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

ISBN: 978-1-61341-883-3

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2009

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

Chapter 4. Basic Tasks

Chapter 5. Additional Tasks

Chapter 6. Advanced Tasks

Chapter 7. Troubleshooting

Chapter 8. MPLAB X IDE vs. MPLAB IDE v8

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 manual layout is as follows:

- **Chapter 1. "What is MPLAB® X IDE?"** – An overview of what MPLAB X IDE is and where help can be found.
- **Chapter 2. "Before You Begin"** – Describes how to install USB drivers for 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"** – Provides descriptions of the basic features of MPLAB X IDE. This is similar to the Tutorial chapter but with more detail.
- **Chapter 5. "Additional Tasks"** – Provides descriptions of additional features of MPLAB X IDE, such as importing MPLAB IDE v8 projects or using the stopwatch.
- **Chapter 6. "Advanced Tasks"** – Provides descriptions of advanced features of MPLAB X IDE, such as working with multiple projects and project configurations.
- **Chapter 7. "Troubleshooting"** – Discusses troubleshooting techniques.
- **Chapter 8. "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 9. "Desktop Reference"** – Provides a reference to MPLAB X IDE desktop items including menus, toolbars and the status bar.
- **Chapter 10. "Windows and Dialogs"** – References NetBeans™ windows and dialogs and discusses MPLAB X IDE specific windows and dialogs.
- **Chapter 11. "Project Files and Folders"** – Explains the folder structure and locations of project files.
- **Chapter 12. "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 "`Readme for MPLAB IDE.htm`" file (an HTML file) in the Readmes subdirectory of the MPLAB X IDE installation directory. The Readme file contains 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 for complete and updated versions of device (PIC® MCU and dsPIC® DSC) data sheets and related device family reference manuals.

NOTES:

Chapter 1. What is MPLAB® X IDE?

1.1 AN OVERVIEW OF EMBEDDED SYSTEMS

MPLAB X IDE is a software program used to develop applications for Microchip microcontrollers and digital signal controllers. It is called an Integrated Development Environment, or IDE, because it provides a single integrated "environment" to develop code for embedded microcontrollers. Experienced embedded systems designers may want to skip ahead to the next chapter. The rest of this chapter briefly explains embedded systems development and how MPLAB X IDE is used.

1.1.1 Description of an "Embedded System"

An embedded system is typically a design making use of the power of a small microcontroller, like the Microchip PIC MCU or dsPIC Digital Signal Controller (DSC). These microcontrollers combine a microprocessor unit (like the CPU in a desktop PC) 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.

1.1.2 Differences Between an Embedded Controller and a PC

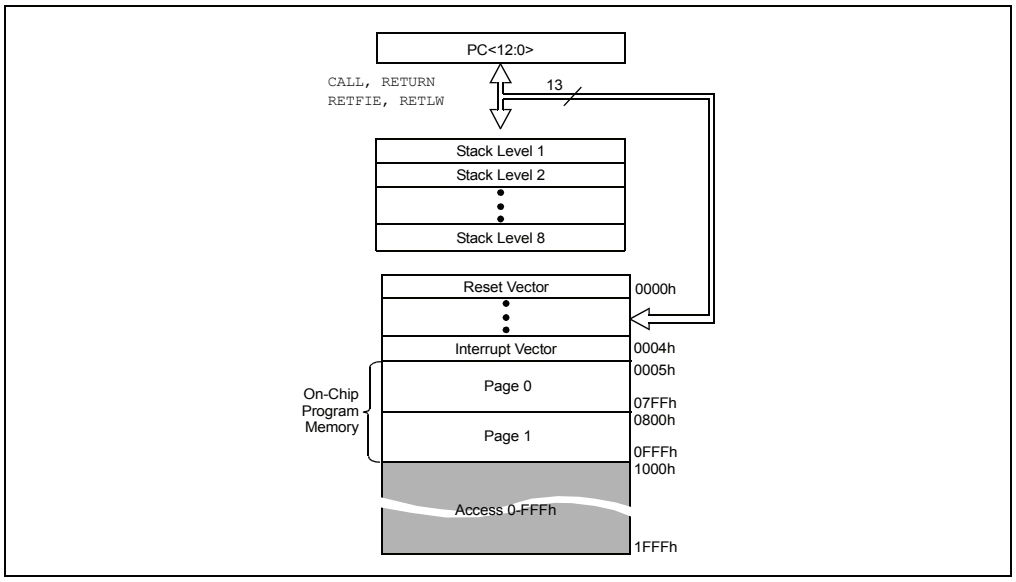
The main difference between an embedded controller and a PC is that the embedded controller is dedicated to one specific task or set of tasks. A PC is designed to run many different types of programs and to connect to many different external devices. An embedded controller has a single program and, as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

1.1.3 Components of a Microcontroller

The PIC MCU has on-chip program memory for the firmware, or coded instructions, to run a program (Figure 1-1). 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



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 1-3. 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 1-2: PIC® MCU DATA SHEET – FILE REGISTERS

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

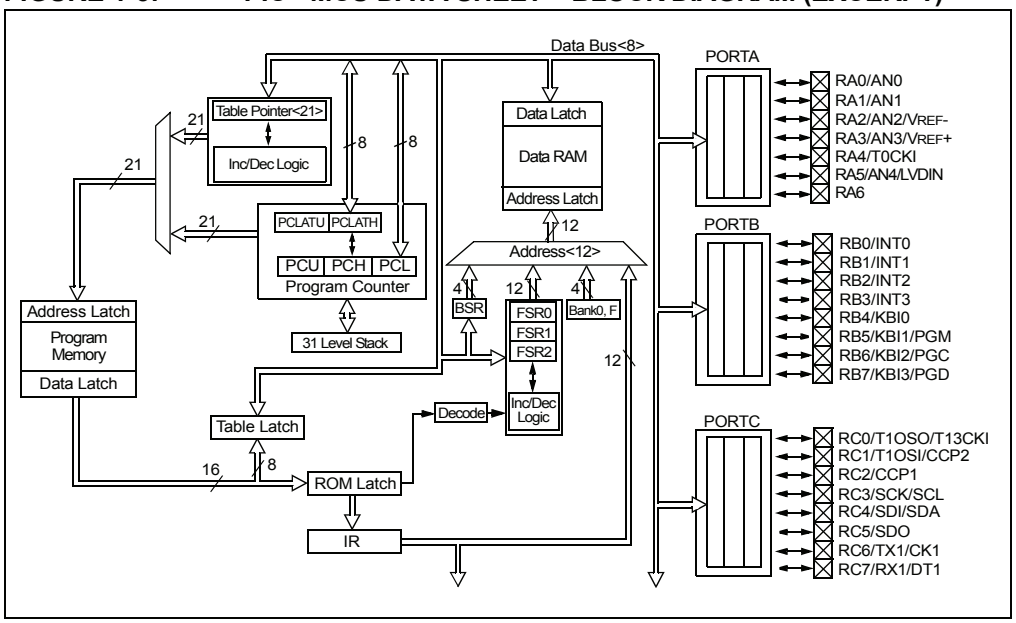
Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

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. 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.

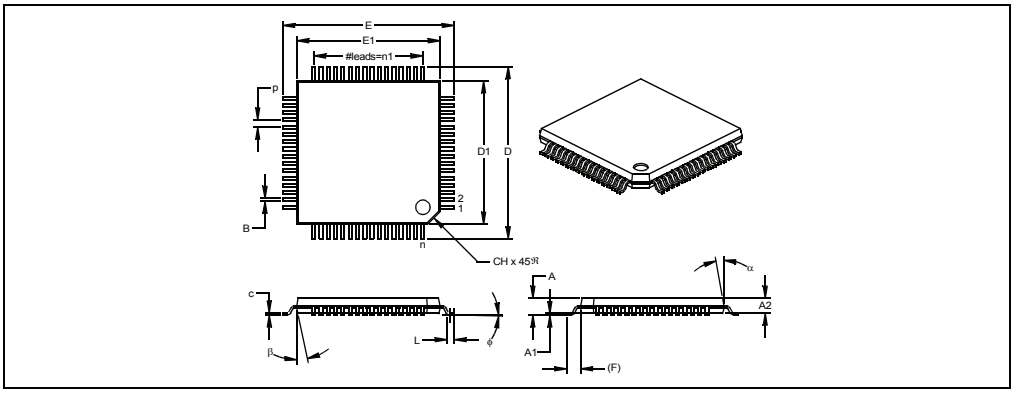
FIGURE 1-3: PIC® MCU DATA SHEET – BLOCK DIAGRAM (EXCERPT)



In order to design such a system, it must be decided which peripherals are needed for an application. Analog-to-Digital Converters (ADCs) allow microcontrollers to connect to sensors and receive changing voltage levels. Serial communication peripherals allow you to stream 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 if the external power is dipping below dangerous levels so 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 determines which PIC MCU to use. Other factors might 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 1-4: EXAMPLE PIC® MCU DEVICE PACKAGE



1.1.4 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.

FIGURE 1-5: PIC® MCU DATA SHEET – TIMING (EXCERPT)

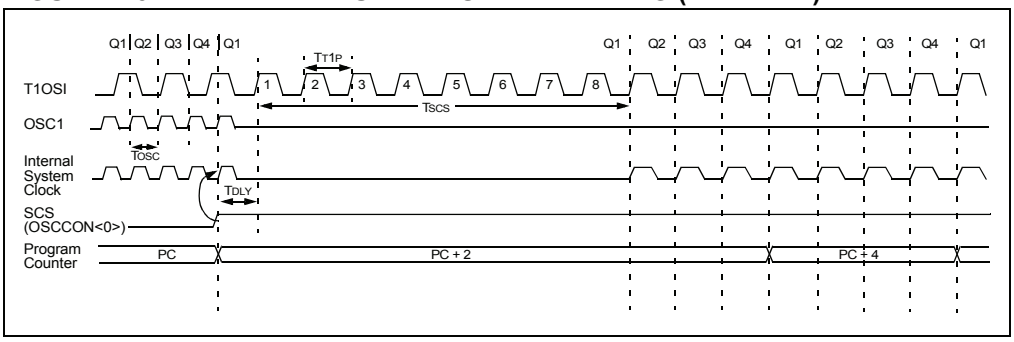
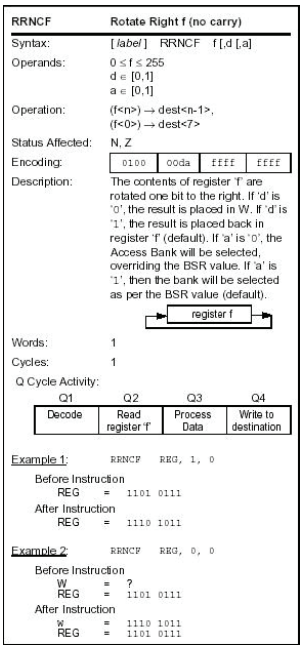


FIGURE 1-6: PIC® MCU DATA SHEET – INSTRUCTIONS (EXCERPT)



RRNCF — **Rotate Right f (no carry)**

Syntax: [*label*] RRNCF f [,d [,a]

Operands: $0 \le f \le 255$, $d \in [0,1]$, $a \in [0,1]$

Operation: (f<n>) → dest<n-1>, (f<0>) → dest<7>

Status Affected: N, Z

Encoding:

0100	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

register f

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: RRNCF REG, 1, 0

Before Instruction
REG = 1101 0111
After Instruction
REG = 1110 1011

Example 2: RRNCF REG, 0, 0

Before Instruction
W = ?
REG = 1101 0111
After Instruction
W = 1110 1011
REG = 1101 0111

2. Compile, assemble and link the software using the assembler and/or compiler and linker to convert your code into "ones and zeroes" – 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 zeroes" 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.

Once 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.

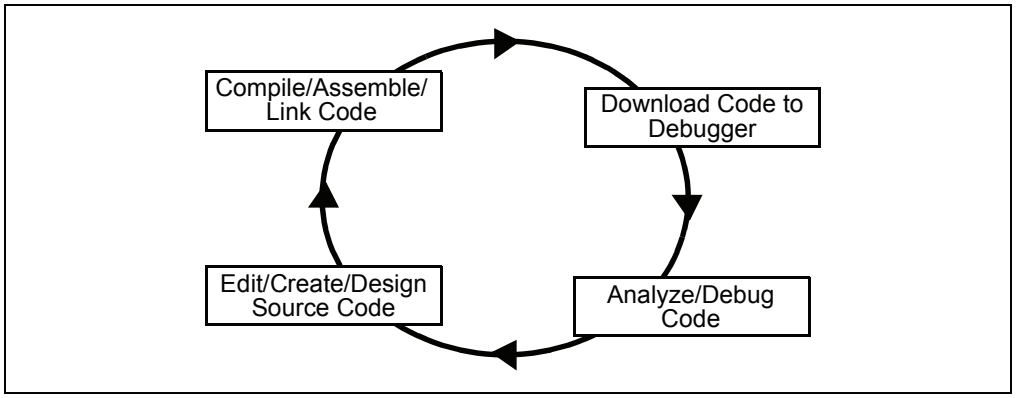
Once the hardware is in a prototype stage, a hardware debugger, such as an in-circuit emulator or an in-circuit debugger, can be used. These debug tools run the code in real time on your actual application by using special circuitry built into many devices with Flash program memory. They can "see into" the target microcontrollers 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 device 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.

1.2 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

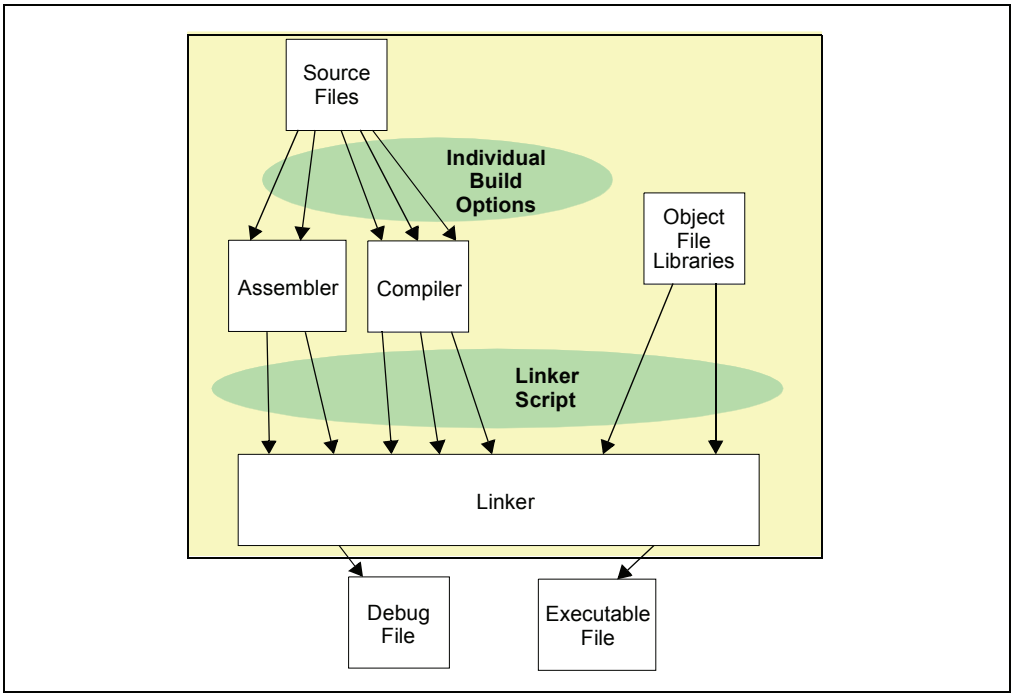


MPLAB X IDE is a "wrapper" that coordinates all the tools from a single graphical user interface, usually automatically. For instance, once code is written, it can be converted to executable instructions and downloaded into a microcontroller to see how it works. In this process multiple tools are needed: 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.

1.3 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.

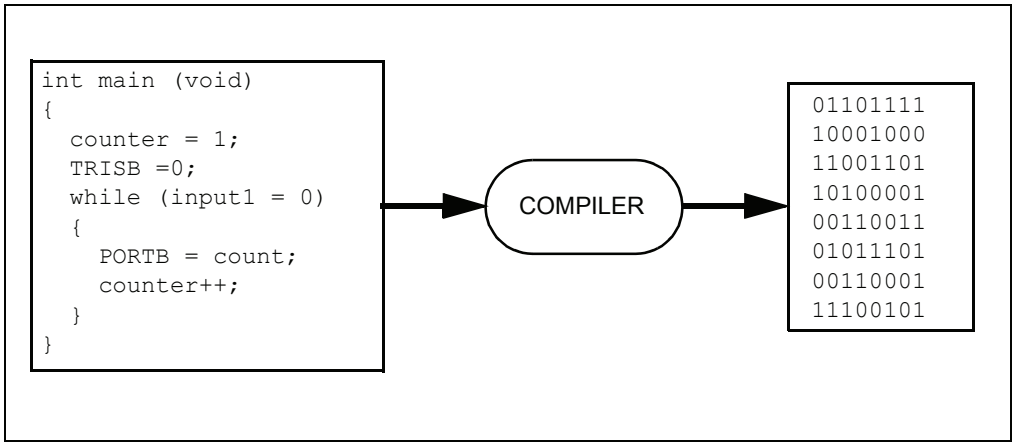
1.4 LANGUAGE TOOLS

Language tools are programs such as cross-assemblers and cross-compilers. Most people are familiar with language tools that run on a computer such as 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 produce code to run on another microprocessor (or microcontroller).

The 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 Watch windows to view variable contents, and lets you single step through the source code, 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 allows the smallest possible memory 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



1.5 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.

1.5.1 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.

1.5.2 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.

1.5.3 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.

1.6 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.

1.7 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 Project 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 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 PM3 programmer, PICkit™ 2, PICkit 3 and MPLAB ICD 3 in-circuit debuggers; and MPLAB REAL ICE™ in-circuit emulator can program code into target devices. MPLAB X IDE offers full control over programming 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-Circuit Debuggers and Emulators**

PICkit™ 2, 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, they 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.

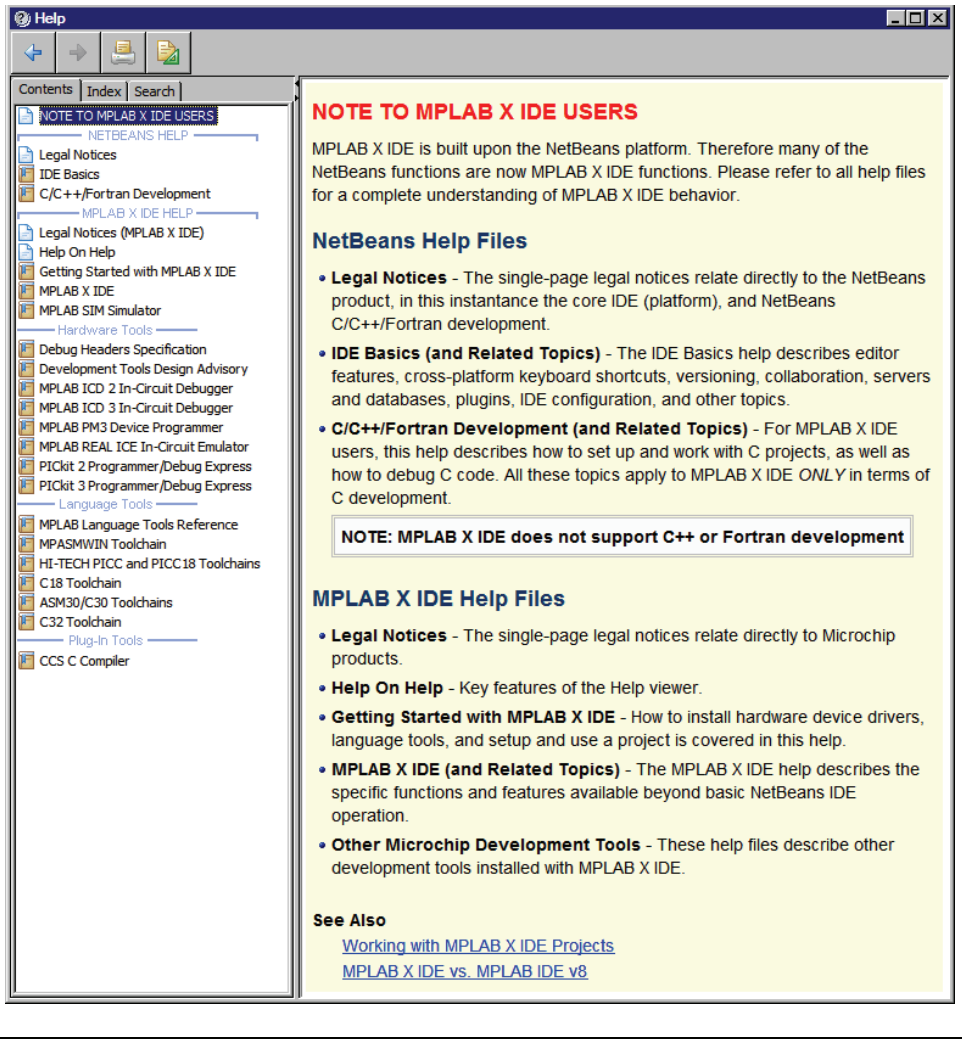
1.8 MPLAB X IDE ONLINE HELP

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 (Figure 1-10). 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.

For a comparison of MPLAB X IDE and MPLAB IDE v8, see **Chapter 8. “MPLAB X IDE vs. MPLAB IDE v8”**.

FIGURE 1-10: CURRENT MPLAB X IDE HELP (INCLUDING NetBeans TOPICS)



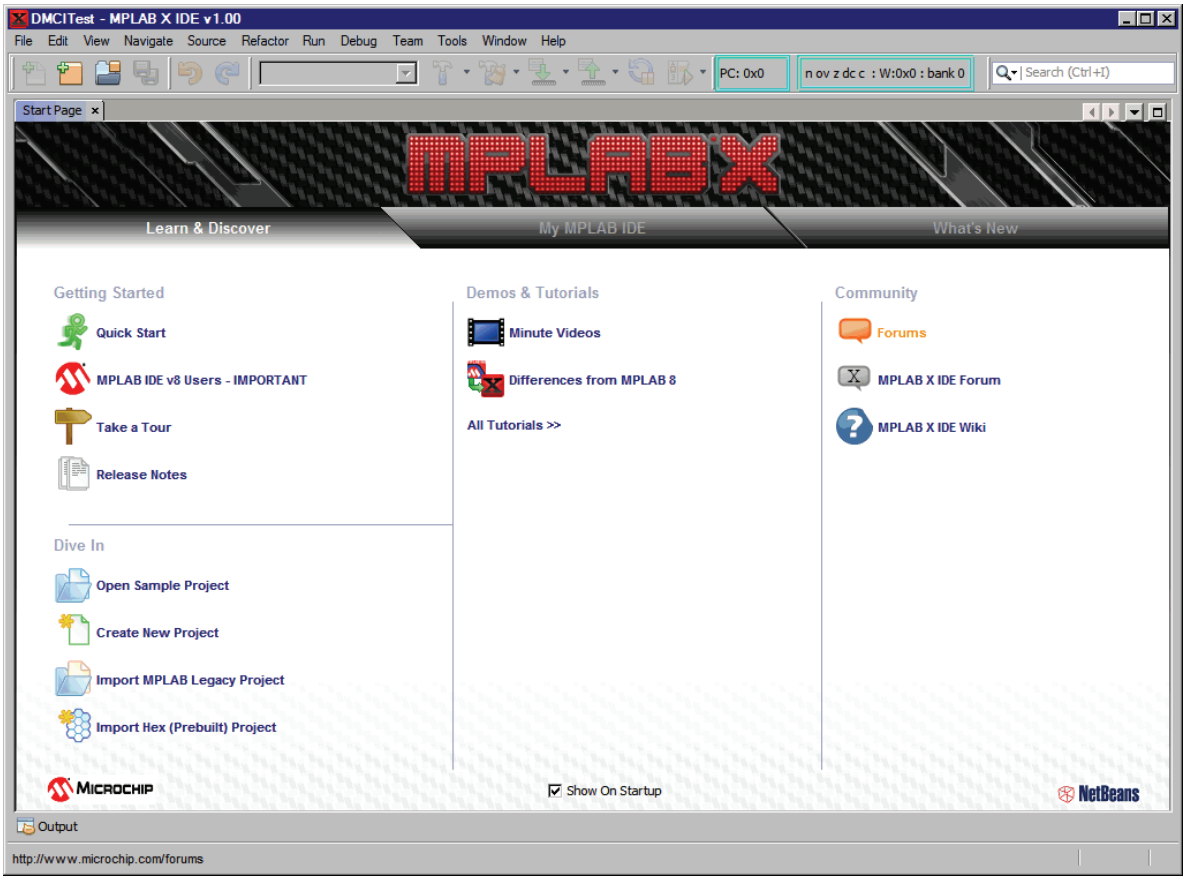
1.9 OTHER MPLAB X IDE DOCUMENTATION

In addition to help, links to other documentation, as well as videos and forums, may be found on the Start Page (Figure 1-11).

Also, some MPLAB IDE v8 documentation may be useful; for example, hardware tool Release Notes that document known hardware issues. See the Microchip web site for these documents:

http://www.microchip.com/mplab

FIGURE 1-11: MPLAB X IDE START PAGE



1.10 WEB SITE

Microchip provides online support via our web site at http://www.microchip.com. This web site is used as a means to make files and information easily available to customers. For more details, see **Support**.

1.11 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**.

NOTES:

Chapter 2. Before You Begin

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

2.1 INSTALL JRE AND MPLAB X IDE

If you are reading this help, you have already acquired and installed the Java Runtime Environment (JRE) and MPLAB X IDE (based on the NetBeans platform.) At this time:

1. Ensure you have the correct JRE for your version of MPLAB X IDE by consulting the "Readme for MPLAB X IDE" accessed from the Release Notes on the Start page.
2. Read the following sections and print them out if you need to perform the actions specified in them. For information on printing out topics, see "Help On Help".
3. Close MPLAB X IDE until the required steps are performed and then re-launch.

2.2 INSTALL THE USB DEVICE DRIVERS (FOR HARDWARE TOOLS)

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

2.2.1 USB Driver Installation for Mac or Linux OSs

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.

2.2.2 USB Driver Installation for Windows® 2000/XP/Vista/7 OSs

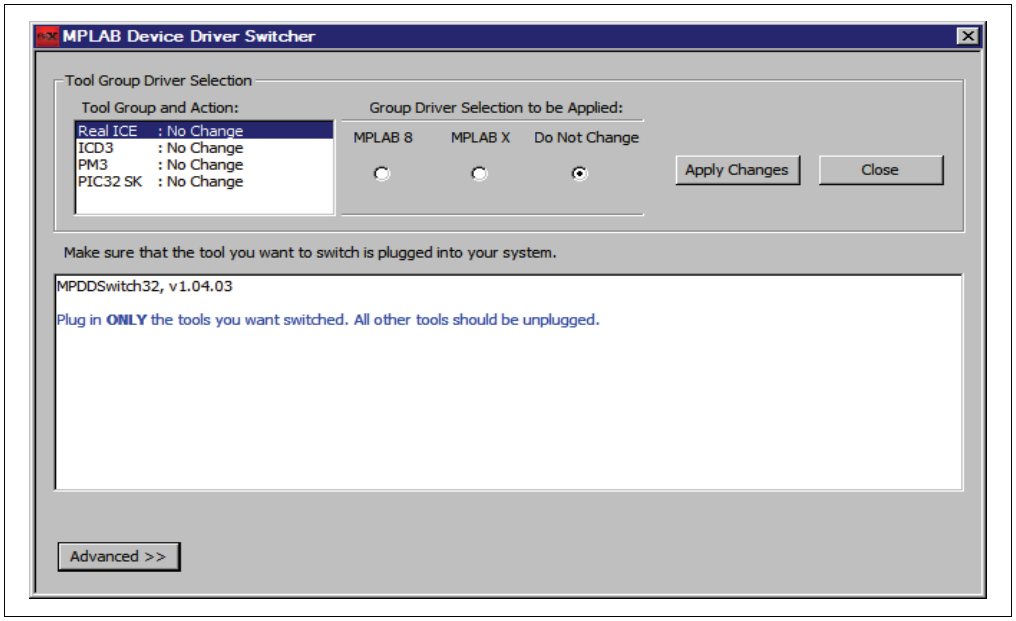
When you install MPLAB X IDE on a Windows PC, you will need to follow the instructions below to correctly install the USB drivers. These instructions apply to the following tools:

- MPLAB REAL ICE in-circuit emulator
- MPLAB ICD 3 in-circuit debugger
- MPLAB PM3 device programmer
- PIC32 Starter Kit (for 32-bit operating systems only prior to Beta 7.10)

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

To install and update USB drivers, you can use a GUI application called MPDDSwitch. This should be available as the desktop icon MPLAB Device Driver Switcher.

FIGURE 2-1: SWITCHER UTILITY



If this does not work, you can use the batch file in the same directory to install the drivers. Follow the instructions below to determine your installation method.

2.2.2.1 ADMINISTRATOR MODE (WINDOWS 7 OS)

In the sections below, you will be asked to use a Switcher executable. To run this on Windows 7, you must be in Administrator mode.

To run the Device Driver Switcher GUI application as administrator, right click on the executable – `MPDDSwitch32.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.

Also, to switch drivers using the helper batch files, open the command prompt in Admin mode and then run the batch file with the appropriate syntax.

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

4. Plug your desired tool into a PC USB connector.
5. Open the PC's Device Manager window. For example, on a Windows XP PC, 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.
6. Expand the "Microchip Tools" section to view the current driver for your tool. The name should be of the form: Microchip *Tool Name*.
7. 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`

8. Under the `Switcher` folder, go to the folder for your operating system - `32Bit` or `64Bit`.
9. Launch the `MPDDSwitch.exe` file.
10. 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.
11. 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.
12. If the GUI fails to install the drivers, follow the directions in the `ReadMe32.txt` or `ReadMe64.txt` file to execute a batch file to install your drivers.
13. 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.

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

If v8 is not installed when you plug in the tool, the "New Hardware Found" wizard will come up.

2.2.2.3.1 MPLAB X IDE Automatic Install (Recommended)

The drivers have been pre-installed during the IDE installation. Use the wizard to install them automatically. Otherwise, you can install them manually as per the next section.

2.2.2.3.2 MPLAB X IDE Manual Install

To prevent the installation of incorrect drivers, follow the wizard to manually install the drivers. When asked to show the location of the drivers, you will need to point to the INF file in the `32Bit` or `64Bit` folder:

```
C:\Program Files\Microchip\MPLAB X IDE\Switcher\32Bit\winusb\x86\
  MCHPWinUSBDevice.inf
```

or

```
C:\Program Files\Microchip\MPLAB X IDE\Switcher\64Bit\winusb\amd64\
  MCHPWinUSBDevice.inf
```

2.2.2.4 TOOL COMMUNICATION ISSUES

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.

If you are having problems communicating with your tool, you may have an older version of `WinUSB` on your system. Go to your system folder, either `C:\Windows\system32` for 32-bit OSs or `C:\Windows\SysWOW64` for 64-bit OSs. Find `WinUSB` and rename it. Then install the drivers again.

2.3 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 setup information.

2.4 INSTALL THE LANGUAGE TOOLS

When you install MPLAB X IDE, the following language tools are installed as well: MPASM toolchain and ASM30 toolchain.

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

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

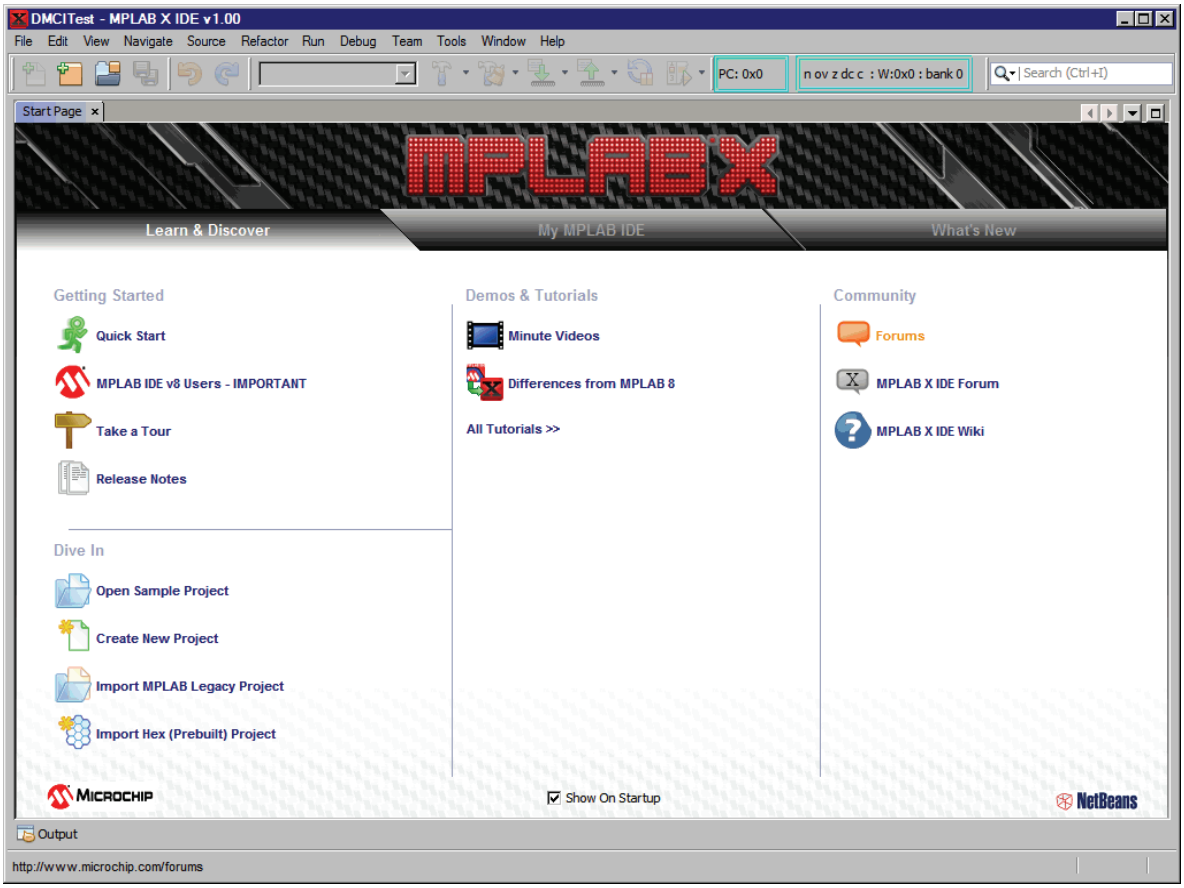
To install the compiler you want, download it and follow the instructions during the download or within the zip file once downloaded.

2.5 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 USBWin 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”.

2.6 LAUNCH MULTIPLE 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.

2.6.1 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**.

2.6.2 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.

2.6.3 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”
```

Chapter 3. Tutorial

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

Setup the Hardware and Software

- Tutorial Equipment
- Installation and Setup

Create and Setup 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

Run and Debug 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

3.1 TUTORIAL EQUIPMENT

The products used in this tutorial are:

Tool	Web Page	Order Number
MPLAB® X IDE	http://www.microchip.com/mplabx	N/A
MPLAB® C Compiler for PIC32 MCUs*	http://www.microchip.com/c32	SW006015 (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
* The evaluation version of this compiler comes free with MPLAB X IDE and was used for this tutorial. However, you may purchase the standard version.		

3.2 INSTALLATION AND SETUP

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 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:

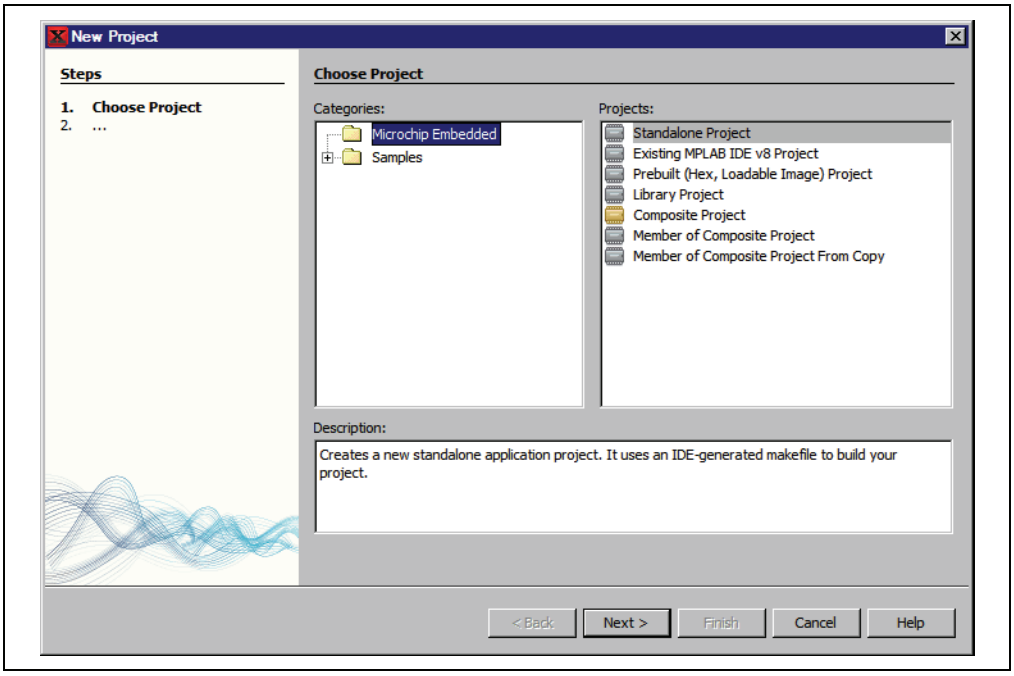
- **Start Page**, "**Learn & Discover**" tab, "Dive In", "Create New Project"
- *File>New Project* (or Ctrl+Shift+N)

The New Project Wizard with launch to guide you through new project setup.

Step 1 will first ask you to choose a project category. This is a NetBeans dialog, so to work with Microchip products you must choose "Microchip Embedded". Secondly, you will choose a project type. For this tutorial choose "Stand-alone Project".

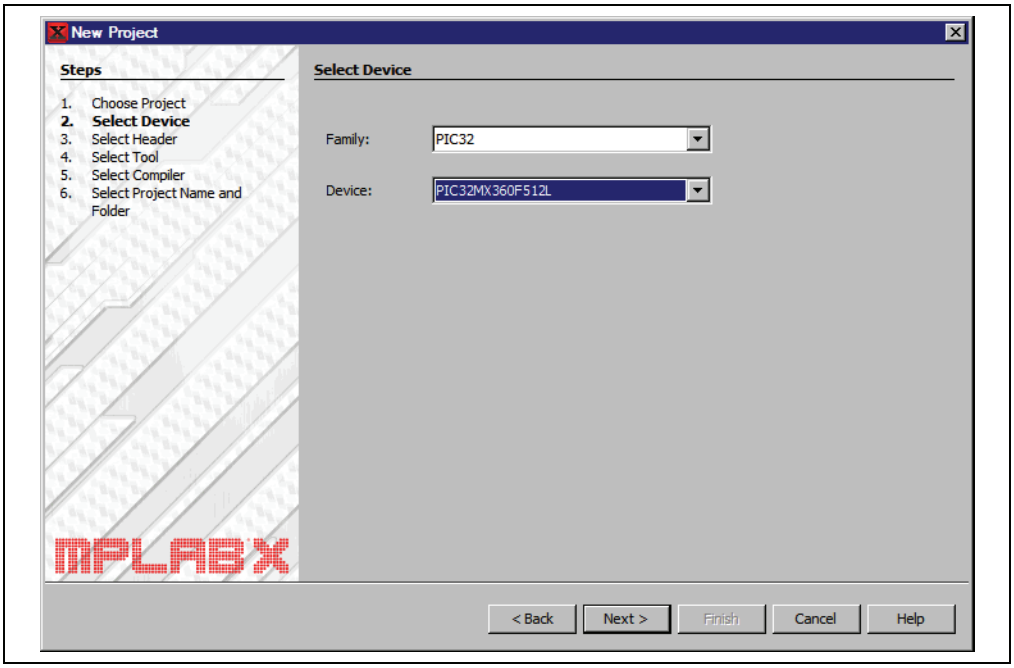
Click **Next>** to move to the next dialog.

FIGURE 3-1: PROJECT WIZARD – CHOOSE PROJECT



Step 2 is an MPLAB X IDE dialog and so will look different from the Step 1 dialog. Choose your device here – PIC32MX360F512L – and then click **Next>**.

FIGURE 3-2: PROJECT WIZARD – SELECT DEVICE



Step 3 will appear 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 involves selecting the tool. Tool support for the selected device is signified by the colored circles in front of the tool name, just as MPLAB IDE v8 had lights on the Device Selection dialog. The circle colors mean the following:

- Green – full support
- Yellow – beta support
- Red – no support yet

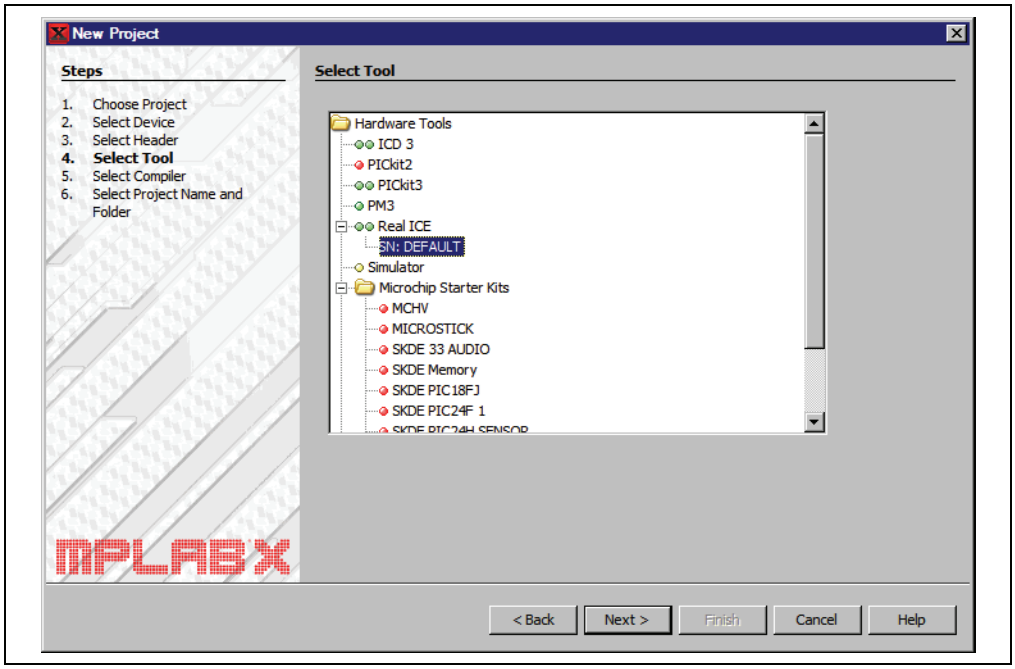
If you cannot see the colors, mouse over a circle to pop up text about support.

The first colored circle shows debugger support and the second shows programmer 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

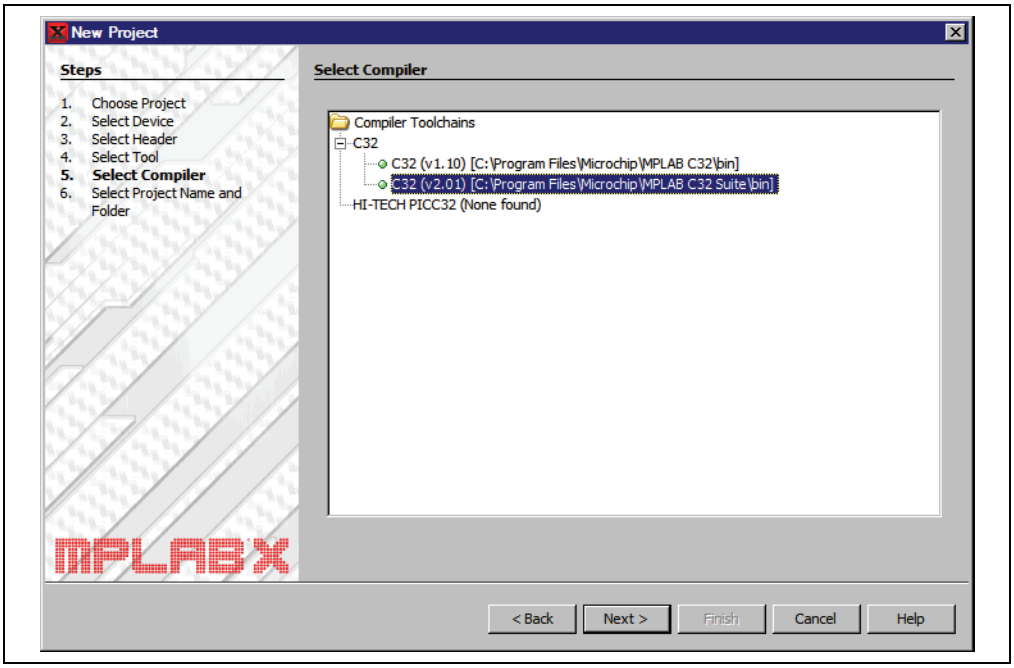


Step 5 involves selecting the language tool, either a C compiler or assembler. Again, the colored circle in front of the compiler name signifies the device support level. Mouse over for text.

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 3-4: PROJECT WIZARD – SELECT LANGUAGE TOOL



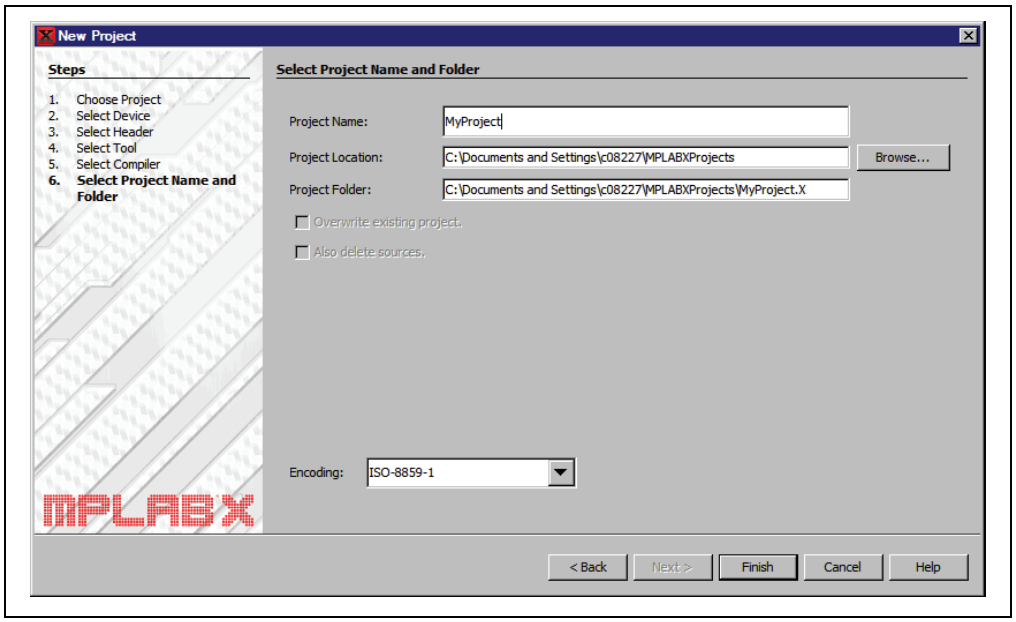
Step 6 involves selecting the project name and location. Enter the project name `MyXProject`. By default, projects will be placed in:

- Windows XP – `C:\Documents and Settings\UserName\MPLABXProject`
- Windows 7 – `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.

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

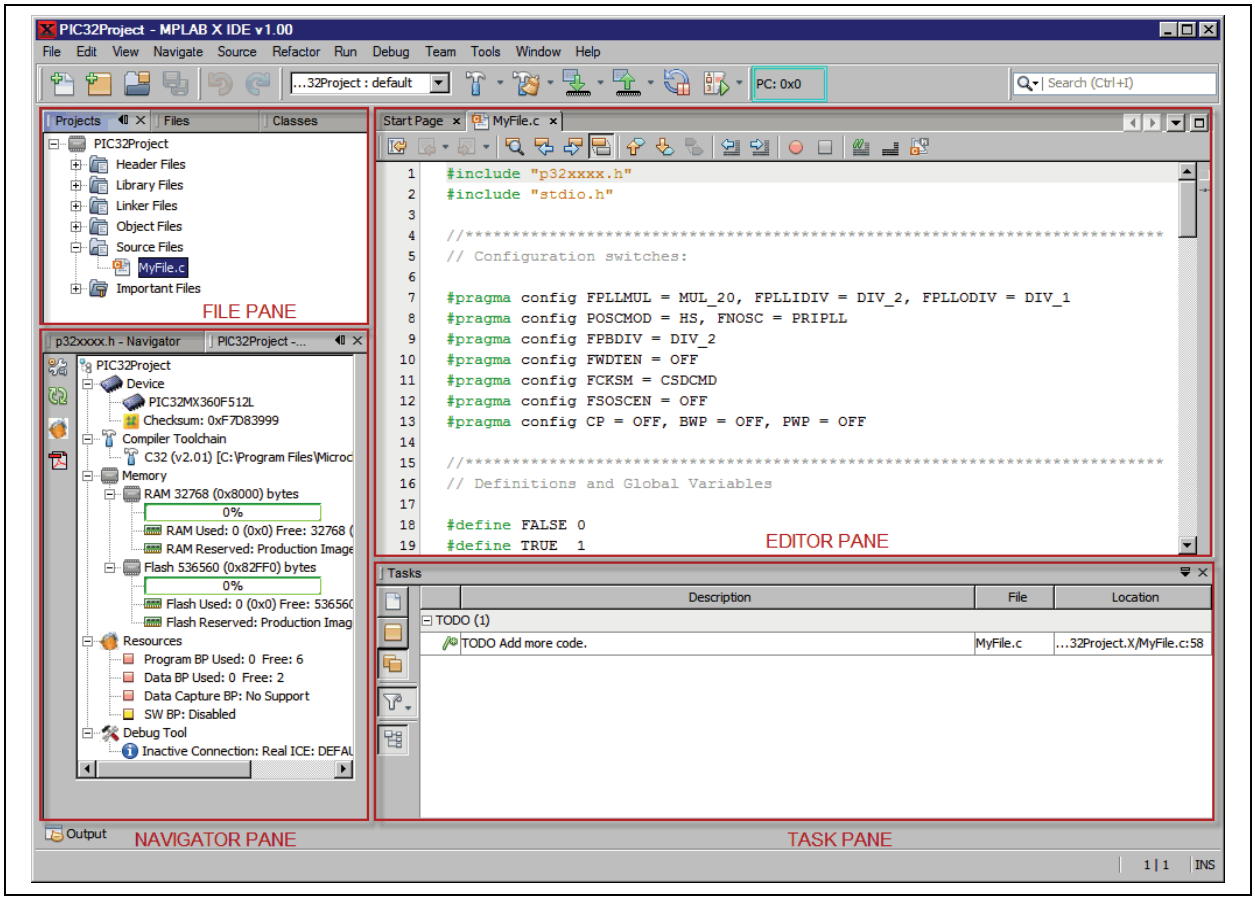
FIGURE 3-5: PROJECT WIZARD – SELECT PROJECT NAME AND FOLDER



3.4 VIEW CHANGES TO THE DESKTOP

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

FIGURE 3-6: MPLAB X IDE DESKTOP



- File pane – A pane with four file-related tabbed windows. The Projects window displays the project tree, the Files window displays the project files, the Classes window displays any classes in the code, and the Services window displays any services available to use for code development.
- Navigator 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.

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.

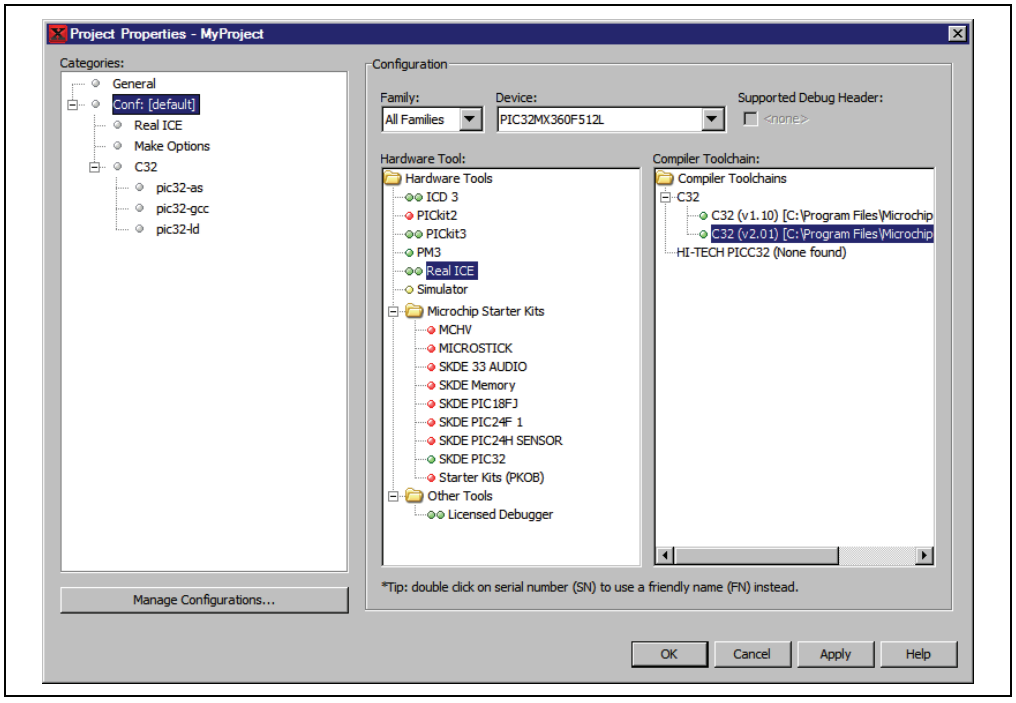
3.5 VIEW OR MAKE CHANGES TO PROJECT PROPERTIES

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 Project window and selecting "Properties".
- clicking on the project name in the Project 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 the previous sections. Then update in this dialog and click **Apply**.

FIGURE 3-7: PROJECT PROPERTIES DIALOG



3.6 SET OPTIONS FOR DEBUGGER, PROGRAMMER OR LANGUAGE TOOLS

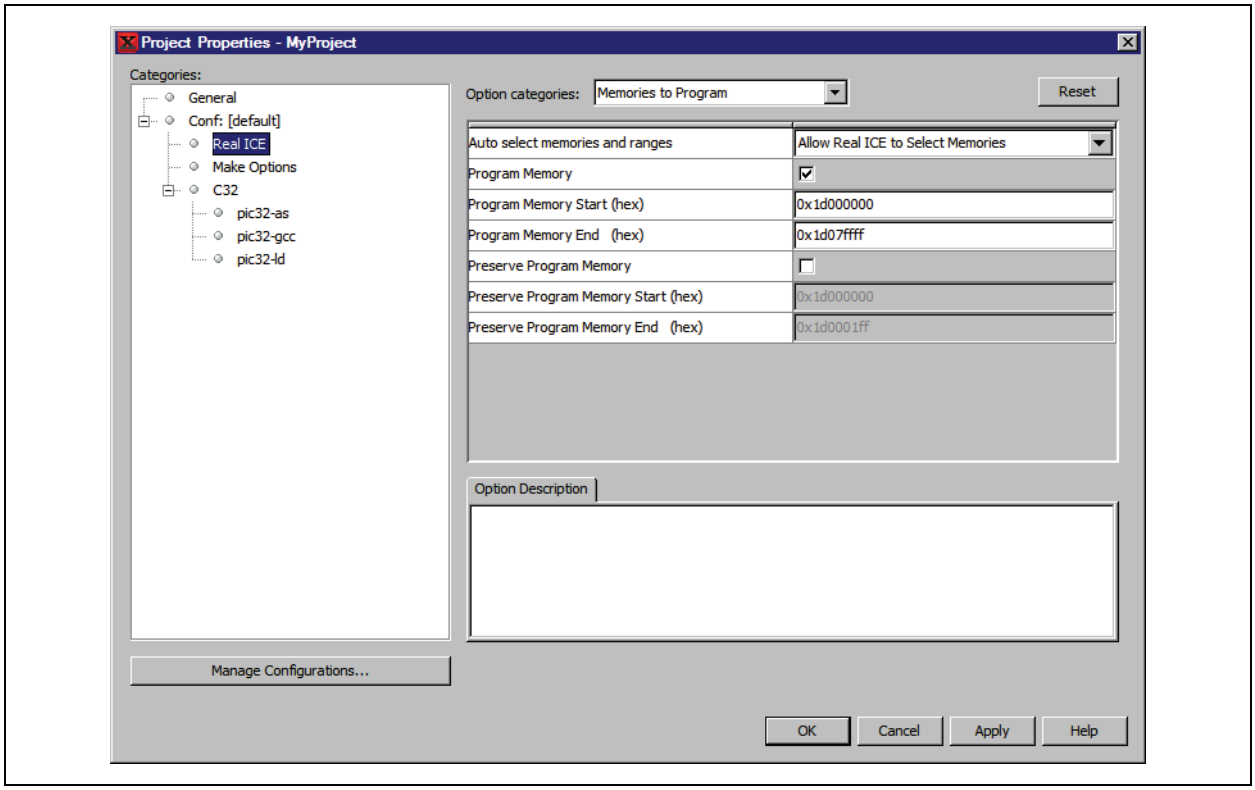
Additionally, you set options for your tools in the Project Properties dialog.

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 3-8: TOOL SETUP PAGE

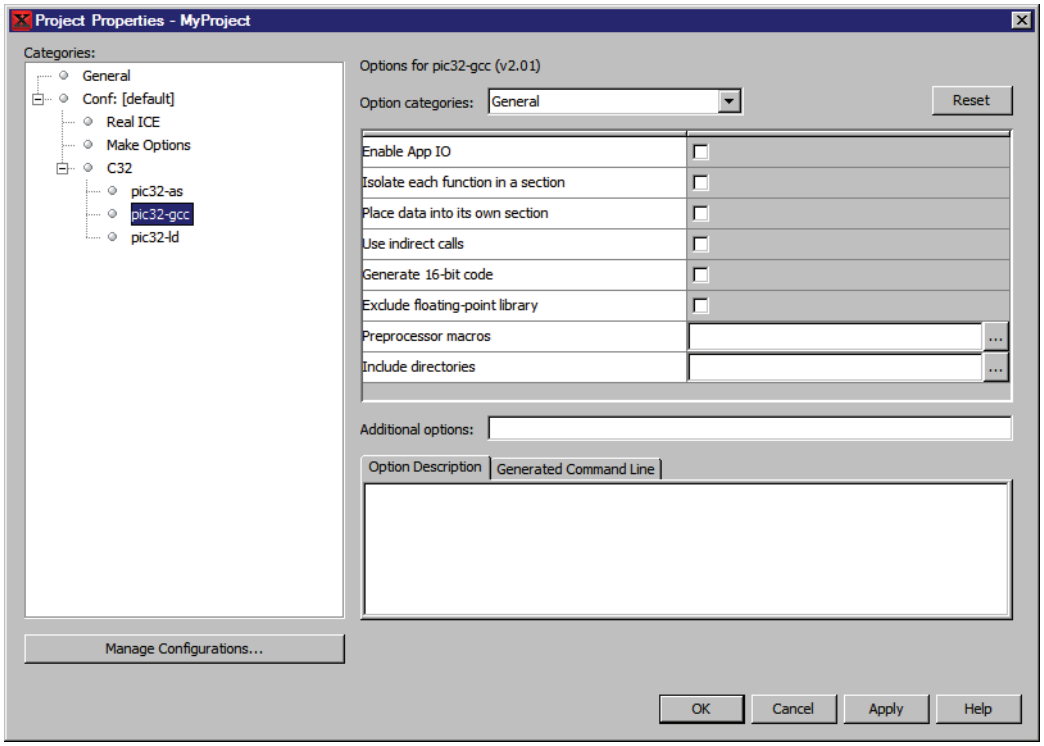


To set up or change language tool options:

- 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 3-9: LANGUAGE TOOL SETUP PAGE



3.7 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*.

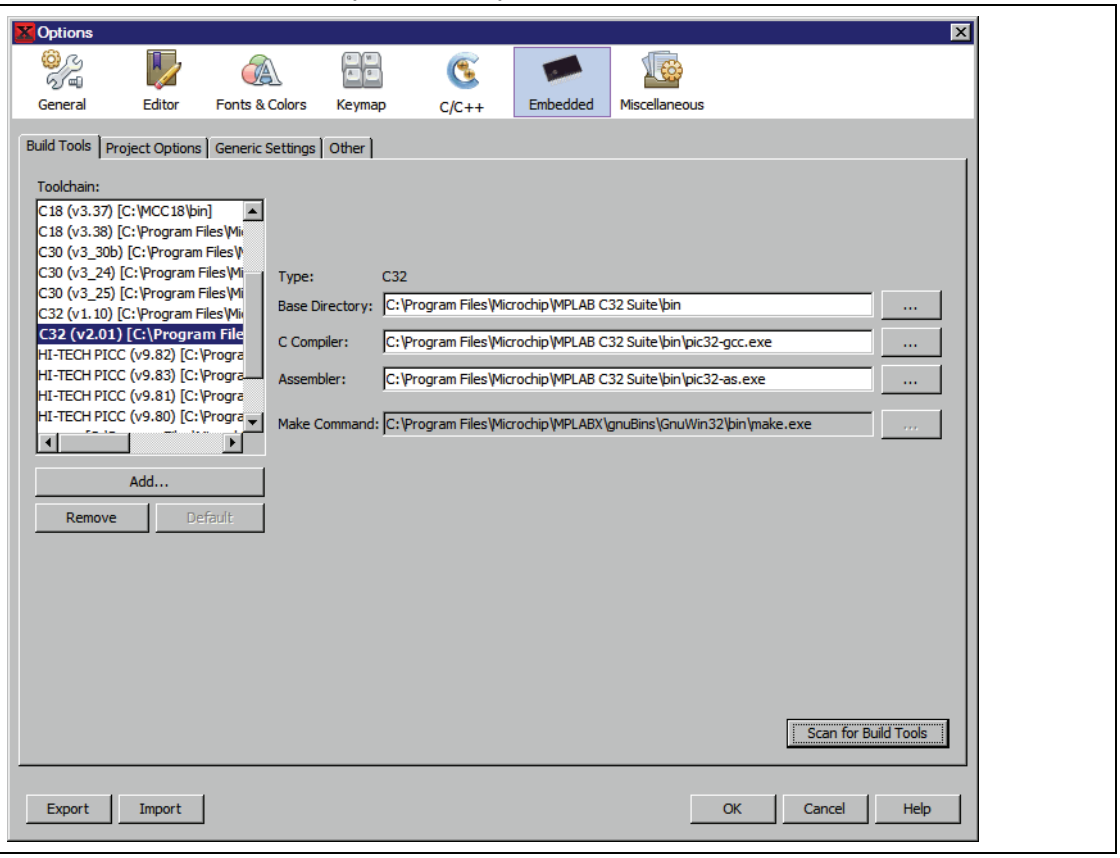
The window should automatically populate with all installed toolchains. If you do not see your tool listed, try the following:

- **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 C32 toolchain is selected.

FIGURE 3-10: LANGUAGE TOOL (COMPILER) LOCATIONS



3.8 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:

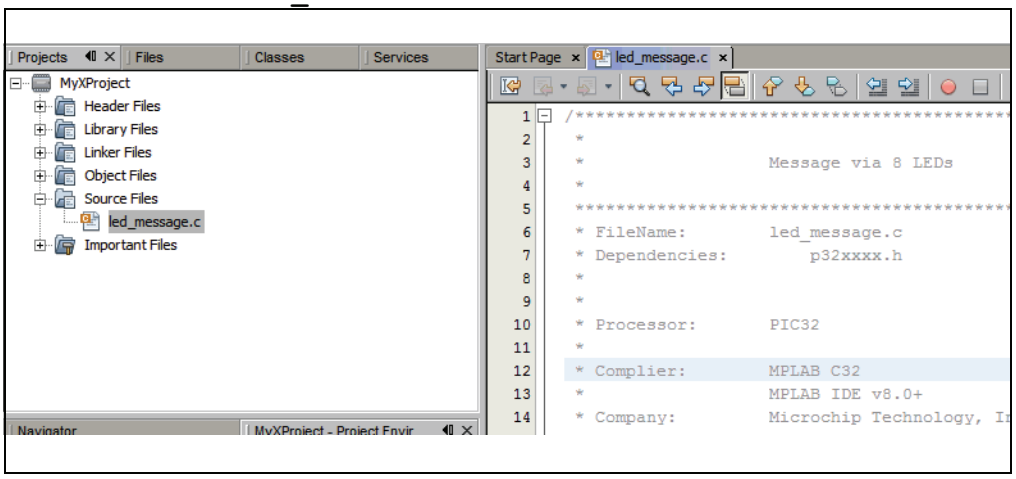
- Right clicking on the project in the Project/File window and selecting "Add Existing Item"
- Right clicking on a logical folder (e.g., Source Files) in the Project/File window and selecting "Add Existing Item"

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. (Use this one.)
- Absolute – Specify the file location by an absolute path.

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 3-11: `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
```

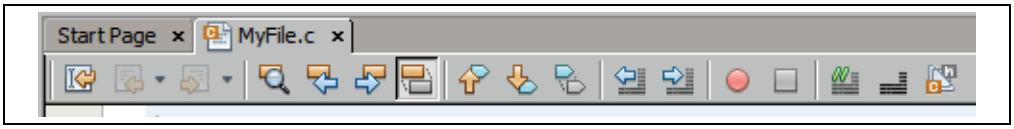
3.9 EDITOR USAGE

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 **Section 6.3 "NetBeans™ Editor"**.

To use editor features, go to the:

- Edit menu (see **Section 9.2.2 "Edit Menu"**.)
- Editor toolbar located at the top of each file's Editor window.
- Window right click (context) menu.

FIGURE 3-12: EDITOR TOOLBAR



3.10 CONFIGURATION BITS

In the sample code, Configuration bits are already set. You should always set your Configuration bits in code. For a summary of Configuration bits settings for different devices, see **Chapter 12. "Configuration Settings Summary"**.

However, you can temporarily change Configuration bits during a Debug Run in the Configuration Bits window (*Window>PIC Memory Views>Configuration Bits*).

3.11 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:

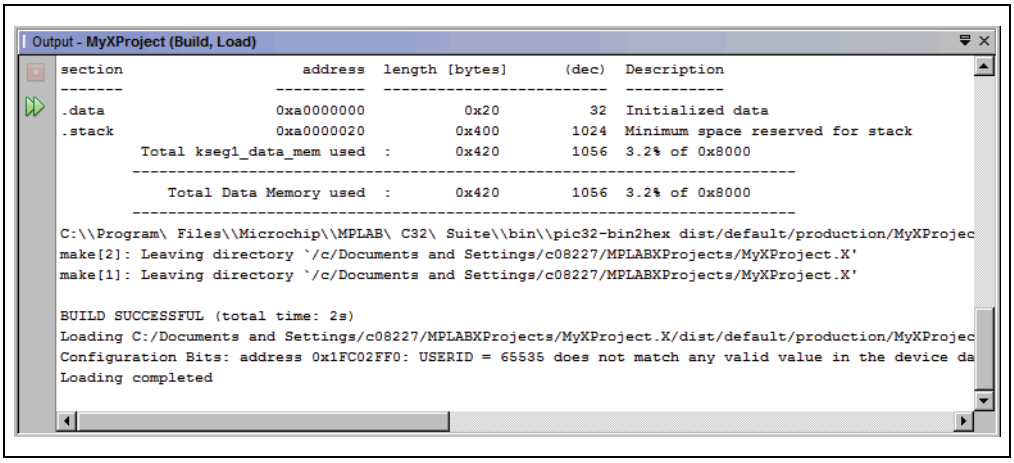
- In the Project window, right click on the project name and select "Build". You may also select "Clean and Build" to remove intermediary files before building.
- Click on the "Build Project" or "Clean and Build Project" toolbar icon.

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 3-13: 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.

3.12 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

3.13 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 table of contents for 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 3.14 "Control Program Execution with Breakpoints"**.

3.14 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.

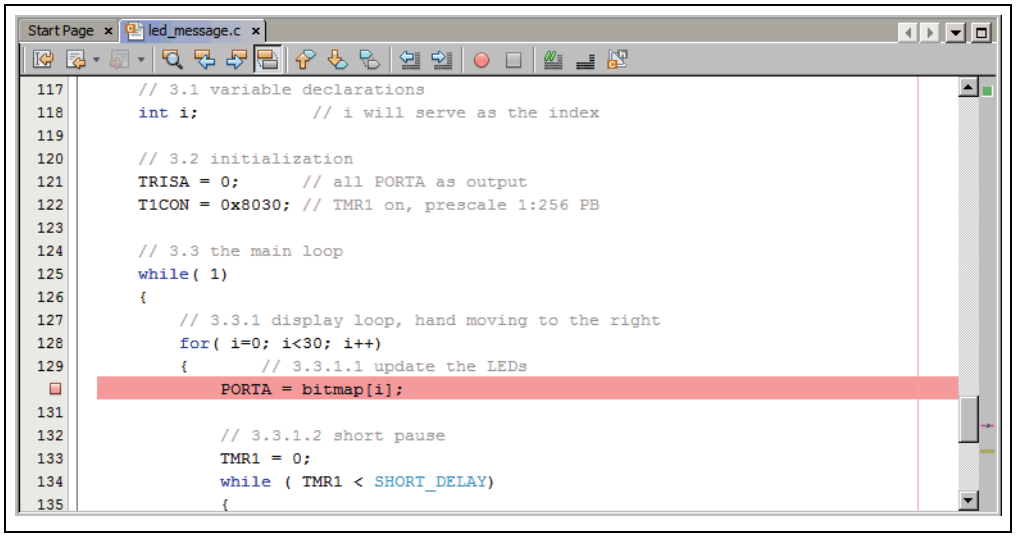
Set a breakpoint on the following line of code:

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

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

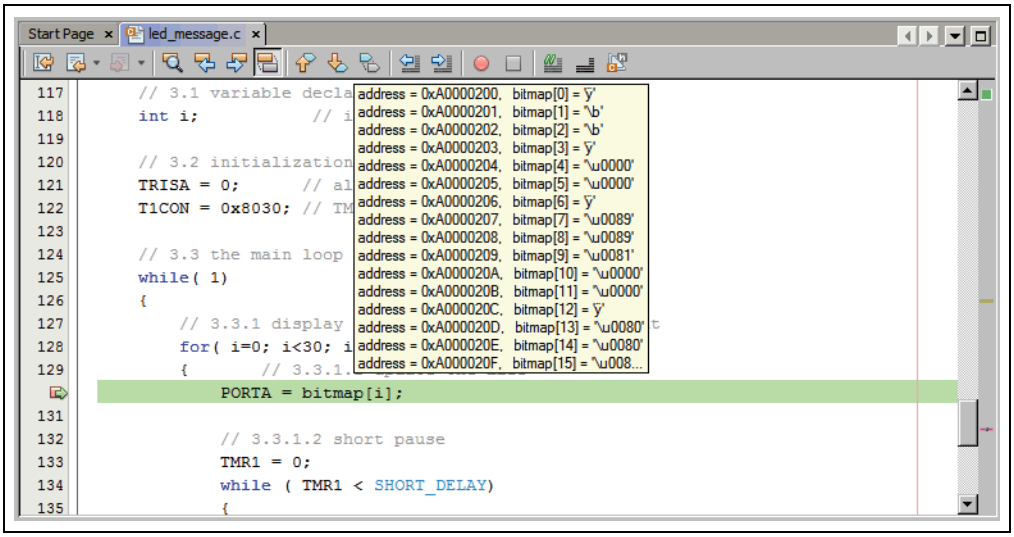
- Click the left margin of the line in the Source Editor
- Press Ctrl-F8

FIGURE 3-14: 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 3-15: 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 table of contents for the NetBeans help topics under *C/C++/Fortran Development>Debugging C/C++/Fortran Applications with gdb>Setting C/C++/Fortran Breakpoints*.

3.15 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.
- Animate – Execute single steps while running, updating the values of the registers as it runs. Animate is slower than Run, but allows you to view changing register values in the Special Function Register window or in the Watch window.

For more on stepping, see the table of contents for 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*.

3.16 WATCH SYMBOL VALUES CHANGE

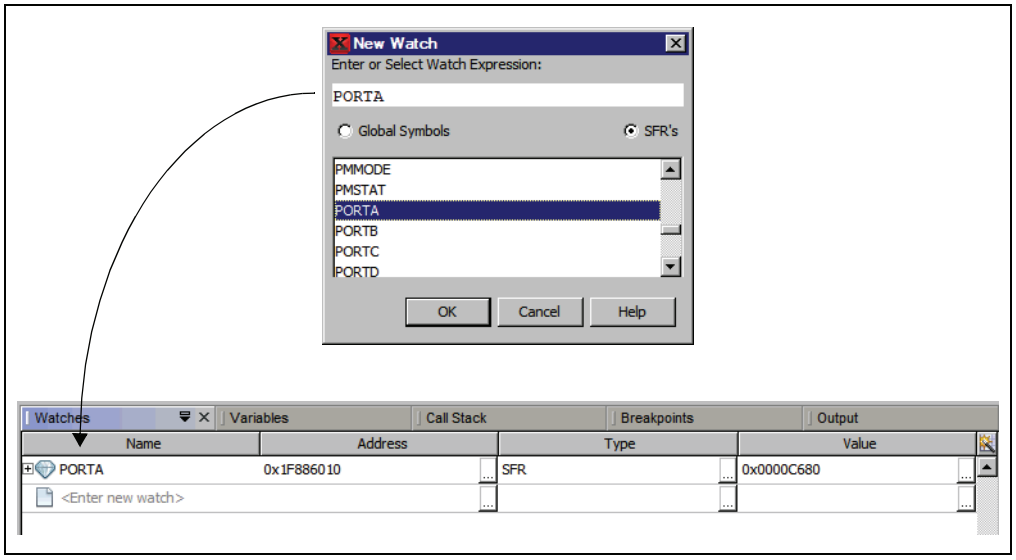
Watch the values of symbols that you select change in the Watches window. Determining if these values are as expected during program execution will help you to debug your code.

To create a new watch:

1. Select *Debug>New Watch*.
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.

Alternatively, a new watch can be created by selecting the symbol name or SFR in the editor and dragging it and dropping it in the Watches window. You could also select the symbol name or SFR in the editor and then select 'New Watch' option from the right click context menu.

FIGURE 3-16: 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 table of contents for 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*.

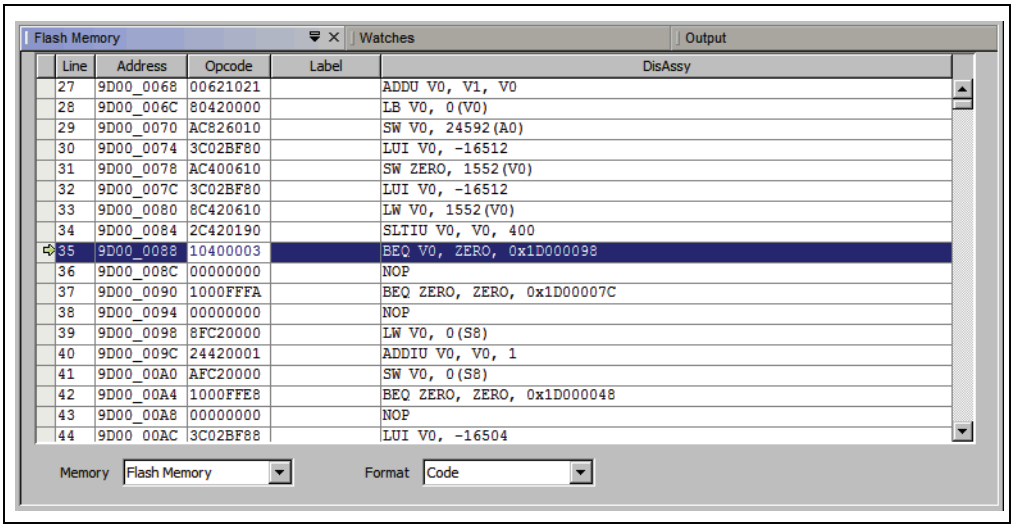
3.17 VIEW DEVICE MEMORY (INCLUDING CONFIGURATION BITS)

MPLAB X IDE has flexible, abstracted memory windows that allow for a more customized view of the different 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>Flash Memory*.
2. The Flash Memory window will open showing the last halt location.

FIGURE 3-17: 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 10.3.4.1 “Memory Window Menu”**.

3.18 PROGRAM A DEVICE

Once your code is debugged, you can program it onto a target device. There are two ways to do this:

- 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.

FIGURE 3-18: PROGRAM ICONS


Run Icon


Hold In Reset Icon


Make and Program Device Icon


Read Device Memory Icon

Chapter 4. Basic Tasks

This chapter provides a step by step guide for working with MPLAB X IDE projects, as summarized in Working with MPLAB X IDE Projects.

4.1 WORKING WITH MPLAB X IDE PROJECTS

The following table shows 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 Project 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 Library and Other Files to a Project.
7. Set File Properties to keep or exclude individual files 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 and Variable 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)

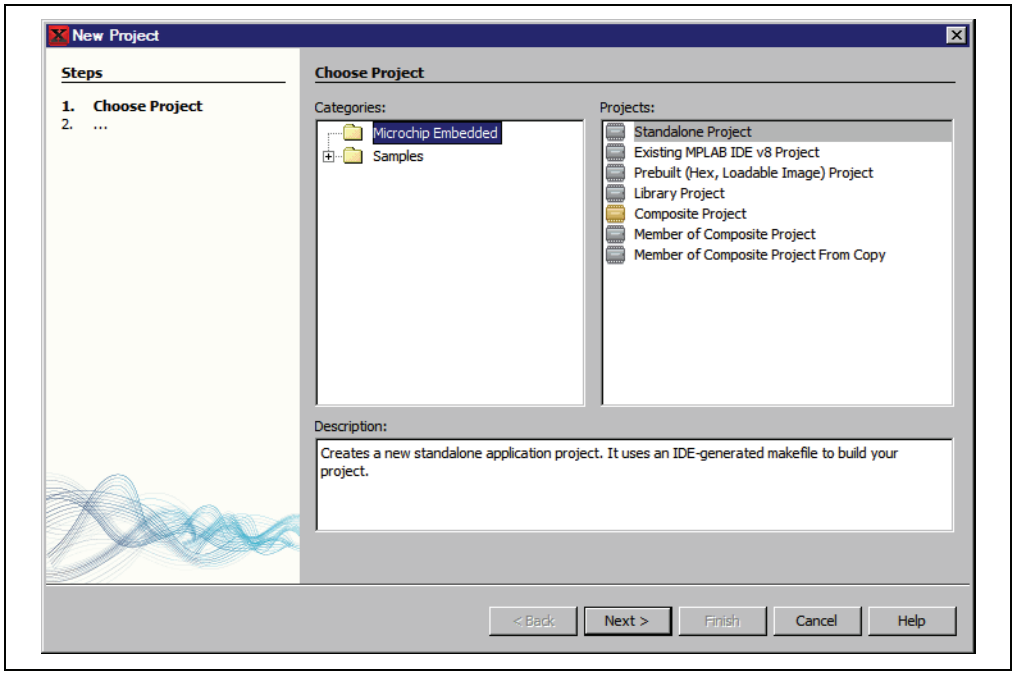
The New Project wizard with launch to guide you through new project setup.

Step 1 will first ask you to choose a project category. This is a NetBeans dialog, so to work with Microchip products you must choose "Microchip Embedded". Secondly, you will choose a project type:

- Stand-alone Project – Create a new C and/or assembly code project.
- Existing MPLAB IDE v8 Project – Convert your existing MPLAB IDE v8 project into an MPLAB X IDE project.
- Prebuilt (Hex, Loadable Image) Project – Load an existing project image into MPLAB X IDE.
- Library Project – Create a new C and/or assembly code project that will build into a library instead of executable hex file.
- Composite Projects – Create a composite project to contain more than one project for application development.

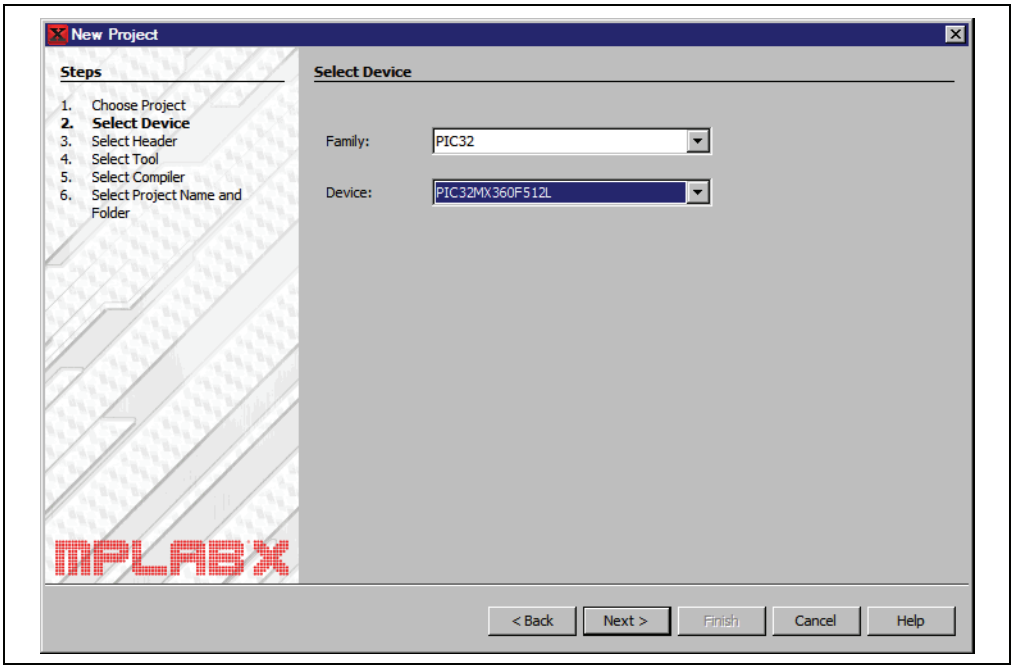
Once you have made your selections, click **Next>** to move to the next dialog.

FIGURE 4-1: PROJECT WIZARD – CHOOSE PROJECT



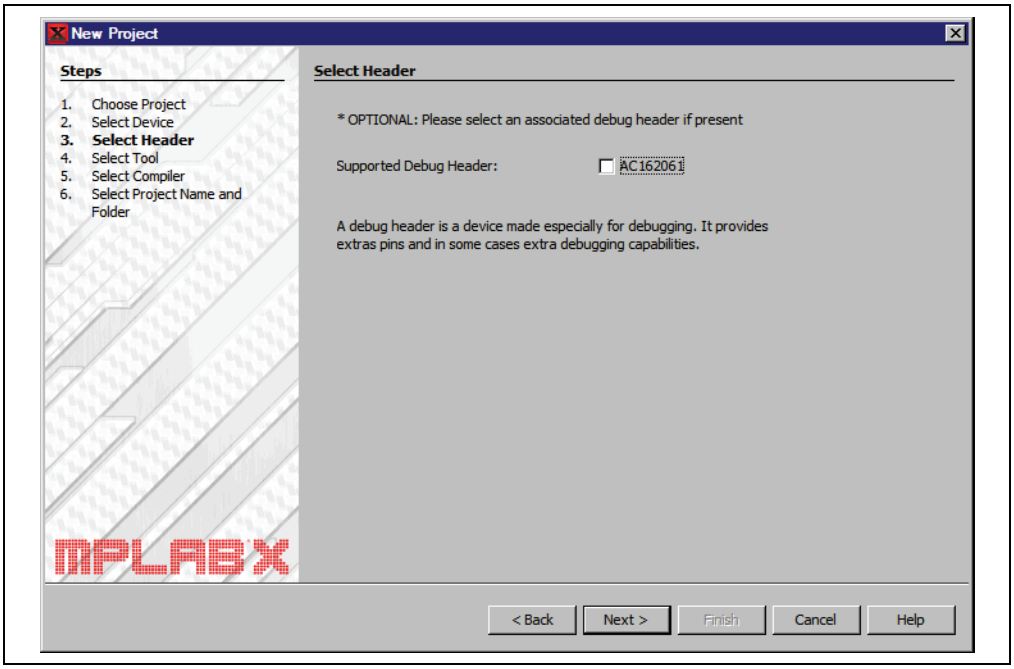
Step 2 is an MPLAB X IDE dialog and so will look different from the Step 1 dialog. Select the device you will be using in your application from the “Device” drop-down list. To narrow your selection list, chose a Family first. Click **Next>**.

FIGURE 4-2: PROJECT WIZARD – SELECT DEVICE



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 Header Specification (DS51292 or online help). Then choose whether or not to use a header. Click **Next>** when done.

FIGURE 4-3: PROJECT WIZARD – SELECT HEADER



Step 4 involves selecting the tool. Tool support for the selected device is signified by the colored circles in front of the tool name, just as MPLAB IDE v8 had lights on the Device Selection dialog. The circle colors mean the following:

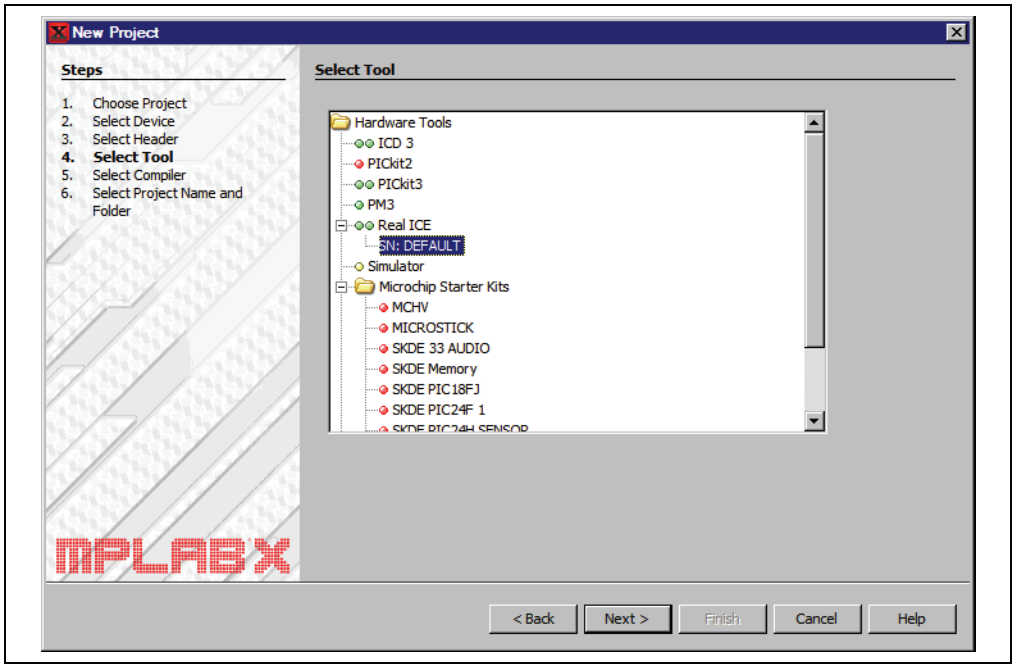
- Green – full support
- Yellow – beta support
- Red – no support yet

If you cannot see the colors, mouse over a circle to pop up text about 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

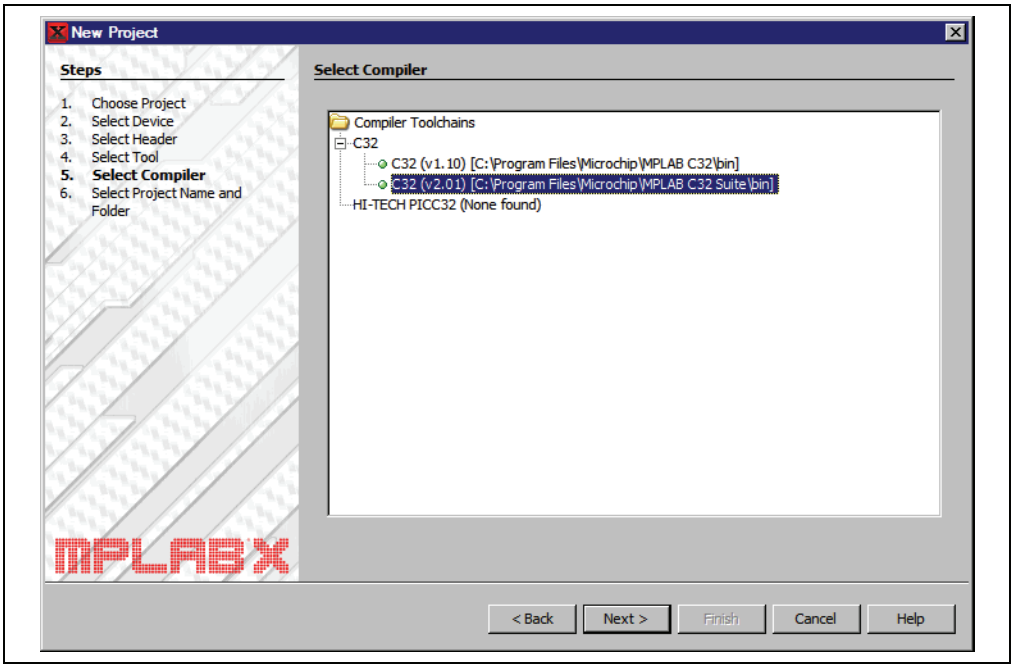


Step 5 involves selecting the language tool, either a C compiler or assembler. Again, the colored circle in front of the compiler name signifies the device support level. Mouse over for text.

Based on the device selected in Step 2, the New Project wizard intelligently displays only those language tools that are suitable for the currently selected device. 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 4-5: PROJECT WIZARD – SELECT LANGUAGE TOOL



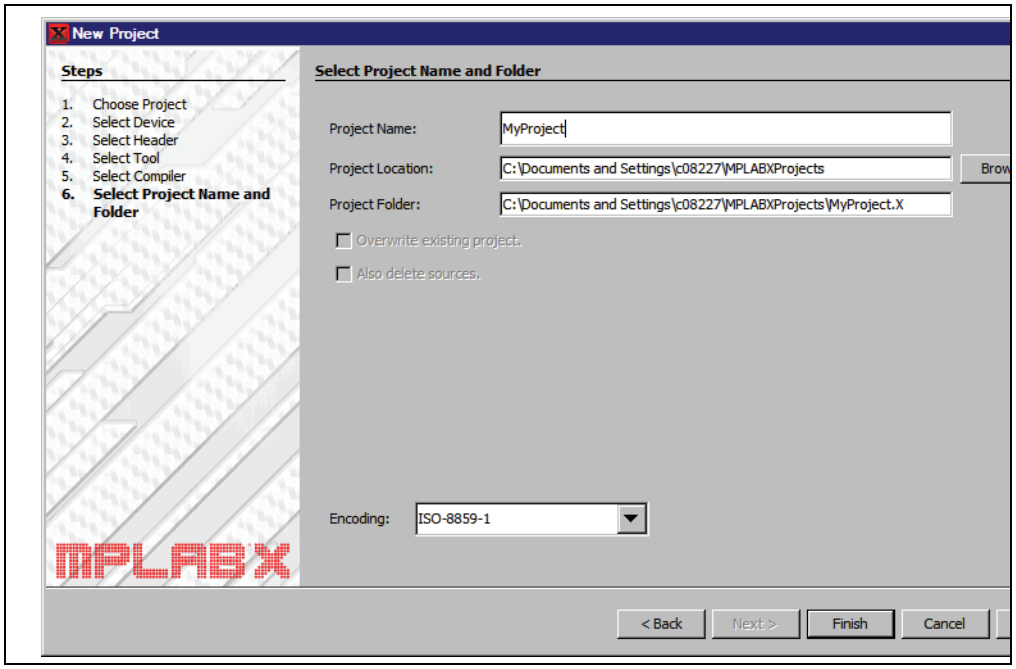
Step 6 involves selecting the project name and location. Enter the project name and then browse to a folder. You can create a new project folder if you need one. By default, projects will be placed in:

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

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

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

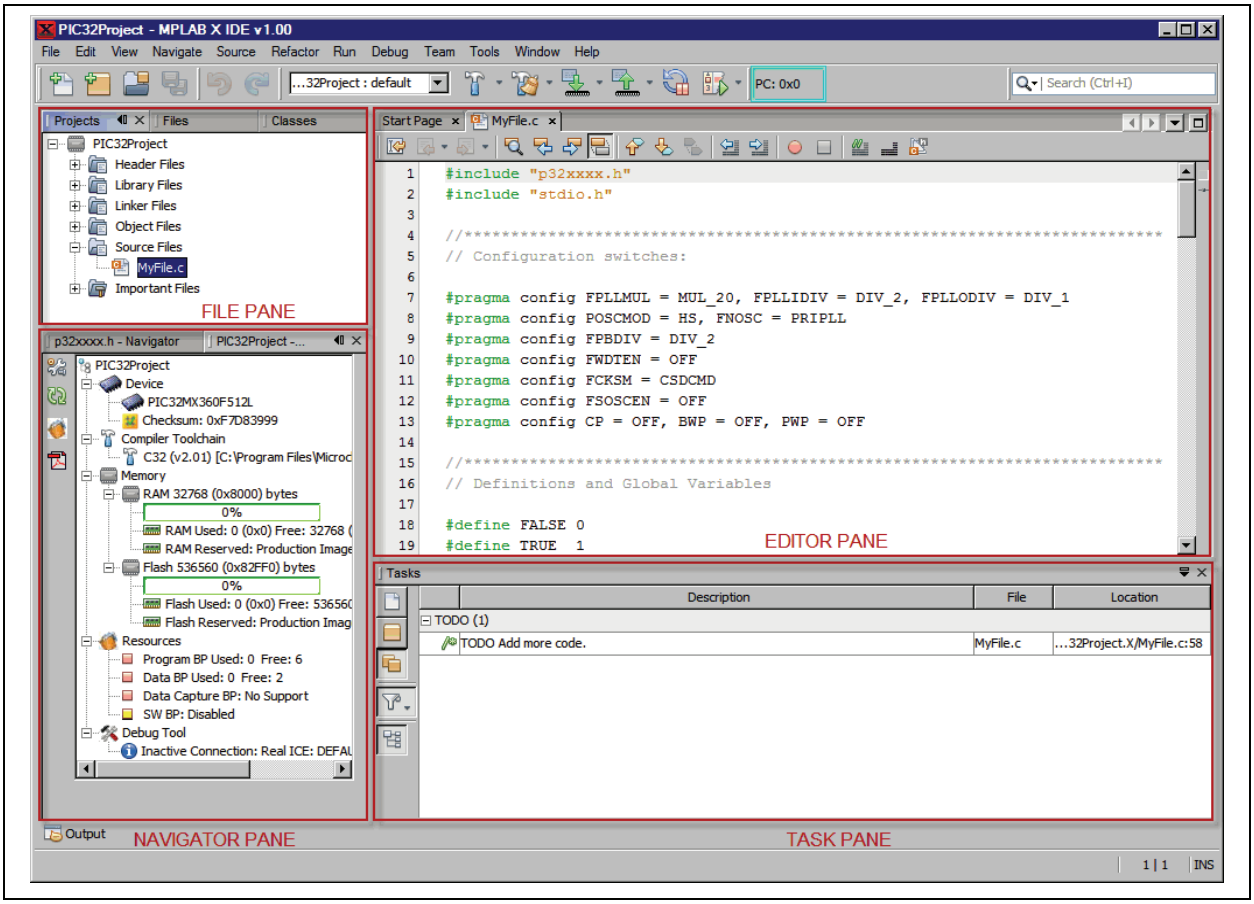
FIGURE 4-6: PROJECT WIZARD – SELECT PROJECT NAME AND FOLDER



4.3 VIEW CHANGES TO THE DESKTOP

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

FIGURE 4-7: MPLAB X IDE DESKTOP



- File pane – A pane with four file-related tabbed windows. The Projects window displays the project tree, the Files window displays the project files, the Classes window displays any classes in the code, and the Services window displays any services available to use for code development.
- Navigator pane – A pane that displays information on the symbols and variables in the file selected in the File pane.
- 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.

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.

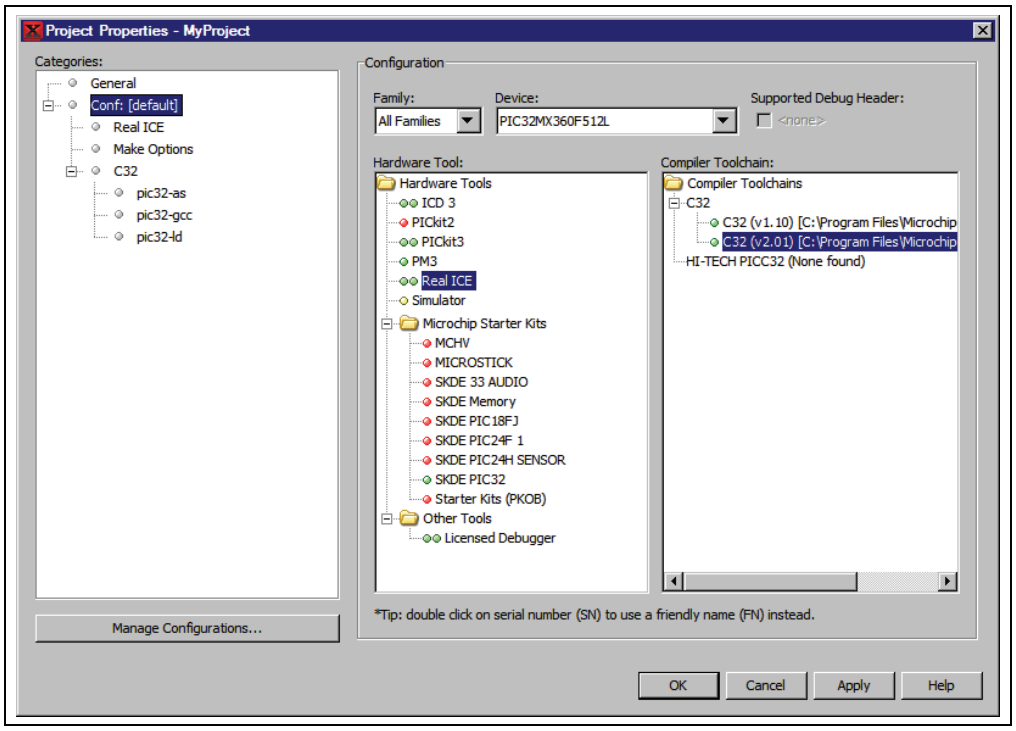
4.4 VIEW OR MAKE CHANGES TO PROJECT PROPERTIES

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 Project window and selecting "Properties".
- clicking on the project name in the Project 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, make a different selection and click **OK**.

FIGURE 4-8: PROJECT PROPERTIES DIALOG



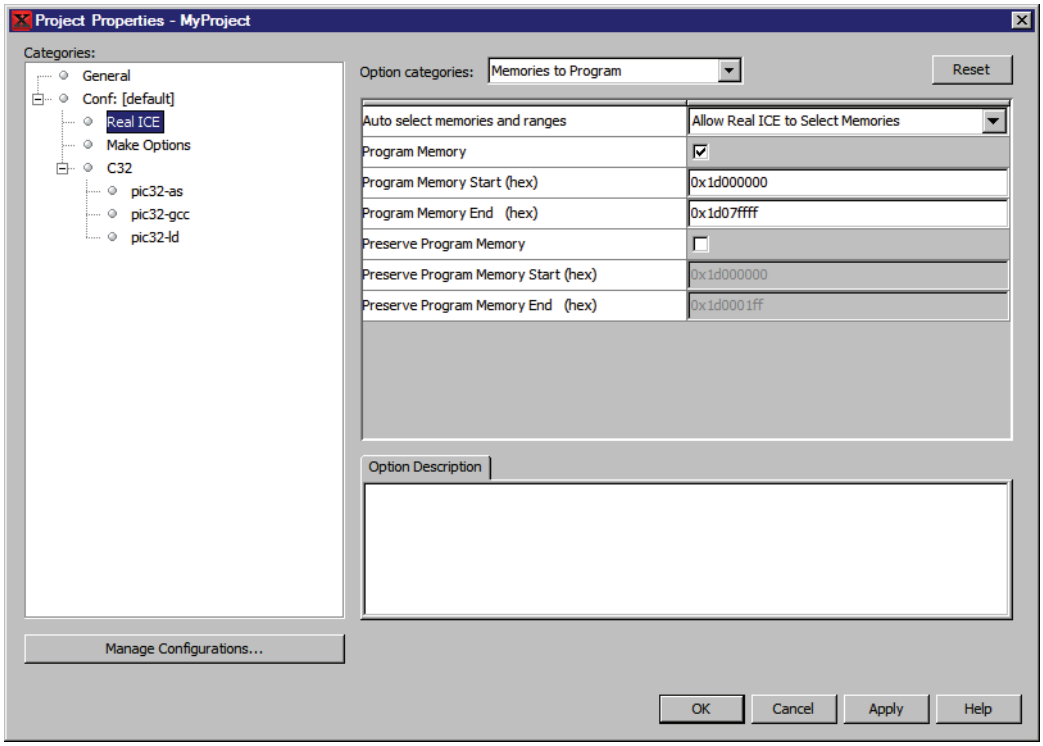
4.5 SET OPTIONS FOR DEBUGGER, PROGRAMMER OR LANGUAGE TOOLS

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

FIGURE 4-9: TOOL SETUP PAGE

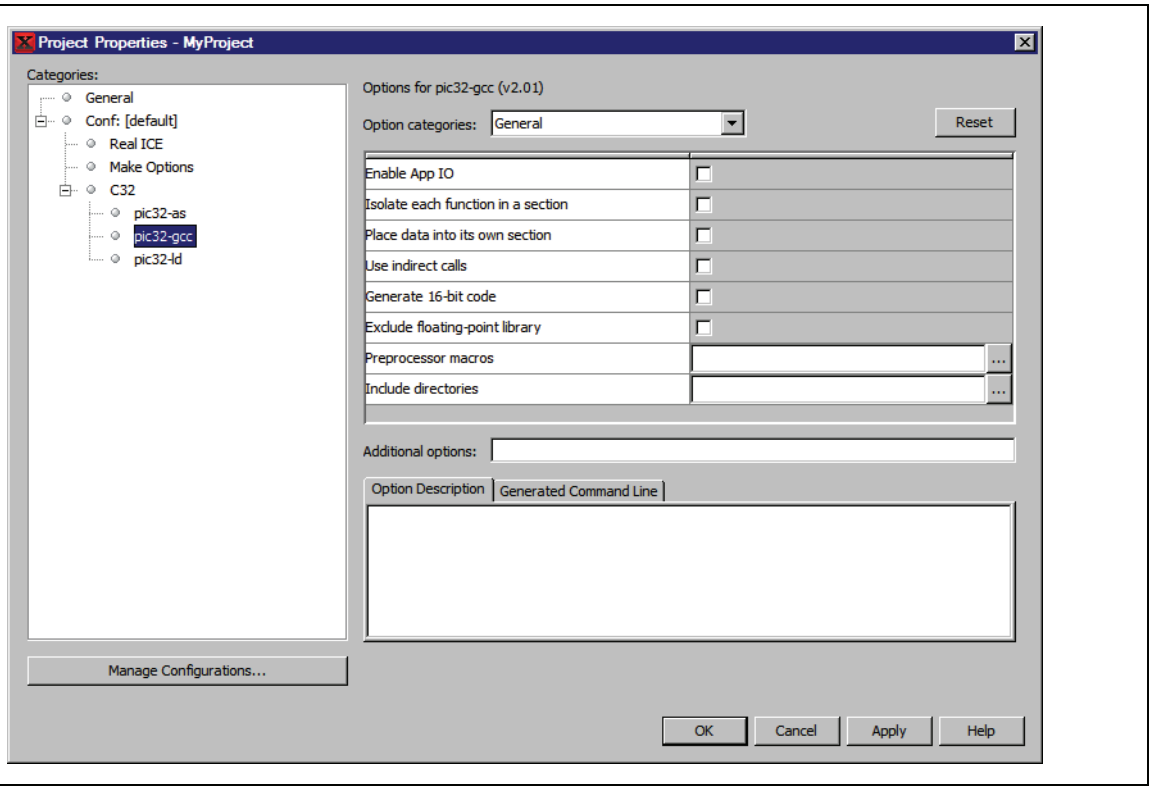


To set up or change language tool options:

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

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

FIGURE 4-10: LANGUAGE TOOL SETUP 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**: 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*.

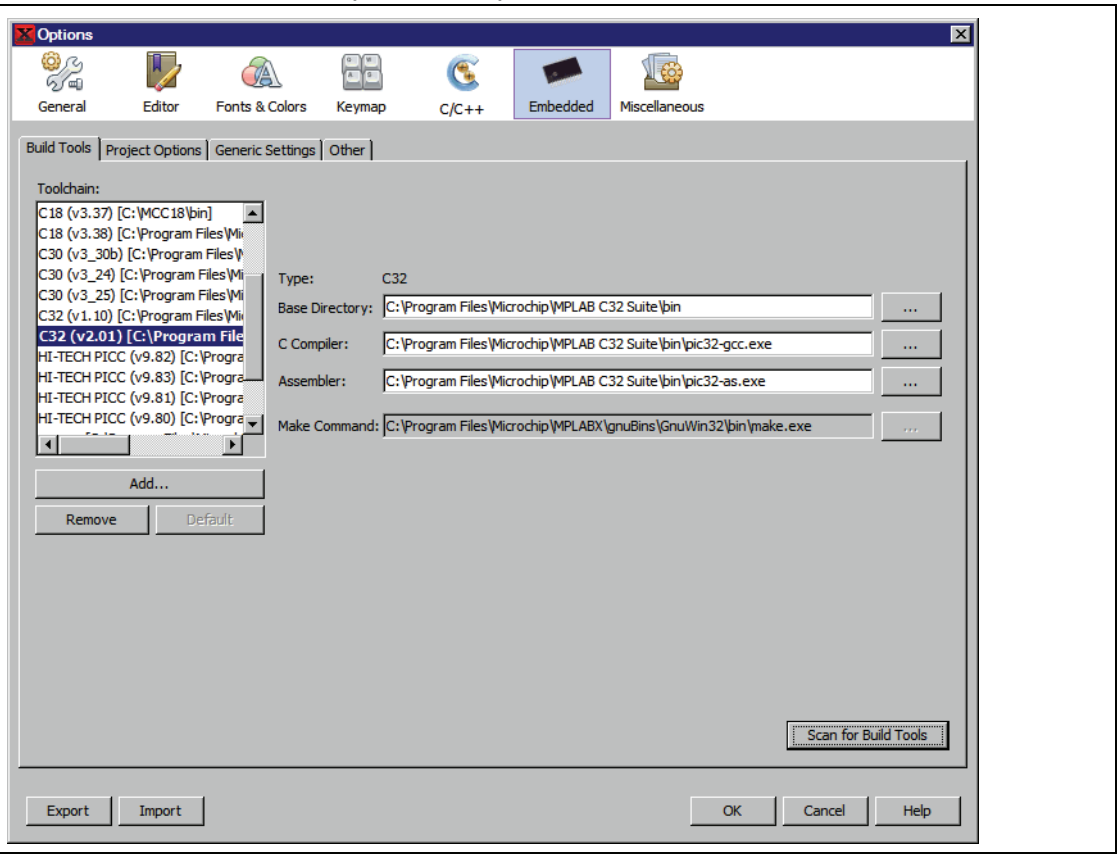
The window should automatically populate with all installed toolchains. If you do not see your tool listed, try the following:

- **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.

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

FIGURE 4-11: LANGUAGE TOOL (COMPILER) LOCATIONS



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 10.3.5.2 "Project Options tab"**).
- Generic Settings – Set up the log file and other project features (**Section 10.3.5.3 "Generic Settings tab"**).
- Other – Edit the lists of accepted file extensions for C source files and header files (**Section 10.3.5.4 "Other tab"**).

4.8 CREATE A NEW PROJECT FILE

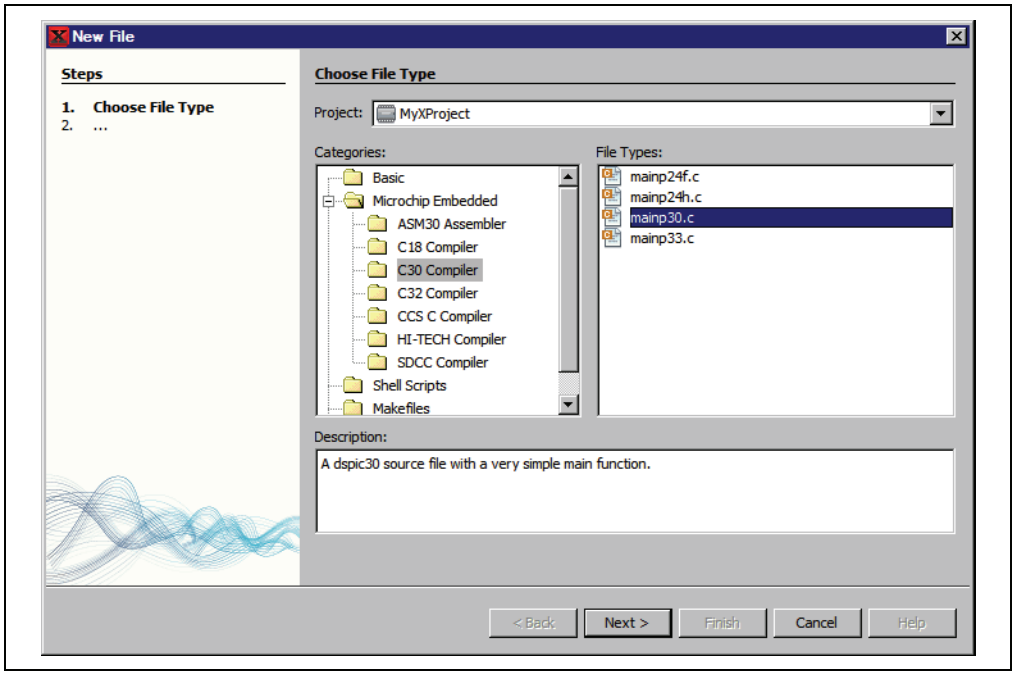
New project files can be created by doing one of the following:

- 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 setup.

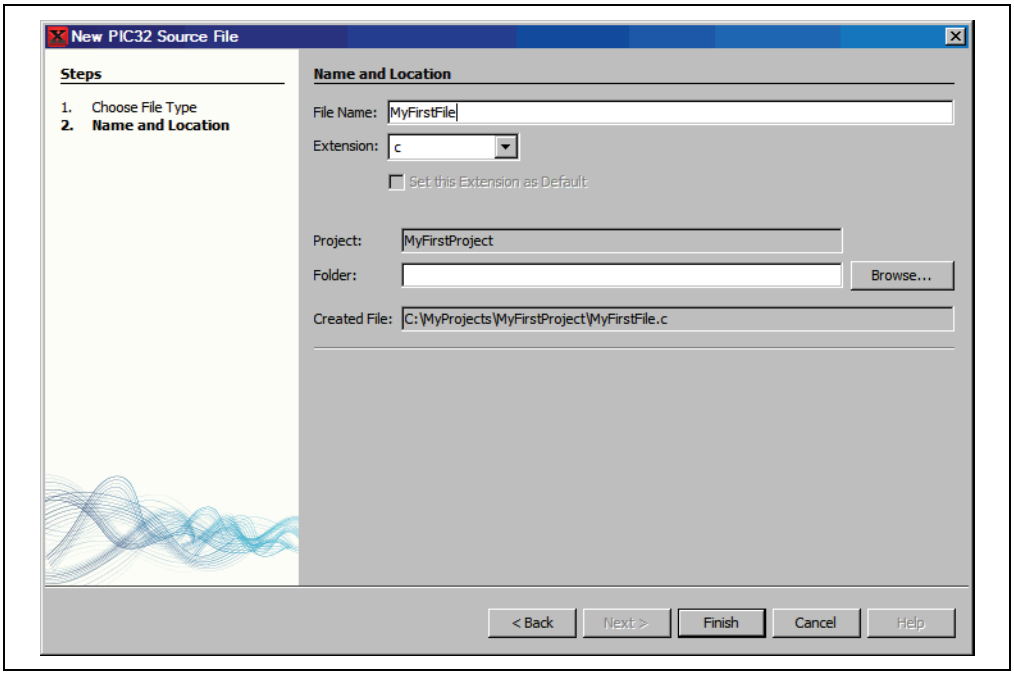
- **Step 1. Choose File Type:** Choose the file category by expanding "Microchip Embedded" to find an appropriate selection. Then select a file type.

FIGURE 4-12: FILE WIZARD – CHOOSE CATEGORY AND TYPE



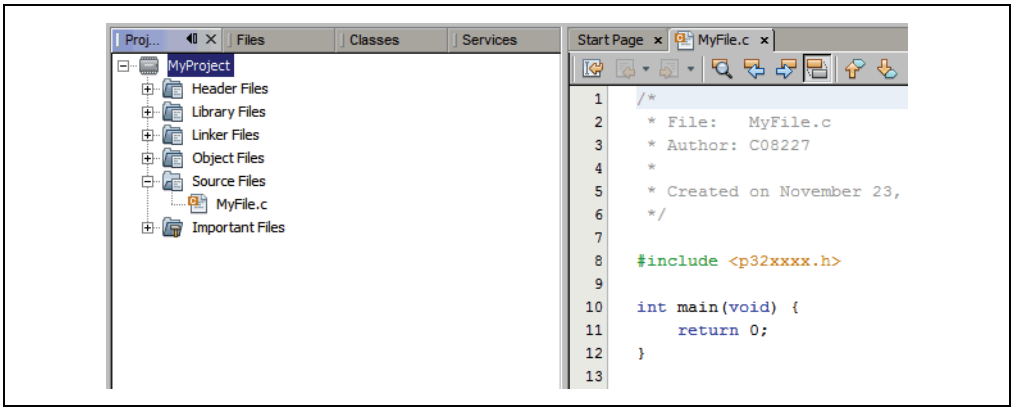
- **Step 2. Name and Location:** Name the file and place it in the project folder.

FIGURE 4-13: 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 4-14: 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.

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, 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. Therefore, you must follow the steps above to direct the project to the v8 folder.

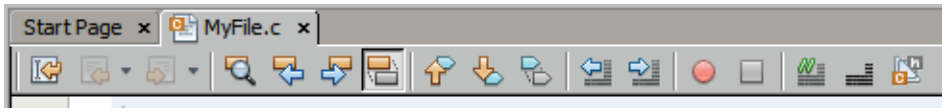
4.10 EDITOR USAGE

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 **Section 6.3 "NetBeans™ Editor"**.

To use editor features, go to the:

- Edit menu (see **Section 9.2.2 "Edit Menu"**.)
- Editor toolbar located at the top of each file's Editor window.

FIGURE 4-15: EDITOR TOOLBAR



- Window right click (context) menu.

To set editor properties:

1. Select *Tools>Options* to open the Options dialog.
2. Click on the **Editor** button. Then click on a tab to set up a 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.

4.11 ADD LIBRARY AND OTHER FILES TO A PROJECT

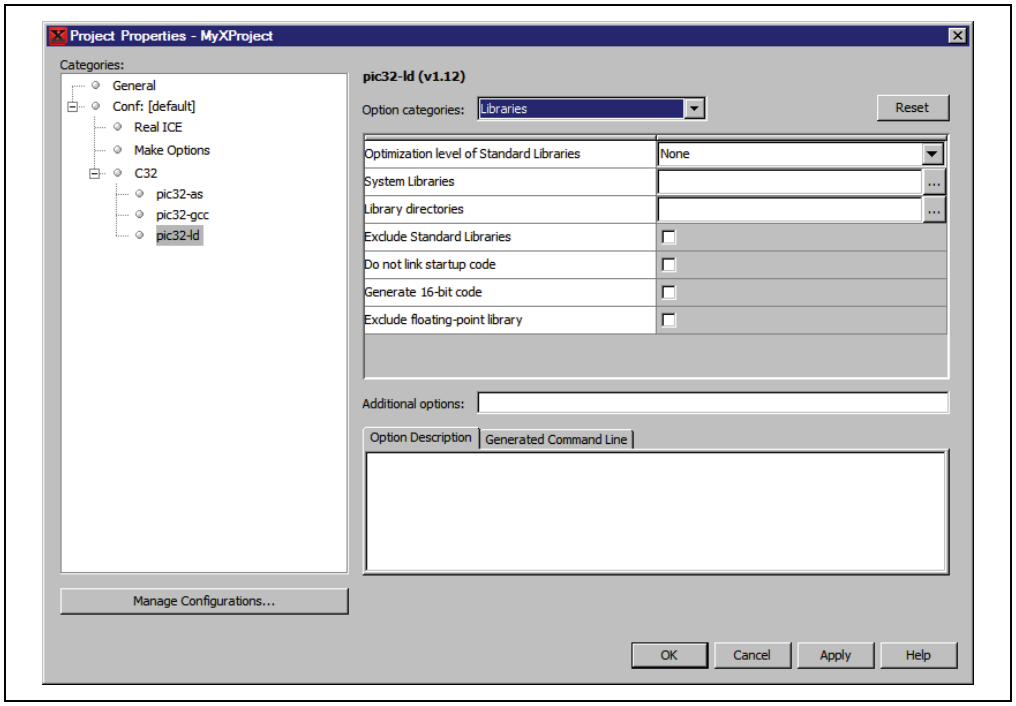
You can reference library files to be used by the linker in the Project Properties dialog. (See **Section 4.4 "View or Make Changes to Project Properties"** to access this dialog.)

In the left pane, click on the name of the linker and then in the right pane, select the option "Libraries" (Figure 4-16).

You may also make any other project that creates a library required for your project, i.e., establish a dependency. For more on this, see the table of contents for 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*.

To add an existing library file to the project, right click on the project or a logical folder in the Project window and select "Add Existing Item". Select the file reference: auto, relative or absolute, and then press <Enter>.

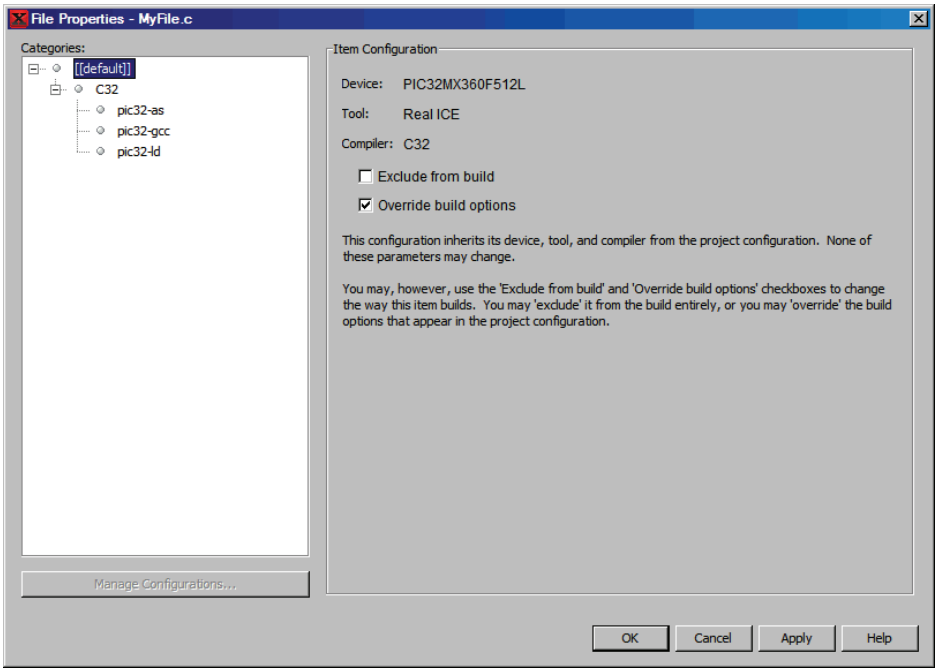
FIGURE 4-16: SELECT AND SET UP LIBRARIES



4.12 SET FILE 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 Project window and selecting “Properties”. Select to exclude the file from the project build or override the project build options with ones selected here. To override build options, check the checkbox and then click “Apply” to see selection options appear under “Categories”.

FIGURE 4-17: FILE PROPERTIES



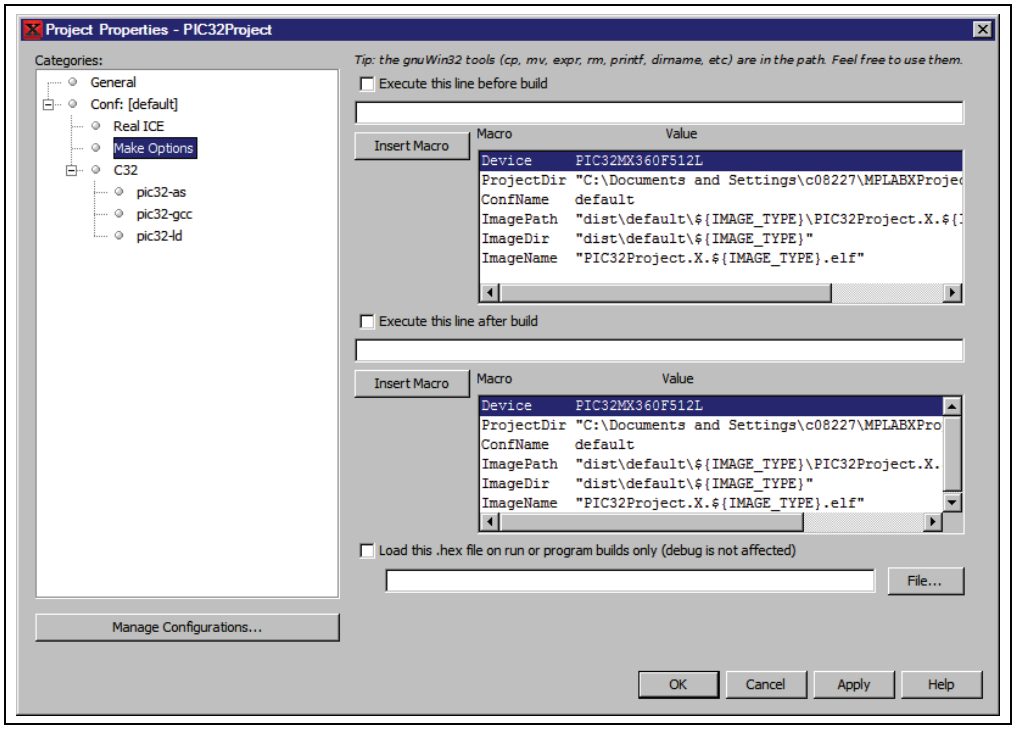
4.13 SET BUILD PROPERTIES

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

TABLE 4-1: BUILD ALTERNATIVE OPTIONS

Item	Description
Execute this line before the build	Pre-build/make steps. Select the macro from the list and click **Insert Macro**. Additional options may be available depending on the compiler used.
Execute this line after the build	Post-build/make steps. Select the macro from the list and click **Insert Macro**. Additional options may be available depending on the compiler used.
Load this hex file on run or program builds only (debug not affected)	Alternative hex file load on build complete.

FIGURE 4-18: MAKE OPTIONS PROPERTIES



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 Project 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.)

To view checksum information:

- Open the Dashboard window (see **Section 5.10 "View the Dashboard Display"**) to see the checksum after a build.

For more information on building your project, also see the table of contents for 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.

To run your application code:

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

Make and Program Device Project Icon

Run progress will be visible in the Output window.

How Run works:

Clicking "Run Project" will cause a build if the make process determines it is necessary.

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 Icon

MPLAB X IDE operation connects to the hardware tool at runtime (Run or Debug Run). To always be connected to the hardware tool (like MPLAB IDE v8), see *Tools>Options*, **Embedded** button, **Generic Settings** tab, "Maintain active connection to hardware tool" checkbox.

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

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.

To debug your application code:

1. In the Project 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 session.

Debug Run Icon

How Debug Run works:

Clicking "Debug Project" will cause a build if the make process determines it is necessary.

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.

MPLAB X IDE operation connects to the hardware tool at runtime (Run or Debug Run). To always be connected to the hardware tool (like MPLAB IDE v8), see *Tools>Options*, **Embedded** button, **Generic Settings** tab, "Maintain active connection to hardware tool" checkbox.

For simulators, a debug session will be started.

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 table of contents for the NetBeans help topic *C/C++/Fortran Development>Debugging C/C++/Fortran Applications with gdb*.

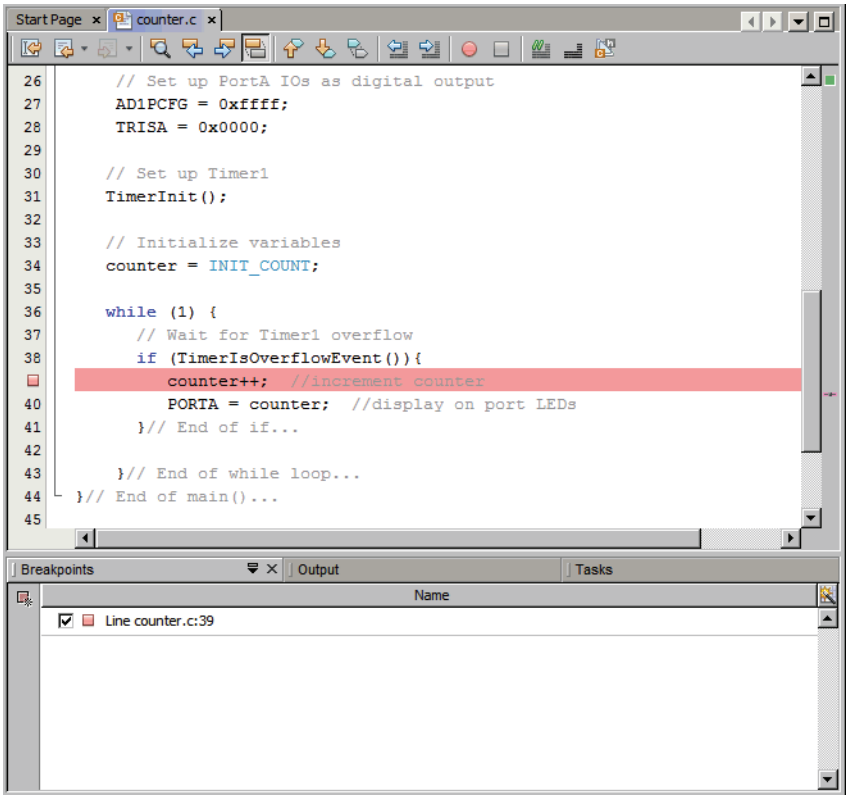
The difference between Run and Debug Run will become apparent when working with debug features, beginning with **Section 4.17 "Control Program Execution with Breakpoints"**.

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 table of contents for the NetBeans help topics under *C/C++/Fortran Development>Debugging C/C++/Fortran Applications with gdb>Setting C/C++/Fortran Breakpoints*.

FIGURE 4-19: BREAKPOINT AND BREAKPOINTS WINDOW



- Breakpoint Usage Considerations

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.

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.
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 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. Drag breakpoints under the sequence to the desired location to set the sequence. 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 5.7 “Use the Stopwatch”**).

To determine breakpoint resources:

- Open the Dashboard window (see **Section 5.10 “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 (**5.10 “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.
- Animate – Execute single steps while running, updating the values of the registers as it runs. Animate is slower than Run, but allows you to view changing register values in the Special Function Register window or in the Watch window.

In addition to the Editor window, you can single-step through code in the Disassembly window (**Section 5.8 "View the Disassembly Window"**) and program memory in a Memory window.

For more on stepping, see the table of contents for 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 AND VARIABLE VALUES CHANGE

Watch the values of symbols and local variables that you select change in the Watches and Variables windows, respectively. 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.

- Watches
- Variables

4.19.1 Watches

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:

You can add a symbol or SRF to the Watches window by using the New Watch dialog or by adding the item directly.

- Use the New Watch dialog by doing one of the following:
  - 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**.
- Add the symbol or SFR directly by doing one of the following:
  - Click on <enter new watch> in the Watches window to type in the symbol or SFR name.
  - Select the symbol or SFR name in the Editor window and drag-and-drop it into the Watches window.

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 symbol or SFR as a runtime watch, follow the instructions under “To add a new watch”, except select “New Runtime Watch” instead of “New Watch”.

To view symbol changes:

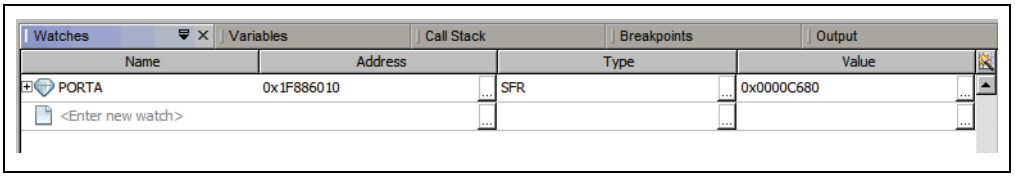
1. Debug Run and then Pause your program.
2. Click the Watches tab to make the window active.
3. For watch symbols, continue to Debug Run and Pause to see changing values. For runtime watch symbols, continue Debug Run and watch the values change as the program executes.

To change the radix of a watch symbol:

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

For more on watches, see the table of contents for 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*.

FIGURE 4-20: WATCHES WINDOW – PROGRAM PAUSE



4.19.2 Variables

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:

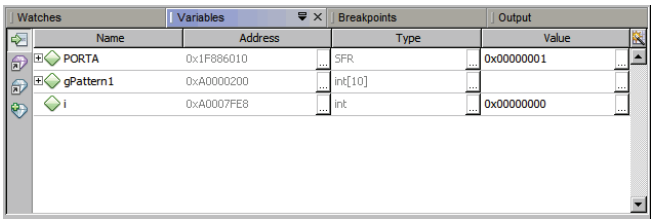
1. Debug Run and then Pause your program.
2. Click the Variables tab to view the window and see the local variable value.

To change the radix of a variable:

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

For more on variables, see the table of contents for 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*.

FIGURE 4-21: VARIABLES WINDOW – PROGRAM PAUSE



4.20 VIEW/CHANGE DEVICE MEMORY (INCLUDING CONFIGURATION BITS)

MPLAB X IDE has flexible, abstracted memory windows that allow for 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.

To view and change device memory:

1. Select a memory view from *Window>PIC Memory Views*. The available choices are:

TABLE 4-2: MEMORY VIEWS

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)
Configuration Bits	All Configuration registers
EE Data Memory	All EE Data memory on the device
Other Memory	Other types of memory

TABLE 4-3: MEMORY VIEWS – PIC32 MCU

Type	Description
Flash 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
Other Memory	Other types of memory

2. A Memory window will open in the Editor pane. From a Memory window, you may further modify your view by selecting the type of memory and memory format in drop-down boxes.

TABLE 4-4: MEMORY WINDOW OPTIONS

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 4-5: MEMORY WINDOW OPTIONS – PIC32 MCU

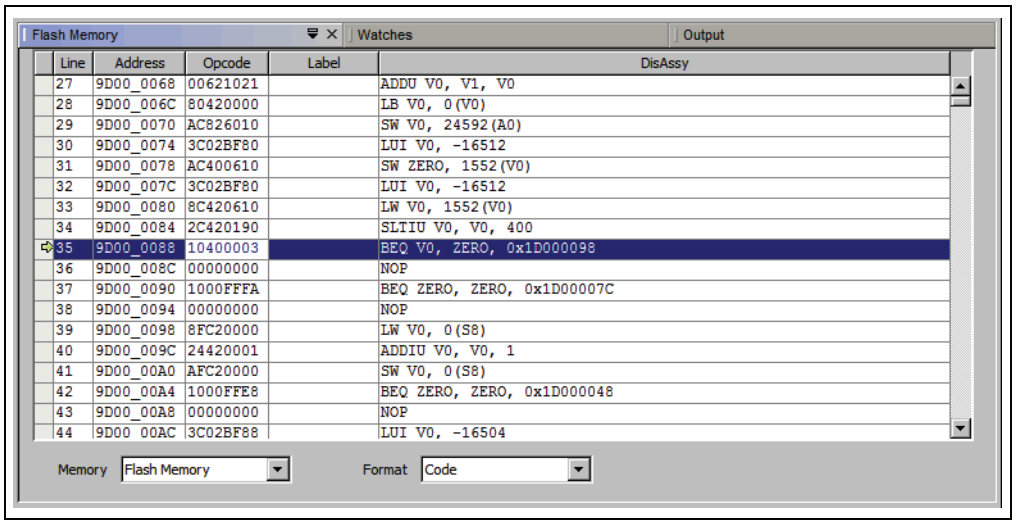
Option	Value	Description
Memory	RAM Memory	All RAM memory on the device
	Flash Memory	All Flash memory on the device
	Peripheral	All Special Function Registers (SFRs)
	Configuration Bits	All Configuration registers
	CPU Memory	All CPU memory
	Memory	All memory
	User ID	User ID memory
Format	Data	Data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

3. After a Debug Run and then Pause, the window will populate with the memory chosen.
4. 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.
5. Use *Debug>Discrete Debugger Operation* to program the target and launch the debugger with the changed data.

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

6. Close the window by clicking the "x" on the memory tab.

FIGURE 4-22: MEMORY WINDOW CONTENT



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 10.3.4.1 "Memory Window Menu"**.

To set Configuration bits:

You must set Configuration bits in code. However, you can temporarily change Configuration bits during debug in the Configuration Bits window. See "To view and change device memory".

You may also export the setting in the Configuration bits window by right clicking in the window and selecting "Generate Source Code to Output". You can then copy the code from the Output window into your code.

For a summary of Configuration bits settings for different devices, see **Chapter 12. "Configuration Settings Summary"**.

4.21 VIEW THE CALL STACK

For 16- and 32-bit devices, a software Call Stack window is available to view `CALL`s and `GOTO`s 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 table of contents for the NetBeans help topic *C/C++/Fortran Development>Debugging C/C++/Fortran Applications with gdb>Using the C/C++/Fortran Call Stack*.

4.22 PROGRAM A DEVICE

Once your code is debugged, you can program it onto a target device. To do this:

- Click **Make and Program Device Project**: The project is built (if necessary) and the device is programmed. The program will immediately begin execution on completion of programming.

Other programming-related functions are:

- **Read Device Memory**: Transfer what is in target memory to MPLAB X IDE.
- **Hold in Reset**: Toggle the device between Reset and Run.
- **Run:** Same as Make and Program Device Project. This icon must be added to the toolbar (*View>Toolbars>Customize*).

FIGURE 4-23: PROGRAM ICONS


Make and Program Device Project Icon


Read Device Memory Icon


Hold In Reset Icon


Run Icon

NOTES:

Chapter 5. Additional Tasks

This chapter provides a guide for performing additional tasks in MPLAB X IDE, as summarized in Performing Additional Tasks.

5.1 PERFORMING ADDITIONAL TASKS

The following table 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. Create Library Projects to build their output as a library.
4. Modify or Create Code Templates to change the default file templates that you use in your project.
5. Switch Hardware or Language Tools used in your project.

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*.)
- For an MPLAB IDE v8 project using the MPLAB C Compiler for PIC24 MCUs and dsPIC DSCs 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.
- When an MPLAB IDE v8 project is imported, if any files have the extensions `.H` or `.C`, they will be renamed with a lower case `.h` and `.c`. The industry standard is uppercase C extensions are for C++ files and lowercase are for C files. Our current compilers are case insensitive for C files and will handle both extensions at this time. However, this will affect you when you check files into your source control since we are renaming the extension of your source code.

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.
- 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 setup.

- **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 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.

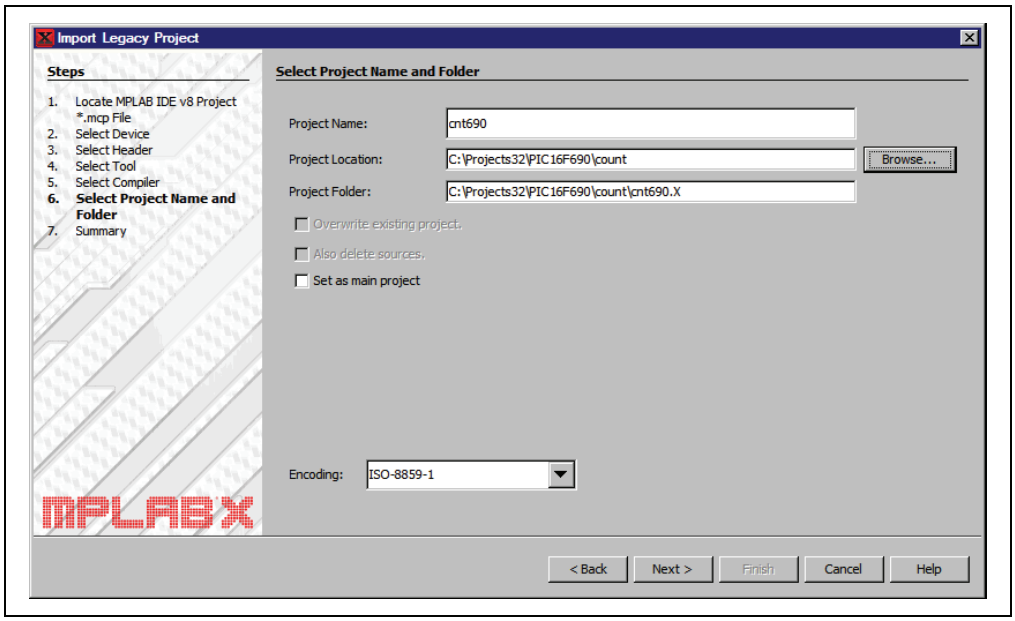
  Main Project:

  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 Project 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. 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.
- 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 setup.

- **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, 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 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.

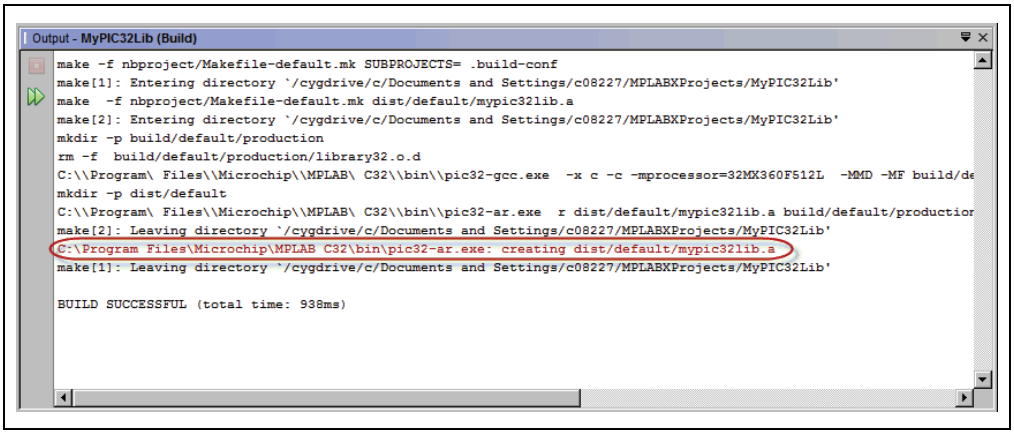
The new project will open in the Project window.

For information on exporting a project as hex, see **Section 10.3.1.2 "Projects Window – Project Menu"**.

5.4 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 5-2: 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 setup. 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 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.

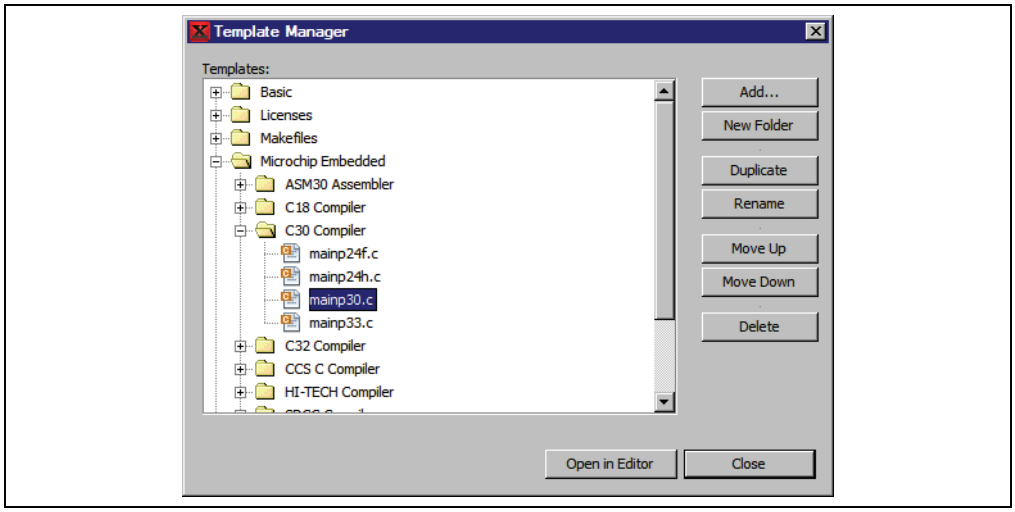
The new project will open in the Project window.

5.5 MODIFY OR CREATE CODE TEMPLATES

When you create a file to add to your project (**Section 4.8 "Create a New Project 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 5-3: TEMPLATE MANAGER



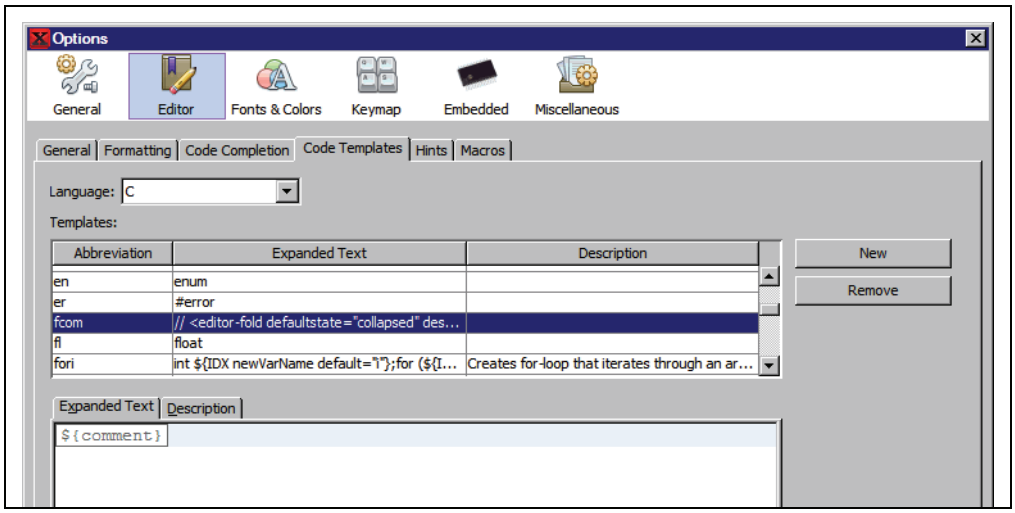
You may set template options by selection *Tools>Options*, **Editor** button, **Code Templates** tab (Figure 5-4).

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.

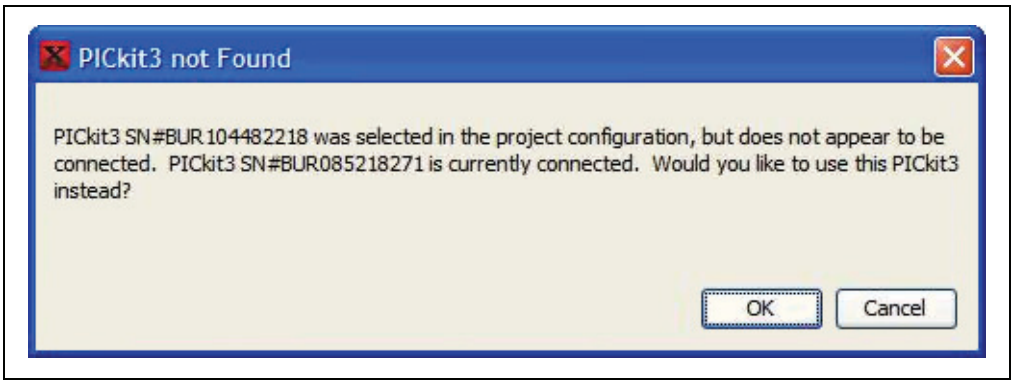
FIGURE 5-4: TEMPLATE OPTIONS



5.6 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 5-5: 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.

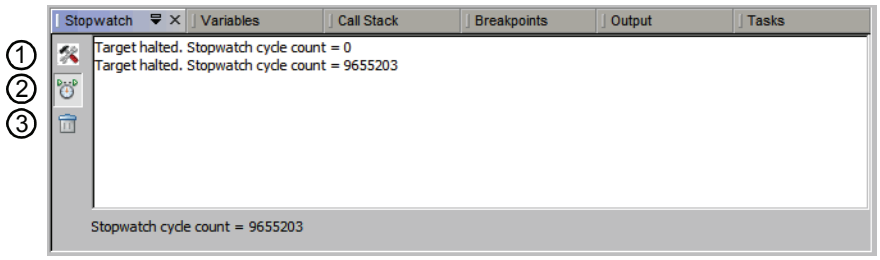
5.7 USE THE STOPWATCH

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 5-6: STOPWATCH WINDOW WITH CONTENT



1. Properties
2. Reset Stopwatch on Run
3. Clear History

5.8 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.

5.9 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:

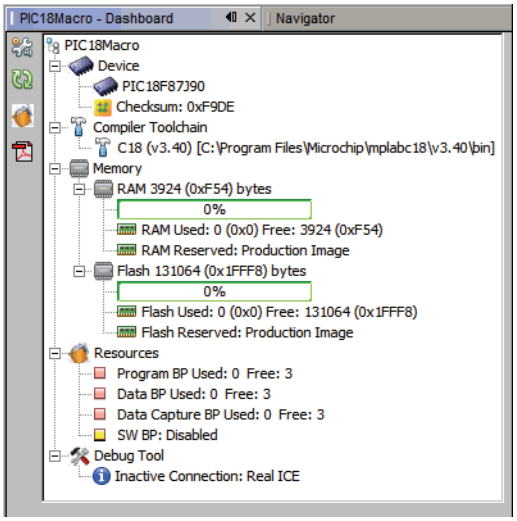
1. Select *Window>Output>Call Graph*.
2. Select the function in the Code window.
3. Right click on the highlighted function and select "Show Call Graph" from the drop-down menu.

For more information, see the table of contents for the NetBeans help topic *C/C++/Fortran Development>C/C++/Fortran Project Basics>Navigating and Editing C/C++/Fortran Source Files>Using the Call Graph*.

5.10 VIEW THE DASHBOARD DISPLAY

Select *Window>Dashboard* to display project information.

FIGURE 5-7: DASHBOARD DISPLAY



The project displayed will either be:

- The active project in the Project window if no main project is selected (*Run>Set Main Project>None*.) Click on a project in the Project 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 5-1: 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 compiler toolchain (compiler, assembler, linker, etc.) used in the project. Includes the path to the executable files.
Memory	The type and amount of memory used by the project, as well as memory reserved for debugging.
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. For 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*, **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.

TABLE 5-2: SIDEBAR ICONS

Icon	Function
	Display the Project Properties dialog.
	Refresh debug tool status. Click this to see hardware debug tool details.
	Toggle software breakpoint. Click to alternately enable or disable software breakpoints.
	Get device data sheet from Microchip web site. 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.

5.11 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 6.4 "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.

For more on the profiler, see the table of contents for the NetBeans help topic *C/C++/Fortran Development>Working with C/C++/Fortran Projects>Profiling C/C++/Fortran Projects*.

5.12 CONTROL SOURCE CODE

MPLAB X IDE has a built-in local file history feature, compliments 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:

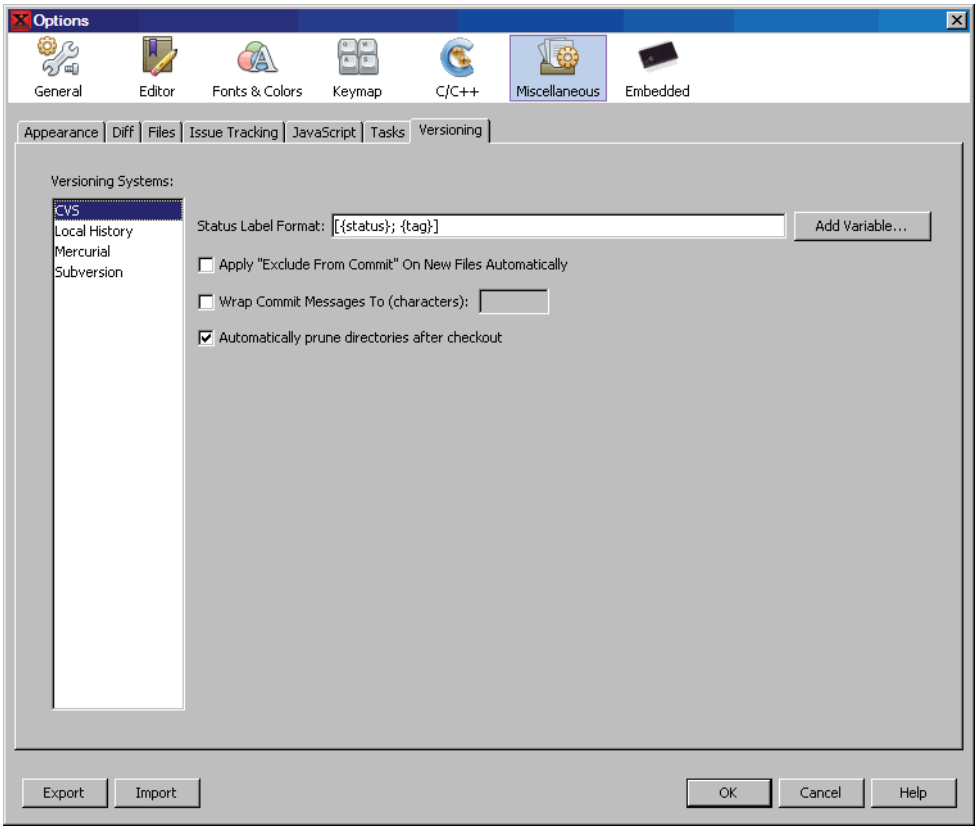
- Right click on the file in the Project or File window and select *Local History>Show Local History*. Any past changes to the file should be listed here.
- Right click on the file in the Project or File window and select *Local History>Revert to*. The Revert to dialog opens with any previous versions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

Source control is available under:

- *Tools>Options*, **Miscellaneous**, **Versioning** – Set up version control programs (see Figure 5-8).
- Team menu – Select version control program submenus.
- *Window>Versioning* – Open version control windows.

FIGURE 5-8: VERSION CONTROL OPTIONS



For more on using local file history and/or source control, see the table of contents for 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/

5.13 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>Team Server* – 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.
- *Windows>Team* – View your project in the Team window.
- *Windows>Chat* – Open the Chat window for team interaction.

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 table of contents for 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/

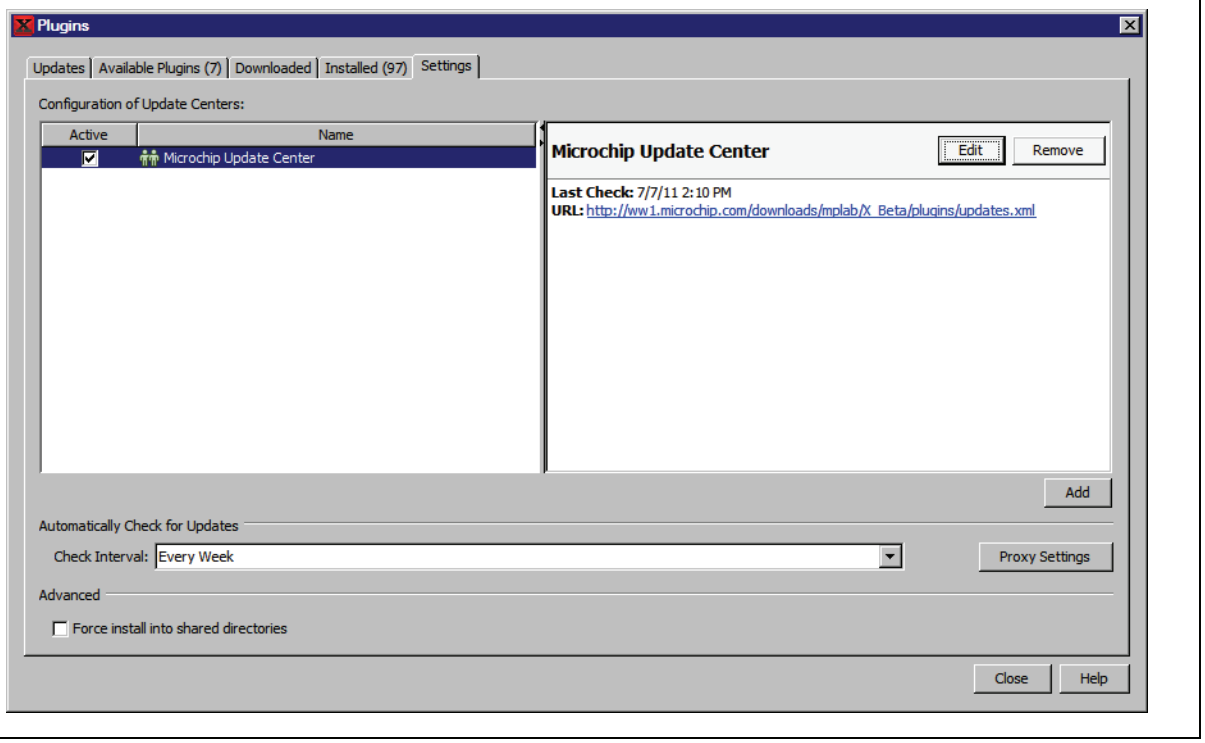
5.14 ADD PLUG-IN TOOLS

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.

To add a Microchip plug-in, you must first configure the Microchip 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 "Microchip Update Center" for the name.
4. Enter the following as the URL:
   http://ww1.microchip.com/downloads/mplab/X_Beta/plugins/updates.xml
5. Click **OK**.

FIGURE 5-9: CONFIGURE MICROCHIP UPDATE CENTER



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

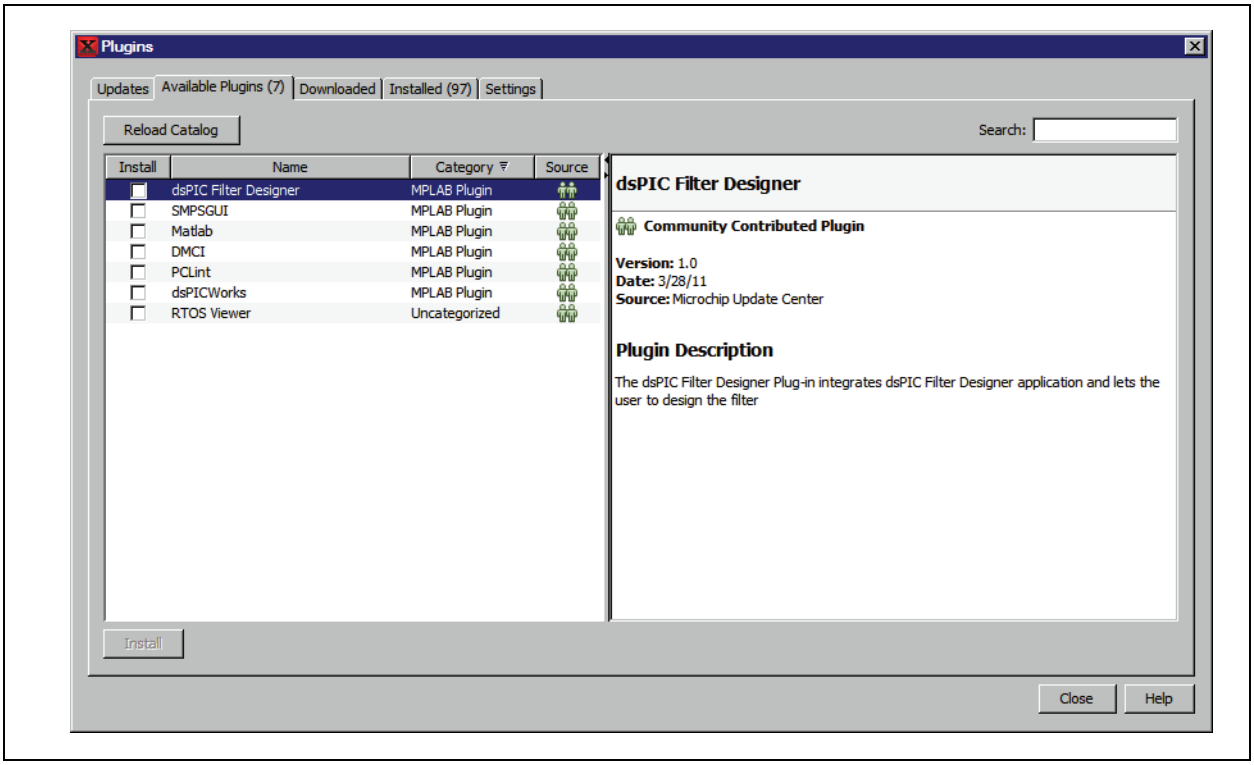
1. Click on the **Available Plugins** tab.
2. Select your plug-in by checking its checkbox.
3. Click **Install**.
4. Follow the on-screen instructions to download and install your plug-in.

> **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 5-10: AVAILABLE MICROCHIP PLUG-IN TOOLS



NOTES:

Chapter 6. Advanced Tasks

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"**.

- Multiple Projects
- Multiple Configurations
- NetBeans™ Editor
- C Code Refactoring

6.1 MULTIPLE PROJECTS

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

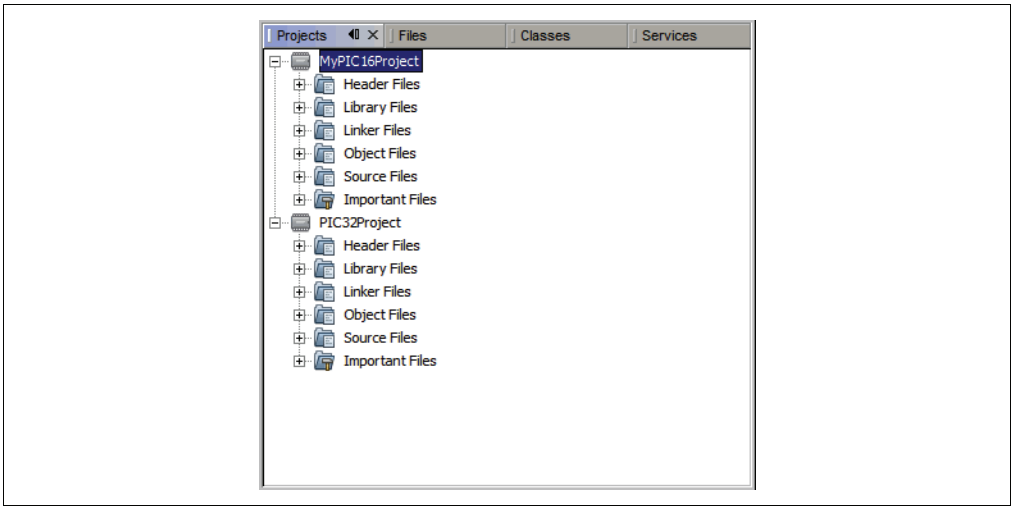
- Work with Multiple Projects – Work on multiple projects with different properties.
- Work with Composite Projects – Work on multiple projects with the same properties.

6.1.1 Work with Multiple Projects

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 10.3.1 "Projects Window"**.

If you have broken down a complex application into multiple projects (the projects are all a part of one application), consider using a composite project (**Section 6.1.2 "Work with Composite Projects"**).

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 may be made 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 project name will then appear as **bold**.

6.1.2 Work with Composite Projects

Creating multiple projects is the best way to develop complex code. In MPLAB IDE v8 you could have several projects open but you had to build, debug and execute them separately. In MPLAB X IDE you can group your projects into a composite project so they may be built, debugged and executed together.

A composite project contains other projects, known as members, and constrains their configurations to the same device and tools. Building a composite project builds its members and the IDE will load these artifacts onto the same device.

6.1.3 To create a composite project:

Use the New Project wizard:

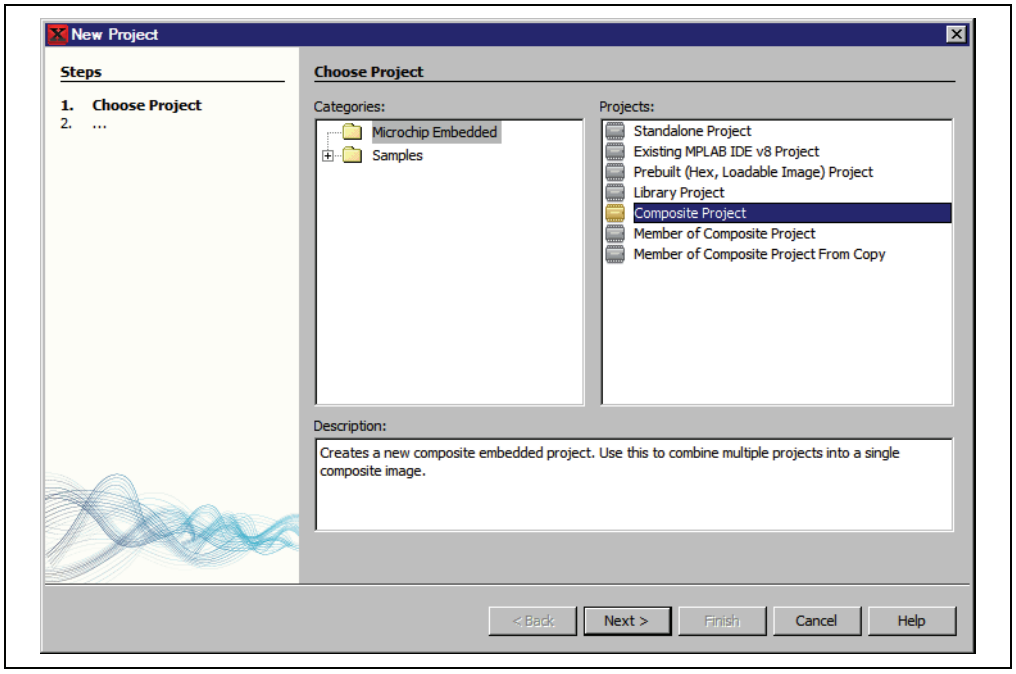
- **Start Page**, "**Learn & Discover**" tab, "Dive In", "Create New Project"
- *File>New Project* (or Ctrl+Shift+N)

The New Project Wizard with launch to guide you through new project setup.

Step 1 will first ask you to choose a project category. This is a NetBeans dialog, so to work with Microchip products you must choose "Microchip Embedded". Secondly, you will choose a project type, in this case "Composite Project".

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

FIGURE 6-2: NEW COMPOSITE PROJECT



For **Step 2** through **Step 5**, select the device, header (if applicable), debug tool and language toolsuite that will be common across all composite project members. If you need more information on these items, refer to **Section 4.2 "Create a New Project"**.

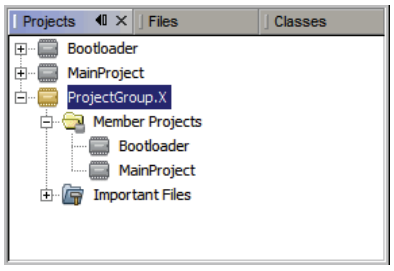
In **Step 6** you will name the composite project and specify a location. Click **Finish** to complete the process.

To add members to a composite project:

Right click on the "Member Projects" folder under the composite project and select on option:

- Add New – Create a new project and add it to the composite project. The project is placed in the composite project folder by default.
- Add Existing – Add an existing project to the composite project. The project is referenced but not moved.
- Add Copy of Exisiting – Add a copy of an existing project to the composite project. The copy of the project is moved under the composite project folder by default.

FIGURE 6-3: COMPOSITE PROJECT EXAMPLE



To build a composite project:

1. Open the composite project if it is not already open and set the suite as the main project (right click on the suite name and select "Set as Main Project").
2. Open all the member projects (double click on each member under "Member Projects to open them). Opening the composite project does not open the member projects.
3. Build the composite project (right click on the project name and select "Build"). All of the member projects will be built and combined into a single executable.

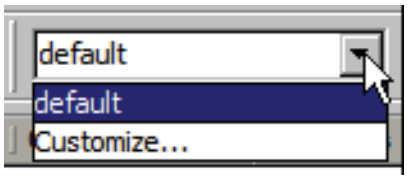
6.2 MULTIPLE CONFIGURATIONS

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 6-4). The Project Properties dialog will open.
- Open the Project Properties dialog by right clicking on the project name and selecting "Properties".

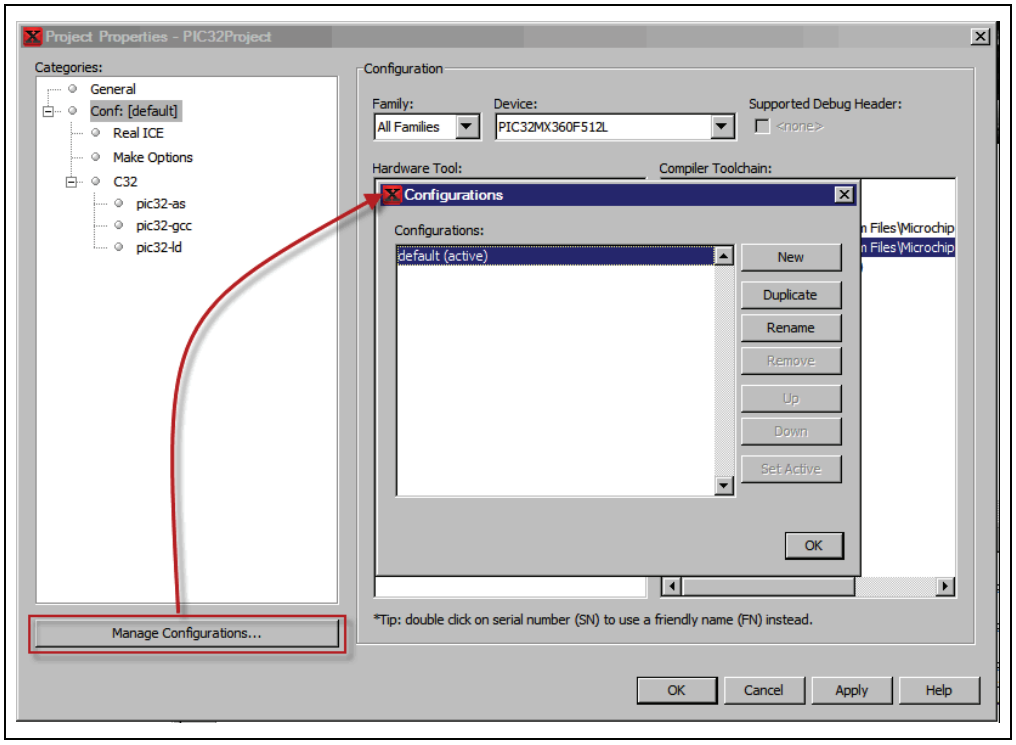
FIGURE 6-4: PROJECT CONFIGURATION DROP-DOWN BOX



In the Project Properties dialog, click **Manage Configurations** to open the Configurations dialog (Figure 6-5). 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.

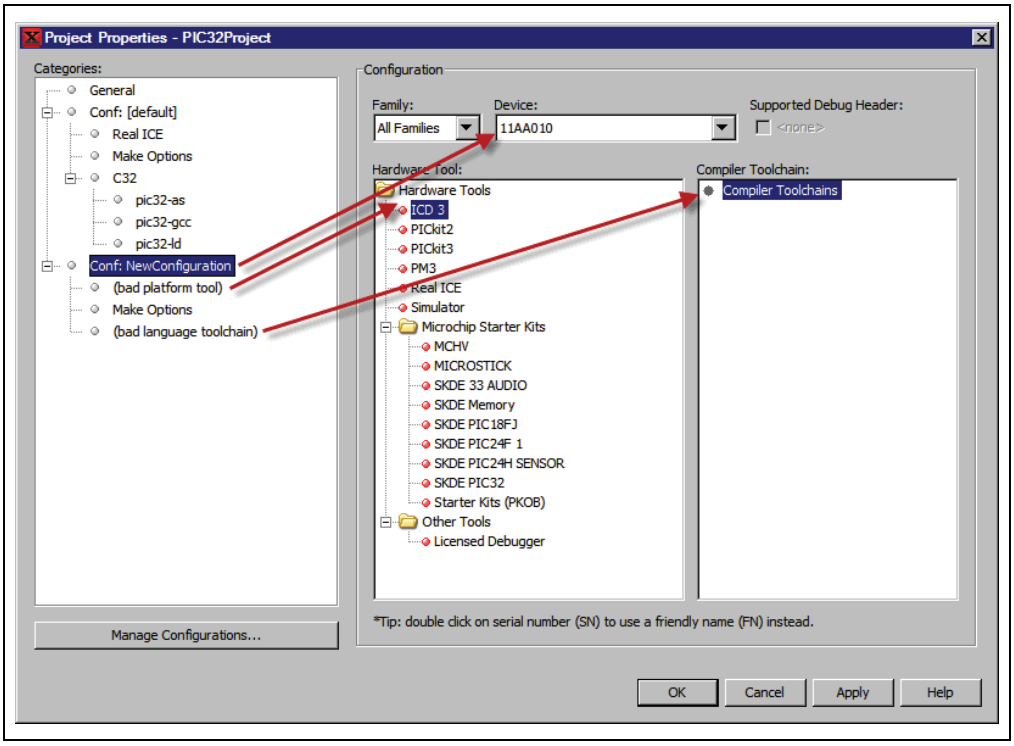
FIGURE 6-5: PROJECT PROPERTIES – CONFIGURATIONS DIALOG



6.2.1 Add a New Configuration

If a new configuration is added, the device and language tools must be assigned in the Project Properties dialog.

FIGURE 6-6: NEW CONFIGURATION



6.2.2 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 debugging.

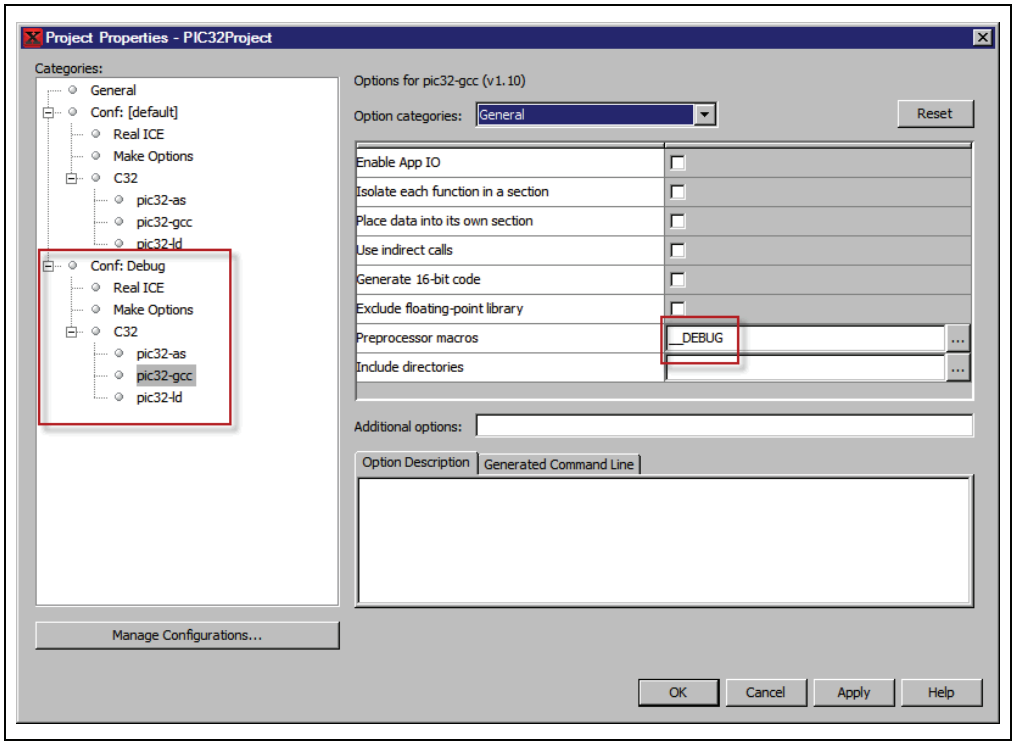
To set up a **debug configuration**:

1. In the Configuration dialog, select a project configuration and click **Duplicate**.
2. Click **Rename** and enter “Debug” in the New Configuration Name dialog.
3. Click **OK** twice to return to the Project Properties dialog. The Debug configuration (Conf: Debug) should now be visible.
4. Click on the compiler or assembler in the toolchain. Under the “General” options category, click on the Preprocessor macro text box.
5. In the Preprocessor Macros dialog, enter `__DEBUG` 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 __DEBUG
  fprintf(stderr,"This is a debugging message\n");
#endif
```

FIGURE 6-7: DEBUG CONFIGURATION



6.3 NETBEANS™ EDITOR

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*.

To use editor features, go to the:

- File menu (see **Section 9.2.1 "File Menu"**) to open a file in an Editor window.
- Edit menu (see **Section 9.2.2 "Edit Menu"**) to use edit commands.
- Editor toolbar located at the top of each file's Editor window (see **Section 9.3.9 "Editor Toolbar"**) to access some edit commands.
- Window right click (context) menu for additional commands.

To set editor properties:

1. Select *Tools>Options* to open the Options dialog.
2. Click on the **Editor** button. Then click on a tab to set up a editor features.
3. Click on the **Fonts and Colors** button. Then click on a tab to set up color options.

Editor features of note:

Editor Feature	Reference
Unicode is supported.	*IDE Basics>Configurating the IDE>About Project Encodings*
Code is colored based on syntax.	*Tools>Options*, **Fonts and Colors** button, **Syntax** tab
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*, **Fonts and Colors** button, **Annotations** tab
Smart code completion makes suggestions and provides hints.	*Tools>Options*, **Editor** button, **Code Completion** tab
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 5.9 "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 5.12 "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 9.2.4 "Navigate Menu"**
Refactor options (rename functions and variables, find all functions, etc.) for improving code structure.	**Section 6.4 "C Code Refactoring"**

6.4 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

6.4.1 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 9.2.6 "Refactor Menu"**).

6.4.2 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.

6.4.3 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 to position the cursor on that 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.

6.4.4 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 contextual 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**.

6.4.5 Moving, Copying and Safely Deleting C Code

These functions are specific to C++ code and are not currently supported.

NOTES:

Chapter 7. Troubleshooting

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, please see "Support" for ways to contact Microchip Technology.

- USB Driver Installation Issues
- Cross-Platform Issues
- MPLAB X IDE Issues
- NetBeans Platform Issues
- Errors
- Forums

7.1 USB DRIVER INSTALLATION ISSUES

To install the correct USB drivers, see **Section 2.2 "Install the USB Device Drivers (For Hardware Tools)"**.

To troubleshoot errors, see **Section 2.2.2.4 "Tool Communication Issues"**.

7.2 CROSS-PLATFORM ISSUES

If you plan on using MPLAB X IDE on different platforms (Windows, Mac, Linux OS), 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`.

7.3 MPLAB X IDE ISSUES

Importing an MPLAB IDE v8 Project

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*.)

7.4 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/69/relnotes.html

7.5 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.

Couldn't reserve space for cygwin's heap (Windows 7 OS)

MPLAB X IDE uses the Cygwin MinGW in its make process. In Windows 7, 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 PC.

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:

http://netbeans.org/bugzilla/show_bug.cgi?id=21236

http://netbeans.org/bugzilla/show_bug.cgi?id=38211

http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=4646857

7.6 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 8. MPLAB X IDE vs. MPLAB IDE v8

MPLAB X IDE differs considerably from MPLAB IDE v8 and before. The topics in this section will help you with migration issues.

- Major Differences
- Feature Differences
- Menu Differences
- Tool Support Differences

8.1 MAJOR DIFFERENCES

1. **MPLAB X IDE is open source and multi-platform**

   MPLAB X IDE is based on the open-source, cross-platform NetBeans IDE. Third parties can easily add functionality as plug-ins.

   MPLAB IDE v8 is proprietary and Windows 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. 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 Project Grouping.

   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 setup information, including multi-project grouping.

3. **MPLAB X IDE allows multiple tool selection**

   MPLAB X IDE allows the selection of multiple tools.

   Example 1: Connect several MPLAB ICD 3 debuggers, with associated target boards, into several PC 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, with an associated target board, and one MPLAB PM3 programmer into one PC USB port each. 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**

   MPLAB X IDE allows the selection of different versions of language tools.

   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.

8.2 FEATURE DIFFERENCES

Because MPLAB X IDE is based on NetBeans and no longer a proprietary application like MPLAB IDE v8 and earlier, many features will be different. Please see the topics under "Getting Started with MPLAB X IDE" to understand the full extent of these changes.

- **Section Chapter 2. "Before You Begin"**
- **Section Chapter 3. "Tutorial"**
- **Section Chapter 5. "Additional Tasks"**

A short list of major feature differences is listed below:

- To run or debug your code, you no longer have to use the build configuration drop-down box or complete three separate steps before you can run your code. Run (*Run>Run Project*) now builds, programs a target device (for hardware tools) and runs your code in one click. Debug Run (*Debug>Debug Project*) 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. See **Section 3.12 "Run Code"** and **Section 3.13 "Debug Run Code"**.
- To program a device you now use either Run (which builds, programs and runs) or Hold in Reset (which builds, programs and holds the target in Reset.) See **Section 3.18 "Program a Device"**.
- More than one build configuration is now possible. Previously, you would select between "Release" and "Debug" from the Build Configuration drop-down box and have use of `__DEBUG` in your own code. Now you can have the same behavior or set up debug configurations as you like. To recreate the MPLAB IDE v8 functionality, you can create your own Debug configuration and `__DEBUG` macro by following the example in **Section 6.2.2 "Add a Duplicate Configuration"**.
- Configuration bits must be set in code. However, you may temporarily change Configuration bits when debugging (see **Section 4.20 "View/Change Device Memory (including Configuration Bits)"**.)
- Memory windows are all now located under *Window>PIC Memory Views*, each of which may be customized for any Microchip device memory type and format. See **Section 3.17 "View Device Memory (including Configuration Bits)"**.
- The breakpoint resources toolbar, checksum toolbar and memory gauge information is now available in one place in the *Window>Dashboard* window. See **Section 5.10 "View the Dashboard Display"**.
- Many additional NetBeans editing and debug features. See NetBeans Help for more details.

8.3 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 table of contents for the NetBeans help topic “IDE Basics” for details.

TABLE 8-1: 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	N/A	All data saved in projects
Save Workspace	N/A	
Save Workspace As	N/A	
Close Workspace	N/A	
Import	File>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	N/A	All data saved in projects
Exit	File>Exit	

TABLE 8-2: 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	

TABLE 8-2: EDIT MENU DIFFERENCES (CONTINUED)

MPLAB® IDE v8	MPLAB® X IDE	Comments
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, Miscella-neous, 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	

TABLE 8-3: 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>Disassem-bly 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>Vari-ables	
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	

TABLE 8-3: VIEW MENU DIFFERENCES (CONTINUED)

MPLAB® IDE v8	MPLAB® X IDE	Comments
Special Function Registers	Window>PIC Memory Views>SFRs	
Watch	Window>Debug-ging>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 8-4: 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	N/A	A project is required for all development.
Package in .zip	Right click on project in Proj-ect tabbed window, select ‘Package’	
Clean	Right click on project in Project tabbed window, select ‘Clean and Build’	
Locate Headers	N/A	
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 Configura-tion	
Build Options	File>Project Properties	Language tool setup
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	

TABLE 8-4: PROJECT MENU DIFFERENCES (CONTINUED)

MPLAB® IDE v8	MPLAB® X IDE	Comments
Version Control	Tools>Options, Miscellaneous, Versioning tab	See also: Team menu, Window>Versioning

TABLE 8-5: DEBUGGER MENU DIFFERENCES

MPLAB® IDE v8	MPLAB® X IDE	Comments
Select Tool	File>New Project File>Project Properties	To select To change
Clear Memory	TBD	Used by Programmers
Run	Run>Run Project Debug>Debug Project Debug>Continue	Programmer Run Debug Run Run from Halt
Animate	N/A	
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 8-6: PROGRAMMER MENU DIFFERENCES

MPLAB® IDE v8	MPLAB® X IDE	Comments
Select Programmer	File>New Project File>Project Properties	To select To change
Enable Programmer	N/A	Once selected, it is enabled.
Disable Programmer		
Program	Program Target Project	On Run toolbar
Verify	Not yet available.	
Read	Upload Target Project	On Run toolbar
Blank Check All	Not yet available.	
Blank Check OTP	Not yet available.	
Erase Flash Device	Not yet available.	
Reset Program Statistics	Not yet available.	
Download OS	Not yet available.	
Settings	File>Project Properties	

TABLE 8-7: 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 8-8: 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	

TABLE 8-9: 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 8-10: HELP MENU DIFFERENCES

MPLAB® IDE v8	MPLAB® X IDE	Comments
Topics	Help>Contents	To view PDFs, see **Section 11.1 "Projects Window View"**.
Release Notes	Help>Online Docs and Support	
Driver Installation	Help>Online Docs and Support	
Check for Updates	Help>Check for Updates	
Web Links	Help>Online Docs and Support	
About MPLAB® IDE	Help>About	

8.4 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 8-11: TOOL SUPPORT DIFFERENCES

Development Tool	In MPLAB® X IDE?	
	Yes	No
MPLAB® ICD 2		X
MPLAB® ICD 3	X	
MPLAB® ICE 2000		X
MPLAB® ICE 4000		X
MPLAB® REAL ICE™	X	
PICkit™ 1		X
PICkit™ 2	X	
PICkit™ 3	X	
MPLAB® PM3	X	
PRO MATE II		X
PICSTART® Plus		X
MPLAB® VDI		X
Plug-Ins		
AN851Bootloader		X
AN901/908		X
DMCI	X	
dsPIC® Filter Design	X	
MATLAB®	X	
PC-Lint	X	
dsPIC30F SMPS Buck Converter	X	
dsPIC33F SMPS Buck/Boost Converter	X	
Segmented Display Designer*	X	
Graphical Display Designer*	X	
Memory Starter Kit	X	
KEELOQ® Plugin		X
* Under development		

NOTES:

Chapter 9. Desktop Reference

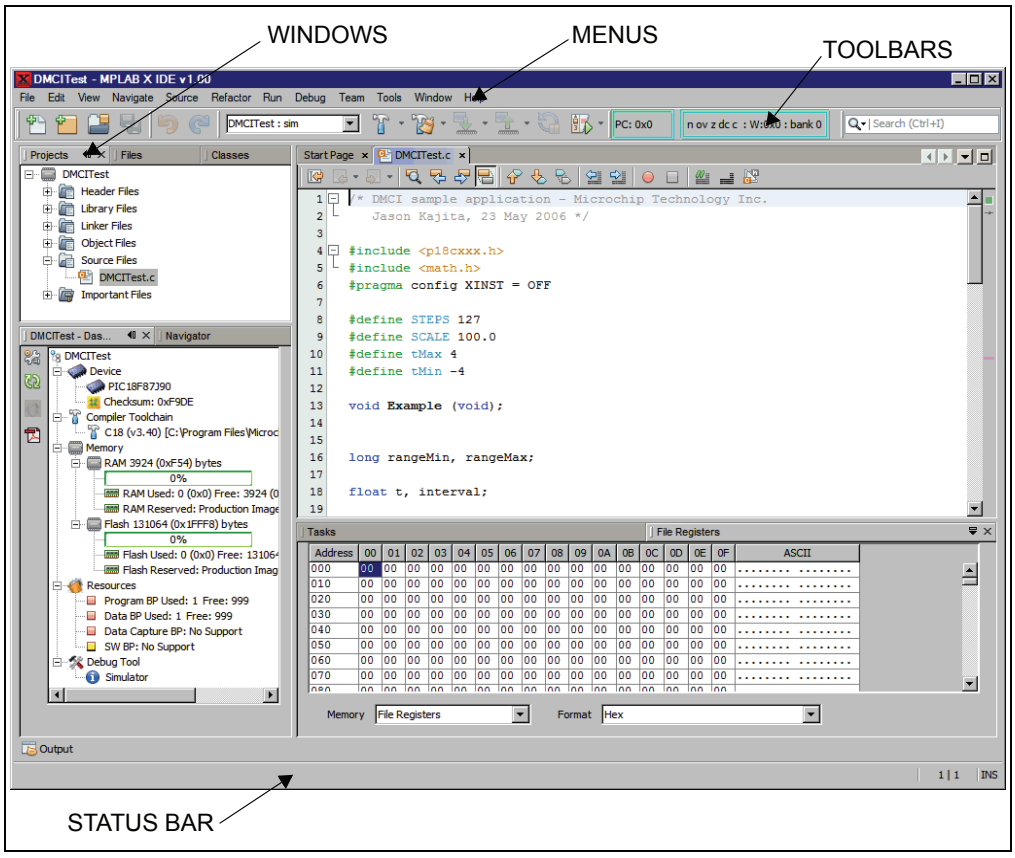
9.1 INTRODUCTION

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 9-1). 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 9-1: MPLAB IDE DESKTOP



9.2 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 10. "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 10.3.1 "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

9.2.1 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 9-1: 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	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 on team server projects, see NetBeans help, "IDE Basics>Collaborative Development".
Close Project (Name)	Closes the current project.
Close All Projects	Closes all open projects.
Open File	Opens an existing file.

TABLE 9-1: FILE MENU OPTIONS (CONTINUED)

Command	Action
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 compiler/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.

9.2.2 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 9-2: 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.

9.2.3 View Menu

Below are the menu items in the View menu. For keyboard shortcuts of some of these menu items, see the table of contents for the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 9-3: 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 Task 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.

9.2.4 Navigate Menu

Below are the menu items in the Navigate menu. For keyboard shortcuts of some of these menu items, see the table of contents for the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 9-4: 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 9-4: 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 backwards 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.

9.2.5 Source Menu

Below are the menu items in the Source menu. For keyboard shortcuts of some of these menu items, see the table of contents for the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 9-5: 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.

9.2.6 Refactor Menu

Below are the menu items in the Refactor menu. The items you see are dependant on the type of object (variable, function, etc.) you are refactoring. For more information, see **Section 6.4 "C Code Refactoring"**.

For keyboard shortcuts of some of these menu items, see the table of contents for the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 9-6: 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(1)	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(1)	Copies a class to the same or a different package.
Safely Delete(1)	Checks for references to a code element and then automatically deletes that element if no other code references it.
Undo	Undos refactoring.
Redo	Redos refactoring.
(1) Java and C++ operation (not currently supported)	

9.2.7 Run Menu

Below are the menu items in the Run menu. For keyboard shortcuts of some of these menu items, see the table of contents for the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 9-7: RUN MENU OPTIONS

Command	Action
Run Project	Runs the main or selected project.
Test Project	Starts JUnit test for project. (Java related)
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.

9.2.8 Debug Menu

Below are the menu items in the Debug menu. For keyboard shortcuts of some of these menu items, see the table of contents for the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 9-8: 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	Perform debug operations one step at a time (discretely): Build, Program Target, Launch Debugger. This is useful for changing 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 until the next breakpoint or the end of the program is reached.
Animate	Automatically execute single steps while running.
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, executes the program up to the function’s first statement and then stops.
Step Instruction	Executes one source line of a program. If the line is a function call, 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 processor.
Set PC at cursor	Sets the program counter (PC) value to the line address of the cursor.
Focus Cursor at PC	Move the cursor to the current PC address.
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 variable to watch.
New Run Time Watch	Adds the specified variable to watch that will change value as the program runs/executes.
Disconnect from Debug Tool	Select to disconnect communications between MPLAB® X IDE and the debug tool. To reconnect, select Run/Debug Run.
Run Debugger/Programmer 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.

9.2.9 Team Menu

Below are the menu items in the Team menu. For keyboard shortcuts of some of these menu items, see the table of contents for the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 9-9: TEAM MENU OPTIONS

Command	Action
Team Server	Creates a team project. For more on team projects, see NetBeans help, "IDE Basics>Collaborative Development".
CVS, Mercurial, Subversion	Displays submenus 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.

9.2.10 Tools Menu

Below are the menu items in the Tools menu. For keyboard shortcuts of some of these menu items, see the table of contents for the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 9-10: TOOLS MENU OPTIONS

Command	Action
Embedded	Visible if a plug-in has been added.
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 Templates 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.

9.2.11 Window Menu

Below are the menu items in the Window menu. For keyboard shortcuts of some of these menu items, see the table of contents for the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 9-11: 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.
Team	Opens the Team window. See **Section 5.13 "Collaborate on Code Development and Error Tracking"**.
Tasks	Opens the Task List window.

TABLE 9-11: WINDOW MENU OPTIONS (CONTINUED)

Command	Action
Dashboard	Open the Dashboard window. See **Section 5.10 "View the Dashboard Display"**.
Palette	Opens the Palette window. (Java related)
Properties	Opens the Properties window containing property information on the currently-selected file.
Chat	Opens a Team Chat window.
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>Test Results	Opens the unit Test Results window for C projects.
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>Threads	Opens the Threads window.
Debugging>Sources	Opens the Sources window.
Debugging>Disassembly	Opens the Disassembly window.
Debugging>Trace	Opens the Trace window. To operate, the tool and device must support trace.
Debugging>PIC App IO	Opens the Application In/Out window. To operate, the tool and device must support App IO.
Debugging>Stop Watch	Opens the Stop Watch window.
Versioning>CVS	Select CVS version control items.
Versioning>Subversion	Select Subversion version control items.
Versioning>Mercurial	Select Mercurial version control items.
PIC Memory Views>*Memory*	Opens specified Memory window. Memories shown depend on the project device.
Simulator>Stimulus	Opens the Stimulus window for the simulator.
Other>Exception Reporter	Opens the Exception Reporter window for exception break-points.
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.
Processes	Opens the Processes window for a running C process.
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.

TABLE 9-11: WINDOW MENU OPTIONS (CONTINUED)

Command	Action
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.

9.2.12 Help Menu

Below are the menu items in the Help menu. For keyboard shortcuts of some of these menu items, see the table of contents for the NetBeans help topic *IDE Basics>Keyboard Shortcuts>Menu Shortcuts*.

TABLE 9-12: 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.
Check for Updates	Searches for updates to the MPLAB® X IDE program.
Start Page	Opens or moves the Start Page tab in front of any other open tabs.
About	Displays a window about MPLAB IDE.

9.3 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.

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 toolsbars.
- Select *View>Toolbars* to show/hide a toolbar, change the contents of some toolbars or create a custom toolbar.

9.3.1 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.

9.3.2 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.

9.3.3 Status Flags Toolbar

The Status Flags Toolbar contains:

- PC – Program Counter (PC) current value.

9.3.4 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.

9.3.5 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. Should be "default".
- Build Project – Builds all the project files.
- Clean and Build Project – Deletes files from previous builds and then builds the all project files.
- Make and Program Device Project – Builds, programs the target and Runs the selected project.
- 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.
- Debug Project – Builds, programs the target and Debug Runs the selected project.

9.3.6 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.
- Animate – Automatically executes single steps while running.
- 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.

9.3.7 Memory Toolbar

The Memory Toolbar displays the current PC memory usage for MPLAB IDE. Click on the display to force garbage collection.

9.3.8 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.

9.3.9 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.

9.4 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.

9.5 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.

NOTES:

Chapter 10. Windows and Dialogs

10.1 INTRODUCTION

The MPLAB X IDE desktop is divided into panes containing tabbed windows. Not all of these are visible until a feature is 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, Files and Services in the top left pane, Navigation in the bottom left pane, and Output in the bottom right pane. The **Start Page** window moved into the top right pane.

MPLAB IDE windows are a combination of basic NetBeans windows and MPLAB IDE specific windows.

- NetBeans Specific Windows and Window Menus
- MPLAB X IDE Specific Windows and Window Menus

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 Specific Dialogs
- MPLAB X IDE Specific Dialogs

10.2 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 on the help file's **Search** tab to search for information on that window. Or see the table of contents for the NetBeans help topic "Managing IDE Windows".

To open most windows, see the **Section 9.2.11 "Window Menu"**.

Windows may be docked and undocked (right click on **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 9.2 "Menus"**).

Set up window properties by selecting *Tools>Options*, **Miscellaneous** Button, **Appearance** tab.

10.3 MPLAB X IDE SPECIFIC WINDOWS AND WINDOW MENUS

MPLAB X IDE uses many NetBeans windows (see **Section 10.2 "NetBeans Specific Windows and Window Menus"**). However, some windows and their related menus are modified or created specifically for embedded use.

- Projects Window
- Project Properties Window
- Dashboard Window
- Memory Windows
- Options Window
- Output Window

10.3.1 Projects Window

The Projects window is a NetBeans window. However, it has been customized to show the relevant virtual folders for an MPLAB X IDE project. Also, the context (right click) menus have been customized with some MPLAB X IDE specific items.

- Projects Window – Logical Folders
- Projects Window – Project Menu
- Projects Window – File Menu

10.3.1.1 PROJECTS WINDOW – LOGICAL FOLDERS

Additional logical folders have been added to apply to MPLAB X projects. Therefore, all folders displayed are:

- **Header Files** – MPLAB X IDE does not use this category when building. Consider it 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 Project window to open the file in an editor.
- **Library Files** – 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.
- **Linker Files** – You do not need to add a linker script to your project. The project language tools will find the appropriate generic linker script for your device. Only if you need to create your own linker script should there be a file here.
  There should be only one file in this folder. If you have more than one linker script, only the first one has any effect. This is the linker script that the tool will use in the link step.
- **Object Files** – This category is for precompiled object files, not object files that result from assembling or compiling the project's source files. Functionally, the files in this category are indistinct from the library files, because both are merely linked in at the final phase of the build.
- **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 one.
  You can add simulator files – SBS, STC, SCL – and double click on them to open the SCL Generator, Stimulus Controller, or editor, respectively.
  You can add project-specific data sheets (PDFs) to this location in the Project window. Then, you can double click on the PDF to launch the data sheet. (This requires that a PDF reader is installed.)

10.3.1.2 PROJECTS WINDOW – PROJECT MENU

Right click on a project name in the Projects window to pop up the Project menu (Figure 10-1). MPLAB X IDE specific menu items are shown in Table 10-1.

FIGURE 10-1: PROJECT CONTEXT MENU

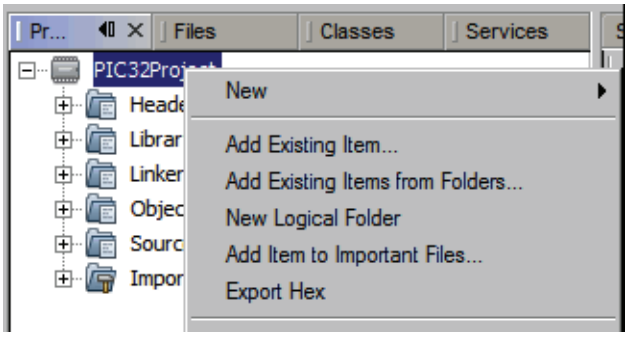


TABLE 10-1: PROJECT CONTEXT MENU ITEMS

Menu Item	Description
New	Add a new item. For MPLAB® X IDE, embedded file types are available.
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.
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.
Hold in Reset	Program the target device and hold in Reset (do not run).
Upload Target Memory	Read from the target device.
Properties	Set project properties. For MPLAB X IDE, the Project Properties window is specific to embedded development. See **Section 10.3.2 "Project Properties Window"**.

10.3.1.3 PROJECTS WINDOW – FILE MENU

Right click on a file name in the Projects window to pop up the File menu. Table 10-2 shows MPLAB X IDE specific menu items.

TABLE 10-2: FILE CONTEXT MENU ITEMS

Menu Item	Description
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.

10.3.2 Project Properties Window

This window is used to view or change the project device, tools and tool settings. For more information, see:

- **Section 3.5 "View or Make Changes to Project Properties"**
- **Section 3.6 "Set Options for Debugger, Programmer or Language Tools"**
- **Section 4.13 "Set Build Properties"**

10.3.3 Dashboard Window

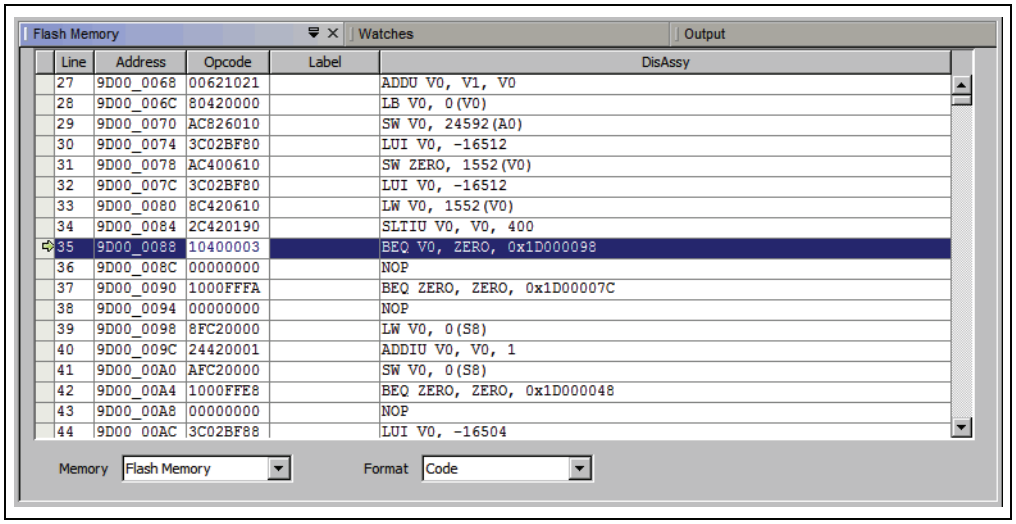
The Dashboard window contains general project information, such as checksums, memory usage and breakpoint resources. See **Section 5.10 "View the Dashboard Display"** for details.

10.3.4 Memory Windows

Memory windows, called 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 4.20 "View/Change Device Memory (including Configuration Bits)"**.

FIGURE 10-2: MEMORY WINDOW CONTENT



10.3.4.1 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 10-3: 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 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.
Center Debug Location	Center the current PC location in the window.
Symbolic Mode	Display disassembled hex code with symbols.
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	Alter spacing of rows in 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 export is to be single column.
Print	Print the contents of this window.
Adjust Table Columns	Select which columns are visible.
Refresh	Refresh the data in this window.

Right clicking on the **Window** tab will display other options, such as Close, Maximize/Minimize window, and Dock/Undock window.

10.3.5 Options Window

The Options window (*Tools>Options*) 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
- Other tab

10.3.5.1 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 3.7 "Set Language Tool Locations"**.

Ensure that you have INSTALLED THE LANGUAGE TOOL or it will not show up on "Tool Collection" 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 toolsuite – includes MPASM assembler, MPLINK linker and utilities.
- ASM30 toolsuite – includes the 16-bit assembler, linker and utilities.
- C32 toolsuite (free version) – includes the 32-bit compiler, assembler, linker and utilities.

Other tools may be obtained from the Microchip web site (www.microchip.com) or third parties.

TABLE 10-4: BUILD TOOLS TAB ITEMS

Item	Description
Tool Collection	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 compilers**.
Duplicate	Duplicate an existing language tool list item and then rename or make other changes.
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 compilers	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.

10.3.5.2 PROJECT OPTIONS TAB

Set options related to the project.

TABLE 10-5: 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 Options	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.
Reuse Output Tabs from Finished Processes	If selected, writes `make` output to a single Output window tab, deleting the output from the previous process. If not selected, opens a new tab for each make process.
Enable dependency checking in generated makefiles (requires support from toolchain)	Add `make state` statements to the makefile. Refer to your language tool documentation about dependency checking support.
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*. For more on the profiler, see the table of contents for the NetBeans help topic *C/C++/Fortran Development>Working with C/C++/Fortran Projects>Profiling C/C++/Fortran Projects*.

10.3.5.3 GENERIC SETTINGS TAB

Set up the log file and other project features.

TABLE 10-6: GENERIC SETTINGS TAB ITEMS

Item	Description
Logging Level	Set the message logging level. **OFF:** No logging **SEVERE:** Log severe 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 location of log file.
Projects Folder	Path and location of the folder where you will place your MPLAB® X IDE projects.
Maintain active connection to hardware tool	If selected, keep hardware tool connected always, not just at runtime. (MPLAB® IDE v8 behavior)
Silent build	Build without generating messages in the Output window.
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.

10.3.5.4 OTHER TAB

Edit the lists of accepted file extensions for C and assembler source files and header files. Also, set the default extension for each type.

Note: At this time, MPLAB X IDE does not support C++ or Fortran programming.

10.3.6 Output Window

The Task pane contains many windows, some inherited from NetBeans and some MPLAB X IDE specific. The Output window contains the MPLAB X IDE output information, shown on tabs within the window.

TABLE 10-7: 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.

10.4 NETBEANS SPECIFIC DIALOGS

NetBeans dialogs are discussed in NetBeans help. To open help on a dialog, click on the Help button on the dialog or, if there is no Help button, hit the <F1> key. If no help can be found, select *Help>Help Contents* and click on the help file's **Search** tab to search for information on that dialog.

To open most dialogs, see **Section 9.2 "Menus"**.

10.5 MPLAB X IDE SPECIFIC DIALOGS

MPLAB X IDE uses many NetBeans dialogs. However, some dialogs are modified or created specifically for embedded use.

- New Project wizard – see **Section 3.3 "Create a New Project"**
- Project Properties – see **Section 3.5 "View or Make Changes to Project Properties"**
- File Properties – see **Section 4.12 "Set File Properties"**
- Hardware Tool Setup – see **Section 3.6 "Set Options for Debugger, Programmer or Language Tools"**
- Language Tool Setup – See **Section 3.6 "Set Options for Debugger, Programmer or Language Tools"** and **Section 3.7 "Set Language Tool Locations"**
- Convert from an MPLAB IDE v8 project to an MPLAB X IDE project – see **Section 5.2 "Import MPLAB Legacy Project"**

NOTES:

Chapter 11. Project Files and Folders

There are two ways to view project files and folders in MPLAB X IDE:

- Projects Window View
- Files Window View

Other MPLAB X IDE file and folder information includes:

- Importing an MPLAB IDE v8 Project – Relative Paths
- Moving a Project
- Building a Project Outside of MPLAB X IDE

11.1 PROJECTS WINDOW VIEW

Project folders viewed in the Projects window are logical (virtual) folders. For more on this window, see the table of contents for the NetBeans help topic "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 11-1: PROJECTS WINDOW – SIMPLE C CODE PROJECT

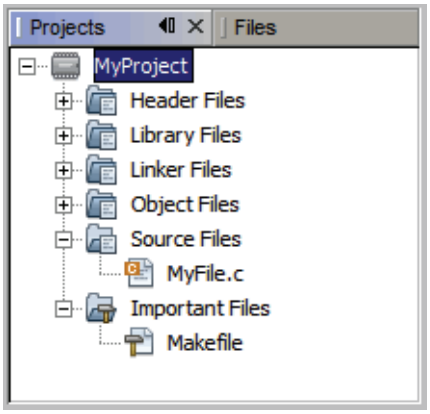


TABLE 11-1: PROJECTS WINDOW DEFINITIONS

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.

11.2 FILES WINDOW VIEW

Project folders viewed in the Files window are actual folders (folders on your PC). For more on this window, see the table of contents for the NetBeans help topic "Files Window" as used with C code projects.

FIGURE 11-2: FILES WINDOW – PROJECT FOLDERS

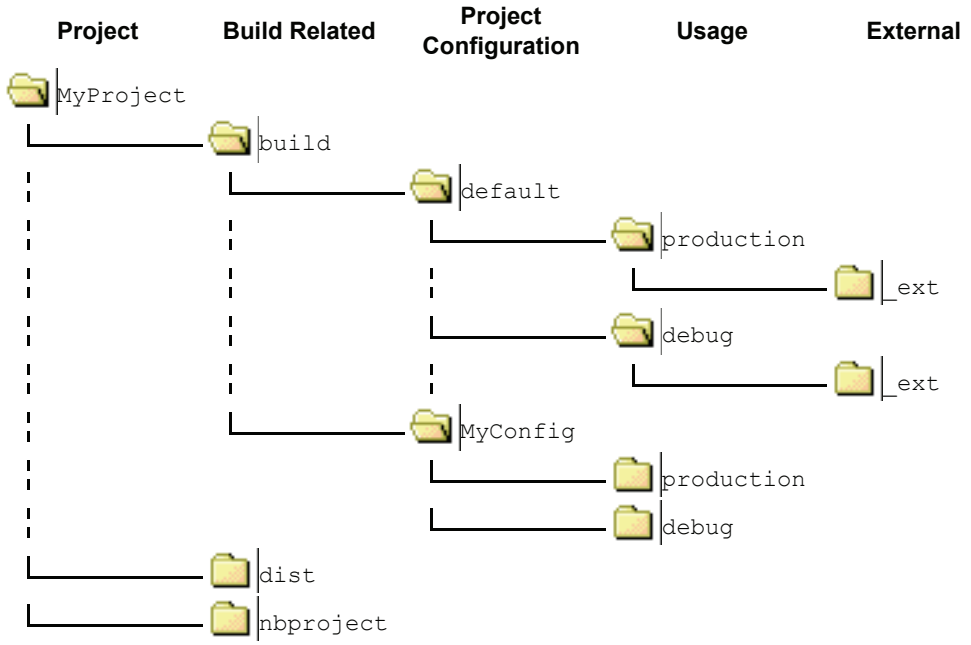


TABLE 11-2: FILES WINDOW DEFINITIONS

Folder	Description
MyProject	The project folder. Contains these files: • Makefile – The master makefile for the project. • C code or assembly application files.
build*	The intermediate files folder. Files are contained in subfolders depending on project configuration, usage and location. The build files are: • Run files (.o) • Debug Run files (.o.d) • HI-TECH® intermediate files (.p1)
dist*	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	The project configurations folders. If no configurations are created by the developer, all code is in default.
production, debug	The production and debug versions folders.
_ext	The external project files folder. If a file is referenced outside the project folder, it is listed here.
*Do not check these folders into source control. A build will create them.	

11.3 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.

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"**.

11.4 MOVING A PROJECT

If you want to move your project, see the table of contents for the NetBeans help topic *C/C++/Fortran Development>Working with C/C++/Fortran Projects>Moving, Copying, or Renaming a C/C++/Fortran Project.*

You can also use an external tool; the project file structure does not require you to use MPLAB X IDE.

11.5 BUILDING A PROJECT OUTSIDE OF MPLAB X IDE

MPLAB X IDE uses makefiles to build its projects. You can build your project outside of the IDE by opening a terminal (command line), navigating to the project folder and running the make utility. Before you do that, you need to tell your operating system the path to the make utility.This is done differently in each of the operating systems supported by MPLAB IDE X.

11.5.1 Windows

When you install MPLAB X IDE, the following directory will be created:

```
C:\Program Files\Microchip\MPLABX\gnuBins\GnuWin32\bin
```

which contains the make and other utilities needed to build a project.

Make sure the path above is the very first thing in your `%PATH%` environment variable:

```
c:\>set PATH=C:\Program Files\Microchip\MPLABX\gnuBins\GnuWin32\bin;
%PATH%
```

For 64 bit systems:

```
c:\>set PATH=C:\Program Files (x86)\Microchip\MPLABX\gnuBins\
GnuWin32\bin;%PATH%
```

The paths above are one-line and wrap because of the page margins.

11.5.2 MacOSX

A Mac out-of-box contains all the tools needed except the GNU make executable. When you install MPLAB X IDE, the following directory will be created:

```
/Applications/microchip/mplabx/mplab_ide.app/Contents/Resources/
mplab_ide/bin
```

This directory contains the make executable. You can add the directory where make is to your path:

```
$export PATH=/Applications/microchip/mplabx/mplab_ide.app/Contents/
Resources/mplab_ide/bin:$PATH
```

The directory and path above are one-line and wrap because of the page margins.

11.5.3 Linux

A normal Linux distro will have the tools MPLAB X IDE needs already installed except the GNU make. Use the package manager for the distro to install the GNU make.

Test that make using:

```
$ which make
/usr/bin/make
```

After setting up the path, you need to change the directory to the MPLAB IDE X project folder. Then you can run:

```
$ make -f nbproject/Makefile-$CONFIGURATION_NAME.mk SUBPROJECTS=
.build-conf
```

Where `$CONFIGURATION_NAME` is the name of the configuration in the project you want to build. MPLAB X IDE always creates a default configuration. You can add others. For the default configuration, the command would be:

```
$ make -f nbproject/Makefile-default.mk SUBPROJECTS= .build-conf
```

If you have a single configuration (default), you can also simply type:

```
$ make
```

To clean (remove the intermediate and final files in the build) a project you type:

```
$ make -f nbproject/$CONFIGURATION_NAME.mk SUBPROJECTS= .clean-conf
```

Again, `$CONFIGURATION_NAME` is the name of the configuration in your project you want to clean:

```
$ make -f nbproject/Makefile-default.mk SUBPROJECTS= .clean-conf
```

This will clean the default configuration. If you have a single configuration you can simply type:

```
$ make clean
```

Chapter 12. Configuration Settings Summary

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.

- MPASMWIN Toolchain
- HI-TECH® PICC™ Toolchain
- HI-TECH® PICC-18™ Toolchain
- C18 Toolchain
- ASM30 Toolchain
- C30 Toolchain
- C32 Toolchain

12.1 MPASMWIN 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.

12.1.1 __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 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.

See also: `__config`

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
```

12.1.2 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.

See also: `config` – Set Processor Configuration Bits (PIC18 MCUs)

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
```

12.2 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.

See also: Configuration Fuses

PICC Example

```
#include <htc.h>

__CONFIG(WDTDIS & XT & UNPROTECT); // Program config. word 1
__CONFIG(FCMEN);                   // Program config. word 2
```

12.3 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.

See also: Configuration Fuses

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);
```

12.4 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.

See also: `#pragma config`

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
```

12.5 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, i.e., PIC24F, PIC24H, dsPIC30F, or dsPIC33F.

Macro case should match what is in the relevant header. For example, `config` is correct but `CONFIG` is not.

See also:

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
```

12.6 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.

12.6.1 PIC24F Configuration Settings

Macros are provided in device header files to set Configuration bits:

```
_confign(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 )
```

12.6.2 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.

See also: Configuration Bits Setup Macros

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);
```

12.7 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.

See also: Configuration Bit Access

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
```

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
- About Microchip Technology

WARRANTY REGISTRATION

If your development tool package includes a Warranty Registration Card, please complete the card and mail it in promptly. Sending in your Warranty Registration Card entitles you to receive new product updates. Interim software releases are available at the Microchip web site.

myMICROCHIP PERSONALIZED NOTIFICATION SERVICE

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 http://www.microchip.com/pcn 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® 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

Microchip provides online support via our web site at http://www.microchip.com. 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

Microchip provides additional online support via our web forums at http://www.microchip.com/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

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document. See our web site for a complete, up-to-date listing of sales offices.

Technical support is available through the web site at http://support.microchip.com.

Documentation errors or comments may be emailed to docerrors@microchip.com.

ABOUT MICROCHIP TECHNOLOGY

Microchip Technology Inc. is a leading provider of microcontroller and analog semiconductors, providing low-risk product development, lower total system cost and faster time to market for thousands of diverse customer applications worldwide. Headquartered in Chandler, Arizona, Microchip offers outstanding technical support along with dependable delivery and quality.

Voice: (480) 792-7200

Fax: (480) 792-7277

myMicrochip: http://www.microchip.com/pcn

Web Site: http://www.microchip.com

Forums: http://www.microchip.com/forums

Support: http://support.microchip.com

NOTES:

Glossary

A

Absolute Section

A section with a fixed (absolute) address that cannot be changed by the linker.

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 lowercase 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 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.

Attribute

Characteristics of variables or functions in a C program which are used to describe machine-specific properties.

Attribute, Section

Characteristics of sections, such as "executable", "read-only", 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.

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.

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 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 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 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.

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.

Identifier

A function or variable name.

IEEE

Institute of Electrical and Electronics Engineers.

Import

Bring data into the MPLAB 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 **–** 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 (ISR) 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 multi-byte data whereby the Least Significant Byte (LSB) 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, 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.

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's in-circuit debuggers that works with MPLAB IDE. *See* ICE/ICD.

MPLAB IDE

Microchip's Integrated Development Environment. MPLAB 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 or stand-alone. Replaces PRO MATE II.

MPLAB REAL ICE™ In-Circuit Emulator

Microchip's next-generation in-circuit emulators that works with MPLAB IDE. *See* ICE/ICD.

MPLAB SIM

Microchip's simulator that works with MPLAB IDE in support of PIC MCU and dsPIC DSC devices.

MPLIB™ Object Librarian

Microchip's librarian that can work with MPLAB 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, though it does not have to be.

MRU

Most Recently Used. Refers to files and windows available to be selected from MPLAB 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 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 being run in Simulator mode.

Nonvolatile 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 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 – 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.

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 ASM30 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

A portion of an application located at a specific address of memory.

Section Attribute

A 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 version.

Simulator

A software program that models the operation of devices.

Single Step

This command steps through code, one instruction at a time. After each instruction, MPLAB 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 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 typically managed by the compiler when developing code in a high-level language.

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 your 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 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 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 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’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.

W

Warning

MPLAB 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

NOTES:

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

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
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
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: 852-2401-1200
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: 86-755-8203-2660
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: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Osaka
Tel: 81-66-152-7160
Fax: 81-66-152-9310

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

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: 886-7-536-4818
Fax: 886-7-330-9305

Taiwan - Taipei
Tel: 886-2-2500-6610
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 - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

11/29/11