MPLAB® 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

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Table of Contents

Preface 1

Part 1 – MPLAB IDE

Chapter 1. What is MPLAB® IDE?

1.1 An Overview of Embedded Systems 9
1.2 The Development Cycle 14
1.3 Project Manager 15
1.4 Language Tools 16
1.5 Target Debugging 17
1.6 Device Programming 18
1.7 Components of MPLAB IDE 18
1.8 MPLAB IDE Documentation 19
1.9 MPLAB IDE On-line Help 19
1.10 MPLAB IDE Updates and Version Numbering 22

Chapter 2. Getting Started with MPLAB IDE: A Basic Tutorial

2.1 Introduction 23
2.2 MPLAB IDE Features and Installation 24
2.3 Tutorial Overview 25
2.4 Selecting the Device 27
2.5 Creating the Project 29
2.6 Setting Up Language Tools 30
2.7 Naming the Project 31
2.8 Adding Files to the Project 32
2.9 Building the Project 34
2.10 Creating Code 35
2.11 Building the Project Again 38
2.12 Testing Code with the Simulator 39
2.13 Tutorial Summary 45

Chapter 3. Walk-Through and Tutorial

3.1 Introduction 47
3.2 Selecting a Device 48
3.3 Setting Up Configuration Bits 48
3.4 Creating Source Code with the Editor 49
3.5 Creating a New Project 50
3.6 Using the Project Wizard 50
3.7 Setting Up the Language Toolsuite 50

3.8 Naming and Locating the Project 51
3.9 Adding Files 51
3.10 Completing the Project 52
3.11 Viewing the Project Window 52
3.12 Setting Build Options 53
3.13 Building The Project 53
3.14 Choosing a Debugger 54
3.15 Running Your Code 55
3.16 Viewing Debug Windows 55
3.17 Using Watch Windows 56
3.18 Using Breakpoints 57
3.19 Choosing a Programmer 58
3.20 Programming Your Part 59
3.21 Using Microchip Help 59

Chapter 4. Projects and Workspaces

4.1 Introduction 61
4.2 Using the Project Wizard 62
4.3 Creating/Updating any Project 63
4.4 Setting Up a Project Structure – Relative Paths 64
4.5 Project Folders and Files 65
4.6 Using A Version-Control System (VCS) 65
4.7 Setting Up/Changing a Project 68
4.8 Using a Single Project and Workspace 71
4.9 Using Multiple Projects in a Single Workspace 71
4.10 Building an Application without a Project 73

Chapter 5. Integrated Tools

5.1 Introduction 75
5.2 Language Toolsuites 75
5.3 Microchip Language Tools 77
5.4 Third Party Language Tools 79
5.5 Editors 81
5.6 Simulators 81
5.7 In-Circuit Emulators 81
5.8 In-Circuit Debuggers 82
5.9 Programmers 82
5.10 Third Party Tools 82

Chapter 6. MPLAB IDE Troubleshooting

6.1 Introduction 83
6.2 Common Problems/FAQ 83
6.3 Error Messages 85
6.4 Limitations 85

Part 2 – MPLAB IDE Reference

Chapter 7. MPLAB IDE Desktop

7.1 Introduction 89
7.2 MPLAB IDE Menu Bar 89
7.3 MPLAB IDE Toolbars 97
7.4 MPLAB IDE Status Bar 99

Chapter 8. MPLAB IDE Windows

8.1 Introduction 101
8.2 Changing Window Data and Properties 102
8.3 Code Display Window Symbols 103
8.4 Project Window 104
8.5 Output Window 107
8.6 Disassembly Listing Window 108
8.7 Hardware Stack Window 108
8.8 Program Memory Window 110
8.9 File Registers Window 113
8.10 EEPROM Window 115
8.11 LCD Pixel Window 116
8.12 Watch Window 118
8.13 Special Function Registers Window 121
8.14 Trace Memory Window 123
8.15 Configuration Bits Window 125
8.16 File (Editor) Window 126

Chapter 9. MPLAB IDE Dialogs

9.1 Introduction 129
9.2 About MPLAB IDE Dialog 130
9.3 Add Watch Dialog 130
9.4 Breakpoints Dialog 131
9.5 Build Options Dialog 132
9.6 Export Hex File Dialog 133
9.7 External Memory Setting Dialog 133
9.8 File Management Dialog 134
9.9 Fill Memory/Registers Dialog 135
9.10 Find In Project Files Dialog 135
9.11 Find and Replace Dialogs 135
9.12 Help Topics Dialog 136
9.13 Import Dialog 136
9.14 New Project Dialog 136
9.15 Project-Display Preferences Dialog 137
9.16 Project Wizard Dialogs 137
9.17 Properties Dialog 137
9.18 Save Project As Dialog 138

9.19 Select Device Dialog 139
9.20 Select Language Toolsuite Dialog 139
9.21 Set Language Tool Location Dialog 139
9.22 Settings Dialog 140
9.23 Table Setup Dialog 143
9.24 User ID Memory Dialog 143
9.25 Version-Control Dialog 144
9.26 Watch Dialog 145

Chapter 10. MPLAB IDE Operational Reference

10.1 Introduction 147
10.2 Command-Line Options 147
10.3 Files Used by MPLAB IDE 147
10.4 Saved Information 148

Part 3 – MPLAB Editor

Chapter 11. Using the Editor

11.1 Introduction 151
11.2 Configuring the Editor 152
11.3 Working with Files 154
11.4 Working with Text 156
11.5 Working with Debug Features 161
11.6 Keyboard Features 162
11.7 Editor Troubleshooting 164

Part 4 – MPLAB SIM

Chapter 12. Simulator Overview

12.1 Introduction 167
12.2 Simulator Features 167
12.3 Simulator Model – PICmicro MCUs 167
12.4 Simulator Model – dsPIC DSCs 177
12.5 Simulator Execution 178

Chapter 13. Getting Started with MPLAB SIM

13.1 Introduction 181
13.2 Using the Stopwatch 181
13.3 Using Stimulus 181
13.4 Using Simulator Trace 182
13.5 Using External Memory 182

Chapter 14. Using Stimulus

14.1 Introduction 187
14.2 SCL Generator Dialog 187
14.3 Stimulus Controller Dialog 194
14.4 Stimulus Input Interaction 195

Chapter 15. Using Stimulus – PIC17 Devices
15.1 Introduction 197
15.2 Using Pin Stimulus 197
15.3 Using File Stimulus 200
Chapter 16. Simulator Troubleshooting
16.1 Introduction 205
16.2 Common Problems/FAQ 205
16.3 Limitations 206
Chapter 17. Simulator Reference
17.1 Introduction 207
17.2 Debugging Functions 207
17.3 Settings Dialog 208
17.4 Settings Dialog – PIC17 Devices 210
Glossary 213
Index 227
Worldwide Sales and Service 232

NOTES:

Preface

NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site (www.microchip.com) to obtain the latest documentation available.

Documents are identified with a "DS" number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is "DSXXXXXA", where "XXXXX" is the document number and "A" is the revision level of the document.

For the most up-to-date information on development tools, see the MPLAB® IDE on-line help. Select the Help menu, and then Topics to open a list of available on-line help files.

INTRODUCTION

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

- Document Layout
- Conventions Used in this Guide
- Recommended Reading
- The Microchip Web Site
- Development Systems Customer Change Notification Service
- Customer Support

DOCUMENT LAYOUT

This document describes how to use MPLAB IDE to develop your firmware. The manual layout is as follows:

Part 1 - MPLAB IDE

- **Chapter 1: What is MPLAB IDE?** – Describes MPLAB IDE and how it can help develop an application.
- **Chapter 2: Getting Started with MPLAB IDE: A Basic Tutorial** – How to install MPLAB IDE v6.xx software and how to use the software to develop an example application.
- **Chapter 3: Walk-Through and Tutorial** – Walks through the necessary steps to develop an application using MPLAB IDE. An example is given with each step.
- **Chapter 4: Projects and Workspaces** – Describes the use of MPLAB Projects and Workspaces when developing an application. Includes information on the Project Wizard, version-control systems and projects and single and multiple file projects.

- **Chapter 5: Integrated Tools** – Describes the language tools (assemblers, compilers), software tools and hardware tools that may be used with MPLAB IDE.
- **Chapter 6: Troubleshooting** – Describes common problems and solutions with MPLAB IDE operation.

Part 2 - MPLAB IDE Reference

- **Chapter 7: MPLAB IDE Desktop** – Describes the MPLAB IDE desktop, including menu bar, toolbars and status bar.
- **Chapter 8: MPLAB IDE Windows** – Describes all MPLAB IDE windows. Includes window symbols definitions.
- **Chapter 9: MPLAB IDE Dialogs** – Describes all MPLAB IDE dialogs.
- **Chapter 10: MPLAB IDE Operational Reference** – Miscellaneous information on command-line options, shortcut (hot) keys, files used by MPLAB IDE and portability information.

Part 3 - MPLAB Editor

- **Chapter 11: Using the Editor** – Describes how to use MPLAB Editor. Includes text handling, configuring the editor, working with files and working with text. Additional information includes keyboard features, editor context (right mouse) menu and troubleshooting.

Part 4 - MPLAB SIM

- **Chapter 12: Simulator Overview** – An overview of MPLAB SIM simulator. Topics discussed are simulator features, models and exectution.
- **Chapter 13: Getting Started with MPLAB SIM** – Describes getting started using MPLAB SIM. Tutorials are suggested, and simulator features are discussed.
- **Chapter 14: Using Stimulus** – Describes the use of simulator stimulus for most PICmicro microcontroller (MCU) and dsPIC digital signal controller (DSC) devices. Discusses stimulus creation with the SCL generator and stimulus control.
- **Chapter 15: Using Stimulus - PIC17 Devices**– Details simulator stimulus use for PIC17CXXX MCU devices. Discusses pin and file stimulus.
- **Chapter 16: Simulator Troubleshooting** – Describes common problems and solutions with MPLAB SIM operation.
- **Chapter 17: Simulator Reference** – Details functions available for use in debugging an application with the simulator.

CONVENTIONS USED IN THIS GUIDE

The following conventions are may be used in this documentation:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	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	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	*File>Save*
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
'b*nnnn*	A binary number where *n* is a digit	'b00100, 'b10
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier font:		
Plain Courier	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`
Italic Courier	A variable argument	*`file`*`.o`, where *`file`* can be any valid filename
`0x`*`nnnn`*	A hexadecimal number where *`n`* is a hexadecimal digit	`0xFFFF`, `0x007A`
Square brackets [ ]	Optional arguments	`mcc18 [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 documentation describes how to use MPLAB 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 IDE, read the "`Readme for MPLAB IDE.txt`" file (an ASCII text file) in the Readmes subdirectory of the MPLAB IDE installation directory. The Readme file contains update information and known issues that may not be included in this documentation.

Readme Files

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

On-line Help Files

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

PICmicro Data Sheets and Family Reference Manuals

See the Microchip web site for complete and updated versions of device data sheets and related device family reference manuals.

dsPIC30F Family Overview (DS70043)

An overview of the dsPIC30F devices and architecture.

dsPIC30F Programmer's Reference Manual (DS70030)

This document provides general information on programming dsPIC30F devices, as well as a complete listing of the instruction set.

THE MICROCHIP WEB SITE

Microchip provides online support via our web site at 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 (FAQ), 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

DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com, click on Customer Change Notification and follow the registration instructions.

The Development Systems product group categories are:

- **Compilers** – The latest information on Microchip C compilers and other language tools. These include the MPLAB C17, MPLAB C18 and MPLAB C30 C compilers; MPASM™ and MPLAB ASM30 assemblers; MPLINK™ and MPLAB LINK30 object linkers; and MPLIB™ and MPLAB LIB30 object librarians.
- **Emulators** – The latest information on Microchip in-circuit emulators.This includes the MPLAB ICE 2000 and MPLAB ICE 4000.
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debugger, MPLAB ICD 2.
- **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 SIM simulator, MPLAB IDE Project Manager and general editing and debugging features.
- **Programmers** – The latest information on Microchip programmers. These include the MPLAB PM3 and PRO MATE® II device programmers and the PICSTART® Plus development programmer.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support
- Development Systems Information Line

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.

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

In addition, there is a Development Systems Information Line which lists the latest versions of Microchip's development systems software products. This line also provides information on how customers can receive currently available upgrade kits.

The Development Systems Information Line numbers are:

1-800-755-2345 – United States and most of Canada

1-480-792-7302 – Other International Locations

NOTES:

Part 1 – MPLAB IDE

Chapter 1. What is MPLAB® IDE? 9
Chapter 2. Getting Started with MPLAB IDE: A Basic Tutorial 23
Chapter 3. Walk-Through and Tutorial 47
Chapter 4. Projects and Workspaces 61
Chapter 5. Integrated Tools 75
Chapter 6. MPLAB IDE Troubleshooting 83

NOTES:

Chapter 1. What is MPLAB® IDE?

1.1 AN OVERVIEW OF EMBEDDED SYSTEMS

MPLAB IDE is a software program that runs on a PC to develop applications for Microchip microcontrollers. 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 **Section 1.7 "Components of MPLAB IDE"**. It is also recommended that **Section 1.9 "MPLAB IDE On-line Help"** and **Section 1.10 "MPLAB IDE Updates and Version Numbering"** be reviewed. The rest of this chapter briefly explains embedded systems development and how MPLAB 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 PICmicro® MCU or dsPIC® Digital Signal Controller (DSCs). 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

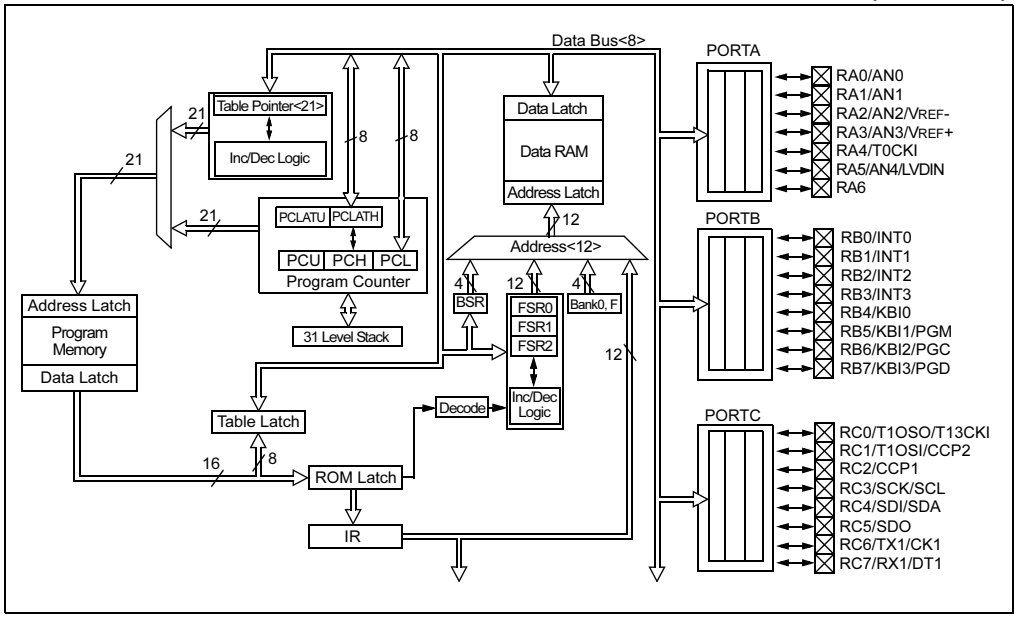
The main difference between an embedded controller and a PC is that the embedded controller is dedicated to one specific task or set of tasks. A PC is designed to run many different types of programs and to connect to many different external devices. An embedded controller has a single program and, as 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. It's 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 PICmicro MCU has program memory for the firmware, or coded instructions, to run a program. It also has "file register" memory for storage of variables that the program will need for computation or temporary storage. It also has a number of peripheral device circuits on the same chip. Some peripheral devices are called I/O ports. I/O ports are pins on the microcontroller that can be 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.

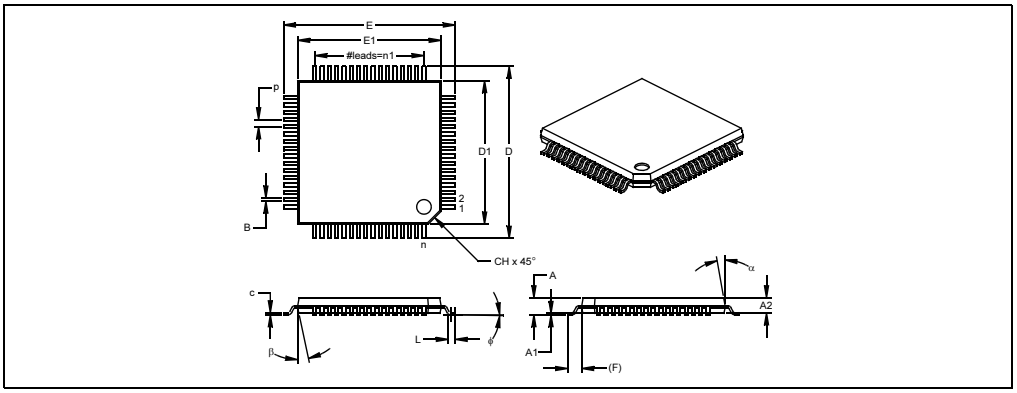
FIGURE 1-1: PICmicro® MCU DATA SHEET - BLOCK DIAGRAM (EXCERPT)



In order to design such a system, it must be decided which peripherals are needed for an application. Analog to Digital converters 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 PICmicro 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 PICmicro 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 1-2: PICmicro DEVICE PACKAGE



1.1.4 Implementing an Embedded System Design with MPLAB IDE

A development system for embedded controllers is a system of programs running on a desktop PC to help write, edit, debug and program code – the intelligence of embedded systems applications – into a microcontroller. MPLAB IDE, runs on a PC and 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 PICmicro or dsPIC 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 1-3: PICmicro MCU DATA SHEET - TIMING (EXCERPT)

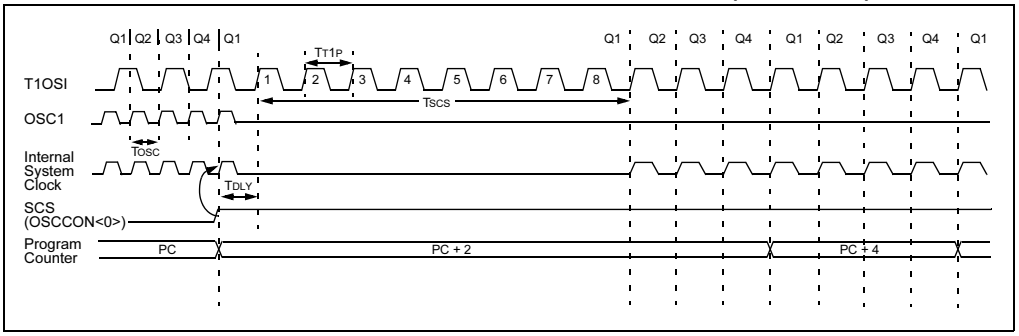
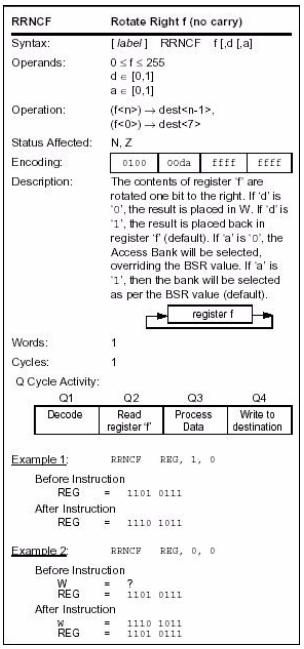


FIGURE 1-4: PICmicro MCU DATA SHEET - INSTRUCTIONS (EXCERPT)



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).

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 PICmicro MCU's. 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 IDE and its components to get through each step without continuously encountering new learning curves.

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

MPLAB IDE really helps with steps 2 through 5. 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, press the "build" button to try again. Often this write-compile-fix loop is done many times for complex code, as the sub-sections are written and tested. MPLAB 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 IDE has components called "debuggers" and free software simulators for all PICmicro and dsPIC 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.

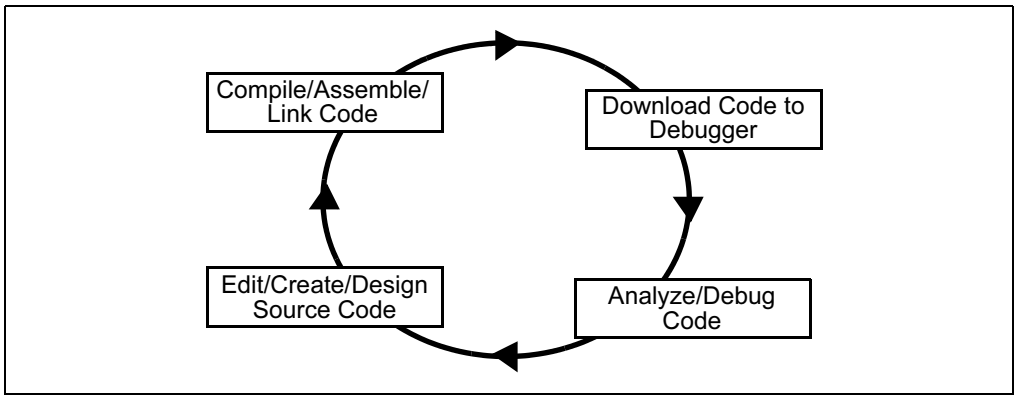
Once the hardware is in a prototype stage, a hardware debugger, such as MPLAB ICE or MPLAB ICD 2 can be used. These debuggers run the code in real time on your actual application. The MPLAB ICE physically replaces the microcontroller in the target using a high-speed probe to give you full control over the hardware in your design. The MPLAB ICD 2 uses special circuitry built into many Microchip MCUs with Flash program memory and can "see into" the target microcontrollers program and data memory. The MPLAB ICD 2 can 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 programmers, such as PICSTART Plus or MPLAB PM3. These programmers verify that the finished code will run as designed. MPLAB IDE supports most PICmicro MCUs and every dsPIC Digital Signal Controller.

1.2 THE DEVELOPMENT CYCLE

The process for writing an application is often described as a development cycle - as 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 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 1-5: THE DESIGN CYCLE

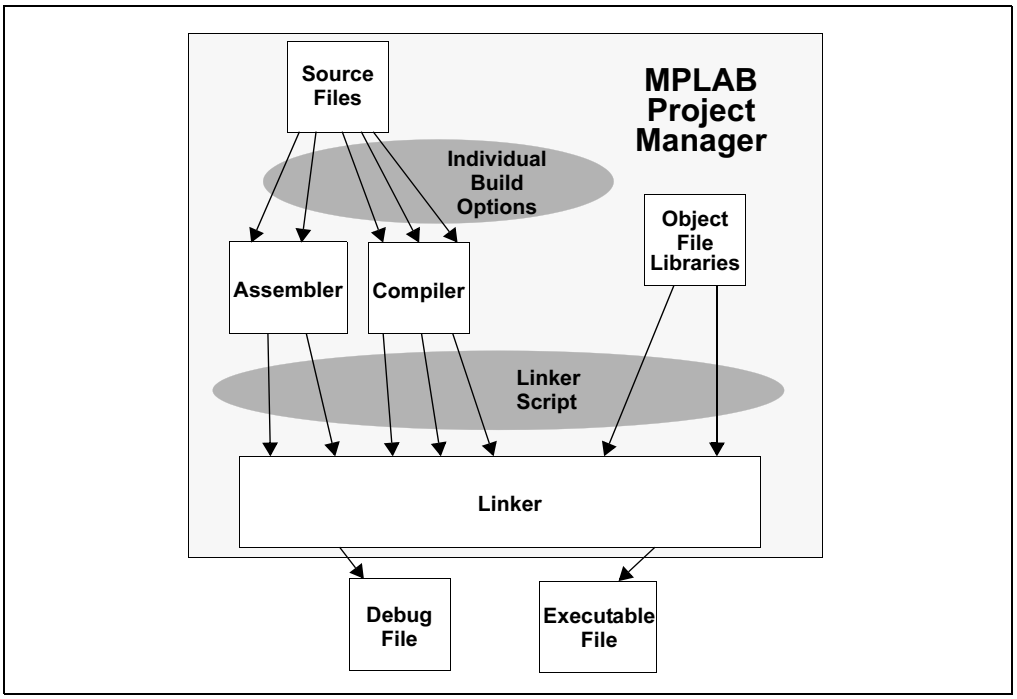


The IDE is a "wrapper" that coordinates all the tools from a single graphical user interface – usually automatically. For instance, once code is written, it can be converted to executable instructions and downloaded into a microcontroller to see how it works. In this process multiple tools are needed: 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”. From the MPLAB project manager, properties of the language tools can be invoked differently for each file, if desired, and a build process integrates all of the language tools operations.

FIGURE 1-6: MPLAB 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 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 IDE to relate the executing machine codes back to the source files.

A text editor is used to write the code. It is not a normal text editor, but an editor specifically designed for writing code for Microchip MCUs. 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, such as finding matching braces in C, commenting and un-commenting out blocks of code, finding text in multiple files and adding special bookmarks. After the code is written, the editor works with the other tools to display code execution in the debugger. Breakpoints 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.

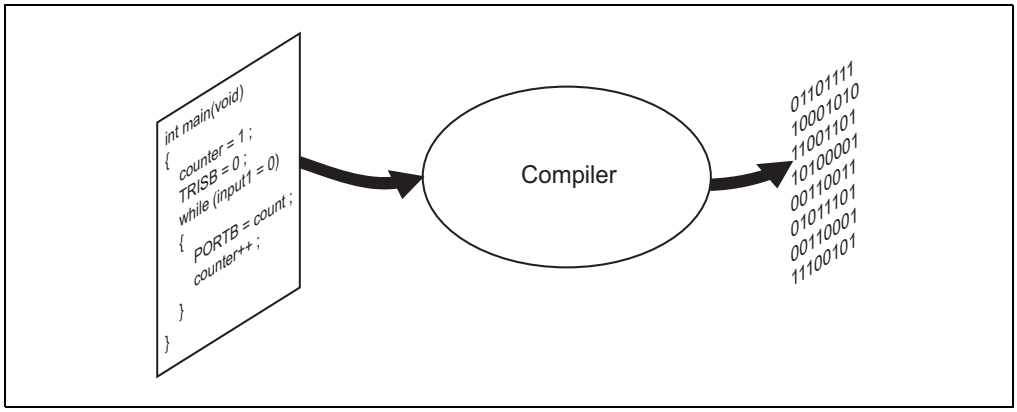
1.4 LANGUAGE TOOLS

Language tools are programs such as cross-assemblers and cross-compilers. Most people are familiar with language tools that run on a PC 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 PC but produce code to run on another microprocessor, hence they "cross-compile" code for a microcontroller that uses an entirely different set of instructions from the PC.

The language tools also produce a debug file that MPLAB IDE uses to correlate the machine instructions and memory locations with the source code. This bit of integration allows the MPLAB 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 for compilers that run and execute on a PC in that they must be very space conscious. The smaller the code produced, the better, because that allows the smallest possible memory for the target, thereby reducing cost. This means that techniques to optimize and enhance the code using machine specific knowledge are desirable. The size of programs for PC's 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 1-7: 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 special instrumentation to analyze the program as it executes in the application.

Simulators are built into MPLAB 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 PC is being used to simulate the operations of the microcontroller. In the case of MPLAB IDE, there are many simulators for each of the PICmicro and the dsPIC processors.

There are two types of hardware that can be used with MPLAB IDE: programmers and hardware debuggers. A programmer simply transfers 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, which use specialized hardware in place of the actual target microcontroller, or they can be 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, single-step to follow instructions as the hardware interacts with its specialized circuitry.

Debugging usually becomes urgent near the end of the projected 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, 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 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 the in-circuit debugger or a device programmer. MPLAB 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™), 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 the MPLAB ICD 2 after manufacturing for quality tests and development of next generation firmware.

1.7 COMPONENTS OF MPLAB IDE

The MPLAB IDE has both built-in components and plug-in modules to configure the system for a variety of software and hardware tools.

1.7.1 MPLAB IDE Built-In Components

The built-in components consist of:

- **Project Manager**

The project manager provides integration and communication between the IDE and the language tools.

- **Editor**

The editor is a full-featured programmer's text editor that also serves as a window into the debugger.

- **Assembler/Linker and Language Tools**

The assembler can be used standalone to assemble a single file, or can be used with the linker to build a project from separate source files, libraries and recompiled objects. The linker is responsible for positioning the compiled code into memory areas of the target microcontroller.

- **Debugger**

The Microchip debugger allows breakpoints, single-stepping, watch windows and all the features of a modern debugger for the MPLAB IDE. It works in conjunction with the editor to reference information from the target being debugged back to the source code.

- **Execution Engines**

There are software simulators in MPLAB IDE for all PICmicro and dsPIC devices. These simulators use the PC to simulate the instructions and some peripheral functions of the PICmicro and dsPIC devices. Optional in-circuit emulators and in-circuit debuggers are also available to test code as it runs in the applications hardware.

1.7.2 Additional Optional Components for MPLAB IDE

Optional components can be purchased and added to the MPLAB IDE:

- **Compiler Language Tools**

MPLAB C17, MPLAB C18 and MPLAB C30 from Microchip provide fully integrated, optimized code. Along with compilers from HI-TECH, IAR, microEngineering Labs, CCS and Byte Craft, they are invoked by the MPLAB IDE project manager to compile code that is automatically loaded into the target debugger for instant testing and verification.

- **Programmers**

PICSTART Plus, PRO MATE II, MPLAB PM3 as well as MPLAB ICD 2 can program code into target microcontrollers. MPLAB 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

- **In-Circuit Emulators**

MPLAB ICE 2000 and MPLAB ICE 4000 are full-featured emulators for the PICmicro and dsPIC devices. They connect to the PC via I/O ports and allow full control over the operation of microcontroller in the target applications.

- **In-Circuit Debugger**

MPLAB ICD 2 provides an economic alternative to an emulator. By using some of the on-chip resources, MPLAB ICD 2 can download code into a target microcontroller inserted in the application, set breakpoints, single step and monitor registers and variables.

1.8 MPLAB IDE DOCUMENTATION

The following documents are available to help you use MPLAB IDE:

- *MPLAB IDE Quick Chart* (DS51410)
- *MPLAB IDE Quick Start Guide* (DS51281)
- *MPLAB IDE User's Guide* (DS51519)

Other documents exist for various Microchip software and hardware tools that work with MPLAB IDE. Check the Microchip web site for downloadable pdf versions of all these documents.

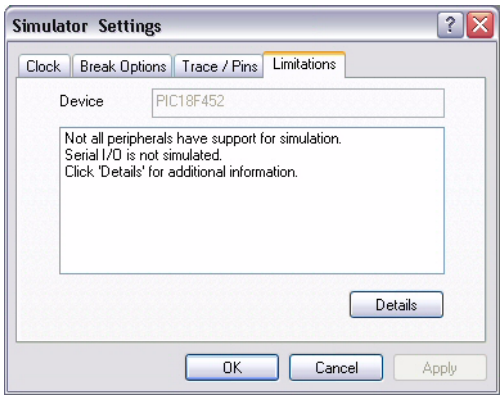
1.9 MPLAB IDE ON-LINE HELP

Since MPLAB IDE is under a constant state of change (see **Section 1.10 "MPLAB IDE Updates and Version Numbering"**) some details in this documentation may change. Dialogs might not appear exactly as they do in this manual, menu lists may be in different order, or may have new items. For this reason, the on-line help is the best reference to the version of MPLAB IDE being used.

MPLAB IDE comes with extensive on-line help, which is constantly being updated. If questions arise while using MPLAB IDE, be sure to check the on-line help for answers. Most importantly, the on-line help lists any restrictions that might exist for a particular tool in support of a particular device. Always try to review this section before working with a new device/tool combination.

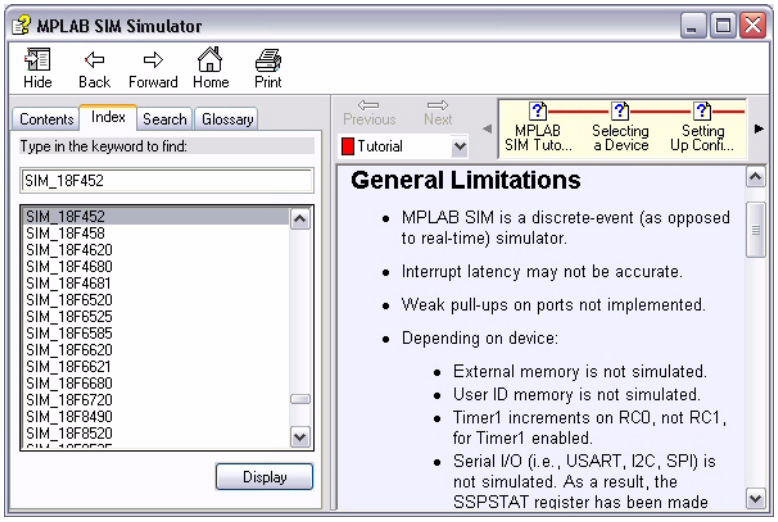
The Limitations tab in the *Debugger>Settings* dialog displays any restrictions the simulator, emulator or in-circuit debugger might have, compared to the actual device being simulated. General limitations are shown in the text area.

FIGURE 1-8: *DEBUGGER>SETTINGS*: LIMITATIONS TAB



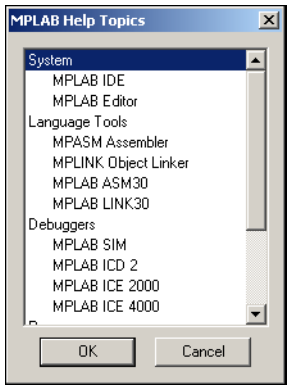
Press the **Details** button to show specific limitations of the device being debugged. From this display, help on general limitations related to the debugger can also be accessed.

FIGURE 1-9: SIMULATOR LIMITATIONS DETAIL



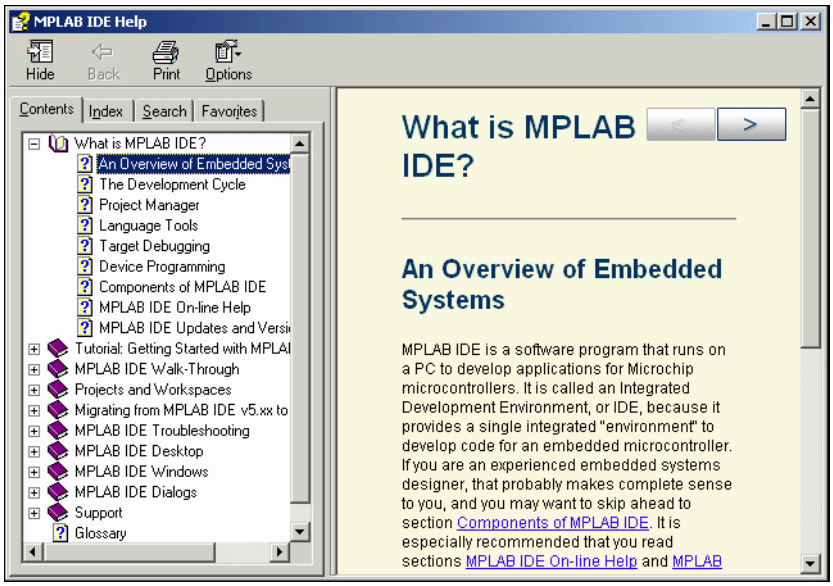
From the main MPLAB IDE Help menu, select *Help>Topics* to get a list of help on MPLAB IDE and all of its components.

FIGURE 1-10: MPLAB IDE *HELP>TOPICS* MENU



MPLAB IDE Help covers all aspects of MPLAB IDE and all of the Microchip tools. It can be viewed in an outline, as an index and with a search utility for help on any MPLAB IDE topic. It also directs users to other types of assistance, such as the Microchip Update Notification system.

FIGURE 1-11: MPLAB IDE HELP DIALOG



1.10 MPLAB IDE UPDATES AND VERSION NUMBERING

MPLAB IDE is an evolving program with thousands of users. Microchip is continually designing new microcontrollers with new features. Many new MPLAB IDE features come from customer requests and from internal usage. Continued new designs and the release of new microcontrollers ensure that MPLAB IDE will continue to evolve. MPLAB IDE is scheduled for a version update approximately every four months to add new device support and new features. Additional "interim" releases are released in between these major releases. The version numbering scheme for MPLAB IDE reflects whether it is a major production release or an interim release. If the version number ends in a zero, i.e., MPLAB IDE v6.50, v.6.60 or v7.00, this identifies a major production release. If the version number ends with a digit other than zero, i.e., v6.41, v6.52 or v7.55, this identifies an interim release. Interim releases are typically provided for early adopters of new devices or components, for quick critical fixes or for previews of new features. These interim releases are not as fully tested as the production releases, and are therefore, not recommended for rigorous design use. It is advised that production releases be used for development work, unless a new device or component is being used or a particular problem has been fixed in an interim release that enables more productive use of MPLAB IDE.

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

To be notified of updates to MPLAB IDE and its components, subscribe to the Development Tools section of the Customer Change Notification service on www.microchip.com.

Chapter 2. Getting Started with MPLAB IDE: A Basic Tutorial

2.1 INTRODUCTION

MPLAB Integrated Development Environment (IDE) is a comprehensive editor, project manager and design desktop for application development of embedded designs using Microchip PICmicro and dsPIC microcontrollers.

The initial use of MPLAB IDE is covered here. How to make projects, edit code and test an application will be the subject of a short tutorial. By going through the tutorial, the basic concepts of the Project Manager, Editor and Debugger can be quickly learned. The complete feature set of MPLAB IDE is covered in later chapters.

This section details the installation and uninstall of MPLAB IDE. It is followed by a simple step-by-step tutorial that creates a project and explains the elementary debug capabilities of MPLAB IDE. Someone unfamiliar with MPLAB IDE will get a basic understanding of using the system to develop an application. No previous knowledge is assumed, and comprehensive technical details of MPLAB IDE and its components are omitted in order to present the basic framework for using MPLAB IDE.

These basic steps will be covered in the tutorial:

- MPLAB IDE Features and Installation
- Tutorial Overview
- Selecting the Device
- Creating the Project
- Setting Up Language Tools
- Naming the Project
- Creating Code
- Building the Project Again
- Testing Code with the Simulator
- Tutorial Summary

2.2 MPLAB IDE FEATURES AND INSTALLATION

MPLAB IDE is a Windows® OS based Integrated Development Environment for the PICmicro MCU families and the dsPIC Digital Signal Controllers. The MPLAB IDE provides the ability to:

- Create and edit source code using the built-in editor.
- Assemble, compile and link source code.
- Debug the executable logic by watching program flow with the built-in simulator or in real time with in-circuit emulators or in-circuit debuggers.
- Make timing measurements with the simulator or emulator.
- View variables in Watch windows.
- Program firmware into devices with device programmers (for details, consult the user's guide for the specific device programmer).

Note:	Selected third party tools are also supported by MPLAB IDE. Check the release notes or readme files for details.

2.2.1 Install/Uninstall MPLAB IDE

To install MPLAB IDE on your system:

Note:	For some Windows OS's, administrative access is required in order to install software on a PC.

- If installing from a CD-ROM, place the disk into a CD drive. Follow the on-screen menu to install MPLAB IDE. If no on-screen menu appears, use Windows Explorer to find and execute the CD-ROM menu, `menu.exe`.
- If downloading MPLAB IDE from the Microchip web site (www.microchip.com), locate the download (.zip) file, select the file and save it to the PC. Unzip the file and execute the resulting file to install.

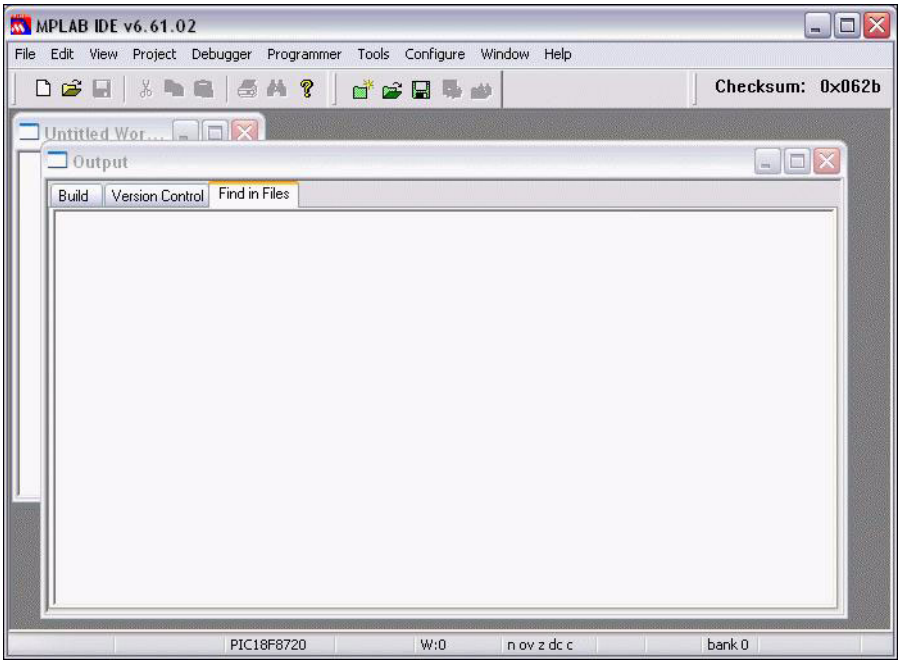
To uninstall MPLAB IDE:

- Select *Start>Settings>Control Panel* to open the Control Panel.
- Double-click on Add/Remove Programs. Find MPLAB IDE on the list and click on it.
- Click **Change/Remove** to remove the program from your system.

2.2.2 Running MPLAB IDE

To start MPLAB IDE, double click on the icon installed on the desktop after installation or select *Start>Programs>Microchip MPLAB IDE vx.x>MPLAB IDE vx.x*. A screen will display the MPLAB IDE logo followed by the MPLAB IDE desktop (Figure 2-1).

FIGURE 2-1: MPLAB IDE DESKTOP



2.3 TUTORIAL OVERVIEW

In order to create code that is executable by the target PICmicro MCU, source files need to be put into a project. The code can then be built into executable code using selected language tools (assemblers, compilers, linkers, etc.). In MPLAB IDE, the project manager controls this process.

All projects will have these basic steps:

- Select Device

  The capabilities of MPLAB IDE vary according to which device is selected. Device selection should be completed before starting a project.
- Create Project

  MPLAB Project Wizard will be used to Create a Project.
- Select Language Tools

  In the Project Wizard the language tools will be selected. For this tutorial, the built-in assembler and linker will be used. For other projects one of the Microchip compilers or other third party tools might be selected.
- Put Files in Project

  Two files will be put into the project, a template file and a linker script. Both of these exist in sub-folders within the MPLAB IDE folder. Using these two files it is easy to get started.
- Create Code

  Some code will be added to the template file to send an incrementing value out an I/O port.
- Build Project

The project will be built – causing the source files to be assembled and linked into machine code that can run on the selected PICmicro MCU.

- Test Code with Simulator

  Finally, the code will be tested with the simulator.

The Project Wizard will easily guide us through most of these steps.

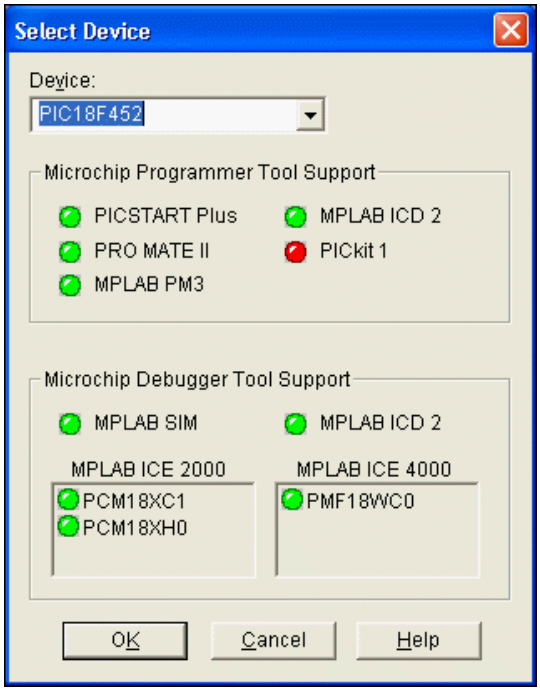
Note:	Some aspects of the user interface will change in future product releases and the screen shots in this tutorial may not exactly match the appearance of the MPLAB IDE desktop in later releases. New features will be added as additional parts are released. None of the functions described in this tutorial will be removed, but more features may be added.The on-line help is the most up-to-date reference for the current version of MPLAB IDE.

2.4 SELECTING THE DEVICE

To show menu selections in this document, the menu item from the top row in MPLAB IDE will be shown after the menu name like this *MenuName>MenuItem*. To choose the *Select Device* entry in the *Configure* menu, it would be written as *Configure>Select Device*.

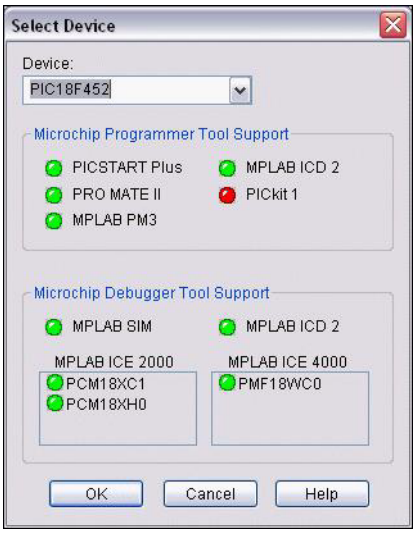
Choose *Configure>Select Device*.

FIGURE 2-2: SELECTING THE DEVICE



In the Device dialog, select the **PIC18F452** from the list if it's not already selected.

FIGURE 2-3: SELECT DEVICE DIALOG



The "lights" indicate which MPLAB IDE components support this device.

- A green light indicates full support.
- A yellow light indicates minimal support for an upcoming part that might not be fully supported in this release by the particular MPLAB IDE component. Components with a yellow light instead of a green light are often intended for early adopters of new parts who need quick support and understand that some operations or functions may not be available.
- A red light indicates no support for this device. Support may be forthcoming or inappropriate for the tool, e.g., dsPIC devices cannot be supported on MPLAB ICE 2000.

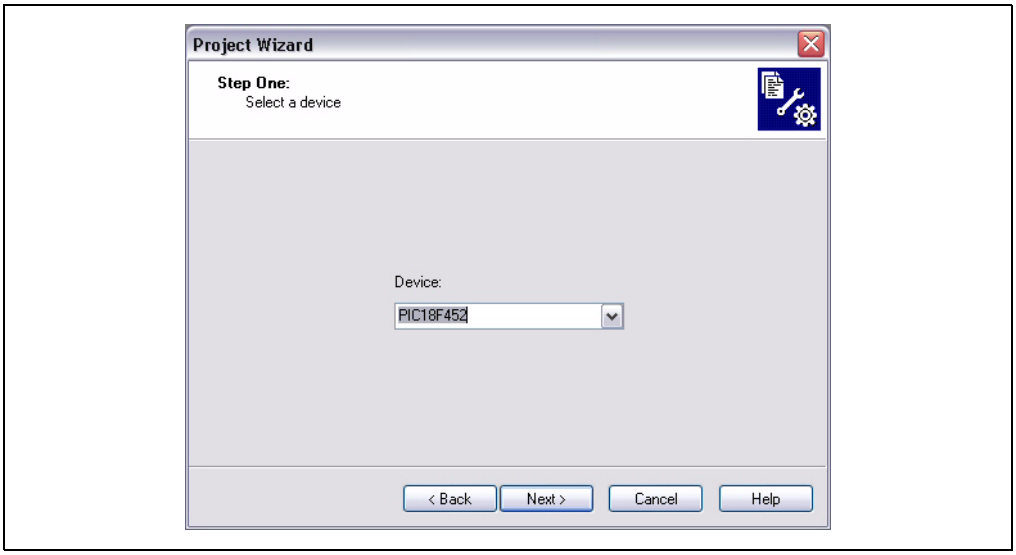
2.5 CREATING THE PROJECT

The next step is to create a project using the Project Wizard. A project is the way the files are organized to be compiled and assembled. We will use a single assembly file for this project and a linker script. Choose *Project>Project Wizard*.

From the Welcome dialog, click on **Next>** to advance.

The next dialog (Step One) allows you to select the device, which we've already done. Make sure that it says PIC18F452. If it does not, select the PIC18F452 from the drop down menu. Click **Next>**.

FIGURE 2-4: PROJECT WIZARD - SELECT DEVICE



2.6 SETTING UP LANGUAGE TOOLS

Step Two of the Project Wizard sets up the language tools that are used with this project. Select "Microchip MPASM Toolsuite" in the Active Toolsuite list box. Then "MPASM" and "MPLINK" should be visible in the Toolsuite Contents box. Click on each one to see its location. If MPLAB IDE was installed into the default directory, the MPASM assembler executable will be:

```
C:\Program Files\MPLAB IDE\MCHIP_Tools\mpasmwin.exe
```

the MPLINK linker executable will be:

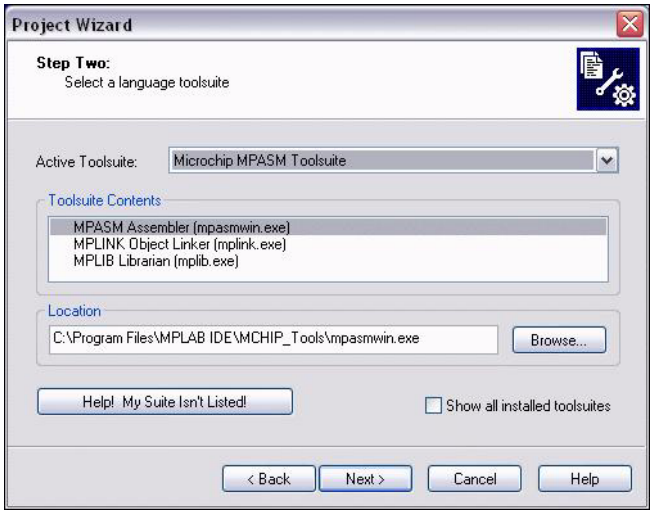
```
C:\Program Files\MPLAB IDE\MCHIP_Tools\mplink.exe
```

and the MPLIB librarian executable will be:

```
C:\Program Files\MPLAB IDE\MCHIP_Tools\mplib.exe
```

If these do not show up correctly, use the browse button to set them to the proper files in the MPLAB IDE subfolders.

FIGURE 2-5: PROJECT WIZARD - SELECT LANGUAGE TOOLS

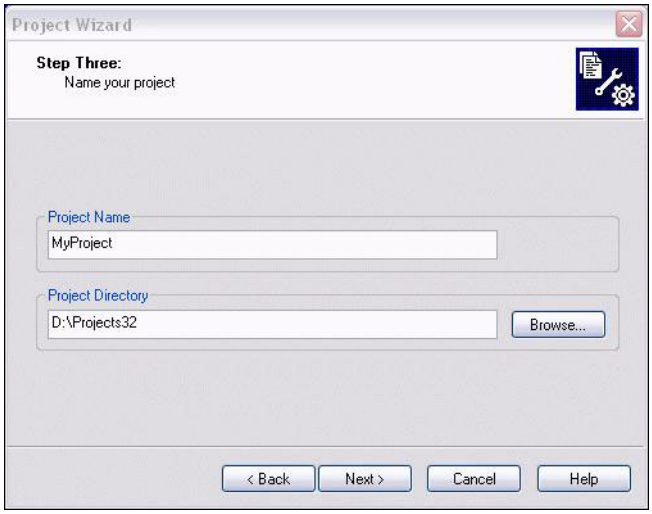


When you are finished, click **Next>**.

2.7 NAMING THE PROJECT

Step Three of the wizard allows you to name the project and put it into a folder. This sample project will be called `MyProject`. Using the Browse button, place the project in a folder named `Projects32`. Click **Next>**.

FIGURE 2-6: PROJECT WIZARD - NAME PROJECT



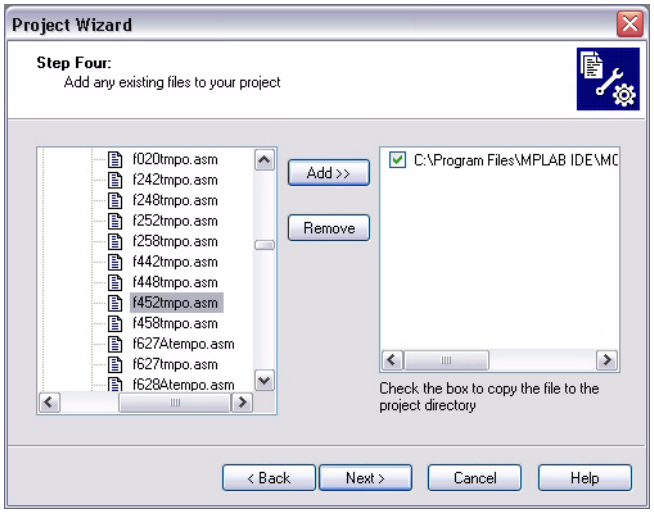
2.8 ADDING FILES TO THE PROJECT

Step Four of the Project Wizard allows file selection for the project. A source file has not yet been selected, so we will use an MPLAB IDE template file. The template files are simple files that can be used to start a project. They have the essential sections for any source file, and contain information that will help you write and organize your code. These files are in the `MPLAB IDE` folder, which by default is in the `Program Files` folder on the PC. There is one template file for each Microchip PICmicro and dsPIC device.

Choose the file named `f452tmpo.asm`. If MPLAB IDE is installed in the default location, the full path to the file will be:

```
C:\Program Files\MPLAB IDE\MCHIP_Tools\TEMPLATE\Object\f452tmpo.asm
```

FIGURE 2-7: PROJECT WIZARD - SELECT TEMPLATE FILE



Press **Add>>** to move the file name to the right panel, and click on the check box at the start of the line with the file name to enable this file to be copied to our project directory.

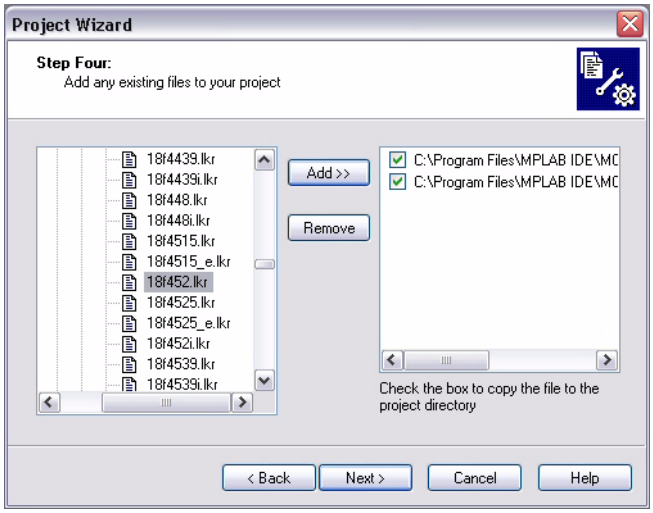
Next, add the second file for our project, the linker script. There is a linker script for each device. These files define the memory configuration and register names for the various parts. The linker scripts are in the folder named `LKR` under the `MCHIP_Tools` folder. Use the file named `18F452.lkr`. The full path is:

```
C:\Program Files\MPLAB IDE\MCHIP_Tools\LKR\18F452.lkr
```

Note: There is also a linker script named `18F452i.lkr`, for use with this device when MPLAB ICD 2 is being used (hence the "`i`" in the name). That linker script reserves areas in memory for MPLAB ICD 2. Since the simulator is being used, we don't need to use that linker script.

To copy this linker script into our project, click on the check box.

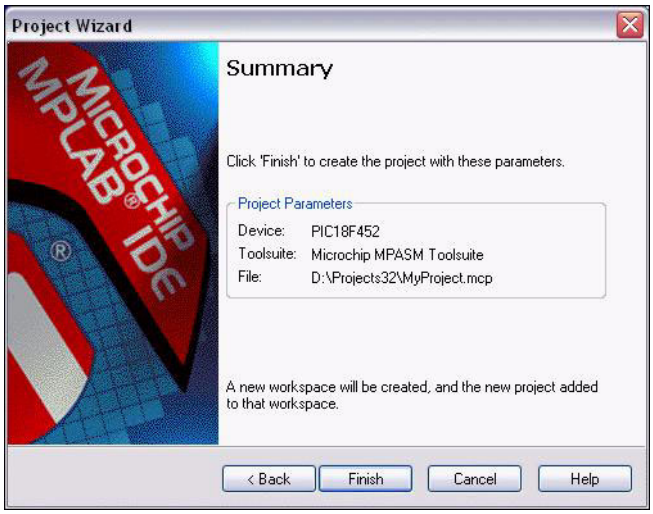
FIGURE 2-8: PROJECT WIZARD - SELECT LINKER SCRIPT



Make sure that your dialog looks like the picture above, with both check boxes checked, then press **Next>** to finish the Project Wizard.

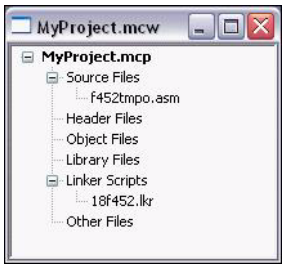
The final screen of the Project Wizard is a summary showing the selected device, the toolsuite and the new project file name.

FIGURE 2-9: PROJECT WIZARD - SUMMARY



After pressing the **Finish** button, review the Project Window on the MPLAB IDE desktop. It should look like Figure 2-10. If the Project Window is not open, select *View>Project*.

FIGURE 2-10: PROJECT WINDOW



TIP: Files can be added and projects saved by using the right mouse button in the project window. In case of error, files can be manually deleted by selecting them and using the right mouse click menu.

2.9 BUILDING THE PROJECT

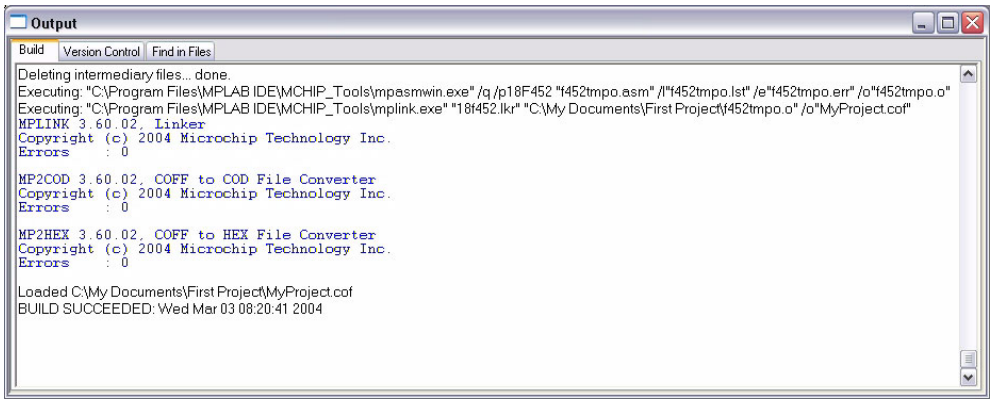
From the Project menu, we can assemble and link the current files. They don't have any of our code in them yet, but this assures that the project is set up correctly.

To build the project, select either:

- *Project>Build All*
- Right-click on the project name in the project window and select Build All
- Click the Build All icon on the Project toolbar. Hover the mouse over icons to see pop-up text of what they represent.

The Output window shows the result of the build process. There should be no errors on any step.

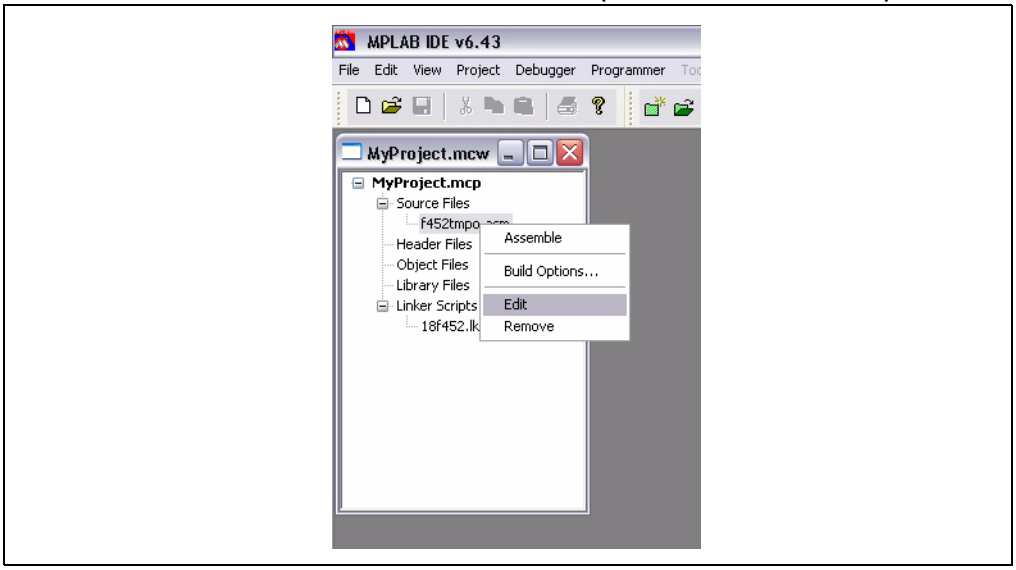
FIGURE 2-11: OUTPUT WINDOW



2.10 CREATING CODE

Open the template file in the project by double clicking on its name in the Project Window, or by selecting it with the cursor and using the right mouse button to bring up the context menu:

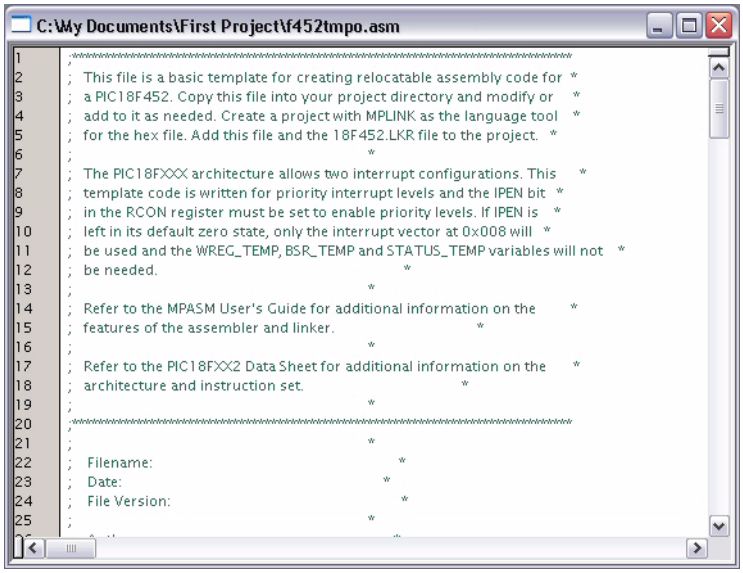
FIGURE 2-12: PROJECT CONTEXT MENU (RIGHT MOUSE CLICK)



The file has some comments at the beginning, and this area can be used as a standard comment information header for the file. For now you'll leave this as it is, but if this were a real project, you could put information about your design here.

Note: Line numbers are shown here. Line numbers may be toggled on/off by right-clicking in the editor window, selecting Properties, and then checking/unchecking "Line Numbers" on the **Editor** tab of the Editor Options dialog.

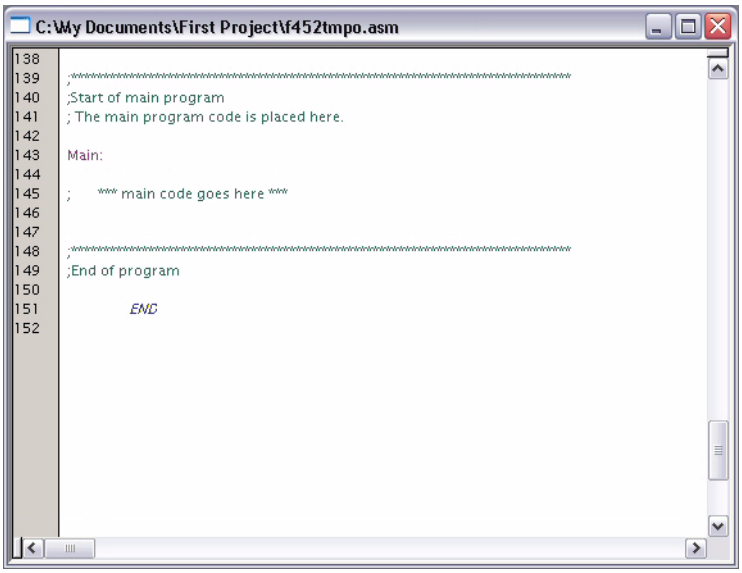
FIGURE 2-13: TEMPLATE FILE



Scroll down to the bottom of the file.

The code in the first part of the file is for more advanced functions such as setting up interrupts and configuration bits in a final application. These details can be ignored at this point with focus on writing the code. The new code will be placed in the file at the point after the symbol `Main` is defined.

FIGURE 2-14: TEMPLATE FILE - MAIN

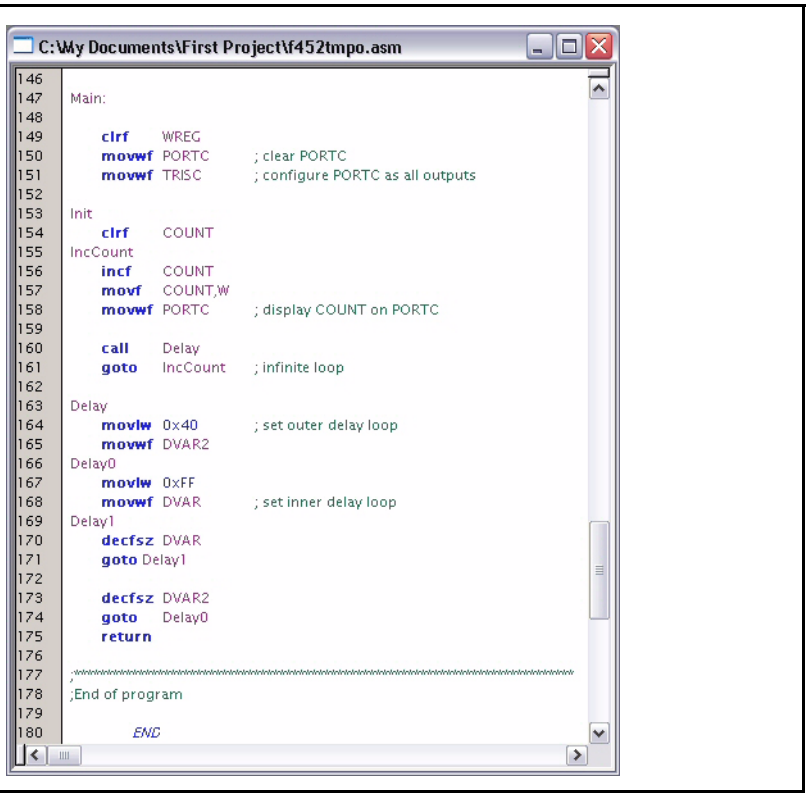


When any source file is opened, you are automatically in the editor. Type in this code:

```
Main:

    clrf   WREG
    movwf  PORTC; clear PORTC
    movwf  TRISC; configure PORTC as all outputs

Init
    clrf   COUNT
IncCount
    incf   COUNT
    movf   COUNT,W
    movwf  PORTC; display COUNT on PORTC

    callDelay
    goto   IncCount; infinite loop

Delay
    movlw  0x40; set outer delay loop
    movwf  DVAR2
Delay0
    movlw  0xFF
    movwf  DVAR; set inner delay loop
Delay1
    decfsz DVAR
    goto   Delay1

    decfsz DVAR2
    goto   Delay0
    return
```

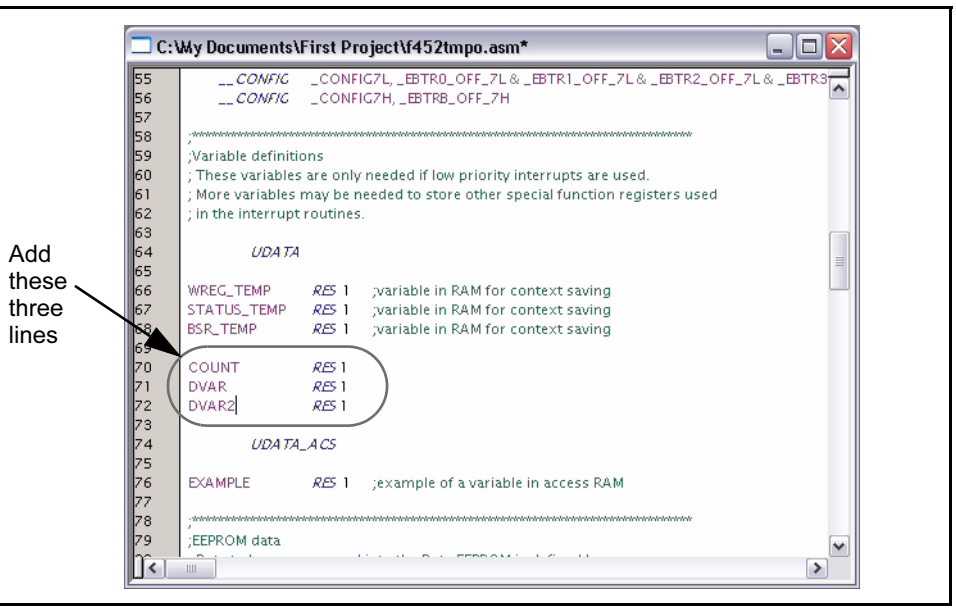
The template file should now look like Figure 2-15.

FIGURE 2-15: TEMPLATE FILE - ADD CODE



In this bit of code, we used three variables named COUNT, DVAR and DVAR2. These variables need to be defined in the template file in the UDATA section for uninitialized data. There are already three variables in this section of the template file, ours can be added at the end using the same format. Each variable is an 8-bit variable, so they only need to reserve 1 byte each.

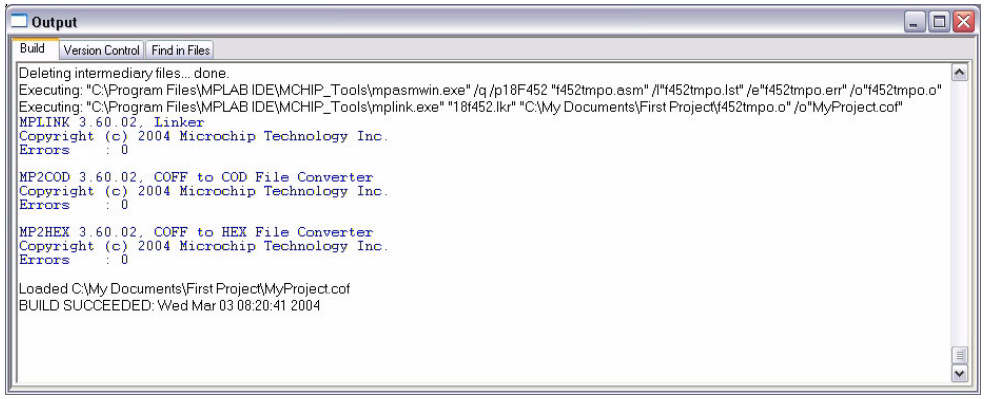
FIGURE 2-16: TEMPLATE FILE - ADD VARIABLES



2.11 BUILDING THE PROJECT AGAIN

Select *Project>Build All* to assemble and link the code. If the code assembled with no errors, the Output Window will look like Figure 2-17.

FIGURE 2-17: BUILD OUTPUT WINDOW



If these do not assemble and link successfully, check the following items and then build the project again:

- Check the spelling and format of the code entered in the editor window. Make sure the new variables and the special function registers, TRISC and PORTC, are in upper case. If the assembler reported errors in the Output window, double click on the error and MPLAB IDE will open the corresponding line in the source code with a green arrow in the left margin of the source code window.
- Check that the correct assembler (MPASM assembler) and linker for PICmicro devices is being used. Select *Project>Set Language Tool Locations*. Click on the plus boxes to expand the Microchip MPASM toolsuite and its executables. Click MPASM Assembler (`mpasmwin.exe`) and review their location in the display. If the location is correct, click **Cancel**. If it is not, change it and then click **OK**. The default search paths can be empty.

Upon a successful build, the output file generated by the language tool will be loaded. This file contains the object code that can be programmed into a PICmicro MCU and debugging information so that source code can be debugged and source variables can be viewed symbolically in Watch windows.

Note:	The real power of projects is evident when there are many files to be compiled/assembled and linked to form the final executable application – as in a real application. Projects keep track of all of this. Build options can be set for each file that access other features of the language tools, such as report outputs and compiler optimizations.

2.12 TESTING CODE WITH THE SIMULATOR

In order to test the code, software or hardware is needed that will execute the PICmicro instructions. A debug execution tool is a hardware or software tool that is used to inspect code as it executes a program (in this case `cnt452.asm`). Hardware tools such as MPLAB ICE or MPLAB ICD 2 can execute code in real devices. If hardware is not available, the MPLAB simulator can be used to test the code. For this tutorial use MPLAB SIM simulator.

The simulator is a software program that runs on the PC to *simulate* the instructions of the PICmicro MCU. It does not run in "real time," since the simulator program is dependent upon the speed of the PC, the complexity of the code, overhead from the operating system and how many other tasks are running. However, the simulator accurately *measures* the time it would take to execute the code if it were operating in real time in an application.

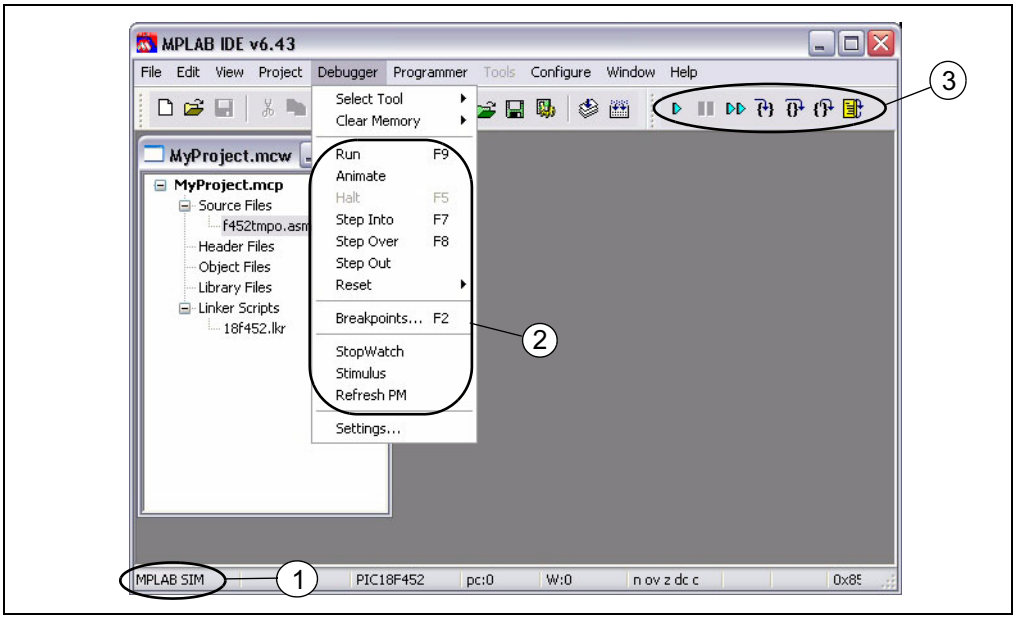
Note: Other debug execution tools include MPLAB ICE 2000, MPLAB ICE 4000 and MPLAB ICD 2. These are optional hardware tools to test code on the application PC board. Most of the MPLAB IDE debugging operations are the same as the simulator, but unlike the simulator, these tools allow the target PICmicro MCU to run at full speed in the actual target application.

Select the simulator as the debug execution tool. This is done from the *Debugger>Select Tool* pull down menu. After selecting MPLAB SIM, the following changes should be seen (see corresponding numbers in Figure 2-18).

1. The status bar on the bottom of the MPLAB IDE window should change to "MPLAB SIM".
2. Additional menu items should now appear in the Debugger menu.
3. Additional toolbar icons should appear in the Debug Tool Bar.

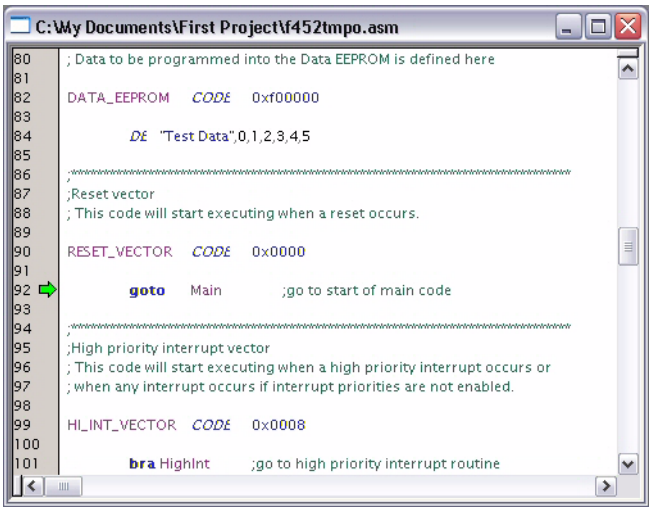
TIP: Position the mouse cursor over a toolbar button to see a brief description of the button's function.

FIGURE 2-18: MPLAB IDE DESKTOP WITH MPLAB SIM AS DEBUGGER



Next, select *Debugger>Reset* and a green arrow shows where the program will begin. This was part of the template file. The first instruction in memory jumps to the label called `Main`, where your code was inserted. This instruction jumps over the PIC18XXXX vector areas in lower memory.

FIGURE 2-19: *DEBUG>RESET*



To single step through the application program, select *Debugger>Step Into*. This will execute the currently indicated line of code and move the arrow to the next line of code to be executed.

There are shortcuts for these commonly used functions in the Debug Tool Bar.

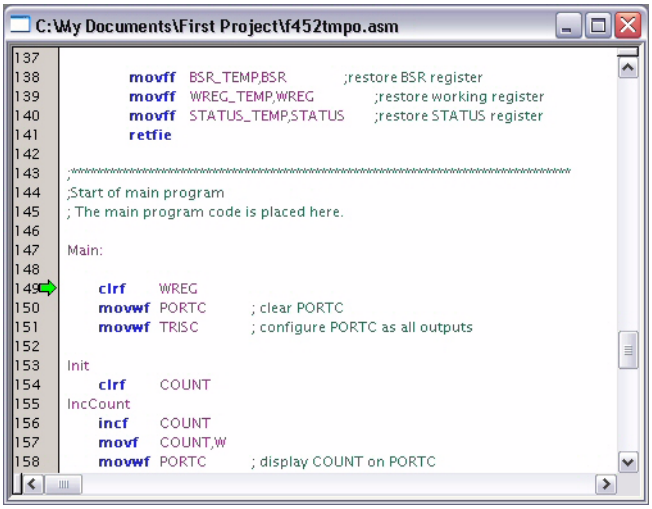
TABLE 2-1: DEBUG SHORT CUT ICONS

Debugger Menu	Toolbar Buttons	Hot Key
Run		F9
Halt		F5
Animate		
Step Into		F7
Step Over		F8
Step Out Of		
Reset		F6

TIP: Click on the appropriate icon on the toolbar or use the hot key shown next to the menu item. This is usually the best method for repeated stepping.

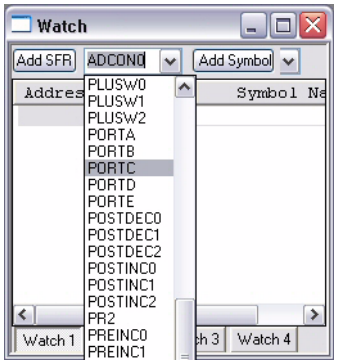
Next, press the Step Into icon or select *Debugger>Step Into* to single step to the code at `Main`.

FIGURE 2-20: *DEBUG>STEP INTO*



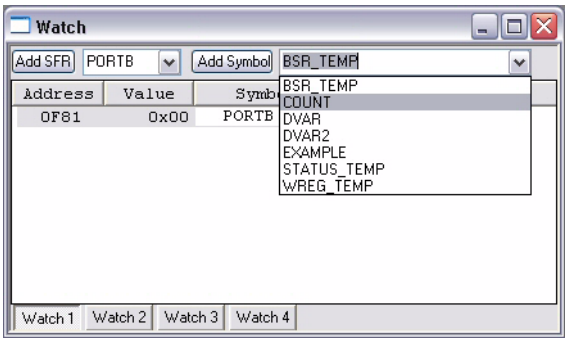
In order to see if the code is operating as intended, sending incrementing values out `PORTC`, watch the values being sent to `PORTC`. Select *View>Watch* to bring up an empty Watch Window. There are two pull downs on the top of the Watch Window. The one on the left labeled “Add SFR” can be used to add the Special Function Register, `PORTC`, into the watch. Select `PORTC` from the list and then click **Add SFR** to add it to the window.

FIGURE 2-21: WATCH - SELECT PORTC



The pull down on the right, allows symbols to be added from the program. Use this pull down to add the COUNT variable into the Watch Window. Select COUNT from the list and then click **Add Symbol** to add it to the window.

FIGURE 2-22: WATCH - SELECT VARIABLE "COUNT"



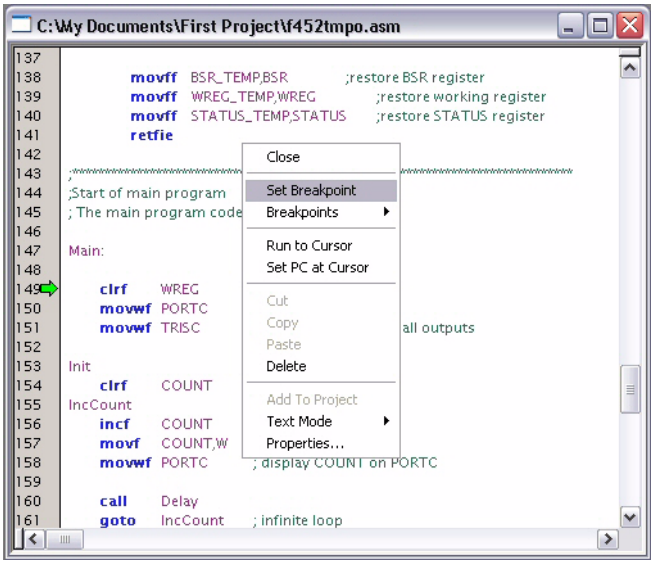
The watch window should now show the address, value and name of the two registers. At this point in the program, they will both be zero.

Note: Items can also be added to the watch window by either dragging them from the SFR, File Register or Editor window or clicking directly in the window under symbol name and typing in the item.

FIGURE 2-23: WATCH - RESET VALUES

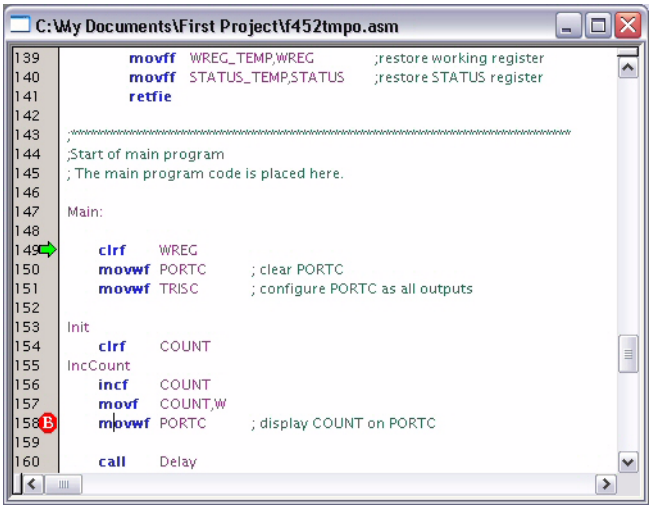
You could continue single stepping through the code, but instead, set a breakpoint just before the first value is sent out to PORTC. To set a breakpoint, put the cursor on the line and click the right mouse button.

FIGURE 2-24: DEBUG CONTEXT MENU (RIGHT MOUSE CLICK ON LINE)



Select Set Breakpoint from the context menu. A red "B" will show on the line. (You can also double-click on a line to add a breakpoint.)

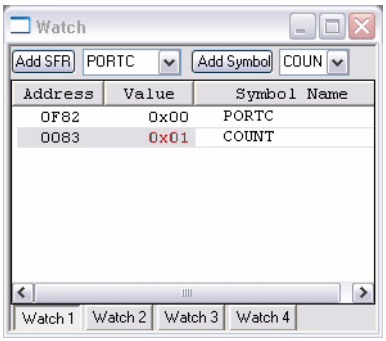
FIGURE 2-25: EDITOR WINDOW - SET BREAKPOINT



Select *Debugger>Run* to run the application. A text message "Running…" will briefly appear on the status bar before the application halts at this first breakpoint.

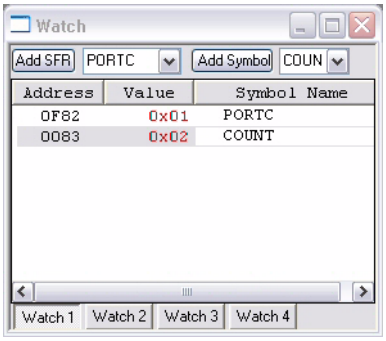
The watch window should now show that the variable COUNT was incremented by one, but since the breakpoint is at the line before the move to PORTC executes, PORTC still has a value of zero.

FIGURE 2-26: WATCH - AT BREAKPOINT



Press the Run icon to execute the code until it hits this point again. The Watch Window should now show both values incremented by one.

FIGURE 2-27: WATCH - NEXT BREAKPOINT

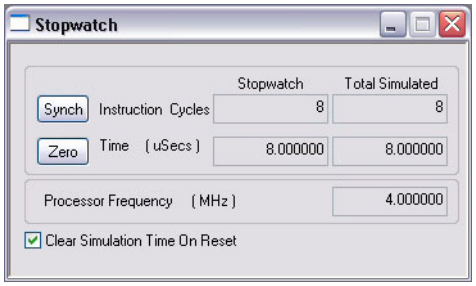


This would seem to indicate that the program is working as designed. You can single step through the code, or run the code more times to verify that it is executing properly. If you single step into the delay loop, you will get stuck executing thousands of steps until reaching the end. To exit out of the delay loop, use *Debugger>Step Out*.

If you are interested in calculating your delay time, the data book could be used to determine how long each instruction would take in your delay loop and you would come up with a pretty accurate number. You can also use the MPLAB StopWatch to measure the delay. Your main interest should be the time each new value of COUNT is being displayed. If you set your breakpoint as was initially done, on the instruction that moves COUNT to PORTC, you can run to the next breakpoint at the same place to measure the time.

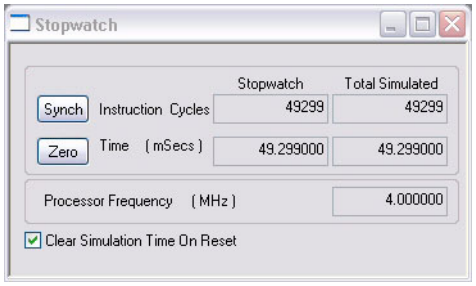
Use *Debugger>StopWatch* to bring up the StopWatch dialog. Make sure that a single breakpoint is set at the `movwf COUNT` instruction, and then press *Debug>Reset* and then *Debug>Run* to halt at the `movwf COUNT` instruction. With the default processor frequency of 4 MHz, the StopWatch should show that it took 8 microseconds to reach the first breakpoint.

FIGURE 2-28: STOPWATCH - AT FIRST BREAKPOINT



Execute Run again to go around the loop once, and note that the StopWatch shows that it took about 49 milliseconds. To change this, you can change the values in the delay loop.

FIGURE 2-29: STOPWATCH - AFTER DELAY



2.13 TUTORIAL SUMMARY

By completing this tutorial, you have performed the major steps for creating, building and testing a simple project. Tasks completed include:

Selecting the device - the PIC18F452.

Using the Project Wizard to create a project, and using the wizard to:

- select the MPLAB IDE built in MPASM assembler and MPLINK linker language tools,
- add files for the project: a template file for the device selected and a linker script to build it properly.

Writing some simple code to send a changing value out an I/O port.

Building the project.

And finally, testing the code with the simulator.

These are the essential steps for getting started with MPLAB IDE. You are now ready to continue exploring the capabilities of MPLAB IDE.

NOTES:

Chapter 3. Walk-Through and Tutorial

3.1 INTRODUCTION

This walk-through and tutorial is designed to help both new and experienced application developers. Step-by-step instructions for developing with MPLAB IDE and various Microchip development tools are given, where each step first discusses general information and then provides a specific example of use. All step examples tie together to form an example application.

Getting Started

- Selecting a Device
- Setting Up Configuration Bits
- Creating Source Code with the Editor

Creating, Setting Up and Building a Project

- Creating a New Project
- Using the Project Wizard
- Setting Up the Language Toolsuite
- Naming and Locating the Project
- Adding Files
- Completing the Project
- Viewing the Project Window
- Setting Build Options
- Building The Project

Debugging

- Choosing a Debugger
- Running Your Code
- Viewing Debug Windows
- Using Watch Windows
- Using Breakpoints

Programming

- Choosing a Programmer
- Programming Your Part

Additional Resources

- Using Microchip Help

3.2 SELECTING A DEVICE

To begin application development, select the Microchip device you wish to develop code for.

To choose a device:

1. Select *Configure>Select Device*.
2. In the Select Device dialog, choose a device from the Device list box. The Microchip Tool Support section will reflect the available tool support for that device.
3. Click **OK**.

The device you have selected should now be reflected on the status bar.

Tutorial Step 1:

Select the PIC18F8720 microcontroller as the device. Many Microchip tools support this device.

3.3 SETTING UP CONFIGURATION BITS

Configuration bits can be programmed (read as '0'), or left unprogrammed (read as '1'), to select various device configurations. Consult your device data sheet for more information.

3.3.1 Setting in MPLAB IDE or in Code

Configuration bits may be set in code (i.e., using the `__config` command or init code) or using the Configuration Bits window (*Configure>Configuration Bits*). Although you can use both methods, you probably should chose one to avoid confusion and overwriting of bit values.

If you want to change configuration bit values in the window, do not enter any in code until you have completed development. Then put your window values into code, rebuild the project (which will then enter the code values into the window) and perform a final run to make sure all is correct.

If you want to change configuration bit values in code, do not make changes to the window. Every time you build, the code values will be loaded into the window and you will lose any changes you have made in the window.

3.3.2 Saving/Clearing Values

Configuration bit window values are saved/restored on close/open of the workspace.

If "Clear program memory upon loading a program" is checked in the Settings dialog (*Configure>Settings*), **Program Loading** tab, configuration bits are cleared upon loading a program (i.e., build, import or open a project).

3.3.3 Setting for External Memory Use

If your device supports external memory, and you select a mode which uses external memory in the configuration bits, you will also need to set the amount of external memory used in the External Memory Settings dialog (*Configure>External Memory*).

Tutorial Step 2:

For your selected device:

- You will use the default configuration bits values, and therefore will not make any changes to the Configuration Bits window.
- Check the settings on the *Configure>Settings*, **Program Loading** tab. The following should be selected: “Clear memory before building a project” and “Clear program memory upon loading a program”.
- The PIC18F8720 device can use external memory. However, for this tutorial, you shall use microcontroller mode, or all on-chip memory.

3.4 CREATING SOURCE CODE WITH THE EDITOR

Select *File>New* to open an empty editor window in which to type your source code. To save your work, select *File>Save*.

For more on the editor, see **Chapter 11. “Using the Editor”**.

Tutorial Step 3:

Create/locate a folder (directory), using Windows Explorer, where you will place application project files. Then, type the following info an editor window, or cut and paste from here. Save the file as `cnt8720.asm` to the project folder when complete.

```
        title "PIC18F8720 Counting Program"
        #include <p18f8720.inc>

        udata  0x60
COUNT  res 1

RST     code   00h
        goto   Start

PGM     code
Start   clrf   WREG      ;clear W register
        movwf  PORTC     ;clear PORTC
        movwf  TRISC     ;config PORTC as outputs
Init
        clrf   COUNT     ;clr count
IncCount
        incf   COUNT,F   ;increment count
        movf   COUNT,W
        movwf  PORTC     ;display on port c
        goto   IncCount  ;loop

        end
```

3.5 CREATING A NEW PROJECT

A project contains the source files needed to build and test an application. A project configuration (workspace) includes the following: processor and display information, such as the nature, size and position of all open windows, toolbars, execution and debug settings. For more on projects and workspaces, see **Chapter 4. "Projects and Workspaces"**.

Project and workspace global setting may be selected on the **Project** and **Workspace** tabs of the Settings dialog (*Configure>Settings*).

Tutorial Step 4:

For the tutorial, keep the default setup for projects and workspaces. Most notably, this means you will be using the one-to-one project-workspace model.

3.6 USING THE PROJECT WIZARD

To create a new project in the current workspace, use the Project Wizard.

- Select *Project>Project Wizard*.
- The Project Wizard Welcome screen will appear. Click **Next>** to continue.
- The Project Wizard Select a Device screen will appear. Verify that the device you have already selected using the Select Device dialog is shown here. To change the device, use the drop-down menu. Click **Next>** to continue.

Tutorial Step 5:

You will use the device selected in step 1, so there is no need to make any changes.

3.7 SETTING UP THE LANGUAGE TOOLSUITE

You will now add language tools for use in the project. The Project Wizard Select a Language Toolsuite screen should be visible.

- Select a language toolsuite (Microchip or Third Party) for your project from the "Active Toolsuite" drop-down menu. Only those language toolsuites that are installed and work with the previously-selected device are available to select from this menu. To see all available (installed) toolsuites, check "Show all installed toolsuites."

  If you still don't see the desired toolsuite in this list, click **Help! My Toolsuite Isn't Listed!** for more information.
- A list of tools in the selected toolsuite are shown in a box under "Toolsuite Contents". A tool with a red "X" preceding it is not installed or the path to the executable is not known to MPLAB IDE. To assign or check assignments of tools to executable files, click on the tool to show the executable and path under "Location of Selected Tool". Type in this text box to enter or change the assignment, or click **Browse** to find the executable.
- Click **Next>** to continue.

Tutorial Step 6:

Set up the language tools for this tutorial as follows:

- For the “Active Toolsuite”, choose “Microchip MPASM Toolsuite”.
- Click on each of the “Toolsuite Contents” – MPASM Assembly, MPLINK Object Linker and MPLIB Object Librarian – to verify their “Location”. Click **Browse** to find the executable if it is not listed.

This toolsuite is installed with MPLAB IDE and its tool executables are located in the `MCHIP_Tools` subdirectory of the MPLAB IDE installation directory.

3.8 NAMING AND LOCATING THE PROJECT

You will now enter a name and location for your new project. The Project Wizard Name Your Project screen should be visible.

To set the directory:

- Type in the path to an existing directory or type in the path to a new directory, in which case you will be prompted to create the directory when you click **Next>**.
- Click **Browse** to browse to an existing directory or to one level above where you wish to place a new directory. Click **OK** in the Browse for Folder dialog. Complete the path if you are creating a new directory and then click **Next>**. You will be prompted to create the directory if it does not exist.

Tutorial Step 7:

Enter “cnt8720” for the “Project Name”. Then click **Browse** and locate the application project directory you set up when you created your source code in step 3.

3.9 ADDING FILES

You will now add files to the project. The Project Wizard Add Existing Files screen should be visible.

- Choose the file(s) to add. Click on a file name to select that file. Ctrl-Click to select more than one file. Click **Add>>** to list file(s) to be added to the new project.
- To remove file(s) from the list, click on the file name(s) to select and then click **Remove**.
- Click **Next>** to continue.

Tutorial Step 8:

You will be adding two files to your project:

- Locate the file “cnt8720.asm” and click on it. Then click **Add>>** to add it to the project.
- Locate the linker script file for PIC18F8720 devices, `18F8720.lkr`, in the `MCHIP_Tools/LKR` subdirectory of the MPLAB IDE installation directory.

> **Note:** If you are planning on using the MPLAB ICD 2 are your debug tool, choose file `18F8720i.lkr` instead.

Click on the file and then click **Add>>** to add it to the project. Also, click the check box next to the added file so that it is copied into the project directory.

3.10 COMPLETING THE PROJECT

You should now review your project setup. The Project Wizard Summary screen should be visible.

- Check the summarized information on the project. If anything is incorrect, use **Back** to return to the dialog you need and change the information.
- Click **Finish** when you are satisfied with the project setup.

Tutorial Step 9:

Click **Finish** to complete the project setup.

3.11 VIEWING THE PROJECT WINDOW

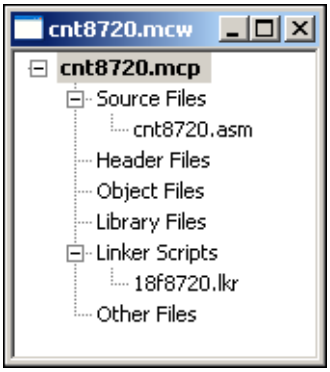
If it is not already open, open the Project window by selecting *View>Project*. The workspace name should be visible in the top bar of the Project window. The name of the project should be visible inside the Project window at the top of the display. A tree structure listing file types and any added files will appear below.

Tutorial Step 10:

Your project window should look like Figure 3-1.

- Right-click on tree items to view various pop-up menus.
- Right-click on "Source Files" and you will see "Add Files", which means you can add more source files to your project after you have initially set it up. Right-click on the file `cnt8720.asm` and you will see "Remove", which is how you can delete files from your project.
- Right-click in an open area of the window and select "Preferences" from the pop-up menu. This will open the Project-Display Preferences dialog.
- Although you will not be changing any preferences for this tutorial, notice the Version Control section. For more on using version control files in your projects, see **Section 4.6 "Using A Version-Control System (VCS)"**.
- Click **Cancel** to close the dialog.

FIGURE 3-1: EXAMPLE PROJECT WINDOW



3.12 SETTING BUILD OPTIONS

MPLAB IDE has default settings for tools and files contained in a project. However, you may want or need to modify these settings.

To set build options, select *Project>Build Options>Project* or right-click on the project name in the Project window and select Build Options from the pop-up menu.

- The Build Options dialog will open.
- Click the **General** tab and enter or **Browse** to paths for output files, include files library files or linker script files.

Note: These are MPLAB IDE paths. Not all language tools use this information.

- Click a specific language tool tab (e.g., MPASM Assembler) and set up operational features for that tool.
- Click **OK**.

To override project settings for a particular project file, e.g., `ProjFile1.asm`, select *Project>Build Options>ProjFile1.asm* or right-click on `ProjFile1.asm` in the Project window and select Build Options from the pop-up menu.

- The File Settings dialog will open.
- Click a specific language tool tab (e.g., MPASM Assembler) and set up operational features for that tool.
- Click **OK**.

Tutorial Step 11:

Set up project build options:

- Select *Project>Build Options>Project*.
- Click the **MPLINK Linker** tab. Click the checkbox by “Generate map file”.
- Click **OK**.

3.13 BUILDING THE PROJECT

Now you are ready to build your project. Select Build All from either:

- the menu bar (*Project>Build All*)
- the pop-up menu (right-click on the project name in the Project window)
- the toolbar ().

During the build:

- For MPASM assembler, a status window will open showing you the progress and final outcome of the build. It will close when complete.
- The Output window will also open. This will contain information about the build, including any errors encountered

If your project does not build successfully, please check the following items and then build the project again:

- Check the spelling and format of any code you entered in the editor window. If there are reported errors in the Output window, double-clicking on an error will indicate the corresponding line in your source code with a green arrow in the gutter of the source code window.
- Check that the correct language tool is being used for the project and for the project files.

Upon a successful build, the debug file (`*.cod`, `*.cof`, `*.elf`, etc.) generated by the language tool will be loaded. This file allows you to debug using your source code and view your variables symbolically in Watch windows.

Tutorial Step 12:

Build your project by selecting Build All as described above. The code should build successfully. If you do encounter build errors, follow the instructions above for troubleshooting build errors.

3.14 CHOOSING A DEBUGGER

Choose a tool to help debug your code from *Debugger>Select Tool*. Microchip provides several types of debug tools:

- **MPLAB SIM** simulator – Simulate device operation in software.
- **MPLAB ICD 2** in-circuit debugger – Debug your code in the device using an ICD and special Flash devices with built-in debug circuitry. See the Readme file in the MPLAB IDE directory for a list of supported devices.
- **MPLAB ICE 4000** in-circuit emulator – Emulate larger-memory PIC18 MCU's and dsPIC DSC's operation in hardware, with access to device memory locations.See the Readme file in the MPLAB IDE directory for a list of supported devices.
- **MPLAB ICE 2000** in-circuit emulator – Emulate most PICmicro MCU's operation in hardware, with access to device memory locations.See the Readme file in the MPLAB IDE directory for a list of supported devices.

Once you have chosen a debug tool, you will see changes in the following on the IDE:

- Debugger menu – Several standard options will be available. Other tool-specific options may also appear. Most of these options also will be available from the right mouse button menu in either the source code (file) window or Program Memory window.
- View menu – Depending on the debugger chosen, other debug options may appear.
- Debug toolbar – Several standard options will be available. Other tool-specific options may also appear.
- Status bar – The debug tool you have selected should now be shown.

Tutorial Step 13:

If you have purchased a hardware or third-party tool for application development, select it now from the Debugger menu. If it is not visible, please consult the documentation that came with the tool for proper installation and setup with MPLAB IDE.

If you do not have an additional tool, select the built-in simulator, MPLAB SIM.

3.15 RUNNING YOUR CODE

No matter what debug tool you select, you will need to run (execute) your code to debug it.

- Select *Debugger>Reset>Processor Reset*. There should be a solid green arrow in the gutter of your source code window, indicating the first source code line that will be executed.
- Select *Debugger>Run* to run your application. You will see "Running…" appear on the status bar. Also, the arrow in the gutter will appear hollow.
- To halt program execution, select *Debugger>Halt*. The line of code where the application halted will be indicated by the solid green arrow.
- You may also single step through the application program by selecting *Debugger>Step Into*. This will execute the currently indicated line of code and move the arrow to the next line of code that will be executed.

Note: You may click on the appropriate icon on the toolbar or use the hot key shown next to the menu item instead of selecting the menu item. This is usually the best approach for repeated running, halting and stepping when debugging.

Tutorial Step 14:

Reset your code (*Debugger>Reset*) and then run it by clicking the Run icon on the toolbar. (Hover your mouse over a toolbar icon to see its meaning.) Then stop the program by clicking the Halt icon.

3.16 VIEWING DEBUG WINDOWS

Standard debug windows allow you to view the contents of program, data or other types of device memory to aid in the debugging of your application. These windows are always visible on the View menu. Some items may be grayed out, however, if they are not supported by the device or debug tool. For more on each debug window, see **Chapter 8. "MPLAB IDE Windows"**.

- Disassembly Listing Window
- Hardware Stack Window
- Program Memory Window
- File Registers Window
- EEPROM Window
- LCD Pixel Window
- Watch Window
- Special Function Registers Window

Additional debug windows may appear on the menu depending on the debug tool you select.

- Trace Memory Window (Simulators and Emulators)

Tutorial Step 15:

You will likely use on the following windows to observe your running code:

- File (Editor) Window – Displays the actual assembly or C code.
- Program Memory Window – Displays your code as Opcode Hex, Machine or Symbolic.
- Disassembly Listing Window – Displays the disassembled code.
- Watch Window – Displays the values of registers and bits that you specify to "watch."

The file window should already be open, containing your code. You will be using the Watch window in the next step.

3.17 USING WATCH WINDOWS

In order to see what the application is doing, you will need to open a debug window to observe register values. Although many registers and their values are displayed in the File Register window and Special Function Register (SFR) window, you usually only wish to see a few, application-specific registers. To set up your own list of registers to view, use Watch windows (*View>Watch*).

To add an item to the Watch window:

- Choose an SFR or Symbol from the drop-down list and then click the corresponding Add button
- Drag an item from the SFR, File Register or File window
- Click directly in the window under symbol name and typing in the item

The file register address of the symbols is listed first, followed by the symbol name and finally the value of the symbol.

For more on Watch windows, see **Section 8.12 "Watch Window"**.

Tutorial Step 16:

You will first set up the Watch window:

- Select COUNT from the symbol selection box at the top of the window. Click **Add Symbol** to add it to the Watch window list. If you want to quickly advance through the list, start typing COUNT after selecting the pull down icon.
- Select WREG from the SFR selection box at the top of the window. Click **Add SFR** to add it to the Watch window list. If you want to quickly advance through the list, start typing WREG after selecting the pull down icon.
- Select PORTC from the SFR selection box at the top of the window. Click **Add SFR** to add it to the Watch window list. If you want to quickly advance through the list, start typing PORTC after selecting the pull down icon.

You should now have these symbols in the Watch window. Next, you will run your code and watch the symbol values change as you step through the program.

- Select *Debugger>Reset>Processor Reset* to reset your application.
- Select *Debugger>Step Into* (or click the equivalent toolbar icon) until you have stepped to the following program line:

```
incf COUNT,F ;increment count
```

- Step one more time to see the value of COUNT in the Watch window change from 00 to 01.
- Step one more time to see the value of WREG in the Watch window change from 00 to 01.
- Step one more time to see the value of PORTC in the Watch window change from 00 to 01.

Note: The values in the Watch window are in color if they were changed by the previous debug operation, and are black if they were not changed by the previous debug operation.

3.18 USING BREAKPOINTS

Breakpoints allow you to specify conditional program halts so that you may observe memory, register or variable values after a run-time execution. You may set breakpoints in either the file (editor) window, the program memory window or the disassembly window.

3.18.1 Breakpoint Setup

There are several ways to set up breakpoints:

1. **Double-click in Gutter** – Double-click in the window gutter next to the line of code where you want the breakpoint. Double-click again to remove the breakpoint.
2. **Pop-up Menu** – Place the cursor over the line of code where you want the breakpoint. Then, right-click to pop up a menu and select Set Breakpoint. Once a breakpoint is set, Set Breakpoint will become Remove Breakpoint and Disable breakpoint. Other options on the pop-up menu under Breakpoints are for deleting, enabling or disabling all breakpoints.
3. **Breakpoint Dialog** – Open the Breakpoint dialog (*Debugger>Breakpoints*) to set, delete, enable or disable breakpoints. You must select a debug tool before this option is available.

3.18.2 Breakpoint Symbols

Symbols relating to breakpoint operation are displayed in the gutter of each supported window. For information on these symbols, see **Section 8.3 “Code Display Window Symbols”**.

3.18.3 Breakpoints in Code

Breakpoints will auto-adjust when source code is modified. This means:

- Breakpoints set on addresses will stay with those addresses, regardless of the code associated with that address.
- Breakpoints set on source code lines will move with those lines of code.

When setting breakpoints in C code, care must be taken if switching between the C code listing and the associated assembly listing when debugging.

- When you set a breakpoint on a line of C code, a breakpoint is set at the first corresponding line of assembly code as well. Both listing displays will show a red breakpoint symbol at the corresponding line of code.
- When you set a breakpoint in the assembly listing on any line of assembly code that represents a single line of C code, a red breakpoint symbol will appear next to the assembly code line and a yellow breakpoint symbol will appear next to the line of C code.
- When you delete a breakpoint in the Program Memory window on any line of assembly code that represents a single line of C code, the red breakpoint symbol will disappear next to the assembly code line but the yellow breakpoint symbol will remain next to the line of C code until all corresponding assembly code breakpoints are deleted.

3.18.4 Breakpoints and MPLAB ICD 2

MPLAB ICD 2 in-circuit debugger supports a limited amount of breakpoints. The number available are displayed in the Breakpoint dialog (*Debugger>Breakpoints*).

Tutorial Step 17:

To run your program again using a breakpoint:

- Select *Debugger>Reset>Processor Reset* to reset your application.
- Set a breakpoint by double-clicking in the gutter next to the following line of code:

```
incf COUNT,F ;increment count
```

- Select *Debugger>Run* to run your program. Program execution should stop at the breakpoint. The values for all items in the Watch window should be 0x00. You may step through your code from this point to watch the values in the Watch window change again.

3.19 CHOOSING A PROGRAMMER

Once you have your code debugged and running smoothly, it is time to program a device and try it in your application.

Select a programmer from *Programmer>Select Programmer*. The programmer you use will depend on the device that you wish to program.

- **PICSTART Plus** and **PICkit 1** development programmers – Used to program most PICmicro MCU devices in a development (nonproduction) environment. See the Readme files in the MPLAB IDE directory for a list of supported devices.
- **MPLAB ICD 2** in-circuit debugger – Used to program certain Flash devices. See the Readme file in the MPLAB IDE directory for a list of supported devices.
- **MPLAB PM3** device programmer – Used to program PIC18 MCU and dsPIC DSC devices. See the Readme file in the MPLAB IDE directory for a list of supported devices.
- **PRO MATE II** device programmer – Used to program most PICmicro MCU devices. See the Readme file in the MPLAB IDE directory for a list of supported devices.

Once you have chosen a programmer, you will see changes in the following on the IDE:

- Programmer menu – Several standard options will be available. Other tool-specific options may also appear. Most of these options also will be available from the right mouse button menu in either the source code (file) window or Program Memory window.
- Programmer toolbar – Several standard options will be available. Other tool-specific options may also appear.
- Status bar – The programmer you have selected should now be shown.

3.20 PROGRAMMING YOUR PART

In general, to program your part:

- Check the values for the configuration bits in your code or in *Configure> Configuration Bits*.
- Check the programming settings in the *Programmer>Settings* dialog.
- Select *Programmer>Program*.

Once you have programmed your part, you are ready to try it in your application. If the part does not function as you had expected, you will need to debug your code and reprogram your part (or program a new part) before placing it in your application again.

3.21 USING MICROCHIP HELP

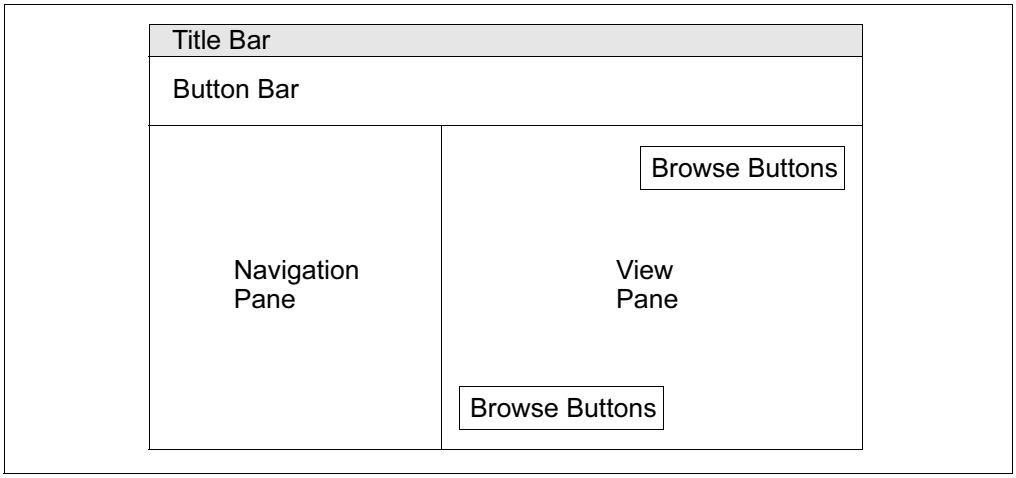
Microchip Technology provides on-line HTML help for MPLAB IDE and each of its development tools. Select *Help>Topics* to bring up a list of available help files in the Help Topics dialog.

- Navigating in the Help Viewer
- Types of Help Available

3.21.1 Navigating in the Help Viewer

The main features of the help viewer are shown in Figure 3-2.

FIGURE 3-2: HELP VIEWER - MAIN FEATURES



3.21.1.1 BUTTON BAR

- Show/Hide – Show or Hide the Left Pane.
- Back – Go back one topic.
- Print – Print currently displayed topic.
- Options – Shows more available options.

3.21.1.2 BROWSE BUTTONS

Browse buttons are located on each displayed topic in the view pane.

- Click **<** to browse to the previous help topic, according to the table of contents.
- Click **>** to browse to the next help topic, according to the table of contents.

3.21.1.3 NAVIGATION PANE

- To view a functionally organized list of all Microchip Help topics, click the **Contents** tab. Click on the + next to a folder to open the folder and see the topics contained within. Click on a topic listed here to view it in the right pane of the viewer.
- To view an alphabetized list of help topics, click on the **Index** tab.
- To search for words in the help topics, click the **Search** tab.
- To create and maintain a list of "favorite" topics, click the **Favorites** tab.
- To close the left pane, click on the **Hide** button (which will change to a **Show** button) on the button bar. To view the left pane, click on the **Show** button (which will change to a **Hide** button.)

3.21.1.4 VIEW PANE

Click on the blue underline text to activate a hyperlink:

- Jump to another topic. To return, click **Back** or browse to other topics.
- Open a pop-up. To close the pop-up, click outside the pop-up box.

3.21.2 Types of Help Available

Tool help is available from the MPLAB IDE Help menu. Help on specific dialog boxes or windows is available from those items. Limitations of a tool are available from the Settings dialog.

3.21.2.1 HELP MENU

Help for each development tool is available from the Help menu. Most help files include the following:

- Tool Overview
- Tutorials/Examples of Use
- General Usage Information
- Troubleshooting Information
- Reference Material

3.21.2.2 DIALOG BOX

Dialog box help is available from a **Help** button on each dialog. Clicking Help displays step-by-step instructions for using a dialog box to complete a task.

3.21.2.3 WINDOW

Window help is available from the **F1** key. Press the **F1** key on your keyboard while a window is active to see information on the contents of that window.

3.21.2.4 LIMITATIONS

Debug tool limitations are available for display after you have selected a device (*Configure>Select Device*) and set the debug tool (*Debugger>Select Tool*). You may then open the Settings dialog (*Debugger>Settings*) and click on the **Limitations** tab.

Both debug and programmer tool limitations are accessible by opening the tool help file (*Help>Topics*) and looking for "Limitations" in the table of contents or index.

Chapter 4. Projects and Workspaces

4.1 INTRODUCTION

Two major features of MPLAB IDE v6.xx are projects and workspaces.

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.

A **workspace** contains information on the selected device, debug tool and/or programmer, open windows and their location and other IDE configuration settings.

The best way to set up your project and its associated workspace is by using the Project Wizard. This will set up one project in one workspace.

To set up more advanced applications, you may set up your project and workspace manually. You can take advantage of the workspace setup and open multiple projects in one workspace. Also, you can tailor each project in a multiple-project workspace to create one part of a larger application (concurrent projects).

If you are working with a single assembly file application, you can use Quickbuild (*Project>Quickbuild*) to assemble your code using MPASM assembler and not create a project. You will, however, have an associated workspace, so your settings will be saved.

Projects

- Using the Project Wizard
- Creating/Updating any Project
- Setting Up a Project Structure – Relative Paths
- Project Folders and Files
- Using A Version-Control System (VCS)
- Setting Up/Changing a Project

Projects and Workspaces

- Using a Single Project and Workspace (like MPLAB IDE v5.xx and earlier)
- Using Multiple Projects in a Single Workspace
- Building an Application without a Project (Quickbuild)

4.2 USING THE PROJECT WIZARD

The project wizard consists of several dialogs which will walk you through setting up your project.

- Project Wizard – Welcome
- Project Wizard – Select Device
- Project Wizard – Select a Language Toolsuite
- Project Wizard – Name Your Project
- Project Wizard – Add Files
- Project Wizard – Summary

4.2.1 Project Wizard – Welcome

Select *Project>Project Wizard* to begin the setup wizard for projects.

Follow the dialogs in the Project Wizard to set up your new project.

You may also use the browse sequence in the Help viewer to view the steps in the wizard.

Click **Next** to continue.

4.2.2 Project Wizard – Select Device

Step 1: Select a Device – The value shown will be either the default MPLAB IDE device or a device that was previously selected using the Select Device dialog. Select a device for your project from the list or start typing in a device name.

Click **Next** to continue.

4.2.3 Project Wizard – Select a Language Toolsuite

Step 2: Select a Language Toolsuite

- Select a language toolsuite for your project from the "Active Toolsuite" drop-down menu. Although third-party language tools may be shown regardless of the device selected, only those Microchip language toolsuites that are installed and work with the selected device are available from this menu. To see all available (installed) toolsuites, check "Show all installed toolsuites."

If you still don't see the desired toolsuite in this list, click **Help! My Toolsuite Isn't Listed!** for more information.

- A list of tools in the selected toolsuite are shown in a box under "Toolsuite Contents". A tool with a red "X" preceding it is not installed or the path to the executable is not known to MPLAB IDE. To assign or check assignments of tools to executable files, click on the tool to show the executable and path under "Location of Selected Tool". Type in this text box to enter or change the assignment, or click **Browse** to find the executable.

Click **Next** to continue.

4.2.4 Project Wizard – Name Your Project

Step 3: Name Your Project – Enter a name and location for your new project. To set the directory:

- Type in the path to an existing directory or type in the path to a new directory, in which case you will be prompted to create the directory when you click **Next**.
- Click **Browse** to browse to an existing directory or to one level above where you wish to place a new directory. Click **OK** in the Browse for Folder dialog. Complete the path if you are creating a new directory and then click **Next**. You will be prompted to create the directory if it does not exist.

4.2.5 Project Wizard – Add Files

Step 4: Add Any Existing Files to Your Project – If you already have files that you would like to add to the new project, select them now:

- Choose the file(s) to add. Click on a file name to select that file. Ctrl-Click to select more than one file. Click **Add>>** to list file(s) to be added to the new project.
- To remove file(s) from the list, click on the file name(s) to select and then click **Remove**.

If you wish to copy, and not move, the select files to the new project, check the checkbox that appears next to the files that have been added.

Click **Next** to continue.

4.2.6 Project Wizard – Summary

Check the summarized information on the project. If anything is incorrect, use **Back** to return to the dialog you need and change the information.

Click **Finish** when you are satisfied with the project setup.

4.3 CREATING/UPDATING ANY PROJECT

The process for creating or updating a project is the same regardless of whether you are using a single- or multiple-project workspace.

1. Create/open a project in the current workspace. The project will appear in the Project window.
   - Create a new project by selecting *Project>New*. Enter the new project name and location in the dialog.
   - Open an existing project by selecting *Project>Open*.
2. Assign/change paths for project files by selecting *Project>Build Options>Project*, General tab.
3. Assign/change language tools in a project by first selecting *Project>Set Language Tool Locations* to specify the path to each language tool. Then select *Project>Set Language Toolsuite* to set the toolsuite for the project.
4. Setup/change language tool properties for the project by selecting *Project>Build Options>Project* or by right-clicking on the project name in the Project window and selecting Build Options. Then click the appropriate language tool tab.

5. Enter files in the project by selecting *Project>Add Files* to Project, by right-clicking on the project name in the Project window and selecting Add Files to Project, or by right-clicking on the type of file in the Project window and selecting Add Files to enter that specific type of file.

Note:	You can add a file to a project, even if the file does not yet exist. This is useful for structuring a project before you actually developed the files. Just remember to create the files before you build the project or, obviously, the build will fail.

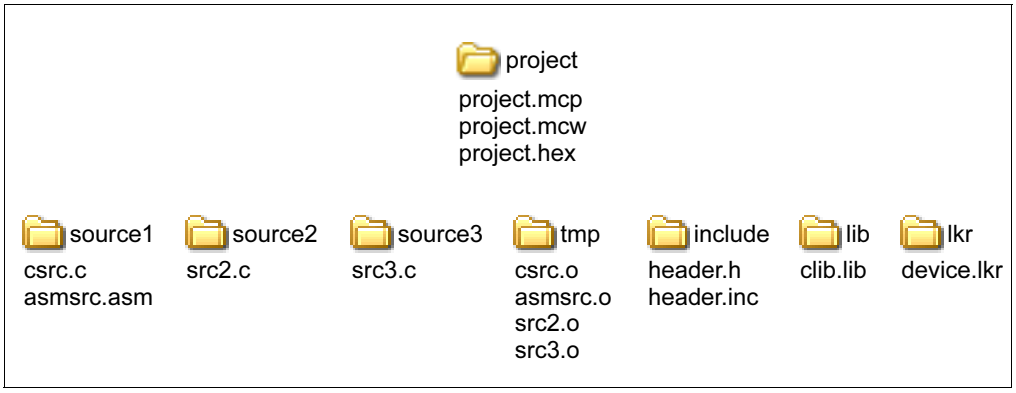
6. Delete files in the project by selecting *Project>Remove File From Project* or by right-clicking on the file in the Project window and selecting Remove.
7. If you want language tool properties for individual files to differ from those set for the project, select *Project>Build Options>filename*, where filename is the name of the individual file, or right-click on the file name in the Project window and select Build Options. These setting will apply only to the individual file.

4.4 SETTING UP A PROJECT STRUCTURE – RELATIVE PATHS

The following is an example of recommended project directory structure for using relative paths on the Build Options General tab. Keeping all project subfolders parallel allows for ease of grouping of intermediates, include, library and linker script files

Note:	Make sure you use unique names for your source code, or object files will be overwritten. E.g., `file.asm` is assembled producing `file.o`, and then `file.c` is compiled producing `file.o` which overwrites the first `file.o`.

FIGURE 4-1: RECOMMENDED PROJECT STRUCTURE



Project>Build Options>Project, General Tab

Output Directory: `C:\project`

Intermediates Directory: `..\tmp`

Assembler Include Path: `..\include`

Include Path: `..\include`

Libraries Path: `..\lib`

Linker Script Path: `..\lkr`

4.5 PROJECT FOLDERS AND FILES

When a project is open, all of its files will be assigned to one of the following folders. The filter associated with a folder determines what files it will contain, as does the order of the folders. If two folders can both accept the same file, the file in the first folder will be used.

- Source Files – These are the only files that the toolsuite will accept as input to its file commands. Example files: `file.asm`, `file.c`.
- Header Files – MPLAB 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. Example files: `header.inc`, `header.h`.
- 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. Example files: `ofile.o`.
- Library Files – The toolsuite should take all of the files in this folder, as well as the object files, and include them in the final link step. Example files: `fft.o`.
- Linker Scripts – There should be only one file in this folder. If the user has 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. Example files: `linker.lkr`, `linker.gld`.
- Intermediary Files (not displayed in Project Window) – The project maintains this folder, but does not display it to the user. As source files are assembled and compiled, links are created to the resulting object files in this folder. Each and every object file in this folder should be included in the link step, in addition to the contents of the Object Files and Library Files folder.

  Note: These files will be destroyed when the user executes the Clean command.

- Output Files (not displayed in Project Window) – This folder is maintained by the project but not displayed. These are the final output files that result from the link step. A Clean will delete all of these files.
- Other Files – Any file that does not fit into any of the other categories will end up in this one. Example files: `file.txt`, `stim.sts`.

4.6 USING A VERSION-CONTROL SYSTEM (VCS)

If you want to use a version-control system when developing your application code, you may now do so in MPLAB IDE using Projects.

- Select *Project>Version-Control* to set up the version-control application for use with MPLAB IDE. (See **Section 9.25 "Version-Control Dialog"**)

Note: Folders/files must already exist in the VCS for MPLAB IDE to use them in a project.

To save an entire project (and workspace) in the VCS:

- Outside of MPLAB IDE, check in the entire project. When you next check it out and open it in MPLAB IDE, VCS-specific information will be available in the MPLAB IDE (see below.)

To save only some project files in the VCS:

- If you have common files contained in the VCS that you want to use in a specific project, you will need to check out those files outside of MPLAB IDE.
- Add the checked-out files to your project. You should now see VCS-specific information in the MPLAB IDE (see below.)

Once your project is set up for a VCS, you should notice the following changes to MPLAB IDE:

- The Project window will show version-control information for the project files, i.e., display check-out status.
- Version-control commands for VCS files may be selected by right-clicking on a file in the project window.
- You may set up refresh options in the Project-Display Preferences Dialog, so you can be aware of file check-in/check-out status.
- The output window will contain a Version Control tab displaying activity between MPLAB IDE and the version-control system.

Additional information on using version-control systems currently supported is listed below.

4.6.1 Microsoft Visual Source Safe

Having some knowledge of VSS and MPLAB IDE is necessary to use VSS from within MPLAB IDE.

For more information on this version-control system, please see the related web site.

- Microsoft Visual Source Safe: http://www.microsoft.com

Certain VSS files are required to be available locally (on your PC) for VSS to work with MPLAB IDE. If you are using VSS on a network drive, you will need to run `NETSETUP.EXE` to install the proper files on your PC.

4.6.2 PVCS

Having some knowledge of PVCS and MPLAB IDE is necessary to use PVCS from within MPLAB IDE.

For more information on this version-control system, please see the related web site.

- PVCS: http://www.merant.com/Products/ECM/VM/home.asp

4.6.3 CVS

Having some knowledge of CVS and MPLAB IDE is necessary to use CVS from within MPLAB IDE.

For more information on this version-control system, please see the related web site.

- CVS: https://www.cvshome.org

4.6.3.1 IMPORTING FILES

If your source files aren't already in the repository, you will have to import them to get started. This creates the directory structure in the repository and places all of the files there as well.

CAUTION

Do not place the workspace (MCW) or build-state (MCS) files on the repository. MPLAB IDE needs to be able to write to these files while running.

An example of an import command is given below. The `-d` argument is not given, the assumption being that you have assiged the appropriate value to the `CVSROOT` environment variable. You want to be inside of the very top directory of your source tree when you give this command. The command will automatically recurse into subdirectories.

```
$ cd TopOfSrcTree
$ cvs import -m "message" DirWithinCVSRoot VendorTag ReleaseTag
```

4.6.3.2 CHECKING OUT FILES

Even if you have your source files locally, you will have to perform a `cvs checkout` once (first-time use) before you can check out files from CVS in MPLAB IDE. This checkout creates the management files that CVS requires to perform commands on various files. All of these management files are stored in a hidden subdirectory of each directory in the source tree. Each of these hidden subdirectories is called `CVS`.

Trying to do a checkout overtop of existing source files without these supporting CVS directories will cause CVS to complain. You should either delete or move away the local copy of the sources before performing the checkout.

An example of performing the checkout is listed below:

```
$ cd DirABOVEWhereIWantMySrcTree
$ cvs checkout MyProject
cvs server: Updating MyProject
U MyProject/main.c
U MyProject/such-and-such-a-file.c
U MyProject/foo.h
```

It is important that you give this command from *above* the directory where you want your source tree to reside, because CVS will actually create the entire source tree within the current working directory.

4.6.3.3 ADDING A NEW DIRECTORY

You can use the `cvs add` command from within MPLAB IDE to add new source files as long as the required directory structure already exists in the repository. The `add` command will NOT work on a file if any of the required directories don't exist in the repository.

To create the directory structure in the repository, you must use the `cvs import` command once again. An example of this is as follows:

```
$ cd TopOfSrcTree
$ cvs import -m "message" MyProject/NewSubDir VendorTag ReleaseTag
N MyProject/NewSubDir/yet-another-file.h

No conflicts created by this import
```

Once again, you should give this command from within the directory that is the top-level directory of your source tree. But this time, when you specify the directory within the repository, you should give the path in the repository that will represent the new directory that you are adding. (That is what `MyProject/NewSubDir` represents in this example.)

4.7 SETTING UP/CHANGING A PROJECT

You can use the following steps to create a new project manually (using the Project Setup Wizard is recommended) or change an existing project's settings.

- Create a New Project
- New Project Window
- Setting Up a Project – Set Up Language Tools
- Setting Up a Project – Choose Toolsuite
- Setting Up a Project – Add Files
- Setting Up a Project – Set Build Options
- Setting Up a Project – Save and Build

4.7.1 Create a New Project

To create a new project in the current workspace:

- Using Windows Explorer, create a directory for your project (e.g. `C:\Proj1`).
- In MPLAB IDE, select *Project>New*. The New Project dialog will open asking you the name and location of the new project.
- Enter the Project name, e.g., `Proj1`.
- Enter the path to your new project. Click Browse to locate your directory.
- Hit **OK**. The Save Workspace As dialog will open.

To save the workspace with the new project in it:

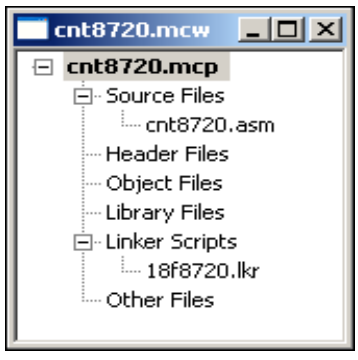
- Enter the Workspace name, i.e., `cnt452`.
- Hit **OK**.

MPLAB IDE will create a project file with the extension `.mcp` and a workspace file with the extension `.mcw`.

4.7.2 New Project Window

If it is not already open, open the Project window by selecting *View>Project*. The workspace name should be visible in the top bar of the Project window. The name of the project should be visible inside the Project window at the top of the display. A tree structure listing file types will appear below.

FIGURE 4-2: EXAMPLE PROJECT WINDOW



4.7.3 Setting Up a Project – Set Up Language Tools

To add language tools for use in the project, select *Project>Set Language Tool Locations* or right-click on the project name in the Project window and select Set Language Tool Locations from the pop-up menu.

- Click on a registered tool in a toolsuite to select it.
- Type in the “Location” or browse to the location of the tool executable by clicking **Browse**.
- Repeat this for each tool you wish to use in the project.
- When you are finished, click **OK**.

Now you will need to assign a toolsuite to the project.

4.7.4 Setting Up a Project – Choose Toolsuite

To select a toolsuite for use in the project, select *Project>Set Language Toolsuite* or right-click on the project name in the Project window and select Set Language Toolsuite from the pop-up menu.

- Select your project toolsuite from the “Active Toolsuite” list.
- Click **OK**.

The toolsuite you use will depend on the device for which your are developing code.

- Microchip language tools – for PICmicro MCU, Microchip memory and KEELOQ devices
- dsPIC language tools – for dsPIC devices
- Third Party tools – for various Microchip devices

4.7.5 Setting Up a Project – Add Files

You will now add files to the project. The basic types of files that are included in a project may be seen by clicking on the “+” sign to the left of the project name in the Project window.

Note:	You may enter the name of a file that does not yet exist. However, before you can build the project, you must create the file with a text editor such as MPLAB Editor.

To insert files in the project, select *Project>Add Files* to Project or right-click on the project name in the Project window and select “Add Files” from the pop-up menu. Additionally, if you know the type of file you will be adding, you may right-click on that type in the project window and select “Add Files” from the pop-up menu.

- The Add Files to Project dialog will open.
- Click on a file or enter its name to select it for insertion. If you do not see your file(s) listed, open the drop-down list under Files of Type to select the appropriate type of file. If you are unsure, select All Files (*.*).
- To select multiple files for insertion, use either Shift-Click to select a continuous range or Control-Click to select several discontinuous files.
- Click **Open**.

4.7.6 Setting Up a Project – Set Build Options

MPLAB IDE has default settings for tools and files contained in a project. However, you may want or need to modify these settings.

To set build options, select *Project>Build Options>Project* or right-click on the project name in the Project window and select Build Options from the pop-up menu.

- The Build Options dialog will open.
- Click the **General** Tab and enter or **Browse** to paths for output files, include files library files or linker script files.

> **Note:** These are MPLAB IDE paths. Not all language tools use this information.

- Click a specific language tool tab (e.g., MPASM Assembler) and set up operational features for that tool.
- Click **OK**.

To override project settings for a particular project file, e.g., `ProjFile1.asm`, select `Project>Build Options>ProjFile1.asm` or right-click on `ProjFile1.asm` in the Project window and select Build Options from the pop-up menu.

- The File Settings dialog will open.
- Click a specific language tool tab (e.g., MPASM Assembler) and set up operational features for that tool.
- Click **OK**.

4.7.7 Setting Up a Project – Save and Build

At this point you should save your project by selecting *Project>Save* or right-clicking on the project name in the Project window and selecting "Save Project".

Now you are ready to build your project. Right-click on the project name in the Project window and select "Build All" from the pop-up menu.

- For MPASM assembler, a status window will open showing you the progress and final outcome of the build. It will close when complete.
- The Output window will also open. This will contain information about the build, including any errors encountered

If your project does not build successfully, please check the following items and then build the project again:

- Check the spelling and format of any code you entered in the editor window. If there are reported errors in the Output window, double-clicking on an error will indicate the corresponding line in your source code with a green arrow in the gutter of the source code window.
- Check that the correct language tool is being used for the project and for the project files.

Upon a successful build, the debug file (`*.cod` or `*.cof`) generated by the language tool will be loaded. This file allows you to debug using your source code and view your variables symbolically in Watch windows.

4.8 USING A SINGLE PROJECT AND WORKSPACE

The most common configuration for application development in MPLAB IDE to is have one project in one workspace. If you wish to use an MPLAB IDE v5.xx-style project (not worry about workspaces), select *Configure>Settings*, **Projects** tab and check “Use one-to-one project-workspace model”.

1. Select the device you will use for development (*Configure>Select Device*).
2. Create/open and set up a project in the current workspace.
3. Build the project.
   - To build the whole project, select either *Project>Build All* or right-click on the project name in the Project window and select Build All.
   - To build only the files that have changed in a project, select either *Project>Make* or right-click on the project name in the Project window and select Make.

Your application is now ready to debug. When your debug session is complete:

- Close the project and its associated workspace by selecting *Project>Save Project* and then *Project>Close*.
- Reopen the project and its associated workspace by selecting *Project>Open*.

4.9 USING MULTIPLE PROJECTS IN A SINGLE WORKSPACE

You may wish to have multiple projects in one workspace. This makes use of one workspace setup (i.e., selected device, debug tool, window setups, etc.) for many projects.

Some applications can benefit from using more than one project in a workspace. Consider the following examples.

EXAMPLE 4-1: BOOTLOADER

A bootloader has its own source code and its only function is to download and program a new version of an application in flash memory. The bootloader would always exist in a fixed location in program memory, and would be executed by a special function of the application. The bootloader would then download a program through a communications port and overwrite the existing application with an new version. Both the bootloader and the application share the same settings for the dsPIC30F device since they will exist on the same part, but they are independent pieces of code. The application knows only the entry address for executing the bootloader. The bootloader knows only the start address for the upgrade code.

EXAMPLE 4-2: TESTS

Another use of multiple projects is when developing a series of tests for a single application. These tests might exist in the same locations in program memory, but never be loaded simultaneously.

EXAMPLE 4-3: APPLICATION CONFIGURATION

A third possibility is that the application has multiple configurations for the end product. An application might be sold under multiple brands or with alternate languages. Each project could contain the branding information or language strings for the application in a separate resource file.

4.9.1 Setting Up Multiple Projects

To set up the workspace to use multiple projects:

1. Select the device you will use for development (*Configure>Select Device*).
2. Create/open and set up the first project in the current workspace.
3. Make sure that the following options are selected. Then repeat #2 for each project you wish to add to the workspace.
   - *Configure>Settings*, **Projects** tab: uncheck "Use one-to-one project-workspace model"
   - *Configure>Settings*, **Program Loading** tab: check "Clear program memory upon loading a program"
4. Make a project active by selecting *Project>Set Active Project>projectname.mcp*, where `projectname.mcp` is the project name, or right-click on the project in the Project window and select Set As Active Project. The project name will appear bolded in the Project window.
5. Build the active project.
   - To build the whole project, right-click on the project name in the Project window and select Build All.
   - To build only the files that have changed in a project, right-click on the project name in the Project window and select Make.

Your application is now ready to debug. When your debug session is complete:

- Save the current active project by selecting *Project>Save Project* or by right-clicking on the project in the Project window and selecting Save Project.
- Select another project as the active project, build this project and debug.
- Remove the active project from the workspace by selecting *Project>Close* or by right-clicking on the project in the Project window and selecting Close Project.
- Close and save all projects and the associated workspace by selecting *File>Close Workspace*. You will be prompted to save.
- Reopen all projects and the associated workspace by selecting *File>Open Workspace*.

4.9.2 Determining an Active Project

An active project is the project in the workspace that the following Project menu and toolbar commands will act on:

- Clean
- Find in Project Files
- Save Project
- Add Files to Project
- Select Language Toolsuite

All other Project menu, toolbar and right mouse menu commands will act on the project specified when the command is chosen. As an example, if you right-click on a project in the Project window that is not the active project, you can build this project by selecting Build All or Make and this project's build results will be loaded into program memory.

To avoid confusion, it is recommended that you make active a project you will be developing and building.

The last project opened in a workspace will automatically be set as the active project. To set a project as the active project, select *Project>Set Active Project> project-name.mcp*, where projectname.mcp is the project name, or right-click on the project in the Project window and select Set As Active Project. The project name will appear in bold in the Project window.

4.9.3 Using Concurrent Projects

You may wish to have multiple projects that create code to occupy different regions of program memory, thus creating one application. Projects used this way are considered concurrent projects.

To create a concurrent-projects workspace:

1. Begin with a multiproject workspace.
2. Uncheck the “Clear program memory upon loading a program” option on the *Configure>Settings*, **Program Loading** tab.

4.10 BUILDING AN APPLICATION WITHOUT A PROJECT

If you only want to assemble a single assembly file with MPASM assembler, you do not need to use projects but may use Quickbuild instead.

1. Select the device you will use for development (*Configure>Select Device*).
2. Make sure Quickbuild is active (*Project>Set Active Project>None* (Quickbuild mode)).
3. Open a file (editor) window and enter your assembly code (*File>New*), open an existing assembly file (*File>Open*) or click on an open edit window with your code to make the window active.

   Note: If a file window is not active, the Quickbuild option will be grayed out on the menu.

4. Select *Project>Quickbuild* file.asm to assemble your application, where file.asm is the name of your active assembly file.

Your application is now ready to debug. When your debug session is complete:

- Save the current MPLAB IDE configuration to a file by selecting *File>Save Workspace*.
- Restore the MPLAB IDE configuration for another debug session by selecting *File>Open Workspace*.

NOTES:

Chapter 5. Integrated Tools

5.1 INTRODUCTION

MPLAB IDE is designed to work with many Microchip and third party developments tools.

Language Tools

- Language Toolsuites
- Microchip Language Tools
- Third Party Language Tools

Software/Hardware Tools

- Editors
- Simulators
- In-Circuit Emulators
- In-Circuit Debuggers
- Programmers
- Third Party Tools

5.2 LANGUAGE TOOLSUITES

MPLAB IDE supports many language toolsuites. Integrated into MPLAB IDE is the Microchip MPASM Toolsuite, but many others can be used, including the Microchip C17, C18 and C30 Toolsuites, as well as language tools from HI-TECH, IAR, CCS, microEngineering Labs and Byte Craft. These are integrated into MPLAB IDE in two ways: using ”plug ins” designed by the manufacturer, and by older style “.MTC” files that can be customized for any language toolsuite.

Language Suite Plug-Ins

Vendors of language tools can write a “plug-in” that can be installed into MPLAB IDE to provide custom dialogs to access the various tool build options. This is the preferred method of support.

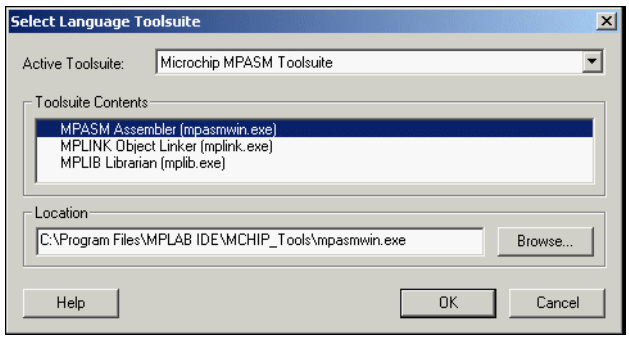
.MTC File Interface

A language tool can be integrated in a generic manner using a special file with the extension `.mtc`. These files are then integrated into a dialog to provide check box access to the various build options of the language tool. Using these files, most of the features of the language toolsuite can be controlled, but all may not be available.

5.2.1 Selecting the Language Toolsuite

Once a workspace is opened, a toolsuite can be selected from *Project>Select Language Toolsuite*. In some cases, MPLAB IDE will know where the various executables for the toolsuite are located, but sometimes these will need to be set manually before MPLAB IDE can use the tools.

FIGURE 5-1: SELECTING A TOOLSUITE



If the selected toolsuite has red X's to the left of the toolsuite components, that means that MPLAB IDE does not know where the executables are. You will have to enter the full path and executable name by typing it in or browsing to the proper executable.

5.2.2 Locating Microchip Language Toolsuites

Microchip toolsuites and their locations are described below.

Microchip MPASM Toolsuite

This toolsuite includes the language tools MPASM assembler, MPLINK object linker and MPLIB object librarian. The executables for these files are `mpasmwin.exe`, `mplink.exe` and `mplib.exe` and are located in the `mchip_tools` subdirectory of the main MPLAB IDE installation directory.For more information on these tools, see Microchip PICmicro Language Tools.

Microchip C17 Toolsuite (Obsolete Product)

This toolsuite includes all the language tools from the Microchip MPASM Toolsuite as well as the MPLAB C17 C compiler. MPLAB C17 is a separate installation from MPLAB IDE. The executable for MPLAB C17 is `mcc.exe` and is installed in `c:\mcc\bin` by default.For more information on these tools, see Microchip PICmicro Language Tools.

Microchip C18 Toolsuite

This toolsuite includes all the language tools from the Microchip MPASM Toolsuite as well as the MPLAB C18 C compiler. MPLAB C18 is a separate installation from MPLAB IDE. The executable for MPLAB C18 is `mcc18.exe` and is installed in `c:\mcc18\bin` by default. For more information on these tools, see Microchip PICmicro Language Tools.

Microchip ASM30 Toolsuite

This toolsuite includes the language tools MPLAB ASM30 assembler, MPLAB LINK30 object linker and MPLAB LIB30 object librarian. The executables for these files are `pic30-as.exe`, `pic30-ld.exe` and `pic30-ar.exe` and are located in the `dspic_tools\bin` subdirectory of the main MPLAB IDE installation directory. For more information on these tools, see Microchip dsPIC Language Tools.

Microchip C30 Toolsuite

This toolsuite includes all the language tools from the Microchip ASM30 Toolsuite as well as the MPLAB C30 C compiler. MPLAB C30 is a separate installation from MPLAB IDE. The executable for MPLAB C30 is `pic30-gcc.exe` and is installed in `c:\pic30\bin` by default. For more information on these tools, see Microchip dsPIC Language Tools.

5.3 MICROCHIP LANGUAGE TOOLS

Currently-supported language tools from Microchip Technology are:

- Microchip PICmicro Language Tools – supports PICmicro MCUs, KEELOQ and memory devices.
- Microchip dsPIC Language Tools – supports dsPIC DSC devices.

Other language tools supporting Microchip devices may be available from third parties. Please see our web site for a complete list of these suppliers.

5.3.1 Microchip PICmicro Language Tools

The Microchip toolsuite supports PICmicro microcontrollers (MCU), Microchip memory and KEELOQ devices. (See the related Readme file in the MPLAB IDE installation directory for a list of supported devices.)

TABLE 5-1: MICROCHIP TOOLSUITE SUPPORT

Tool	Name	Executable
C Compiler – PIC18X devices	MPLAB C18 C Compiler	`mcc18`
C Compiler – PIC17C devices **(Obsolete)**	MPLAB C17 C Compiler	`mcc`
Assembler	MPASM Assembler	`mpasmwin`
Linker	MPLINK Object Linker	`mplink`
Librarian	MPLIB Object Librarian	`mplib`

Additional command-line tool executables and utilities are available. See language tool documentation for more information.

Help Files

From the MPLAB IDE Help menu, select Topics and then select under Language Tools either:

- MPASM Assembler
- MPLINK Object Linker (Including MPLIB Object Librarian)

Documentation

Please find the following C compiler documentation on the MPLAB IDE CD-ROM (DS51123) or the Microchip web site.

- MPLAB C18 C Compiler Getting Started (DS51295)
- MPLAB C18 C Compiler User's Guide (DS51288)
- MPLAB C18 C Compiler Libraries (DS51297)
- MPLAB C17 C Compiler User's Guide (DS51290)
- MPLAB C17 C Compiler Libraries (DS51296)

5.3.2 Microchip dsPIC Language Tools

The Microchip dsPIC toolsuite supports dsPIC digital signal controller (DSC) devices. (See the related Readme file in the MPLAB IDE installation directory for a list of supported devices.)

TABLE 5-2: MICROCHIP dsPIC TOOLSUITE

Tool	Name	Executable
C Compiler	MPLAB C30	`pic30-gcc`
Assembler	MPLAB ASM30	`pic30-as`
Linker	MPLAB LINK30	`pic30-ld`
Archiver/Librarian	MPLAB LIB30	`pic30-ar`

Additional command-line utilities are available. See language tool documentation for more information.

Help Files

From the MPLAB IDE Help menu, select Topics and then select under Language Tools either:

- MPLAB ASM30
- MPLAB LINK30 (includes MPLAB LIB30 and other Utilities)

Documentation

Please find the following dsPIC DSC documentation on the MPLAB IDE CD-ROM (DS51123) or the Microchip web site.

- dsPIC30F Family Reference Manual (DS70046)
- dsPIC30F Programmer's Reference Manual (DS70030)

Please find the following dsPIC language tools documentation on the MPLAB IDE CD-ROM (DS51123) or the Microchip web site.

- dsPIC Language Tools Getting Started (DS70094)
- MPLAB ASM30, MPLAB LINK30 and Utilities User's Guide (DS51317)
- MPLAB C30 C Compiler User's Guide (DS51284)
- dsPIC30F Language Tools Libraries (DS51456)
- dsPIC30F Language Tools Quick Reference Card (DS51322)

5.4 THIRD PARTY LANGUAGE TOOLS

Currently-supported third-party language tools in MPLAB IDE are:

- B Knudsen Data Language Tools
- Byte Craft Language Tools
- CCS Language Tools
- HI-TECH Language Tools
- IAR Language Tools
- microEngineering Labs Language Tools

Other language tools supporting Microchip devices may be available from third parties. Please see the Microchip web site for a complete list of these suppliers.

5.4.1 B Knudsen Data Language Tools

MPLAB IDE supports the following B Knudsen Data language tools:

Tool	Name	Executable
C Compiler for PIC12/14/16 devices	CC5X	`cc5x`
C Compiler for PIC18 devices	CC8E	`cc8e`

For more information, see the B Knudsen Data web site (http://www.bknd.com).

5.4.2 Byte Craft Language Tools

MPLAB IDE supports the following Byte Craft language tools:

Tool	Name	Executable
Assembler/C Compiler for PIC12/14/16/17 devices	MPC	`mpc`

For more information, see the Byte Craft web site (http://www.bytecraft.com).

5.4.3 CCS Language Tools

MPLAB IDE supports the following Custom Computer Services (CCS) language tools:

Tool	Name	Executable
Command-line C Compiler for PIC12/16 (12-bit) devices	PCB	`pcb`
Command-line C Compiler for PIC12/14/16 (14-bit) devices	PCM	`pcm`
Command-line C Compiler for PIC18 devices	PCH	`pch`
Windows C Compiler for PIC12/14/16 devices	PCW	`pcw`
Windows C Compiler for PIC12/14/16/18 devices (with add-on for dsPIC devices)	PCWH	`pcwh`

For more information, see the CCS web site (http://www.ccsinfo.com).

5.4.4 HI-TECH Language Tools

MPLAB IDE supports the following HI-TECH language tools:

Tool	Name	Executable
C Compiler, Assembler, Linker – PIC12/14/16 devices	PICC	picc
C Compiler, Assembler, Linker – PIC18 devices	PICC18	picc18
C Compiler, Assembler, Linker – limited memory	PICC Lite	picl
C Compiler, Assembler, Linker – limited functionality	PICC Demo	picc
C Compiler, Assembler, Linker – dsPIC devices	dsPICC	dspicc

To set up HI-TECH language tools for use in your project:

- Select *Project>Set Language Toolsuite*. In the Select Language Toolsuite dialog, select either the HI-TECH PICC Toolsuite for support of PIC12/14/16 devices or HI-TECH PICC18 Toolsuite for support of PIC18 devices. Click **OK** or continue by clicking **Set Tool Locations**.

  Note: PICC Lite or Demo users, select the HI-TECH PICC Toolsuite.

- If you closed the previous dialog, select *Project>Set Language Tool Locations*. In the Set Language Tool Location dialog, find the toolsuite you will be using in the list and click on the first tool in this suite you will be using.
- View the executable path in the Location of Selected Tool text box. If there is no location/path listed, enter one or browse for one.

  Note: PICC Lite users, enter/browse to the corresponding executable for the regular HI-TECH tool, i.e., instead of `picc.exe`, use `picl.exe`.

- Click **OK** until all language tool dialogs are closed.

For more information, see the HI-TECH web site (http://www.htsoft.com).

5.4.5 IAR Language Tools

MPLAB IDE supports the following IAR language tools:

Tool	Name	Executable
C Compiler for PIC16/17 devices	IAR PICmicro 16/17 C Compiler	iccpic
Assembler for PIC16/17 devices	IAR PICmicro 16/17 C Compiler	apic
C/EC++ Compiler for PIC18 devices	IAR PICmicro PIC18 C Compiler	iccpic18
Assembler for PIC18 devices	IAR PICmicro PIC18 C Compiler	apic18
C/EC++ Compiler for dsPIC devices	IAR dsPIC C Compiler	iccdspic
Assembler for dsPIC devices	IAR dsPIC C Compiler	adspic
Linker	IAR Linker	xlink

For more information, see the IAR web site (http://www.iar.com).

5.4.6 microEngineering Labs Language Tools

MPLAB IDE supports the following microEngineering Labs language tools:

Tool	Name	Executable
Basic Compiler, most PICmicro devices	PicBasic Compiler	pb
Basic Compiler, all PICmicro devices	PicBasic Pro Compiler	pbp

For more information, see the microEngineering Labs web site (http://www.melabs.com).

5.5 EDITORS

MPLAB IDE supports the MPLAB Editor.

Help Files

From the MPLAB IDE Help menu, select Topics and then select under System:

- MPLAB Editor

Related Menus

- File
- Edit

5.6 SIMULATORS

The following Microchip simulator is supported:

Simulator Name	Devices Simulated*
MPLAB SIM Simulator	PICmicro MCUs, dsPIC DSCs

* See the related Readme file in the MPLAB IDE installation directory for a list of supported devices.

Help Files

From the MPLAB IDE Help menu, select Topics and then select under Debugger:

- MPLAB SIM

Related Topics

- Third Party Tools

5.7 IN-CIRCUIT EMULATORS

The following Microchip in-circuit emulators are supported:

Emulator Name	Devices Emulated*
MPLAB ICE 2000	PICmicro MCUs
MPLAB ICE 4000	PIC18 MCUs, dsPIC DSCs

* See the related Readme file in the MPLAB IDE installation directory for a list of supported devices.

Help Files

From the MPLAB IDE Help menu, select Topics and then select under Debuggers:

- MPLAB ICE 2000
- MPLAB ICE 4000

Related Topics

- Third Party Tools

5.8 IN-CIRCUIT DEBUGGERS

The following Microchip in-circuit debuggers are supported:

Debugger Name	Devices Supported*
MPLAB ICD 2	PICmicro Flash MCUs, dsPIC Flash DSCs

* See the related Readme file in the MPLAB IDE installation directory for a list of supported devices.

Help Files

From the MPLAB IDE Help menu, select Topics and then select under Debuggers:

- MPLAB ICD 2

5.9 PROGRAMMERS

The following Microchip programmers are supported:

Programmer Name	Devices Programmed*
MPLAB PM3 Device Programmer	PICmicro MCUs, dsPIC DSCs
PRO MATE II Device Programmer	PICmicro MCUs, Memory devices, KEELOQ devices
PICSTART Plus Development Programmer	Most PICmicro MCUs
PICkit 1 Development Programmer	Selected PICmicro MCUs
MPLAB ICD 2 In-Circuit Debugger	PICmicro Flash MCUs, dsPIC Flash DSCs

* See the related Readme file in the MPLAB IDE installation directory for a list of supported devices.

Help Files

From the MPLAB IDE Help menu, select Topics and then select under Programmers either:

- MPLAB PM3
- PRO MATE II
- PICSTART Plus

Additionally, under Debuggers, you can select:

- MPLAB ICD 2

Documentation

Please find the following documentation on the MPLAB IDE CD-ROM (DS51123) or the Microchip web site.

- PICkit 1 Flash Starter Kit User's Guide (DS40051)

Related Topics

- Third Party Tools

5.10 THIRD PARTY TOOLS

Other development tools supporting Microchip devices may be available from third parties. Please see the Microchip web site (www.microchip.com) for a complete list of these suppliers.

Chapter 6. MPLAB IDE Troubleshooting

6.1 INTRODUCTION

This section is designed to help you troubleshoot any problems, errors or issues you encounter while using MPLAB IDE. If none of this information helps you, please see the Preface for ways to contact Microchip Technology.

Topics covered are:

- Common Problems/FAQ – A list of the most commonly-encountered problems and frequently asked questions.
- Error Messages – A list of error messages produced by this tool.
- Limitations – Limitations of the tool.

6.2 COMMON PROBLEMS/FAQ

- I cannot get my language tools to work with MPLAB IDE.
- I can't see all of the characters in the displays.
- I have recently ported my projects from one PC to another. They don't work or worked for a while but now do not.
- Some of the items on the Project menu and toolbar are grayed out.
- When stepping through high-level code, if I step into a function with no source code, a bunch of Program Memory windows start popping up.
- I cannot find all of my object files.
- My program keeps resetting. What's going on?
- I can't Halt Animate.
- I tried to Step Over / Step Out of a function/subroutine, and now my program is hung up.

I cannot get my language tools to work with MPLAB IDE.

MPLAB IDE v6.xx is significantly different from MPLAB IDE v5.xx and lower. You will probably need an upgrade to work with this new version. Please go to our web site or third party web site to get the latest version of your language tool.

I can't see all of the characters in the displays.

When using a System Display Font Size of Large Fonts (125% normal), and selecting an Italic font, some characters may display truncated on the top-right border. The Tetradyne SourceView editor control used in the source editor supports these fonts: Courier, Courier New and Fixedsys.

I have recently ported my projects from one PC to another. They don't work or worked for a while but now do not.

Project files (.mcp) are portable. Workspace files (.mcw) may or may not be portable. If you try to move your projects and workspaces between PC's with different operating systems or different hardware configurations, you will need to update your workspace or create a new one.

See Saved Information for more on what is saved in these files.

Some of the items on the Project menu and toolbar are grayed out.

Certain context-sensitive Project menu and toolbar items need an active project set before they will become available. Check *Project>Set Active Project* and make sure None (Quickbuild) is not chosen. An active project's name will be bolded in the Project window.

When stepping through high-level code, if I step into a function with no source code, a bunch of Program Memory windows start popping up.

MPLAB IDE will open another Program Memory window if you step into an instruction that does not have an associated line of source code (e.g., library routines) and the program memory window does not have focus.

To avoid this, first select *Configure>Settings*, **Debugger** tab. Then check "Browse for source if file not found" and "Show disassembly if source is unavailable".

I cannot find all of my object files.

Make sure you have your project directory structure set up in a parallel manor, or your object files will be grouped in the same directory as your source files instead of the project tmp directory.

Also, make sure your source files use unique names (`file1.asm` and `file2.c`, not `file.asm` and `file.c`) so that they will not produce object files with the same name, which will overwrite each other.

My program keeps resetting. What's going on?

Check the configuration bits settings (*Configure>Configuration Bits*) for your selected device. Some reset functions (such as Watchdog Timer Reset) are enabled by default.

I can't Halt Animate.

To Halt Animate, use the menu option *Debugger>Halt* instead of the toolbar Halt or **F5**.

I tried to Step Over / Step Out of a function/subroutine, and now my program is hung up.

Stepping over or out of a function or subroutine will actually execute all or some of the code in that function/subroutine. Therefore, there is the potential to be caught in an endless loop. Select Halt to regain control in MPLAB IDE.

6.3 ERROR MESSAGES

Cannot Find *filename*

The file *filename* was not created due to an error in the build. This means you will need to debug your code before a build can complete successfully.

filename is missing

Check your installation for missing files.

Hex file not found

The project did not generate a hex file. Errors in your code will prevent hex file generation and cause this error.

Language Tool Not Properly Installed

After installing a new version of MPLAB IDE, make sure you provide MPLAB IDE with the correct location of any language tools.

No Debugging Information Available

Make sure you have selected the option to create a `.cod` or `.cof` file with your build.

Unresolved Breakpoints

This message displays when breakpoints aren't set on a valid line, i.e., when breakpoints are set on optimized code, or code that does not have a valid assembly instruction associated with the breakpoint line. For information on how to fix an unresolved breakpoints, see **Section 9.4 "Breakpoints Dialog"**.

6.4 LIMITATIONS

MPASM Assembler Limitations

Some assembler limitations can appear to be MPLAB IDE limitations. These include:

- A **62** character length restriction for file and path names in the debug (COD) file.
- Include file path information entered in MPLAB IDE will NOT be used by the assembler.

See MPASM assembler documentation, Limitations section, for more information.

COD files with EEData

COD files with EEData may not view properly in the Disassembly window.

NOTES:

Part 2 – MPLAB IDE Reference

Chapter 7. MPLAB IDE Desktop 89
Chapter 8. MPLAB IDE Windows 101
Chapter 9. MPLAB IDE Dialogs 129
Chapter 10. MPLAB IDE Operational Reference 147

NOTES:

Chapter 7. MPLAB IDE Desktop

7.1 INTRODUCTION

The MPLAB IDE desktop is a resizable window that operates independent of the rest of the menu items. The desktop consists of a menu bar, tool bars, a status bar and any open windows and/or dialogs. The bars are covered here. Windows and dialogs are discussed in Chapter 8 and Chapter 9 respectively.

Topics covered in this chapter:

- MPLAB IDE Menu Bar
- MPLAB IDE Toolbars
- MPLAB IDE Status Bar

7.2 MPLAB IDE MENU BAR

All 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.

Shortcut (hot) keys for menu items are listed next to the menu item. Example: The shortcut keys for Print are Control-P (CTRL+P). You may set up hot keys under *Configure>Settings*, **Hot Keys** tab.

Available menus are:

- File
- Edit
- View
- Project
- Debugger
- Programmer
- Tools
- Configure
- Window
- Help

7.2.1 File

Below are the menu items in the File menu. For more on the editor and the items shown here, see **Chapter 11. "Using the Editor"**.

- New

  Displays an empty editor window named "Untitled" for the file. (See **Section 8.16 "File (Editor) Window"**.) When you close the window, you will be prompted to name the file.
- Open

  Opens an existing source file. You may select multiple files in the Open dialog (see **Section 9.8 "File Management Dialog"**) by clicking the filenames while holding down the CTRL or Shift key.

- Close

  Closes the active editor window. If the file has changed since it was saved last, you will be prompted to save changes.
- Save

  Saves the active editor window to disk under its original file name.
- Save As

  Opens the Save As dialog (see **Section 9.8 "File Management Dialog"**), with the addition of an "Encoding" list box. Select the type of file encoding here. Allows you to save the active editor window to disk under a new file name.
- Save All

  Saves all open editor windows to disk.
- Open Workspace

  Opening a workspace closes the previous workspace before opening the new one. For more on workspaces, see **Chapter 4. "Projects and Workspaces"**.
- Save Workspace

  Saving a workspace saves the current workspace. Workspaces may be automatically saved on a close by setting this under *Configure>Settings*, **Workspace** tab.
- Save Workspace As

  Opens the Save As dialog (see **Section 9.8 "File Management Dialog"**), allowing you to rename/relocate the current workspace before saving.
- Close Workspace

  Closing a workspace returns you to the default startup workspace configuration.
- Import

  Import a debug or hex file into your MPLAB IDE project. (See **Section 9.13 "Import Dialog"**.)
- Export

  Export a Hex file from your MPLAB IDE project. (See **Section 9.6 "Export Hex File Dialog"**.)
- Print

  Prints the active edit window. The Print dialog will open to allow you to set the printer and print options.
- Recent Files

  Displays a list of files opened during the current MPLAB IDE session. Set the amount of displayed files under *Configure>Settings*, **Workspace** tab.
- Recent Workspaces

  Displays a list of workspace opened during the current MPLAB IDE session. Set the amount of displayed workspaces under *Configure>Settings*, **Workspace** tab.
- Exit

  Closes the MPLAB IDE application.

7.2.2 Edit

Below are the menu items in the Edit menu. For more on the editor and the items shown here, see **Chapter 11. "Using the Editor"**.

- Undo

  Reverses the last change made to the active window. When there is no edit action to undo, the menu command is grayed out and you cannot select the command.
- Redo

  Reverses the effect of the most recent Undo operation in the active window. When there is no edit action to redo, the menu command is grayed out and you cannot select the command.
- Cut

  Deletes the selected text in the current window and places it on the clipboard. After this operation you can paste the deleted text into another MPLAB Editor window, into a different location in the same MPLAB Editor window, or into another Windows application.
- Copy

  Copies the selected text in the current window onto the clipboard. After this operation, you can paste the copied text into another MPLAB Editor window, into another location in the same MPLAB Editor window, or into another Windows application.
- Paste

  Pastes the contents of the clipboard into the current window at the insertion point. You can only perform this operation if the clipboard contains data in text format. **MPLAB Editor does not support pasting of bitmaps or other clipboard formats**.
- Delete

  Deletes the selected text.
- Select All

  Selects all text and graphics in the Edit window.
- Find

  Opens the Find dialog. (See **Section 9.11 "Find and Replace Dialogs"**.)
- Find Next

  Find the next instance of Find text.

  F3 repeats the last Find.

  Shift+F3 reverses the direction of the last Find.
- Replace

  Opens the Replace dialog. (See **Section 9.11 "Find and Replace Dialogs"**.)
- Go to

  Go to the specified line of text in the editor window.
- Advanced

  Advanced editing features. Includes making selected text all uppercase or lowercase, commented text or regular code or indented or unindented text.

  In addition there is a Match function. Go to the brace that is the matching brace for the brace at the cursor. This function works for curly braces, parentheses, angle brackets and square brackets.
- Bookmarks

  Work with bookmarks. Toggle a bookmark (alternately enable/disable a bookmark), go to the next or previous bookmark or disable all bookmarks.
- Properties

  Opens the Editor Options dialog (see **Section 11.2.1 "Editor Options Dialog"**.)

7.2.3 View

Below are the menu items in the View menu. Any item not applicable to the selected device will be disabled. Selecting an item in this menu will make that item visible on the desktop.

- **Section 8.4 "Project Window"**
- **Section 8.5 "Output Window"**
- **Section 7.3 "MPLAB IDE Toolbars"**
- **Section 8.6 "Disassembly Listing Window"**
- **Section 8.7 "Hardware Stack Window"**
- **Section 8.8 "Program Memory Window"**
- **Section 8.9 "File Registers Window"**
- **Section 8.10 "EEPROM Window"**
- **Section 8.11 "LCD Pixel Window"**
- **Section 8.12 "Watch Window"**
- **Section 8.13 "Special Function Registers Window"**
- *Tool-Specific Windows* – Depending on what debug tool you have selected (*Debugger>Select Tool*), tool-specific items, (e.g., **Section 8.14 "Trace Memory Window"**), may appear on this menu.

7.2.4 Project

Below are the menu items in the Project menu. For more on projects, see **Chapter 4. "Projects and Workspaces"**.

- Project Wizard

  Use the Project Wizard to help you set up a new project. For more on the wizard, see **Section 4.2 "Using the Project Wizard"**.
- New

  Create a new project in the workspace.

  Opens the New Project dialog. You will be asked to enter a name and path for the new project. (See **Section 9.14 "New Project Dialog"**.)
- Open

  Add an existing project to the workspace and set as active. Opens the Open Project dialog (see **Section 9.8 "File Management Dialog"**.)
- Close

  Close the current project in the workspace. You will be prompted to save the current project before closing.

  Closing a project does not close the workspace. Use *Edit>Workspace>Close* to close the workspace.
- Set Active Project

  Select a project as the active project in the workspace. For more on active projects, see **Section 4.9 "Using Multiple Projects in a Single Workspace"**.

  To enable Quickbuild, select None.

- Quickbuild (*filename*)

  Build a single assembly file using MPASM assembler without having to create a project (no linker). None/Enable Quickbuild must be selected under Set Active Project. The assembly file must be open in a file window and the window must be active.

  There may be assembler limitations with this procedure. See **Section 6.4 "Limitations"**.
- 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.
- Build All

  Build the project by compiling/assembling all files. A project must be open before this item will appear.
- Make

  Build the project by compiling/assembling only files that have changed since the last build. A project must be open before this item will appear.
- Build Options

  Set and view options for the active project and individual files using the Build Options dialog (see **Section 9.5 "Build Options Dialog"**.)
- Find in Project Files

  Find text in multiple files of the active project. (See **Section 9.10 "Find In Project Files Dialog"**.) Results are displayed in the Output window.
- Save Project

  Save the active project.
- Save Project As

  Opens the Save Project As dialog. (See **Section 9.18 "Save Project As Dialog"**.)
- Add Files to Project

  Insert files into the active project. (See **Section 9.8 "File Management Dialog"**.) Depending on the type of file, MPLAB IDE will sort the files into the correct type within the project window tree.
- Remove Files from Project

  Remove (delete) files from active project. Files are not deleted from the directory.
- Select Language Toolsuite

  Select the toolsuite you will use for your project, e.g., Microchip Toolsuite. (See **Section 9.20 "Select Language Toolsuite Dialog"**.)
- Set Language Tool Locations

  Set the paths/directories to the language tools you will be using in your project, i.e., match a language tool name in MPLAB IDE (ex: MPASM assembler) with an executable (ex: `C:\Program Files\MPLAB IDE\MCHIP_Tools\mpasmwin.exe`). (See **Section 9.21 "Set Language Tool Location Dialog"**.)
- Select Version-Control System

  Set up your project to use files from a version-control system (Visual Source Safe.) For more information, see **Section 4.6 "Using A Version-Control System (VCS)"**.

7.2.5 Debugger

Below are the menu items in the Debugger menu.

Note: Single stepping may be very slow when using a debugger if your selected device has EEPROM data and (1) you have a programmer enabled or (2) you have the EEPROM window open, either of which will attempt to access the data on each step. To improve speed, disable the programmer or close/minimize the EEPROM window.

- Select Tool

  Select a debug tool. The default is None. The list of available debuggers will depend on which ones you have installed. The order of items on the list is subject to installation order.
- Clear Memory

  Clear all or only certain types of MPLAB IDE memory used in the project, e.g., program, data, EEPROM, configuration.
- *Basic Debug Options*
- *Tool-Specific Options* – Depending on what debug tool you have selected (*Debugger>Select Tool*), additional tool-specific items, such as "Stopwatch", may appear on this menu.

Basic Debug Options

Once you have selected any debug tool, the Debugger menu will add the following options:

- Run

  Execute program code until a breakpoint is encountered or until Halt is selected.

  Execution starts at the current program counter (as displayed in the status bar). The current program counter location is also represented as a pointer in the Program Memory window. While the program is running, several other functions are disabled.
- Animate

  Animate causes the debugger to actually execute single steps while running, updating the values of the registers as it runs.

  Animate runs slower than the Run function, but allows you to view changing register values in the Special Function Register window or in the Watch window.

  To Halt Animate, use the menu option *Debugger>Halt* instead of the toolbar Halt or **F5**.
- Halt

  Halt (stop) the execution of program code. When you click Halt, status information is updated.
- Step Into

  Single step through program code.

  For assembly code, this command executes one instruction (single or multiple cycle instructions) and then halts. After execution of one instruction, all the windows are updated.

  For C code, this command executes one line of C code, which may mean the execution of one or more assembly instruction, and then halts. After execution, all the windows are updated.

  Note: Do not step into a Sleep instruction.

- Step Over

  Execute the instruction at the current program counter location. At a CALL instruction, Step Over executes the called subroutine and halts at the address following the CALL. If the Step Over is too long or appears to have “hung”, click Halt.
- Step Out

  Step out of a subroutine. If you are single stepping through subroutine code, you may finish executing the rest of the subroutine code and halt at the address following the subroutine CALL by using Step Out.
- Reset

  Issue the specified reset, either MCLR, Watchdog Timer, Brown Out or Processor reset. Reset options and actions depend on the device selected.
- Breakpoints

  Open the Breakpoint dialog. Set multiple breakpoints in this dialog. For other ways to set a breakpoint, see **Section 3.18 “Using Breakpoints”**.
- Settings

  Open a tool-specific settings dialog. Set up tool functions here. Also, find tool limitations.

7.2.6 Programmer

Below are the menu items in the Programmer menu.

- Select Programmer

  Select a programmer. The default is None. The list of available programmers will depend on which ones you have installed. The order of items on the list is subject to installation order.
- *Basic Programmer Options*
- *Programmer-Specific Options* – Depending on what programmer you have selected (*Programmer>Select Programmer*), additional programmer-specific items, such as “Load SQTP File”, may appear on this menu.

Basic Programmer Options

Depending on the programmer chosen, different options may appear on the Programmer menu. Basic items that will generally be available are:

- Enable Programmer

  Establish communications between MPLAB IDE and the programmer. This is grayed out if the programmer is already enabled.
- Disable Programmer

  End communications between MPLAB IDE and the programmer. This is grayed out if the programmer is already disabled.
- Program

  Program specified memory areas: program memory, configuration bits, ID locations and/or EEPROM data.
- Verify

  Verify programming of specified memory areas: program memory, configuration bits, ID locations and/or EEPROM data.
- Read

  Read specified memory areas: program memory, configuration bits, ID locations and/or EEPROM data.
- Blank Check All

  Check to see that all device memory is erased/blank.

- Blank Check OTP

  For OTP devices, check to see that program, data and EEPROM memory is erased/blank.
- Erase Flash Device

  Erase all data on the PICmicro Flash MCU device including memory, ID and configuration bits.
- Reset Program Statistics

  Set programming statistics (e.g., errors) to default values.
- Download OS

  Download the latest operating system for your programmer.

  Note: PICSTART Plus must be upgraded before this feature is usable. Follow the instructions in the help for this tool.
- About

  View information about your tool in this dialog.
- Settings

  Opens a tool-specific settings dialog. Set up information about your tool in this dialog, i.e., Memory Ranges and Communications Port Setup, as well as Voltages and SQTP, if applicable.

7.2.7 Tools

Below are the menu items in the Tool menu.

- MPLAB Macros

  Enable Microsoft macro capability for use with MPLAB IDE.
- Visual Initializer

  Set up your development code visually. See on-line help for MPLAB VDI for more on how this tool operates.

7.2.8 Configure

Below are the menu items in the Configure menu.

Note: Not all items may be available depending on device and debug tool selected.

- Select Device

  Select the device for your development mode. (See **Section 9.19 "Select Device Dialog"**.) Select the development tool under the Debugger or Programmer menu.
- Configuration Bits

  Select values for the device configuration bits. (See **Section 8.15 "Configuration Bits Window"**.) Setting these values will affect both debugger and programmer operation.
- External Memory

  Select whether to use external memory or not. Also specify external memory range. (See **Section 9.7 "External Memory Setting Dialog"**.)
- ID Memory

  Enter value into ID memory. (See **Section 9.24 "User ID Memory Dialog"**.)
- Settings

  Enter default setting for the workspace, debugger, program loading, hot keys and projects. (See **Section 9.22 "Settings Dialog"**.)

7.2.9 Window

Below are the menu items in the Window menu.

- Close All
  Close all open windows.
- Cascade
  Arrange open windows to overlap so that each title bar is visible.
- Tile Horizontally
  Arrange open windows in smaller sizes to fit next to each other horizontally.
- Tile Vertically
  Arrange open windows in smaller sizes to fit next to each other vertically.
- Arrange Icons
  Arrange all iconized windows on the bottom of the IDE.
- *Open windows*
  A list of all open windows is displayed. Click on a window name to make that window active.

7.2.10 Help

Below are the menu items in the Help menu.

- Topics
  Select a help file from the list on the dialog. (See **Section 9.12 "Help Topics Dialog"**.)
- Readme Files
  View an HTML list of all Readme files available for Microchip tools. Click on a link to view the actual file.
- About MPLAB IDE
  Review MPLAB IDE trademarking and component version information. (See **Section 9.2 "About MPLAB IDE Dialog"**.)

7.3 MPLAB IDE 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.

Toolbars Features

- Click and drag the toolbar to make it a floating toolbar.
- Click and drag the toolbar to the top or sides of MPLAB IDE desktop to dock it.
- Click and drag the toolbar off the MPLAB IDE desktop.
- Hover the mouse pointer over an icon to pop up the icon name.
- Right-click on a toolbar to change the contents or show/hide toolbar.

Toolbars Available

- Standard Toolbar
- Project Manager Toolbar
- Debug Toolbar
- Programmer Toolbar
- Checksum Toolbar

7.3.1 Standard Toolbar

The Standard (Edit) Toolbar currently contains button icons for the following functions:

- New File – Open a new file window
- Open File – Open an existing file in a window
- Save File – Save the current file window contents to a file
- Cut – Cut selected text to clipboard
- Copy – Copy selected text to clipboard
- Paste – Paste text from clipboard
- Print File – Print contents of active file window
- Find – Open the Find dialog for finding text in active file window
- Help – Displays MPLAB IDE Help selection dialog

7.3.2 Project Manager Toolbar

The Project Manager Toolbar currently contains button icons for the following functions:

- New Project – Set the name and location of a new project
- Open Project – Open an existing project
- Save Workspace – Save the current project and workspace to files
- Build Options – View or change project settings
- Find in Project Files – Opens a Find dialog to search for text in all project files

If a project is loaded, additional items may be found:

- Make – Build only the files in the active project that have changed
- Build All – Build all files in the active project

7.3.3 Debug Toolbar

The Debug Toolbar currently contains button icons for the following functions:

- Run – Run program
- Halt – Halt program execution
- Animate – Continually step into instructions – Halt using *Debugger>Halt*
- Step Into – Step into next instruction
- Step Over – Step over next instruction
- Step Out – Step out of subroutine
- Reset – Perform MCLR reset

Depending on the debug tool chosen, other icons may appear.

7.3.4 Programmer Toolbar

Depending on the programmer chosen, different button icons may appear on the Program Toolbar. Basic icons that will generally be available are:

- Blank Check All/Blank Check – Check that the device memory is blank.
- Read – Read device memory as specified in *Programmer>Settings*, **Program** tab.
- Program – Program device memory as specified in *Programmer>Settings*, **Program** tab.
- Verify – Verify that target memory has been correctly programmed.
- Erase Flash Device – If the device is Flash, erase the device.
- Program Statistics – Display programing statistics, such as how many times programming passed, failed and the total number of attempts to program.

7.3.5 Checksum Toolbar

This toolbar only displays the checksum value. Checksum algorithms can be found in the device's programming specification. Programming spec's may be found on our web site.

When a device is code protected, you may use the unprotected checksum to determine the device checksum.

Previously, this item had been on the status bar. Its place has been taken by banking information.

7.4 MPLAB IDE STATUS BAR

The status bar provides up-to-date information on the status of your MPLAB IDE session.

When an application is running, it displays “Running” and a progress bar.

When an application is not running, the information provided includes:

Item	Title	Typical Entry	Description
1	Current debug tool	MPLAB SIM	Displays currently selected debug tool.
2	Current programmer	PICSTART Plus	Displays currently selected programmer
3	Current processor	PIC18F452	Displays the currently selected processor
4	Current program counter	pc:0x5f	Displays the current program counter
5	Current w register value	W:0x00	Displays current w register value.
6	Status bits	ov Z dc c	Upper Case = Set (1) Lower Case = Reset (0)
7	Global break enable	Bk On	Displays current status of Global Break Enable.
8	Processor frequency	4MHz	Displays current processor frequency
9	Banking information	bank 0	Displays current bank in data memory
10*	Line No., Column-Windows Open	Ln 1 Col 1	Displays current line number and column in file
11*	Insert/strikeover	INS	Toggles typing mode between insert and strikeover INS = Insert Characters OVR = Type over characters
12*	Write/read only	WR	Displays Write/Read Only Status WR = Editable File RO = Read Only File

* Only available when a file (editor) window has focus.

NOTES:

Chapter 8. MPLAB IDE Windows

8.1 INTRODUCTION

MPLAB IDE windows behave as normal Windows applications. Other standard Windows features include window sizing buttons and vertical and horizontal scroll bars. Additional MPLAB IDE window features are:

- In-place editing of data
- Dockable windows
- Click-and-drag to change window column width and order
- Right-click (context) menu to change window font and color

Most windows available in MPLAB IDE are listed under the View menu. Depending on the tools you have installed, tool-specific windows may be available for viewing.

General

- Changing Window Data and Properties
- Code Display Window Symbols

View Menu

- Project Window
- Output Window
- Disassembly Listing Window
- Hardware Stack Window
- Program Memory Window
- File Registers Window
- EEPROM Window
- LCD Pixel Window
- Watch Window
- Special Function Registers Window
- Trace Memory Window

Configure Menu

- Configuration Bits Window

File Menu

- File (Editor) Window

8.2 CHANGING WINDOW DATA AND PROPERTIES

MPLAB IDE windows have some or all of the listed properties, depending on the type and use of the window.

8.2.1 Changing Window Data – Edit-in-Place or Choose from List

MPLAB IDE window data may be edited as described below. If you cannot edit the data, then this information is not available for you to change.

- Data may be edited "in place", i.e., double-click to select an item and then type in a new value.
- Date may be chosen from a drop-down list when only certain choices are possible.

8.2.2 Changing Window Location – Dock/Undock

Docking

- Windows may be made dockable (able to be attached to the top or sides of the MPLAB IDE desktop) by selecting "Dockable" from the window system menu (click in the upper left-hand corner).

> **Note:** File (editor) windows may not be made dockable.

- Dockable windows may be docked by dragging them to the desired docking location.
- Dockable windows may be floated, i.e., dragged off of the MPLAB IDE desktop.

Undocking

- Dockable windows may be undocked by double-clicking on the gripper bar. 
- Docked windows may be made undockable by right-clicking on the gripper bar and selecting "Dockable" from the menu.
- Undocked windows may be made undockable by right-clicking on the title bar and selecting "Dockable" from the menu.

8.2.3 Changing Window Columns – Size, Visibility, Order

- Columns can be resized as follows: move the cursor over the line between columns till it changes to the resize cursor and then click and drag to widen or narrow the column.
- Columns can be made visible/invisible as follows:
  - Right-click on any column head and check an item from the list to make it visible.
  - Right-click on any column head and select More to open the window Properties dialog, **Column Settings** tab, (see **Section 9.17 "Properties Dialog"**) where you may set column visibility.
  - Right-click in the window and select Properties. On the **Column Settings** tab, you may set column visibility.
- Columns can be ordered as follows:
  - Click on any column head and drag it to the desired location.
  - Right-click on any column head and select More to open the window Properties dialog, **Column Settings** tab, (see **Section 9.17 "Properties Dialog"**) where you may set column location.
  - Right-click in the window and select Properties. On the **Column Settings** tab, you may set column location.

8.2.4 Changing Window Fonts and Colors

- Window fonts and change colors may be specified by right-clicking in the window and selecting Properties, **General** tab (see **Section 9.17 "Properties Dialog"**).
- Select the background colors for the SFR window by right-clicking in the window and selecting Properties, **Background Colors** tab (see **Section 8.13 "Special Function Registers Window"**).

8.3 CODE DISPLAY WINDOW SYMBOLS

In the gutter of some windows, the following symbols may appear:

	Current location (line of code) of the program counter. Program execution is halted (stopped.)
	Location of the program counter before the program was run. Program is currently executing (running.)
	Breakpoint set.
	Breakpoint disabled.
	Breakpoint set on an address. This symbol is displayed only in the file (editor) window, since one line of code could correspond to several addresses. See **Section 3.18 "Using Breakpoints"** for more information.
	Code coverage enabled for this location (Program Memory window, MPLAB ICE 2000/4000 only.)
	Complex trigger point set at this location (Program Memory window, MPLAB ICE 2000/4000 only.)
	Filter in trace (File window.) See **Section 11.5.1 "Filter Trace"** for more information.
	Filter out trace (File window.) See **Section 11.5.1 "Filter Trace"** for more information.

8.4 PROJECT WINDOW

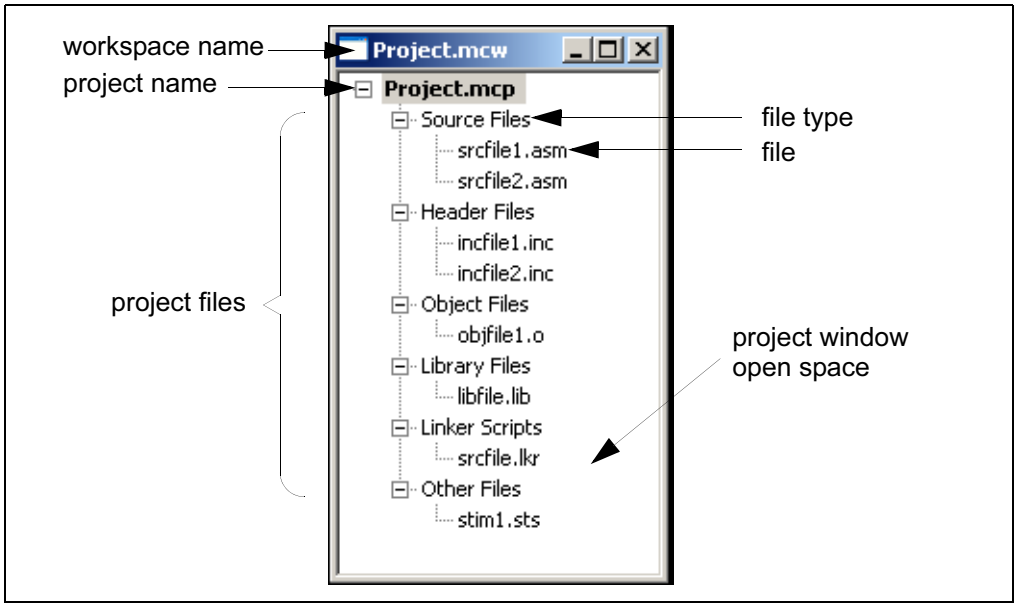
The Project window contains a summary of information about the project. Select *View>Project* to alternately view or close the Project window.

- Project Window Display
- Project Window Menus
- Project Window FAQ

8.4.1 Project Window Display

The title bar of the Project window contains the name of the project workspace. The window itself contains a list of projects and project files by file type in tree format. The general structure of the project tree for a single project is shown in Figure 8-1.

FIGURE 8-1: PROJECT TREE GENERAL STRUCTURE



If the project name is bolded, it is the active project. If the project name is followed by an asterisk (e.g., `c:\project1\project.mcp*`), either the project settings have changed or the project is new, and it needs to be saved.

For information on the types of files listed in this window, see **Section 4.5 "Project Folders and Files"**.

8.4.2 Project Window Menus

Depending on where you click in the project window, different menus are available.

- Display Menu
- Project Menu
- File Type Menu
- File Menu

8.4.2.1 DISPLAY MENU

If you right-click on any open space in the project window, a menu with project display commands will be displayed.

- Refresh

  This command causes the IDE to go out and check the status of all the files in all projects.

  For version-control system files, the files' status is updated in the project window.

  For missing files that you have relocated, “file not found” text will be removed.
- Preferences

  Opens a dialog to set project preferences. See **Section 9.15 “Project-Display Preferences Dialog”**.

8.4.2.2 PROJECT MENU

If you right-click on a project file, a menu with project level commands will be displayed.

- Set Active

  Select a project as the active project in the workspace. For more on active projects, see **Section 4.9 “Using Multiple Projects in a Single Workspace”**. To enable Quickbuild, select None.
- Clean

  Removes all intermediary project files, such as object, hex and debug files. These files are recreated from other files when a project is built.
- Build All

  You must have a project open before this option is visible.

  Build all files in the project, i.e., compile/assemble all files as specified in the project. Build All is available in the right mouse button menu of the project window as well.
- Make

  You must have a project open before this option is visible.

  Build only the files in the project that have changed, i.e., compile/assemble these files as specified in the project. Make is available in the right mouse button menu of the project window as well.
- Build Options

  Set and view options for the project and individual files. See **Section 9.5 “Build Options Dialog”**.
- Save

  Save the active project.
- Save As

  Save the active project to a new location/name. See **Section 9.18 “Save Project As Dialog”**.
- Close

  Close/remove the selected project in the workspace.
- Add Files

  Insert files into the project. Depending on the type of file, MPLAB IDE will sort the files into the correct type within the project window tree.

Note:	Adding Header Files to the project window will not cause these files to be added to the project build. You must use a `#include` statement in your code.

- Reload

  This command causes the IDE to go out to disk and read the project file in again. It also does a Refresh immediately after. This functionality is useful if you modify your project and then decide you don't want the modifications. As long as you haven't saved and overwritten the old project, you can "Reload" and get back to your original project.
- Refresh

  This command causes the IDE to go out and check the status of all the files in the project.

  For version-control system files, the files' status is updated in the project window.

  For missing files that you have relocated, "file not found" text will be removed.
- Select Language Toolsuite

  Select the toolsuite you will use for your project, e.g., Microchip MPASM Toolsuite. See **Section 9.20 "Select Language Toolsuite Dialog"**.
- Select Version-Control System

  Set up your project to use files from a version-control system. See **Section 9.25 "Version-Control Dialog"**.

8.4.2.3 FILE TYPE MENU

If you right-click on a file type in the project tree, a menu with file type commands will be displayed.

- Library Link Order (Library Files Only)

  Opens the Library Link Order dialog, which shows you the true order of the library files and allows you to resequence them. This is important for MPLAB C30 users.
- Add Files

  Insert files into the project. Depending on the type of file, MPLAB IDE will sort the files into the correct type within the project window tree.

  Note: Adding Header Files to the project window will not cause these files to be added to the project build. You must use a `#include` statement in your code.
- Filter

  Change the way files are filtered to determine file type. For example, to be able to add an assembly file with extension .a under Source Files, add .a to the list of file extensions in the Filter dialog.

8.4.2.4 FILE MENU

If you right-click on a file in the project tree, a menu with file commands will be displayed.

Note: If you are using a version-control system, additional commands that can be used on the selected file.

- Assemble/Compile

  Assemble/compile the selected file as appropriate.
- Build Options

  Set and view options for the project and individual files. See **Section 9.5 "Build Options Dialog"**.
- Edit

  Open the selected file in a window for editing, if appropriate.
- Remove

  Remove the selected file from the project. The file is not deleted from the directory.

8.4.3 Project Window FAQ

How do I:

- Create/Update a project?

  See **Section 4.3 "Creating/Updating any Project"**.
- Build a project?

  Right-click on the project in the workspace that you wish to build. In the menu, select Build All to build the entire project or Make to recompile/reassemble any changed files and then build.

  The results of this build will be loaded into program memory, even if the selected project is not the active project.
- Use projects with workspaces?

  See **Chapter 4. "Projects and Workspaces"**.
- Set a project as active?

  To set a project as the active project, select *Project>Set Active* *Project>project-name.mcp*, where `projectname.mcp` is the project name, or right-click on the project in the Project window and select Set As Active Project.

8.5 OUTPUT WINDOW

Selecting *View>Output* opens the Output window. This window contains tabbed information about program output.

- **Build** tab – Lists messages from a project build. Build messages are the result of language tools selected (*Project>Select Language Toolsuite*) and build options set (*Project>Build Options>Project*).
- **Version Control** tab – Displays version control information, if a version control system is used in the project (*Project>Version Control*).
- **Find in Files** tab – Lists the result of *Project>Find in Project Files*.
- Depending on the functionality selected, other output tabs may be available.

Below are the menu items in the Output window right mouse button menu.

- Select All

  Selects all text and graphics in the Edit window.
- Copy

  Copies the selected text in the current window onto the clipboard. After this operation, you can paste the copied text into another MPLAB Editor window, into another location in the same MPLAB Editor window, or into another Windows application.
- Clear Page

  Remove all text in the selected output tab.

8.6 DISASSEMBLY LISTING WINDOW

Select *View>Disassembly Listing* to view disassembled code in this window. Breakpoints may be set in code.

Code execution information may be displayed in the form of symbols in the gutter of the window.

Note: COD files containing EEData may not work properly. Both the start of the program and EEData appear as Line 0 in the line number table.

Below are the menu items in the Disassembly window right mouse button menu.

- Set/Remove Breakpoint
  Set or remove a breakpoint at the currently-selected line.
- Enable/Disable Break
  Enable or disable a breakpoint at the currently selected line.
- Breakpoints
  Disable, enable or remove all breakpoints.
- Run To Cursor
  Run the program to the current cursor location. Formerly Run to Here.
- Set PC at Cursor
  Set the program counter (PC) to the cursor location.
- Copy
  Copy selected text to clipboard. Select text by (1) clicking and dragging mouse over text or (2) clicking at the beginning of text and shift-clicking at the end of text.
- Select All
  Select all text in the window.
- Output to File
  Write the displayed window contents to a text file using a Save As dialog (see **Section 9.8 "File Management Dialog"**.)
- Print
  Print the contents of the window.
- Properties
  Open the Disassembly Options dialog. Set display and functional options. This dialog is similar to the Editor Options dialog (see **Section 11.2.1 "Editor Options Dialog"**.)

8.7 HARDWARE STACK WINDOW

The Hardware Stack window displays the contents of the hardware stack. The number of available levels depends on the selected device.

Note: If Disable Stack Overflow Warning is cleared, MPLAB IDE will display stack overflow and underflow warnings when they occur. This is not supported on all processor modules.

- Hardware Stack Window Display
- Hardware Stack Window Menu
- Hardware Stack Window FAQ

8.7.1 Hardware Stack Window Display

Data is displayed in the following columns:

- TOS (if available) – A pointer showing the top of stack (TOS).
- Stack Level – Stack level number. The total number of levels is dependent on the device selected.
- Stack Return Address – Return addresses on the stack.
- Location – Function name + offset. Information on location in a function.

Stack Return Address

Displays the current contents of the hardware stack. Previously used values are still displayed, since those values are still in the device. The Top of Stack (TOS) indicator shows the current stack pointer location.

8.7.2 Hardware Stack Window Menu

Below are the menu items in the Hardware Stack right mouse button menu.

- Close
  Close this window.
- Pop Stack
  Move address at Top of Stack onto program counter.
- Set Top of Stack
  Set the Top of Stack to the current cursor location.
- Center Stack Location
  Center the currently selected line in the window.
- Output to File
  Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 9.8 “File Management Dialog”**), with the addition of an “Output Range”. Enter the “Start” and “End” lines to output.
- Print
  Print the contents of the window.
- Refresh
  Refresh the data in this window.
- Properties
  Set up fonts and colors. See **Section 9.17 “Properties Dialog”**.

8.7.3 Hardware Stack Window FAQ

How do I:

- Understand the Return Address information?
  See **Section 8.7.1 “Hardware Stack Window Display”**
- Pop the stack?
  Click on the right mouse button in the window to open a menu. Select Pop Stack.
- Set Top of Stack (TOS)?
  Click on the right mouse button in the window to open a menu. Select Set Top of Stack.

8.8 PROGRAM MEMORY WINDOW

The Program Memory window displays locations in the range of program memory for the currently selected processor. If external program memory is supported by the selected device and enabled, it will also appear in the Program Memory window.

- Program Memory Window Display
- Program Memory Window Menu
- Program Memory Window FAQ

8.8.1 Program Memory Window Display

You may change the way opcodes are displayed in the program memory window by clicking on one of the buttons on the bottom of the window:

- Opcode Hex
- Machine or Symbolic
- PSV Mixed (dsPIC devices only)
- PSV Data (dsPIC devices only)

8.8.1.1 OPCODE HEX

This format displays program memory information as hex code. The window will have the following columns:

- Address – Hexadecimal address of the opcode in the next column.
- *Opcode Blocks* – Hexadecimal opcode, shown in 2- or 3-byte blocks. For most PICmicro MCU's these blocks represent words. For PIC18CXXX devices, the blocks represent 2 bytes. For dsPIC devices, the blocks represent 3 bytes.

  The opcode block that is highlighted represents the current location of the program counter.
- ASCII – ASCII representation of the corresponding line of opcode.

8.8.1.2 MACHINE OR SYMBOLIC

Machine format displays disassembled hex code with no symbolic information. Symbolic format displays disassembled hex code with symbols. The window will have the following columns:

- *Debug Info* – Information useful for debugging. A pointer shows the current location of the program counter.
- Line – Reference line number
- Address – Opcode hexadecimal address.
- Opcode – Hexadecimal opcode, shown in 2- or 3-byte blocks. For most PICmicro MCU's these blocks represent words. For PIC18CXXX devices, the blocks represent 2 bytes. For dsPIC devices, the blocks represent 3 bytes.
- Label (Symbolic Only) – Opcode label in symbolic format.
- Disassembly – A disassembled version of the opcode mnemonic.

8.8.1.3 PSV MIXED (dsPIC DEVICES ONLY)

This format displays program memory as opcode and the PSV area (CORCON register, PSV bit set). The window will have the following columns:

- *Debug Info* – Information useful for debugging. A pointer shows the current location of the program counter.
- Line – Reference line number.
- Address – Opcode hexadecimal address.
- Opcode – Hexadecimal opcode, shown in 3-byte blocks.
- PSV Address – Data space hexadecimal address of the opcode.
- Data – Opcode formatted as data.
- Label – Opcode label in symbolic format.
- Disassembly – A disassembled version of the opcode mnemonic.

For more information on dsPIC devices, see *dsPIC30F Family Reference Manual* (DS70046).

8.8.1.4 PSV DATA (dsPIC DEVICES ONLY)

This format displays program memory as file registers, for when program space is visible in data space (CORCON register, PSV bit set). The window will have the following columns:

- Address – Program space hexadecimal address of the data.
- PSV Address – Data space hexadecimal address of the data.
- Data Blocks – Hexadecimal data, shown in 3-byte blocks.

The data block that is highlighted represents the current location of the program counter.

- ASCII – ASCII representation of the corresponding line of data.

For more information on dsPIC devices, see *dsPIC30F Family Reference Manual* (DS70046).

8.8.2 Program Memory Window Menu

Below are the menu items in the Program Memory right mouse button menu.

- Close
  Close this window.
- Set/Remove Breakpoint (Machine/Symbolic Only)
  Set or remove a breakpoint at the currently-selected line.
- Enable/Disable Break (Machine/Symbolic Only)
  Enable or disable a breakpoint at the currently selected line.
- Breakpoints
  Disable, enable or remove all breakpoints.
- Run To Cursor
  Run the program to the current cursor location. Formerly Run to Here.
- Set PC at Cursor
  Set the program counter (PC) to the cursor location.
- Center Debug Location
  Center the current PC line in the window.
- Cursor Tracks Debug Location
  Cursor (arrow) will track the current debug location.

- Find

  Find text specified in the Find dialog in this window.
- Find Next

  Find the next instance of Find text.

  F3 repeats the last Find.

  Shift+F3 reverses the direction of the last Find.
- Go To

  Go to the address specified in the Go To dialog.
- Import Table

  Open the Import dialog (see **Section 9.8 "File Management Dialog"**.) "Opcode Hex" must be selected for this item to be available.
- Export Table

  Open the Export As dialog (see **Section 9.8 "File Management Dialog"**.) "Opcode Hex" must be selected for this item to be available.
- Fill Memory

  Fill memory from Start Address to End Address with the value in Data. See **Section 9.9 "Fill Memory/Registers Dialog"**.
- Output to File

  Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 9.8 "File Management Dialog"**), with the addition of an "Output Range". Select the type of range, either "Lines" or "Address", and then enter the "Start" and "End" values to output.
- Print

  Print the contents of the window.
- Refresh

  Refresh the data in this window.
- Properties

  Set up window properties. See **Section 9.17 "Properties Dialog"**.

8.8.3 Program Memory Window FAQ

How do I:

- Enable external (off-chip) memory, for parts that support this?

  Select *Configure>External Memory*. In the External Memory dialog, check "Use External Memory", enter a range and click **OK**. External Memory will now appear in the Program Memory window.

  > **Note:** For some tools, you may actively have to upload external memory to MPLAB IDE before values will appear in the window.
- Fill program memory with a value?

  Right-click in the window and select "Fill Memory" to open a dialog where you may enter fill data.
- Set a breakpoint?

  See **Section 3.18 "Using Breakpoints"**.

8.9 FILE REGISTERS WINDOW

The File Register window displays all the file registers of the selected device. When a file register value changes, or the processor is interrogated, the data in the File Register window is updated.

Note: To speed up debugging with certain hardware tools, close this window. Use the SFR or Watch window instead.

- File Registers Window Display
- File Registers Window Menu
- File Registers Window FAQ

8.9.1 File Registers Window Display

You may change the way data is displayed in the file register window by clicking on one of the buttons on the bottom of the window.

- Hex
- Symbolic
- XY Data (dsPIC devices only)

8.9.1.1 HEX

This format displays file register information as hex data. The window will have the following columns:

- Address – Hexadecimal address of the data in the next column.
- *Data Blocks* – Hexadecimal data, shown in 1- or 2-byte blocks.
- ASCII – ASCII representation of the corresponding line of data.

8.9.1.2 SYMBOLIC

This format displays each file register symbolically with corresponding data in hex, decimal, binary and character formats. The window will have the following columns:

- Address – Data hexadecimal address.
- Hex – Hexadecimal data, shown in 1- or 2-byte blocks.
- Decimal – Data in decimal format.
- Binary – Data in binary format.
- Char – Data in character format.
- Symbol Name – Symbolic name for the data.

8.9.1.3 XY DATA (dsPIC DEVICES ONLY)

This format displays file register information as hex data. The window will have the following columns:

- Address – X hexadecimal address of the data.
- Y Bus – Y hexadecimal address of data, if supported.
- *Data Blocks* – Hexadecimal data, shown in 2-byte blocks.
- ASCII – ASCII representation of the corresponding line of data.

For more information on dsPIC devices, see *dsPIC30F Family Reference Manual* (DS70046).

8.9.2 File Registers Window Menu

Below are the menu items in the File Registers right mouse button menu.

- Close

  Close this window.
- Full Memory Update

  Updates the entire contents of this window after a halt or single step if enabled.

  By default, this option is enabled. If the window is open, only the data visible is updated. If the window is closed, no data is updated.
- Find

  Find text specified in the Find dialog in this window.
- Find Next

  Find the next instance of Find text.

  F3 repeats the last Find.

  Shift+F3 reverses the direction of the last Find.
- Go To

  Go to the address specified in the Go To dialog.
- Import Table

  Open the Import dialog (see **Section 9.8 "File Management Dialog"**.)
- Export Table

  Open the Export As dialog (see **Section 9.8 "File Management Dialog"**.)
- Fill Registers

  Fill registers from Start Address to End Address with the value in Data. See **Section 9.9 "Fill Memory/Registers Dialog"**.
- Output to File

  Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 9.8 "File Management Dialog"**), with the addition of an "Output Range". Select the type of range, either "Lines" or "Address", and then enter the "Start" and "End" values to output.
- Print

  Print the contents of the window.
- Refresh

  Refresh the data in this window.
- Properties

  Select background colors for SRF's and unallocated memory. Also, set up generic window properties (see **Section 9.17 "Properties Dialog"**.)

8.9.3 File Registers Window FAQ

How do I:

- Fill all registers with a value?

  Right-click in the window and select "Fill Registers" to open a dialog where you may enter fill data.

8.10 EEPROM WINDOW

The EEPROM window displays EEPROM data for any microcontroller device that has EEPROM data memory (e.g., PIC16F84A.) Data/opcode hex information of the selected device is shown. When an EEPROM register value changes or the processor is halted, the data in the EEPROM window is updated.

Note:	The start of EEPROM data memory needs to be specified for use with programmers. For most PICmicro MCU's, the start should be at 0x2100 (org H'2100'). For PIC18FXXXX devices, the start should be at 0xF00000 (org H'F00000'). Please check the programming specification for your selected device to determine the correct address.

- EEPROM Window Display
- EEPROM Window Menu
- EEPROM Window FAQ

8.10.1 EEPROM Window Display

This display format shows data in the following columns:

- Address – Hexadecimal address of the data in the next column.
- Data Blocks – Hexadecimal data, shown in 1- or 2-byte blocks.
- ASCII – ASCII representation of the corresponding line of data.

8.10.2 EEPROM Window Menu

Below are the menu items in the EEPROM right mouse button menu.

- Close
  Close this window.
- Find
  Find text specified in the Find dialog in this window.
- Find Next
  Find the next instance of Find text.
  F3 repeats the last Find.
  Shift+F3 reverses the direction of the last Find.
- Go To
  Go to the address specified in the Go To dialog.
- Import Table
  Open the Import dialog (see **Section 9.8 “File Management Dialog”**.)
- Export Table
  Open the Export As dialog (see **Section 9.8 “File Management Dialog”**.)
- Fill Memory
  Fill memory from Start Address to End Address with the value in Data. See **Section 9.9 “Fill Memory/Registers Dialog”**.
- Output to File
  Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 9.8 “File Management Dialog”**), with the addition of an “Output Range”. Select the type of range, either “Lines” or “Address”, and then enter the “Start” and “End” values to output.
- Print
  Print the contents of the window.

- Refresh
  Refresh the data in this window.
- Properties
  Set up window properties. See **Section 9.17 "Properties Dialog"**.

8.10.3 EEPROM Window FAQ

How do I:

- Fill EEPROM memory with a value?
  Right-click in the window and select "Fill Registers" to open a dialog where you may enter fill data.
  If you are using a hardware debug tool, additional steps may be required to write EEPROM on the part. See documentation for your tool.

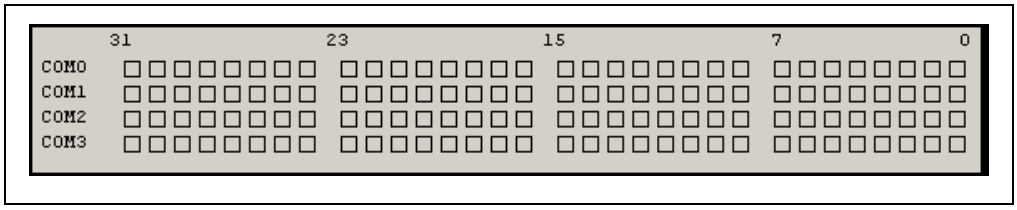
8.11 LCD PIXEL WINDOW

The LCD Pixel window displays LCD output (upper portion of window) when the device LCD function is enabled (SFR's in lower portion of window). All LCD pixels (squares) will be background gray when LCD functionality is disabled. When enabled, pixels will appear either white (off or 0) or dark gray (on or 1).

- LCD Pixel Window Display
- LCD Pixel Window Menu
- LCD Pixel Window FAQ

8.11.1 LCD Pixel Window Display

For devices that support LCD output, you can view results of this by selecting *View>LCD Pixel*.



- LCD display area
  LCD disabled: all LCD pixels will be background gray color
  LCD enabled: off (0) LCD pixels will appear white and on (1) LCD pixels will appear dark gray

> **Note:** You will have to enable LCD functionality through an LCD control register bit (e.g., for PIC16C924, set LCDCON register bit 7 (LCDEN) to 1). Consult your device data sheet.

- Address – Address of LCD-related SFR. Click to alternate list order of address, i.e., high/low or low/high.
- SFR Name – Symbolic name of special function register related to LCD function.
- Hex – Hexidecimal value of SFR contents.
- Decimal – Decimal value of SFR contents.
- Binary – Binary value of SFR contents.
- Char – ANSI character value of SFR contents.

8.11.2 LCD Pixel Window Menu

Below are the menu items in the LCD Display window right mouse button menu.

- Close

  Close this window.
- Bitfield Mouseover

  Enable/disable bitfield mouseover. When enabled, mousing over a symbol in the window will pop up a display of the bitfield for that symbol.
- Find

  Find text in this window specified in the Find dialog. See **Section 9.11 “Find and Replace Dialogs”**.
- Find Next

  Find the next instance of Find text.

  F3 repeats the last Find.

  Shift+F3 reverses the direction of the last Find.
- Go To

  Go to the address specified in the Go To dialog.
- Fill Registers

  Fill registers from Start Address to End Address with the value in Data. See **Section 9.9 “Fill Memory/Registers Dialog”**.
- Output to File

  Write the displayed window contents to a text file.
- Print

  Print the contents of the window.
- Refresh

  Refresh the data in this window.
- Properties

  Set up window properties. See **Section 9.17 “Properties Dialog”**.

8.11.3 LCD Pixel Window FAQ

How do I:

- Enable LCD operation?

  Consult your device data sheet to determine the LCD control registers needed to set up LCD functionality (e.g., for PIC16C924, set LCDCON register bit 7 (LCDEN) to 1 to enable). Set up these registers for LCD operation in your code or find these SFR's in the lower portion of the window and click in a value field of the SFR (hex, decimal, binary or char) to enter an appropriate value.
- Turn LCD pixels on or off in the display?

  Consult your device data sheet to determine the LCD pixel control registers. Set up these registers for LCD operation in your code or find these SFR's in the lower portion of the window and click in a value field of the SFR (hex, decimal, binary or char) to enter an appropriate value. The corresponding value should appear in the LCD display in the upper portion of the window.

8.12 WATCH WINDOW

The Watch window allows you to monitor program symbols while running your program. Up to four different watches may be set up on different tabs of this window.

- Watch Window Display
- Watch Window Menu
- Watch Window FAQ

8.12.1 Watch Window Display

Data is displayed in the following columns:

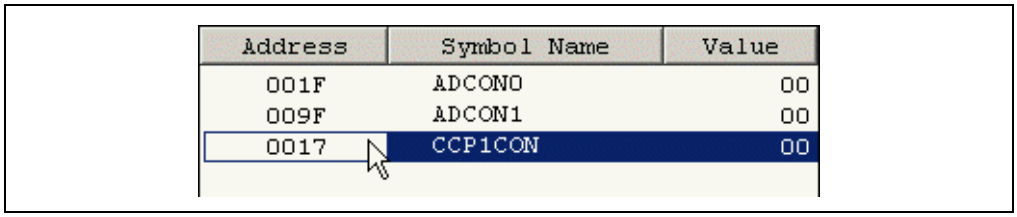
- Address – Hexadecimal address of the SFR or Symbol.

  To add an absolute address to a Watch view, click in the Address column of the first available row and enter a hex address.
- Symbol Name – Name of the SFR or Symbol.

  To add a special function register to a Watch view, select the SFR from the drop-down list and then click **Add SFR**. To add a symbol to a Watch view, select the symbol from the drop-down list and then click **Add Symbol**.
- Value – Current value of the SFR or Symbol.

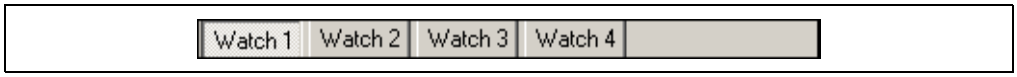
  To change a value in a Watch view, click in the Value column and type the new value. Then click outside the column to see the value update.

You may add radix information to the display (i.e., Hex, Decimal, Binary, Char) by right-clicking on the column header bar.



Watch variables may be rearranged using drag-and-drop. Also, you may drag-and-drop SFR's or symbols from the SFR, File Register or Editor window to the Watch window. Conversely, you may drag-and-drop items from the Watch window to the Editor window.

The Watch window now has sorting capability for all columns.



You may select and set up any one of four Watch views by clicking on one of the buttons on the lower left of the window.

Watch Window – C Language Usage

If you are developing your application code in C, the following features are available to you:

- Bitfield value mouseover available. Enable/disable this feature using the right-mouse button menu in the Watch window. The Watch window must be the active window (in focus) for this feature to work.
- Click in the Symbol Name field and enter a pointer name, e.g., `*count`. The variable must be a pointer to an intrinsic type or structure.
- Click in the Symbol Name field and enter a structure member name, e.g., `porta.ra0`. The variable must be of the form `struct.membername`.
- Click in the Symbol Name field and enter a structure pointer member name, e.g., `porta->pin`. The variable must be of the form `struct->pointername`.

Symbol names may be typed, copied/pasted or dragged/dropped. Also, you can select from the SFR/Symbol lists, and then edit the Symbol name to add the extra info.

Note: The watch window will display only 32 bits for MPLAB C30 type `long long int` and only 16 bits for MPLAB C18 type `short long int`. Change the display through the Watch dialog.

Example 1:

```
int J = 2;
int* pint = &J;
```

Enter in Watch: `*pInt`, Value: 2

Example 2:

```
char[] string = "Test";
char* pstring = string;
```

Enter in Watch: `*pstring`, Value: "Test"

Example 3:

```
typedef struct
{
int str_int;
char str_ch;
} TestStruct;
TestStruct struct1;

TestStruct* pstruct1 = &struct1;
pstruct1->str_ch = 'A';
```

Enter in Watch: `*pstruct1`, Value: tree for struct.

Enter in Watch: `struct1.str_int`, Value: 'A'

8.12.2 Watch Window Menu

Below are the menu items in the Watch window right mouse button menu.

- Close
  Close this window.
- Find
  Find text specified in the Find dialog in this window.
- Find Next
  Find the next instance of Find text.
  F3 repeats the last Find.
  Shift+F3 reverses the direction of the last Find.
- Add
  Add a Watch item to the currently-selected Watch tab. See **Section 9.3 "Add Watch Dialog"**.
- Delete
  Delete the selected Watch item from the currently-selected Watch tab.
- Save Watch Tab
  Save the contents of the currently-select Watch tab to a file.
- Load Watch Tab
  Load the contents of a file containing previously-saved Watch information into the currently-selected tab.
- Add Watch Tab
  Add a Watch tab to the Watch window. You can have as many as 16 Watch tabs.
- Rename Watch Tab
  Rename the currently-selected Watch tab.
- Remove Watch Tab
  Remove the currently-selected Watch tab. You can have no fewer than four Watch tabs.
- Import Table
  Save Watch data to a table. See **Section 9.23 "Table Setup Dialog"**.
- Export Table
  Load Watch data from a table. See **Section 9.23 "Table Setup Dialog"**.
- Output to File
  Write the displayed window contents to a text file.
- Print
  Print the contents of the window.
- Refresh
  Refresh the data in this window.
- Properties
  Opens the Watch dialog, so you can set up fonts and colors, as well as other Watch window properties. See **Section 9.26 "Watch Dialog"**.

8.12.3 Watch Window FAQ

How do I:

- Add a watch (address, SFR or symbol) to a watch window?

  Select the Watch window you want by clicking on the appropriate button in the bottom of the window.

  To add a watch at a specific **address**, click in the Address column of the first available row and enter a hex address.

  To add a **special function register** to a Watch view, select the SFR from the list next to the Add SFR button and then click on the button. To add a **symbol** to a Watch view, select the symbol from the list next to the Add Symbol button and then click on the button.
- Save the watch contents to a file?

  Click the right mouse button in the window to open a menu. Select Save Watch to save the currently selected watch to a file. Or, select Export to save Watch values to a file.
- Load a previously saved watch window?

  Click the right mouse button in the window to open a menu. Select Load Watch to load a previously saved watch into the currently selected watch. Or, select Import to load Watch values.
- Change the order of watch items?

  Simply drag and drop items where you want them in the watch list.
- Watch a single bit from a register?

  See **Section 9.26 "Watch Dialog"**.
- Watch C pointers and structure members?

  See Watch Window – C Language Usage.

8.13 SPECIAL FUNCTION REGISTERS WINDOW

The Special Function Registers window displays the contents of the Special Function Registers (SFRs) for the selected processor.

Note:	If a data memory register is not physically implemented on a device, it may not appear in the SFR list. Some tools, such as simulators, may allow you to see registers that do not exist on the actual device, such as prescalars.

The format provided by this window is more useful for viewing the SFRs than the normal file register window, since each SFR name is included and several number formats are presented. Whenever a break occurs, the contents of the Special Function Registers are updated.

Note:	If "Freeze Peripherals On Halt" is selected, the I/O port bits in the SFR or the watch windows will not update when single stepping. The pin will be modified, but the read request to retrieve the new value is blocked by the freeze and cannot be updated until the next step or run command.

- SFRs Window Display
- SFRs Window Menu
- SFRs Window FAQ

8.13.1 SFRs Window Display

Data is displayed in the following columns.

- Address – SFR hexadecimal address.
- SFR Name – Symbolic name for the SFR.
- Hex – Hexadecimal value, shown in 1-byte blocks.
- Decimal – Value in decimal format.
- Binary – Value in binary format.
- Char – Value in character format.

8.13.2 SFRs Window Menu

Below are the menu items in the Special Function Registers right mouse button menu.

- Close

  Close this window.
- Bitfield Mouseover

  Enable/disable bitfield value mouseover. SFR window must be the active window (in focus) for this feature to work.
- Find

  Find text specified in the Find dialog in this window.
- Find Next

  Find the next instance of Find text.

  F3 repeats the last Find.

  Shift+F3 reverses the direction of the last Find.
- Go To

  Go to the address specified in the Go To dialog.
- Fill Registers

  Fill registers from Start Address to End Address with the value in Data. See **Section 9.9 "Fill Memory/Registers Dialog"**.

Note:	Not all SFR's are writable and some are mirrored.

- Output to File

  Write the displayed window contents to a text file. Uses a Save As dialog (see **Section 9.8 "File Management Dialog"**), with the addition of an "Output Range" and tab-delimited option checkbox. Select the type of range, either "Lines" or "Address", and then enter the "Start" and "End" values to output.
- Print

  Print the contents of the window.
- Refresh

  Refresh the data in this window.
- Properties

  Set up window properties. See **Section 9.17 "Properties Dialog"**.

8.13.3 SFRs Window FAQ

How do I:

- Fill all registers with a value?

  Right-click in the window and select "Fill Registers" to open a dialog where you may enter fill data.

8.14 TRACE MEMORY WINDOW

The Trace memory window (under the View menu) helps you to monitor much of the processor operation. Up to 32767 instruction cycles can be displayed.

- Trace Window Display
- Trace Window Menu
- Trace Window FAQ

8.14.1 Trace Window Display

The Trace memory window contains the following information for each execution cycle:

- Line Number (**Line**) – Cycle's position relative to the trigger or halting point.
- Address (**Addr**) – Address of the instruction being fetched from program memory.
- Opcode (**Op**) – Instruction being fetched.
- Label (**Label**) – Label (if any) associated with the program memory address.
- Instruction (**Instruction**) – Disassembled instruction.
- Source Data Address (**SA**) – Address or symbol of the source data, if applicable.
- Source Data Value (**SD**) – Value of the source data, if applicable.
- Destination Data Address (**DA**) – Address or symbol of the destination data, if applicable.
- Destination Data Value (**DD**) – Value of the destination data, if applicable.
- Time Stamp (**Cycles/Time**) – Time stamp value in cycles or seconds
- External Inputs (**Probe #**) – Value of the external inputs (if any).

The approximate trigger cycle will be numbered as cycle 0. All other cycles will be numbered based on this cycle. Cycles that occurred before the trigger point will have a negative cycle number, and cycles that occurred after the trigger point will have a positive number.

The Trace memory window may be left open at all times, moved or resized. Data in the window may be edited “in place”.

8.14.2 Trace Window Menu

Below are the menu items in the Trace memory right-click menu.

- Close

  Close this window.
- Find

  Opens the Find dialog. In the Find What field, enter a string of text you want to find, or select text from the drop-down list. You can also select text in the edit window or place the cursor over a word you want to search for, before you open the Find dialog.

  In the Find dialog you may select any of the available options and the direction you want to search. Up searches backward from the insertion point, Down searches forward.
- Find Next

  Find the next instance of Find text.

  F3 repeats the last Find.

  Shift+F3 reverses the direction of the last Find.

- Go To

  Jump to the line specified in the dialog. Line number is shown in the first column of the window.
- Go To Source Line

  Go to the source code line corresponding to the selected trace code line.
- Jump to Trigger

  Jump to the location of the trigger.
- Jump to Top

  Jump to the top of the window.
- Jump to Bottom

  Jump to the bottom of the window.
- Reload

  Reload the trace memory window with the contents of the trace buffer.
- Reset Time Stamp

  Reset the time stamp conditionally on processor reset, on run or manually. Or, force an immediate reset by selecting Reset Now.
- Display Time

  Display the time stamp as a cycle count, in elapsed seconds or in engineering format.
- Symbolic Disassembly

  Instead of numeric address for SFR's and symbols, display the names in listing.
- Output to File

  Export the contents of the trace memory window to a file. Uses a Save As dialog (see **Section 9.8 "File Management Dialog"**), with the addition of cycle and tab information. Enter a "Start" and "End" cycle to write to the file. Also specify if the text is to be tab-delimited.
- Print

  Print the contents of the trace memory window.
- Refresh

  Refresh the viewable contents of the window.
- Properties

  Set up window properties. See **Section 9.17 "Properties Dialog"**.

8.14.3 Trace Window FAQ

How do I:

- Set up a simple trace?

  In the file (editor) window containing application code, select either "Add Filter-in Trace" or "Add Filter-out Trace" from the right-mouse menu. For more information on filter trace, see **Section 11.5 "Working with Debug Features"**.
- Set up a more complex trace?

  Many debug tools allow you to set up several conditions to define a trace. Consult the documentation for that tool to determine how to set up this type of trace.

8.15 CONFIGURATION BITS WINDOW

The Configuration Bits window displays information on configuration bit settings. MPLAB IDE v6.xx now recognizes configuration bits set in code as well as in the Configuration Bits window. Any configuration bit whose value is not set in code or the window defaults to 1.

Configuration bit settings made in code are reflected in the window upon loading a program (i.e., build, import or open a project). Configuration bit settings made in the window are reflected immediately. MPLAB IDE uses the current values of the window during debugger execution.

If “Clear program memory upon loading a program” is checked in *Configure>Settings*, **Program Loading** tab, configuration bits are cleared and then code changes are loaded upon loading a program. If no configuration bit settings are specified in code, the bits are simply cleared.

When you close a workspace, MPLAB IDE saves the latest configuration bit information in the workspace file. When you open a workspace, MPLAB IDE clears memory, loads code setups of configuration bits and then loads the last Configuration Bits window information saved in the workspace.

- Configuration Bits Window Display
- Configuration Bits Window Menu
- Configuration Bits Window FAQ

8.15.1 Configuration Bits Window Display

Selecting *Configure>Configuration Bits* opens the Configuration Bits window. Use this window to set configuration bit values for your selected device.

Note:	These values are obtained when building a source file and are used to control the operation of the debug tool.

- Address – The address of the configuration word/byte.
- Value – The current value of the configuration word/byte.
- Category – The name of the configuration bit in the corresponding configuration word/byte.
- Setting – The current setting of the configuration bit. Use the drop-down list to change the setting. The Value of the configuration word/byte will change accordingly.

8.15.2 Configuration Bits Window Menu

Below are the menu items in the Configuration Bits right mouse button menu. Click on an item to expand or collapse the description.

- Close

  Close this window.
- Reset to Defaults

  Restore the window defaults. This would correspond to the power-up state of the configuration bits on the actual device.
- Refresh

  Refresh the data in this window.
- Properties

  Set up window properties. See **Section 9.17 “Properties Dialog”**.

8.15.3 Configuration Bits Window FAQ

How do I:

- Clear configuration bits upon loading a program?
  Select *Configure>Settings* and click on the **Program Loading** tab. Check the box for Clear configuration bits upon program loading.
- Develop with the Configuration Bits window?
  To use only the Configuration Bits window to develop your application:
  - Do not set any configuration bits in code, i.e., do not use initialization data.
  - *Configure>Settings*, **Program Loading** tab, "Clear program memory upon loading a program" should be unchecked.

  Once you have completed development, you will then have to copy your Configuration Bits window settings into initialization data.
- Develop with configuration bits set in code?
  To use only configuration bits in code, i.e., initialization data, to develop your application:
  - Do not change any configuration bit settings in the Configuration Bits window.
  - *Configure>Settings*, **Program Loading** tab, "Clear program memory upon loading a program" should be checked.

8.16 FILE (EDITOR) WINDOW

A File window contains the source code for your application. You may open an existing source code text file in this window (*File>Open*) or open a blank window and type in your code (*File>New*). Once you have included a file in a project, and if you have it open when saving the project, it will open every time you open your project (*Project>Open*).

Text in a file window is edited using the MPLAB Editor. See documentation for the editor for more on its operation.

- File Window Display
- File Window Menu
- File Window FAQ

8.16.1 File Window Display

You may split the File window by clicking and dragging on the rectangular "nub" before the scroll arrow (see image below). Splitting the window allows you to view different sections of long code.

There is a right mouse button menu available in this window which contains several text editing and debugging options. Additional text editing options may be found in the Edit menu (see MPLAB Editor help for more information.) Additional debugging options may be found on the Debugger menu (after you have selected a debug tool.)

Code execution information may be displayed in the form of symbols in the gutter of the window.

If you mouseover (more your cursor over) a variable, the variable address is displayed in a pop-up.

8.16.2 File Window Menu

Below are the menu items in the File window right mouse button menu.

- Remove All Filter Traces

  Remove all filter trace tags on code text. See **Section 11.5 “Working with Debug Features”**.
- Add Filter-in Trace

  Add filter-in trace tags on selected code text. See **Section 11.5 “Working with Debug Features”**.
- Add Filter-out Trace

  Add filter-out trace tags on selected code text. See **Section 11.5 “Working with Debug Features”**.
- Remove Filter Trace

  Remove filter trace tags on selected code text. See **Section 11.5 “Working with Debug Features”**.
- Close

  Close active window.
- Set/Remove Breakpoint

  Set or remove a breakpoint at the currently-selected line.
- Enable/Disable Break

  Enable or disable a breakpoint at the currently selected line.
- Breakpoints

  Disable, enable or remove all breakpoints.
- Run to Cursor

  Run the program to the current cursor location. Formerly Run to Here.
- Set PC at Cursor

  Set the program counter (PC) to the cursor location.
- Cut

  Deletes the selected text in the current window and places it on the clipboard. After this operation you can paste the deleted text into another MPLAB Editor window, into a different location in the same MPLAB Editor window, or into another Windows application.
- Copy

  Copies the selected text in the current window onto the clipboard. After this operation, you can paste the copied text into another MPLAB Editor window, into another location in the same MPLAB Editor window, or into another Windows application.
- Paste

  Pastes the contents of the clipboard into the current window at the insertion point. You can only perform this operation if the clipboard contains data in text format. MPLAB Editor does not support pasting of bitmaps or other clipboard formats.
- Delete

  Deletes the selected text.
- Add to Project

  Insert file into the current project. Depending on the type of file, MPLAB IDE will sort the file into the correct type within the project window tree.

- Advanced

  Set advanced text features. Make selected text all uppercase or lowercase, a comment or not a comment, indented or outdented or match it if a brace, bracket or parenthesis.
- Bookmark

  Manage bookmarks. Toggle (enable/disable) a bookmark, move to the next or previous bookmarks or clear all bookmarks. See MPLAB Editor on-line help for more on bookmarks.
- Text Mode

  Customize text display based on development mode, i.e., device architecture and programming language.
- Properties

  Set Editor Options, either display or functional options. See **Section 11.2.1 "Editor Options Dialog"**.

8.16.3 File Window FAQ

How do I:

- Color my code based on its type?

  MPLAB IDE will automatically color your code. To change this setting, right-click in the window, select Text Mode and then select your desired type, either device-specific assembly, C, Basic or SCL (simulator control language).
- Set the color-coding?

  Right-click in the window and select Properties. In the Editor Options dialog, click the **Text** tab and **Choose Colors**.

Chapter 9. MPLAB IDE Dialogs

9.1 INTRODUCTION

MPLAB IDE dialog boxes behave as normal Windows applications. The basic dialogs available in MPLAB IDE are listed below. Depending on the tools you have installed, other dialogs may be available.

Dialogs covered in this chapter, listed alphabetically:

- About MPLAB IDE Dialog
- Add Watch Dialog
- Breakpoints Dialog
- Build Options Dialog
- Export Hex File Dialog
- External Memory Setting Dialog
- File Management Dialog (Open, Save As, Add Files)
- Fill Memory/Registers Dialog
- Find In Project Files Dialog
- Find and Replace Dialogs
- Help Topics Dialog
- Import Dialog
- New Project Dialog
- Project-Display Preferences Dialog
- Project Wizard Dialogs
- Properties Dialog
- Save Project As Dialog
- Select Device Dialog
- Select Language Toolsuite Dialog
- Set Language Tool Location Dialog
- Settings Dialog
- Table Setup Dialog
- User ID Memory Dialog
- Version-Control Dialog
- Watch Dialog

9.2 ABOUT MPLAB IDE DIALOG

Select *Help>About MPLAB IDE* to open the About MPLAB IDE dialog. This is an informational dialog.

Click **OK** to close the dialog.

Trademark Information

The first section contains trademark information for Microchip products. In addition, references to other products are discussed.

Component and Plug-in Information

A list box contains the following information:

- Name – Name of the component/plug-in
- Version – Version number for that component/plug-in
- MPLAB Certified – Whether or not the component/plug-in is MPLAB Certified. For more information on certification, contact: joe.drzewiecki@microchip.com.

Clicking on an item in the list will reveal the path to that module in "Module path".

9.3 ADD WATCH DIALOG

Right-click in the Watch window and select **Add** to open the Add Watch dialog. Use this dialog to add a watch item to the currently-selected Watch tab. When you have finished adding items, click **Close.**

Related windows/dialogs are:

- **Section 8.12 "Watch Window"**
- **Section 9.23 "Table Setup Dialog"**
- **Section 9.26 "Watch Dialog"**

Dialog	Description
Add an SFR	Select a Special Function Register (SFR) name from the pull-down list. Then click **Add SFR**.
Add a Symbol	Select a symbol name from the pull-down list. Then click **Add Symbol**.
Add an Absolute Address or Address Range	First select a type of memory. The range of possible addresses will be displayed in Range. Second, select the format in which you will enter the address. Finally, enter a Start Address and either an End Address or the Word Count. Click **Add Address**.

9.4 BREAKPOINTS DIALOG

Select *Debugger>Breakpoints* to open the Breakpoints dialog. You must select a debug tool before this option is available on the Debugger menu.

Note: There are other ways to set breakpoints. See **Section 3.18 "Using Breakpoints"**.

Dialog	Description
Break At:	Enter the source code line number or address of a breakpoint. Hit <Enter> to place the breakpoint in the list below.
Program Memory Breakpoints	List of breakpoints set in program memory. These breakpoints may be selected and then acted on by the buttons to the right.
Active Breakpoint Limit	Total number of breakpoints that may be set for selected debug tool.
Available Breakpoints	How many breakpoints are left after subtracting the code breakpoints AND advanced breakpoints (if applicable). **Note:** "Run to Cursor" and "Step Over" commands use a breakpoint to run and stop. If none is available, these commands step-and-compare until they reach their stop location, which may cause them to execute slower.

Enter a breakpoint

Enter a breakpoint location in the "Break At" box.

- Enter a line number, e.g., 27. You must also enter the source file path and name for this to work, e.g., `c:\project1\program1.asm, 27`.

- Enter a line address (in hex), e.g., `002A`.

You will see the breakpoint you are typing also appear in the "Program Memory Breakpoints" box. Once you have entered a breakpoint, hit return to see it entered in the "Program Memory Breakpoints" box. A checkbox will appear in front of the breakpoint so that you may enable/disable this breakpoint.

Remove a breakpoint

Click on a breakpoint in the "Program Memory Breakpoints" box to highlight it. Then click the **Remove** button.

Remove all breakpoints

Click the **Remove All** button to delete all breakpoints from the "Program Memory Breakpoints" box.

Enable or disable a breakpoint

Click the checkbox in front of the breakpoint to alternately enable/disable it.

Enable or disable all breakpoints

Click either the **Enable All** or **Disable All** button to enable or disable all the breakpoints listing in the "Program Memory Breakpoints" box, respectively.

Fix unresolved breakpoints

If you have set a breakpoint on a line of high-level language code (e.g., C code) that cannot be resolved (into assembly code), you will get a warning message. Also, in the breakpoint dialog, a yellow question mark will appear next to the unresolved breakpoint.

Often optimized code will cause this problem. Rebuild your application without optimizing and try the breakpoint again.

Another solution is to simply move the breakpoint to another line of code that can be resolved.

Save/cancel changes

Click **OK** to save and implement your changes, and close the dialog. Click **Cancel** to close the dialog without saving or implementing your changes.

9.5 BUILD OPTIONS DIALOG

Select *Project>Build Options>Project* to open the Build Options dialog. You must have a project open before this option is selectable on the Project menu.

9.5.1 General Tab

Enter or browse for directories and search paths that MPLAB IDE will use when building a project. Directories are one specific path to files, whereas search paths may include several paths, separated by semicolons (;). Relative paths (e.g., ..\tmp) will be relative to the source file location.

Note: These are MPLAB IDE paths only, e.g., MPASM assembler does not use the Assembler Include Path or Include Path information. Please consult your language tool documentation to determine if it uses this information.

Dialog	Description
Output Directory	Path to directory containing files that result only from a full build, specifically the link step. This is where the COD, COF and hex files go, as well as list and map files. These files are affected by *Project>Clean*.
Intermediates Directory	Path to the directory containing intermediate files generated when a project is built. To delete these files, use *Project>Clean*.
Assembler Include Path	Path(s) to the directory(ies) containing assembler include files (.inc).
Include Path	Path(s) to the directory(ies) containing include files (.h).
Library Path	Path(s) to the directory(ies) containing library files (.lib, .arc).
Linker Script Path	Path(s) to the directory(ies) containing linker script files (.lkr, .gld).
Suite Defaults	Replaces the current directory and path selection with default values. Defaults are set in *Project>Set Language Tool Locations*.

9.5.2 Language Tool Tabs

Set up language tools in the selected suite. See on-line help for each tool for more information.

9.6 EXPORT HEX FILE DIALOG

The Export Hex File dialog is available from *File>Export*.

Use Export to save program memory, a section of program memory, EEPROM memory, configuration bits or user ID to a `.hex file`. There are two formats allowed. The INHX32 format should be used for most purposes. The INHX8S format saves the memory area as two files: one for the even bytes and one for the odd bytes. This is useful where program memory is stored in pairs of external 8-bit wide memory chips.

To export a hex file:

1. Click on the **File Format** tab. Select the Hex file format.
2. Click on the **Memory Areas** tab. Select which memory areas to export and, if applicable, their range.
3. Click **OK**.
4. In the Save As dialog, enter the name and directory location of the file and click **Save**.

9.7 EXTERNAL MEMORY SETTING DIALOG

Some PICmicro MCU and dsPIC DSC devices support extended program memory through the use of off-chip or external memory. These devices have modes you may select, such as Extended Microcontroller (some on-chip and some off-chip program memory) and Microprocessor (all off-chip program memory).

To enable external memory use in a debug tool, a device that supports this feature must be selected (*Configure>Select Device*). Then you may select *Configure>External Memory*. This will open the External Memory Setting dialog.

- **Use External Memory** – Enable/disable external memory use
- **Address Range** – The **Start** address is always set at the end of internal memory. Enter the **End** address for external memory

Once enabled, the external program memory will appear in the program memory window.

Click **OK** to save your changes and close the dialog. Click **Cancel** to close the dialog without saving your changes.

9.8 FILE MANAGEMENT DIALOG

A file management dialog allows you to manage source and project files. The types of file management dialogs in MPLAB IDE are:

Input

- **Open** dialog – Open an existing file, project or workspace or import an existing debug file.
- **Add Files to Project** dialog – Insert source files into your project.
- **Import** dialog – Import tabular data from a file into a memory window.

Output

- **Save As** dialog – Save a file, project or workspace in a different folder or with a different name.
- **Export As** dialog – Export tabular data from a memory window into a file.

9.8.1 File Management

Control	Dialog	Description
Look In	Input	Select the folder containing the file/project from the drop-down list. Default is previously-viewed directory.
Save In	Output	
Menu of Files	All	A list of files/projects found in the selected folder. The types of files displayed is selected in the "Files of Type"/ "Save As Type" list box. Click on a file here to select it or enter a name in the "File Name" text box. This text box has the following edit features: delete file, rename file and drag-and-drop copy file.
File Name	All	Enter a name for the file/project or select one from the Menu of Files text box.
Files of Type	Input	Select from the list the types of files you wish to view in the Menu of Files text box.
Save As Type	Output	
Start Address	Import/Export As	Specify the address at which to start the import/export of data.
End Address, Single Column Output	Export As	Specify the address at which to stop the export of data. Also check the checkbox for "Single Column Output" or uncheck for multicolumn output.

9.8.2 Navigation Buttons

- Go to last folder visited
- Up one level
- Create new folder
- View menu of files as Thumbnails, Tiles, Icons, List or Details.

9.9 FILL MEMORY/REGISTERS DIALOG

To fill memory with a value, right-click in one of the windows below and select “Fill Memory”.

- **Section 8.8 “Program Memory Window”**
- **Section 8.10 “EEPROM Window”**

To fill registers with a value, right-click in one of the windows below and select “Fill Registers”.

- **Section 8.9 “File Registers Window”**
- **Section 8.13 “Special Function Registers Window”**

Enter a “HI-TECH” and an “End Address” as the fill range. If you want to save this range so it always appears in this dialog, check “Retain Address Range”.

Enter a value in “Data” to be filled in each address in the specified range or check ”Randomize Data”. Check “Sequence Start” to fill memory with sequential data.

Select the “Data Radix”, either Hexadecimal or Decimal.

Click **Write** to fill memory or register range. Click **Close** to abort.

9.10 FIND IN PROJECT FILES DIALOG

Selecting *Project>Find In Project Files* opens a dialog that allows you to search for a text string in multiple files in a project. The search results appear in the Output window.

9.11 FIND AND REPLACE DIALOGS

Use the Find dialog (*Edit>Find*) to find an occurrence of a text string in a file (editor) window. Use the Replace (*Edit>Replace*) dialog to substitute one text string for another in a file (editor) window.

Item	Dialog	Description
Find what:	Find, Replace	Enter a string of text you want to find, or select text from the drop-down list. You can also select text in the file window or place the cursor over a word you want to search for, before you open the dialog.
Replace with:	Replace	Enter a string of text with which you want to replace the find text, or select text from the drop-down list.
Match whole word only	Find, Replace	Text in the file window must match the “Find what” text as a whole word, not part of a larger word.
Match case	Find, Replace	Text in the file window must match the case of the “Find what” text. E.g., “prog1” would not be a match for “Prog1”.
Direction	Find, Replace	Up searches backward from the cursor position; Down searches forward.
Find Next	Find, Replace	Highlight the next occurrence of the “Find what” text.
Replace	Replace	Replace the highlighted text with the “Replace with” text.
Replace All	Replace	Replace all occurrences of “Find what” text with “Replace with” text.

9.12 HELP TOPICS DIALOG

Select *Help>Topics* to open the MPLAB Help Topics dialog. Use this dialog to select a help file to display. Help files are arranged according to the type of development tool they describe, i.e., System, Language Tools, Debuggers, Programmers and (Other) Tools. You may only select one help file to open.

Double-click on your selection or click to select and then click **OK** to open the help file. Click **Cancel** to close the dialog without opening a help file.

9.13 IMPORT DIALOG

The Import dialog is available from *File>Import*. It is basically an Open dialog (see **Section 9.8 "File Management Dialog"**.)

Use Import to load a `.hex` file into memory. The hex file may contain data for program memory, data EEPROM, configuration bits and ID locations. This is useful if you want to program a device using a previously assembled or compiled project. When you do this, if the associated debug file is in the same directory as the `.hex` file, the symbols and line number information will also be loaded into the current project.

You can also import either of the debug files: the `.cof` or `.cod` file. These contain both the memory information from the associated `.hex` file and the symbol information from when that project was last built. These will be no notification that the import has completed, but the data in the IDE windows will change.

9.14 NEW PROJECT DIALOG

Enter the name and location of a new project (*Project>New*).

- Name – Enter a name for the new project.
- Directory – Enter a directory for the new project. You may do this in one of two ways:
  - Type in the path to an existing directory or type in the path to a new directory, in which case you will be prompted to create the directory when you click **OK**.
  - Click Browse to browse to an existing directory or to one level above where you wish to place a new directory. Click **OK** in the Browse for Folder dialog. Complete the path if you are creating a new directory and then click **OK**. You will be prompted to create the directory if it does not exist.

9.15 PROJECT-DISPLAY PREFERENCES DIALOG

Right-click in an empty area of the Project Window and select Preferences from the pop-up menu to display the Project-Display Preferences dialog. Use this dialog to set preferences for the Project window.

Dialog	Description
Display Project Nodes As	Select how you want project file types displayed on the project tree, either as simply names, or names with paths.
Display File Nodes As	Select how you want project files displayed on the project tree, either as simply file names, file names with paths or as specified internally in the file.
Refresh Version-Controlled Files	If your project is set up to use version-controlled files (*Project>Select Version-Control System*), select when you want the project window refreshed to reflect the version control status.Refresh means asking the version-control system, “Who has this file checked out?” for each and every file. This can be very slow, especially if you have many files in your project.If you disable both “Refresh on application focus” and “Auto refresh”, you will still get refreshes after performing version-control operations, and in response to the “Refresh” command on the Project window context menus.

9.16 PROJECT WIZARD DIALOGS

Select *Project>Project Wizard* to launch a sequence of dialogs for setting up your project. For more information, refer to **Section 4.2 “Using the Project Wizard”**.

9.17 PROPERTIES DIALOG

Right-clicking in a window and selecting Properties from the pop-up menu will display a dialog where you may set window properties such as font type and color, background color and column settings. Depending on the window, this dialog will may have different tabs. The generic tabs used are described below.

9.17.1 Column Settings Tab

Column heading used in the window display will be listed in a list box.

To Show/Hide a Column

- Check/uncheck the checkbox next to an item to display/hide the column.
- Click on an item to select it. Then use the **Show** or **Hide** buttons to display or hide the column.

To Reorder Columns

- Click on an item to select it. Then use the **Move Up** or **Move Down** buttons to display or hide the column.

To View Column Width

- Click on an item to select it. The column width will be shown in “Selected column width in pixels”.

To Change Column Width

- Make the window active.
- Move the cursor over the line between columns till it changes to the resize cursor
- Click and drag to widen or narrow the column.

To Restore Default Settings

- Click **Defaults**.

9.17.2 General Tab

Set up the font and change color for all debug windows, i.e., what you set up here will globally determine settings for all debug (View menu) windows.

Fonts

Click **Select Font** to open the standard Windows Font dialog where, for a selected script (e.g., Western), you may set up font type, font style and font size. To choose only non-proportional fonts in the Font dialog, check the checkbox for "Show Fixed Pitch Only".

Colors

Click **Change Color** to open the standard Windows Color dialog where you may set the change color. Change color is the color the text is displayed in when a change to that text occurs, e.g., when a `movwf PORTB` changes the value of PORTB from 00 to 55 in the SFR window, the 55 will be displayed in the change color.

9.18 SAVE PROJECT AS DIALOG

Select *Project>Save Project As* to open the Save Project As dialog.

When a project is saved to a new directory, all files in the project directory and its sub-directories will be copied to the new directory along with the new project and workspace files. Any files which exist outside of the project tree are considered to be shared files and are not copied. The context menu on the Project window will change to reflect the new location of the project.

To make projects (and related workspaces) portable, MPLAB IDE stores files in the project directory and its subdirectories with relative paths and files outside of the project directory with full paths.

The Save Project As dialog is functionally the same as a standard Save As dialog (see **Section 9.8 "File Management Dialog"**).

9.19 SELECT DEVICE DIALOG

To choose a device for development, select *Configure>Select Device* to open the Select Device dialog.

All devices currently supported by this version of MPLAB IDE will be listed under Device. Select from the list or start typing in device name.

Once you have selected a device from this list, the Microchip Programmer Tool Support section and the Microchip Debugger Tool Support section will reflect the available tool support for that device from Microchip Technology:

- Green button – support available
- Yellow button – Beta support available
- Red button – support not currently available

Note:	Mousing over a button will display its related support.

Click **OK** to save your changes and close the dialog. Click **Cancel** to close the dialog without saving your changes.

To choose a programmer, select from the list of *Programmer>Select Programmer*.

To choose a debug tool, select from the list of *Debugger>Select Tool*.

9.20 SELECT LANGUAGE TOOLSUITE DIALOG

Select *Project>Select Language Toolsuite* to open the Select Language Toolsuite dialog. You must have a project open before this option is selectable on the Project menu.

Use this dialog to select the suite of language tools you will use in your project. See documentation for the language tools to ensure that they support the device you will be using.

- **Active Toolsuite** – Select the toolsuite you will use.
- **Toolsuite Contents** – View the language tools associated with the toolsuite selected above. If these are not the tools you wanted, choose another toolsuite. Click on a language tool to see its location.
- **Location** – Change the path or file, enter new path or file information or **Browse** for the executable file of the language tool highlighted in the above list box.

A red “X” opposite a tool indicates that it has not been installed or that MPLAB IDE does not know where to find the tool. See **Section 9.21 “Set Language Tool Location Dialog”** form more information.

9.21 SET LANGUAGE TOOL LOCATION DIALOG

Select *Project>Set Language Tool Locations* to open the Set Language Tool Location dialog.

Use this dialog to set the path to individual language tool executables within a toolsuite.

- **Registered Tools** – Find the toolsuite you will be using (e.g., Microchip MPASM Toolsuite). Click on the “+” to expand.
  - Click on the “+” next to “Executables” to expand. Click on a tool to see its currently assigned path in “Location”.
  - Click on the “+” next to “Default Search Paths and Directories” to expand. Click on a path to see its current assigned in “Location”.
- **Location** – Change the path or file, enter new path or file information or **Browse** for the path or file.

9.22 SETTINGS DIALOG

Select *Configure>Settings* to open the Settings dialog and set up general MPLAB IDE functions. Click **OK** to save your changes and close the dialog. Click **Cancel** to close the dialog without saving your changes.

- Workspace Tab – set up workspace options
- Debugger Tab – set up debug options
- Program Loading Tab – set up clear/don't clear memory on program load
- Hot Keys Tab – set up hot keys for various IDE operations
- Projects Tab – set up project options

9.22.1 Workspace Tab

Select *Configure>Settings* and click the **Workspace** tab to set up Workspace options.

A **workspace** contains information on the selected device, debug tool and/or programmer, open windows and their location and other IDE configuration settings.

Option	Description
Automatically save workspace upon closing	To automatically save the contents of an active workspace when closing MPLAB IDE, select Yes. To not save the contents, select No. To be prompted to save the contents, select Prompt. Saving a workspace also saves project information.
Reload last workspace at startup	Reload the last workspace setting on startup of MPLAB IDE. This will also reload the last open project.
Recent files list contains	Specify the number of recent files to include on the list found under *File>Recent Files*.
Recent workspaces list contains	Specify the number of recent workspaces to include on the list found under *File>Recent Workspaces*.

9.22.2 Debugger Tab

Select *Configure>Settings* and click the **Debugger** tab to set up debug options.

Option	Description
Automatically save files before running	Make sure all project files are saved before running the selected debugger.
Browse for source if file is not found	If source file cannot be found, search for it using paths known to MPLAB IDE.
Show disassembly if source is unavailable	If source file cannot be found, open disassembly window.
Remove breakpoints upon importing a file	Clear all breakpoints upon importing a file, i.e., adding a debug/hex file using *File>Import*.

9.22.3 Program Loading Tab

Select *Configure>Settings* and click the **Program Loading** tab to set up whether memory is cleared or uncleared on program loading. A program load means a build, make or import.

When to Clear Memory

Option	Description
Clear memory before building a project	Clear all memory before a project is built.
Clear memory after successfully building a project	Clear all memory only after a project has built successfully (no errors.)

What Memory is to be Cleared

Option	Description
Clear program memory upon loading a program	Clear/Don't clear program memory on program load.Uncheck when using concurrent projects.
Clear configuration bits upon loading a program	Clear/Don't clear configuration bit values on program load.
Clear EE data upon loading a program	Clear/Don't clear EEPROM data memory on program load.
Clear user ID upon loading a program	Clear/Don't clear the user ID value on program load.

9.22.4 Hot Keys Tab

Select *Configure>Settings* and click the **Hot Keys** tab to set up hot key functions for various IDE operations. Hot keys or shortcut keys, are keyboard shortcuts to IDE menu or toolbar commands.

Function	Description
Hot Key mapping scheme	Use the default settings, or save your own using **Save As**. A file with a `.hot` extension will be created with your custom settings.
Command	Select the command for which you wish to assign a hot key, or to view the currently-selected hot key for that command.
Hot Key combination	Enter the hot key for the selected command or view the currently-selected hot key for the selected command.If you hit **Delete** or **Backspace**, this field will be cleared and display “None”. When you enter a hot key, the key value will then be displayed here. **Note:** You must hit the actual key(s) you wish to use, not type in the name/function. E.g., To set a hot key combination of **Ctrl + F8** for a command, hold down the **Ctrl** key and then hit the **F8** key.
Hot Key currently in use by	If the hot key you have chosen is already in use, its corresponding command will be displayed here.

9.22.4.1 VALID HOT KEYS

The following keys are not valid hot keys:

- Enter
- Backspace
- Delete
- Tab
- Any alphanumeric keys or shift of above keys

If you select a key that is not valid, Hot Key combination will reflect the invalid selection by displaying None.

9.22.4.2 MULTIPLE ASSIGNMENTS OF HOT KEYS

It is possible for hot keys to be assigned to multiple commands. If a conflict does exist, the hot keys will be applied to the first 'loaded' command encountered in the table.

E.g., Hot keys are assigned to an MPLAB ICD 2 debugger command and an MPLAB PM3 programmer command. However, MPLAB ICD 2 is not enabled when they keys are assigned, but MPLAB PM3 is. Therefore, only the MPLAB PM3 command would be applied. If both MPLAB ICD 2 and MPLAB PM3 are enabled, only the command first encountered in the hot key table will be applied.

Note: Assigning a hot key to a command does not remove any prior command assignments of that hot key.

9.22.5 Projects Tab

Select *Configure>Settings* and click the **Projects** tab to set up project options.

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.

Option	Description
Close open source files on project close	When a project is closed, close all related open source files.
Clear output window before build	Clear the contents of the output window and then build.
Save project before build	Save the active project setup before building.
Save files before build	To save all active project files before building, select Yes. To not save the files, select No. To be prompted to save the files, select Prompt.
Halt build on first failure	Halt build when the first error occurs. (Many errors may be reported for one source file that fails.)
Use one-to-one project-workspace model	Allow only one project in a workspace. For more on single and multiple project workspaces, see **Chapter 4. "Projects and Workspaces"**.

9.23 TABLE SETUP DIALOG

Right-click in a Watch window and select “Import Table” or “Export Table” to open the Table Setup dialog. Import or Export Watch Value data from or to a table file. Export SFR/Symbol values to a file and then import them back as needed.

Related windows/dialogs are:

- **Section 8.12 “Watch Window”**
- **Section 9.3 “Add Watch Dialog”**
- **Section 9.26 “Watch Dialog”**

Action	Dialog	Description
Import/Export	**Start Address**	Enter start address for type of memory selected below. Defaults to address of selected item in Watch window.
Export	**End Address**	Enter end address for type of memory selected below. Defaults to address of selected item in Watch window.
Import/Export	**Memory**	Choose type of memory to import/export. Defaults to full range when clicked.
Import/Export	**Symbol Start Address**	The symbolic representation of selected start address, if applicable.
Export	**Single Column Output**	Output written in a single column.
Import/Export	**Type Formatted**	Import use: Take a spreadsheet column and import it to an array of floats. Export use: Move floating point/decimal data into spreadsheet or graphing programs.

9.24 USER ID MEMORY DIALOG

Some PICmicro MCU and dsPIC DSC devices have memory locations where you can store checksum or other code identification (ID) numbers. These locations are readable and writable during program/verify. Depending on the device, they also may be accessible during normal execution through the `TBLRD` and `TBLWT` instructions.

Select *Configure>ID Memory* to open the User ID Memory dialog.

Dialog	Description
Use Unprotected Checksum	Use the unprotected (i.e., code protect off) checksum as the user ID. See the programming spec for your device to see how this checksum is calculated. **Note:** You must click **OK** and then reopen this dialog to see the unprotected checksum as the user id. Also, this value may change after any make or build.

9.25 VERSION-CONTROL DIALOG

Select *Project>Version Control* to open the Version-Control dialog. Set up your MPLAB IDE Project to use a version-control system (VCS).

- Version-Control System – Select an available version-control system from the drop-down list.
- System – Set up the necessary parameters to use your selected VCS, if required.
- For Project "*project*.mcp" – Set up the necessary project information for project *project*.mcp.

For more on version-control systems, see **Section 4.6 "Using A Version-Control System (VCS)"**.

Microsoft Visual SourceSafe

For Project "*project*.mcp"	Description
SRCSAFE.INI file	Specify the path to and name of the Visual SourceSafe initialization file for the database you will be using, e.g., `C:\VSS\VSS_6.0\win32\scrsafe.ini`. Click **Browse** to locate the file.
VSS project path	Specify the path within Visual SourceSafe to the project files, e.g., `$/Project Files/Project2`.

PVCS

System	Description
EXE directory	Enter or browse to the directory containing the PVSC executable files.
Java directory	Enter or browse to the directory containing PVSC java scripts.
Remember user name and password	Check to automatically access the PVSC system after entering your user name and password the first time.
For Project "*project*.mcp"	**Description**
Database directory	Enter or browse to the directory containing the PVSC database, from which the project files will come.
Archive directory	Enter or browse to the directory containing the PVSC archive, to which project files may be saved.

CVS

System	Description
CVS Executable	Enter or browse to the directory containing the CVS executable files.
Remember user name and password*	Check to automatically access the CVS system after entering your user name and password the first time.
For Project "*project*.mcp"	**Description**
Host	Enter the host name or IP address of the machine that is the CVS server. This should match the host name or IP address given to the `cvs login` command.
Port	Enter the port number for the CVS service on the CVS server. You can leave this blank if the server is using the default port value of 2401.
Root	Since a CVS server can host multiple roots, provide the name of the root with a leading forward slash.
Module	Enter either a real CVS module or a top-level directory within the given CVS root.

* The first time you use CVS with MPLAB IDE, you must log in to CVS outside of MPLAB IDE, i.e.;

```
$ cvs -d :pserver:MyUserName@ServerHostName:/CVSRoot login
Logging in to :pserver:MyUserName@ServerHostName:2401:/CVSRoot
CVS password: ********
```

Once you have done this, CVS will work with MPLAB IDE.

9.26 WATCH DIALOG

Set up the Watch window by right-clicking in the Watch window and selecting Properties. The Watch dialog has the following tabs:

- Watch Properties Tab
- Preferences Tab
- General Tab (See **Section 9.17.2 "General Tab"**.)

Related windows/dialogs are:

- **Section 8.12 "Watch Window"**
- **Section 9.3 "Add Watch Dialog"**
- **Section 9.23 "Table Setup Dialog"**

9.26.1 Watch Properties Tab

This tab is used to set the properties for the specific symbols in the Watch window.

Property	Description
Symbol	Select the watch symbol for which to set properties. Choose from the currently-selected and displayed symbols in the window.
Size	Select the bit size of the selected watch symbol value. Choose from 8, 16, 24, 32, 40 or 64 bits, depending on device.
Format	Select the format of the selected watch symbol value. Choose from: - Hex(adecimal) - IEEE Float - Binary - Single Bit - Decimal - dsPIC Integer - ASCII - dsPIC Fractional - MCHP Float - Double **Note:** With single bit, no information about which bit is being displayed will be shown in the Symbol Name.
Byte Order	View or change the byte order of the selected watch symbol.
Memory	View the type of memory for the selected watch symbol.

9.26.2 Preferences Tab

This tab is used to select default settings for new symbols added to the Watch window.

Setting	Description
Float Format	Select a floating-point format. • Use COFF float type – Use the information in the COFF file. **This is the default.** • IEEE 754 32-bit – Use with MPLAB C18 (v2.40 and later), MPLAB C30 and HI-TECH 'C' with -D32x. • IEEE Modified 24-bit – Use with HI-TECH 'C' compilers with -D24x. • Microchip High:Low – Use with MPLAB C18 compilers before v2.40. • Microchip Low:High – Use with CCS compilers. The float value selected will also be used to format the editor mouse-overs.
Unknown Type Format	Select an unknown type format. This format is used when no type can be specified. Choose from Hex(adecimal), Binary, Decimal or Char. Also choose the byte order (High:Low or Low:High). The format selected will be used for SFRs, absolute address and ASM symbols.

Chapter 10. MPLAB IDE Operational Reference

10.1 INTRODUCTION

Reference information about the operation of MPLAB IDE is discussed here.

- Command-Line Options
- Files Used by MPLAB IDE
- Saved Information

10.2 COMMAND-LINE OPTIONS

MPLAB IDE can be invoked through the command-line interface as follows:

`mplab [`*`file`*`] [/`*`option`*`]`

`file` = workspace.mcw

Open the workspace workspace.mcw in MPLAB IDE. Any projects contained in the workspace will be opened also.

`option` = nosplash

Do not display splash screen on program launch

10.3 FILES USED BY MPLAB IDE

The default extensions of MPLAB IDE files are listed in Table 10-1.

TABLE 10-1: MPLAB IDE DEFAULT EXTENSIONS

Extension	Definition
a	Archive (library) file – MPLAB LIB30
asm	Assembly language source file – MPASM assembler
c	C source file
chm	Compiled HTML Help file
cod	File containing symbolic information and object code
cof	File containing symbolic information and object code
err	Error file generated by assembler/compiler
evt	Event file – MPLAB ICE 2000
exe	Program/utility file
fsti	File stimulus file – MPLAB SIM for PIC17 MCUs
gld	Linker script file – MPLAB LINK30
h	C include file
hex	Machine code in hex format file
inc	Assembly language include file
lib	Library file – MPLIB librarian
lkr	Linker script file – MPLINK linker
lst	Absolute listing file generated by assembler/compiler
map	Map file generated by linker
mch	Exported data file

TABLE 10-1: MPLAB IDE DEFAULT EXTENSIONS (CONTINUED)

Extension	Definition
mcp	Project information file
mps	Build state file for Make
mcw	Workspace information file
o	Object file
psti	Pin stimulus file – MPLAB SIM for PIC17 MCUs
rsti	Register stimulus file – MPLAB SIM for PIC17 MCUs
ssti	Synchronous stimulus file – MPLAB SIM for PIC17 MCUs
s	Assembly language source file – MPLAB ASM30
sbs	SCL generator workbook file – MPLAB SIM
scl	SCL file from SCL generator – MPLAB SIM
stc	Stimulus scenario file – MPLAB SIM
trc	Trace save file
trg	Trigger file – MPLAB ICE 2000
xrf	Cross-reference file – MPASM assembler

10.4 SAVED INFORMATION

Information concerning your setup of MPLAB IDE is saved as follows.

Workspaces

A workspace contains the following information:

- Selected device, debug tool and/or programmer.
- Debug tool/programmer settings information.
- *Configure>Settings*, **Program Loading** tab information.
- Configuration bits settings.
- Open IDE windows and their location.
- Other IDE system settings.

This information is saved in the `.mcw` file.

Projects

A project contains the following information:

- The group of files needed to build an application.
- File associations to various build tools.
- Build options.

This information is saved in the `.mcp` file.

Registry

The following information is saved in the registry file of the Windows OS:

- Language tool names and installation locations.
- Most items on the *Configure>Settings*, **Workspace** tab.
- All items on the *Configure>Settings*, **Projects** tab.

INI Files

The following information is saved in initialization (`.ini`) files:

- Editor settings in the `mpeditor.ini` file.
- Individual window docking information is the `mplab.ini` file.

Part 3 – MPLAB Editor

Chapter 11. Using the Editor 151

NOTES:

Chapter 11. Using the Editor

11.1 INTRODUCTION

The MPLAB Editor is an integrated part of the MPLAB IDE Integrated Development Environment. The editor is always available when MPLAB IDE is running.

The MPLAB IDE and MPLAB Editor are designed to provide developers with an easy and quick method to develop and debug firmware for Microchip Technology's PICmicro microcontroller (MCU) and dsPIC digital signal controller (DSC) product families. For more information on these devices, visit the Microchip web site.

The editor functions in a File window of MPLAB IDE. Menu items (on File and Edit menus and a right-mouse menu) are available for editor control.

The MPLAB Editor allows you to use the following features for general text editing.

11.1.1 Inserting, Selecting and Deleting Text

You can enter text in either insert or strike-over mode. MPLAB Editor shows the mode as either "INS" or "OVR" on the status bar. The text you enter may contain any combination of the standard alpha-numeric and punctuation characters. You can enter additional characters with special keys and\or key combinations.

Text selection features allow you to select a character, word, entire line or other blocks of text. You can copy or delete any selected text. MPLAB Editor's built-in find and replace feature allows you to search for and replace text.

11.1.2 Indenting Text and Removing Indentation

You can change the indentation level of one or more lines of text.

You can specify whether new lines are automatically indented.

11.1.3 Delimiting Source Code Blocks

You can find matching brackets, such as braces and parentheses, which often delimit sections of text or program sources.

11.1.4 Undoing Edit Actions

You can reverse edit actions with the Undo command, and do them over again with the Redo command.

11.2 CONFIGURING THE EDITOR

The editor may be configured using two dialogs:

- Editor Options Dialog
- Editor Color Customization Dialog

11.2.1 Editor Options Dialog

You can set editor properties in the Editor Options dialog. Select *Edit>Properties* or right-click in a file (editor) window and selecting Properties.

- Editor Tab
- Text Tab
- Sizes Tab

11.2.1.1 EDITOR TAB

Set up editor properties using the Editor tab on the Editor Options dialog as shown in Table 11-1.

TABLE 11-1: EDITOR OPTIONS DIALOG

Option	Description
Line Numbers	When checked, line numbers are visible in the display.
Print Line Numbers	When checked, line numbers are printed out.
Enable Color Printing	When checked, print-out will be in color. This is useful for preserving syntax information.
Word Wrap	When checked, word wrap is enabled.
Auto Indent	When checked, a new line is automatically indented by the same number of spaces and/or tabs as the line above it.
Protect Read Only Files	When checked, read-only files cannot be edited. When unchecked, you can edit the file and save the changes to another file.
Enable Variable Mouseover Values	When checked, hovering mouse over a variable will pop up its value.
Show Address in Mouseover	When checked, prepends the address to the mouseover, e.g., 'Addr = 0x0003 Name = 0x00'
Double-click Toggles Breakpoint	When checked, double-clicking on a line will alternately enable/disable the breakpoint. When unchecked, double-clicking selects the text under the caret.
Find Wrap MessageBox	When checked, a message box is displayed when the find wraps top or bottom, in addition to the current message of "Find Wrapped" in the status bar position panel.
Use Tabbed Window	When checked, displays tabs within a single window, instead of separate windows for each file.

11.2.1.2 TEXT TAB

Set up text properties using the Text tab on the Editor Options dialog.

11.2.1.2.1 Fonts

- **Select Font** – Click to select the editor font.

For Unicode-enabled operating systems only: In the **National Language Code Page:** field, select the value to be used to translate from Unicode text to ANSI.

11.2.1.2.2 Colors

- **Choose Colors** – Click to select the context-sensitive colors in the Editor Color Customization Dialog. Determine the context under Text Mode.
- **Default Colors** – Click to revert to default context-sensitive colors.
- **User Defined Color File** – Color may also be applied to keywords you define in a file. Create the file as specified below, then enter or browse to the file to select it in this tab of the dialog. Click on **Choose Colors** to assign colors to your selected file keywords.
  - The format of the file should be one keyword per line with a carriage return after each keyword.
  - Keywords in the file must be alphanumeric with an alpha character as the first character. Keywords that use other characters will be ignored.
  - Keywords should be unique. If you choose an already defined directive or reserved keyword, then the color displayed is indeterminate.

11.2.1.3 SIZES TAB

Set up sizing properties using the Sizes tab on the Editor Options dialog.

Option	Description
Tab Size	Enter a value between 1 and 16 to specify the width of tabs. **Insert spaces** – Select this to insert spaces up to the next tab position when you type a tab into text. **Keep tabs** – Select this to insert a hard tab character when you type a tab into text.
Gutter Size	Enter a value between 3 and 12 to specify the gutter width on windows that have gutters.
Default Window Size	Enter the size you would like the editor window to use as a default, as the number of characters displayed horizontally.

11.2.2 Editor Color Customization Dialog

You can set color options to be used by the editor in the Editor Options dialog, **Text** tab. The colors will be used for all files which will be opened subsequently having the same syntax type as the currently active source file.

The color change dialog contains a list of specific symbol types which have their own coloring. The meaning of each kind of symbol type depends on the kind of file being colored. As each symbol type is selected, the “Foreground Color” and “Background Color” buttons show the current colors for that type. To change a color, click on the button and the Colors dialog will come up to allow you to design a new color. You may also specify that a symbol type appear in bold and/or italic face by checking the appropriate box. If you want to see the effect of your current changes without closing the dialog, click **Apply**. Otherwise your changes will take effect when you click **OK**, which closes the dialog.

Syntax coloring identifies text in each of the following categories. The syntax type of the source file determines which category a piece of text belong in.

- ALL Settings – global color setting for all categories
- Whitespace – “invisible” characters such as spaces and tabs
- Special Symbols – punctuation and operators
- Floating-Point Numbers
- Implicit-Base Numbers – numbers in the default number base for the file
- Binary Integers
- Octal Integers

- Decimal Integers
- Hexadecimal Integers
- String Constants
- Character Constants
- Preprocessor Directives
- Trigraphs and Digraphs – special key combinations used to represent Special Symbols which do not appear on the keyboard (obsolescent C feature)
- Comments
- Labels
- Reserved Words
- Bad – invalid characters
- Editor Window – text not belonging in any of the above categories
- User Defined File – text defined by the user file specified on Text tab

11.3 WORKING WITH FILES

The following editor features are available for working with files:

- Creating a New File
- Opening Files
- Printing Files
- Saving Files
- Closing Files
- Syntax Type
- Using Bookmarks

11.3.1 Creating a New File

To create a new file:

1. Click the New File icon, select New from the File menu or press <CTRL> + <N>.
2. A new window will open named "Untitled". This is your new file. To give it the name you want it to have, select *File>SaveAs*.

11.3.2 Opening Files

To open an existing file:

1. There are two ways to select an existing file to open.
   - Click the Open File icon, select Open from the File menu or press <CTRL> + <O>. The Select Input File dialog opens. In the dialog, browse to the location of the file you want to open and select it. Click the Open button.
   - Select the file from the Project Tree by double clicking on it.
2. The selected file is displayed in its own editor window. If the selected file is already open, its current editor window will become the active window.

11.3.3 Printing Files

To print an open file:

1. Make sure the window that contains the file you want to print is the active window.
2. Click the Print icon, select Print from the File menu or press <CTRL> + <P>. The Print dialog box is displayed.
3. Select the print options you want and click **OK**. The file is printed on the selected printer.

11.3.4 Saving Files

To save a file:

1. Make sure the window that contains the file you want to save is the active window.
2. Click the Save icon, select Save from the File menu or press <CTRL> + <S>. The file is saved with the name on its window.

To save a file with a different name:

1. Make sure the window that contains the file you want to save is the active window.
2. From the File menu, select Save As. The New File Name dialog displays.
3. In the dialog, browse to the folder where you want to save the file.
4. In the File Name field, modify the name of the file if you want to save the file with a different name.
5. Click **Save**.

11.3.5 Closing Files

There are several ways of closing a file, as shown below:

- From the File Menu:
  - Make sure the window containing the file you want to close is the active window.
  - From the File menu, select Close. If the file has changed since it was saved last, you will be prompted to save your changes.
- From the System Button on the file window, select Close.
- Click the Close Button on the file window.
- Type <CTRL> + <F4> when the file you want to close is the active window.

11.3.6 Syntax Type

For the purposes of syntax coloring, the MPLAB Editor recognizes the syntax of various assemblers and compilers. You may specify the language type a source file contains.

1. Right-click in the file for which you wish to set the syntax type.
2. Select Text Mode from the right-mouse button menu in an editor window. You will get a menu containing the syntax types which the editor currently recognizes.
   - Disabled – No text formatting.
   - PIC16C5x Asm – 12-Bit core MCU devices
   - PIC16Cxxx Asm – 14-Bit core MCU devices
   - PIC17Cxxx Asm – 16-Bit core MCU devices
   - PIC18Cxxx Asm – Enhanced 16-Bit core MCU devices
   - dsPIC Asm – dsPIC30F DSC devices
   - Asm Listing – Assembler listing
   - C – C language listing
   - Basic – Basic listing
   - SCL – Simulator Control Language listing
3. Select the closest matching language type for your file.

11.3.7 Using Bookmarks

From the context (right-mouse) menu, select Bookmarks. Use bookmarks to easily locate specific lines in a file.

- Toggle Bookmark – Sets/removes an unnamed bookmark at the currently selected line.
- Next/Previous – Go to the next/previous bookmark.
- Clear All – Clear all bookmarks.

11.4 WORKING WITH TEXT

The following editor features are available for working with text:

- Selecting Text
- Moving to a Specific Line
- Cutting/Deleting Text
- Copying Text
- Pasting Text
- Finding Text
- Replacing Text
- Matching Braces
- Undoing Editor Actions
- Redoing Editor Actions
- Indent and Outdent
- Formatting Text

11.4.1 Selecting Text

To select any block of characters:

- With the mouse:
  - Click the mouse at one end of the text you wish to select.
  - Hold down the left mouse button and move the mouse cursor to the other end of the text.
  - Release the left mouse button. The text you selected is now highlighted.
- With the keyboard:
  - Using the navigation keys, move the text cursor to one end of the text you wish to select.
  - Hold down a shift key and move the text cursor to the other end of the text.
  - Release the shift key. The text you selected is now highlighted.

To select a whole word or line:

Note:	This function is available if you do not have "Double-click Toggles Breakpoint" checked in the Editor Options.

- Word: Double-click the word you want to select. The word is now highlighted.
- Line: Double-click a word in the line you want to select. Then click again. The line is now highlighted.

To select the entire contents of the file:

Press <CTRL> + <A> or select Select All from the Edit menu. The entire buffer is now highlighted.

Note:	If you move the cursor after selecting text, the text will no longer be selected.

11.4.2 Moving to a Specific Line

No matter where the cursor is in a file, you can quickly move it to any line that you specify.

To move to a specific location in the active window:

1. Select Goto Line from the Edit menu or press <CTRL> + <G>. A Goto Line dialog displays.
2. Enter the line number in the dialog and click the **OK** button. The editor will move the cursor to the specified line and scroll to display the line, if necessary.

Note:	If you enter a number larger than the number of lines in the file, the cursor will move to the last line in the file.

11.4.3 Cutting/Deleting Text

To cut text:

1. Select the text to cut.
2. Click the Cut icon, select Cut from the Edit menu, press <CTRL> + <X> or press <Shift> + <Delete>.

The selected text is deleted from the source document but moved to the Windows clipboard so that it may be pasted elsewhere.

To delete text:

1. Select the text to delete.
2. Select Select from the Edit menu or press <Delete>.

The selected text is deleted from the source document. If you deleted in error, immediately perform an Undo.

To remove single characters:

1. To remove the character to the left of the caret, press <Backspace>.
2. To delete the character under the caret, press <Delete>.

11.4.4 Copying Text

To copy text:

1. Select the text you want to copy.
2. Click the Copy icon, select Copy from the Edit menu, press <CTRL> + <C> or press <CTRL> + <Insert>.

To copy a column of text:

1. Place the cursor at a corner of the text column you want to copy.
2. Hold down the Alt key and then click-and-drag with the mouse to select the text column.
3. Click the Copy icon, select Copy from the Edit menu, right-click on the selection and choose Copy, press <CTRL> + <C> or press <CTRL> + <Insert>.

Note:	If the wrong column is highlighted, make sure your cursor is on the same line as your mouse pointer for proper selection.

The selected text is copied to the Windows clipboard so it can be pasted elsewhere.

11.4.5 Pasting Text

You can paste any text contained in the Windows clipboard into a file in the MPLAB editor. You can paste text that you have copied from the same file or from another application. Text must exist in the Windows clipboard before it can be pasted.

To paste text from the Windows clipboard:

1. Move the caret to the point where you want to insert the text.
2. Click the Paste icon, select Paste from the Edit menu, press <CTRL> + <V> or press <Shift> + <Insert>.

Note:	Pasting text does not remove it from the clipboard; you may paste it multiple times.

11.4.6 Finding Text

To find text in a file:

1. Make sure the file you want to search is open in the active window.
2. Select Find from the Edit menu or press <CTRL> + <F>. The Find dialog displays.
3. If you selected text before opening the Find dialog, that text is displayed in the "Find What" field. If the text you want to search for is not displayed in the "Find What" field, enter the text you want to find or select a previously found text string from the drop-down list.
4. Select any of the Find options in the dialog that you want to use for this search. These options are:
   - Match whole word only
   - Match case
   - Direction – Up
   - Direction – Down
5. Click **Find Next**. The editor will move the caret to the next (previous) matching text in the file. If there is no such text, the editor will display a message saying so.

To repeat the previous find:

Press <F3>, or, if the Find dialog is still up, click **Find Next**. To repeat the previous find in the opposite direction, press <Shift> + <F3>.

11.4.7 Replacing Text

To find and replace text in a file:

1. Make sure the file you want to edit is open in the active window.
2. From the Edit menu, select Replace or press <CTRL> + <H>.
3. If you selected text in the file before opening the Replace dialog, that text is displayed in the “Find What” field. If the text you want to replace is not displayed in the “Find What” field, enter the text you want replace.
4. In the “Replace With” field, enter the replacement text.
5. Select any of the Find/Replace options in the dialog that you want to use for this search. These are:
   - Match whole word only
   - Match case
6. Click:
   - **Find Next** to do a simple find
   - **Replace** to replace the closest matching text
   - **Replace All** to replace all matching text
   - Cancel to close the dialog with no further action.

To repeat a Replace operation:

1. Press <F3>.
2. If the Find/Replace dialog is still open, click **Replace**.

The most recently performed Find/Replace operation is executed, including all Find/Replace options that were selected.

11.4.8 Matching Braces

To recognize a section of bracketed text:

1. Move the caret to an opening/closing symbol.
2. Press <CTRL> + <M>, select *Edit>Match* or right-click in the edit window and select *Advanced>Match*. The cursor will move to the matching symbol.

Opening/closing symbols are: [ ] { } ( ) < >

11.4.9 Undoing Editor Actions

If you have just made a change to a file, you can reverse the effect of the last change you just made.

- Select Undo from the Edit menu or press <CTRL> + <Z>.

You can repeat the Undo action multiple times in succession. Each will undo the action prior to the last undo.

11.4.10 Redoing Editor Actions

If you have just reversed a change with the Undo function, you can redo the change.

- Select Redo from the Edit menu or press <CTRL> + <Y>.

You can repeat the Redo action multiple times in succession.

11.4.11 Indent and Outdent

You may indent or outdent text by the tab width specified in *Edit>Properties>Editor Options*, **Tabs** tab. Then use <Tab>/<Shift>+<Tab> or Indent Block/Outdent Block commands (Right-click menu, Advanced.)

11.4.11.1 INDENTING

Tabs or spaces will be inserted as specified in the Editor Options dialog.

Single line

1. Move the caret to the start of the line you want to indent.
2. Make sure the editor is in Insert mode
3. Press <Tab> once for each level you want to indent the line.

Auto Indent

New line automatically indented like the previous line.

Block of lines

1. Select the lines in the block you want to indent.
2. Press <Tab> once for each level you want to indent the lines in the block.

11.4.11.2 OUTDENTING

Single line

1. Move the caret to the start of the line you want to outdent.
2. Press <Shift> + <Tab> once for each level you want to outdent the line.

Block of lines

1. Select the lines in the block you want to outdent.
2. Press <Shift>+<Tab> key once for each level you want to outdent the lines in the block.

Note:	You cannot outdent a line backwards through column 1.

11.4.12 Formatting Text

You can format text from options selected in the context (right-mouse) menu.

- Right-click menu, Text Mode. Format all text according to the syntax of your selected assembler or compiler and device.
- *Advanced>Format Text*. Make all selected text either all uppercase or all lowercase.
- Properties, Editor Options dialog. Set up the file (editor) window.

11.5 WORKING WITH DEBUG FEATURES

When a debugger is selected in MPLAB IDE, several code debugging features are available in the editor window.

- Filter Trace
- Breakpoints

11.5.1 Filter Trace

You may set up a simple filter trace in the editor window. However, the debug tool trace feature must not be enabled to do this. Also, Filter-in and Filter-out trace are mutually exclusive, i.e., setting one will clear the other.

Filter-In Trace

You may select (filter) the code that is traced into the trace buffer (for display in a Trace window) by:

- selecting the code to add to the trace in the editor window
- right-clicking in the window and selecting “Add Filter-in Trace” from the menu

Filter-Out Trace

You may select (filter) the code that is to be excluded from the trace buffer and thereby trace only the unselected code into the trace buffer by:

- selecting the code to exclude from the trace in the editor window
- right-clicking in the window and selecting “Add Filter-out Trace” from the menu

Removing Traces

To remove the code from the trace, right-click in the filter code and select “Remove Filter Trace”.

To remove multiple sections that have be used for trace, right-click in the window and select “Remove All Filter Traces”.

11.5.2 Breakpoints

Breakpoints stop executing code at a designated line so that you may observe register, bit and variable values for debugging.

Breakpoints are a standard feature of MPLAB IDE Debug tools. For more information, see **Section 3.18 “Using Breakpoints”**.

11.6 KEYBOARD FEATURES

The following keys are specified for the editor:

- Shortcuts
- Movement and Selection
- Special Characters

11.6.1 Shortcuts

The following keyboard shortcuts are available for the editor. To change shortcut key assignments, use *Configure>Settings*, **Hot Keys** tab.

11.6.1.1 EDITING

Keystroke	Result
CTRL + Z	Undo last edit
CTRL + Y	Redo last edit
CTRL + X, Shift + Delete	Cut selected text
CTRL + C, CTRL + Insert	Copy selected text
CTRL + V, Shift + Insert	Paste text
Delete	Delete selected text
CTRL + A	Select all text in the window
CTRL + F	Find
CTRL + H	Find/Replace
F3	Repeat find/replace down
SHIFT + F3	Repeat find/replace up
CTRL + G	Goto line
CTRL + M	Match brace

11.6.1.2 FILE MANAGEMENT

Keystroke	Result
CTRL + N	Create new text file
CTRL + O	Open existing file
CTRL + F4	Close existing file
CTRL + S	Save active file
CTRL + P	Print active file

11.6.1.3 NAVIGATION

Keystroke	Result
CTRL + F6	Next open window
SHIFT + CTRL + F6	Previous open window
CTRL + K	Toggle Bookmark
CTRL + L	Next Bookmark
F1	Help

11.6.2 Movement and Selection

The keyboard keystrokes shown in Table 11-2 may be used when editing text.

TABLE 11-2: KEYSTROKES FOR EDITING TEXT

Keystroke	Result
CTRL + A	Selects all the text in the file
UP	Moves the cursor up by one line
SHIFT + UP	Moves the cursor up by one line, extending the selection
DOWN	Moves the cursor down by one line
SHIFT + DOWN	Moves the cursor down by one line, extending the selection
LEFT	Moves the cursor left by one character
SHIFT + LEFT	Moves the cursor left by one character, extending the selection
CTRL + LEFT	Moves the cursor left by one word
CTRL + SHIFT + LEFT	Moves the cursor left by one word, extending the selection
RIGHT	Moves the cursor right by one character
SHIFT + RIGHT	Moves the cursor right by one character, extending the selection
CTRL + RIGHT	Moves the cursor right by one word
CTRL + SHIFT + RIGHT	Moves the cursor right by one word, extending the selection
PGDN	Moves the cursor down by one page
SHIFT + PGDN	Moves the cursor down by one page, extending the selection
PGUP	Moves the cursor up by one page
SHIFT + PGUP	Moves the cursor up by one page, extending the selection
HOME	Moves the cursor to the start of the line
SHIFT + HOME	Moves the cursor to the start of the line, extending the selection
CTRL + HOME	Moves the cursor to the start of the file
CTRL + SHIFT + HOME	Moves the cursor to the start of the file, extending the selection
END	Moves the cursor to the end of the line
SHIFT + END	Moves the cursor to the end of the line, extending the selection
CTRL + END	Moves the cursor to the end of the file
CTRL + SHIFT + END	Moves the cursor to the end of the file, extending the selection

11.6.3 Special Characters

To enter any single character:

1. Put the keyboard keypad into numeric mode, either with <NumLock> or by holding down <Shift>.
2. Hold down <Alt>.
3. Type the decimal number of the character you want (between 0 and 256.)
4. Release <ALT>.

If the character you typed has a visible form, it will appear at the caret in your file window.

Insert/Overtype Modes:

Use the Insert key to toggle between INS and OVR mode, shown on the status bar.

Key(s)	Insert Mode	Overtype Mode
Enter	Insert NewLine	Move to first character of next line, if any
TAB or CTRL + I	Insert Tab	Move to next tab position, if any

11.7 EDITOR TROUBLESHOOTING

This section is designed to help you troubleshoot any problems, errors or issues you encounter while using MPLAB Editor.

11.7.1 Common Problems/FAQ

Some items in the context menu are disabled.

Depending on your selection of debug and/or programmer tools, some items may not be applicable and so are grayed.

Nothing happens when I press <CTRL> + <M>.

The text cursor must be on an opening/closing symbol when you press <CTRL> + <M>. When it is, Match will move it to the matching symbol. If there is no match, the cursor does not move.

Opening/closing symbols are: [ ] { } ( ) < >

When I insert a carriage return, the new line always starts in column 1, regardless of the current indentation level.

Auto Indent must be checked in the Tab Settings dialog for auto indent to work.

11.7.2 Save Changes to File?

You have changed this file since the last time you saved it to disk. Before MPLAB Editor closes it, you have a chance to update the disk copy.

Command	Result
Yes	Saves the contents of the window to disk. Prompts to name the file if it is unnamed.
No	Closes the file and discards the changes made since the last save.
Cancel	Abandons the close and resumes editing the window.

Part 4 – MPLAB SIM

Chapter 12. Simulator Overview 167
Chapter 13. Getting Started with MPLAB SIM 181
Chapter 14. Using Stimulus 187
Chapter 15. Using Stimulus – PIC17 Devices 197
Chapter 16. Simulator Troubleshooting 205
Chapter 17. Simulator Reference 207

NOTES:

Chapter 12. Simulator Overview

12.1 INTRODUCTION

MPLAB SIM is a discrete-event simulator for:

- PICmicro microcontroller (MCU) families
- dsPIC digital signal controller (DSC) families

The simulator is integrated into MPLAB IDE integrated development environment. The MPLAB SIM debugging tool is designed to model operation of Microchip Technology's MCU and DSC devices to assist in debugging software for these devices.

If you are new to MPLAB SIM, you may want to begin with the MPLAB IDE tutorial.

- Simulator Features
- Simulator Model – PICmicro MCUs
- Simulator Model – dsPIC DSCs
- Simulator Execution

12.2 SIMULATOR FEATURES

MPLAB SIM allows you to:

- Modify object code and immediately re-execute it
- Inject external stimuli to the simulated processor
- Set pin and register values at prespecified intervals
- Trace the execution of the object code (MCU's only)

12.3 SIMULATOR MODEL – PICmicro MCUs

PICmicro MCU's use Harvard architecture (separate program and data memory) as opposed to Von Neumann architecture (combined program and data memory.) Therefore, program memory instructions are not limited to 8-bit data lengths. Program memory, or core, instruction length is used to group PICmicro MCU's.

At the time this documentation was produced, the following cores are associated with the listed devices:

- 12-Bit Core Device Model – PIC12C5XX, PIC12CE5XX, PIC16X5X, PIC16C505
- 14-Bit Core Device Model – PIC12C67X, PIC12CE67X, PIC12F629/675, PIC16
- 16-Bit Core Device Model – PIC17
- 16-Bit Core Device Model – PIC18

The 12-bit, 14-bit and 16-bit (PIC17) core devices have word-addressed program memory space.

The 16-bit (PIC18) core devices have a byte-organized program memory space. There are some restrictions in the silicon on how the program memory space can be accessed, especially when using long writes to program memory. The simulator may not show the same restrictions in all situations. Consult the data sheet for proper operation.

12.3.1 12-Bit Core Device Model – PIC12C5XX, PIC12CE5XX, PIC16X5X, PIC16C505

The following topics discuss the 12-bit core device features modeled in the simulator.

- 12-Bit Core I/O Pins
- 12-Bit Core CPU
- 12-Bit Core Peripherals

12.3.1.1 12-BIT CORE I/O PINS

The 12-bit core devices have I/O pins multiplexed with other peripherals (and therefore referred by more than one name). MPLAB SIM recognizes only the following pin names as valid I/O pins:

Note: Pins are only available as described in the data sheet of the specific device.

- MCLR
- T0CKI
- GP0-GP5
- RA0-RA3
- RB0-RB7
- RC0-RC7
- RD0-RD7
- RE4-RE7

12.3.1.2 12-BIT CORE CPU

Reset Conditions

All reset conditions are supported by the MPLAB SIM simulator.

The Time-out (TO) and Power-down (PD) bits in the STATUS register reflect appropriate reset condition. This feature is useful for simulating various power-up and time-out forks in your code.

A MCLR reset during normal operation can easily be simulated by driving the MCLR pin low (and then high) via stimulus or by selecting *Debugger>Reset>MCLR Reset*.

Watchdog Timer

The Watchdog Timer is fully simulated in the MPLAB SIM simulator.

The period of the WDT is determined by the pre/postscaler settings in the OPTION_REG register. The basic period (with pre/postscaler at 1:1) is approximated, to the closest instruction cycle multiple, in the device data sheet.

In the Configuration Bits dialog (*Configuration>Configuration Bits*) you enable/disable the WDT and set the pre/postscaler.

A WDT time-out is simulated when WDT is enabled, proper pre/postscaler is set and WDT actually overflows. On WDT time-out, the simulator will halt or reset, depending on the selection in the Break Options tab of the Settings dialog.

12.3.1.3 12-BIT CORE PERIPHERALS

Along with core support, MPLAB SIM fully supports the TIMER0 timer/counter module in both internal and external clock modes.

It is important to remember that, because MPLAB SIM executes on instruction cycle boundaries, resolutions below 1 TCY cannot be simulated.

12.3.2 14-Bit Core Device Model – PIC12C67X, PIC12CE67X, PIC12F629/675, PIC16

The following topics discuss the 14-bit core device features modeled in the simulator.

- 14-Bit Core I/O Pins
- 14-Bit Core Interrupts
- 14-Bit Core CPU
- 14-Bit Core Peripherals

12.3.2.1 14-BIT CORE I/O PINS

The 14-bit core devices have I/O pins multiplexed with other peripherals (and therefore referred by more than one name). MPLAB SIM recognizes only the following pin names as valid I/O pins:

Note:	Pins are only available as described in the data sheet of the specific device.

- MCLR
- GP0-GP5
- RA0-RA5
- RB0-RB7
- RC0-RC7
- RD0-RD7
- RE0-RE7

12.3.2.2 14-BIT CORE INTERRUPTS

The following interrupts are supported:

- Timer0 overflow
- Timer1 overflow
- Timer2
- CCP1
- CCP2
- Change on Port RB <7:4>
- External interrupt from RB0/INT pin
- Comparators
- A/D complete
- EEPROM write complete

12.3.2.3 14-BIT CORE CPU

Reset Conditions

All reset conditions are supported by the MPLAB SIM simulator.

The Time-out (TO) and Power-down (PD) bits in the STATUS register reflect appropriate reset condition. This feature is useful for simulating various power-up and time-out forks in the user code.

A MCLR reset during normal operation or during Sleep can easily be simulated by driving the MCLR pin low (and then high) via stimulus or by selecting *Debugger> Reset>MCLR Reset*.

Sleep

When executing a Sleep instruction, MPLAB SIM will appear "asleep" until a wake-up from sleep condition occurs. For example, if the Watchdog Timer has been enabled, it will wake the processor up from sleep when it times out (depending upon the pre/postscaler setting).

An example of a wake-up-from-sleep condition would be Timer1 wake up from sleep. In this case, when the processor is asleep, Timer1 would continue to increment until it overflows. If the interrupt is enabled, the timer will wake the processor on overflow and branch to the interrupt vector.

Watchdog Timer

The Watchdog Timer is fully simulated in the MPLAB SIM simulator.

The period of the WDT is determined by the pre/postscaler settings in the OPTION_REG register. The basic period (with pre/postscaler at 1:1) is approximated, to the closest instruction cycle multiple, in the device data sheet.

In the Configuration Bits dialog (*Configuration>Configuration Bits*) you enable/disable the WDT and set the pre/postscaler.

A WDT time-out is simulated when WDT is enabled, proper pre/postscaler is set and WDT actually overflows. On WDT time-out, the simulator will halt or reset, depending on the selection in the Break Options tab of the Settings dialog.

12.3.2.4 14-BIT CORE PERIPHERALS

Along with core support, MPLAB SIM supports the following peripheral modules, in addition to general purpose I/O. Consult the data sheet for the particular device you are using for information on which symbols are implemented.

Note:	Even if peripheral functionality is not supported in the simulator, the peripheral registers will be available as read/write.

- Timers
- CCP/ECCP
- Comparators (Limited)
- A/D Converter (Limited)
- USART
- EEPROM Data Memory

The delays are implemented on all peripherals, but the interrupt latency is not.

Timers

Timer0 (and the interrupt it can generate on overflow) is fully supported, and will increment by the internal or external clock. Clock input must have a minimum high time of 1 TCY and a minimum low time of 1 TCY due to stimulus requirements.

Timer1 in its various modes is supported, except when running in counter mode by an external crystal. MPLAB SIM supports Timer1 interrupts generated on overflow, and interrupts generated by wake-up from sleep. The external oscillator on RC0/RC1 is not simulated, but a clock stimulus can be assigned to those pins.

Timer2 and the interrupt that can be generated on overflow are fully supported.

CCP/ECCP

Capture

MPLAB SIM fully supports capture and the interrupt generated.

Compare

MPLAB SIM supports compare mode, its interrupt and the special event trigger (resetting Timer1 by CCP1).

PWM

PWM output is supported (resolution greater than 1 TCY only).

Comparators (Limited)

Only comparator modes that do not use Vref are simulated in MPLAB SIM.

A/D Converter (Limited)

All the registers, timing function and interrupt generation are implemented. The simulator, however, does not load any meaningful value into A/D result register (ADRES) at the end of a conversion.

Note:	If you have trouble with I/O pins on processors that have A/D (PIC16C74, PIC16F877, etc.), make certain that the ADCON registers are configuring those pins for digital I/O rather than for analog input. For most processors, these default to analog inputs and the associated pins cannot be used for I/O until the ADCON (or ADCON1) register is set properly.

USART

USART functionality is supported.

EEPROM Data Memory

The EEPROM data memory is fully simulated. The registers and the read/write cycles are fully implemented. The write cycle time is approximated to 10 ms (to nearest instruction cycle multiple).

The simulator simulates the functions of WRERR and WREN control bits in the EECON1 register.

12.3.3 16-Bit Core Device Model – PIC17

The following topics discuss the 16-bit core device features modeled in the simulator.

- 16-Bit Core (PIC17) I/O Pins
- 16-Bit Core (PIC17) Interrupts
- 16-Bit Core (PIC17) CPU
- 16-Bit Core (PIC17) Processor Modes
- 16-Bit Core (PIC17) Peripherals

12.3.3.1 16-BIT CORE (PIC17) I/O PINS

PIC17 devices have I/O pins multiplexed with other peripherals and therefore referred by more than one name. MPLAB SIM recognizes only the following pin names as valid I/O pins:

Note: Pins are only available as described in the data sheet of the specific device.

- MCLR
- RA0-RA5
- RB0-RB7
- RC0-RC7
- RD0-RD7
- RE0-RE2

12.3.3.2 16-BIT CORE (PIC17) INTERRUPTS

The following interrupts are supported:

- External interrupt on INT pin
- TMR0 overflow interrupt
- External interrupt on RA0 pin
- Port B input change interrupt
- Timer/Counter1 interrupt
- Timer/Counter2 interrupt
- Timer/Counter3 interrupt
- Capture1 interrupt
- Capture2 Interrupt

12.3.3.3 16-BIT CORE (PIC17) CPU

Reset Conditions

All reset conditions are supported by the MPLAB SIM simulator.

The Time out (TO) and Power-Down (PD) bits in the CPUSTA register reflect appropriate reset condition. This feature is useful for simulating various power-up and time-out forks in the user code.

A MCLR reset during normal operation or during Sleep can easily be simulated by driving the MCLR pin low (and then high) via stimulus.

Sleep

When executing a Sleep instruction, MPLAB SIM will appear "asleep" until a wake-up from sleep condition occurs. For example, if the Watchdog Timer has been enabled, it will wake the processor up from sleep when it times out (depending upon the postscaler setting).

An example of a wake-up-from-sleep condition would be an input change on Port B. If the interrupt is enabled and the GLINTD bit is set, the processor will wake up and will resume executing from the instruction following the Sleep command. If the GLINTD = 0, the normal interrupt response will take place.

Watchdog Timer

The Watchdog Timer is fully simulated in the MPLAB SIM simulator.

The period of the WDT is determined by the postscaler configuration bits WDTPS0:1. The basic period (with postscaler at 1:1) is approximated, to the closest instruction cycle multiple, in the device data sheet. Setting the configuration bits WDTPS0:1 to 00 will disable the WDT.

In the Configuration Bits dialog (*Configuration>Configuration Bits*) you enable/disable the WDT and set the postscaler.

A WDT time-out is simulated when WDT is enabled, proper postscaler is set and WDT actually overflows. On WDT time-out, the simulator will halt or reset, depending on the selection in the Break Options tab of the Settings dialog.

12.3.3.4 16-BIT CORE (PIC17) PROCESSOR MODES

The following processor modes are supported by MPLAB SIM for devices which allow them:

- Microcontroller (Default)
- Microprocessor
- Microprocessor w/Boot
- Extended Microcontroller

Set the processor mode in code (`__config`) or the Configuration Bits dialog.

12.3.3.5 16-BIT CORE (PIC17) PERIPHERALS

Along with providing core support, MPLAB SIM supports the following peripheral modules, in addition to general purpose I/O:

- Timer0
- Timer1 and Timer2
- Timer3 and Capture
- PWM

The delays are implemented on all peripherals, but the interrupt latency is not.

Timer0

Timer0 and the interrupt it can generate on overflow is fully supported, and will increment by the internal or external clock. Delay from external clock edge to timer increment has also been simulated, as well as the interrupt latency period. Clock input must have a minimum high time of 1 TCY and a minimum low time of 1 TCY due to the stimulus file requirements.

Timer1 and Timer2

Timer1 and Timer2 in its various modes is fully supported. Delays from clock edge to increment (when configured to increment from rising or falling edge of external clock) is simulated as well as the interrupt latency periods. Clock input must have a minimum high time of 1 TCY and a minimum low time of 1 TCY due to the stimulus file requirements.

Timer3 and Capture

The MPLAB simulator fully supports Timer3 and the Capture module in all of its modes. Delays from clock edge to increment (when configured in external mode), delay for capture and interrupt latency periods are fully supported. Clock input must have a minimum high time of 1 TCY and a minimum low time of 1 TCY due to the stimulus file requirements.

PWM

Both PWM outputs are supported (resolution greater than 1 TCY only).

12.3.4 16-Bit Core Device Model – PIC18

The following topics discuss the enhanced 16-bit core device features modeled in the simulator.

- 16-Bit Core (PIC18) I/O Pins
- 16-Bit Core (PIC18) Interrupts
- 16-Bit Core (PIC18) CPU
- 16-Bit Core (PIC18) Peripherals

12.3.4.1 16-BIT CORE (PIC18) I/O PINS

PIC18 devices have I/O pins multiplexed with other peripherals (and therefore referred by more than one name). MPLAB SIM recognizes only the following pin names as valid I/O pins:

Note: Pins are only available as described in the data sheet of the specific device.

- MCLR
- RA0-RA5
- RB0-RB7
- RC0-RC7
- RD0-RD7
- RE0-RE2

12.3.4.2 16-BIT CORE (PIC18) INTERRUPTS

The following interrupts are supported:

- External interrupt on INT pin
- TMR0 overflow interrupt
- External interrupt on RA0 pin
- Port B input change interrupt
- Timer/Counter1 interrupt
- Timer/Counter2 interrupt
- Timer/Counter3 interrupt
- Capture1 interrupt
- Capture2 Interrupt

12.3.4.3 16-BIT CORE (PIC18) CPU

Reset Conditions

All reset conditions are supported by the MPLAB SIM simulator.

The Time out (TO) and Power-Down (PD) bits in the RCON register reflect appropriate reset condition. This feature is useful for simulating various power-up and time-out forks in the user code.

You cannot reset by toggling MCLR using stimulus control. You must use *Debugger> Reset>MCLR Reset*.

Sleep

When executing a Sleep instruction, MPLAB SIM will appear “asleep” until a wake-up from sleep condition occurs. For example, if the Watchdog Timer has been enabled, it will wake the processor up from sleep when it times out (depending upon the pre/postscaler setting).

An example of a wake-up-from-sleep condition would be an input change on Port B. If the interrupt is enabled and the GLINTD bit is set, the processor will wake up and will resume executing from the instruction following the Sleep command. If the GLINTD = 0, the normal interrupt response will take place.

Watchdog Timer

The Watchdog Timer is fully simulated in the MPLAB SIM simulator.

The period of the WDT is determined by the pre/postscaler configuration bits WDTPS0:2. The basic period (with pre/postscaler at 1:1) is approximated, to the closest instruction cycle multiple, in the device data sheet.

Setting the configuration bit WDTEN to 0 will disable the WDT, unless it is enabled by the SWDTEN bit of the WDTCON register. Setting the configuration bit WDTEN to 1 will enable the WDT regardless of the value of the SWDTEN bit.

In the Configuration Bits dialog (*Configuration>Configuration Bits*) you enable/disable the WDT and set the pre/postscaler.

A WDT time-out is simulated when WDT is enabled, proper pre/postscaler is set and WDT actually overflows. On WDT time-out, the simulator will halt or reset, depending on the selection in the Break Options tab of the Settings dialog.

12.3.4.4 16-BIT CORE (PIC18) PERIPHERALS

Along with core support, MPLAB SIM supports the following peripheral modules, in addition to general-purpose I/O:

- Timers
- CCP/ECCP
- Comparators (Limited)
- A/D Converter (Limited)
- USART
- EEPROM Data Memory

The delays are implemented on all peripherals, but the interrupt latency is not.

Timers

Timer0 (and the interrupt it can generate on overflow) is fully supported, and will increment by the internal or external clock. Clock input must have a minimum high time of 1 TCY and a minimum low time of 1 TCY due to stimulus requirements.

All Other Timers in their various modes are supported, except for modes using an external crystal. MPLAB SIM supports Timer interrupts generated on overflow, and interrupts generated by wake-up from sleep. Although the external oscillator is not simulated, a clock stimulus can be assigned to those pins.

CCP/ECCP

Capture

MPLAB SIM fully supports capture and the interrupt generated.

Compare

MPLAB SIM supports compare mode, its interrupt and the special event trigger (resetting a Timer by CCP).

PWM

PWM output is supported (resolution greater than 1 TCY only).

Comparators (Limited)

Only comparator modes that do not use Vref are simulated in MPLAB SIM.

A/D Converter (Limited)

All the registers, timing function and interrupt generation are implemented. The simulator, however, does not load any meaningful value into A/D result register (ADRES) at the end of a conversion.

Note:	If you have trouble with I/O pins on processors that have A/D (PIC16C74, PIC16F877, etc.), make certain that the ADCON registers are configuring those pins for digital I/O rather than for analog input. For most processors, these default to analog inputs and the associated pins cannot be used for I/O until the ADCON (or ADCON1) register is set properly.

USART

USART functionality is supported.

EEPROM Data Memory

The EEPROM data memory is fully simulated. The registers and the read/write cycles are fully implemented. The write cycle time is approximated to 10 ms (to nearest instruction cycle multiple).

The simulator simulates the functions of WRERR and WREN control bits in the EECON1 register.

12.4 SIMULATOR MODEL – dsPIC DSCs

The dsPIC digital signal controller (DSC) is a combination of a digital signal processing (DSP) core and PICmicro microcontroller (MCU) peripheral features. DSCs use a modified Harvard architecture to provide separate program memory and 16-bit data memory spaces.

The following topics discuss the dsPIC device features modeled in the simulator.

- I/O Pins
- Exceptions (Traps/Interrupts)
- System Integration Block
- Memory
- Peripherals

12.4.1 I/O Pins

The dsPIC devices have I/O pins multiplexed with other peripherals (and therefore referred by more than one name). The simulator recognizes only the pin names specified in the standard device headers as valid I/O pins. Therefore, you should refer to the header file for your device (`p`*`DeviceNumber`*`.inc`) to determine the correct pin names.

12.4.2 Exceptions (Traps/Interrupts)

The simulator supports all core and peripheral traps and interrupts, even if the peripheral is currently not supported.

The dsPIC core features a vectored exception processing structure for up to 8 traps and 54 interrupts or 62 total independent vectors. Each interrupt is prioritized, based on a user-assigned priority between 1 and 7 (1 being the lowest priority and 7 being the highest). If a conflict occurs (two interrupts at the same priority), interrupt service will be based on a predetermined 'natural order' which is hardware-based.

12.4.3 System Integration Block

Reset Sources

All reset sources are supported by the MPLAB SIM simulator.

Status bits from the RCON register are set or cleared differently in different Reset situations, as indicated in device data sheet. These bits are used in software to determine the nature of the Reset.

A MCLR reset during normal operation or during Sleep/Idle can easily be simulated by driving the MCLR pin low (and then high) via stimulus or by selecting *Debugger> Reset>MCLR Reset*.

Sleep/Idle

When executing a PWRSAV instruction, MPLAB SIM will appear “asleep” or “idle” until a wake-up condition occurs. For example, if the Watchdog Timer has been enabled, it will wake the processor up from sleep when it times out (depending upon the pre/postscaler setting).

An example of a wake-up-from-sleep condition would be Timer1 wake up from sleep. In this case, when the processor is asleep, Timer1 would continue to increment until it matches the period counter. If the interrupt is enabled, the timer will wake the processor and branch to the interrupt vector.

Watchdog Timer

The Watchdog Timer is fully simulated in the MPLAB SIM simulator.

The Watchdog Timer can be "Enabled" or "Disabled" through a configuration bit (FWDTEN) in the Configuration register FWDT. Setting FWDTEN = 1 enables the Watchdog Timer. Setting FWDTEN = 0 allows user software to enable/ disable the Watchdog Timer via the SWDTEN (RCON<5>) control bit.

The period of the WDT is determined by the pre/postscaler settings in the FWDT register. The basic period (with pre/postscaler at 1:1) is approximated, to the closest instruction cycle multiple, in the device data sheet.

In the Configuration Bits dialog (*Configuration>Configuration Bits*) you enable/disable the WDT and set the pre/postscalers.

A WDT time-out is simulated when WDT is enabled, proper pre/postscaler is set and WDT actually overflows. On WDT time-out, the simulator will halt or reset, depending on the selection in the Break Options tab of the Settings dialog.

12.4.4 Memory

dsPIC device memory – program flash and data RAM – is simulated except for features dependent on security aspects. I.e., protected memory is not simulated.

12.4.5 Peripherals

MPLAB SIM supports all peripheral modules, with functional exceptions listed in the limitations.

The delays are implemented on all peripherals, but the interrupt latency is not.

12.5 SIMULATOR EXECUTION

MPLAB SIM operation is specified in the following topics.

- Execution Speed
- Execution on Instruction Cycle Boundaries
- I/O Timing

12.5.1 Execution Speed

When MPLAB SIM is simulating running in real-time, instructions are executing as quickly as the PC's CPU will allow. This is usually slower than the actual device would run at its rated clock speed.

The speed at which the simulator runs depends on the speed of your computer and how many other tasks you have running in the background. The software simulator must update all of the simulated registers and RAM, as well as monitor I/O, set and clear flags, check for break and trace points in software and simulate the instruction with instructions being executed on your computer's CPU.

The execution speed of a discrete-event software simulator is orders of magnitude less than a hardware oriented solution. Slower execution speed may be viewed as a handicap or as a tool. The simulator attempts to provide the fastest possible simulation cycle, and depending upon the mode of operation, can operate on the order of milliseconds per instruction.

Note:	Often loops will be used in your code to generate timing delays. When using the simulator, you might wish to decrease these time delays or conditionally remove those sections of your code with “IFDEF” statements to increase simulation speed.

In general, when this discussion says “real-time” and you are in the simulator mode, the software simulation is executing simulated PICmicro MCU code as fast as your PC can simulate the instructions.

12.5.2 Execution on Instruction Cycle Boundaries

The simulator executes on instruction cycle boundaries, and resolutions shorter than one instruction cycle (TCY) can not be simulated. The simulator is a discrete-event simulator where all stimuli are evaluated, and all responses are generated, at instruction boundaries, or TCY = 4*Tosc, where Tosc is the input clock period. Therefore, some physical events can not be accurately simulated. These fall into the following categories:

- Purely asynchronous events
- Events that have periods shorter than one instruction cycle

The net result of instruction boundary simulation is that all events get synchronized at instruction boundaries, and events smaller than one instruction cycle are not recognized.

The following functions and peripherals are affected by simulation on instruction cycle boundaries:

Note:	Not all peripherals listed here are currently supported by the simulator.

- Clock pulse inputs smaller than one cycle cannot be simulated even though timer prescalers are capable of accepting clock pulse inputs smaller than one cycle.
- PWM output pulse resolution less than one cycle is not supported.
- Compares greater than 8 bits are not supported.
- In unsynchronized counter mode, clock inputs smaller than one cycle cannot be used.
- The oscillator waveform on RC0/RC1 pins cannot be shown.
- Serial I/O is not simulated.

12.5.3 I/O Timing

External timing in the simulator is processed only once during each instruction cycle. Transient signals, such as a spikes on MCLR smaller than an instruction cycle, will not be simulated.

Note:	Stimulus is injected into the simulator prior to the next instruction cycle.

NOTES:

Chapter 13. Getting Started with MPLAB SIM

13.1 INTRODUCTION

If you are new to MPLAB IDE and the simulator, please refer to the following topics to help you set up MPLAB IDE for use with MPLAB SIM:

- **Chapter 2. "Getting Started with MPLAB IDE: A Basic Tutorial"**
- **Chapter 3. "Walk-Through and Tutorial"**

Once you are familiar with basic simulator operation under MPLAB IDE, the following topics will help you work with simulator-specific features.

- Using the Stopwatch
- Using Stimulus
- Using Simulator Trace
- Using External Memory

13.2 USING THE STOPWATCH

Reach this dialog from *Debugger>Stopwatch*.

Instruction Cycles and Time

The simulator updates the Instruction Cycles and Time fields, including the Time units, as your program runs.

- Click Synch to synchronize the stopwatch to the total simulated values.
- Click Zero to set the Instruction Cycles and Time values to zero at any time.
- Check "Clear Simulation Time On Reset" to set the Instruction Cycles and Time values to zero whenever the program is reset.

Note: This option should always be checked when using stimulus.

Processor Frequency

This field, including Frequency units, will be set from the Clock tab or Osc/Trace tab of the *Debugger>Settings* dialog.

13.3 USING STIMULUS

Simulator stimulus is a powerful tool that allows you to control inputs to the simulator and exercise your application code. Stimulus control is detailed and covered in separate chapters:

- **Chapter 14. "Using Stimulus"**
- **Chapter 15. "Using Stimulus – PIC17 Devices"**

13.4 USING SIMULATOR TRACE

How to use MPLAB SIM simulator trace is discussed here.

- Tracing Execution
- Trace Window
- Export Trace Buffer

13.4.1 Tracing Execution

Tracing allows you to record the step-by-step execution of your code and examine this recording.

Note:	Simulation speed is reduced when using trace.

- When tracing is enabled in the Trace/Pins tab or Osc/Trace tab of the Settings dialog, the simulator will record the instructions executed into a trace buffer.
- Tracing will stop when execution stops, as by a Halt, Break or Breakpoint.
- The Trace window will display the trace buffer when opened. It will update its contents upon request.

13.4.2 Trace Window

Open the Trace Window from *View>Simulator Trace*.

If the trace buffer is empty, the window will say "No items to display". Otherwise, the window will display the contents of the trace buffer. Traces of instructions as they are executed are written to a buffer of 8192 lines. When this buffer is full, the next trace wraps to the top of the buffer.

For more information on the Trace window and its context menu, see **Section 8.14 "Trace Memory Window"**.

13.4.3 Export Trace Buffer

Select Export from the right mouse menu in the Trace window to display the Export As dialog.

- File name – Name the text file to which trace data will be written
- Save as type – The type of file is already selected as trace (*.trc)
- Start Cycle, End Cycle – Specify the range of cycles written

13.5 USING EXTERNAL MEMORY

Some Microchip devices allow the extension or replacement of program memory resources with external (off-chip) memory devices. How the emulator functions for these modes is discussed here.

- External Memory Access
- Simulating External Memory Access

13.5.1 External Memory Access

The amount of external memory accessed by selected devices and the method of access is discussed here.

13.5.1.1 PIC17CXXX PROGRAM MEMORY MODES

PIC17CXXX devices are capable of operating in any one of several program memory modes, using combinations of on-chip and external program memory. Available program memory modes are as follows:

- The **Microprocessor** mode permits access only to external program memory; the contents of the on-chip program memory are ignored. The program counter permits access to a linear program memory space and defines the amount of accessible program memory (see **Section 13.5.1.4 "Program Counter"**.) Microprocessor mode is the default mode of an unprogrammed device.
- The **Microcontroller** and **Protected Microcontroller** modes access only on-chip program memory. Attempts to read above the physical limit of the on-chip memory results in unknown data. The Protected Microcontroller mode also enables the code protection feature.
- The **Extended Microcontroller** mode allows access to both internal and external program memories as a single block. The device can access its entire on-chip program memory; above this, the device accesses external program memory up to the program counter accessible limit (see **Section 13.5.1.4 "Program Counter"**.) Execution automatically switches between the two memories, as required.

In all modes, the device has complete access to data RAM. For more information, consult the device data sheet section "Memory Organization".

13.5.1.2 PIC18F6XXX/8XXX PROGRAM MEMORY MODES

Certain PIC18F6XXX/8XXX devices are capable of operating in any one of several Program Memory modes, using combinations of on-chip and external program memory. Please see the data sheet for your device to determine if it supports external memory.

For supported devices, available Program Memory modes are as follows:

- The **Microprocessor** mode permits access only to external program memory; the contents of the on-chip Flash memory are ignored. The program counter permits access to a linear program memory space and defines the amount of accessible program memory (see **Section 13.5.1.4 "Program Counter"**.)
- The **Microprocessor with Boot Block** mode accesses on-chip Flash memory from address 000000h to the end of the boot block. Above this, external program memory is accessed all the way up to the program counter accessible limit (see **Section 13.5.1.4 "Program Counter"**.) Program execution automatically switches between the two memories, as required.
- The **Microcontroller** mode accesses only on-chip Flash memory. Attempts to read above the physical limit of the on-chip Flash causes a read of all '0's (a NOP instruction.)
- The **Extended Microcontroller** mode allows access to both internal and external program memories as a single block. The device can access its entire on-chip Flash memory; above this, the device accesses external program memory up to the program counter accessible limit (see **Section 13.5.1.4 "Program Counter"**.) As with Boot Block mode, execution automatically switches between the two memories, as required.

In all modes, the device has complete access to data RAM and EEPROM. For more information, consult the device data sheet section "Memory Organization".

13.5.1.3 PIC18C601/801 ROMLESS DEVICES

For PIC18C601/801 devices, all program memory address space is external. The on-chip program counter permits access to a linear program memory space and defines the amount of accessible program memory (see **Section 13.5.1.4 "Program Counter"**.)

There is a provision for configuring the last 512 bytes of general purpose user RAM as program memory, called "Boot RAM". See the device data sheet for more information.

13.5.1.4 PROGRAM COUNTER

The size of the program counter will determine how much program memory can be accessed, e.g., a 21-bit program counter permits access to a 2-Mbyte (1-Mword) program memory space (on-chip, off-chip or a combination of both types of program memory.)

13.5.1.5 CONFIGURATION BITS

For PIC17CXXX and PIC18F6XXX/8XXX devices, a Program Memory mode is set by using configuration bits. Depending on the device, the Processor Mode Select bits are located in a configuration register which is programmed when the device is programmed.

For PIC18C601/801 devices, External Bus Data Width is set as a configuration bit.

For more information, consult the device data sheet section "Special Features of the CPU".

13.5.1.6 EXTERNAL MEMORY INTERFACE

The External Memory Interface is a feature that allows the microcontroller to access external memory devices (such as Flash, EPROM, SRAM, etc.) as program or data memory.

PIC17 Devices

When either Microprocessor or Extended Microcontroller mode is selected, PORTC, PORTD and PORTE are configured as the system bus. PORTC and PORTD are the multiplexed address/data bus and PORTE<2:0> is for the control signals. External components are needed to demultiplex the address and data.

For more information, see the Memory Organization, External Memory Interface section of the device data sheet.

PIC18 Devices

Using the MEMCON register, the following may be configured:

- External bus enable and disable
- 16-Bit mode – Word Write mode, Byte Select mode or Byte Write mode
- Wait – Table read/write bus cycle wait counts (0-3 TCY)

For more information, see the External Memory Interface section of the device data sheet.

13.5.2 Simulating External Memory Access

Simulating a device that uses external memory requires the following steps:

1. Setting Configuration Bits
2. Setting External Memory

13.5.2.1 SETTING CONFIGURATION BITS

To set the values of configuration bits for your selected device, open the Configuration Bits window by selecting *Configure>Configuration Bits*. In the Category column, find the bits you need to set and click on them to change their value.

> **Note:** Configuration bits may also be set in code using `__config`. Refer to the device data sheet and header file (`.inc` or `.h`) for more information.

13.5.2.2 SETTING EXTERNAL MEMORY

To set up MPLAB IDE and MPLAB SIM to recognize external memory, select *Configure>External Memory*. Then check “Use External Memory” and enter a range.

NOTES:

Chapter 14. Using Stimulus

14.1 INTRODUCTION

During simulation, the program being executed by the simulator may require stimuli from the outside. This stimulus could be a level change or a pulse to an I/O pin of a port. It could be a change to the values in an SFR (Special Function Register) or other data memory.

In addition, stimulus may need to happen at a certain instruction cycle or time during the simulation. Alternately, stimulus may need to occur when a condition is satisfied; for example, when the execution of program has reached a certain instruction address during simulation.

Basically, there are two types of stimulus:

- Synchronous – Regular, repeating series of high/low voltages with adjustable duty cycle set by number of clock cycles per high/low time.
- Asynchronous – Fired by you, the user.

To define when, what and how external stimuli are to happen to a program, you would use the following:

- SCL Generator Dialog
- Stimulus Controller Dialog

If you will be using multiple forms of stimulus input, you should be aware of be aware of input interaction (see **Section 14.4 "Stimulus Input Interaction"**).

14.2 SCL GENERATOR DIALOG

Use the SCL Generator dialog to create synchronous stimulus using the behind-the-scenes, powerful Simulator Control Language (SCL). If you want to use only asynchronous stimulus, you may go directly to the Stimulus Controller Dialog to set this up.

The SCL Generator dialog allows you to enter stimulus information which is saved in a file called a workbook. To open a new workbook, select *Debugger>SCL Generator> New Workbook*. To open an existing workbook for editing, select *Debugger> SCL Generator>Open Workbook*.

The SCL Generator dialog has the tabs listed below. All tabs have the same default unit values.

- Pin/Register Actions
- Advanced Pin/Register
- Clock Stimulus
- Register Injection
- Register Trace

When setting up multiple tabs, be aware of input interaction (see **Section 14.4 "Stimulus Input Interaction"**).

Default Unit Values

For all tabs of the SCL Generator, the following applies:

- Times units are always in decimal, i.e, instruction cycles (cy), milliseconds (ms), microseconds (us) and nanoseconds (ns).
- PC values are always in hex.
- Pin values are always either 0 or 1.
- Register values are always in hex.
- Bitfield values are always in binary.

Label Constructs

Labels must be of the format:

alpha[_|alpha|numeric]

where:

- **alpha** = alphabetic character(s)
- **_** = underscore character
- **numeric** = numeric character(s)

All labels must:

- begin with an alphabetic character
- not end with an underscore (_)
- not include two underscores together

SFR Values As Triggers

Triggers and traces will ONLY occur when the SFR is updated by the user, not the peripheral.

For example, the trigger dialog is set up to fire when `TMR2 = 0x06`. When TMR2 is incremented past 0x06, the trigger will not fire. However, if the following sequence is executed in user code:

```
MOVLW 0X06
MOVWF TMR2
```

then the trigger will occur.

14.2.1 Pin/Register Actions

Basic synchronous pin and/or register actions may be entered here. For more complex actions, use the Advanced Pin/Register tab.

To enter data in this tab:

1. Select the unit of time in the "Time Units" column that you will use to trigger all stimulus.
2. Click on the text that says "Click Here to Add Signals" to open the Add/Remove Signals dialog (see **Section 14.2.3 "Add/Remove Signals Dialog"**.) In this dialog, you select the pins, registers or other signals to which you will apply stimulus. These selections will become the titles of the columns.
3. Fill out each row, entering a trigger time ("Time") and value for each pin/register column.

   Note: Care must be taken when using SFR values as triggers.

4. To delete a row, select it and then click **Delete Row**.

Once the tab is filled out, you may proceed to another tab or click **General SCL From All Tabs** to create the workbook file.

To scroll through signals:

- If you add more signals to the dialog than will fit in the window, a scroll bar will appear.
- You may scroll through all the signals to view their values.

> **Note:** The Time column remains fixed when you scroll through the signal columns.

14.2.2 Advanced Pin/Register

Advanced (complex) synchronous pin/register actions may be entered here. For basic actions, use the Pin/Register Actions tab.

Define your conditions first, and then define the triggers.

14.2.2.1 DEFINE CONDITIONS

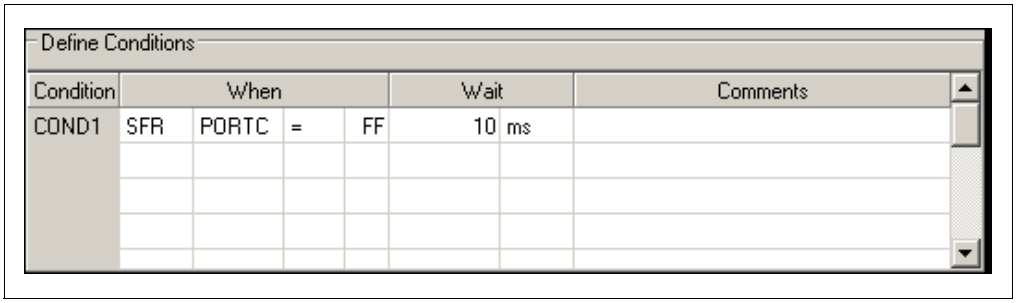
Define the conditions for one or more stimuli in each row of this section as shown in Table 14-1.

TABLE 14-1: DEFINITIONS OF STIMULI CONDITIONS

Item	Definition
Condition	A name for the condition you are specifying is automatically generated when you enter data in any other column. This label will be used to identify the condition in the Condition column of the Define Triggers section of this tab.
When	Define the condition. I.e., the condition is true when the value of the pin/register in Column 2 (its type specified in Column 1) has the relationship of Column 3 to the value of Column 4. **Column 1:** Select the type of pin/register, either “SFR”, “Bitfield”, “Pin” or “All” of the above. This will filter the content of Column 2. **Column 2:** Select the pin/register to which the condition will apply. **Column 3:** Select the condition, either equal (=), not equal (!=), less than or equal (<=), greater than or equal (>=), less than (<) or greater than (>). **Column 4:** Enter the value for the condition. **Note:** Care must be taken when using SFR values as triggers.
Wait	Once the condition defined in When is true, specify how long to wait until the stimulus is applied. **Column 1:** Wait time value. **Column 2:** Wait time value units.
Comments	Add descriptive information about the condition.

Set up a condition, COND1, such that when the value of register PORTC equals FF, stimulus defined in “Define Triggers” is applied 10 ms later.

EXAMPLE 14-1: REGISTER EQUALS A VALUE



14.2.2.2 DEFINE TRIGGERS

Define stimulus triggers in each row of this section as shown in Table 14-2, with reference to the conditions set up in "Define Conditions".

TABLE 14-2: DEFINITIONS OF STIMULUS TRIGGERS

Item	Definition
Enable	Alternately enable or disable the trigger setup in this row.
Condition	Refers to the label of the condition set up in the "Define Conditions" section of this tab. Select a condition from the list.
Type	Select whether the trigger condition will apply once (1x) or continuously/ repeatedly (Cont).
Re-Arm Delay	If Type = Cont, then enter a delay until the trigger condition is checked again. The delay value is entered in the first column and the delay value unit is selected in the second column.
Click Here to Add Signals	Click on the text that says "(Click here to add/remove signals)" to open the Add/Remove Signals dialog (see **Section 14.2.3 "Add/Remove Signals Dialog"**.) In this dialog, you select the pins, registers or other signals to which you will apply stimulus. These selections will become the titles of the columns.

To scroll through signals:

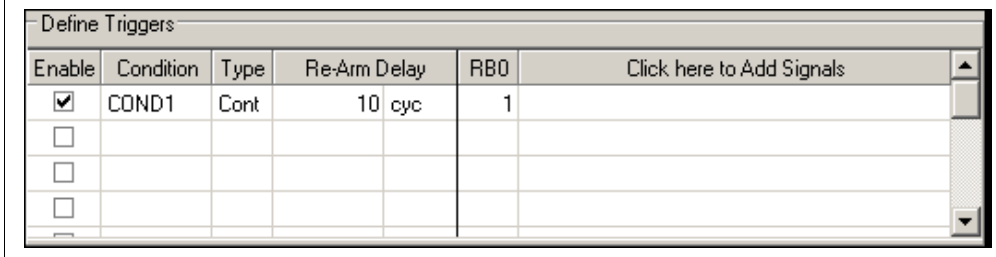
- If you add more signals than will fit in the window, a scroll bar will appear.
- You may scroll through all the signals to view their values.

> **Note:** The columns to the left of the signal column(s) remains fixed when you scroll through the signal columns.

For the condition set up in the previous example, COND1, set up the following stimulus trigger:

1. Make the pin RB0 high for the first instance of the condition.
2. Wait 10 instruction cycles and check for the condition again. If and when it occurs, make pin RB0 high again.
3. Repeat step 2 until the program is halted.

EXAMPLE 14-2: MAKE PIN HIGH ON REGISTER VALUE



14.2.2.3 "DEFINE CONDITIONS" WAIT VS. "DEFINE TRIGGER" RE-ARM DELAY

In "Define Conditions", the Wait time refers to the time from when the condition is true until the stimulus is applied. In "Define Trigger", the post-trigger delay is time from when the stimulus was applied until the condition for triggering is checked again.

14.2.3 Add/Remove Signals Dialog

Use this dialog to add or remove pins or registers to which stimulus will be applied according to the associated Pin/Register tab.

To Add a Signal:

- From the “Available Signals” drop-down box, select a type of signal, either “SFR and Field”, “Pin Only” or “All Signals”.
- In the list below, either:
  - Click on a signal and then click **Add** to add the signal to the “Selected Signals” list.
  - Double-click on a signal to add it to the “Selected Signals” list.

> **Note:** Pins that support peripheral I/O (e.g., CCP2) as well as digital I/O (e.g., RB3 or RC1) will only be listed by the digital I/O name, although the peripheral I/O function will be available if properly configured and supported in the simulator.

This signal will now appear as a column head on the associated Pin/Register tab.

To Delete a Signal:

- In the “Selected Signals” list, either:
  - Click on the signal and then click **Remove** to remove the signal from this list.
  - Double-click on the signal to remove the signal from this list.

This signal will now **not** appear as a column head on the associated Pin/Register tab.

To Change the Order of Signals on the Pin/Register tab:

- Click on the signal from the “Selected Signals” list.
- Click either **Move Up** or **Move Down** to change order of this signal in the list.

Its location will now be reflected in the order of the column heads on the associated Pin/Register tab.

14.2.4 Clock Stimulus

Pulse (high and low) values applied to a pin is clocked stimulus as shown in Table 14-3. Clocked stimulus may be entered here.

TABLE 14-3: DEFINITIONS OF CLOCK STIMULUS

Item	Definition	
Label	Name the clock stimulus you are specifying (optional).	
Pin	Choose the pin on which you will apply clocked stimulus.	
Initial	Enter the initial state of the clocked stimulus, either low or high.	
Low Cycles	Enter a value for the number of low cycles in a clock pulse.	
High Cycles	Enter a value for the number of high cycles in a clock pulse.	
Begin	Click here to reflect the selection in the Begin section:	
	Always (default)	Begin stimulus immediately on program run.
	PC=	Begin stimulus when the program counter equals the entered value.
	Cycle=	Begin stimulus when the instruction cycle count equals the entered value.
	Pin=	Begin stimulus when the selected pin has the selected value (low or high).
End	Click here to reflect the selection in the End section:	
	Never (default)	Apply stimulus until program halt.
	PC=	End stimulus when the program counter equals the entered value.
	Cycle=	End stimulus when the instruction cycle count equals the entered value.
	Pin=	End stimulus when the selected pin has the selected value (low or high).
Comments	Add descriptive information about the stimulus.	

14.2.5 Register Injection

Registers may be injected with values set up in a file, see Table 14-4. Enter information for register injection here.

TABLE 14-4: DEFINITIONS OF REGISTER INJECTIONS

Item	Definition
Label	Name the register injection you are specifying (optional).
Destination	Select the destination register* for data injection.
Trigger	Select when to trigger injection, either on Demand or when the PC equals a specified value (see next column). If the peripheral is implemented, it can only be triggered on demand. If the peripheral is not implemented, it must be triggered on PC. E.g., if the ADON bit is set and the register ADRESL is accessed, injection into ADRESL will occur.
PC Value	If Trigger=PC, enter a PC value for triggering injection.
Data Filename	Browse for the injection file.
Rewind	**Yes** – Once all the data from the file has been injected, start again from the beginning of the file. **No** – Once all the data from the file has been injected, the last value will continue to be used for injection.
Format	Select the format of the injection file. **Hex** – ASCII hexadecimal **Raw** – Raw image, interpreted as binary **SCL** – SCL format. See spec for definition. **Dec** – ASCII decimal
Comments	Add descriptive information about the register injection.

* Not all registers may be displayed. E.g., selecting the ADRESL register on the PIC18F458 for injection will actually inject stimulus into both the ADRESH and ADRESL registers.

EXAMPLE 14-3: REGISTER STIMULUS FILE IN HEX

```
110
02E
A38
541
1A0
0FD
```

14.2.6 Register Trace

The value of a specified register may be saved to a file (traced) during a run for certain conditions, see Table 14-5. Specify register trace information here.

TABLE 14-5: DEFINITIONS OF REGISTER TRACE

Item	Definition
Label	Name the register trace you are specifying (optional).
Source	Select the source register for tracing.
Trigger	Select when to trigger the trace, either on Demand or when the PC equals a specified value (see next column). If the peripheral is implemented, it can only be triggered on demand. If the peripheral is not implemented, it must be triggered on PC. E.g., if the ADON bit is set and the register ADRESL is accessed, injection into ADRESL will occur.
PC Value	If Trigger=PC, enter a PC value for triggering tracing.
Trace Filename	Browse for a location for the trace file.
Format	Select the format of the trace file. **Hex** – ASCII hexadecimal **Raw** – Raw image, interpreted as binary **SCL** – SCL format. See spec for definition. **Dec** – ASCII decimal
Comments	Add descriptive information about the register trace.

14.3 STIMULUS CONTROLLER DIALOG

Use the Stimulus Control dialog to control how stimulus is used by MPLAB SIM.

The Stimulus Control dialog allows you to set up what stimulus is used and saves this information in a scenario. To open a new scenario, select *Debugger>Stimulus Controller>New Scenario*. To open an existing scenario for editing, select *Debugger> Stimulus Controller >Open Scenario*.

When setting up SCL file input as well as asynchronous stimulus, be aware of Stimulus Input Interaction.

14.3.1 Stimulus (SCL) File

If you have developed a synchronous stimulus file using the SCL Generator Dialog, click **Attach** to search for the file and location, and attach it to the controller.

To remove an SCL file from the controller, click **Detach**.

If you wish to merge two SCL files into one, click **Import/Merge**.

14.3.2 Asynchronous Stimulus

Enter any asynchronous, or user-fired, stimulus here, row by row. To remove a row, select the row and then click **Delete Row**.

Item	Definition
Fire	Click the button in this column corresponding to the row for which you want to trigger stimulus. Obviously, you must set up all other row items before you can use Fire.
Pin	Select or change the pin on which stimulus will be applied. Available pin names are listed in a drop-down list. The actual names depend on the device selected in the IDE.
Action	Select or change a stimulus action. You may choose Pulse, High, Low or Toggle.
Width	If Pulse was chose for the Action, then you may specify or change a pulse width value here. Enter the units for this value in the next cell of the row.
Units	If Pulse was chose for the Action, then you may specify or change a pulse width unit here. Enter the value for this unit in the previous cell of the row.
Comments	Specify or change a comment about the stimulus.

14.4 STIMULUS INPUT INTERACTION

If a pin (e.g., RB1) assignment and a port (e.g., PORTB.RB) assignment happen on the same cycle, the port supersedes the pin assignment. But, if they happen on different cycles, you can mix and match pin and port assignment.

In addition, you can use PORT injection with pin/port assignment.

NOTES:

Chapter 15. Using Stimulus – PIC17 Devices

15.1 INTRODUCTION

Stimulus functions allow you to direct the simulator to apply artificial stimuli while debugging your code. You can set pins high or low and inject values directly into registers.

Select Stimulus from the Debugger menu, then choose the tab for the stimulus type.

Topics covered in this chapter:

- Using Pin Stimulus – Synchronous or Asynchronous stimulus on I/O pins
- Using File Stimulus – Triggered stimulus on I/O pins or file registers, set up from files

15.2 USING PIN STIMULUS

Pin Stimulus is user-defined. It may be:

- Asynchronous – Fired by the user.
- Synchronous – Regular, repeating series of high/low voltages with adjustable duty cycle set by number of clock cycles per high/low time.

Pins can be set high or low at program counter addresses. Addresses in the list will be looked for sequentially.

The contents of the pin stimulus list can be edited, saved and restored from a file.

Note:	Changing stimulus files outside the Stimulus dialog is NOT recommended.

- Creating/Editing Pin Stimulus
- Applying Pin Stimulus
- Pin Stimulus Display

15.2.1 Creating/Editing Pin Stimulus

1. Select *Debugger>Stimulus Controller* and then click the **Pin Stimulus** tab.
2. To create a new file, click **Add Row** to create a new row for entering data. To open an existing pin stimulus file for editing, click **Load** and select the file using the Open dialog.
3. Click in the "Type" column of the table row for which you wish to add or change data. Select or change the type of stimulus (Async or Synch). Depending on the stimulus type chosen, each cell gets an appropriate control which allows you to specify the value of that cell.

Asynchronous Setup

a) Click on "Pin" to select or change the pin on which stimulus will be applied. Available pin names are listed. The actual names depend on the device selected in the IDE.
b) Click on "Action" to select or change a stimulus action. You may choose Pulse, High, Low or Toggle.
c) Click on "Comments" to specify or change a comment which will be saved and restored in the pin stimulus file.

Synchronous Setup

a) Click on "Pin" to select or change the pin on which stimulus will be applied. Available pin names are listed. The actual names depend on the device selected in the IDE.
b) Click on "High Cycles" to set or change the number of Clock cycles till the high state is applied. Then click on "Low Cycles" to set or change the number of Clock cycles till the low state is applied.
c) Click on "Invert" to set or change whether high and low cycles are inverted.
d) Click on "Comments" to specify or change a comment which will be saved and restored in the pin stimulus file.

4. When you have completed adding or editing data in a row, click **Edit Complete**.
5. Continue adding or editing other rows of data till your pin stimulus is complete. If you wish to remove a row, click on a cell in the row to select the row and then click **Delete Row**.
6. When you are done, click **Save** and save the pin stimulus file (.psti).

15.2.2 Applying Pin Stimulus

Once you have created/edited a pin stimulus file, you may apply the stimulus as follows:

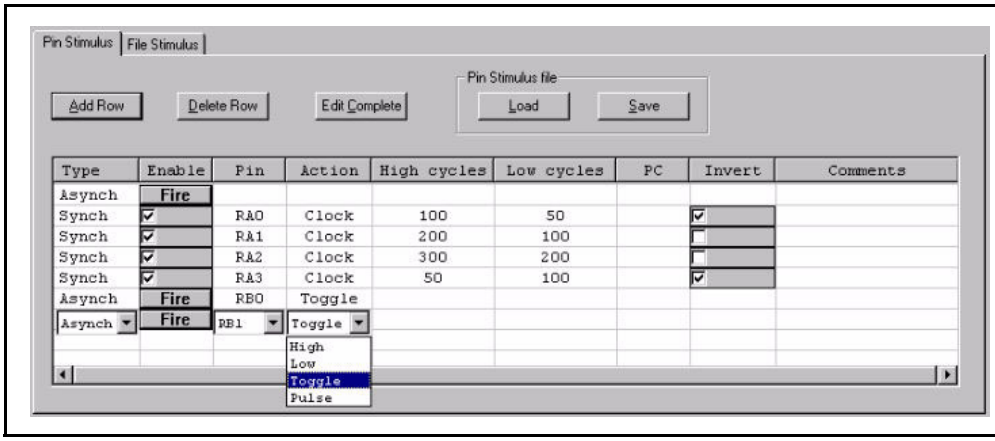
- Asynchronous stimulus (Type = Asynch): Press the Fire button. The designated event occurs on the designated pin.
- Synchronous stimulus (Type = Synch): Check the enable field. A message is sent to MPLAB SIM to update the stimuli. Thus whatever is displayed is current.

Note:	The Stimulus dialog must be open for stimulus to be active.

15.2.3 Pin Stimulus Display

Select *Debugger>Stimulus* and then click the Pin Stimulus tab to set up pin stimulus. Closing the Stimulus dialog disables stimulus.

Note 1:	If any files are changed externally while in use by this dialog tab, the changes will NOT be reflected in the stimuli passed to the simulator. Changing stimulus files outside the Stimulus dialog is NOT recommended.
2:	Unsaved changes will be lost without notice when you close this dialog.



Buttons – Row Edit

- Add Row – allows you to add a new pin stimulus. A default pin row at the bottom of the pin list will be created and the cursor will be moved there for editing.
- Delete Row – removes the row at the cursor from the list.
- Edit Complete – turns off the display of any list boxes, so all the data appear clearly.

Buttons – Pin Stimulus File

- Load – allows you to specify a previously-saved pin stimulus file and add its contents to the stimulus list.
- Save – allows you to save the current contents of the stimulus list to a pin stimulus file of your choice.

Table Entries

- Type – has a choice list containing “Asynch” and “Synch”. The Type you select determines what is available in some of the remaining cells in that row.
- Enable – has a check box for Synch stimulus, a Fire button for Asynch Stimulus. This is not a data cell.
- Pin – contains the names of the available pins. The actual names depend on the device selected in the IDE.

- Action
  Type = Asynch:
  Pulse: Change the state of the pin to its opposite and return.
  High: Change the state of the pin to high.
  Low: Change the state of the pin to low.
  Toggle: Change to state of the pin to its opposite.
  Type = Synch: Clock
- High cycles – number of cycles high state will be applied for Synch stimulus.
- Low cycles – number of cycles low state will be applied for Synch stimulus.
- Invert – inverts high and low cycles when checked for Synch stimulus.
- Comments – allows you to specify a comment which will be saved and restored if you maintain Pin Stimulus in a file.

15.3 USING FILE STIMULUS

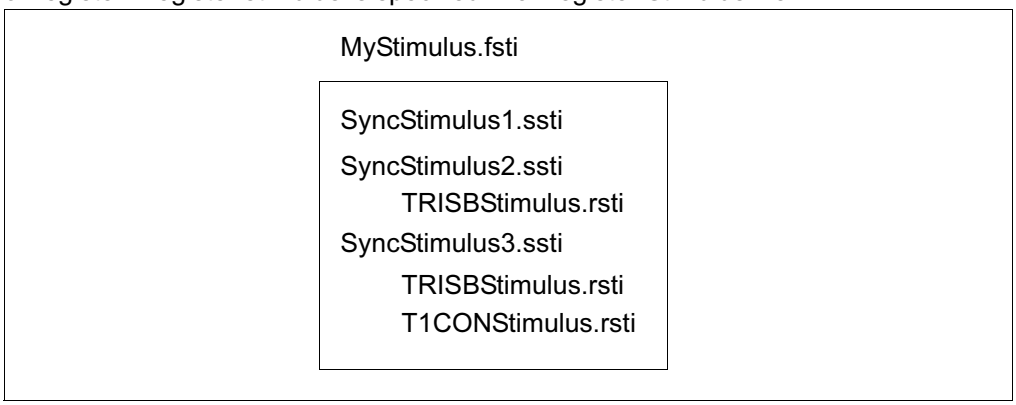
MPLAB SIM triggers changes to pin and register values specified in a file by using File Stimulus. These files specify values that will be sent to a pin or register when a trigger is fired. This trigger may be a cycle count for pins or a program memory address for registers.

Note 1:	Changing stimulus files outside the Stimulus dialog is NOT recommended.
2:	If you have already set pin stimulus on a port pin, you will not be able to inject stimulus into that pin or corresponding register using file stimulus.

- Creating/Editing File Stimulus
- Applying File Stimulus
- File Stimulus Display

15.3.1 Creating/Editing File Stimulus

Select *Debugger>Stimulus Controller* and then click the **File Stimulus** tab. A File Stimulus file is composed of one or more Synchronous Stimulus files. A Synchronous Stimulus file contains information on triggers used for applying stimulus to either a pin or register. Register stimulus is specified in a Register Stimulus file.



- Creating/Editing a File Stimulus File (.fsti)
- Creating/Editing a Synchronous Stimulus File (.ssti)
- Creating/Editing a Register Stimulus File (.rsti)

15.3.1.1 CREATING/EDITING A FILE STIMULUS FILE (.FSTI)

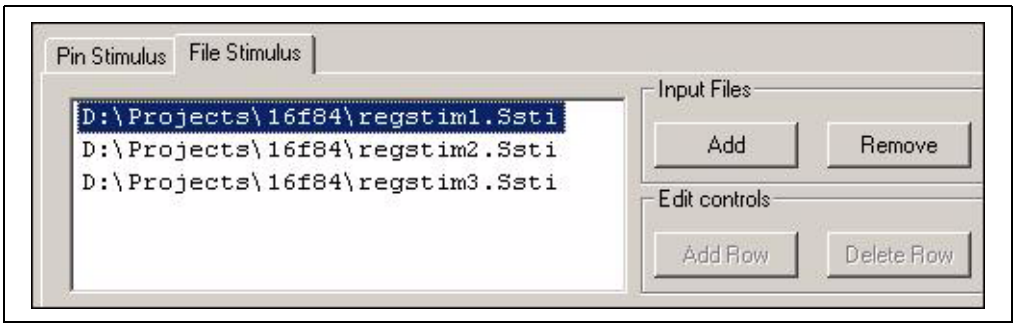
A file stimulus file consists of one or more synchronous stimulus files (*.ssti) that will be applied to the simulator.

To create a file stimulus file:

1. Create or open one or more synchronous stimulus files in the file list (below the tab).
2. Click **Save Setup** under File Stimulus.

To edit an existing file stimulus file:

1. Click **Load Setup** under File Stimulus to open an existing file stimulus file.
2. Use the file list and “Input Files” buttons **Add** and **Remove** to edit the list.



15.3.1.2 CREATING/EDITING A SYNCHRONOUS STIMULUS FILE (.SSTI)

1. If there are no stimulus files listed in the file list (below the tab), click **Add** under “Input Files” to open the Open file dialog. Browse to a location to create a new file or to edit an existing file. To create a new stimulus file, enter a name and click **Open**. To open an existing stimulus file for editing, click on it to select it and click **Open**. The stimulus file location and name should now appear in the file list.
2. Click on the stimulus file in the list to select it for editing. Then click **Edit**. If you realize you have selected the wrong file, click **Cancel** and then select another file.
3. For a new file, click **Add Row** to create a new row for entering data.
4. Click in the “Trigger On” column of the table row for which you wish to add or change data. Select or change the type of trigger (Cycles and PC). Depending on the trigger type, each cell gets an appropriate control which allows you to specify the value of that cell.

   Cycles Setup

   a) Click on “Trig Value” to set the cycle count at which the trigger fires.
   b) From “Pin/Register”, choose the pin to which the value in the next column will be applied when the trigger fires. The actual pin names available depend on the device selected.

Note:	Do not choose a register. It will not work with cycle stimulus.

   c) Click on “Value” and enter or change a value to be applied when the trigger occurs. For a Pin, either 0 or 1. These values will be applied in turn, one at each trigger event.
   d) For setting up additional pins, click **Add Column** to add “Pin/Register” and “Value” columns.
   e) Click on “Comments” to enter or change a text string which will be saved and restored in the file.

PC Setup

a) Click on "Trig Value" to set the PC address at which the trigger fires.
b) From "Pin/Register", choose the register to which the value in the next column will be applied when the trigger fires. The actual register names available depend on the device selected.

Note:	Do not choose a pin. It will not work with PC stimulus.

Note:	You cannot inject file values into port registers, e.g., PORTB. Use pin values. Also, you cannot inject file values into the W register.

c) Click on "Value" and enter or change a value to be applied when the trigger occurs. For a Register, the name of a file (`*.rsti`) which contains a sequence of values. These values will be applied in turn, one at each trigger event.
d) Click on "Comments" to enter or change a text string which will be saved and restored in the file.
e) For register stimulus, you may only have one stimulus file attached to one register and vice versa. If you add additional rows/columns, only the last occurrence will be executed.
f) If you wish to remove a row, click on a cell in the row to select the row and then click **Delete Row**.

5. When you are done, click **Save** to save the synchronous stimulus file.

15.3.1.3 CREATING/EDITING A REGISTER STIMULUS FILE (.RSTI)

A register stimulus file contains register stimulus information in hexadecimal, return-delimited format. Any text editor or spreadsheet application may be used to create or edit this file. The default extension is .rsti, but any extension may be used. Assign a register stimulus file to a register in the Value column of a Synchronous Stimulus file (*.ssti).

Note:	You may only have one register stimulus file assigned to one register at a time.

EXAMPLE 15-1: REGISTER STIMULUS FILE

```
10
2E
38
41
A0
FD
```

15.3.2 Applying File Stimulus

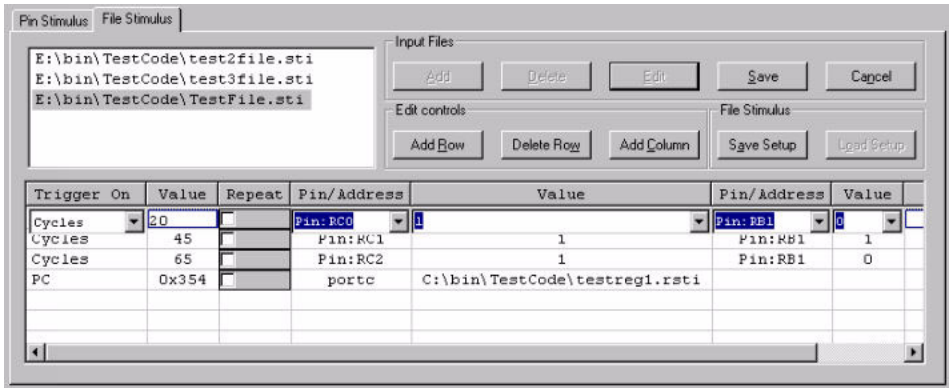
Once you have created/edited a file stimulus file, the stimulus is applied as the program is run under MPLAB IDE. When the designated trigger is activated, the designated event occurs.

Note:	The Stimulus dialog must be open for stimulus to be active.

15.3.3 File Stimulus Display

Select *Debugger>Stimulus Controller* and then click the File Stimulus tab to set up file stimulus. Closing the Stimulus dialog disables stimulus.

Note 1:	This is only available when the MPLAB SIM tool has been selected from the Debugger menu.
2:	If any files are changed externally while in use by this dialog tab, the changes will NOT be reflected in the stimuli passed to the simulator. Changing stimulus files outside the Stimulus dialog is NOT recommended.
3:	Unsaved changes will be lost without notice when you close this dialog.



File List

List of input files available for use as file stimulus. Add, delete, edit and save files from this list by using the Input Files buttons.

Buttons – Input Files

- Add – brings up a browse dialog to select a new file to add to the file list. You may select an existing file or enter the name of a file to be created.
- Delete – deletes the highlighted file from the list (not from your drive)
- Edit – displays the contents of the selected file in the stimulus list and allows you to edit the contents. It also disables the Add and Delete buttons.
- Save – saves the contents of the stimulus list into a file, clears the stimulus list, disables editing and reenables the Add and Delete buttons.
- Cancel – Cancel selection of file.

Note:	You may display and edit only one file from the file list at a time.

Buttons – Edit Controls

- Add Row – creates a default row at the bottom of the stimulus list and moves the cursor there for editing.
- Delete Row – removes the row at the cursor from the list.
- Add Column – adds another Pin/Register-Value pair of columns to the stimulus list to the left of the Comments column, allowing you specify additional stimuli for each trigger.

Buttons – File Stimulus

These buttons enable you to save and restore combined file setups.

- Save Setup saves the names of the files currently in the file list into a file.
- Load Setup loads the filenames in a file into the file list, just as though you had added them individually.

Note: There is currently no way to view the combined contents of a File Stimulus setup.

Table Entries

- Trigger On – choose between Cycles (Instruction Cycles) and PC (Program Counter). The selection determines what is available in some of the remaining cells in that row.
- Trig Value – Cycle count or the PC address at which the trigger fires. Value entered can be decimal or hexadecimal. Hex numbers must be preceded with 0x.
- Pin/Register – where to apply the value; a choice among the names of the available pins and registers. The actual names depend on the device selected in the IDE.
- Value – value to be applied when the trigger occurs. For a Pin, either 0 or 1. For a Register, a single value or the name of a file which contains a sequence of values. These values will be applied in turn, one at each trigger event.
- Comments – text string which will be saved and restored in the file.

Note: A Register stimulus will be effective only with a PC trigger. You may specify a Cycle trigger for a Register, but it will have no effect. In addition, Register values must be specified from a file; specifying a numeric value will have no effect.

Chapter 16. Simulator Troubleshooting

16.1 INTRODUCTION

This section is designed to help you troubleshoot any problems, errors or issues you encounter while using MPLAB SIM. If none of this information helps you, please see Preface for ways to contact Microchip Technology.

- Common Problems/FAQ
- Limitations

16.2 COMMON PROBLEMS/FAQ

The simulator does not use the stimulus file I created

If your file is incorrect, the simulator will not notify you; it will simply ignore its contents. You should only set up your stimulus files using either:

- the SCL Generator and Stimulus Control dialogs
- the Pin Stimulus and File Stimulus dialogs (PIC17 Devices)

Do not edit files externally.

In addition, there may be pin/port interaction concerns. See the documentation for the type of stimulus you are using.

I cannot attach more than one register stimulus file to a register

MPLAB SIM only allows one register stimulus file (.rsti) per register in stimulus for PIC17 MCU devices.

I cannot set conditional breakpoints.

Conditional breakpoints are not yet implemented.

The symbolic breakpoints I set in the Breakpoint dialog do not work.

This dialog does not recognize symbolic address. However, if you enter a symbol name whose starting characters can be interpreted as a valid hex address, a breakpoint will be set at that address.

The simulator is not recognizing my breakpoints, although they seem to be set and enabled in my code windows.

Be sure "Global Break Enable" is set on the **Break Options** tab, Settings dialog.

I cannot reset my program using MCLR

For dsPIC or PIC18F devices, you cannot reset the device using pins. You must use the reset commands on the Debugger menu.

My program keeps resetting

Check the configuration bits settings (*Configure>Configuration Bits*) for your selected device. Some reset functions (such as Watchdog Timer Reset) are enabled by default.

The simulator is not halting on WDT time-out.

Be sure "Global Break Enable" is set in the **Break Options** tab, Settings dialog and you have enabled WDT time-out in the Configuration Bits dialog.

16.3 LIMITATIONS

General and device-specific limitations for the emulator may be found in on-line help for MPLAB SIM.

Chapter 17. Simulator Reference

17.1 INTRODUCTION

Once MPLAB SIM has been selected as the debug tool in MPLAB IDE (*Debugger> Select Tool>MPLAB SIM*), the following simulator-specific features are available:

- Debugging Functions
- Settings Dialog
- Settings Dialog – PIC17 Devices

17.2 DEBUGGING FUNCTIONS

Simulator-specific debug items are added to the following MPLAB IDE features:

- Debugger Menu – Standard debugging operations plus Stopwatch and Stimulus
- View Menu – Simulator Trace
- Right Mouse Button Menus – Standard debugging operations
- Toolbar – Standard Debug toolbar
- Status Bar – MPLAB SIM will be identified as the debug tool

17.2.1 Debugger Menu

In addition to the standard MPLAB IDE Debug menu items, the following tool-specific items will appear:

Stopwatch

Time the execution of your code with a stopwatch. For more information, see.

Stimulus Controller

For PIC17 devices: Setup I/O Pin and File Register stimuli for your code and apply them as your program executes. For more information, see **Chapter 15. "Using Stimulus – PIC17 Devices"**.

For all other devices: Set up stimulus using the Stimulus Control dialog. For more information, see **Chapter 14. "Using Stimulus"**.

SCL Generator

Develop code in SCL to control stimulus firing. For more information, see **Chapter 14. "Using Stimulus"**.

Profile

Show a profile of the current state of the simulator in the Output window or clear (reset) the current state. For more information, see **Chapter 14. "Using Stimulus"**.

Refresh PM

Refresh program memory to restart your program.

17.2.2 View Menu

Simulator Trace

Display the window containing the current memory trace of your program's execution. For more information, see **Section 13.4 "Using Simulator Trace"**.

17.3 SETTINGS DIALOG

Select *Debugger>Settings* to open the Settings dialog.

Note:	This is only available when the simulator has been selected from the Debugger Menu.

This dialog is composed of several tabs for setting up debugger features and functions.

- Osc/Trace tab
- Break Options tab
- SCL Options tab
- UART1 I/O tab
- Limitations tab

17.3.1 Osc/Trace tab

Select *Debugger>Settings* and then click the Osc / Trace tab.

Processor Frequency

In this dialog you set the clock speed at which you want your code to run. These values will also be used to set certain fields in the Stopwatch.

1. Select the Units of frequency
2. Enter the Frequency value

Note:	Configuration bit settings will have no effect on processor speed, e.g., if using x4 PLL, simulator will run at processor frequency, not 4 times processor frequency.

Trace Options

Enable/disable trace by checking/unchecking the "Enable Trace" checkbox. To stop program execution when the trace buffer is full, check "Break on Trace Buffer Full".

To see the trace window, select *View>Simulator Trace*. For more on tracing, see.

17.3.2 Break Options tab

Select *Debugger>Settings* and then click the **Break Options** tab. Set up break options for specific processor areas or conditions.

Messages are displayed as they are generated in the Output window under the MPLAB SIM tab. The messages will be displayed in a standard format as shown below:

```
Component-message number :  message
```

For example,

```
CORE-E0001: Stack Error occurred
ADC-W0001: Cannot change ADCON1 SSRC when ADON=1
```

Core

- Break – Report and Break on Core errors
- Ignore – Ignore Core errors (Trap, normal behavior)
- Report – Report and Trap on Core errors.

Peripheral

- Break – Report and Break on Peripheral errors
- Ignore – Ignore Peripheral errors (Trap/Reset, normal behavior)
- Report – Report and Trap on Peripheral errors

Stack Full / Underflow

Note: If the core is set to break on errors, you may break on a stack error even if you do not have the stack set to break. This occurs because the stack is part of the core.

- Break – Break on Stack error
- Reset – Reset on Stack error (Trap, normal behavior)

WDT Time-out

- Break – Break on WDT
- Break+Warn – Report and Break on WDT
- Reset – Reset on WDT (normal behavior)

17.3.3 SCL Options tab

Select *Debugger>Settings* and then click the SCL Options tab. Set up SCL error screening options.

Messages are displayed as they are generated in the Output window under the MPLAB SIM tab. The messages will be displayed in a standard format as shown below:

```
Component-message number :  message
```

For example,

```
CORE-E0001: Stack Error occurred
ADC-W0001: Cannot change ADCON1 SSRC when ADON=1
```

SCL

- Report and Break on SCL errors
- Ignore SCL errors (Trap/Reset, normal behavior)
- Report and Trap on SCL errors.

17.3.4 UART1 I/O tab

Select *Debugger>Settings* and then click the UART1 I/O tab.

Combined with the UART is the ability to use the standard C library I/O capability. When the "Enable UART1 I/O" option is checked, the UART1 is used to read data from a file and write data to a file or the output window.

Input

The stimulus file used for input is selected by clicking the **Browse** button and using the Open file dialog.

Checking "Rewind Input" means that when a stimulus file is read to its end, it will wrap and restart at the beginning, i.e., you will have a continuous loop of input data. Unchecking this item means that when a stimulus file is read to its end, it will stop and no more stimulus input will be injected (until you reset.)

Output

If a file is used for the output, click "File" and select the file name by clicking the **Browse** button and using the Save As file dialog.

If the Output window is to be used for output, click "Window".

17.3.5 Limitations tab

Select *Debugger>Settings* and then click the Limitation tab.

This dialog presents known MPLAB SIM limitations for the device currently selected (*Configure>Select Device*).

If you want additional information about device limitations, click **Details**.

17.4 SETTINGS DIALOG – PIC17 DEVICES

Select *Debugger>Settings* to open the Settings dialog.

Note:	This is only available when the simulator has been selected from the Debugger Menu.

This dialog is composed of several tabs for setting up debugger features and functions.

- Clock tab
- Break Options tab
- Trace/Pins tab
- Limitations tab

17.4.1 Clock tab

Select *Debugger>Settings* and then click the Clock tab.

In this dialog you set the clock speed at which you want your code to run. These values will also be used to set certain fields in the Stopwatch.

1. Select the Units of frequency
2. Enter the Frequency value

17.4.2 Break Options tab

Select *Debugger>Settings* and then click the Break Options tab.

Global Break Enable

If this is checked, program breakpoints will be enabled.

Stack Options

These options control the simulator's behavior on Stack Full/Underflow

- Disable Stack Full/Underflow Warning

If this is checked, the simulator will suppress warnings when it detects Stack Full and Stack Underflow conditions.

Choose one of the following to determine the simulator behavior when it detects Stack Full and Stack Underflow conditions.

- Break on Stack Full/Underflow
- Reset on Stack Full/Underflow

Note:	The Stack group is disabled unless the STACK OVERFLOW RESET is enabled in the Configuration Bits dialog.

WDT (Watchdog Timer) Options

Choose one of the following to determine the simulator behavior when it detects WDT Time-out.

- Break On WDT time-out
- Reset on WDT time-out

Note:	The WDT group is disabled if the WDT is disabled in the Configuration Bits dialog.

17.4.3 Trace/Pins tab

Select *Debugger>Settings* and then click the Trace/Pins tab.

MCLR Pull-up Enabled

If this is checked, MCLR Pull-up will be enabled.

Trace Options

- Trace Enabled

If this is checked, instruction execution will be traced.

- Break on Trace Buffer Full

If this is checked, the simulator will halt when the trace buffer is full.

17.4.4 Limitations tab

Select *Debugger>Settings* and then click the Limitations tab.

This dialog presents known MPLAB SIM limitations for the device currently selected (*Configure>Select Device*).

If you want additional information about device limitations, click the **Details** button.

NOTES:

Glossary

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).

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).

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 PICmicro microcontroller.

Archive

A collection of relocatable object modules. It is created by assembling multiple source files to object files, and then using the archiver to combine the object files into one library file. A library can be linked with object modules and other libraries to create executable code.

Archiver

A tool that creates and manipulates libraries.

ASCII

American Standard Code for Information Interchange is a character set encoding that uses 7 binary digits to represent each character. It includes upper and lower case letters, digits, symbols and control characters.

Assembler

A language tool that translates assembly language source code into machine code.

Assembly Language

A programming language that describes binary machine code in a symbolic form.

Asynchronous Stimulus

Data generated to simulate external inputs to a simulator device.

Breakpoint, Hardware

An event whose execution will cause a halt.

Breakpoint, Software

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

A general-purpose programming language which features economy of expression, modern control flow and data structures and a rich set of operators.

Calibration Memory

A special function register or registers used to hold values for calibration of a PICmicro microcontroller on-board RC oscillator or other device peripherals.

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.

Configuration Bits

Special-purpose bits programmed to set PICmicro 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.

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.

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.

Device Programmer

A tool used to program electrically programmable semiconductor devices such as microcontrollers.

Digital Signal Controller

A microcontroller device with digital signal processing capability, i.e., Microchip dsPIC devices.

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.

DSC

See Digital Signal Controller.

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.

Emulation

The process of executing software loaded into emulation memory as if it were firmware residing on a microcontroller device.

Emulation Memory

Program memory contained within the emulator.

Emulator

Hardware that performs emulation.

Emulator System

The MPLAB ICE 2000 and MPLAB ICE 4000 emulator systems include the pod, processor module, device adapter, cables and MPLAB IDE software.

Environment – IDE

The particular layout of the desktop for application development.

Environment – MPLAB PM3

A folder containing files on how to program a device. This folder can be transferred to a SD/MMC card.

EPROM

Erasable Programmable Read Only Memory. A programmable read-only memory that can be erased usually by exposure to ultraviolet radiation.

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.

Export

Send data out of the MPLAB IDE in a standardized format.

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 PIC17 or PIC18 device.

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.

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 PICmicro 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.

GPR

General Purpose Register. The portion of device data memory (RAM) available for general use.

Halt

A stop of program execution. Executing Halt is the same as stopping at a breakpoint.

Hex Code

Executable instructions stored in a hexadecimal format code. hex code is contained in a hex file.

Hex File

An ASCII file containing hexadecimal addresses and values (hex code) suitable for programming a device.

High Level Language

A language for writing programs that is further removed from the processor than assembly.

ICD

In-Circuit Debugger. MPLAB ICD 2 is Microchip's in-circuit debugger.

ICE

In-Circuit Emulator. MPLAB ICE 2000 and MPLAB ICE 4000 are Microchip's in-circuit emulators.

IDE

Integrated Development Environment. MPLAB IDE is Microchip's integrated development environment.

Import

Bring data into the MPLAB IDE from an outside source, such as from a hex file.

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.

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.

Interrupt Handler

A routine that processes special code when an interrupt occurs.

Interrupt Request

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

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.

IRQ

See Interrupt Request.

ISO

See International Organization for Standardization.

ISR

See Interrupt Service Routine.

Librarian

See Archiver.

Library

See Archive.

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.

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.

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

Macroinstruction. 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.

Make Project

A command that rebuilds an application, re-compiling only those source files that have changed since the last complete compilation.

MCU

Microcontroller Unit. An abbreviation for microcontroller. Also uC.

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 PIC17 and 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 PIC17 and 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 PICmicro microcontroller devices, KEELOQ devices and Microchip memory devices.

MPLAB ASM30

Microchip's relocatable macro assembler for dsPIC30F digital signal controller devices.

MPLAB C1X

Refers to both the MPLAB C17 and MPLAB C18 C compilers from Microchip. MPLAB C17 is the C compiler for PIC17 devices and MPLAB C18 is the C compiler for PIC18 devices.

MPLAB C30

Microchip's C compiler for dsPIC30F digital signal controller devices.

MPLAB ICD 2

Microchip's in-circuit debugger that works with MPLAB IDE. The ICD supports Flash devices with built-in debug circuitry. The main component of each ICD is the module. A complete system consists of a module, header, demo board, cables and MPLAB IDE software.

MPLAB ICE 2000/MPLAB ICE 4000

Microchip's in-circuit emulators work with MPLAB IDE. MPLAB ICE 2000 supports PICmicro MCUs. MPLAB ICE 4000 supports PIC18F MCUs and dsPIC30F DSCs. The main component of each ICE is the pod. A complete system consists of a pod, processor module, cables and MPLAB IDE Software.

MPLAB IDE

Microchip's Integrated Development Environment.

MPLAB LIB30

MPLAB LIB30 archiver/librarian is an object librarian for use with COFF object modules created using either MPLAB ASM30 or MPLAB C30 C compiler.

MPLAB LINK30

MPLAB LINK30 is an object linker for the Microchip MPLAB ASM30 assembler and the Microchip MPLAB C30 C compiler.

MPLAB PM3

A device programmer from Microchip. Programs PIC18 microcontrollers and dsPIC digital signal controllers. Can be used with MPLAB IDE or stand-alone. Will obsolete PRO MATE II.

MPLAB SIM

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

MPLIB Object Librarian

MPLIB librarian is an object librarian for use with COFF object modules created using either MPASM assembler (mpasm or mpasmwin v2.0) or MPLAB C1X C compilers.

MPLINK Object Linker

MPLINK linker is an object linker for the Microchip MPASM assembler and the Microchip MPLAB C17 or C18 C compilers. 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.

Nesting Depth

The maximum level to which macros can include other macros.

Node

MPLAB IDE project component.

Non Real-Time

Refers to the processor at a breakpoint or executing single step instructions or MPLAB IDE being run in simulator mode.

Non-Volatile Storage

A storage device whose contents are preserved when its power is off.

NOP

No Operation. An instruction that has no effect when executed except to advance the program counter.

Object Code

The machine code generated by an assembler or compiler.

Object File

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.

Off-Chip Memory

Off-chip memory refers to the memory selection option for the PIC17 or PIC18 device where memory may reside on the target board, or where all program memory may be supplied by the Emulator.

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.

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 IBM or compatible personal computer running a supported Windows operating system.

PICmicro MCUs

PICmicro microcontrollers (MCUs) refers to all Microchip microcontroller families.

PICSTART Plus

A developmental device programmer from Microchip. Programs 8-, 14-, 28- and 40-pin PICmicro microcontrollers. Must be used with MPLAB IDE Software.

Pod, Emulator

The external emulator box that contains emulation memory, trace memory, event and cycle timers and trace/breakpoint logic.

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.

PRO MATE II

A device programmer from Microchip. Programs most PICmicro microcontrollers as well as most memory and KEELOQ devices. Can be used with MPLAB IDE or stand-alone.

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 Memory

The memory area in a device where instructions are stored. Also, the memory in the emulator or simulator containing the downloaded target application firmware.

Project

A set of source files and instructions to build the object and executable code for an application.

Prototype System

A term referring to a user's target application or target board.

PWM Signals

Pulse Width Modulation Signals. Certain PICmicro MCU devices have a PWM peripheral.

Qualifier

An address or an address range used by the Pass Counter or as an event before another operation in a complex trigger.

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.

Real-Time

When released from the halt state in the emulator or MPLAB ICD mode, 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 MPLAB ICE is enabled and constantly captures all selected cycles, and all break logic is enabled. In the emulator or MPLAB ICD, the processor executes in real-time until a valid breakpoint causes a halt, or until the user halts the emulator. In the simulator real-time simply means execution of the microcontroller instructions as fast as they can be simulated by the host CPU.

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.

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.

Scenario

For MPLAB SIM simulator, a particular setup for stimulus control.

SFR

See Special Function Registers.

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 though 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 some formal programming language which can be translated into or machine code or executed by an interpreter.

Source File

An ASCII text file containing source code.

Special Function Registers

The portion of data memory (RAM) dedicated to registers that control I/O processor functions, I/O status, timers or other modes or peripherals.

Stack, Hardware

Locations in PICmicro 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.

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 step over subroutines. This command executes the code in the subroutine and then stops execution at the return address to the subroutine.

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. Select Halt to regain control of program execution.

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.

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.

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."

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.

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.

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.

Warning

An alert that is provided to warn you of a situation that would cause physical damage to a device, software file or equipment.

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

A timer on a PICmicro microcontroller that resets the processor after a selectable length of time. The WDT is enabled or disabled and set up using configuration bits.

WDT

See Watchdog Timer.

Workbook

For MPLAB SIM stimulator, a setup for generation of SCL stimulus.

NOTES:

Index

Numerics

12-Bit Core Model 168
14-Bit Core Model 169
16-Bit Core Model - PIC17 171
16-Bit Core Model - PIC18 174
16-Bit Mode 184

A

About 96
About MPLAB IDE Dialog 130
Active Project 72
Add Files to Project Dialog 134
Add Watch Dialog 130
Animate 94
Auto Indent 152

B

Background Colors 103
Bank, Data Memory 99
Bookmarks 91, 156
Boot Block Mode 183
Break Options Tab 208, 210
Breakpoint Symbol 103
Breakpoint, Unresolved 132
Breakpoints 95, 108, 111, 127, 129, 152, 161
Breakpoints Dialog 131
Build a Project 107
Build Options Dialog 132

C

C Code 94
Change Colors 103
Change Fonts 103
Changing Window Data and Properties 102
Checkmark 103
Checksum 99
Clean 93
Clear Memory 94
Clock Tab 210
Code Coverage Symbol 103
Colors 153
  Background 103
  Change 103
Column Settings 102
Common Problems 83, 164, 205
Complex Trigger Symbol 103
Concurrent Projects 73
Configuration Bits 96, 125
Configuration Bits Window 125
  Display 125
  FAQ 126
  Menu 125
Configure Menu 96
Configuring the Editor 152
Copy 91, 107, 108, 127, 157
Customer Notification Service 5
Customer Support 5
Cut 91, 127, 157
CVS 66

D

Debugger Menu 94, 207
Default Window Size 153
Delete 91, 120, 157
Desktop 89
Disassembly Window 108
Docking
  Toolbars 97
  Windows 102
Documentation
  Conventions 3
  Layout 1
dsPIC Core Model 177
dsPIC Language Tool 78
dsPIC30F Documents 4

E

Edit Menu 91
Editing In Place 102
Editor Color Customization 153
Editor Options 152
  Editor 152
  Sizes 153
  Text 152
Editor tab 152
Editor Window 126
Editors 81
EEPROM Window 115
  Display 115
  FAQ 116
  Menu 115
Error Messages 85
Export Hex File Dialog 133
Extended Microcontroller Mode 183
External Memory 182
  Configuration Bits 184, 185
  Interface 184
  Settings Dialog 185
External Memory Setting Dialog 133

F

FAQ 164, 205
Favorites, Help 60
File

Closing 155
Creating 154
Opening 154
Printing 154
Saving 155
File Commands 106
File Management Dialog 134
File Menu 89
File Registers Window 113
Display 113
FAQ 114
Menu 114
File Stimulus 200
Applying 202
Creating/Editing 200
Display 203
File Stimulus File 201
File Type Commands 106
File Window 126
Display 126
FAQ 128
Menu 127
Fill Memory/Registers Dialog 135
Filter Trace 161
Find 91
Find in Project Files 93
Find in Project Files Dialog 135
Find Next 91
Floating
Toolbars 97
Windows 102
Fonts 152
Change 103
Frequency of Processor 99

G

General Window Settings 103
Global Break Enable 99, 211
Go To 91

H

Hardware Stack Window 108
Display 109
FAQ 109
Menu 109
Help
Contents Tab 60
Favorites 60
Index Tab 60
Search Tab 60
Help Topics Dialog 136
HI-TECH Language Tools 80

I

IAR Language Tools 80
Import Dialog 136
In-Circuit Debuggers 82
In-Circuit Emulators 81
Indent 159
Index, Help 60
Integrated Tools 75
Internet Address 4

K

Keyboard Features 162

L

LCD Pixel Window 116
Display 116
FAQ 117
Menu 117
Limitations 85
Limitations Tab 210, 211
Line Numbers, Show 152

M

Match 159
Matching Braces 159
MCLR Pull-up Enabled 211
MEMCON Register 184
Memory
Modes 183
ROMless Devices 184
Menu Bar 89
Microchip Internet Web Site 4
Microchip Language Tools 77
Microcontroller Mode 183
Microprocessor Mode 183
Microsoft Visual Source Safe 66
Modes, Memory 183
Mouseover 152
Movement and Selection Keys 163
Moving to a Specific Line 157
Multiple Projects in a Single Workspace 71

N

National Language Code Page 152
New Project Dialog 136

O

Open Dialog 134
Osc/Trace Tab 208
Outdent 159
Output Window 107

P

Paste 91, 158
Pin Stimulus 197
Applying 198
Creating/Editing 197
Display 199
Print
Color 152
Line Numbers 152
Profile 207
Program Counter 99, 184
Program Loading 140
Program Memory Window 110
Display 110
FAQ 112
Menu 111
Programmers 82
Project Commands 105

Project Menu 92
Project Tree 104, 106
Project Window 68, 104
    Display 104
    FAQ 107
    Menues 104
Project Wizard Dialogs 137
Project-Display Preferences 137
Projects
    Active Project 72
    Creating/Updating 63
    Definition 61
    Project Wizard 62
    Setting Up/Changing 68
    Structure 64
Protected Microcontroller Mode 183
Pulse 103
PVCS 66

Q

Question Mark, Yellow 132
Quickbuild 73, 92, 93

R

Read Only Files 152
Reading, Recommended 4
Readme 4
Redo 91, 151, 159
Register Stimulus File 202
Replace 91

S

Save As 90
Save As Dialog 134
Save Project As Dialog 138
SCL Generator 187, 207
SCL Options Tab 209
Search, Help 60
Select All 91
Select Device Dialog 139
Select Language Toolsuite Dialog 139
Set Language Tool Location Dialog 139
Set Up Language Tools 132
Settings 95, 96
Settings Dialog 140
    PIC18X and dsPIC Devices 208
    PICmicro MCU's 210
Shortcut Keys 162
Simulator
    dsPIC DSC Model 177
    Execution 178
    Features 167
    PICmicro MCU Model 167
    Trace 182, 208
Simulators 81, 167
Single Assembly File 73
Single Project and Workspace 71
Sizes tab 153
Source Code Blocks, Delimiting 151
Source Safe 66
Special Character Keys 163
Special Function Registers Window 121
    Display 122
    FAQ 122
    Menu 122
Speed 178, 182, 208, 210
Stack 209, 211
Stack Return Address 109
Status Bar 99
Status Bits 99
Stimulus 187
    SCL Generator 187
    Stimulus Controller 194
Stimulus - PIC17 Devices 197
    File 200
    Pin 197
Stimulus Controller 194, 207
Stopwatch 181, 207
Symbols in Windows 103
Symbols, Breakpoint Dialog 132
Synchronous Stimulus File 201
Syntax Type 155

T

Tab size 153
Table of Contents, Help 60
Table Setup Dialog 143
Text
    Copying 157
    Cutting 157
    Deleting 151, 157
    Finding 158
    Formatting 160
    Indent 151
    Inserting 151
    Outdent 151
    Pasting 158
    Removing 157
    Replacing 159
    Selecting 151, 156
Text Mode 155
Text tab 152
Third Party Tools 82
Tool Menu 96
Toolbars 97
    Debug Buttons 40
    Docking 97
    Floating 97
Trace 208, 211
Trace Memory Window 123
    Display 123
    FAQ 124
    Menu 123
Trace/Pins Tab 211
Troubleshooting 164

U

UART1 I/O Tab 210
Undo 91, 151, 159
Unit Values 188
User ID Memory Dialog 143

V

Version-Control Dialog 144
Version-Control System 65
View Menu 92, 208
VSS 66

W

W Register 99
Watch Dialog 145
Watch Window 118
    Display 118
    FAQ 121
    Menu 120
Watchdog Timer 84, 209, 211
Window Menu 97
Windows
    Docking 102
    Floating 102
Wizard 62
Word Wrap 152
Workspaces, Definition 61
WWW Address 4

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://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Alpharetta, GA
Tel: 770-640-0034
Fax: 770-640-0307

Boston
Westford, MA
Tel: 978-692-3848
Fax: 978-692-3821

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

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

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

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

San Jose
Mountain View, CA
Tel: 650-215-1444
Fax: 650-961-0286

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8676-6200
Fax: 86-28-8676-6599

China - Fuzhou
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

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 - Shunde
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

China - Qingdao
Tel: 86-532-502-7355
Fax: 86-532-502-7205

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-2229-0061
Fax: 91-80-2229-0062

India - New Delhi
Tel: 91-11-5160-8631
Fax: 91-11-5160-8632

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

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Taiwan - Hsinchu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

EUROPE

Austria - Weis
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark - Ballerup
Tel: 45-4420-9895
Fax: 45-4420-9910

France - Massy
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Ismaning
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

England - Berkshire
Tel: 44-118-921-5869
Fax: 44-118-921-5820

10/20/04