAB X IDE. Other feature are discussed in **Chapter 4. "Basic Tasks"** and **Chapter 5. "Additional Tasks"**.

- Speed Up MPLAB X IDE
- Work with Multiple Projects
- Work with Multiple Configurations
- Create User MakeFile Projects
- Log Data
- Customize Toolbars

6.2 SPEED UP MPLAB X IDE

If MPLAB X IDE is operating too slowly, consider the following:

- Increase Computer Heap
- Debug Tool Usage

6.2.1 Increase Computer Heap

You can modify the amount of memory allocated to MPLAB X IDE in the file `mplab_ide.conf`. We recommend you back up this file before you start editing it. If you change the contents of this file, the changes will be reflected the next time you run MPLAB X IDE.

Windows OS 64 Bit

```
C:\Program Files (x86)\Microchip\MPLABX\mplab_ide\etc
```

Windows OS 32 Bit

```
C:\Program Files\Microchip\MPLABX\mplab_ide\etc
```

Linux OS

```
/opt/microchip/mplabx/mplab_ide/etc
```

Mac OS X

```
/Applications/microchip/mplabx/Contents/Resources/mplab_ide/etc
```

The following line contains the default values:

```
default_options="-J-Dnb.FileChooser.useShellFolders=false
-J-Dcrownking.stream.verbosity=very-quiet -J-Xms256m -J-Xmx512m
-J-XX:PermSize=128m -J-XX:MaxPermSize=384m -J-XX:+UseConcMarkSweepGC
-J-XX:+CMSClassUnloadingEnabled"
```

The bolded areas are:

-Xms256m tells the JVM to start with at least 256 MB for the heap.

-Xms512m tells the JVM to allocate as much as 512 MB for the heap, but no more.

-XX:PermSize=128m tells the JVM to allocates 128 MB for space needed to keep track of additional data that does not go on the heap.

-XX:MaxPermSize=384m tells the JVM to allocate no more than 384 MB for the additional data that does not go on the heap.

You should not need to modify the PermSize or the MaxPermSize unless you get the error `java.lang.OutOfMemoryError: PermGen space`.

In general, the most important area is `-Xms512m`; this limits the maximum amount of heap that MPLAB X IDE will use. While it might seem that having a large number of heap could be helpful, memory used for MPLAB X IDE means less memory for other applications and system functions.

You can monitor how much memory the IDE is using by enabling the Memory monitor. Either right click on an empty space in the toolbar area and select memory, or select *View>Toolbars>Memory*.

The upper number will not go beyond 512 MB unless you change the value in `mplab_ide.conf`. It is recommended that you change the `-Xms512m` setting in 128 MB increments. If you have a lot of memory, you can increase it by more than 128 MB, but make sure you leave enough memory for the rest of the system.

6.2.2 Debug Tool Usage

When using debug tools slows down MPLAB X IDE:

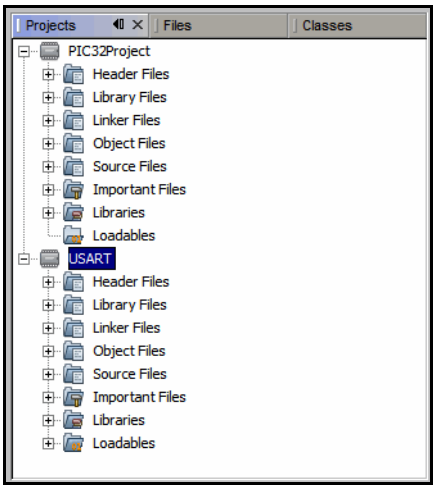
1. Only display windows you need view. The MPLAB X IDE debugger will examine each open window when a debug tool is used.
2. Reduce updates on Stack window during debug stepping. In the *Tools>Options* window, **Embedded** button, **Generic Settings** tab, check "Disable auto refresh for call stack view during debug sessions" and "On mouse-over structure and array expressions, evaluate integral members only".

6.3 WORK WITH MULTIPLE PROJECTS

MPLAB X IDE allows you to work with more than one project.

If you need to work on more than one project at a time, multiple projects may be opened in MPLAB X IDE and viewed in the Projects window. For more on this window, see **Section 12.11 "Projects Window"**.

FIGURE 6-1: MULTIPLE PROJECTS IN THE PROJECTS WINDOW



Active Projects

Projects may be made active by clicking on them in the Projects window.

Main Project

One project is selected as the main project by doing one of the following:

- right clicking the project name and selecting "Set as Main Project".
- selecting *Run>Set Main Project*.

The main project name will then appear in **bold**.

You can also work without setting an active project. In that case the IDE will know which project you are working on by context, i.e., if you set the editor focus on a file, then the project that owns that file will become the project that receives the actions, e.g., when you press the **Debug Run** button. It is your choice whether to explicitly make a project the 'main' project, or you use the editor focus to select the main project.

If you have set a main project and wish to remove it:

1. Right click in an empty area of the Projects view.
2. Use "Set Main Project" to change the active project or to select no active project.

To work with multiple projects:

1. Create (*File>New Project*) or open (*File>Open Project*) two or three projects.
2. Start debugging each project, i.e., click on a project in the Projects window and then select *Debug>Debug Project*.
3. Open the Sessions window (*Window>Debugging>Sessions*) and you can switch between any currently running debug sessions.

When one debug session is switched to another, the Watch window and variables plus the memory will switch to show the currently selected project being debugged. The Status bits should also follow the debug project. The Dashboard will follow the last selected whether it is a debug project or a project in the project window. This is by design.

To group multiple projects:

Another way to work with multiple projects is in groups. Select *File>Project Group* and pick or create a project group.

For more on creating project groups, see the NetBeans help topic Project Group: Create New Group Dialog Box.

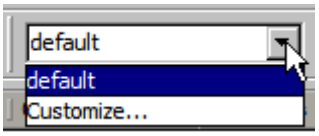
6.4 WORK WITH MULTIPLE CONFIGURATIONS

MPLAB X IDE allows multiple configurations for the same project. This may be useful for code that can be compiled on multiple platforms (such as the Microchip Application Libraries Demo Projects.)

When you create a new project, a “default” configuration is created. To create your own configuration, start by doing one of the following:

- Use the drop down menu on the toolbar and select “Customize” (Figure 6-2). The Project Properties dialog will open.
- Open the Project Properties dialog by right clicking on the project name and selecting “Properties”.

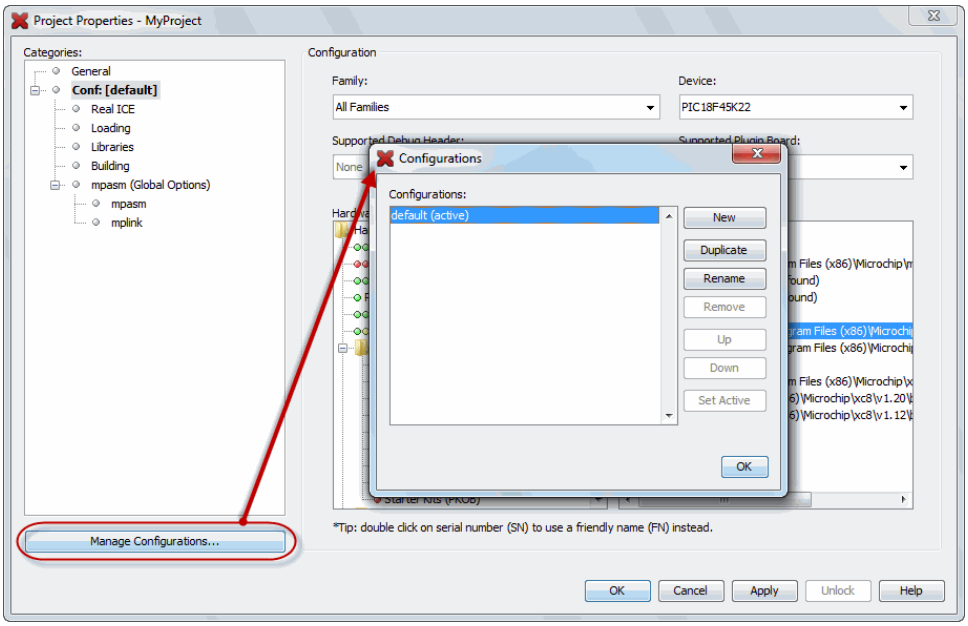
FIGURE 6-2: PROJECT CONFIGURATION DROP-DOWN BOX



In the Project Properties dialog, click **Manage Configurations** to open the Configurations dialog. Existing configurations can be renamed or a new configuration can be added or duplicated from an existing one.

When more than one configuration is created for a project, the active one can be selected from **Manage Configurations** or from the drop-down menu.

FIGURE 6-3: PROJECT PROPERTIES – CONFIGURATIONS DIALOG

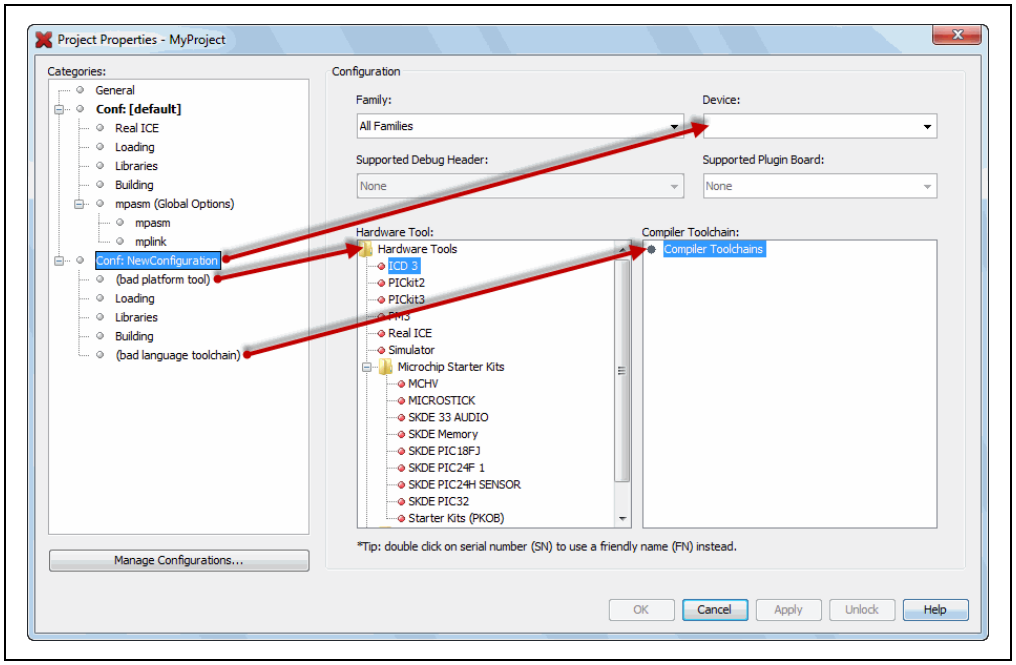


6.4.1 Add a New Configuration

If a new configuration is added, some items must be assigned in the Project Properties dialog.

- Device - You must select this first to see the hardware tool and compiler support.
- Hardware Tool
- Compiler Toolchain

FIGURE 6-4: NEW CONFIGURATION



6.4.2 Add a Duplicate Configuration

You can add a configuration that is the duplicate of an existing one, and then make edits from there.

A good use of a duplicate configuration is for the creation your own debug configuration. Although MPLAB X IDE provides debug macros for use with Microchip tools (**Section 4.16.2 “Debug Macros Generated”**), you may want to use your own debug macros or you may want to set up the same debug capabilities with third party tools.

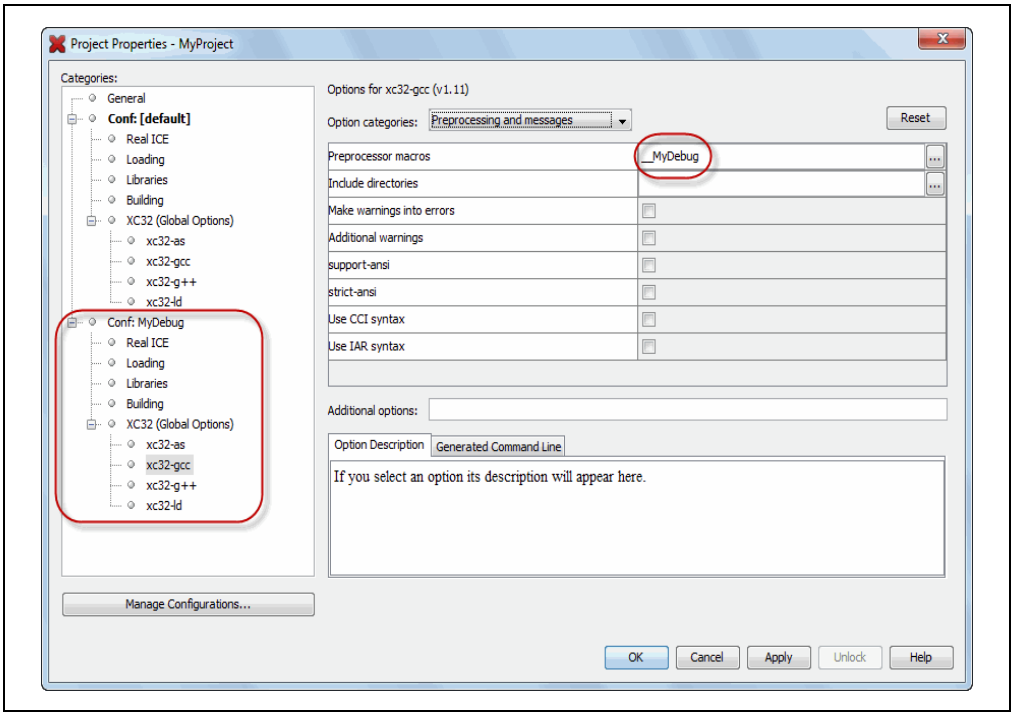
To set up a **debug configuration**:

1. In the Configuration dialog, select a project configuration and click **Duplicate**.
2. Click **Rename** and enter a name in the New Configuration Name dialog, such as “MyDebug”.
3. Click **OK** twice to return to the Project Properties dialog. The Debug configuration (Conf: MyDebug) should now be visible.
4. Click on the compiler in the toolchain. Under the “Preprocessing and messages” options category, locate the option that allows you to define a macro and click on the associated text box.
5. In the pop-up dialog, enter a macro name, such as `__MyDebug`, and click **OK**.

You may now switch to the Debug configuration when you want to debug. You can use the preprocessor macro in conditional text:

```
#ifdef __MyDebug
  fprintf(stderr,"This is a debugging message\n");
#endif
```

FIGURE 6-5: DEBUG CONFIGURATION



6.5 CREATE USER MAKEFILE PROJECTS

Create a project that makes use of an external makefile. This is useful if you have an existing project built outside of MPLAB X IDE, but you now want to use MPLAB X IDE to do debugging.

To create the makefile project, use the New Project wizard.

6.5.1 New Project Setup

To open the New Project wizard, do one of the following:

- On the **Start Page**, click on the **Learn & Discover** tab, "Dive In" section, "Create New Project" link.
- *File>New Project* (or Ctrl+Shift+N)

A wizard with launch to guide you through new project set up.

- **Step 1. Choose Project:** Select the "Microchip Embedded" category and choose from the project type "User Makefile Project".
- **Step 2. Select Device:** Select the device used in your application from the "Device" drop-down list. To narrow your selection list, chose a Family first.
- **Step 3. Select Header:** This step will appear if a header is available for your selected device. To determine if a header is required for debug or if your device has on-board debug circuitry, consult the *Processor Extension Pak and Debug Header Specification* (DS51292 or online help). Then choose whether or not to use a header.
- **Step 3 or 4. Select Tool:** Select the development tool you will be using to debug your application from the list.

  The level of tool support for the selected device will be displayed next to the tool name. That is, green for full support, yellow for beta support and red for no support yet.
- **Step 4 or 5. Create User Makefile Project:** Enter information to set up your makefile project. For details, see **Section 6.5.2 "User MakeFile Project Settings"**.
- **Step 5 or 6. Select Project Name and Folder:** Select a name and location for your new project. You may browse to a location.

The new project will open in the Projects window.

For information on exporting a project as hex, see **Section 12.11.2 "Projects Window – Project Menu"**.

6.5.2 User MakeFile Project Settings

Select Makefile project settings in the “User Makefile Project Settings” window of the New Project wizard. Change Makefile project settings in the Project Properties window, “Makefile” category, after the project is created. Each window has the same items.

TABLE 6-1: USER MAKEFILE PROJECT SETTINGS

Item	Description
Working Directory	Specify the location of the external project. This is the default location for the build, debug build and clean commands.
Build command	When the command to Run is requested in MPLAB X IDE (icon or menu), perform the build according to this command instead.
Debug build command	When the command to Debug Run is requested in MPLAB X IDE (icon or menu), perform the debug build according to this command instead.
Clean command	When the command to Clean is requested in MPLAB X IDE (icon or menu), perform the clean according to this command instead.
Image name	Enter the path and name of the production image file (hex).
Debug image name	Enter the path and name of the debug image file (elf or cof).

Data Entry Instructions:

- Forward (/) slashes are recommended in your path. However, you may use backward (\) slashes instead.
- A path and file name without spaces is recommended. However, if you do have spaces, use escape spaces (backward slash before the space) when you enter a path and file name in this window. GNU make is known to cause weird issues with spaces as spaces are their separators.
- Working Directory: Browsing to the directory is recommended as this will take care of formatting issues for you.
- Command paths (‘Build command’, ‘Debug build command’ and ‘Clean command’):
  - May be absolute or relative to the working directory.
  - For Windows operating system (OS), need to be in quotes (“).
  - For other OSs, do not need to be in quotes, but need any spaces escaped.
- Image paths (‘Image name’ and ‘Debug image name’)
  - Must be absolute.
  - Need any spaces escaped.

6.6 LOG DATA

Log files are useful for capturing your program execution and debugging problems.

When you have an MPLAB X IDE error or issue and need to contact technical support (see "Customer Support"), you should capture data in two log files: MPLAB X IDE log file and the NetBeans platform log file.

6.6.1 MPLAB® X IDE Log File

The MPLAB X IDE log file generates data based on the Java Logger class.

To set up a log file:

1. Select *Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Embedded** button, **Diagnostics** tab.
2. Select a logging level from the drop-down box. For details see **Section 12.12.5 "Diagnostics Tab"**.

> **Note:** The higher the logging level, the more data is collected, but the slower your application will run.

3. Select a location (path) for the log file.

To log data:

1. Set up the log file using logging level "Finest".
2. Take note of the log file name and location.
3. Repeat the steps to cause the error or issue.
4. Find the log file(s) and send to technical support.

6.6.2 NetBeans Platform Log File

The NetBeans log file generates information on the executing NetBeans Platform.

To log data:

5. Open the log file the Output window by selecting *View>IDE log*.
6. Right click in the window and select "Clear".
7. Repeat the steps to cause the error or issue.
8. Right click in the window and select "Save As" to save the text to a file.
9. Send the file to technical support

6.7 CUSTOMIZE TOOLBARS

You can customize MPLAB X IDE toolbars using the Customize Toolbars window. Select *View>Toolbars>Customize* to open the window.

Available icons include Clean Only, Run, Set PC to Cursor, etc.

Add a function to a toolbar:

Drag an icon from the Customize Toolbars window to a toolbar.

Remove a function from a toolbar:

Drag an icon from a toolbar to the Customize Toolbars window.

Add your own toolbar:

Click “New Toolbar” and name the new toolbar.

Change toolbar icon size:

- Check the checkbox “Small Toolbar Icons” to make the icons smaller.
- Uncheck to make the icons larger.

Revert to default toolbar:

Click “Reset Toolbars”.

Available functions are shown in the tables below.

TABLE 6-2: BUILD FUNCTIONS

Function	Details
Make	performs a make (builds project files if they have been updated) without cleaning
Make Clean	cleans (remove previously built project files) and makes the project

TABLE 6-3: DEBUG FUNCTIONS

Function	Details
New Data Breakpoint	sets a new data breakpoint at the address
New Run Time Watch	adds the specified symbol to watch that will change value as the program runs/executes
New Watch	enters an expression or select an SFR to watch in the Watches window
PIC AppIO	opens the PIC AppIO window. Your debug tool and device must support Application I/O See your debug tool documentation for details.
Program Device for Debugging Main Project	programs the device from a debug image The program will immediately begin execution on completion of programming.
Disassembly	opens the disassembly window.
Disconnect from Debug Tool	disconnects communications between MPLAB® X IDE and the debug tool To reconnect, select Run/Debug Run.
Focus Cursor at PC	moves the cursor to the current PC address and centers this address in the window
PC Profiling	opens the PC Sampling window See MPLAB REAL ICE in-circuit emulator documentation for more on PC sampling and profiling.
Reset	resets the device.
Set PC at Cursor	sets the program counter (PC) value to the line address of the cursor

TABLE 6-3: DEBUG FUNCTIONS (CONTINUED)

Function	Details
Step Instruction	executes one machine instruction. If the instruction is a function call, executes the function and returns control to the caller
Status Toolbar Action	displays the status toolbar
Launch Debugger Main Project	launches the debugger for the main project This is the final step of the discrete build process: Build, Program Target, Launch Debugger.It is useful for changing Memory window setting during debug and using starter kits.
Debug Main Project	debugs the main project
New Watch	enters an expression to watch in the Watches window
Attach Debugger	connects communications between MPLAB® X IDE and the debug tool A debug tool is automatically connected during a Run or Debug Run, and disconnected at the end of a run. To keep the debug tool connected at all times, check *Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Embedded** button, **General Settings** tab, "Maintain active connection to hardware tool".
Continue	resumes debugging after "Pause" until the next breakpoint or the end of the program is reached.
Debug File	starts debugging session for currently selected file.
Debug Test File	starts debugging test for file in JUnit. (Java related)
Apply Code Changes	apply any changes in the code to the executing program.
Finish Debugger Session	ends the debugging session.
Make Callee Current	makes the method being called the current call Only available when a call is selected in the Call Stack window.
Make Caller Current	makes the calling method the current call Only available when a call is selected in the Call Stack window.
Pause	pauses debugging Use "Continue" to resume.
Run to Cursor	runs the current project to the cursor's location in the file and stops program execution
Step Into	executes one source line of a program If the line is a function call, executes the program up to the function's first statement and then stops.
Step Over Expression	steps over the expression and then stops the debugging.
Step Out	executes one source line of a program If the line is a function call, executes the function and returns control to the caller.
Step Over	executes one source line of a program If the line is a function call, executes the entire function and then stops.

TABLE 6-4: EDIT FUNCTIONS

Function	Details
Next Bookmark	cycles forward through the bookmarks
Previous Bookmark	cycles backward through the bookmarks
Clear Document Bookmarks	clears all bookmarks in the document
Quick Search	displays the Quick Search toolbar
Find in Projects	finds specified text, object names, object types within projects
Replace in Projects	replaces text, object names, object types within projects
Copy	copies the current selection to the clipboard
Cut	deletes the current selection and places it on the clipboard
Delete	deletes the current selection
Find	finds a text string
Paste	pastes the contents of the clipboard into the insertion point
Redo	reverses (one at a time) a series of Undo commands
Undo	reverses (one at a time) a series of editor actions, except Save

TABLE 6-5: HELP FUNCTIONS

Function	Details
Start Page	displays the Start Page.
Individual Help Files	display individual pop-up help files, e.g., MPLAB® X IDE Help
Help	displays the Help window.

TABLE 6-6: PROFILE FUNCTIONS

Function	Details
Start Sampling IDE	begins sampling for PC Profiling/Sampling See MPLAB® REAL ICE in-circuit emulator documentation for more on PC sampling and profiling.

TABLE 6-7: PROJECT FUNCTIONS

Function	Details
Programmer to Go PICkit 3™ Main Project	uses the Programmer to Go feature of PICkit 3™
Erase Device Memory Main Project	erases device memory for the main project
Hold in Reset	toggles the device between Reset and Run
Make and Program Device Main Project	The main project is built (if necessary) and the device is programmed. The program will immediately begin execution on completion of programming.
Read Device Memory Main Project	transfers what is in target memory to MPLAB® X IDE
Read Device Memory to File	transfers what is in target memory to the specified file
Build Main Project	builds all the files of the main project
Build for Debugging Main Project	builds all the files of the main project for debugging
Clean and Build Main Project	cleans (removes) all generated files and then rebuilds all the files of the main project
Build Project	builds all the files of the selected project
Clean Main Project	cleans (removes) all generated files of the main project
Clean Project	cleans (removes) all generated files of the selected project.
Compile File	compiles the selected file using the project compiler
New File	launches New File wizard
New Project	launches New Project wizard
Open Project	opens an existing project
Clean and Build Project	cleans (removes) all generated files and then rebuilds all the files of the selected project
Run Main Project	The main project is built (if necessary) and the device is programmed. The program will immediately begin execution on completion of programming. Same as "Make and Program Device Main Project".
Run Project	The selected project is built (if necessary) and the device is programmed. The program will immediately begin execution on completion of programming.
Run File	runs the currently selected file
Test Project	starts JUnit test for current project (Java related)
Test File	starts JUnit test for current file (Java related)
Open Required Projects	opens dependent projects
Projects	opens the Projects window
Files	opens the Files window

TABLE 6-8: CVS* FUNCTIONS

Function	Details
Commit	commits local changes to files into the repository
Diff	shows file revisions between repository versions and your local working copies
Show Annotations	displays commit message, author, date, and revision number information in the left margin of files open in the Source Editor
Revert Modifications	reverts local versions of files to versions stored in the repository
Update	updates local versions of files with changes committed to the repository

* CVS is not included with MPLAB® X IDE and must be acquired separately. For more on CVS, see http://www.nongnu.org/cvs.

TABLE 6-9: SYSTEM FUNCTIONS

Function	Details
Next Error	scrolls the Source Editor to the line that contains the next build error
Previous Error	scrolls the Source Editor to the line that contains the previous build error
Run SQL	runs SQL statements and scripts For more on SQL and data base support, see: NetBeans Help>IDE Basics>Servers and Databases.
SQL History	provides a list of SQL statements that have been previously executed in the SQL Editor
Keep Prior Tabs	toggles keeping SQL results tabs from the previous execution open/closed
Open File	opens the Open dialog
Properties	opens the Project Properties window
Save	saves the current file
Save All	saves all open files. If the “Compile on Save” feature is selected, this will also compile/build project files.

TABLE 6-10: TERMINAL FUNCTIONS

Function	Details
org-netbeans-modules-dlight-terminal-action-LocalTerminal(→)	opens the Terminal window
org-netbeans-modules-dlight-terminal-action-RemoteTerminal(→)	opens the Remote Terminal window

TABLE 6-11: TOOLS FUNCTIONS

Function	Details
Check CVS	select CVS version control items For more on CVS, see http://www.nongnu.org/cvs

TABLE 6-12: VIEW FUNCTIONS

Function	Details
Web Browser	opens the NetBeans download page in your default browser
IDE Log	shows the same information as *View>IDE Log*
Classes	opens the Classes window
Customize Zoom	customize zoom on data
Zoom In	zoom in on data
Zoom Out	zoom out on data

TABLE 6-13: WINDOW FUNCTIONS

Function	Details
Debug>Breakpoints	opens the Breakpoints window
Debug>Call Stack	opens the Call Stack window
Debug>Variables	opens the Variables window
Debug>Sessions	opens the Sessions window
Debug>Sources	opens the Sources window
Debug>Threads	opens the Threads window This window lists all threads in the current debugging session.
Debug>Watches	opens the Watches window
Select Document In>Select In Favorites	opens Favorites window and selects current document within it
Terminal	opens the Terminal window
Remote Terminal	opens the Remote Terminal window
Terminal (Experimental)	opens the Experimental Terminal window
Stopwatch	opens the Stopwatch window
Trace	opens the Trace window
Disassembly Listing File	opens the disassembly listing file in an editor window
Project Environment	opens the Project Environment (Dashboard) window
Analyzer	opens the Simulator Analyzer window
Stimulus	opens the Simulator Stimulus window
Services	opens the Services window
Output	opens the Output window
Properties	opens the Project Properties window
Call Graph	opens the Call Graph window
Hierarchy View	opens the Include Hierarchy window This lets you inspect all header and source files that are directly or indirectly included in a source file, or all source and header files that directly or indirectly include a header file.
Macro Expansion View	opens the Macro Expansion window This window expands macros in code.
Thread Map	opens the Threads window
Reporter Result	opens the Exception Reporter window for exception breakpoints
Favorites	opens the Favorites window.
Test Results	opens the unit Test Results window for C/C++ projects
Chat	opens a Team Chat window
Team	opens a team project. For more on team server projects See NetBeans help, "IDE Basics>Collaborative Development".
Navigator	opens the Navigator window

TABLE 6-13: WINDOW FUNCTIONS (CONTINUED)

Function	Details
Palette	opens the Palette window (Java related)
Find Usages Results	opens the Find Usages window
Refactoring Preview	opens a Preview window of refactoring results
Search Results	opens the Search window
Tasks	opens the Tasks window

TABLE 6-14: XML FUNCTIONS

Function	Details
Check File	checks an XML file for well-formedness
Check DTD	opens the DTDs and XML Schemas Manager
Validate File	checks an XML file for syntax errors in accordance with an XML schema or DTD
XSL Transform	transform XML document(s) using an XSL stylesheet

NOTES:

Chapter 7. Editor

7.1 INTRODUCTION

MPLAB X IDE is built upon the NetBeans platform, which provides a built-in editor to create new code or change existing code.

General information on this editor is available under the NetBeans help topic *IDE Basics>Basic File Features*. C compiler information regarding the editor is available under the NetBeans help topic *C/C++/Fortran Development>Working with C/C++/Fortran Projects>Editing Source Files in C/C++/Fortran Projects>About Editing C and C++ Files*. MPLAB X IDE Editor features are discussed below.

- Editor Usage
- Editor Options
- Code Folding
- C Code Refactoring

7.2 EDITOR USAGE

To being using the Editor, you can create or open an existing file using *File>New File* or *File>Open File*, respectively. Additional controls and features are described in the following sections.

7.2.1 Desktop Controls

The following desktop items are associated with the Editor:

- File menu (see **Section 11.2.1 "File Menu"**) to open a file in an Editor window.
- Edit menu (see **Section 11.2.2 "Edit Menu"**) to use edit commands.
- Editor toolbar located at the top of each file's Editor window (see **Section 11.3.9 "Editor Toolbar"**) to access some edit commands.
- Window right click (context) menu for additional commands.

7.2.2 Hyperlinks in C Code

Hyperlink navigation lets you jump from the invocation of a function, variable, or constant to its declaration.

To use a hyperlink, do one of the following:

- Mouse over a function, variable, or constant while pressing <Ctrl> (Windows and Linux) or <Command> (Mac). A hyperlink appears, along with a tooltip with information about the element. Click the hyperlink and the editor jumps to the declaration. Press <Alt> + <Left Arrow> to jump back to the invocation.
- Mouse over an identifier and press <Ctrl> + <B> or <Command> + <B>. The editor jumps to the declaration. Press <Alt> + <Left Arrow> to jump back to the invocation.

Press <Alt> + <Left Arrow> and <Alt> + <Right Arrow> to move backward and forward through the history of the cursor position.

Also, see the NetBeans Help topic:

C/C++/Fortran Development>Working with C/C++/Fortran Projects>Navigating Source Files and Projects>Using Hyperlink Navigation

7.2.3 Hyperlinks in ASM Code

To navigate to header files included in assembly source files, press <Ctrl> (Windows and Linux) or <Command> (Mac) while placing the mouse cursor over the file name referenced by a `#include` statement. Then click the mouse select button to open the include file in its own file tab in the editor.

7.2.4 Editor Features of Note

The following table summarizes some of the more frequently-used features of the editor.

TABLE 7-1: EDITOR FEATURES

Editor Feature	Reference
Unicode is supported.	By default, newly-created projects in the IDE use ISO-8859-1 character encoding. To change this: • Right click the project name in the Projects window and choose Properties. • In the left column under "Categories", select "General". • On the bottom of the page, find "Encoding" and change.
Code is colored based on syntax.	*Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Fonts and Colors** button, **Syntax** tab. Based on Encoding set during Project creation.
Errors are flagged as code is typed.	*C/C++/Fortran Development>Working with C/C++/Fortran Projects>Editing Source Files in C/C++/Fortran Projects>Error Highlighting*
Colored markers provide quick access to multiple symbols, errors, etc.	*Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Fonts and Colors** button, **Annotations** tab
Smart code completion makes suggestions and provides hints.	*Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Editor** button, **Code Completion** tab
Assembly and C code may be collapsed and expanded	**Section 7.4 "Code Folding"**
Right clicking on a function (`delay(x)`) finds usages. This can limit find within a function (e.g., a local `i` variable).	Find Usages dialog
Right clicking on a function (`delay(x)`) shows a call graph The call graph has buttons on the side to switch order, etc.	**Section 5.16 "View The Call Graph"**
Create comments with task keywords (example: `//TODO`) in the source code and tasks will be scanned and added automatically to the Task window.	*IDE Basics>Basic File Features>Working with Task Lists*
File history is available to view recent changes and revert, even without a version control system.	**Section 5.19 "Control Source Code"** – local history
Navigation is simplified with items such as "Go to file", "Go to type", "Go to symbol", "Go to header", and "Go to declaration".	**Section 11.2.4 "Navigate Menu"**
Refactor options (rename functions and variables, find all functions, etc.) for improving code structure.	**Section 7.5 "C Code Refactoring"**

7.3 EDITOR OPTIONS

To set editor options:

1. Select *Tools>Options* (*mplab_ide>Preferences* for Mac OS X) to open the Options dialog.
2. Click on the **Editor** button. Then click on a tab to set up editor features.

Each tab and its options are listed below.

TABLE 7-2: GENERAL TAB

Item	Description
Code Folding	Check to enable Code Folding and select which types of code to fold. For more on Code Folding, see **Section 7.4 "Code Folding"**.
Camel Case Behavior	Check to enable camel case navigation.

TABLE 7-3: FORMATTING TAB

Item	Description
Language	Select the programming language to which the formatting will apply.
Category	Select a category, currently only "Tabs and Indents".
Expand Tabs to Spaces	Check to make tabs into spaces.
Number of Spaces per Indent	Enter the equivalent number of spaces per indent.
Tab Size	Enter the size of a tab.
Right Margin	Enter the size of the right margin.

TABLE 7-4: CODE COMPLETION TAB

Item	Description
Language	Select the programming language to which code completion will apply.
Completion options	Check to enable code completion options. For more on code completion, see the NetBeans help topic *C/C++/Fortran Development>Working with C/C++/Fortran Projects>Editing Source Files in C/C++/Fortran Projects>Using Code Completion*

TABLE 7-5: CODE TEMPLATES TAB

Item	Description
Language	Select the programming language to which the template will apply.
Templates	Enter template information for the language specified above. **Abbreviation:** Enter an abbreviation to type into the editor. **Expanded Text:** After typing the abbreviation and the "Expand template on" character(s), expand the abbreviation to this text in the editor. **Description:** Add an optional description of the template item. For more on code templates, see the NetBeans help topic *C/C++/Fortran Development>Working with C/C++/Fortran Projects>Editing Source Files in C/C++/Fortran Projects>Using Code Templates*
Expand template on	Select character(s) that will be typed into the editor to expand the abbreviated text to the expanded text.

TABLE 7-6: HINTS TAB

Item	Description
Language	Select the programming language to which the hints will apply.
Hint Window	In the Hint window, select an occurrence and specify the "hint" you wish to receive in the event of the occurrence: Error, Warning or Warning on current line.

TABLE 7-7: MACROS TAB

Item	Description
Macros	Add, delete and define editor macros. • To create a new macro, click the **New** button, enter the name for the new macro, then select the new macro in the list and enter the code in the Macro Code editor. • To set a keyboard shortcut for a macro, select a macro from the list, click the **Set Shortcut** button and enter the shortcut in the dialog box. Use this short-cut to run your macro. • To remove a macro, select a macro from the list and click the **Remove** button. • To modify the macro code, select a macro from the list and edit the code in the Macro Code editor.
Macro Code	Click on the macro name above and then type in your macro code. For a list of macro keywords, see: https://netbeans.org/kb/docs/ide/macro-keywords.html **Note:** It is generally easier to add a new macro by recording it than by adding one manually in the Macro Code editor. Use *Edit>Start/Stop Macro Recording*.

7.4 CODE FOLDING

Code folding allows you to hide some sections of code so you can focus on others. Sections of C or assembly code may be collapsed (hidden) or expanded (displayed), depending on your selected options.

- Code Folding Usage
- Custom Code Folding
- Custom Code Folding - ASM
- Expand/Collapse Custom Code Folding

7.4.1 Code Folding Usage

Code folding is enabled by default. For most C or assembly code, sections of code that may be folded are signified by "-" and "+" icons to the left of the code.

- Enable and Set Up Code Folding
- Expand/Collapse Code Folding
- Code Folding – Inline Assembly and MPLAB C18 C

7.4.1.1 ENABLE AND SET UP CODE FOLDING

Use the following steps to enable and set up code folding:

1. Select *Tools>Options* (*mplab_ide>Preferences* for Mac OS X) to open the Options dialog.
2. Click on the **Editor** button.
3. Click on the **General** tab and check "Use code folding".
4. Check other options to specify sections to fold. If you do not see the type of code you want to fold listed there, you can do a custom fold.

7.4.1.2 EXPAND/COLLAPSE CODE FOLDING

Within the editor, click the "-" icon next to a method and it folds. Click the "+" icon next to a folded method and it expands. Folded code is denoted by an ellipsis box. Hold the cursor over the ellipsis box and the IDE displays the collapsed method.

FIGURE 7-1: EXPANDED CODE

```
80   void LCDProcessEvents(void)
81   {
82           switch (_uLCDstate) {
83                   case 1:
84                           if (_uLCDloops) _uLCDloops--;
85                           else _uLCDstate = 0;
86                           break;
87                   case 2:
```

FIGURE 7-2: COLLAPSED CODE

```
80   void LCDProcessEvents(void)
81   {...}
151  /*****************************************************************************
152    * EOF
153    *****************************************************************************
```

7.4.1.3 CODE FOLDING – INLINE ASSEMBLY AND MPLAB C18 C

Code folding does not work for inline assembly code in MPLAB C18 when the `_asm` and `_endasm` directives are used. The work-around is to put the code block near the end of the code/file.

7.4.2 Custom Code Folding

To fold your own area of code, you can use customized code folding.

- Custom Code Folding - C
- Custom Code Folding - ASM
- Expand/Collapse Custom Code Folding

7.4.2.1 CUSTOM CODE FOLDING - C

To custom fold C code, do one of the following:

- Type the following comments around your code:

```
// <editor-fold defaultstate="collapsed" desc="user-description">
C code block to fold
// </editor-fold>
```

OR

- Type `fcom` and press the <Tab> key to automatically enter the comment text above.

After the comment has been entered, customize it for your code:

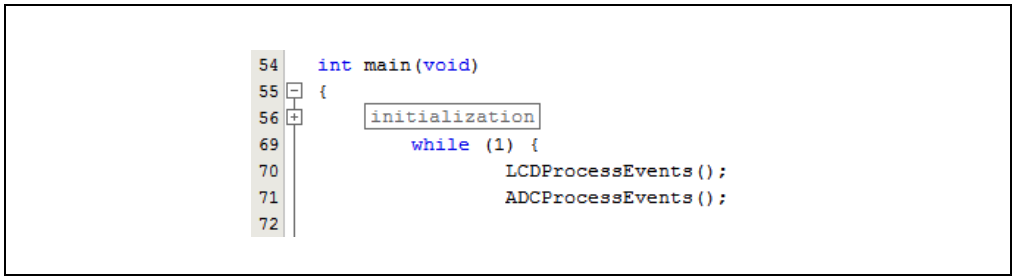
`defaultstate`	Enter either `collapsed` or `expanded`,
`desc`	Enter a description of the code. Once collapsed, this description can still be seen for reference.

An example of custom code folding is shown below.

FIGURE 7-3: EXPANDED CODE

```
54  int main(void)
55  {
56      // <editor-fold defaultstate="collapsed" desc="initialization">
57      _display_state = DISP_HELLO; // Start from displaying of PIC24 banners
58      AD1PCFG = 0xffff; // Setup PortA IOs as digital
59      SPIMPolInit(); // Setup SPI to communicate to EEPROM
60      EEPROMInit(); // Setup EEPROM IOs
61      UART2Init(); // Setup the UART
62      TimerInit(); // Setup the timer
63      mLCDInit(); // Setup the LCD
64      BtnInit(); // Setup debounce processing
65      ADCInit(); // Setup the ADC
66      BannerStart(); // Setup the banner processing
67      RTCCInit(); // Setup the RTCC
68      // </editor-fold>
69          while (1) {
70                  LCDProcessEvents();
71                  ADCProcessEvents();
```

FIGURE 7-4: COLLAPSED CODE



For additional information see the NetBeans help topic *C/C++/Fortran Development>Working with C/C++/Fortran Projects>Editing Source Files in C/C++/Fortran Projects>Folding Blocks of Code*.

7.4.2.2 CUSTOM CODE FOLDING - ASM

To custom fold assembly code, do the following:

- For MPASM assembler or MPLAB XC16 assembler code, type the following comments around your code:

```
; <editor-fold defaultstate="collapsed" desc="user-description">
ASM code block to fold
; </editor-fold>
```

- For MPLAB XC32 assembler code, type the following comments around your code:

```
// <editor-fold defaultstate="collapsed" desc="user-description">
ASM code block to fold
// </editor-fold>
```

OR

- For MPASM assembler or MPLAB XC16 assembler code, type `fcom;` and press the <Tab> key to automatically enter the relevant comment text above.
- For MPLAB XC32 assembler code, type `fcom//` and press the <Tab> key to automatically enter the relevant comment text above.

Assembly code will fold in the same way that C code folds. For an example of code folding, see the C code example above.

7.4.2.3 EXPAND/COLLAPSE CUSTOM CODE FOLDING

To expand/collapse code blocks, go to the View menu, or right-click in the Editor window, and select one of the following:

1. *Code Folds>Collapse Fold* to hide the block of code.
2. *Code Folds>Expand Fold* to display the block of code.
3. *Code Folds>Collapse All* to hide all folding blocks of code.
4. *Code Folds>Expand All* to display all folding blocks of code.

For pictures of unfolded (expanded) and folded (collapsed) code, see Figure 7-3 and Figure 7-4.

7.5 C CODE REFACTORING

> **Note:** To see this feature, refer to the **Start Page**, **My MPLAB IDE** tab, "Extend MPLAB" section, "Selecting Simple or Full-Featured Menus" topic.

Refactoring is the use of small transformations to restructure code without changing any program behavior. Just as you factor an expression to make it easier to understand or modify, you refactor code to make it easier to read, simpler to understand, and faster to update. And just as a refactored expression must produce the same result, the refactored program must be functionally equivalent with the original source.

Some common motivations for refactoring code include:

- Making the code easier to change or easier to add a new feature
- Reducing complexity for better understanding
- Removing unnecessary repetition
- Enabling use of the code for other needs or more general needs
- Improving the performance of your code

The IDE's refactoring features simplify code restructuring by evaluating the changes that you want to make, showing you the parts of your application that are affected, and making all necessary changes to your code. For example, if you use the Rename operation to change a class name, the IDE finds every usage of that name in your code and offers to change each occurrence of that name for you.

- Refactor Menu
- Undoing Refactoring Changes
- Finding Function Usages
- Renaming a Function or Parameter
- Moving, Copying and Safely Deleting C Code

7.5.1 Refactor Menu

When you use the IDE's refactoring operations, you can change the structure of your code and have the rest of your code updated to reflect the changes you have made.

Refactoring operations are available on the Refactor menu (see **Section 11.2.6 "Refactor Menu"**).

7.5.2 Undoing Refactoring Changes

You can undo any changes that you made using the commands in the Refactor menu by following the steps below. When you undo a refactoring, the IDE rolls back all the changes in all the files that were affected by the refactoring.

To undo a refactoring command:

1. Go to the Refactor menu in the main menu bar.
2. Choose Undo.
   The IDE rolls back all the changes in all the files that were affected by the refactoring.

If any of the affected files have been modified since the refactoring took place, the Refactoring Undo is not available.

7.5.3 Finding Function Usages

You can use the Find Usages command to determine everywhere a function is used in your project's source code.

To find where a function is used in your project:

1. In the Navigator window or the Source Editor window, right click the function name and choose Find Usages (Alt-F7).
2. The Find Usages command displays the code lines that call the function. In the Find Usages dialog box, click **Find**.

   The Usages window displays the file name and the line of code for each usage found in that file.

To jump to a specific occurrence of the function, do one of the following actions:

- In the Usages window, use the up and down arrow buttons in the left pane to navigate from one occurrence of the function to the next.
- Double click a line of code to open the file and position the cursor on that particular line of code.

Additional IDE Find Mechanisms

The other IDE tools that enable you to search for all the places where specific text is used in a project include:

- **Finding and Replacing Text**. Searches for all the places where specific text is used in a source file that is open in the Editor. Choose *Edit>Find* to open the Find dialog box, or choose *Edit>Replace* to open the Replace dialog box. These commands finds all matching strings, regardless of whether the string is a C code element.
- **Find in Projects**. As with the Find command, the Find in Projects command searches for matching strings, regardless of whether the string is a function name. Choose *Edit>Find in Projects* to open the "Find in Projects" dialog box and then type the string of text that you are looking for.

To find where a function is declared in a source file, you can double click the function in the Navigator window. If the function is declared in a different source file, right click the function and choose *Navigate>Go To Declaration* from the contextual menu.

7.5.4 Renaming a Function or Parameter

To rename a function or parameter and update references to it throughout your project:

1. In the Source Editor, right click the function or parameter and choose *Refactor>Rename* from the context menu.
2. In the Rename dialog box, type the New Name.
3. Optionally, click **Preview**. In the Refactoring window, at the bottom of the Source Editor, review the lines of code that will be changed and clear the checkbox of any code that you do not want changed.
4. Click **Do Refactoring** to apply the selected changes.

For quick in-place renaming, place the cursor in the item that you want to rename, and press Ctrl-R. Then type the new name. To finish renaming, press **Escape**.

7.5.5 Moving, Copying and Safely Deleting C Code

These functions are specific to C++ code. See **Section 11.2.6 "Refactor Menu"**.

NOTES:

Chapter 8. Project Files and Folders

8.1 INTRODUCTION

There are several ways to view project files and folders in MPLAB X IDE:

- Projects Window View
- Files Window View
- Favorites Window View
- Classes Window View

Other MPLAB X IDE file and folder information includes:

- Viewing User Configuration Data
- Importing an MPLAB IDE v8 Project – Relative Paths
- Moving, Copying or Renaming a Project

8.2 PROJECTS WINDOW VIEW

Project folders viewed in the Projects window are logical (virtual) folders. For more on this window, see **Section 12.11 "Projects Window"**.

At the least, you will need to add source files. MPLAB X IDE will find many default files for you (header files, linker scripts). You may add library and precompiled object files, as well as edited header and linker script files. Files that will not be included in the build may be placed under "Important Files".

FIGURE 8-1: PROJECTS WINDOW – SIMPLE C CODE PROJECT

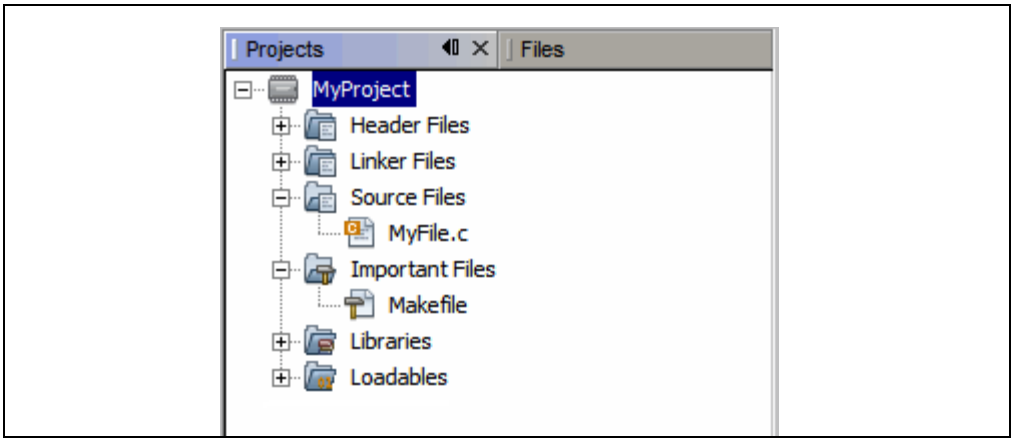


TABLE 8-1: PROJECTS WINDOW DEFINITIONS

Virtual Folder	Files Contained
`Header Files`	header files (`.h` or `.inc`)
`Library Files`	library files (`.a` or `.lib`)
`Linker Files`	linker files (`.ld`, `.gld` or `.lkr`)
`Object Files`	precompiled object files (`.o`)
`Source Files`	source files (`.c` or `.asm`)
`Important Files`	important files, such as makefiles Other documents can be placed here, such as data sheet PDFs.

8.3 FILES WINDOW VIEW

Project folders viewed in the Files window are actual folders (folders on your personal computer). For more on this window, see the NetBeans help topic "Files Window" as used with C code projects.

FIGURE 8-2: FILES WINDOW - SIMPLE C CODE PROJECT

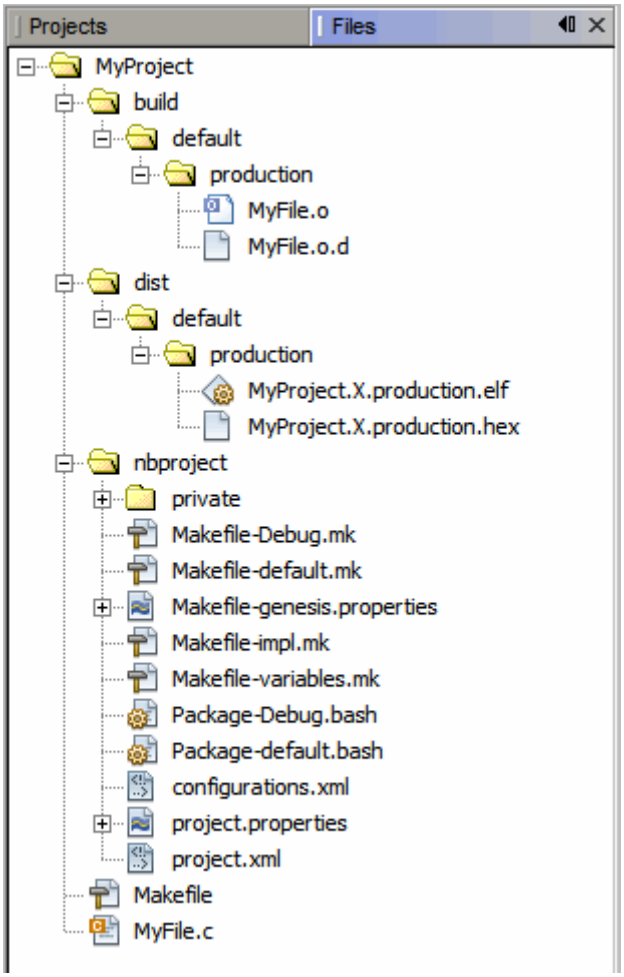


TABLE 8-2: FILES WINDOW DEFINITIONS

Folder	Description
`MyProject`	The project folder, which contains the `Makefile` file and C code or assembly application files. `Makefile` is the master makefile for the project. This file is generated at project creation time and it is not touched after that (it will not be regenerated). You can make changes to this file if you are familiar with GNU make. But MPLAB X IDE provides ways to add a pre-step and post-step (Project Properties) which can be used instead of modifying the `Makefile` itself.
`build`(1)	The intermediate files folder. Files are contained in subfolders depending on project configuration, usage and location. The build files are: • Run files (`.o`) • Dependency files (`.o.d`) • HI-TECH® intermediate files (`.p1`)
`dist`(1)	The output files folder. Files are contained in subfolders depending on project configuration, usage and location. The distribution files are: • Executable files (`.hex`) • ELF or COFF object files (`.elf` or `.cof`) • library files (`.a` or `.lib`)
`nbproject`	The makefile and metadata folder. Contains these files: • Project makefile • Configuration-specific makefiles • `project.xml` IDE-generated metadata file • `configurations.xml` metadata file
`default`, `MyConfig`(2)	The project configurations folders. If no configurations are created by the developer, all code is in `default`.
`production`, `debug`(2)	The production and debug versions folders.
`_ext`(2)	The external project files folder. If a file is referenced outside the project folder, it is listed here.

Note 1: You do not need to check these folders into source control. A build will create them. See also **Section 5.19 "Control Source Code"**.

2: These items are not shown in Figure 8-2.

8.4 FAVORITES WINDOW VIEW

The Favorites window (*Window>Favorites*) enables you to access any file or folder on your computer or network, whether or not it is in a project.

When you first open the Favorites window, it only contains your computer's home directory.

- To add a file or folder, you can right click in the Favorites window and choose "Add to Favorites".
- To add a file, you can right click on the file name in the Project window and select *Tools>Add to Favorites*.

For more on this window, see the NetBeans help topic "Favorites Window".

8.5 CLASSES WINDOW VIEW

For compilers that support C++, you can view the class files in your project using the Classes window (*Window>Classes*).

For more on this window, see the NetBeans help topic "C/C++/Fortran Development>Working with C/C++/Fortran Projects>Navigating Source Files and Projects>Using the Classes Window" as used with C++ code projects.

8.6 VIEWING USER CONFIGURATION DATA

The MPLAB X IDE user directory (userdir) stores user configuration data such as window layouts, editor settings, menu and toolbar customizations, and various module settings such as the list of known compilers. In addition, when you install plugins (*Tools>Plugins*), the plugin information (code) is stored in the userdir, not in the MPLAB X installation directory.

To find out where your user directory is located, select *Help>About* and find the path next to **Userdir**.

This information is created and managed by MPLAB X IDE.

8.7 IMPORTING AN MPLAB IDE V8 PROJECT – RELATIVE PATHS

For an MPLAB IDE v8 project located at:

`C:\MyProjects\mplab8project`

On importing to MPLAB X IDE you will get:

`C:\MyProjects\mplab8project\mplab8project.X`

The MPLAB X IDE imported project is placed by default under the MPLAB IDE v8 project to preserve maintainability of both projects. However, you can place the MPLAB X IDE project folder anywhere. Also, the name of the project will be set as *`mplab8project`*`.X` but you can rename it if you wish. There is a convention to finish the name of an MPLAB X IDE project with `.X` but this is just a convention.

The new project will not copy the source files into its folder, but instead will reference the location of the files in the v8 folder. To create an independent MPLAB X IDE project, create a new project and copy the MPLAB IDE v8 source files to it.

For details, see **Section 5.2 “Import MPLAB Legacy Project”**.

8.8 MOVING, COPYING OR RENAMING A PROJECT

Once you have created a project, you may realize you need to make changes. Using the commands on the context (right click) project menu, you may move, copy or rename your project from within MPLAB X IDE. There is also an option to delete the project. For details, see.**12.11.2 “Projects Window – Project Menu”**

You can also use an external tool; the project file structure does not require you to use MPLAB X IDE.

8.9 DELETING A PROJECT

To delete a project in MPLAB X IDE:

1. In the Projects window, right click on the project name and select “Delete”.
2. In the “Delete Project” dialog, you can delete the project file from the computer or the project file with all of the source files. Make your selection and click **Yes**.
3. The project will be deleted so that MPLAB X IDE can no longer see it. However, some makefile information will remain on the computer.

To delete all project files from the computer:

1. Delete or close your project in MPLAB X IDE.
2. Exit MPLAB X IDE (see Note below).
3. Delete the files.

> **Note:** MPLAB X IDE does not release all resources assigned to a project when closed. MPLAB X IDE runs on Java so releasing resources does not occur immediately. Under Java it is best to let the JRE decide when to call the garbage collection (GC) within the software code. However, you can force GC by enabling the memory toolbar and clicking on it.

NOTES:

Chapter 9. Troubleshooting

9.1 INTRODUCTION

This section is designed to help you troubleshoot any problems, errors or issues you encounter while using MPLAB X IDE. If none of this information helps you, see "Support" for ways to contact Microchip Technology.

- USB Driver Installation Issues
- Cross-Platform Issues
- MPLAB X IDE Issues
- NetBeans Platform Issues
- Errors
- Forums

9.2 USB DRIVER INSTALLATION ISSUES

To install the correct USB drivers, see **Section 2.3 "Install the USB Device Drivers (For Hardware Tools)"**.

To troubleshoot errors, see **Section 2.3.2.7 "Tool Communication Issues"**.

9.3 CROSS-PLATFORM ISSUES

If you plan on using MPLAB X IDE on different platforms (Windows, Mac, or Linux operating systems), be aware of these issues:

- Use the forward slash "/" in relative paths. The backslash "\" works only on Windows OS platforms. Example: `#include headers/myheader.h`.
- Linux OS is case-sensitive, so `genericTypeDefs.h` is not the same as `GenericTypeDefs.h`.

9.4 MPLAB X IDE ISSUES

Importing an MPLAB IDE v8 Project - Settings

Settings that were saved in a workspace in MPLAB IDE v8 (such as tool settings) will not be transferred to the new MPLAB X IDE project. Refer to the MPLAB IDE v8 help for what is stored in a workspace (*MPLAB IDE Reference>Operational Reference>Saved Information*.)

Importing an MPLAB IDE v8 Project - Modified Linker Scripts

If you have modified your MPLAB IDE v8 project linker script so that it includes an object file, the linker will be unable to find the file when imported into MPLAB X IDE because the build paths for MPLAB IDE v8 and MPLAB X IDE are different.

So you may see an error like:

```
<install path>ld.exe: cannot find file.o
```

since in MPLAB IDE v8 all build-related files are in one directory, whereas in MPLAB X IDE build files are in different subdirectories.

You can edit your linker script to work with MPLAB X IDE by using wild cards. For example, change:

```
/* Global-namespace object initialization - MPLAB v8*/
.init   :
{
  KEEP (crti.o(.init))
 :
} >kseg0_program_mem
```

to:

```
/* Global-namespace object initialization - MPLAB X*/
.init   :
{
  KEEP (*crti.o(.init))
 :
} >kseg0_program_mem
```

Alternatively, you can use an address attribute that allows you to place functions in C code.

```
int __attribute__((address(0x9D001000))) myfunction (void) {}
```

This allows you to place a function at an address without modifying the default linker script.

9.5 NETBEANS PLATFORM ISSUES

The NetBeans platform may have issues concerning the platform release used for MPLAB X IDE. For more help, visit the NetBeans web site (www.netbeans.org).

See also:

http://netbeans.org/community/releases/index.html

9.6 ERRORS

Errors can take many forms in the IDE, most commonly as icons in windows or messages in the Output window. Hovering over icons will pop up text that may explain the issue. For text messages, please refer to online help to search for the error.

Some errors are listed below.

Command line too long (Windows XP and later)

When a project's makefile is run, each command that is executed is passed to the local native shell. For Windows XP and later operating systems, there is a limit of 8191 characters. Therefore, for a project with hundreds of c files that link, the limit could be surpassed and this error will be displayed.

Linux OS and Mac OS X can usually take very long command lines. If you want to find out how long for your system, you can type on the command line `getconf ARG_MAX`.

Response File Workaround

If you are using either the MPLAB XC32 C compiler or MPLAB XC8 C compiler v1.01 and above, you may be able to use a response file when calling the linker to work around this issue.

For MPLAB XC8, in the Project Properties window (*File>Project Properties*), click in "Categories" on "XC8 linker". Under "Option categories" select "Additional Options". Check the checkbox next to "Use response file to link".

For MPLAB XC32, in the Project Properties window (*File>Project Properties*), click in "Categories" on "xc32-ld". Under "Option categories" select "General". Check the checkbox next to "Use response file to link".

Library Workaround

For all other compilers, you can create an MPLAB X IDE library project and move some source files from your main project to the library project. Then add the library project to your main project.

To create a library project select *File>New Project*, click the "Microchip Embedded" category, and then select "Library project" as the project.

To add the library to your main project, open the Project Properties window i.e., *File>Project Properties*), click in "Categories" under "Libraries", and then click the **Add Library Project** button.

Couldn't reserve space for Cygwin™ heap (Windows 7 or 8)

MPLAB X IDE uses the Cygwin MinGW in its make process. In Windows 7 or 8, you may have virtual memory allocation issues.

To change the Virtual Memory settings, click *Start>Control Panel*. Then click System and then *Security>System>Advanced Systems Setting or System Protection*. On the **Advanced** tab, click on the Performance **Settings** button. In this dialog click the **Advanced** tab and then click on the **Change** button. Enter a custom size value as follows:

- Initial Size (MB) = Currently Allocated (shown at the bottom)
- Maximum Size (MB) = Recommended (shown at the bottom)

Click **Set**, click **OK**, and reboot your personal computer.

Could not access the URL through the external browser. Check the browser configuration

In MPLAB X IDE, select *Tools>Options*, **General** Tab. In the "Web Browser" drop-down list, select your browser. Click **OK**.

For more information see:

Bug 21236 - Ext-Browser: Message "Could not access the URL" to guide from Tools menu (http://netbeans.org/bugzilla/show_bug.cgi?id=21236)

Bug 38211 - A problem with using IE external browser in the IDE (http://netbeans.org/bugzilla/show_bug.cgi?id=38211)

9.7 FORUMS

If you do not see your issue here, check out our forums at:

http://www.microchip.com/forums/f238.aspx

Here you will find discussions and recently posted solutions to problems.

Chapter 10. MPLAB X IDE vs. MPLAB IDE v8

10.1 INTRODUCTION

MPLAB X IDE differs considerably from MPLAB IDE v8 and before. The topics in this section will help you with migration issues.

- Major Differences
- Menu Differences
- Tool Support Differences

10.2 MAJOR DIFFERENCES

Because MPLAB X IDE is based on the NetBeans platform, many features will be different from MPLAB IDE v8. Please see the following for more detailed information:

- **Chapter 2. "Before You Begin"**
- **Chapter 3. "Tutorial"**
- **Chapter 5. "Additional Tasks"**

A short list of major feature differences is presented below:

1. **MPLAB X IDE is based on the open-source, cross-platform NetBeans platform.** Third parties can easily add functionality as plug-ins. MPLAB X IDE components that are specific to Microchip products are still proprietary.

   MPLAB IDE v8 is proprietary and Windows operating system based. Third parties can add to v8 with design information from the MPLAB development group.

2. **MPLAB X IDE is project-based (no workspaces).** In MPLAB X IDE, you must create a project to develop your application. Creating a project involves selecting a device, as well as selecting and setting up language tools, debug tools, programming tools and other project specifics. This ensures all items needed for successfully developing an application are present. Multiple project grouping is handled by multiple configurations.

   MPLAB IDE v8 is device-based. Although it is always highly recommended that you use a project in v8 to create your application, it is not required. Workspaces are used to contain some set up information, including multi-project grouping.

   As projects have changed significantly, reading **Chapter 3. "Tutorial"** is recommended.

3. **MPLAB X IDE allows multiple tool selection**

   Example 1: Connect several MPLAB ICD 3 debuggers into several computer USB ports. Then access the Project properties to easily switch between the debuggers, which are identified by their serial numbers (SN).

   Example 2: Connect one MPLAB ICD 3 debugger into a computer USB port and one MPLAB PM3 programmer into another USB port. Then access the Project properties to easily switch between the tools.

   MPLAB IDE v8 does not allow multiple tool selection.

4. **MPLAB X IDE allows multiple language tool version selection**

   Example: Install two versions of the MPLAB C Compiler for PIC18 MCUs. Then access the Project properties to easily switch between versions of compiler toolchains.

   MPLAB IDE v8 does not allow multiple language tool version selection.

5. **MPLAB X IDE allows multiple debug and programming sessions.** MPLAB X IDE allows you may have multiple debug sessions active in one IDE. For more information, see **Section 6.3 "Work with Multiple Projects"**.

   MPLAB IDE v8 allowed one debug or programming session. MPLAB IDE v8 allowed you to have multiple projects open in the IDE. However, you could only debug or program with one project at a time.

6. **MPLAB X IDE allows multiple build configurations.** MPLAB X IDE allows more than one build configuration. For more information, see **Section 6.4 "Work with Multiple Configurations"**.

   MPLAB IDE v8 allowed only two configurations. MPLAB IDE v8 allowed you to select only between "Release" or "Debug" from the Build Configuration drop-down box and have use of `__DEBUG` in your own code.

   To recreate the MPLAB IDE v8 functionality in MPLAB X IDE, you can create your own Debug configuration and `__DEBUG` macro by following the example in **Section 6.4.2 "Add a Duplicate Configuration"**.

7. **MPLAB X IDE provides multi-step options to debug and program.** MPLAB X IDE has a "Debug Project" icon that builds, programs a target device with your program and a debug executive (for hardware tools) and runs your code in Debug mode in one click. Also available is a "Make and Program" icon that builds, programs a target device (for hardware tools) and runs your code in one click. If you do not want your program to run after make and program, use the "Hold in Reset" icon instead.

8. **MPLAB IDE v8 required several manual steps to debug or program.** MPLAB IDE v8 had a procedure that required completion before debugging or running code:
   a) select the correct build configuration (Release or Debug)
   b) build/make your code
   c) program the target with the code (for hardware tools)
   d) run your code.

   For some tools, e.g., MPLAB Starter Kits, you still need to perform some steps independently. MPLAB X IDE provides this functionality under *Debug>Discrete Debugger Operation*.

9. **MPLAB X IDE uses configuration bits set in code.** MPLAB X IDE requires that Configuration bits be set in code. However, you may temporarily change Configuration bits in the Configuration bits window when in a debug session and then save these setting to a file to paste into your code later (see **Section 4.21.4 "Set Configuration Bits"**.)

   MPLAB IDE v8 used configuration bits set in code or a window. MPLAB IDE v8 allowed you to set Configuration bits in either code or the Configuration bits window. However, settings made in the window had to be manually entered into code.

10. **MPLAB X IDE debug tools are only connected during a session.** MPLAB X IDE only connects debug or programmer tools to the target during a debug or programming session. Otherwise they are not connected.

    MPLAB IDE v8 debug tools were always connected. MPLAB IDE v8 connected to the debug and programmer tools as soon as the tool was selected. This configuration did not allow for multiple sessions.

    To maintain this connection at all times in MPLAB X IDE, like MPLAB IDE v8, go to *Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Embedded**, **Generic Settings** tab, and check “Maintain active connection to hardware tool”.

11. **MPLAB X IDE information is consolidated in one display.** MPLAB X IDE has a Dashboard window that contains breakpoint resources, checksum and memory gauge information. See **Section 5.17 “View the Dashboard Display”**. This window organizes this and future information in one place.

    MPLAB IDE v8 information is spread over several displays. MPLAB IDE v8 had a breakpoint resources toolbar, checksum toolbar and memory gauge window. Each feature was accessed differently.

12. **MPLAB X IDE has many NetBeans features.** MPLAB X IDE has many NetBeans editing and debug features. (See NetBeans Help for more details.) Periodically, MPLAB X IDE will update the NetBeans platform it is based upon. Then the IDE will be updated to the new NetBeans features. The MPLAB X IDE release notes will specify the NetBeans platform version that each version of MPLAB X IDE is built upon.

    MPLAB IDE v8 had its own, proprietary features. MPLAB IDE v8 was a proprietary product. As such, third-party and community development was more difficult.

10.3 MENU DIFFERENCES

The following tables highlight the menu changes from MPLAB IDE v8 to MPLAB X IDE. Due to the changes in MPLAB X IDE, not all MPLAB IDE v8 menu items will map identically. Major differences are discussed under Comments.

Additional menu items may be available in MPLAB X IDE. See the NetBeans help topic "IDE Basics" for details.

TABLE 10-1: FILE MENU DIFFERENCES

MPLAB® IDE v8	MPLAB® X IDE	Comments
New	File>New File	Select associated project in file wizard.
Add New File to Project		
Open	File>Open File	
Close	File>Close	
Save	File>Save	
Save As	File>Save As	
Save All	File>Save All	
Open Workspace	Not available	All data saved in projects
Save Workspace		
Save Workspace As		
Close Workspace		
Import	File>Import	Useful for Hex Import
Export	Right click on file in project in Project tabbed window, select 'Export Hex'	
Print	File>Print File>Print to HTML File>Page Setup	
Recent Files	File>Open Recent File	
Recent Workspaces	Not available	All data saved in projects
Exit	File>Exit	

TABLE 10-2: EDIT MENU DIFFERENCES

MPLAB® IDE v8	MPLAB® X IDE	Comments
Undo	Edit>Undo	
Redo	Edit>Redo	
Cut	Edit>Cut	
Copy	Edit>Copy	
Paste	Edit>Paste Edit>Paste Formatted	
Delete	Edit>Delete	
Select All	Edit>Select All Edit>Select Identifier	
Find	Edit>Find Edit>Find Selection	
Find Next	Edit>Find Next Edit>Find Previous	
Find In Files	Edit>Find in Projects Edit>Replace in Projects	
Replace	Edit>Replace	
Go To	Navigate>Go to Line Navigate>Go to Declaration	
Go To Locator	Navigate>Go to Symbol	
Go Backward	Navigate>Back	
Go Forward	Navigate>Forward	
External DIFF	Tools>Diff	See also Tools>Options (mplab_ide>Preferences for Mac OS X), Miscellaneous, **Diff** tab
Advanced	Source>Format Source>Shift Left/Right Source>Move Up/Down Source>Toggle Comment	
Bookmarks	Navigate>Toggle Bookmark Navigate>Next Bookmark Navigate>Previous Bookmark	
Properties	Tools>Options>**Editor** tab Tools>Options>Fonts & Colors	Use mplab_ide>Preferences instead of Tools>Options for Mac OS X.

TABLE 10-3: VIEW MENU DIFFERENCES

MPLAB® IDE v8	MPLAB® X IDE	Comments
Project	Window>Projects	
Output	Window>Output>Output	
Toolbars	View>Toolbars	
CPU Registers*	Window>PIC Memory Views>CPU	
Call Stack	Window>Debugging>Call Stack	
Disassembly Listing	Window>Output>Disassembly Listing File	
EEPROM	Window>PIC Memory Views>EE Data Memory	Customize in window
File Registers	Window>PIC Memory Views>Data Memory	
Flash Data	Window>PIC Memory Views>Data Memory	
Hardware Stack	Debug>Stack	
LCD Pixel	Not yet available	
Locals	Window>Debugging>Variables	
Memory*	Window>PIC Memory Views>Other Memory	Customize in window
Program Memory	Window>PIC Memory Views>Flash Memory	
SFR/Peripherals*	Window>PIC Memory Views>Peripherals	
Special Function Registers	Window>PIC Memory Views>SFRs	
Watch	Window>Debugging>Watches	See also: Debug>New Watch
Memory Usage Gauge	Window>Dashboard	Memory usage is in the PE window
Trace	Window>Debugging>Trace	See also: File>Project Properties to enable trace

* PIC32 MCUs Only

TABLE 10-4: PROJECT MENU DIFFERENCES

MPLAB® IDE v8	MPLAB® X IDE	Comments
Project Wizard	File>New Project	Always invoked for new project
New	File>New Project	
Open	File>Open Project File>Open Recent Project	
Close	File>Close Project	
Set Active Project	Run>Set Main Project	
Quickbuild	Not supported	A project is required for all development.
Package in .zip	Right click on project in Project tabbed window, select ‘Package’	
Clean	Right click on project in Project tabbed window, select ‘Clean and Build’	
Locate Headers	Right click on project name in Projects window and select “Locate Headers”	
Export Makefile	Not yet available.	
Build All	Run>Run Project Debug>Debug Project	Programmer build/run Debug build/run
Make		
Build Configuration	File>Project Group Run>Set Project Configuration	There used to be only two configurations: Release and Debug. Now you determine how many you need.
Build Options	File>Project Properties	Language tool set up
Save Project	File>Save File>Save All	
Save Project As	File>Save As	
Add Files to Project	Right click on folder in project in Project tabbed window, select ‘Add Existing Item’	
Add New File to Project	File>New File	
Remove File from Project	Right click on file in project in Project tabbed window, select ‘Remove From Project’	
Select Language Toolsuite	File>Project Properties	
Set Language Tool Locations	Tools>Options, Embedded	Use mplab_ide>Preferences instead of Tools>Options for Mac OS X.
Version Control	Tools>Options, Miscellaneous, **Versioning** tab	Use mplab_ide>Preferences instead of Tools>Options for Mac OS X. See also Team menu, Window>Versioning

TABLE 10-5: DEBUGGER MENU DIFFERENCES

MPLAB® IDE v8	MPLAB® X IDE	Comments
Select Tool	File>New Project File>Project Properties	to select to change
Clear Memory	not yet available	used by programmers
Run	Run>Run Project Debug>Debug Project Debug>Continue	Programmer Run Debug Run Run from Halt
Animate	not supported	
Halt	Debug>Pause	
Step Into	Debug>Step Into	
Step Over	Debug>Step Over	
Step Out	Debug>Step Out	
Reset	Debug>Reset	
Breakpoints	Debug>New Breakpoint Debug>Toggle Breakpoint	
Settings	File>Project Properties	
Stopwatch	Window>Debugging>Stopwatch	

TABLE 10-6: PROGRAMMER* MENU DIFFERENCES

MPLAB® IDE v8	MPLAB® X IDE	Comments
Select Programmer	File>New Project File>Project Properties	to select to change
Enable Programmer	not available	once selected, it is enabled
Disable Programmer		
Program	Program Target Project	on Run toolbar
Verify	not available	see the MPLAB IPE
Read	Upload Target Project	on Run toolbar
Blank Check All	not available	see the MPLAB IPE
Blank Check OTP	not available	see the MPLAB IPE
Erase Flash Device	available on custom toolbar	see **Section 6.7 "Customize Toolbars"**
Reset Program Statistics	not available.	
Download OS	File>Project Properties, debug tool category, Firmware option	The IDE will automatically choose the most up-to-date firmware to download
Settings	File>Project Properties	

* For programmers, see also the MPLAB IPE.

TABLE 10-7: TOOLS MENU DIFFERENCES

MPLAB® IDE v8	MPLAB® X IDE	Comments
Data Monitor and Control Interface (DMCI) and other plugins	Tools>Plugins Tools>Embedded	to install to use
MPLAB® Macros	Edit>Start Macro Recording Edit>Stop Macro Recording	
RTOS Viewer	Tools>Plugins Tools>Embedded	to install to use

TABLE 10-8: CONFIGURE MENU DIFFERENCES

MPLAB® IDE v8	MPLAB® X IDE	Comments
Select Device	File>New Project File>Project Properties	to select to change
Configuration Bits	Window>PIC Memory Views>Configuration Bits	
External Memory	Window>PIC Memory Views>Other Memory	customize in window
ID Memory	Window>PIC Memory Views>Other Memory	customize in window
Settings	Tools>Options	Use mplab_ide>Preferences instead of Tools>Options for Mac OS X.

TABLE 10-9: WINDOW MENU DIFFERENCES

MPLAB® IDE v8	MPLAB® X IDE	Comments
Close All	Window>Close All Documents	
Cascade	not supported.	For document management, see Window>Documents.
Tile Horizontally		
Tile Vertically		
Arrange Icons		
Window Sets		
Create Window Set		
Destroy Window Set		
Recent Windows		

TABLE 10-10: HELP MENU DIFFERENCES

MPLAB® IDE v8	MPLAB® X IDE	Comments
Topics	Help>Contents	To view PDFs, see **Section 8.2 "Projects Window View"**.
Release Notes	Help>Online Docs and Support	
Driver Installation	Help>Online Docs and Support	
Check for Updates	not available	Check the Microchip website for updates.
Web Links	Help>Online Docs and Support	
About MPLAB® IDE	Help>About	

10.4 TOOL SUPPORT DIFFERENCES

The following table lists the available Microchip development tools and whether or not they will be supported in MPLAB X IDE.

TABLE 10-11: TOOL SUPPORT DIFFERENCES

Development Tool	In MPLAB® X IDE?	
	Yes	No
PICkit 3	X	
PICkit 2	X	
PICkit 1		X
MPLAB ICD 3	X	
MPLAB ICD 2		X
MPLAB REAL ICE in-circuit emulator	X	
MPLAB ICE 2000		X
MPLAB ICE 4000		X
MPLAB PM3	X	
PRO MATE II		X
PICSTART® Plus		X
MPLAB VDI		X

TABLE 10-12: PLUGIN SUPPORT DIFFERENCES

Plug-Ins	In MPLAB® X IDE?	
	Yes	No
DMCI	X	
dsPIC® Filter Design	X	
dsPIC30F SMPS Buck Converter	X	
dsPIC33F SMPS Buck/Boost Converter*	X	
dsPICworks™	X	
Graphical Display Designer	X	
MATLAB®	X	
Memory Starter Kit	X	
PC-Lint	X	
RTOS Viewer	X	
Segmented Display Designer*	X	
AN851 Bootloader		X
AN901 BLDC Tuning Interface		X
AN908 ACIM Tuning Interface		X
KEELOQ® Plugin		X

* Under development

Chapter 11. Desktop Reference

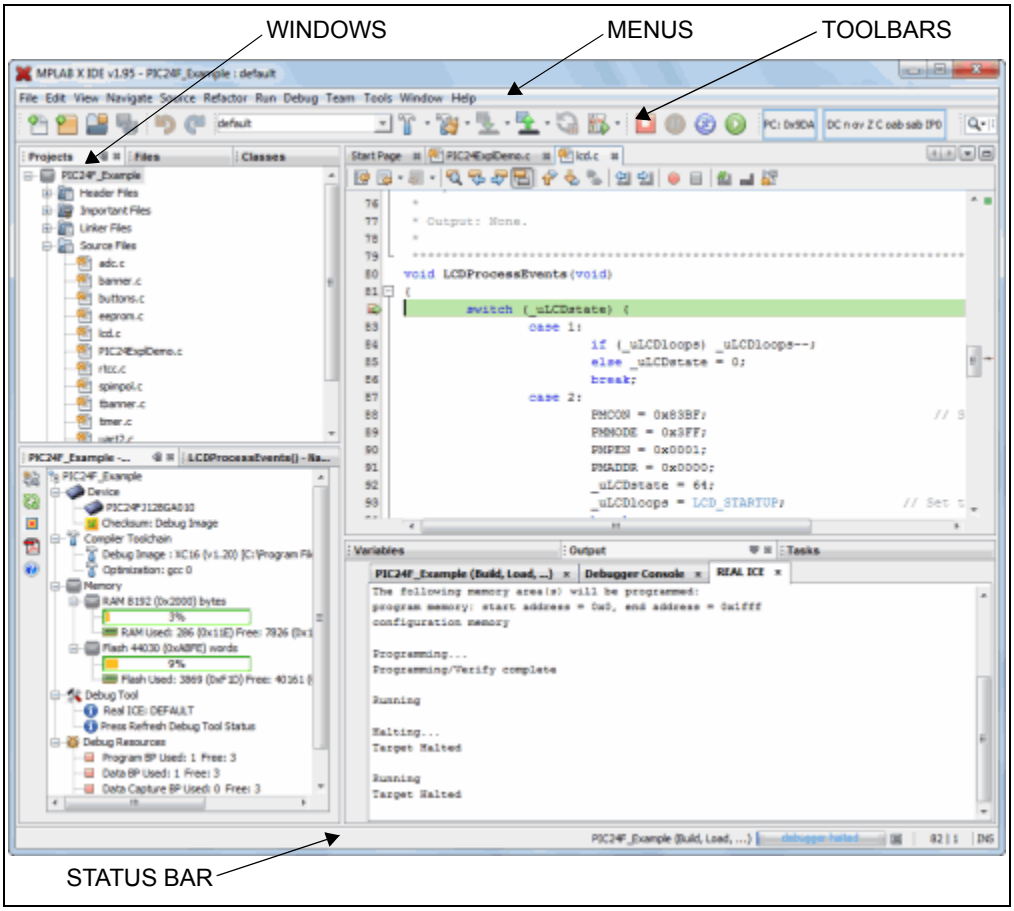
11.1 INTRODUCTION

The MPLAB X IDE desktop is a resizable window that operates independent of the rest of the desktop items. The desktop consists of menus, toolbars, a status bar and tabbed windows (Figure 11-1). Menus, toolbars and the status bar are discussed here. Windows and dialogs are discussed in their own section.

Note: When the NetBeans help topic refers to a workspace, it is talking about a desktop. It is ***not*** referring to the MPLAB IDE v8 (and earlier) workspace.

- Menus
- Toolbars
- Status Bar
- Grayed out or Missing Items and Buttons

FIGURE 11-1: MPLAB IDE DESKTOP



11.2 MENUS

Many MPLAB IDE functions are accessible as menu items through the menu bar located across the top of the desktop. Menu items followed by ellipses (...) will open a dialog. For more on dialogs, see **Chapter 12. "MPLAB X IDE Windows and Dialogs"**.

Shortcut keys for menu items are listed next to the menu item. Example: The shortcut keys for "New File" are Control-N (CTRL+N). More shortcut key information is available in the NetBeans help topic under "IDE Basics>Keyboard Shortcuts".

Menu items may be grayed out for various reasons. See **Section 13.4 "Grayed out or Missing Items and Buttons"**.

Additional context menus are available by right clicking in a window. For more on these menus, see **Section 12.11 "Projects Window"**.

Available menus are listed below.

> **Note:** See the **Start Page**, *My MPLAB IDE>Extend MPLAB>Selecting Simple or Full-Featured Menus* to see all available menus and menu items. Not all of these items will be applicable to embedded development.

- File Menu
- Edit Menu
- View Menu
- Navigate Menu
- Source Menu
- Refactor Menu
- Run Menu
- Debug Menu
- Team Menu
- Tools Menu
- Window Menu
- Help Menu

11.2.1 File Menu

Below are the menu items in the File menu.

For keyboard shortcuts of some of these menu items, see the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 11-1: FILE MENU OPTIONS

Command	Action
New Project	Creates a new project with the New Project wizard
New File	Creates a new file with the New File wizard
Open Project	Opens an existing project
Open Recent Project	Displays a list of all recently-opened projects for selection.
Import	To import one of the following: Hex/ELF... (Prebuilt) File – built with another tool MPLAB IDE v8 Project – launch Import Legacy Project wizard
Open Team Project	Opens a team project For more information on team server projects, see NetBeans help, "IDE Basics>Collaborative Development".
Close Project	Closes the current project
Close All Projects	Closes all open projects

TABLE 11-1: FILE MENU OPTIONS (CONTINUED)

Command	Action
Open File	Opens an existing file
Open Recent File	Displays a list of all recently-opened files for selection
Project Group	Associates the current project with a group
Project Properties	Opens the Project Properties dialog
Save	Saves the current file
Save As	Saves the current file under a new path and/or name
Save All	Saves all open files If the “Compile on Save” feature is selected, this will also compile/build project files.
Page Setup	Sets up the page for printing
Print	Shows print preview of current file and allows printing
Print to HTML	Prints the current file to a new file in HTML format
Exit	Quits MPLAB® IDE

11.2.2 Edit Menu

Below are the menu items in the Edit menu.

For keyboard shortcuts of some of these menu items, see the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 11-2: EDIT MENU OPTIONS

Command	Action
Undo	Reverses (one at a time) a series of editor actions, except Save
Redo	Reverses (one at a time) a series of Undo commands
Cut	Deletes the current selection and places it on the clipboard
Copy	Copies the current selection to the clipboard
Paste	Pastes the contents of the clipboard into the insertion point
Paste Formatted	Pastes the formatted contents of the clipboard into the insertion point
Delete	Deletes the current selection
Select All	Selects everything in the current document or window
Select Identifier	Selects the word nearest the cursor
Find Selection	Finds instances of the current selection
Find Next	Finds next instance of found text
Find Previous	Finds previous instance of found text
Find	Finds a text string
Replace	Finds a string of text and replaces it with the string specified
Find Usages	Finds usages and subtypes of selected code
Find in Projects	Finds specified text, object names, object types within projects
Replace in Projects	Replaces text, object names, object types within projects
Start Macro Recording	Start recording keystrokes
Stop Macro Recording	Stop recording keystrokes To manage macros, select *Tools>Options*, **Editor** button, **Macros** tab.

11.2.3 View Menu

Below are the menu items in the View menu.

For keyboard shortcuts of some of these menu items, see the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 11-3: VIEW MENU OPTIONS

Command	Action
Editors	Select an available editor to view your files.
Code Folds>Collapse Fold	If the insertion point is in a foldable section of text, collapses those lines into one line
Code Folds>Expand Fold	If the currently selected line in the Source Editor represents several folded lines, expands the fold to show all of the lines
Code Folds>Collapse All	Collapses all foldable sections of text in the Source Editor
Code Folds>Expand All	Expands all foldable sections of text in the Source Editor
Web Browser	Opens the default web browser to the NetBeans home page
IDE Log	Opens the MPLAB IDE log file in a tab of the Output window
Toolbars>File, etc.	Shows the associated toolbar when selected (checked)
Toolbars>Small Toolbar Icons	Uses small icons on the toolbars when selected (checked)
Toolbars>Reset Toolbars	Resets to the default toolbar setup
Toolbars>Customize	Customizes the items on an existing toolbar and allows creation of a new one
Show Editor Toolbar	Shows the editor toolbar on the **File** tab when selected (checked)
Show Line Numbers	Shows line numbers when selected (checked)
Show Diff Sidebar	Shows the DIFF sidebar when selected (checked)
Show Versioning Labels	Shows versioning labels when selected (checked)
Synchronize Editor with Views	Synchronizes the editor with open views when selected (checked)
Show Profiler Metrics	Shows the profiler metrics when selected (checked)
Full Screen	Expand window to full length and breadth of screen

11.2.4 Navigate Menu

Below are the menu items in the Navigate menu.

For keyboard shortcuts of some of these menu items, see the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 11-4: NAVIGATE MENU OPTIONS

Command	Action
Go to File	Finds and opens a specific file
Go to Type	Finds and opens a specific class or interface
Go to Symbol	Finds the symbol name as specified
Go to Previous Document	Opens the document last opened before the current one
Go to Source	Displays the source file containing the definition of the selected class
Go to Declaration	Jumps to the declaration of the item under the cursor
Go to Super Implementation	Jumps to the super implementation of the item under the cursor
Last Edit Location	Scrolls the editor to the last place where editing occurred
Back	Navigates back
Forward	Navigates forward

TABLE 11-4: NAVIGATE MENU OPTIONS (CONTINUED)

Command	Action
Go to Line	Jumps to the specified line
Toggle Bookmark	Sets a bookmark on a line of code
Next Bookmark	Cycles forward through the bookmarks
Previous Bookmark	Cycles backward through the bookmarks
Next Error	Scrolls the Source Editor to the line that contains the next build error
Previous Error	Scrolls the Source Editor to the line that contains the previous build error
Select in Projects	Opens Projects window and selects current document within it
Select in Files	Opens Files window and selects current document within it
Select in Classes	Opens Classes window and selects current document within it
Select in Favorites	Opens Favorites window and selects current document within it

11.2.5 Source Menu

Below are the menu items in the Source menu.

For keyboard shortcuts of some of these menu items, see the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 11-5: SOURCE MENU OPTIONS

Command	Action
Format	Formats the selected code or the entire file if nothing is selected
Remove Trailing Spaces	Removed spaces at the end of the line
Shift Left	Moves the selected line or lines one tab to the left
Shift Right	Moves the selected line or lines one tab to the right
Move Up	Moves the selected line or lines one line up
Move Down	Moves the selected line or lines one line down
Duplicate Up	Copies the selected line or lines one line up
Duplicate Down	Copies the selected line or lines one line down
Toggle Comment	Toggles the commenting out of the current line or selected lines
Complete Code	Shows the code completion box
Insert Code	Pops up a context aware menu that you can use to generate common structures such as constructors, getters, and setters
Fix Code	Displays editor hints The IDE informs you when a hint is available when the light bulb is displayed.
Show Method Parameters	Selects the next parameter. You must have a parameter selected (highlighted) for this shortcut to work.
Show Documentation	Shows documentation for item under the cursor
Insert Next Matching Word	Generates the next word used elsewhere in your code as you type its beginning characters
Insert Previous Matching Word	Generates the previous word used elsewhere in your code as you type its beginning characters
Scan for external changes	Scans file for changes made outside of MPLAB IDE

11.2.6 Refactor Menu

Below are the menu items in the Refactor menu. The items you see are dependent on the type of object (variable, function, etc.) you are refactoring. For more information, see **Section 7.5 "C Code Refactoring"**.

For keyboard shortcuts of some of these menu items, see the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 11-6: REFACTOR MENU OPTIONS

Command	Action
Rename	Enables you to change the name of a variable or function to something more meaningful In addition, it updates all source code in your project to reference the element by its new name.
Move	Moves a class to another package or into another class In addition, all source code in your project is updated to reference the class in its new location
Copy	Copies a class to the same or a different package
Safely Delete	Checks for references to a code element and then automatically deletes that element if no other code references it
Change Function Parameter	Changes the amount and name of parameters for the selected function
Undo	Undoes refactoring
Redo	Redoes refactoring

11.2.7 Run Menu

Below are the menu items in the Run menu.

For keyboard shortcuts of some of these menu items, see the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 11-7: RUN MENU OPTIONS

Command	Action
Run Project	Runs the main or selected project
Test Project	Starts JUnit test for project (Java related)
Build Project	Build all the files in a project
Clean and Build Project	Remove (clean) previously generated project files and then rebuild the files in a project
Batch Build Project	Build multiple configurations of a project (only embedded available)
Set Project Configuration	Selects project configuration – should be "default"
Set Main Project	Set the main project by selecting from a list of open projects
Run File	Runs the currently selected file
Test File	Starts JUnit test for current file. (Java related)
Check File	Check a file against a standard. (XML related)
Validate File	Validate a file against a standard. (XML related)
Repeat Build/Run	Run again after halt
Stop Build/Run	End run

11.2.8 Debug Menu

Below are the menu items in the Debug menu.

For keyboard shortcuts of some of these menu items, see the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 11-8: DEBUG MENU OPTIONS

Command	Action
Debug Project	Debugs the main or selected project
Debug File	Starts debugging session for currently selected file
Debug Test File	Starts debugging test for file in JUnit (Java related)
Discrete Debugger Operation	Performs debug operations one step at a time (discretely): Build, Program Target, Launch Debugger This is useful for changing the Memory window setting during debug and using starter kits.
Finish Debugger Session	Ends the debugging session
Pause	Pauses debugging – use “Continue” to resume.
Continue	Resumes debugging after “Pause” until the next breakpoint or the end of the program is reached
Step Over	Executes one source line of a program If the line is a function call, executes the entire function and then stops.
Step Into	Executes one source line of a program If the line is a function call, it executes the program up to the function’s first statement and then stops.
Step Instruction	Executes one machine instruction If the instruction is a function call, it executes the function and returns control to the caller.
Run to Cursor	Runs the current project to the cursor’s location in the file and stop program execution
Reset	Resets the device
Set PC at cursor	Sets the program counter (PC) value to the line address of the cursor
Focus Cursor at PC	Moves the cursor to the current PC address and centers this address in the window
Stack>Make Callee Current	Makes the method being called the current call Only available when a call is selected in the Call Stack window.
Stack>Make Caller Current	Makes the calling method the current call Only available when a call is selected in the Call Stack window.
Stack>Pop Topmost Call	Pops the call on top of the stack
Toggle Line Breakpoint	Adds a line breakpoint or removes the breakpoint at the cursor location in the program
New Breakpoint	Sets a new breakpoint at the specified line, exception, or method
New Watch	Adds the specified symbol to watch
New Run Time Watch	Adds the specified symbol to watch that will change value as the program runs/executes
Disconnect from Debug Tool	Disconnects communications between MPLAB® X IDE and the debug tool To reconnect, select Run/Debug Run.
Run Debugger/Programmer Self Test	Performs a debug tool self-test For tools that support a self test, follow the tool documentation to set up the hardware and then run this test to confirm proper operation.

11.2.9 Team Menu

Below are the menu items in the Team menu.

For keyboard shortcuts of some of these menu items, see the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 11-9: TEAM MENU OPTIONS

Command	Action
CVS, Mercurial, Subversion	displays submenus that are specific to each version management system Please see the product documentation for more on the submenu options.
Local History	show local history of a file or revert file to history version
Find Issues	find an issue in a version control system
Report an Issue	report an issue to a version control system
Create Build Job	create a build using a version control system

11.2.10 Tools Menu

Below are the menu items in the Tools menu.

For keyboard shortcuts of some of these menu items, see the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 11-10: TOOLS MENU OPTIONS

Command	Action
Embedded	visible if a plug-in has been added – select the plug-in from the submenu.
Licenses	• allows roaming in and out of compiler licenses • allows activation of a license For more information, see XCLM documentation under your compiler install path, in the `docs` directory.
Apply Diff Patch	select the Diff patch file and apply to your code
Diff	compares two files selected in the IDE
Add to Favorites	adds selected file to Favorites window
Templates	opens the Template Manager
DTDs and XML Schemas	opens the DTDs and XML Schemas Manager
Plugins	opens the Plugins Manager For details, see NetBeans help, "IDE Basics>Plugins>About Managing Plugins".
Options	opens the Options dialog For Mac OS X: Use *mplab_ide>Preferences*.

11.2.11 Window Menu

Below are the menu items in the Window menu.

For keyboard shortcuts of some of these menu items, see the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 11-11: WINDOW MENU OPTIONS

Command	Action
Projects	Opens the Projects window
Files	Opens the Files window
Classes	Opens the Classes window
Favorites	Opens the Favorites window
Services	Opens the Services window
Tasks	Opens the Task List window
Dashboard	Opens the Dashboard window See **Section 5.17 "View the Dashboard Display"**.
Output>Output	Opens or moves to front the Output window
Output>Search Results	Opens or moves to front the Search window
Output>Find Usages Results	Shows results of "Find Usages" in window
Output>Versioning Output	Shows results of version control action in window
Output>Refactoring Preview	Opens a Preview window of refactoring results
Output>Disassembly Listing File (Project *Project*)	Opens the disassembly listing for the project *Project*
Output>Call Graph	Opens the Call Graph window
Navigating>Navigator	Opens the Navigator window
Navigating>Hierarchy	Opens a Hierarchy window
Debugging>Variables	Opens the Local Variables debugger window
Debugging>Watches	Opens the Watches debugger window
Debugging>Call Stack	Opens the Call Stack debugger window
Debugging>Breakpoints	Opens the Breakpoints window
Debugging>Sessions	Opens the Sessions window
Debugging>Sources	Opens the Sources window
Debugging>Disassembly	Opens the Disassembly window
Debugging>PIC AppIO	Opens the Application In/Out window See MPLAB® REAL ICE™ in-circuit emulator documentation for more on Application I/O.
Debugging>Trace	opens the Trace window See MPLAB REAL ICE in-circuit emulator documentation for more on trace.
Debugging>Stopwatch	opens the Stopwatch window
Debugging>PC Sampling	opens the PC Sampling window See MPLAB REAL ICE in-circuit emulator documentation for more on PC sampling and profiling.
Versioning>CVS	select CVS version control items For more on CVS, see http://www.nongnu.org/cvs
Versioning>Subversion	select Subversion version control items For more on Subversion, see http://subversion.tigris.org
Versioning>Mercurial	select Mercurial version control items For more on Mercurial, see http://mercurial.selenic.com
PIC Memory Views>*Memory*	opens specified Memory window Memories shown depend on the project device.

TABLE 11-11: WINDOW MENU OPTIONS (CONTINUED)

Command	Action
Simulator>Stimulus	opens the Simulator Stimulus window
Simulator>Analyzer	opens the Simulator Analyzer window
Simulator>IOPin	opens the Simulator IO Pin window
Other>Exception Reporter	opens the Exception Reporter window for exception breakpoints
Other>CSS Preview	opens the Preview window for a cascading style sheet
Other>CSS Style Builder	opens the Style Builder window for cascading style sheet rules
Other>Macro Expansion	opens the Macro Expansion window to see macro structure
Editor	opens an empty Editor window
Close Window	closes the current tab in the current window If the window has no tabs, the whole window is closed.
Maximize Window	maximizes the Source Editor or the present window
Undock Window	detaches the window from the IDE
Clone Document	clones the active document
Close All Documents	closes all open documents in the Source Editor
Close Other Documents	closes all open documents except the active one
Documents	opens the Documents dialog box, in which you can save and close groups of open documents
Reset Windows	resets windows to their default settings

11.2.12 Help Menu

Below are the menu items in the Help menu.

For keyboard shortcuts of some of these menu items, see the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 11-12: HELP MENU OPTIONS

Command	Action
Help Contents	displays the JavaHelp viewer with all installed help sets
Online Docs and Support	opens the NetBeans support web page
Keyboard Shortcuts Card	displays the keyboard shortcuts document
Start Page	opens or moves the **Start Page** tab in front of any other open tabs
About	displays a window about MPLAB® IDE

11.3 TOOLBARS

MPLAB IDE displays different toolbars depending on which features or tools you are using. The icons in these toolbars provide shortcuts to routine tasks. To add or remove icons from a toolbar, or create a new toolbar, see **Section 6.7 "Customize Toolbars"**.

Toolbar buttons may be grayed out for various reasons. See **Section 13.4 "Grayed out or Missing Items and Buttons"**.

Toolbars Available

The following basic toolbars are available.

- File Toolbar
- Clipboard Toolbar
- Status Flags Toolbar
- Undo/Redo Toolbar
- Run Toolbar
- Debug Toolbar
- Memory Toolbar
- Quick Search Toolbar
- Editor Toolbar

Toolbar Features

Toolbars have the following features:

- Hover the mouse pointer over an icon to pop up the icon function.
- Click and drag the toolbar to another location in the toolbar area.
- Right click in the toolbar area to show/hide a toolbar or change the contents of some toolbars.
- Select *View>Toolbars* to show/hide a toolbar, change the contents of some toolbars or create a custom toolbar.

11.3.1 File Toolbar

The File Toolbar currently contains button icons for the following functions. These functions are also on the File menu.

- New File – Creates a new file with the New File wizard.
- New Project – Creates a new project with the New Project wizard.
- Open Project – Opens an existing project.
- Save All Files – Saves all open files.

11.3.2 Clipboard Toolbar

The Clipboard Toolbar currently contains button icons for the following functions. These functions are also on the Edit menu.

- Cut – Deletes the current selection and places it on the clipboard.
- Copy – Copies the current selection to the clipboard.
- Paste – Pastes the contents of the clipboard into the insertion point.

11.3.3 Status Flags Toolbar

The Status Flags Toolbar contains:

- PC – Program Counter (PC) current value.

11.3.4 Undo/Redo Toolbar

The Undo/Redo Toolbar currently contains button icons for the following functions. These functions are also on the Edit menu.

- Undo – Reverses (one at a time) a series of editor actions, except Save.
- Redo – Reverses (one at a time) a series of Undo commands.

11.3.5 Run Toolbar

The Run Toolbar currently contains button icons for the following functions. These functions are also on the Run, Debug and project context menus.

- Set Project Configuration – Selects project configuration. Choose "default" or "Customize".
- Build Project – Builds all the project files. Click on down arrow for other related options.
- Clean and Build Project – Deletes files from previous builds and then builds the all project files. Click on down arrow for other related options.
- Make and Program Device Project – Builds, programs the target and Runs the selected project. Click on down arrow for other related options.
- Hold in Reset – Builds, programs the target and holds in Reset the selected project.
- Read Device Memory – Reads target device memory and loads into MPLAB X IDE. Click on down arrow for other related options.
- Debug Project – Builds, programs the target and Debug Runs the selected project.

11.3.6 Debug Toolbar

The Debug Toolbar currently contains button icons for the following functions. These functions are also on the Debug menu.

- Finish Debugger Session – Ends the debugging session.
- Pause – Pauses debugging. Use "Continue" to resume.
- Reset – Runs the current project to the cursor's location in the file and stop program execution.
- Continue – Resumes debugging until the next breakpoint or the end of the program is reached.
- Step Over – Executes one source line of a program. If the line is a function call, executes the entire function then stops.
- Step Over Expression – Steps over the expression and then stops the debugging.
- Step Into – Executes one source line of a program. If the line is a function call, executes the program up to the function's first statement and stops.
- Step Out – Executes one source line of a program. If the line is a function call, executes the functions and returns control to the caller.
- Run to Cursor – Runs the current project to the cursor's location in the file and stop program execution.
- Apply Code Changes – Apply any changes in the code to the executing program.
- Set PC at cursor – Sets the program counter (PC) value to the line address of the cursor.
- Focus Cursor at PC – Moves the cursor to the current PC address and centers this address in the window.

11.3.7 Memory Toolbar

The Memory Toolbar displays the current PC memory usage for MPLAB IDE. Click on the display to force garbage collection.

11.3.8 Quick Search Toolbar

The Quick Search Toolbar displays a search text box. Click the down arrow next to the magnifying glass to select the type of search.

11.3.9 Editor Toolbar

The Editor Toolbar currently contains button icons for the following functions. These functions are also on the Edit and Source menus. This toolbar appears at the top of the tab containing the current file source code.

- Last Edited – Moves to the line that contains the last edit made.
- Back – Navigates back.
- Forward – Navigates forward.
- Find Selection – Finds the first occurrence of the selected text.
- Find Previous Occurrence – Finds the previous occurrence of the selected text.
- Find Next Occurrence – Finds the next occurrence of the selected text.
- Toggle Highlight Search – Turns on/off text selected for search.
- Previous Bookmark – Cycles backwards through the bookmarks.
- Next Bookmark – Cycles forward through the bookmarks.
- Toggle Bookmark – Sets a bookmark on a line of code.
- Shift Left – Moves the selected line or lines one tab to the left.
- Shift Right – Moves the selected line or lines one tab to the right.
- Start Macro Recording – Start recording keystrokes.
- Stop Macro Recording – Stop recording keystrokes.
- Comment – Makes selected line into a comment by adding “//”.
- Uncomment – Makes selected comment into a line by removing “//”.
- Go to Header/Source – Moves between the header and related source code.

11.4 STATUS BAR

The status bar will provide up-to-date information on the status of your MPLAB IDE session. Currently only editor information is provided.

11.5 GRAYED OUT OR MISSING ITEMS AND BUTTONS

There are several reasons why a menu item, toolbar button or status bar item may be grayed out (unavailable) or missing:

- The item/button is related to a device feature that the selected device does not have, e.g., the PIC16F877A does not support external memory.
- The item/button is related to a tool feature that the selected tool does not have, e.g., "Step Out" is not available on MPLAB ICD 3.
- The item/button is project-related and no project has been selected, e.g., project build will not be available (No Active Project).
- The item/button is not supported for the selected device or tool.
- The item/button is performing its function and so cannot be selected again, e.g., "Run Project" is grayed out when the program is running.
- The item/button is mutually exclusive to another item, e.g., "Pause" is available when the program is running while "Continue" is grayed out, and "Continue" is available when the program is halted while "Pause" is grayed out.

Chapter 12. MPLAB X IDE Windows and Dialogs

12.1 INTRODUCTION

The MPLAB X IDE desktop is divided into panes containing tabbed windows. Not all of these are visible until a feature has been selected.

As an example, when you first open MPLAB X IDE, only the **Start Page** is open. After you open a project, the basic windows open – namely Project and Files in the top left pane, Navigator in the bottom left pane, and Output in the bottom right pane. The **Start Page** window moves into the top right pane.

MPLAB X IDE windows are a combination of basic NetBeans windows and MPLAB X IDE specific windows. Dialogs open when selected from menu items. As with windows, MPLAB X IDE dialogs are a combination of basic NetBeans dialogs and MPLAB X IDE specific dialogs. For information on NetBeans windows and dialogs, see **Chapter 13. "NetBeans Windows and Dialogs"**.

12.2 MPLAB X IDE WINDOWS MANAGEMENT

Information on managing IDE windows may be found in the NetBeans help topic *IDE Basics>Configuring the IDE>Configuring the IDE's Workspace>Managing IDE Windows*.

Additional window information is provided below.

12.2.1 Window Data Updates

Open windows are updated on a program halt (except for Flash memory windows, which must be read.) A program halt includes a halt after a run and stepping. Halt updates can have the following effects:

- Speed – Updating takes time. To decrease update time, close any unused windows.
- Data overwrites – The value of a file register displayed in an open window is read on halt. See your device data sheet for the register operation on read.

12.2.2 Window Data Changes

MPLAB X IDE window data may be edited as described below. If you cannot edit the data, then this information is not available for you to change.

- Data may be edited "in place". Either double click to select an item and then type in a new value, or click on the ellipsis (...) next to an item and type the new value in the window that pops up.
- Data may be chosen from a drop-down list when only certain choices are possible.

12.2.3 Window Focus

To ensure that you have a window in focus, click not only on the window frame, but also on a button, a table cell, or a drop-down combo box.

12.3 MPLAB X IDE WINDOWS WITH RELATED MENUS AND DIALOGS

MPLAB X IDE uses some NetBeans windows as-is. However, other windows and their related menus are modified or created specifically for embedded use.

Specific Windows*	Related Menu
Breakpoints Window	Window>Debugging>Breakpoints
Customize Toolbars Window	View>Toolbars
Licenses Windows	Tools>Licenses
Dashboard Window	Window
Memory Windows	Window>PIC Memory Views
Output Window	Window>Output>Output
Project Properties Window	File
Projects Window	Window
Tools Options Embedded Window	Tools>Options (Windows, Linux OS) mplab_ide>Preferences (Mac OS X)
Trace Window	Window>Debugging
Watches Window	Window>Debugging
Wizard Windows	File>New Project File>New File

* Depending on the tools you have installed, tool-specific windows may be available for viewing. See documentation about your tool for information regarding a specific window.

12.4 BREAKPOINTS WINDOW

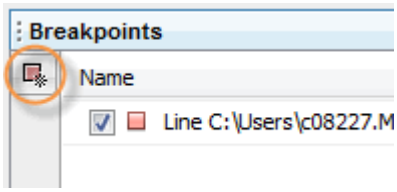
The Breakpoints window is used for setting and viewing breakpoints in code. Open the window by selecting *Window>Debugging>Breakpoints*.

For more on using the Breakpoints window, see **Section 4.17 "Control Program Execution with Breakpoints"**.

12.4.1 New Breakpoint Dialog

Click the **Create New Breakpoint** button on the Breakpoint window to open the New Breakpoint dialog.

FIGURE 12-1: NEW BREAKPOINT BUTTON



The options available for the breakpoint are determined by the type of breakpoint selected, and the selected device (not all options are available for all devices).

12.4.2 Line Breakpoints

The following options are available for breakpoints specified on a line of code.

TABLE 12-1: BREAKPOINT TYPE: LINE – SETTINGS

Item	Description
Settings	Create a new Line break point with a left mouse click on the Editor gutter next to the file line. Or, select “Toggle Line Breakpoint” from the Editor context menu.

12.4.3 Data Breakpoints

The following options are available for data memory breakpoints.

TABLE 12-2: BREAKPOINT TYPE: DATA – SETTINGS

Item	Description
Project	Select an open project from the drop-down list This is the project in which code will contain the breakpoint.
Symbols	Enter the name of a global symbol, or SFR, or browse to one by clicking **Symbols**. Accessing this symbol according to “Breaks on” triggers a pause in code execution.
Enable Range Address	Check to set a range breakpoint. Uncheck to set a single breakpoint.
Address Address (Start)	Depending on the selection of “Enable Range Address”, this item can be “Address” or “Address (Start)”. Enter a hexadecimal address in data memory Accessing this address according to “Breaks on” triggers a pause in code execution.
Address (End)	If “Enable Range Address” is checked, this box is enabled. Enter a hexadecimal address in data memory.
Breaks on	**Read**, **Write**, **Read or Write**: Break code execution when the symbol or address stated above is read, written, or either read or written. **Read Specific Value**, **Write Specific Value**, **Read or Write Specific Value**: Break code execution when the symbol or address stated above is read and has the value specified below, written with the value specified below, or either read or written according to the value specified below. **Note:** For dsPIC DSCs, there are read and write options for the X and Y buses.
Value	For the “Breaks on” selection of Read Specific Value, Write Specific Value, or Read or Write Specific Value, enter a hexadecimal value here.
Value Comparison	*For PIC16F1xxx devices only-* Compare to “Value” as specified: **= Value**: Equal to value **!= Value**: Not equal to value **> Value**: Greater than value **< Value**: Less than value
Data Value Mask	*For PIC16F1xxx devices only-* Use mask when comparing to “Value” Enter a value in the range 0x00 to 0xhh, where: 0x00: No bits compared 0xhh: All bits compared

TABLE 12-3: BREAKPOINT TYPE: DATA - PASS COUNT

Item	Description
Condition	Determine when the break specified under "Breaks on" occurs. **Always Break**: Always break when the "Breaks on" condition is met. **Break occurs Count Instructions after Event**: After an event ("Breaks on" condition) occurs, execute Count Instructions before actually breaking. **Event must occur Count times**: An event ("Breaks on" condition) must occur Count times before actually breaking.
Count*	According to the Condition specified, enter either a count for the number of instructions after an event or the number of events.

* See also **Section "For some devices (PIC16F1xxx MCUs), enhanced event breakpoints actions are available."**.

TABLE 12-4: BREAKPOINT TYPE: DATA

Item	Description
Trigger Options	*For PIC16F1xxx devices only-* Select when to trigger, either: • Do not trigger out when breakpoint is reached • trigger out when breakpoint is reached
Interrupt Context	*For PIC16F1xxx devices only-* Interrupt Context qualifier for address/data breakpoints. Select from: • Always break (break in both ISR and main code) • Break in main line (non-interrupt) context only – break in main code only • Break in interrupt context only – break in ISR code only

12.4.4 Address Breakpoints

The following options are available for program memory breakpoints.

TABLE 12-5: BREAKPOINT TYPE: ADDRESS – SETTINGS

Item	Description
Project	Select an open project from the drop-down list This is the project in which code will contain the breakpoint.
Enable Range Address	Check to set a range breakpoint. Uncheck to set a single breakpoint.
Address Address (Start)	Depending on the selection of "Enable Range Address", this item may be "Address" or "Address (Start)". Enter a hexadecimal address in program memory. Accessing this address according to "Breaks on" triggers a pause in code execution.
Address (End)	If "Enable Range Address" is checked, this box is enabled. Enter a hexadecimal address in data memory.
Breaks on	**Program Memory Execution**: Break code execution when the address specified above is reached. **TBLRD Program Memory**: Break code execution when a table read to the address specified above occurs. **TBLWT Program Memory**: Break code execution when a table write to the address specified above occurs.

TABLE 12-6: BREAKPOINT TYPE: ADDRESS - PASS COUNT

Item	Description
Condition	Determine when the break specified under “Breaks on” occurs **Always Break**: Always break when the “Breaks on” condition is met. **Break occurs Count Instructions after Event**: After an event (“Breaks on” condition) occurs, execute Count Instructions before actually breaking. **Event must occur Count times**: An event (“Breaks on” condition) must occur Count times before actually breaking.
Count*	According to the Condition specified, enter either a count for the number of instructions after an event or the number of events.

* See also **Section “For some devices (PIC16F1xxx MCUs), enhanced event breakpoints actions are available.”**.

TABLE 12-7: BREAKPOINT TYPE: ADDRESS

Item	Description
Trigger Options	*For PIC16F1xxx devices only.* Select when to trigger, either: • Do not trigger out when breakpoint is reached • trigger out when breakpoint is reached
Interrupt Context	*For PIC16F1xxx devices only.* Interrupt Context qualifier for address/data breakpoints. Select from: • Always break (break in both ISR and main code) • Break in main line (non-interrupt) context only - break in main code only • Break in interrupt context only - break in ISR code only

12.4.5 Event Breakpoints

The following options are available for event breakpoints.

TABLE 12-8: BREAKPOINT TYPE: EVENT

Item	Description
Project	Select an open project from the drop down list This is the project whose code will contain the breakpoint.
Break on clock mode switch	Break when the clock mode switches
Break on Reset instruction	Break when a device Reset instruction occurs
Break on Sleep	Break when Sleep is entered
Break on stack over/underflow	Break when the stack either overflows or underflows
Break on wake up	Break when the device wakes up from sleep
Break when watchdog timer has expired	Break when the watchdog timer period has ended
Break on execution out of bounds	Break when the program attempts to move out of normal program memory/space
Break on MCLR Reset	Break on a Master Clear (MCLR) Reset
Break on trigger in signal	Break when a Trigger In pulse is detected

* Select event(s) from the list that will cause executing code to pause (break). Some events may not be available for your device.

For some devices (PIC16F1xxx MCUs), enhanced event breakpoints actions are available.

Action	Description
Break	Break (halt) execution per option specified
Trigger out	Emit a trigger out pulse per option specified
Break and trigger out	Break (halt) execution AND Emit a trigger out pulse per option specified

12.4.6 Pass Count Operation

Using a pass count allows you to delay breaking until after a specified count.

Break occurs Count Instructions after Event

Count is the number of instructions that execution passes after the breakpoint but before stopping.

For example,

- 0 specifies that execution stops immediately
- 1 specifies that execution stops after one additional instruction
- 10 specifies that execution stops after ten additional instructions

Event must occur Count times

Count is the number of times that execution passes the event until stopping.

For example,

- 0 specifies that execution stops immediately
- 1 specifies that execution passes the event one time, then stops the next time (the second time the event occurs)
- 10 specifies that execution passes the event ten times, then stops the next time (the eleventh time the event occurs)

12.5 CUSTOMIZE TOOLBARS WINDOW

You can customize MPLAB X IDE toolbars using the Customize Toolbars window. Select *View>Toolbars>Customize* to open the window.

For customization instructions, see **Section 6.7 "Customize Toolbars"**.

12.6 LICENSES WINDOWS

MPLAB X IDE provides two pop-up windows for managing network compiler licenses. Select *Tools>Licenses* to open these windows.

> **Note:** The license manager cannot detect whether or not a network server connection exists before it tries to connect. If you are using Windows OS and attempt to connect to a network when no network connection is available, you will experience delays.

TABLE 12-9: NETWORK COMPILER LICENSE WINDOWS

Tools>License Menu Item	Window	Description
Roam Network License	Network License Management	Roam a network license in or out. **Note:** For roam out, there is a 60-minute delay before the server license can be used by other clients.
Activate Network License	Activate Network License	Enter an activation key to active a network compiler license.

For details on network licenses, see "*Installing and Licensing MPLAB® XC C Compilers*" (DS50002059).

12.7 DASHBOARD WINDOW

The Dashboard window contains general project information, such as checksums, memory usage, and breakpoint resources. See **Section 5.17 "View the Dashboard Display"** for details.

12.8 MEMORY WINDOWS

Memory windows (*Window>PIC Memory Views*) display the many types of device memory, such as SFRs and Configuration bits. Use the "Memory" and "Format" drop-down boxes to customize your window.

For more on these controls, see **Section 4.21 "View/Change Device Memory (including Configuration Bits)"**.

Available PIC Memory Views: 8- and 16-Bit Devices

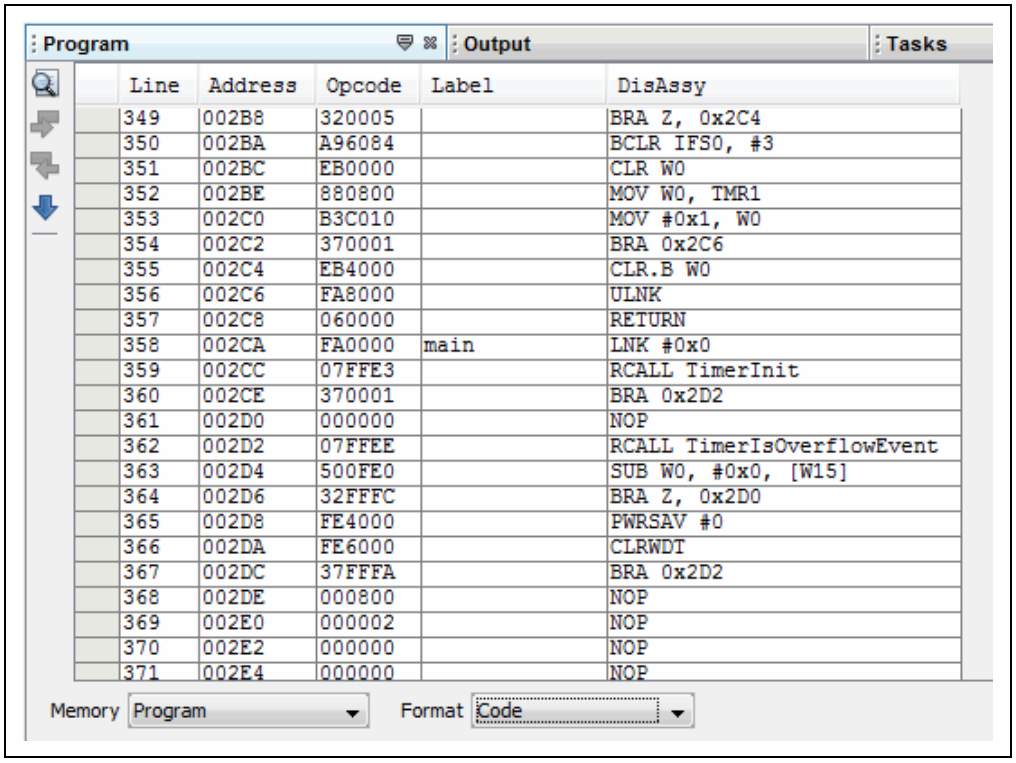
- Program Memory Window
- File Registers Window
- SFRs Window
- Configuration Bits Window
- EE Data Memory Window
- Other Memory Window

Available PIC Memory Views: 32-Bit Devices

- Execution Memory Window
- Data Memory Window
- Peripherals Window
- Configuration Bits Window
- CPU Registers Window
- User ID Memory Window

Memory Window Context (Right Click) Menu: the following section provides information about the context menu, **Section 12.8.13 "Memory Window Menu"**.

FIGURE 12-2: MEMORY WINDOW WITH CONTENT



12.8.1 Program Memory Window

The Program Memory window displays locations in the range of program memory for the currently selected processor. If external program memory is supported by the selected device and enabled, it will also appear in the Program Memory window.

For the MPLAB X Simulator, when a program memory value changes or the processor is halted, the data in the Program Memory window is updated.

For any Microchip hardware debug tool (e.g., MPLAB REAL ICE in-circuit emulator), when a program memory value changes or the processor is halted, the data in the Program Memory window is **not** updated; you must do a Read of device memory.

You may change the way opcodes are displayed in the program memory window by clicking one of the drop-down boxes on the bottom of the window:

- Hex
- Code
- PSV Mixed (dsPIC DSC/PIC24 devices only)
- PSV Data (dsPIC DSC/PIC24 devices only)

12.8.1.1 CODE

Code format displays disassembled hex code with symbols. The window will have the following columns:

- Debug Info – Information useful for debugging
  A pointer shows the current location of the program counter.
- Line – Reference line number
- Address – Opcode hexadecimal address
- Opcode – Hexadecimal opcode, shown in 2- or 3-byte blocks
  For most PIC MCUs, these blocks represent words.
  For PIC18CXXX devices, the blocks represent 2 bytes.
  For dsPIC DSC devices, the blocks represent 3 bytes.
- Label (Symbolic Only) – Opcode label in symbolic format.
- Disassembly – A disassembled version of the opcode mnemonic.

12.8.1.2 HEX

This format displays program memory information as hex code. The window will have the following columns:

- Address – Hexadecimal address of the opcode in the next column
- *Opcode Blocks* – Hexadecimal opcode, shown in 2- or 3-byte blocks
  For most PIC MCUs these blocks represent words. For PIC18CXXX devices, the blocks represent 2 bytes. For dsPIC DSC devices, the blocks represent 3 bytes. The opcode block that is highlighted represents the current location of the program counter.
- ASCII – ASCII representation of the corresponding line of opcode

12.8.1.3 PSV MIXED (dsPIC DSC/PIC24 DEVICES ONLY)

This format displays program memory as opcode and the PSV area (CORCON register, PSV bit set). The window will have the following columns:

- *Debug Info* – Information useful for debugging. A pointer shows the current location of the program counter.
- Line – Reference line number.
- Address – Opcode hexadecimal address.
- PSV Address – Data space hexadecimal address of the opcode.
- Data – Opcode formatted as data.
- Opcode – Hexadecimal opcode, shown in 3-byte blocks.
- Label – Opcode label in symbolic format.
- Disassembly – A disassembled version of the opcode mnemonic.

For more information, see the "*dsPIC30F Family Reference Manual"* (DS70046).

12.8.1.4 PSV DATA (dsPIC DSC/PIC24 DEVICES ONLY)

This format displays program memory as file registers, for when program space is visible in data space (CORCON register, PSV bit set). The window will have the following columns:

- Address – Program space hexadecimal address of the data
- PSV Address – Data space hexadecimal address of the data
- *Data Blocks* – Hexadecimal data, shown in 3-byte blocks
  The highlighted data block represents the current location of the program counter.
- ASCII – ASCII representation of the corresponding line of data

For more information, see the *dsPIC30F Family Reference Manual* (DS70046).

12.8.2 File Registers Window

The File Register window displays all the file registers of the selected device. When a file register value changes, or the processor is interrogated, the data in the File Register window is updated.

Note: To speed up debugging with certain hardware tools, close this window. Use the SFR or Watch window instead.

You may change the way data is displayed in the file register window by clicking one of the buttons on the bottom of the window.

- **Hex**
- **Symbol**
- **Dual Port (dsPIC33FJ DSC/PIC24HJ MCU devices only)**
- **XY Data (dsPIC DSC devices only)**

12.8.2.1 HEX

This format displays file register information as hex data. The window has the following columns:

- Address – Hexadecimal address of the data in the next column
- *Data Blocks* – Hexadecimal data, shown in 1- or 2-byte blocks
- ASCII – ASCII representation of the corresponding line of data

12.8.2.2 SYMBOL

This format displays each file register symbolically with corresponding data in hex, decimal, binary and character formats. The window has the following columns:

- Address – Data hexadecimal address
- Symbol Name – Symbolic name for the data
- Radix Information – Hex, Decimal, Binary, Char
  Radix information is displayed in these four columns. Hex is shown in 1- or 2-byte blocks.

12.8.2.3 DUAL PORT (dsPIC33FJ DSC/PIC24HJ MCU DEVICES ONLY)

This format displays file register information as hex data. The window has the following columns:

- Address – Data hexadecimal address
- DMA Address – Offset from the silicon DMA address
- *Data Blocks* – Hexadecimal data, shown in 1- or 2-byte blocks
- ASCII – ASCII representation of the corresponding line of data

For information on dsPIC33F DSC and PIC24H MCU devices, see the Microchip web site for device data sheets and dsPIC33F and PIC24H Reference Manual sections.

12.8.2.4 XY DATA (dsPIC DSC DEVICES ONLY)

This format displays file register information as hex data. The window has the following columns:

- Address – X hexadecimal address of the data
- Y Bus – Y hexadecimal address of data, if supported
- *Data Blocks* – Hexadecimal data, shown in 2-byte blocks
- ASCII – ASCII representation of the corresponding line of data

For more information on dsPIC DSC devices, see "*dsPIC30F Family Reference Manual*" (DS70046).

12.8.3 SFRs Window

The Special Function Registers (SFRs) window displays the contents of the SFRs for the selected processor. The format provided by this window is more useful for viewing the SFRs than the normal file register window, since each SFR name is included and several number formats are presented. To view only a few SFRs, you might want to use a Watch window, which may help with speed issues when using hardware debug tools (i.e., faster window update rate.)

Whenever a break occurs, the contents of the Special Function Registers are updated.

Visible Registers

If a data memory register is not physically implemented on a device, it may not appear in the SFR list. Some tools, such as simulators, may allow you to see registers that do not exist on the actual device, such as prescalers.

Single Stepping

If “Freeze Peripherals On Halt” is selected, the I/O port bits in the SFR or the Watch windows will not update when single stepping. The pin will be modified, but the read request to retrieve the new value is blocked by the freeze and cannot be updated until the next step or run command.

- Individual
- Peripheral

12.8.3.1 INDIVIDUAL

In this display, SFRs are listed by address.

Data is displayed in the following columns.

- Address – SFR hexadecimal address
- Name – Symbolic name for the SFR
- Radix Information – Hex, Decimal, Binary
  Radix information is displayed in these four columns. Hex is shown in 1-byte blocks.

12.8.3.2 PERIPHERAL

In this display, SFRs are grouped according to their related device peripherals.

Data is displayed in the following columns:

- Address – SFR hexadecimal address
- Name – Name of peripheral or symbolic name for the SFR
- Radix Information – Hex, Decimal, Binary, Char
  Radix information is displayed in these four columns. Hex is shown in 1-byte blocks.

Click the **Select Peripheral** button to view the SFRs for only the peripherals selected.

12.8.4 Configuration Bits Window

Details about using the Configuration Bits window is discussed in **Section 4.21.4 "Set Configuration Bits"**.

Data is displayed in the following columns.

TABLE 12-10: CONFIGURATION BITS COLUMNAR DISPLAY

Column Head	Definition
Address	The address of the configuration word/byte
Name	The name of the Configuration Register
Value	The current value of the configuration word/byte
Field*	For configuration bits set in code, the field portion of the macro As an example, `WDTE` is the field portion of the macro `_WDTE_OFF`.
Option*	For configuration bits set in code, the option portion of the macro As an example, `OFF` is the option portion of the macro `_WDTE_OFF`.
Category	The name of the Configuration bit in the corresponding configuration word/byte
Setting	The current setting of the Configuration bit. Use the drop-down list to change the setting The Value of the configuration word/byte will change accordingly.

* Not all devices supported.

12.8.5 EE Data Memory Window

The EEPROM window displays EEPROM data for any microcontroller device that has EEPROM data memory (e.g., PIC16F84A). Data/opcode hex information of the selected device is shown.

For the MPLAB X Simulator, when an EEPROM register value changes or the processor is halted, the data in the EEPROM window is updated.

For any Microchip hardware debug tool (e.g., MPLAB REAL ICE in-circuit emulator), when an EEPROM register value changes or the processor is halted, the data in the EEPROM window is **not** updated; you must do a Read of device memory.

The start of EEPROM data memory needs to be specified for use with programmers. The table below shows some generic values, but please check the programming specification for your selected device to determine the correct address.

TABLE 12-11: PROGRAMMER EEPROM START ADDRESS

Device	Address
Most PIC1X MCUs	0x2100
PIC16F19XX MCUs	0x1E000
PIC18F MCUs	0xF00000
PIC24F MCUs	0x7FFE00

This display format shows data in the following columns:

- Address – Hexadecimal address of the data in the next column
- Data Blocks – Hexadecimal data, shown in 1-, 2- or 4-byte blocks, selectable from the menu
- ASCII – ASCII representation of the corresponding line of data

12.8.6 Other Memory Window

This is a flexible memory window that allows you to select the other windows by using the Memory drop-down list on the bottom of the window.

12.8.7 Execution Memory Window

The Execution Memory window displays locations in the range of program and/or data memory for the currently selected PIC32MX device.

For the MPLAB X Simulator, when an execution memory value changes or the processor is halted, the data in the Execution Memory window is updated.

For any Microchip hardware debug tool (e.g., MPLAB REAL ICE in-circuit emulator), when an execution memory value changes or the processor is halted, the data in the Execution Memory window is **not** updated; you must do a Read of device memory.

You may select the type of memory displayed in the window by clicking on one of the tabs on the bottom of the window:

- Code View - Program Memory
- Data View - Data Memory

12.8.7.1 CODE VIEW - PROGRAM MEMORY

This format displays program memory information as hex code. The window will have the following columns:

- Line – Reference line number corresponding to memory address.
- Address – Physical hexadecimal address of the opcode.
- Opcode – Hexadecimal opcode, shown in 4-byte blocks. The opcode that is highlighted represents the current location of the program counter.
- Label – Opcode label in symbolic format.
- Disassembly – A disassembled version of the opcode mnemonic.

12.8.7.2 DATA VIEW - DATA MEMORY

This format displays data memory information as hex code. The window will have the following columns:

- Address – Hexadecimal address of the data in the next column.
- Data Blocks – Hexadecimal data, shown in 4-byte blocks.
- ASCII – ASCII representation of the corresponding line of data.

12.8.8 Data Memory Window

The Data Memory window displays locations in the range of data and/or program memory for the currently selected PIC32MX device.

You may select the type of memory displayed in the window by clicking on one of the tabs on the bottom of the window:

- Data View - Data Memory
- Code View - Program Memory

12.8.8.1 DATA VIEW - DATA MEMORY

This format displays data memory information as hex code. The window will have the following columns:

- Address – Hexadecimal address of the data in the next column.
- Data Blocks – Hexadecimal data, shown in 4-byte blocks.
- ASCII – ASCII representation of the corresponding line of data.

12.8.8.2 CODE VIEW - PROGRAM MEMORY

This format displays program memory information as hex code. The window will have the following columns:

- Line – Reference line number corresponding to memory address
- Address – Physical hexadecimal address of the opcode
- Opcode – Hexadecimal opcode, shown in 4-byte blocks
  The opcode that is highlighted represents the current location of the program counter.
- Label – Opcode label in symbolic format
- Disassembly – A disassembled version of the opcode mnemonic

12.8.9 Peripherals Window

The Peripherals window displays the contents of the SFRs that relate to the device peripherals. To view only a few SFRs, you may prefer to use a Watch window, which may help with speed issues when using hardware debug tools (i.e., faster window update rate.)

Whenever a break occurs, the contents of the SFRs are updated.

Visible Registers

If a data memory register is not physically implemented on a device, it may not appear in the SFR list. Some tools, such as simulators, may allow you to see registers that do not exist on the actual device, such as prescalers.

Single Stepping

If "Freeze Peripherals On Halt" is selected, the I/O port bits in the SFR or the Watch windows will not update when single stepping. The pin will be modified, but the read request to retrieve the new value is blocked by the freeze and cannot be updated until the next step or run command.

Data is displayed in the following columns.

- Address – SFR physical hexadecimal address.
- Virtual – SFR virtual hexadecimal address as defined by the Bus Matrix.
- Name – Symbolic name for the SFR.
- Radix Information – Hex, Decimal, Binary, Char
  You may add radix information to the display by right clicking on the column header bar. Hex is shown in 4-byte blocks.

12.8.10 Configuration Bits Window

Same as **Section 12.8.4 "Configuration Bits Window"**.

12.8.11 CPU Registers Window

The CPU Registers window displays the contents of the SFRs that relate to the device CPU. To view only a few CPU SFRs, you may prefer to use a Watch window, which may help with speed issues when using hardware debug tools (i.e., faster window update rate.)

Whenever a break occurs, the contents of the CPU Registers are updated.

Visible Registers

If a data memory register is not physically implemented on a device, it may not appear in the SFR list. Some tools, such as simulators, may allow you to see registers that do not exist on the actual device, such as prescalers.

Single Stepping

If “Freeze Peripherals On Halt” is selected, the I/O port bits in the SFR or the Watch windows will not update when single stepping. The pin will be modified, but the read request to retrieve the new value is blocked by the freeze and cannot be updated until the next step or run command.

Data is displayed in the following columns.

- Address – SFR physical hexadecimal address.
- Name – Symbolic name for the SFR.
- Radix Information – Hex, Decimal, Binary, Char

  You may add radix information to the display by right clicking on the column header bar.
- Virtual – SFR virtual hexadecimal address as defined by the Bus Matrix.

12.8.12 User ID Memory Window

Some devices have memory locations where you can store checksum or other code identification (ID) numbers. These locations are readable and writable during program/verify. Depending on the device, they also may be accessible during normal execution through the `TBLRD` and `TBLWT` instructions.

Data is displayed in the following columns.

- Address – User ID hexadecimal address. Right click to view either the Virtual Address or Physical Address.
- User ID – Contents (in hex) of User ID memory.

Consult your device programming specification to determine what values may be entered here. For most devices, this sets the low nibble of the device ID word; the high nibble is set to ‘0’. The high nibble can be only be written to programmatically, such as by using Table Writes.

12.8.13 Memory Window Menu

Right clicking in the Memory window will display various options as shown below. Not all options are available for all windows.

TABLE 12-12: MEMORY WINDOW MENU ITEM

Item	Description
Virtual Address Physical Address	Display the type of address checked under the Address column.
Hex Display Width	Set the hexadecimal display width. Options depend on the device selected. 32-bit example: One byte, e.g., 00 01 02 ... 0E 0F Two bytes, e.g., 00 02 04 ... 0C 0E Four bytes, e.g., 00 04 08 0C
Run to Cursor	Run the program to the current cursor location.
Set PC at Cursor	Set the Program Counter (PC) to the cursor location.
Focus Cursor at PC	Move the cursor to the current PC address and center this address in the window.
Toggle Breakpoint	Toggle (on/off) existing breakpoint.
Symbolic Mode	Display disassembled hex code with symbols.
Verbose Labels	Show internal compiler labels.
Fill Memory	Fill memory from Start Address to End Address with the value in Data. Specify other options in the Fill Memory dialog.
Go To	Go to the address/function specified in the Go To dialog.
Find	Find text specified in the Find dialog.
Enable Multiline Rows	Allow multiple lines in the Configuration Bits window.
Output To File	Write the displayed window contents to a text file. Specify a range of data to output in the Output to File Range dialog.
Import Table	Import tabular data from a file into a Memory window. Specify a range of data to import in the Import Table Range dialog.
Export Table	Export tabular data from a Memory window into a file. Specify a range of data to export in the Export Table Range dialog. Also, specify if the export is to be in a single column.
Print	Print the contents of this window. **NOTE:** If you have a large memory device, the number of pages printed can become very large. A suggestion is to print the window contents to a file (Print dialog, **General** tab, "Print to File" checkbox) and then select which pages from the file you need to print.
Adjust Table Columns	Auto adjust the columns.

Right clicking on the **Window** tab will display other options, such as Close, Maximize/Minimize window, and Dock/Undock window.

12.9 OUTPUT WINDOW

The Task pane contains many windows, some inherited from NetBeans and some that are specific to MPLAB X IDE. The Output window contains the MPLAB X IDE output information. It is shown on tabs within the window.

TABLE 12-13: OUTPUT WINDOW TAB ITEMS

Item	Description
Debugger Console	Shows main debug actions, such as “User program running”.
Tool-specific	Shows tool firmware version, device ID, and action status.
Build, Load	Shows information and status on the build, and program loading.
Clean, Build, Load	Shows information and status on the clean, build, and program loading.
Peripheral Output	Shows technical output from peripherals such as the UART with the Simulator as the debug tool.

Right clicking in the Output window will display various options as shown below.

TABLE 12-14: OUTPUT WINDOW MENU ITEMS

Menu Item	Description
Copy	Copy selected text from the Output window to the clipboard.
Paste	Paste selected text from the clipboard to the Output window.
Find	Find the selected text, or enter other text to find, in the Output window. You may use regular expressions and match case.
Find Next	Find the next occurrence of the Find text.
Find Previous	Find the previous occurrence of the Find text.
Filter	Filter the output by text or regular expression.
Wrap Text	Wrapped the text in the Output window.
Larger Font	Make the font larger.
Smaller Font	Make the font smaller.
Choose Font	Select the font type, style and size.
Save As	Save the selected text to a file.
Clear	Clear all text in the **Output window** tab.
Close	Close the Output window.

12.10 PROJECT PROPERTIES WINDOW

This window is used to view or change the project device, tools and tool settings. For more information, see:

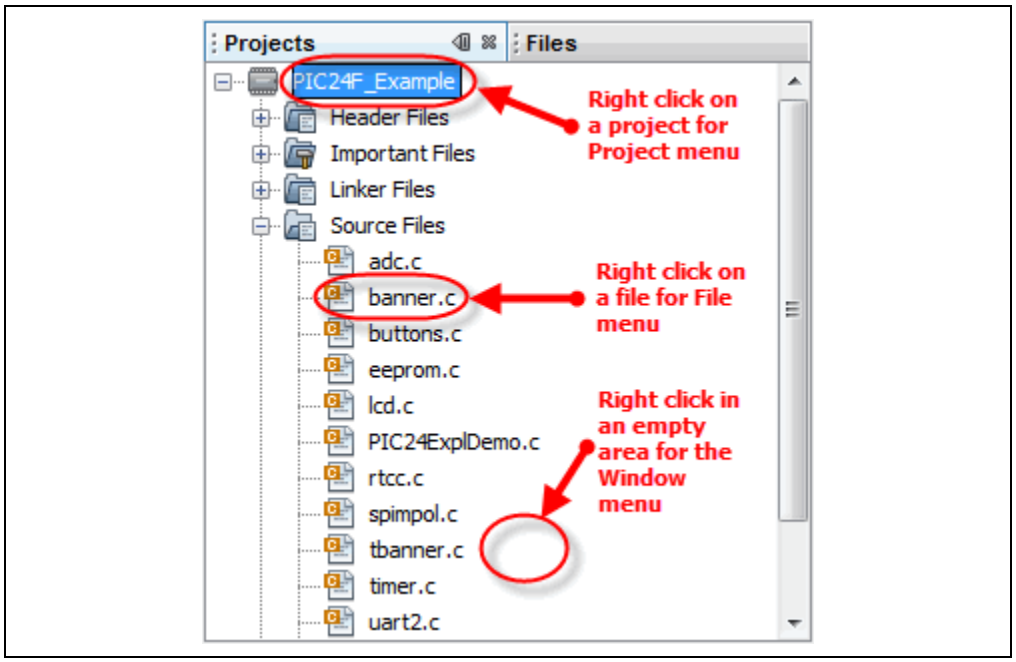
- **Section 4.4 "View or Make Changes to Project Properties"**
- **Section 4.5 "Set Options for Debugger, Programmer or Language Tools"**
- **Section 5.4 "Loadable Projects and Files"**
- **Section 4.11 "Add and Set Up Library and Object Files"**
- **Section 4.13 "Set Build Properties"**

12.11 PROJECTS WINDOW

The Projects window is a NetBeans window. However, it has been customized to show the relevant logical (virtual) folders for an MPLAB X IDE project. Also, the context (right click) menus have been customized with items that are specific to MPLAB X IDE.

- Projects Window – Logical Folders
- Projects Window – Project Menu
- Projects Window – File Menu
- Projects Window – Window Menu

FIGURE 12-3: PROJECT WINDOW CONTEXT MENUS



12.11.1 Projects Window – Logical Folders

Additional logical folders have been added that apply to MPLAB X projects. Therefore, all folders displayed can be categorized as one of the following types of files:

- **Header Files** – MPLAB X IDE does not use this category when building. Consider it as a means to document a project's dependency on a header file, and a convenient method to access these files. You can double click on a file in the Projects window to open the file in an editor.
- **Linker Files** – You do not need to add a linker script to your project, because the project language tools will find the appropriate generic linker script for your device. So, this folder will be empty, unless you have created your own linker script.

  There should be only one file in this folder. If you have more than one linker script, only the first one has any effect – it is the linker script that the tool will use in the link step.
- **Source Files** – These are the only files that the toolchain will accept as input to its file commands.
- **Important Files** – Any file that does not fit into any of the other categories will end up in this folder.
  You can add project-specific data sheets (PDFs) to this location in the Projects window. Then, you can double click on the PDF to launch the data sheet. (This requires that a PDF reader is installed.)
- **Libraries** – The toolchain should take all of the files in this folder, as well as the object files, and include them in the final link step. For more information see **Section 4.11 "Add and Set Up Library and Object Files"**.
- **Loadables** – Projects and files to combine with or replace your project hex files. For more information see **Section 5.4 "Loadable Projects and Files"**.

12.11.2 Projects Window – Project Menu

Right click on a project name in the Projects window to pop up the Project menu. Menu items are shown in Table 12-15.

TABLE 12-15: PROJECT CONTEXT MENU ITEMS

Menu Item	Description
New	Add a new item. For MPLAB® X IDE, embedded file types are available.
Add Existing Item	Add an existing file to the project. The path to the file may be auto (selected by MPLAB X IDE), absolute or relative. Additionally, you may choose to copy the file into the project folder.
Add Existing Items from Folders	Add all files contained in the specified folder(s). You may also specify a pattern to ignore certain folders from inclusion. Click **Default** to see the default pattern. See also *Tools>Options*, **Miscellaneous** button, **Files** tab, "Files ignored by the IDE".
New Logical Folder	Create a new logical folder. To view actual folders, see the Files window.
Locate Headers	Locates all the header files (.h) called out in the C code project files and presents them in a checklist window so they may be added to the project. There are several reasons why you might want to add headers to your project: • The files can be opened from the Projects window under "Header Files" instead of hunting for them on your computer. • The project created by "Save Project As" will not contain header files unless they are added to the project; therefore this project may not build if any user-generated header files are required. • The zipped project created by "Package Project as .zip" will not contain header files unless they are added to the project; therefore the unzipped project may not build if any user-generated header files are required. This feature is able to locate user-generated header files for all Microchip toolchains.
Add Item to Important Files	Add an existing item to the project. The path to the file may be auto (selected by MPLAB X IDE), absolute or relative. Additionally, you may choose to copy the file into the project folder. This is useful for adding simulator files, data sheet PDFs, and other files to the project for your reference.
Export Hex	Export the project build as a Hex file. In MPLAB IDE v8, export is an extraction of memory objects. In MPLAB X IDE, export is the hex file result of a build; therefore, it needs to include all necessary auxiliary memory settings for configuration and EEPROM in code.
Build	Build the project according to the options selected in the Project Properties window. **Note:** When you Debug or Run, your project is built automatically.
Clean and Build	Clean and then Build the project according to the options selected in the Project Properties window. **Note:** When you Debug or Run, your project is built automatically.
Clean	Clean the project by deleting the outputs of previous builds.

TABLE 12-15: PROJECT CONTEXT MENU ITEMS (CONTINUED)

Menu Item	Description
Package	Package the current project into a .zip file. While MPLAB X IDE has the ability to zip files up, it cannot unzip them. So a separate program will be required to unzip the project. To package the project, this feature examines the project file to determine the location of the project files to include. Only files contained in the project folder and those using relative, not absolute, paths will be included. Items included in the package are the source files, makefile, and nbproject directory.
Set Configuration	Opens the Project Properties dialog so you can set the project configuration.
Run	Execute the project code. For details, see **Section 4.15 "Run Code"**.
Debug	Execute the project code in a debug environment. For details, see **Section 4.16 "Debug Run Code"**.
Step In	Step through Paused code running in the debug environment. For details, see **Section 4.18 "Step Through Code"**.
Make and Program Device	Program the target device and hold in Reset (do not run).
Set as Main Project	Set this project as the main project. Useful when you are working with multiple projects. See also **Section 6.3 "Work with Multiple Projects"**.
Open Required Projects	Load all other projects that are required for the selected project to run.
Close	Close selected project.
Rename	Rename the project.
Move	Move the project to another location. Along with the project, move the source files that are inside the project directory. Files outside the project directory are not moved. However, the project still maintains reference to the files outside the project directory. This is by design.
Copy	Create a copy of the project. If the source files are within the project directory, the source files are also copied to the new location. Files outside the project directory are not copied. However, the project still maintains reference to the files outside the project directory. This is by design.
Delete	Delete the project files. The project file is deleted but not the contents under the project directory. Only on selecting "Also delete sources" are all the source files deleted.
Code Assistance	Select assistance in creating code: code folding, code completion, etc.
Find	Find specified text in files in this project.
Share on Team Server	Share this project on a Team Server. For details, see **Section 5.20 "Collaborate on Code Development and Error Tracking"**.
Versioning	Control this project's version by using a version control system. For details, see **Section 5.19 "Control Source Code"**.
Local History	View or revert to the local history for the project. For details, see **Section 5.19 "Control Source Code"**.
Properties	Set project properties. For MPLAB X IDE, the Project Properties window is specific to embedded development. See **Section 12.10 "Project Properties Window"**.

12.11.3 Projects Window – File Menu

Right click on a file name in the Projects window to pop up the File menu. Table 12-16 shows the specific menu items.

TABLE 12-16: FILE CONTEXT MENU ITEMS

Menu Item	Description
Open	Open this file in a tabbed Editor window.
Cut	Remove the file from the project but place a copy of it on the clipboard.
Copy	Place a copy of the file on the clipboard.
Paste	Paste the clipboard copy of a file into the project.
Compile File	Compile only this file.
Remove from Project	Remove the file from the project. This does not delete the file from the PC. To the delete the file from the project and the computer, use the Delete key.
Rename	Rename the file.
Save as Template	Save this file as a template file.
Local History	View or revert to the local history for the file. For details, see **Section 5.19 "Control Source Code"**.
Tools	File tools include: • **Apply Diff Patch:** Apply an existing patch created by Diff. • **Diff to:** Show the differences between this file and the one specified here. • **Add to Favorites:** Add this file to the Favorites window.
Properties	Set file properties differently from the project properties. Select to exclude the file from the build or override the project build options by selecting a different configuration. See **Section 4.12 "Set File and Folder Properties"** for details.

12.11.4 Projects Window – Window Menu

Right click in an empty area in the Projects window to pop up the Window menu. Table 12-17 shows the specific menu items.

TABLE 12-17: FILE CONTEXT MENU ITEMS

Menu Item	Description
New Project	Launch the New Project wizard. For more information, see **Section 4.2 "Create a New Project"**.
New File	Launch the New File wizard. For more information, see **Section 4.8 "Create a New File"**.
Open Project	Open an existing project.
Open Recent Project	Open an existing project for the list of recent projects.
Open Project Group	Open an existing project group from the list of project groups.
Run Project	Run the main project.
Set Main Project	Set the main project from the list of open projects.

12.12 TOOLS OPTIONS EMBEDDED WINDOW

Open this window using *Tools>Options* (*mplab_ide>Preferences* for Mac OS X).

The Options window is a NetBeans window. However, it has been customized for MPLAB X IDE projects through the addition of an **Embedded** button. After clicking on this button, the following tabs and options will be available.

- Build Tools Tab
- Project Options Tab
- Generic Settings Tab
- Suppressible Messages Tab
- Diagnostics Tab
- Other Tab

12.12.1 Build Tools Tab

The information on this tab is accessed differently for Mac computers. Access the build tools from *mplab_ide>Preferences* from the main menu bar. See **Section 3.3.5 "Set Language Tool Locations"**.

Ensure that you have INSTALLED THE LANGUAGE TOOL or it will not show up on the "Toolchain" list. If you know you have installed it but it is not on the list, click **Scan for compilers**. If it is still not found click **Add** to add the tool to the list.

The following language tools are included with MPLAB X IDE:

- MPASM toolchain – includes MPASM assembler, MPLINK linker and utilities.

Other tools may be obtained from the Microchip web site (www.microchip.com) or third parties.

TABLE 12-18: BUILD TOOLS TAB ITEMS

Item	Description
Toolchain	Shows a list of language tools installed on your computer. Select the tool for your project and ensure that the paths (that apply) to the right are correct. **Base directory:** Path to the tool's main folder **C compiler:** Path to the C compiler (if available) **Assembler:** Path to the assembler (if available) **Make command:** Name of the make command generated by MPLAB® X IDE.
Add	Add a new language tool item to the list. Also consider using **Scan for Build Tools**.
Remove	Remove a language tool item from the list. This does not remove the language tool from the computer.
Default	Click on a tool and then click **Default** to make this tool the default compiler/assembler for the selected device.
Scan for Build Tools	Scan the computer for installed compilers/assemblers in various default locations, ***not*** the whole system. If you install in a different location, add the compiler manually.

12.12.2 Project Options Tab

Set options related to the project.

TABLE 12-19: PROJECT OPTIONS TAB ITEMS

Item	Description
Make Options	Enter make options to use when building projects. These options are toolchain dependent. See your language tool documentation.
File Path Mode	Specify how to store file path information in a project. **Auto:** Paths to files inside project folder stored as relative; paths to files outside project folder stored as absolute. **Always Relative:** All paths stored as relative to project folder. **Always Absolute:** All paths stored as absolute (full path).
Save All Modified Files Before Running Make	If selected, saves all unsaved files in the IDE before running `make`. It is recommended to leave this property selected because modifications to files in the IDE are not recognized by make unless they are first saved to disk.
Show binary files in Project view	If selected, the Projects view shows all files in a directory tree, including binary objects. This option is most relevant to projects created with existing sources, which might place sources and binaries in the same location. Deselect this option to more easily see your C source files and header files.
Show profiler indicators during run (new projects only)	If selected, profiler tools such as CPU Usage and Memory Usage are set up to run by default when newly created projects are run. The tools that are shown are determined by the Profile Configuration selected in *Tools>Profiler Tools*.
Use parallel make (`make -j 2`*n*)	If selected, `make` will execute several processes at a time, where `-j` (or `--jobs`) is the option to run in parallel and `2`*n* is the number of processes, where *n* is the number of processors available on your computer. If your computer does not support parallel processing, parallel make will be disabled. If you wish, you can specify more processes by using "Make Options". Example: `-j 10`. **Note 1:** For MPLAB XC16 or MPLAB C30, Procedural Abstraction needs to be turned off to use parallel make (*File>Project Properties*, compiler category, "Optimizations" option category: uncheck "Unlimited procedural abstraction"). **Note 2:** MPASMX cannot run under parallel make, either as a toolchain or part of an MPLAB C18 project. The parallel make option is therefore ignored in projects using MPASMX toolchain or in projects using the C18 toolchain that contain at least one `.asm` file.

12.12.3 Generic Settings Tab

Set up the log file and other project features.

TABLE 12-20: GENERIC SETTINGS TAB ITEMS

Item	Description
Projects Folder	Path and name of the folder where you will place your MPLAB® X IDE projects.
Close open source file on project close	Close any open source files in the Editor window when you close your project.
Clear output window before build	Clear out the contents of the Output window when you build your project.
Remove breakpoints upon importing a file	When importing a file into your project, remove all existing project breakpoints.
Halt build on first failure	When building, halt the process on the first failure. The selected project language tool can be set up to determine which errors are produced. Go to the Project Properties dialog and select the language tool under “Categories”. Then look for through the Option Categories to find one that allows you to set up errors, warnings and/or messages.
Maintain active connection to hardware tool	If selected, keep hardware tool connected always, not just at runtime (MPLAB® IDE v8 behavior). When switching projects with this option selected (e.g., when developing bootloading applications), ensure tool and device are the same to avoid error messages.
Clear tool output window on new session (debug, program, upload)	Clear out the contents of the Output window when you begin a Run, Debug Run or upload.
Silent build	Build without generating messages in the Output window.
Enable alternate watch list views during debug session	Display three watch view diamonds in the Watches window. Associate a watch view with a watch variable. When you click on a watch view diamond, only the variables associated with that view will be displayed. So, this feature works like a filter.
Reset @	Select action on Reset. **Main:** Stop at main() on Reset **Reset Vector:** Stop at the Reset vector on Reset
Debug start-up	Select action on debug start. **Run:** Start execution immediately **Main:** Stop at main() **Reset Vector:** Stop at the Reset vector
Default Charset	Select the default character set for the project.

12.12.4 Suppressible Messages Tab

Select error and warming messages to suppress. Double click on available folders to drill down to available items to suppress.

12.12.5 Diagnostics Tab

Set up the log file and other diagnostic features. See **Section 6.6 "Log Data"**.

TABLE 12-21: GENERIC SETTINGS TAB ITEMS

Item	Description
Logging Level	Set the message logging level. **OFF:** No logging **SEVERE:** Log severe (error) messages only **WARNING:** Log warning messages only **INFO:** Log informational messages only **CONFIG:** Log configuration information only **FINE:** Log some module to module communication **FINER:** Log more module to module communication **FINEST:** Log all module to module communication
Log File	Path and name of log file.
USB Circular Log	Applies only to MPLAB REAL ICE in-circuit emulator, MPLAB ICD 3 and PICkit 3.
Start New Logging Session/Pause Logging	Click buttons to begin a new logging session or pause the logging session in progress.
Circular USB log file location	Specify the path for the log file.
Circular USB log file max size in KBs	Specify the size of the log file.

12.12.6 Other Tab

Edit the lists of accepted file extensions for C/C++ and assembler source files and header files. Also, set the default extension for each type.

Note: MPLAB X IDE does not support Fortran programming.

12.13 TRACE WINDOW

Tracing allows you to record the step-by-step execution of your code and examine this recording. Trace is currently available for the following tools:

- Simulator
- MPLAB REAL ICE in-circuit emulator

Right clicking on a trace column in the window will pop up the context menu (Table 12-22). Depending on the tool you are using, you may or may not see all menu items.

Dialogs associated with trace are defined in Table .

TABLE 12-22: TRACE WINDOW CONTEXT MENU

Menu Item	Description
Symbolic Mode	For the “Instruction” column, toggle between displaying literal register addresses (e.g., 0x5) or symbolic register macros (e.g., PORTA).
Go To	**Trigger:** Move to the trigger line (0). **Top:** Move to the top trace line. **Bottom:** Move to the bottom trace line. **Trace Line:** Specify and go to a trace line location. Opens a Go To dialog.
Go To Source Line	Select a trace line and then select this option to go to the corresponding line in source code.
Display Time	For the “Cycle” column (will display if not previously visible): **As Hex Cycle Count:** Display cycle count as hexadecimal **As Decimal Cycle Count:** Display cycle count as decimal **In Seconds Elapsed:** Display cycle count in seconds elapsed **In Engineering Format:** Display cycle count in the appropriate engineering format (powers of 10^3).
Clear Trace File	Clear the data in the trace display.
Find	Find items in trace display. Opens a Find dialog.
Output To File	Save the trace data to a file. Opens Define Range dialog, which opens a Save dialog.
Print	Print the data. Opens a Print dialog.
Adjust Table Columns	Auto-adjust the columns in the trace display to fit the data.
Reload View	Reload the original data view for the trace display at pause.

TABLE 12-23: TRACE DIALOGS

Dialog	Description
Go To	Specify a trace line to go to.
Find	Find a line number or other data in the trace display.
Output to File Range	Specify a range of lines to output to a file. Click **OK** to proceed to the Save dialog to save the data to a text file.
Save	Save data to a text file.
Print	Specify the printer, page setup and appearance before printing.

12.14 WATCHES WINDOW

Use the Watches window to watch the values of symbols that you select change. To open the Watches window, select *Window>Debugging>Watches*.

12.14.1 Watches Operation

For information on using the Watches window, see:

- **Section 4.19 "Watch Symbol Values Change"**
- the NetBeans Help topic *C/C++/Fortran Development>Debugging C/C++/Fortran Applications with gdb>Viewing C/C++/Fortran Program Information>Creating a C/C++/Fortran Watch*.

12.14.2 Watches Menu

Right click on a row to open the Watches menu. The items available will depend on whether a symbol is in the row or what debug tool you are using.

TABLE 12-24: WATCHES WINDOW MENU ITEMS

Menu Item	Description
New Watch	Add a new symbol to watch
New Runtime Watch	*MPLAB REAL ICE In-Circuit Emulator Only* Add a new runtime watch for the selected symbol.
Run Time Update Interval	*MPLAB REAL ICE In-Circuit Emulator Only* Specify the rate at which the symbol value will be updated. "No Delay" will update as quickly as your personal computer is able. If you are seeing errors in the runtime data, you can add a delay to decrease the update speed.
Export All Watches to List File	Export information about the symbols to be watched to a file (`.xwatch`).
Delete All	Remove all the watched symbols from the Watches window.

Note: You can also right-click in a symbol in code to add it to a New Watch or New Runtime Watch.

12.14.3 Watches Display

The display has the following features:

- Icons
- Columns
- Actions

12.14.3.1 ICONS

Icons are found to the left of the name object in the Name column:

TABLE 12-25: NAME COLUMN ICONS

Icon	Description
	Watch object
	Watch object in Program Memory
	Static field of an object
	Non-static field of an object

12.14.3.2 COLUMNS

You can change the columns displayed in the window by right clicking on a heading to pop up the “Change Visible Columns” dialog.

TABLE 12-26: CHANGE VISIBLE COLUMNS ITEMS

Dialog Item	If Checked
Name	Show name of column (always checked)
Address	Show memory address of variable
Binary	Show binary formatted value
Char	Show character formatted value
Decimal	Show decimal formatted value
Type	Show type of watch variable
Value	Show value of watch expression (in hexadecimal)

12.14.3.3 ACTIONS

Actions are on buttons on the left side of the window:

TABLE 12-27: NAME COLUMN ICONS

Button	Action
	Toggle button: - Show verbose (qualified) names of member fields. - Show brief (relative) names of member fields.
	Import watches from list file. Right click for options.
	Export all watches to list file.
	Set the default numeric format for the Value field.

12.15 WIZARD WINDOWS

MPLAB X IDE uses many NetBeans windows. However, some windows have been modified or created specifically for embedded use.

- New Project wizard – see **Section 4.2 "Create a New Project"**
- New File wizard – see **Section 4.8 "Create a New File"**
- Import ImageFile wizard – see **Section 5.3.3 "Import Image File Wizard"**

Chapter 13. NetBeans Windows and Dialogs

13.1 INTRODUCTION

MPLAB IDE windows are a combination of basic NetBeans windows and MPLAB IDE specific windows. Dialogs open when selected from menu items. As with windows, MPLAB IDE dialogs are a combination of basic NetBeans dialogs and MPLAB IDE specific dialogs.

NetBeans windows and dialogs are discussed here, with references to documentation in the Help. For information on MPLAB X IDE specific windows and dialogs, see **Chapter 12. "MPLAB X IDE Windows and Dialogs"**.

- NetBeans Specific Windows and Window Menus
- NetBeans Specific Dialogs

13.2 NETBEANS SPECIFIC WINDOWS AND WINDOW MENUS

NetBeans windows are discussed in NetBeans help. To open help on a window, click on the **Window** tab to select it and then hit the <F1> key. If no help can be found, select *Help>Help Contents* and click the help file's **Search** tab to search for information on that window. Or, see the NetBeans help topic "Managing IDE Windows".

To open most windows, see the **Section 11.2.11 "Window Menu"**.

Windows may be docked and undocked (right click on the **Window** tab) and have window-specific pop-up, or context, menus with items such as Fill, Goto, and Edit Cells. Right clicking in a window or on an item in the window will pop up a context menu. Most content menu items are also located on menus on the desktop menu bar (see **Section 11.2 "Menus"**).

Set up window properties by selecting *Tools>Options* (*mplab_ide>Preferences* for Mac OS X), **Miscellaneous** button, **Appearance** tab.

13.3 NETBEANS SPECIFIC DIALOGS

NetBeans dialogs are discussed in NetBeans help. To open help on a dialog, click the **Help** button on the dialog or, if there is no **Help** button, press the <F1> key. If no help can be found, select *Help>Help Contents* and click the help file's **Search** tab to search for information on that dialog.

To open most dialogs, see **Section 11.2 "Menus"**.

NOTES:

Appendix A. Configuration Settings Summary

A.1 INTRODUCTION

Examples of how to set Configuration bits in code for different language tools and related devices are shown below. For more information on how to set Configuration bits, see your language tool documentation. For some language tools, a configurations settings document is available listing all configuration settings for a device. Otherwise, consult your device header file for macros.

Another option is to use the Configuration Memory window to set bits and then click "Generate Source Code to Output". See **Section 4.21.4 "Set Configuration Bits"**.

- MPASM Toolchain
- HI-TECH® PICC™ Toolchain
- HI-TECH® PICC-18™ Toolchain
- C18 Toolchain
- ASM30 Toolchain
- C30 Toolchain
- C32 Toolchain
- XC Toolchains

A.2 MPASM TOOLCHAIN

Two types of assembler directives are used to set device configuration bits: `__config` and `config`. DO NOT mix `__config` and `config` in the same code.

A.2.1 __config

The directive `__config` is used for PIC10/12/16 MCUs. It may be used for PIC18 MCUs (excluding PIC18FXXJ devices) but the `config` directive is recommended. The syntax is as follows:

```
__config expr        ;For a single configuration word
```

or

```
__config addr, expr  ;For multiple configuration word
```

where:

addr: Address of the Configuration Word. May be literal but usually represented by a macro.
Note: Macros *must* be listed in ascending register order.

expr: Expression representing the value to which the specified Configuration bits will be set. May be literal but usually represented by a macro or macros ANDed together.

Macros are specified in the device include file (`*.inc`) that is located in the Windows operating system (OS) default directory:

```
C:\Program Files\Microchip\MPLABX\mpasmx
```

Directive case does not matter; `__CONFIG` or `__config` is acceptable. Macro case should match what is in the header.

Example – PIC10/12/16 MCUs

```
#include p16f877a.inc

;Set oscillator to HS, watchdog time off, low-voltage prog. off
__CONFIG _HS_OSC & _WDT_OFF & _LVP_OFF
```

Example – PIC18 MCUs

```
#include p18f452.inc

;Oscillator switch enabled, RC oscillator with OSC2 as I/O pin.
__CONFIG    _CONFIG1, _OSCS_OFF_1 & _RCIO_OSC_1

;Watch Dog Timer enable, Watch Dog Timer PostScaler count - 1:128
__CONFIG    _CONFIG3, _WDT_ON_3 & _WDTPS_128_3
```

A.2.2 config

The directive `config` is used for PIC18 MCUs (including PIC18FXXJ devices). The syntax is as follows:

`config` *`setting=value`* [, *`setting=value`*]

where:

`setting:`	Macro representing a Configuration bit or bits.
`value:`	Macro representing the value to which the specified Configuration bit(s) will be set. Multiple settings may be defined on a single line, separated by commas. Settings for a single configuration byte may also be defined on separate lines.

Macros are specified in the device include file (`*.inc`) that is located in the Windows OS default directory:

`C:\Program Files\Microchip\MPLABX\mpasmx`

Directive case does not matter; `__CONFIG` or `__config` is acceptable. Macro case should match what is in the header.

Example – PIC18 MCUs

```
#include p18f452.inc

;Oscillator switch enabled, RC oscillator with OSC2 as I/O pin.
CONFIG     OSCS=ON, OSC=LP

;Watch Dog Timer enable, Watch Dog Timer PostScaler count - 1:128
CONFIG     WDT=ON, WDTPS=128
```

A.3 HI-TECH® PICC™ TOOLCHAIN

A macro specified in the `htc.h` header file is used to set device Configuration Words for PIC10/12/16 MCUs:

`__CONFIG(x);`

where

x: Expression representing the value to which the specified Configuration bits will be set. May be literal but usually represented by a macro or macros ANDed together.

Macros are specified in the device header file (`*.h`) that is located in the Windows OS default directory:

`C:\Program Files\HI-TECH Software\PICC\version\include`

where *version* is the version number of the compiler.

For devices that have more than one Configuration Word location, each subsequent invocation of `__CONFIG()` will modify the next Configuration Word in sequence.

Macro case should match what is in the relevant header. For `htc.c`, `__CONFIG()` is correct but `__config()` is not.

PICC Example

```
#include <htc.h>

__CONFIG(WDTDIS & XT & UNPROTECT); // Program config. word 1
__CONFIG(FCMEN);                   // Program config. word 2
```

A.4 HI-TECH® PICC-18™ TOOLCHAIN

A macro specified in the `htc.h` header file is used to set device Configuration Words for PIC18 MCUs:

`__CONFIG(n,x);`

where

n: Configuration register number

x: Expression representing the value to which the specified Configuration bits will be set. May be literal but usually represented by a macro or macros ANDed together.

Macros are specified in the device header file (`*.h`) that is located in the Windows OS default directory:

`C:\Program Files\HI-TECH Software\PICC\version\include`

where *version* is the version number of the compiler.

Macro case should match what is in the relevant header. For `htc.c`, `__CONFIG()` is correct but `__config()` is not.

PICC-18 Example

```
#include <htc.h>

//Oscillator switch enabled, RC oscillator with OSC2 as I/O pin.
__CONFIG(1, LP);

//Watch Dog Timer enable, Watch Dog Timer PostScaler count - 1:128
__CONFIG(2, WDTEN & WDTPS128);
```

A.5 C18 TOOLCHAIN

The `#pragma config` directive specifies the device-specific configuration settings (i.e., Configuration bits) to be used by the application:

`#pragma config` *`setting-list`*

where

`setting-list:` A list of one or more *`setting-name`* = *`value-name`* macro pairs, separated by commas.

Macros are specified in the device header file (`*.h`) that is located in the Windows OS default directory:

`C:\program files\microchip\mplabc18\version\.h`

Pragma case does not matter; either `#PRAGMA CONFIG` or `#pragma config` is acceptable. Macro case should match what is in the header.

Example

```
#include <p18cxxx.h>

/*Oscillator switch enabled, RC oscillator with OSC2 as I/O pin.*/
#pragma config OSCS = ON, OSC = LP

/*Watch Dog Timer enable, Watch Dog Timer PostScaler count - 1:128*/
#pragma config WDT = ON, WDTPS = 128
```

A.6 ASM30 TOOLCHAIN

A macro specified in the device include file is used to set Configuration bits:

`config` *`__reg, value`*

where

`__reg:` Configuration register name macro.

`value:` Expression representing the value to which the specified Configuration bits will be set. May be literal but usually represented by a macro or macros ANDed together.

Macros are specified in the device include file (`*.inc`) that is located in the Windows OS default directory:

`C:\Program Files\Microchip\MPLAB ASM30 Suite\Support\`*`device`*`\inc`

where *`device`* is the abbreviation of the selected 16-bit device, such as PIC24H or dsPIC33F.

Macro case should match what is in the relevant header. For example, `config` is correct but `CONFIG` is not.

Example

```
.include "p30fxxxx.inc"

;Clock switching off, Fail-safe clock monitoring off,
; Use External Clock
config __FOSC, CSW_FSCM_OFF & XT_PLL16

;Turn off Watchdog Timer
config __FWDT, WDT_OFF
```

A.7 C30 TOOLCHAIN

Two types of compiler macros are used to set device Configuration bits: one type for PIC24F MCUs and one type for dsPIC30F and dsPIC33F/PIC24H devices.

A.7.1 PIC24F Configuration Settings

Macros are provided in device header files to set Configuration bits:

`_config`*`n`*`(`*`value`*`);`

where

n: Configuration register number.

value: Expression representing the value to which the specified Configuration bits will be set. May be literal but usually represented by a macro or macros ANDed together.

Macros are specified in the device header file (`*.h`) that is located in the Windows OS default directory:

`C:\Program Files\Microchip\MPLAB C30\support\PIC24F\h`

Macro case should match what is in the relevant header. For example, `_CONFIG1` is correct but `_config1` is not.

Example – PIC24F MCUs

```
#include "p24Fxxxx.h"

//JTAG off, Code Protect off, Write Protect off, COE mode off, WDT off
_CONFIG1( JTAGEN_OFF & GCP_OFF & GWRP_OFF & COE_OFF & FWDTEN_OFF )

//Clock switching/monitor off, Oscillator (RC15) on,
// Oscillator in HS mode, Use primary oscillator (no PLL)
_CONFIG2( FCKSM_CSDCMD & OSCIOFNC_ON & POSCMOD_HS & FNOSC_PRI )
```

A.7.2 dsPIC30F/33F/PIC24H Configuration Settings

Macros are provided in device header files to set Configuration bits:

`_reg``(`*`value`*`);`

where

_reg: Configuration register name macro.

value: Expression representing the value to which the specified Configuration bits will be set. May be literal but usually represented by a macro or macros ANDed together.

Macros are specified in the device header file (`*.h`) that is located in the Windows OS default directory:

`C:\Program Files\Microchip\MPLAB C30\support\`*`device`*`\h`

where *device* is the abbreviation of the selected 16-bit device, i.e., PIC24H, dsPIC30F, or dsPIC33F.

Macro case should match what is in the relevant header. For example, `_FOSC` is correct but, `_fosc` is not.

Example – dsPIC30F DSCs

```
#include "p30fxxxx.h"

//Clock switching and failsafe clock monitoring off,
// Oscillator in HS mode
_FOSC(CSW_FSCM_OFF & HS);

//Watchdog timer off
_FWDT(WDT_OFF);

//Brown-out off, Master clear on
_FBORPOR(PBOR_OFF & MCLR_EN);
```

Example – dsPIC33F/PIC24H Devices

```
#include "p33fxxxx.h"

// Use primary oscillator (no PLL)
_FOSCSEL(FNOSC_PRI);

//Oscillator in HS mode
_FOSC(POSCMD_HS);

//Watchdog timer off
_FWDT(FWDTEN_OFF);

//JTAG off
_FICD(JTAGEN_OFF);
```

A.8 C32 TOOLCHAIN

The `#pragma config` directive specifies the device-specific configuration settings (i.e., Configuration bits) to be used by the application:

```
# pragma config setting-list
```

where

setting-list: A list of one or more *setting-name* = *value-name* macro pairs, separated by commas.

Macros are specified in the device header file (`*.h`) that is located in the Windows OS default directory:

```
C:\Program Files\Microchip\MPLAB C32\pic32mx\include\proc
```

Pragma case does not matter; either `#PRAGMA CONFIG` or `#pragma config` is acceptable. Macro case should match what is in the header.

Example

```
#include "p32xxxx.h"

//Enables the Watchdog Timer,
// Sets the Watchdog Postscaler to 1:128
#pragma config FWDTEN = ON, WDTPS = PS128

//Selects the HS Oscillator for the Primary Oscillator
#pragma config POSCMOD = HS
```

A.9 XC TOOLCHAINS

To create code that is as portable as possible, refer to the Common Compiler Interface (CCI) chapter, config macro, in each of the following documents:

- MPLAB XC8 C Compiler User's Guide (DS52053) or related help file
- MPLAB XC16 C Compiler User's Guide (DS52071) or related help file
- MPLAB XC32 C Compiler User's Guide (DS51686) or related help file

NOTES:

Appendix B. Working Outside the IDE

B.1 INTRODUCTION

MPLAB X IDE is designed to help you write, debug and release applications for embedded systems. However, your company may have requirements that make code development outside the IDE necessary.

You can build your code outside MPLAB X IDE following the instructions in this chapter. If you need to debug this code, you can import it into an MPLAB X IDE project. See **5.3 "Prebuilt Projects"** and **5.4 "Loadable Projects and Files"** for more information.

Alternatively, you can compile your code with debug information and use the command-line Microchip Debugger (MDB), which comes with each version of MPLAB X IDE, to debug your code. For more on MDB, see the *Microchip Debugger (MDB) User's Guide* (DS50002102), found in the `<MPLAB X IDE installation>/docs` directory.

- Building a Project Outside of MPLAB X IDE
- Compiling for Debug Outside of MPLAB X IDE

B.2 BUILDING A PROJECT OUTSIDE OF MPLAB X IDE

MPLAB X IDE uses the GNU make as its build tool. For Linux, you are expected to have make installed. For Mac OS X and Windows operating systems, the installation of MPLAB X IDE provides this program. The locations are:

- **Windows 32-Bit OS** - `C:\Program Files\Microchip\MPLABX\gnuBins\GnuWin32\bin`
- **Windows 64-Bit OS** - `C:\Program Files (x86)\Microchip\MPLABX\gnuBins\GnuWin32\bin`
- **Mac OS X** - `/Applications/microchip/mplabx/mplab_ide.app/Contents/Resources/mplab_ide/bin`

MPLAB X IDE automatically adds the directory containing the make to its own path variable. If you want to build outside of the IDE, you must add the directory to the PATH environmental variable.

The `Makefile` in the MPLAB X IDE project directory can be called directly to build the default configuration:

Command - Type on a single line	Description
`$ make clean`	To remove all intermediate objects and final images
`$ make`	To create the production image (Hex file)
`$ make TYPE_IMAGE=DEBUG_RUN`	To create the debug image (COF/ELF file)

If the project has more than one configuration, then:

Command - Type on a single line	Description
`$ make -f Makefile CONF=Configuration clean`	To remove all intermediate objects and final images for configuration `Configuration`
`$ make -f Makefile CONF=Configuration`	To create the production image (Hex file) for configuration `Configuration`
`$ make -f Makefile CONF=Configuration TYPE_IMAGE=DEBUG_RUN`	To create the debug image (COF/ELF file) for configuration `Configuration`

The names of the images by default are (with respect to the MPLAB X IDE directory):

```
dist/$CONF_NAME/production/$PROJ_NAME.production.hex
dist/$CONF_NAME/debug/$PROJ_NAME.debug.cof (or elf)
```

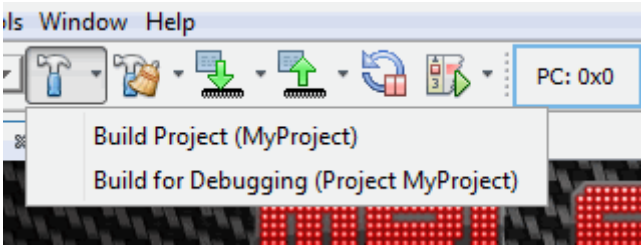
B.3 COMPILING FOR DEBUG OUTSIDE OF MPLAB X IDE

To compile code for debugging outside of MPLAB X IDE, you must ensure you pass the correct parameters to the compiler and/or the linker to reserve the areas required by the specific tool being used. The parameter values depend on the debug tool being used and on the compiler being used. Furthermore, the version of the compiler might determine the specific mechanism used to reserve these areas.

MPLAB X IDE has the knowledge to determine what needs to be passed to the compiler/assembler/linker. So, the best way to find out how to reserve areas for debugging is to create a small project in the IDE using the desired device, debug tool and compiler, remembering that the IDE can support multiple versions of the same compiler. So, while creating the project in the IDE, make sure you select the correct compiler version.

After you have created the small project, build the project for debugging. Save the text in the output window showing the build steps for debugging. Then build the project again (not for debugging) and save the text in the output window showing the build steps for production. Compare the two sets of instructions being passed to the compiler/assembler/linker.

FIGURE B-1: BUILD ICON MENU



Some examples of the differences in the output window are:

- Compiler/assembler/linker being called with the `__DEBUG` macro.
- Compiler/assembler/linker being called with the `__MPLAB_DEBUG` macro.
- Linker being called with `__ICD2_RAM` or `-mreserves`. These two are mutually exclusive, i.e., the linker is called with one of them but not the other.
- Compiler/assembler/linker being called with the name of the debug tool to be used.

This is not an exhaustive list.

NOTES:

Appendix C. Revision History

Revision A (November 2011)

- Initial release of this document.

Revision B (October 2012)

- JRE installation now automatic with Windows and Linux operating systems. Mac operating system instructions provided. Discussed in Chapter 2. "Before You Begin", 2.2 "Install JRE and MPLAB X IDE".
- Corrected information on device driver names and paths in Chapter 2.
- Added description of meaning of the two lights next to a debug tool in Chapter 3. "Tutorial" and Chapter 4. "Basic Tasks".
- Updated Icon definitions in Chapter 3. "Tutorial" and Chapter 4. "Basic Tasks".
- Added a list of language toolchain abbreviations in Chapter 4. "Basic Tasks", 4.3 "Create a New Project", 4.3.5 "Step 5: Select Compiler".
- Multiple dialogs updated to show projects using the new MPLAB XC C compiler.
- Added information on the Libraries category in the Project Properties dialog in Chapter 4. "Basic Tasks", 4.12 "Add and Setup Library and Object Files".
- Added information on normalizing a hex file under Chapter 4. "Basic Tasks", 4.14 "Set Build Properties".
- Added information on how to add a literal value to a Watches window in Chapter 4. "Basic Tasks", 4.20 "Watch Symbol Values Changes".
- Updated several sections for recent support of C++ (currently for the MPLAB XC32++ Compiler).
- Added section on loadable projects in Chapter 5. "Additional Tasks", 5.5 "Loadable Projects and Files".
- Discussed how to import embedded projects from other applications into MPLAB X IDE in Chapter 5. "Additional Tasks", 5.7 "Other Embedded Projects".
- Added information on working with samples projects in Chapter 5. "Additional Tasks", 5.8 "Sample Projects".
- Described how to work with other files, like XML, in Chapter 5. "Additional Tasks", 5.9 "Work with Other Types of Files".
- Added information on how to work with log files in Chapter 6. Advanced Tasks", 6.4 "Log Data".
- Described how to add functions to a toolbar in Chapter 6. Advanced Tasks", 6.5 "Customize Toolbars".
- Added Chapter 7 Editor, that discusses MPLAB X IDE Editor usage, options and features.
- Described an error message you might receive if you modify or move the default linker script, and how to avoid this. See Chapter 8. "Troubleshooting", 8.4 "MPLAB X IDE Issues".
- Combined "Major Differences" and "Feature Differences" into Chapter 9. "MPLAB X IDE vs. MPLAB IDE v8", 9.2 "Major Differences". Explained NetBeans platform is open source, but MPLAB X IDE is proprietary.

- In Chapter 9. "MPLAB X IDE vs. MPLAB IDE v8", updated 9.3 "Menu Differences" to reflect new functions in MPLAB X IDE and 9.4 "Tool Support Differences" to reflect new plug-in support.
- In Chapter 10. "Desktop Reference", updated 10.2 "Menus" to reflect new functions in MPLAB X IDE.
- Split "Windows and Dialogs" into Chapter 11. "MPLAB X IDE Windows and Dialogs" and Chapter 12. "NetBeans Windows and Dialogs" to better delineate the different windows. Updated window menus.
- Updated project file structure in Chapter 13. "Project Files and Folders", 13.2 "Files Window View". Added information on makefiles.
- Updated discussion of MPLAB XC C compiler configuration bits to reference the common c interface (CCI) in Chapter 14. "Configuration Settings Summary", 14.8 "XC Toolchains".

Revision C (March 2014)

- Overall Changes: Removed references to old tools. Added information on Windows 8. Updated screens and text to match the latest GUI.
- Chapter 1. "What is MPLAB X IDE?": Updated figure references. Mentioned stand-alone tool help and the Microchip wiki as resources.
- Chapter 2. "Before You Begin": Updated information in 2.3.2 "USB Driver Installation for Windows® XP/7/8 Operating Systems". Added compiler licensing information under 2.5 "Install the Language Tools". Added 2.7.1 "Setting Up Hardware Tools to Work with Multiple Instances".
- Chapter 4. "Basic Tasks": Removed 4.1 "Introduction". Added 4.6.1 "Add or Change a Toolchain" and 4.6.2 "About Toolchain Paths". Added how to exclude file/folders from build under 4.12 "Set File and Folder Properties". Expanded Table 4-3 into sections under 4.13 "Set Build Properties". Added 4.15.2 "Run Considerations". Added 4.16.2 "Debug Macros Generated" and 4.16.3 "Debug Considerations". Added symbol information in 4.19 "Watch Symbol Values Change". Updated 4.21.2 "Change Device Memory". Added 4.23.1 "Set Project Programming Properties".
- Chapter 5. "Additional Tasks": Removed 5.1 "Introduction". Updates 5.4 "Loadable Projects and Files" and created a new section, 5.5 "Loadable Projects and Files - Bootloaders". Added 5.12 "Modify Project Folders and Encoding" and 5.13 "Speed Up Build Times". Updated text to match new display options in 5.17 "View the Dashboard Display". Fixed using version control steps and updated project files that need to be saved into a repository under 5.19 "Control Source Code". Updated options for 5.20 "Collaborate on Code Development and Error Tracking". Reordered sections under 5.21 "Add Plug-In Tools" and added 5.21.4 "Plug-In Code Location".
- Chapter 6. Advanced Tasks: Added 6.1 "Speed Up MPLAB X IDE" and 6.5 "Create User MakeFile Projects". Added information about working without setting an active project and grouping projects under 6.3 "Work with Multiple Projects". Removed statement that debug configuration needed (it is not) from 6.4 "Work with Multiple Configurations". Log file requirements update under 6.6 "Log File".
- Chapter 7. "Editor": "Editor Features of Note" moved to 7.2.4 under 7.2 "Editor Usage". Also added 7.2.1. "Desktop Controls", 7.2.2 "Hyperlinks in C Code" and 7.2.3 "Hyperlinks in ASM Code". Macros definition expanded in Table 7-6 "Macros Tab". Rearranged 7.4 "Code Folding" into 7.4.1 "Code Folding Usage" (includes MPLAB C18 and assembly folding issue) and 7.4.2. "Custom Code Folding".

- Chapter 8. “Project Files and Folders”: This chapter moved from Chapter 13. Added 8.4 “Favorites Window View”, 8.5 “Classes Window View”, 8.6 “Viewing User Configuration Data” and 8.9 “Deleting a Project”. Changed “Moving a Project” to 8.8 “Moving, Copying or Renaming a Project”. Moved “Building a Project Outside of MPLAB X IDE” to Appendix B. “Working Outside the IDE”.
- Chapter 9. “Troubleshooting”: Added information to 9.6 “Errors”.
- Chapter 10. “MPLAB X IDE vs. MPLAB IDE v8”: Clarified under 10.2 “Major Differences”, item 3. “MPLAB X IDE allows multiple tool selection” and item 9. “MPLAB X IDE uses configuration bits set in code”. Other minor updates for MPLAB X IDE additional selections.
- Chapter 11. “Desktop Reference”: 11.2.10 “Tools Menu” updated for plug-ins and compiler licensing. Other minor updates for MPLAB X IDE additional selections.
- Chapter 12. “MPLAB X IDE Windows and Dialogs”: Added 12.2 “MPLAB X IDE Windows Management”. In 12.3 “MPLAB X IDE Windows with Related Menus and Dialogs”, added new windows to table and moved windows subsections to their own sections. Added 12.4 “Breakpoints Window” and 12.6 “Licenses Window”. Extensive updates to add descriptions of different memory windows under 12.8 “Memory Windows”. 12.11 “Projects Window” menus updates. 12.12 “Tools Options Embedded Window” options update, including the addition of 12.12.5 “Diagnostics Tab”. 12.14 “Watches Window” content broken up into three sections.
- Appendix A: “Configuration Settings”: Change from Chapter 14 to an appendix. Added information about the Configuration Memory window.
- Appendix B: “Working Outside the IDE”: B.1 “Building a Project Outside of MPLAB X IDE” from Chapter 8 “Project Files and Folders”. Added B.2 “Compiling for Debug Outside of MPLAB X IDE”.
- Appendix C: “Revision History”: This chapter updated for major changes from version B to C.

NOTES:

Support

INTRODUCTION

Please refer to the items discussed here for support issues.

- Warranty Registration
- myMicrochip Personalized Notification Service
- The Microchip Web Site
- Microchip Forums
- Customer Support
- Contact Microchip Technology

WARRANTY REGISTRATION

Web Site: http://www.microchipdirect.com

Registering your development tool entitles you to receive new product updates. Interim software releases are available at the Microchip web site.

myMICROCHIP PERSONALIZED NOTIFICATION SERVICE

myMicrochip: http://www.microchip.com/pcn

Microchip's personal notification service helps keep customers current on their Microchip products of interest. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool.

Please visit myMicrochip to begin the registration process and select your preferences to receive personalized notifications. A FAQ and registration details are available on the page, which can be opened by selecting the link above.

When you are selecting your preferences, choosing "Development Systems" will populate the list with available development tools. The main categories of tools are listed below:

- **Compilers** – The latest information on Microchip C compilers, assemblers, linkers and other language tools. These include all MPLAB C compilers; all MPLAB assemblers (including MPASM™ assembler); all MPLAB linkers (including MPLINK™ object linker); and all MPLAB librarians (including MPLIB™ object librarian).
- **Emulators** – The latest information on Microchip in-circuit emulators. This includes the MPLAB REAL ICE™ in-circuit emulator.
- **In-Circuit Debuggers** – The latest information on Microchip in-circuit debuggers. These include the PICkit™ 2, PICkit 3 and MPLAB ICD 3 in-circuit debuggers.
- **MPLAB® IDE** – The latest information on Microchip MPLAB IDE, the Windows®-based Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB IDE Project Manager, MPLAB Editor and MPLAB SIM simulator, as well as general editing and debugging features.
- **Programmers** – The latest information on Microchip programmers. These include the device (production) programmers MPLAB REAL ICE in-circuit emulator, MPLAB ICD 3 in-circuit debugger, MPLAB PM3 and development (nonproduction) programmers PICkit 2 and 3.
- **Starter/Demo Boards** – These include MPLAB Starter Kit boards, PICDEM demo boards, and various other evaluation boards.

THE MICROCHIP WEB SITE

Web Site: http://www.microchip.com

Microchip provides online support via our web site. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

MICROCHIP FORUMS

Forums: http://www.microchip.com/forums

Microchip provides additional online support via our web forums. Currently available forums are:

- Development Tools
- 8-bit PIC MCUs
- 16-bit PIC MCUs
- 32-bit PIC MCUs

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Technical support is available through the web site at:

http://support.microchip.com

Documentation errors or comments may be emailed to docerrors@microchip.com

CONTACT MICROCHIP TECHNOLOGY

You may call or fax Microchip Corporate offices at the numbers below:

Voice: (480) 792-7200

Fax: (480) 792-7277

NOTES:

Glossary

A

Absolute Section

A GCC compiler section with a fixed (absolute) address that cannot be changed by the linker.

Absolute Variable/Function

A variable or function placed at an absolute address using the OCG compiler's `@ address` syntax.

Access Memory

PIC18 Only – Special registers on PIC18 devices that allow access regardless of the setting of the Bank Select Register (BSR).

Access Entry Points

Access entry points provide a way to transfer control across segments to a function which may not be defined at link time. They support the separate linking of boot and secure application segments.

Address

Value that identifies a location in memory.

Alphabetic Character

Alphabetic characters are those characters that are letters of the arabic alphabet (a, b, …, z, A, B, …, Z).

Alphanumeric

Alphanumeric characters are comprised of alphabetic characters and decimal digits (0,1, …, 9).

ANDed Breakpoints

Set up an ANDed condition for breaking, i.e., breakpoint 1 AND breakpoint 2 must occur at the same time before a program halt. This can only be accomplished if a data breakpoint and a program memory breakpoint occur at the same time.

Anonymous Structure

16-bit C Compiler **–** An unnamed structure.

PIC18 C Compiler **–** An unnamed structure that is a member of a C union. The members of an anonymous structure may be accessed as if they were members of the enclosing union. For example, in the following code, `hi` and `lo` are members of an anonymous structure inside the union `caster`.

```
union castaway
 int intval;
 struct {
  char lo; //accessible as caster.lo
  char hi; //accessible as caster.hi
 };
} caster;
```

ANSI

American National Standards Institute is an organization responsible for formulating and approving standards in the United States.

Application

A set of software and hardware that may be controlled by a PIC® microcontroller.

Archive/Archiver

An archive/library is a collection of relocatable object modules. It is created by assembling multiple source files to object files, and then using the archiver/librarian to combine the object files into one archive/library file. An archive/library can be linked with object modules and other archives/libraries to create executable code.

ASCII

American Standard Code for Information Interchange is a character set encoding that uses 7 binary digits to represent each character. It includes upper and lower case letters, digits, symbols and control characters.

Assembly/Assembler

Assembly is a programming language that describes binary machine code in a symbolic form. An assembler is a language tool that translates assembly language source code into machine code.

Assigned Section

A GCC compiler section which has been assigned to a target memory block in the linker command file.

Asynchronously

Multiple events that do not occur at the same time. This is generally used to refer to interrupts that may occur at any time during processor execution.

Asynchronous Stimulus

Data generated to simulate external inputs to a simulator device.

Attribute

GCC Characteristics of variables or functions in a C program which are used to describe machine-specific properties.

Attribute, Section

GCC Characteristics of sections, such as "executable", "readonly", or "data" that can be specified as flags in the assembler `.section` directive.

B

Binary

The base two numbering system that uses the digits 0-1. The rightmost digit counts ones, the next counts multiples of 2, then $2^2 = 4$, etc.

Bookmarks

Use bookmarks to easily locate specific lines in a file.

Select Toggle Bookmarks on the Editor toolbar to add/remove bookmarks. Click other icons on this toolbar to move to the next or previous bookmark.

Breakpoint

Hardware Breakpoint: An event whose execution will cause a halt.

Software Breakpoint: An address where execution of the firmware will halt. Usually achieved by a special break instruction.

Build

Compile and link all the source files for an application.

C

C\C++

C is a general-purpose programming language which features economy of expression, modern control flow and data structures, and a rich set of operators. C++ is the object-oriented version of C.

Calibration Memory

A special function register or registers used to hold values for calibration of a PIC microcontroller on-board RC oscillator or other device peripherals.

Central Processing Unit

The part of a device that is responsible for fetching the correct instruction for execution, decoding that instruction, and then executing that instruction. When necessary, it works in conjunction with the arithmetic logic unit (ALU) to complete the execution of the instruction. It controls the program memory address bus, the data memory address bus, and accesses to the stack.

Clean

Clean removes all intermediary project files, such as object, hex and debug files, for the active project. These files are recreated from other files when a project is built.

COFF

Common Object File Format. An object file of this format contains machine code, debugging and other information.

Command Line Interface

A means of communication between a program and its user based solely on textual input and output.

Compiled Stack

A region of memory managed by the compiler in which variables are statically allocated space. It replaces a software or hardware stack when such mechanisms cannot be efficiently implemented on the target device.

Compiler

A program that translates a source file written in a high-level language into machine code.

Conditional Assembly

Assembly language code that is included or omitted based on the assembly-time value of a specified expression.

Conditional Compilation

The act of compiling a program fragment only if a certain constant expression, specified by a preprocessor directive, is true.

Configuration Bits

Special-purpose bits programmed to set PIC microcontroller modes of operation. A Configuration bit may or may not be preprogrammed.

Control Directives

Directives in assembly language code that cause code to be included or omitted based on the assembly-time value of a specified expression.

CPU

See Central Processing Unit.

Cross Reference File

A file that references a table of symbols and a list of files that references the symbol. If the symbol is defined, the first file listed is the location of the definition. The remaining files contain references to the symbol.

D

Data Directives

Data directives are those that control the assembler's allocation of program or data memory and provide a way to refer to data items symbolically; that is, by meaningful names.

Data Memory

On Microchip MCU and DSC devices, data memory (RAM) is comprised of General Purpose Registers (GPRs) and Special Function Registers (SFRs). Some devices also have EEPROM data memory.

Data Monitor and Control Interface (DMCI)

The Data Monitor and Control Interface, or DMCI, is a tool in MPLAB X IDE. The interface provides dynamic input control of application variables in projects. Application-generated data can be viewed graphically using any of 4 dynamically-assignable graph windows.

Debug/Debugger

See ICE/ICD.

Debugging Information

Compiler and assembler options that, when selected, provide varying degrees of information used to debug application code. See compiler or assembler documentation for details on selecting debug options.

Deprecated Features

Features that are still supported for legacy reasons, but will eventually be phased out and no longer used.

Device Programmer

A tool used to program electrically programmable semiconductor devices such as microcontrollers.

Digital Signal Controller

A A digital signal controller (DSC) is a microcontroller device with digital signal processing capability, i.e., Microchip dsPIC DSC devices.

Digital Signal Processing\Digital Signal Processor

Digital signal processing (DSP) is the computer manipulation of digital signals, commonly analog signals (sound or image) which have been converted to digital form (sampled). A digital signal processor is a microprocessor that is designed for use in digital signal processing.

Directives

Statements in source code that provide control of the language tool's operation.

Download

Download is the process of sending data from a host to another device, such as an emulator, programmer or target board.

DWARF

Debug With Arbitrary Record Format. DWARF is a debug information format for ELF files.

E

EEPROM

Electrically Erasable Programmable Read Only Memory. A special type of PROM that can be erased electrically. Data is written or erased one byte at a time. EEPROM retains its contents even when power is turned off.

ELF

Executable and Linking Format. An object file of this format contains machine code. Debugging and other information is specified in with DWARF. ELF/DWARF provide better debugging of optimized code than COFF.

Emulation/Emulator

See ICE/ICD.

Endianness

The ordering of bytes in a multi-byte object.

Environment

MPLAB PM3 **–** A folder containing files on how to program a device. This folder can be transferred to a SD/MMC card.

Epilogue

A portion of compiler-generated code that is responsible for deallocating stack space, restoring registers and performing any other machine-specific requirement specified in the runtime model. This code executes after any user code for a given function, immediately prior to the function return.

EPROM

Erasable Programmable Read Only Memory. A programmable read-only memory that can be erased usually by exposure to ultraviolet radiation.

Error/Error File

An error reports a problem that makes it impossible to continue processing your program. When possible, an error identifies the source file name and line number where the problem is apparent. An error file contains error messages and diagnostics generated by a language tool.

Event

A description of a bus cycle which may include address, data, pass count, external input, cycle type (fetch, R/W), and time stamp. Events are used to describe triggers, breakpoints and interrupts.

Executable Code

Software that is ready to be loaded for execution.

Export

Send data out of the MPLAB IDE/MPLAB X IDE in a standardized format.

Expressions

Combinations of constants and/or symbols separated by arithmetic or logical operators.

Extended Microcontroller Mode

In extended microcontroller mode, on-chip program memory as well as external memory is available. Execution automatically switches to external if the program memory address is greater than the internal memory space of the PIC18 device.

Extended Mode (PIC18 MCUs)

In Extended mode, the compiler will utilize the extended instructions (i.e., ADDFSR, ADDULNK, CALLW, MOVSF, MOVSS, PUSHL, SUBFSR and SUBULNK) and the indexed with literal offset addressing.

External Label

A label that has external linkage.

External Linkage

A function or variable has external linkage if it can be referenced from outside the module in which it is defined.

External Symbol

A symbol for an identifier which has external linkage. This may be a reference or a definition.

External Symbol Resolution

A process performed by the linker in which external symbol definitions from all input modules are collected in an attempt to resolve all external symbol references. Any external symbol references which do not have a corresponding definition cause a linker error to be reported.

External Input Line

An external input signal logic probe line (TRIGIN) for setting an event based upon external signals.

External RAM

Off-chip Read/Write memory.

F

Fatal Error

An error that will halt compilation immediately. No further messages will be produced.

File Registers

On-chip data memory, including General Purpose Registers (GPRs) and Special Function Registers (SFRs).

Filter

Determine by selection what data is included/excluded in a trace display or data file.

Fixup

The process of replacing object file symbolic references with absolute addresses after relocation by the linker.

Flash

A type of EEPROM where data is written or erased in blocks instead of bytes.

FNOP

Forced No Operation. A forced NOP cycle is the second cycle of a two-cycle instruction. Since the PIC microcontroller architecture is pipelined, it prefetches the next instruction in the physical address space while it is executing the current instruction. However, if the current instruction changes the program counter, this prefetched instruction is explicitly ignored, causing a forced NOP cycle.

Frame Pointer

A pointer that references the location on the stack that separates the stack-based arguments from the stack-based local variables. Provides a convenient base from which to access local variables and other values for the current function.

Free-Standing

An implementation that accepts any strictly conforming program that does not use complex types and in which the use of the features specified in the library clause (ANSI '89 standard clause 7) is confined to the contents of the standard headers `<float.h>`, `<iso646.h>`, `<limits.h>`, `<stdarg.h>`, `<stdbool.h>`, `<stddef.h>` and `<stdint.h>`.

G

GPR

General Purpose Register. The portion of device data memory (RAM) available for general use.

H

Halt

A stop of program execution. Executing Halt is the same as stopping at a breakpoint.

Heap

An area of memory used for dynamic memory allocation where blocks of memory are allocated and freed in an arbitrary order determined at runtime.

Hex Code\Hex File

Hex code is executable instructions stored in a hexadecimal format code. Hex code is contained in a hex file.

Hexadecimal

The base 16 numbering system that uses the digits 0-9 plus the letters A-F (or a-f). The digits A-F represent hexadecimal digits with values of (decimal) 10 to 15. The rightmost digit counts ones, the next counts multiples of 16, then 16^2 = 256, etc.

High Level Language

A language for writing programs that is further removed from the processor than assembly.

I

ICE/ICD

In-Circuit Emulator/In-Circuit Debugger: A hardware tool that debugs and programs a target device. An emulator has more features than an debugger, such as trace.

In-Circuit Emulation/In-Circuit Debug: The act of emulating or debugging with an in-circuit emulator or debugger.

-ICE/-ICD: A device (MCU or DSC) with on-board in-circuit emulation or debug circuitry. This device is always mounted on a header board and used to debug with an in-circuit emulator or debugger.

ICSP

In-Circuit Serial Programming. A method of programming Microchip embedded devices using serial communication and a minimum number of device pins.

IDE

Integrated Development Environment, as in MPLAB IDE/MPLAB X IDE.

Identifier

A function or variable name.

IEEE

Institute of Electrical and Electronics Engineers.

Import

Bring data into the MPLAB IDE/MPLAB X IDE from an outside source, such as from a hex file.

Initialized Data

Data which is defined with an initial value. In C,

```
int myVar=5;
```

defines a variable which will reside in an initialized data section.

Instruction Set

The collection of machine language instructions that a particular processor understands.

Instructions

A sequence of bits that tells a central processing unit to perform a particular operation and can contain data to be used in the operation.

Internal Linkage

A function or variable has internal linkage if it can not be accessed from outside the module in which it is defined.

International Organization for Standardization

An organization that sets standards in many businesses and technologies, including computing and communications. Also known as ISO.

Interrupt

A signal to the CPU that suspends the execution of a running application and transfers control to an Interrupt Service Routine (ISR) so that the event may be processed. Upon completion of the ISR, normal execution of the application resumes.

Interrupt Handler

A routine that processes special code when an interrupt occurs.

Interrupt Service Request (IRQ)

An event which causes the processor to temporarily suspend normal instruction execution and to start executing an interrupt handler routine. Some processors have several interrupt request events allowing different priority interrupts.

Interrupt Service Routine (ISR)

Language tools – A function that handles an interrupt.

MPLAB IDE/MPLAB X IDE – User-generated code that is entered when an interrupt occurs. The location of the code in program memory will usually depend on the type of interrupt that has occurred.

Interrupt Vector

Address of an interrupt service routine or interrupt handler.

L

L-value

An expression that refers to an object that can be examined and/or modified. An l-value expression is used on the left-hand side of an assignment.

Latency

The time between an event and its response.

Library/Librarian

See Archive/Archiver.

Linker

A language tool that combines object files and libraries to create executable code, resolving references from one module to another.

Linker Script Files

Linker script files are the command files of a linker. They define linker options and describe available memory on the target platform.

Listing Directives

Listing directives are those directives that control the assembler listing file format. They allow the specification of titles, pagination and other listing control.

Listing File

A listing file is an ASCII text file that shows the machine code generated for each C source statement, assembly instruction, assembler directive, or macro encountered in a source file.

Little Endian

A data ordering scheme for multibyte data whereby the least significant byte is stored at the lower addresses.

Local Label

A local label is one that is defined inside a macro with the LOCAL directive. These labels are particular to a given instance of a macro's instantiation. In other words, the symbols and labels that are declared as local are no longer accessible after the ENDM macro is encountered.

Logic Probes

Up to 14 logic probes can be connected to some Microchip emulators. The logic probes provide external trace inputs, trigger output signal, +5V, and a common ground.

Loop-Back Test Board

Used to test the functionality of the MPLAB REAL ICE in-circuit emulator.

LVDS

Low Voltage Differential Signaling. A low noise, low-power, low amplitude method for high-speed (gigabits per second) data transmission over copper wire.

With standard I/0 signaling, data storage is contingent upon the actual voltage level. Voltage level can be affected by wire length (longer wires increase resistance, which lowers voltage). But with LVDS, data storage is distinguished only by positive and negative voltage values, not the voltage level. Therefore, data can travel over greater lengths of wire while maintaining a clear and consistent data stream.

Source: http://www.webopedia.com/TERM/L/LVDS.html

M

Machine Code

The representation of a computer program that is actually read and interpreted by the processor. A program in binary machine code consists of a sequence of machine instructions (possibly interspersed with data). The collection of all possible instructions for a particular processor is known as its "instruction set".

Machine Language

A set of instructions for a specific central processing unit, designed to be usable by a processor without being translated.

Macro

Macro instruction. An instruction that represents a sequence of instructions in abbreviated form.

Macro Directives

Directives that control the execution and data allocation within macro body definitions.

Makefile

Export to a file the instructions to Make the project. Use this file to Make your project outside of MPLAB IDE/MPLAB X IDE, i.e., with a `make`.

Make Project

A command that rebuilds an application, recompiling only those source files that have changed since the last complete compilation.

MCU

Microcontroller Unit. An abbreviation for microcontroller. Also uC.

Memory Model

For C compilers, a representation of the memory available to the application. For the PIC18 C compiler, a description that specifies the size of pointers that point to program memory.

Message

Text displayed to alert you to potential problems in language tool operation. A message will not stop operation.

Microcontroller

A highly integrated chip that contains a CPU, RAM, program memory, I/O ports and timers.

Microcontroller Mode

One of the possible program memory configurations of PIC18 microcontrollers. In microcontroller mode, only internal execution is allowed. Thus, only the on-chip program memory is available in microcontroller mode.

Microprocessor Mode

One of the possible program memory configurations of PIC18 microcontrollers. In microprocessor mode, the on-chip program memory is not used. The entire program memory is mapped externally.

Mnemonics

Text instructions that can be translated directly into machine code. Also referred to as opcodes.

Module

The preprocessed output of a source file after preprocessor directives have been executed. Also known as a translation unit.

MPASM™ Assembler

Microchip Technology's relocatable macro assembler for PIC microcontroller devices, KeeLoq® devices and Microchip memory devices.

MPLAB *Language Tool* for *Device*

Microchip's C compilers, assemblers and linkers for specified devices. Select the type of language tool based on the device you will be using for your application, e.g., if you will be creating C code on a PIC18 MCU, select the MPLAB C Compiler for PIC18 MCUs.

MPLAB ICD

Microchip in-circuit debugger that works with MPLAB IDE/MPLAB X IDE. *See* ICE/ICD.

MPLAB IDE/MPLAB X IDE

Microchip's Integrated Development Environment. MPLAB IDE/MPLAB X IDE comes with an editor, project manager and simulator.

MPLAB PM3

A device programmer from Microchip. Programs PIC18 microcontrollers and dsPIC digital signal controllers. Can be used with MPLAB IDE/MPLAB X IDE or stand-alone. Replaces PRO MATE II.

MPLAB REAL ICE™ In-Circuit Emulator

Microchip's next-generation in-circuit emulator that works with MPLAB IDE/MPLAB X IDE. *See* ICE/ICD.

MPLAB SIM

Microchip's simulator that works with MPLAB IDE/MPLAB X IDE in support of PIC MCU and dsPIC DSC devices.

MPLIB™ Object Librarian

Microchip's librarian that can work with MPLAB IDE/MPLAB X IDE. MPLIB librarian is an object librarian for use with COFF object modules created using either MPASM assembler (mpasm or mpasmwin v2.0) or MPLAB C18 C Compiler.

MPLINK™ Object Linker

MPLINK linker is an object linker for the Microchip MPASM assembler and the Microchip C18 C compiler. MPLINK linker also may be used with the Microchip MPLIB librarian. MPLINK linker is designed to be used with MPLAB IDE/MPLAB X IDE, though it does not have to be.

MRU

Most Recently Used. Refers to files and windows available to be selected from MPLAB IDE/MPLAB X IDE main pull down menus.

N

Native Data Size

For Native trace, the size of the variable used in a Watch window must be of the same size as the selected device's data memory: bytes for PIC18 devices and words for 16-bit devices.

Nesting Depth

The maximum level to which macros can include other macros.

Node

MPLAB IDE/MPLAB X IDE project component.

Non-Extended Mode (PIC18 MCUs)

In Non-Extended mode, the compiler will not utilize the extended instructions nor the indexed with literal offset addressing.

Non Real Time

Refers to the processor at a breakpoint or executing single-step instructions or MPLAB IDE/MPLAB X IDE being run in simulator mode.

Non-Volatile Storage

A storage device whose contents are preserved when its power is off.

NOP

No Operation. An instruction that has no effect when executed except to advance the program counter.

O

Object Code/Object File

Object code is the machine code generated by an assembler or compiler. An object file is a file containing machine code and possibly debug information. It may be immediately executable or it may be relocatable, requiring linking with other object files, e.g., libraries, to produce a complete executable program.

Object File Directives

Directives that are used only when creating an object file.

Octal

The base 8 number system that only uses the digits 0-7. The rightmost digit counts ones, the next digit counts multiples of 8, then $8^2 = 64$, etc.

Off-Chip Memory

Off-chip memory refers to the memory selection option for the PIC18 device where memory may reside on the target board, or where all program memory may be supplied by the emulator. The **Memory** tab accessed from *Options>Development Mode* provides the Off-Chip Memory selection dialog box.

Opcodes

Operational Codes. *See* Mnemonics.

Operators

Symbols, like the plus sign '+' and the minus sign '-', that are used when forming well-defined expressions. Each operator has an assigned precedence that is used to determine order of evaluation.

OTP

One Time Programmable. EPROM devices that are not in windowed packages. Since EPROM needs ultraviolet light to erase its memory, only windowed devices are erasable.

P

Pass Counter

A counter that decrements each time an event (such as the execution of an instruction at a particular address) occurs. When the pass count value reaches zero, the event is satisfied. You can assign the Pass Counter to break and trace logic, and to any sequential event in the complex trigger dialog.

PC

Personal Computer or Program Counter.

PC Host

Any PC running a supported Windows operating system.

Persistent Data

Data that is never cleared or initialized. Its intended use is so that an application can preserve data across a device Reset.

Phantom Byte

An unimplemented byte in the dsPIC architecture that is used when treating the 24-bit instruction word as if it were a 32-bit instruction word. Phantom bytes appear in dsPIC hex files.

PIC MCUs

PIC microcontrollers (MCUs) refers to all Microchip microcontroller families.

PICkit 2 and 3

Microchip's developmental device programmers with debug capability through Debug Express. See the Readme files for each tool to see which devices are supported.

Plug-ins

The MPLAB IDE/MPLAB X IDE has both built-in components and plug-in modules to configure the system for a variety of software and hardware tools. Several plug-in tools may be found under the Tools menu.

Pod

The enclosure for an in-circuit emulator or debugger. Other names are "Puck", if the enclosure is round, and "Probe", not be confused with logic probes.

Power-on-Reset Emulation

A software randomization process that writes random values in data RAM areas to simulate uninitialized values in RAM upon initial power application.

Pragma

A directive that has meaning to a specific compiler. Often a pragma is used to convey implementation-defined information to the compiler. MPLAB C30 uses attributes to convey this information.

Precedence

Rules that define the order of evaluation in expressions.

Production Programmer

A production programmer is a programming tool that has resources designed in to program devices rapidly. It has the capability to program at various voltage levels and completely adheres to the programming specification. Programming a device as fast as possible is of prime importance in a production environment where time is of the essence as the application circuit moves through the assembly line.

Profile

For MPLAB SIM simulator, a summary listing of executed stimulus by register.

Program Counter

The location that contains the address of the instruction that is currently executing.

Program Counter Unit

16-bit assembler – A conceptual representation of the layout of program memory. The program counter increments by 2 for each instruction word. In an executable section, 2 program counter units are equivalent to 3 bytes. In a read-only section, 2 program counter units are equivalent to 2 bytes.

Program Memory

MPLAB IDE/MPLAB X IDE – The memory area in a device where instructions are stored. Also, the memory in the emulator or simulator containing the downloaded target application firmware.

16-bit assembler/compiler – The memory area in a device where instructions are stored.

Project

A project contains the files needed to build an application (source code, linker script files, etc.) along with their associations to various build tools and build options.

Prologue

A portion of compiler-generated code that is responsible for allocating stack space, preserving registers and performing any other machine-specific requirement specified in the runtime model. This code executes before any user code for a given function.

Prototype System

A term referring to a user's target application, or target board.

Psect

The OCG equivalent of a GCC section, short for program section. A block of code or data which is treated as a whole by the linker.

PWM Signals

Pulse Width Modulation Signals. Certain PIC MCU devices have a PWM peripheral.

Q

Qualifier

An address or an address range used by the Pass Counter or as an event before another operation in a complex trigger.

R

Radix

The number base, hex, or decimal, used in specifying an address.

RAM

Random Access Memory (Data Memory). Memory in which information can be accessed in any order.

Raw Data

The binary representation of code or data associated with a section.

Read Only Memory

Memory hardware that allows fast access to permanently stored data but prevents addition to or modification of the data.

Real Time

When an in-circuit emulator or debugger is released from the halt state, the processor runs in Real Time mode and behaves exactly as the normal chip would behave. In Real Time mode, the real time trace buffer of an emulator is enabled and constantly captures all selected cycles, and all break logic is enabled. In an in-circuit emulator or debugger, the processor executes in real time until a valid breakpoint causes a halt, or until the user halts the execution.

In the simulator, real time simply means execution of the microcontroller instructions as fast as they can be simulated by the host CPU.

Recursive Calls

A function that calls itself, either directly or indirectly.

Recursion

The concept that a function or macro, having been defined, can call itself. Great care should be taken when writing recursive macros; it is easy to get caught in an infinite loop where there will be no exit from the recursion.

Reentrant

A function that may have multiple, simultaneously active instances. This may happen due to either direct or indirect recursion or through execution during interrupt processing.

Relaxation

The process of converting an instruction to an identical, but smaller instruction. This is useful for saving on code size. MPLAB XC16 currently knows how to `relax` a `CALL` instruction into an `RCALL` instruction. This is done when the symbol that is being called is within +/- 32k instruction words from the current instruction.

Relocatable

An object whose address has not been assigned to a fixed location in memory.

Relocatable Section

16-bit assembler – A section whose address is not fixed (absolute). The linker assigns addresses to relocatable sections through a process called relocation.

Relocation

A process performed by the linker in which absolute addresses are assigned to relocatable sections and all symbols in the relocatable sections are updated to their new addresses.

ROM

Read Only Memory (Program Memory). Memory that cannot be modified.

Run

The command that releases the emulator from halt, allowing it to run the application code and change or respond to I/O in real time.

Run-time Model

Describes the use of target architecture resources.

Runtime Watch

A Watch window where the variables change in as the application is run. See individual tool documentation to determine how to set up a runtime watch. Not all tools support runtime watches.

S

Scenario

For MPLAB SIM simulator, a particular setup for stimulus control.

Section

The GCC equivalent of an OCG psect. A block of code or data which is treated as a whole by the linker.

Section Attribute

A GCC characteristic ascribed to a section (e.g., an `access` section).

Sequenced Breakpoints

Breakpoints that occur in a sequence. Sequence execution of breakpoints is bottom-up; the last breakpoint in the sequence occurs first.

Serialized Quick Turn Programming

Serialization allows you to program a serial number into each microcontroller device that the Device Programmer programs. This number can be used as an entry code, password or ID number.

Shell

The MPASM assembler shell is a prompted input interface to the macro assembler. There are two MPASM assembler shells: one for the DOS version and one for the Windows operating system version.

Simulator

A software program that models the operation of devices.

Single Step

This command steps though code, one instruction at a time. After each instruction, MPLAB IDE/MPLAB X IDE updates register windows, watch variables, and status displays so you can analyze and debug instruction execution. You can also single step C compiler source code, but instead of executing single instructions, MPLAB IDE/MPLAB X IDE will execute all assembly level instructions generated by the line of the high level C statement.

Skew

The information associated with the execution of an instruction appears on the processor bus at different times. For example, the executed opcodes appears on the bus as a fetch during the execution of the previous instruction, the source data address and value and the destination data address appear when the opcodes is actually executed, and the destination data value appears when the next instruction is executed. The trace buffer captures the information that is on the bus at one instance. Therefore, one trace buffer entry will contain execution information for three instructions. The number of captured cycles from one piece of information to another for a single instruction execution is referred to as the skew.

Skid

When a hardware breakpoint is used to halt the processor, one or more additional instructions may be executed before the processor halts. The number of extra instructions executed after the intended breakpoint is referred to as the skid.

Source Code

The form in which a computer program is written by the programmer. Source code is written in a formal programming language which can be translated into machine code or executed by an interpreter.

Source File

An ASCII text file containing source code.

Special Function Registers (SFRs)

The portion of data memory (RAM) dedicated to registers that control I/O processor functions, I/O status, timers or other modes or peripherals.

SQTP

See Serialized Quick Turn Programming.

Stack, Hardware

Locations in PIC microcontroller where the return address is stored when a function call is made.

Stack, Software

Memory used by an application for storing return addresses, function parameters, and local variables. This memory is dynamically allocated at runtime by instructions in the program. It allows for reentrant function calls.

Stack, Compiled

A region of memory managed and allocated by the compiler in which variables are statically assigned space. It replaces a software stack when such mechanisms cannot be efficiently implemented on the target device. It precludes reentrancy.

MPLAB Starter Kit for *Device*

Microchip's starter kits contains everything needed to begin exploring the specified device. View a working application and then debug and program you own changes.

Static RAM or SRAM

Static Random Access Memory. Program memory you can read/write on the target board that does not need refreshing frequently.

Status Bar

The Status Bar is located on the bottom of the MPLAB IDE/MPLAB X IDE window and indicates such current information as cursor position, development mode and device, and active tool bar.

Step Into

This command is the same as Single Step. Step Into (as opposed to Step Over) follows a CALL instruction into a subroutine.

Step Over

Step Over allows you to debug code without stepping into subroutines. When stepping over a `CALL` instruction, the next breakpoint will be set at the instruction after the `CALL`. If for some reason the subroutine gets into an endless loop or does not return properly, the next breakpoint will never be reached. The Step Over command is the same as Single Step except for its handling of `CALL` instructions.

Step Out

Step Out allows you to step out of a subroutine which you are currently stepping through. This command executes the rest of the code in the subroutine and then stops execution at the return address to the subroutine.

Stimulus

Input to the simulator, i.e., data generated to exercise the response of simulation to external signals. Often the data is put into the form of a list of actions in a text file. Stimulus may be asynchronous, synchronous (pin), clocked and register.

Stopwatch

A counter for measuring execution cycles.

Storage Class

Determines the lifetime of the memory associated with the identified object.

Storage Qualifier

Indicates special properties of the objects being declared (e.g., `const`).

Symbol

A symbol is a general purpose mechanism for describing the various pieces which comprise a program. These pieces include function names, variable names, section names, file names, struct/enum/union tag names, etc. Symbols in MPLAB IDE/MPLAB X IDE refer mainly to variable names, function names and assembly labels. The value of a symbol after linking is its value in memory.

Symbol, Absolute

Represents an immediate value such as a definition through the assembly `.equ` directive.

System Window Control

The system window control is located in the upper left corner of windows and some dialogs. Clicking on this control usually pops up a menu that has the items “Minimize,” “Maximize,” and “Close.”

T

Target

Refers to user hardware.

Target Application

Software residing on the target board.

Target Board

The circuitry and programmable device that makes up the target application.

Target Processor

The microcontroller device on the target application board.

Template

Lines of text that you build for inserting into your files at a later time. The MPLAB Editor stores templates in template files.

Tool Bar

A row or column of icons that you can click on to execute MPLAB IDE/MPLAB X IDE functions.

Trace

An emulator or simulator function that logs program execution. The emulator logs program execution into its trace buffer which is uploaded to MPLAB IDE/MPLAB X IDE's trace window.

Trace Memory

Trace memory contained within the emulator. Trace memory is sometimes called the trace buffer.

Trace Macro

A macro that will provide trace information from emulator data. Since this is a software trace, the macro must be added to code, the code must be recompiled or reassembled, and the target device must be programmed with this code before trace will work.

Trigger Output

Trigger output refers to an emulator output signal that can be generated at any address or address range, and is independent of the trace and breakpoint settings. Any number of trigger output points can be set.

Trigraphs

Three-character sequences, all starting with ??, that are defined by ISO C as replacements for single characters.

U

Unassigned Section

A section which has not been assigned to a specific target memory block in the linker command file. The linker must find a target memory block in which to allocate an unassigned section.

Uninitialized Data

Data which is defined without an initial value. In C,

```
int myVar;
```

defines a variable which will reside in an uninitialized data section.

Upload

The Upload function transfers data from a tool, such as an emulator or programmer, to the host PC or from the target board to the emulator.

USB

Universal Serial Bus. An external peripheral interface standard for communication between a computer and external peripherals over a cable using bi-serial transmission. USB 1.0/1.1 supports data transfer rates of 12 Mbps. Also referred to as high-speed USB, USB 2.0 supports data rates up to 480 Mbps.

V

Vector

The memory locations that an application will jump to when either a Reset or interrupt occurs.

Volatile

A variable qualifier which prevents the compiler applying optimizations that affect how the variable is accessed in memory.

W

Warning

MPLAB IDE/MPLAB X IDE – An alert that is provided to warn you of a situation that would cause physical damage to a device, software file, or equipment.

16-bit assembler/compiler – Warnings report conditions that may indicate a problem, but do not halt processing. In MPLAB C30, warning messages report the source file name and line number, but include the text '`warning`:' to distinguish them from error messages.

Watch Variable

A variable that you may monitor during a debugging session in a Watch window.

Watch Window

Watch windows contain a list of watch variables that are updated at each breakpoint.

Watchdog Timer (WDT)

A timer on a PIC microcontroller that resets the processor after a selectable length of time. The WDT is enabled or disabled and set up using Configuration bits.

Workbook

For MPLAB SIM stimulator, a setup for generation of SCL stimulus.

NOTES:

Index

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110

Canada - Toronto
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Düsseldorf
Tel: 49-2129-3766400

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Pforzheim
Tel: 49-7231-424750

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Venice
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Poland - Warsaw
Tel: 48-22-3325737

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

10/28/13

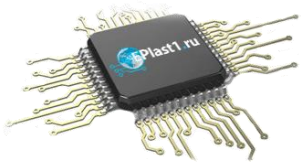
Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits,General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.



Как с нами связаться

Телефон: 8 (812) 309 58 32 (многоканальный)
Факс: 8 (812) 320-02-42
Электронная почта: org@eplast1.ru
Адрес: 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.