



**MPLAB<sup>®</sup> IDE  
SIMULATOR, EDITOR  
USER'S GUIDE**

---

---

**Note the following details of the code protection feature on PICmicro® MCUs.**

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products 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 PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be 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 product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

---

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

#### **Trademarks**

The Microchip name and logo, the Microchip logo, KEELOQ, MPLAB, PIC, PICmicro, PICSTART and PRO MATE are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


FilterLab, microID, MXDEV, MXLAB, PICMASTER, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

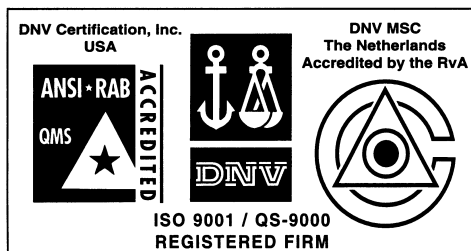
dsPIC, dsPICDEM.net, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICC, PICDEM, PICDEM.net, rPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated. Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.



*Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.*



---

---

## Table of Contents

---

---

### Preface

Introduction .....	1
Highlights .....	1
About This Guide .....	1
Warranty Registration .....	3
Recommended Reading .....	4
The Microchip Internet Web Site .....	5
Development Systems Customer Notification Service .....	6
Customer Support .....	8

# MPLAB<sup>®</sup> IDE User's Guide

---

---

## Part 1 - Getting Started with MPLAB IDE

### Chapter 1. MPLAB IDE Preview

1.1	Introduction .....	9
1.2	Highlights .....	9
1.3	What is MPLAB IDE .....	9
1.4	How MPLAB IDE Helps You .....	9
1.5	MPLAB IDE – An Integrated Development Environment (IDE) .....	10
1.6	MPLAB IDE Development Tools .....	11

### Chapter 2. MPLAB IDE Installation

2.1	Introduction .....	13
2.2	Highlights .....	13
2.3	Host Computer System Requirements .....	13
2.4	Obtaining the Program Files .....	13
2.5	Installing MPLAB IDE .....	14
2.6	Uninstalling MPLAB IDE .....	16

### Chapter 3. Getting Started with MPLAB IDE – A Tutorial

3.1	Introduction .....	17
3.2	Highlights .....	17
3.3	Setting up the Development Mode .....	18
3.4	Creating a Simple New Project .....	19
3.5	Creating a Simple New Source File .....	25
3.6	Entering Source Code .....	26
3.7	Assembling the Source File .....	27
3.8	Running Your Program .....	28
3.9	Opening Other Windows for Debugging .....	29
3.10	Using a Watch Window .....	29
3.11	Setting a Break Point .....	32
3.12	Summary .....	32

# Table of Contents

---

---

## Chapter 4. Projects Tutorial

4.1	Introduction .....	35
4.2	Highlights .....	35
4.3	Overview of Projects .....	35
4.4	Making a Project with One MPASM™ Assembler Source File .....	38
4.5	Compiling a Single MPASM Assembler Source File Without Creating a Project .....	45
4.6	Making a Project with Multiple MPASM Assembler Source Files using MPLINK™ Linker .....	49
4.7	Making a Project with Other Tools .....	58

## Part 2 - Using MPLAB IDE

### Chapter 1. MPLAB Editor

1.1	Introduction .....	59
1.2	Highlights .....	59
1.3	What is the MPLAB Editor .....	59
1.4	How MPLAB Editor Helps You .....	60
1.5	MPLAB Editor Features .....	60
1.6	MPLAB Editor Functions .....	61

### Chapter 2. Debugging and MPLAB SIM Simulator

2.1	Introduction .....	65
2.2	Highlights .....	65
2.3	MPLAB IDE Debugging Functions .....	65
2.4	Real-Time Program Execution .....	66
2.5	MPLAB SIM Simulator Environment .....	67
2.6	Simulator Considerations .....	68
2.7	Break and Trace Points .....	69
2.8	Conditional Break Dialog .....	75
2.9	Stimulus Functions .....	76

# MPLAB<sup>®</sup> IDE User's Guide

---

---

2.10	12-Bit Core Device Simulator Issues .....	86
2.11	14-Bit Core Device Simulator Issues .....	88
2.12	16-Bit Core Device Simulator Issues .....	93
2.13	Enhanced 16-Bit Core Device Simulator Issues .....	97

## Part 3 - MPLAB IDE Reference

### Chapter 1. MPLAB IDE Toolbar and Menu Options

1.1	Introduction .....	101
1.2	Highlights .....	101
1.3	MPLAB IDE Desktop .....	102
1.4	File Menu .....	104
1.5	Project Menu .....	117
1.6	Edit Menu .....	118
1.7	Debug Menu .....	129
1.8	Programmer Menu .....	149
1.9	Options Menu .....	152
1.10	Tools Menu .....	185
1.11	Window Menu .....	187
1.12	Help Menu .....	216

### Chapter 2. MPLAB IDE Toolbar and Status Bar Definitions

2.1	MPLAB IDE Toolbars .....	219
2.2	MPLAB IDE Status Bar .....	222

# Table of Contents

---

---

## **Appendix A. MPLAB IDE Key Mapping Functions**

A.1	Introduction .....	225
A.2	MPLAB IDE Key Mapping Functions .....	225

## **Appendix B. MPLAB Editor Default Key Commands**

B.1	Introduction .....	233
B.2	Highlights .....	233
B.3	Function Keys .....	233
B.4	Movement Keys .....	235
B.5	Control Keys .....	237
B.6	Formatting and Editing Keys .....	238

## **Appendix C. File Extensions Used by MPLAB IDE**

<b>Glossary</b> .....	241
Introduction .....	241
Highlights .....	241
Terms .....	241
<b>Index</b> .....	259
<b>Worldwide Sales and Service</b> .....	272

# MPLAB<sup>®</sup> IDE User's Guide

---

---



## Preface

---

---

### Introduction

This first chapter contains general information that will be useful to know before running MPLAB IDE.

### Highlights

The information you will garner from this chapter:

- About This Guide
- Warranty Registration
- Recommended Reading
- The Microchip Internet Web Site
- Development Systems Customer Notification Service
- Customer Support

### About This Guide

#### Document Layout

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

- **Part 1, Chapter 1: MPLAB IDE Preview** – An overview of what MPLAB IDE is and how it works.
- **Part 1, Chapter 2: MPLAB IDE Installation** – How to install MPLAB IDE on your computer.
- **Part 1, Chapter 3: Getting Started with MPLAB IDE – A Tutorial** – How to begin using MPLAB IDE.
- **Part 1, Chapter 4: MPLAB IDE Projects Tutorial** – A tutorial on how to use MPLAB IDE projects.
- **Part 2, Chapter 1: MPLAB Editor** – A discussion of the basic MPLAB Editor functions and features.
- **Part 2, Chapter 2: Debugging and MPLAB SIM Simulator** – A discussion of MPLAB IDE debugging functions and related MPLAB SIM simulator considerations.
- **Part 3, Chapter 1: MPLAB IDE Menu and Toolbar Options** – A description of the options available via the MPLAB IDE menus and toolbars. This chapter includes all menu options associated with the MPLAB Editor.

# MPLAB<sup>®</sup> IDE User's Guide

---

- **Part 3, Chapter 2: MPLAB IDE Toolbar and Status Bar Definitions** – Identifies each MPLAB IDE Toolbar button and its function, and discusses how to interpret the information displayed on the MPLAB IDE Status Bar.
- **Appendix A: MPLAB IDE Key Mapping Functions** – Lists the available MPLAB IDE key mapping functions.
- **Appendix B: MPLAB Editor Default Key Commands** – Describes the default key commands specific to the MPLAB Editor and lists the equivalent menu command (if any).
- **Appendix C: File Extensions Used by MPLAB IDE** – Lists the types of files that MPLAB IDE uses and identifies each file type's default extension.
- **Glossary** – A glossary of terms used in this guide.
- **Index** – Cross-reference listing of terms, features and sections of this document.
- **Worldwide Sales and Service** – A listing of Microchip sales and service locations and telephone numbers worldwide.

## Conventions Used in this Guide

This manual uses the following documentation conventions:

**Table: Documentation Conventions**

Description	Represents	Examples
Code (Courier font):		
Plain characters	Sample code Filenames and paths	<code>#define START</code> <code>c:\autoexec.bat</code>
Angle brackets: < >	Variables	<code>&lt;label&gt;</code> , <code>&lt;exp&gt;</code>
Square brackets [ ]	Optional arguments	<code>MPASMWIN</code> <code>[main.asm]</code>
Curly brackets and pipe character: {   }	Choice of mutually exclusive arguments An OR selection	<code>errorlevel {0 1}</code>
Lower case characters in quotes	Type of data	<code>"filename"</code>
Ellipses...	Used to imply (but not show) additional text that is not relevant to the example	<code>list</code> <code>["list_option...</code> <code>, "list_option"]</code>
0xnnn	A hexadecimal number where n is a hexadecimal digit	<code>0xFFFF</code> , <code>0x007A</code>

**Table: Documentation Conventions (Continued)**

<b>Description</b>	<b>Represents</b>	<b>Examples</b>
Italic characters	A variable argument; it can be either a type of data (in lower case characters) or a specific example (in uppercase characters).	<code>char isascii (char, ch);</code>
Interface (Helvetica font):		
Underlined, italic text with right arrow	A menu selection from the menu bar	<i><u>File &gt; Save</u></i>
Bold characters	A window or dialog button to click	<b>OK, Cancel</b>
Characters in angle brackets < >	A key on the keyboard	<Tab>, <Ctrl-C>
Documents (Helvetica font):		
Italic characters	Referenced books	<i>MPLAB IDE User's Guide</i>

## Documentation Updates

All documentation becomes dated, and this user's guide is no exception. Since MPLAB IDE and other Microchip tools are constantly evolving to meet customer needs, some MPLAB IDE dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site at <http://www.microchip.com> to obtain the latest documentation available.

## Warranty Registration

Please complete the enclosed Warranty Registration Card and mail it promptly. Sending in your Warranty Registration Card entitles you to receive new product updates. Interim software releases are available at the Microchip web site.

# MPLAB<sup>®</sup> IDE User's Guide

---

## Recommended Reading

This user's guide describes how to use MPLAB IDE. The user may also find the data sheets for specific microcontroller devices informative in developing firmware.

### **README.LAB**

For the latest information on using MPLAB IDE, read the README.LAB file (an ASCII text file) on the MPLAB IDE CD-ROM. README.LAB contains update information that may not be included in the *MPLAB IDE User's Guide*.

### **README.XXX**

For the latest information on using other tools, refer to an information file about the product that is more current than the printed manual. Check the MPLAB IDE folder for other README files. (In the case of MPASM assembler, for instance, the file is called README.ASM.)

### **Microchip Technology Library CD-ROM (DS00161)**

This CD-ROM contains comprehensive application notes, data sheets, and technical briefs for all of Microchip products. To obtain this CD-ROM, contact the nearest Microchip Sales and Service location (see back page).

### **Embedded Control Handbook Vol.1 & 2 and the Embedded Control Update 2000 (DS00092, DS00167, and DS00711)**

These handbooks contain a wealth of information about microcontroller applications. To obtain these documents, contact the nearest Microchip Sales and Service location (see back page).

The application notes described in these manuals are also obtainable from Microchip Sales and Service locations or from the Microchip web site (<http://www.microchip.com>).

### **Microsoft Windows<sup>®</sup> Manuals**

This manual assumes that users are familiar with the Microsoft Windows operating system. Many excellent references exist for this software, and should be consulted for general operation of Windows.

### The Microchip Internet Web Site

Microchip provides on-line support on the Microchip World Wide Web (WWW) site.

The web site is used by Microchip as a means to make files and information easily available to customers. To view the site, the user must have access to the Internet and a web browser, such as Netscape<sup>®</sup> Communicator or Microsoft<sup>®</sup> Internet Explorer<sup>®</sup>. Files are also available for FTP download from our FTP site.

### Connecting to the Microchip Internet Web Site

The Microchip web site is available by using your favorite Internet browser to connect to:

**[www.microchip.com](http://www.microchip.com)**

The file transfer site is available by using an FTP service to connect to:

**<ftp://ftp.microchip.com>**

The web site and file transfer site provide a variety of services. Users may download files for the latest Development Tools, Data Sheets, Application Notes, User's Guides, Articles, and Sample Programs. A variety of Microchip specific business information is also available, including listings of Microchip sales offices, distributors and factory representatives. Other data available for consideration includes:

- Latest Microchip Press Releases
- Technical Support Section with Frequently Asked Questions
- Design Tips
- Device Errata
- Job Postings
- Microchip Consultant Program Member Listing
- Links to other useful web sites related to Microchip Products
- Web conferences for products, Development Systems, and technical information
- Listing of seminars and events

## Development Systems Customer Notification Service

Microchip provides a customer notification service to help our customers keep current on Microchip products with the least amount of effort. Once you subscribe, you will receive e-mail notification whenever we change, update, revise or have errata related to that product family or development tool you have selected.

Click on the Customer Change Notification link on the Microchip web site at [www.microchip.com](http://www.microchip.com). From there you may register and select the product groups and product categories about which you want to receive notifications.

The Development Systems product categories are:

- Compilers
- Emulators
- In Circuit Debuggers
- MPLAB
- Programmers

You may return to this location on the web site at any time to change your selected product categories or your profile information. You may also review product change notification documents for all categories that have been posted.

### Compilers

The latest information on Microchip C compilers, Linkers and Assemblers. These include the MPLAB C17 and MPLAB C18 C Compilers, MPLINK<sup>™</sup> Linker, MPASM<sup>™</sup> Assembler as well as the Librarian, MPLIB<sup>™</sup> for MPLINK Linker.

### Emulators

The latest information on Microchip emulators. These include the MPLAB ICE 2000 in-circuit emulator and PICMASTER<sup>®</sup> emulator.

### In-Circuit Debuggers

The latest information on Microchip in-circuit debuggers, such as the MPLAB ICD.

## MPLAB

The latest information on MPLAB IDE, the Windows Integrated Development Environment for development systems tools. This list is focused on MPLAB IDE, MPLAB SIM Simulator, MPLAB IDE Project Manager and general editing and debugging features. For specific information on MPLAB IDE compilers, linkers and assemblers, subscribe to the COMPILERS list. For specific information on MPLAB IDE emulators, subscribe to the EMULATORS list. For specific information on MPLAB IDE device programmers, please subscribe to the PROGRAMMERS list.

## Programmers

The latest information on Microchip PICmicro<sup>®</sup> microcontroller (MCU) device programmers. These include PRO MATE<sup>®</sup> II and PICSTART<sup>®</sup> Plus.

## Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Corporate Applications Engineer (CAE)
- Hotline

Customers should call their distributor, representative, or field application engineer (FAE) for support. Local sales offices are also available to help customers. See the back cover for a listing of sales offices and locations.

Corporate applications engineers (CAEs) may be contacted at (480) 792-7627.

In addition, there is a Systems Information and Upgrade Line. This line provides system users a listing of the latest versions of all of Microchip's development systems software products. Plus, this line provides information on how customers can receive any currently available upgrade kits.

The Hotline Numbers are:

1-800-755-2345 for U.S. and most of Canada, and

1-480-792-7302 for the rest of the world.



---

---

## Chapter 1. MPLAB IDE Preview

---

---

### 1.1 Introduction

This chapter will give an overview of MPLAB IDE.

### 1.2 Highlights

In this chapter, we discuss:

- What is MPLAB IDE
- How MPLAB IDE Helps You
- MPLAB IDE – An Integrated Development Environment (IDE)
- MPLAB IDE Development Tools

### 1.3 What is MPLAB IDE

MPLAB IDE is a Windows<sup>®</sup>-based Integrated Development Environment (IDE) for the Microchip Technology Incorporated PICmicro<sup>®</sup> microcontroller (MCU) families. MPLAB IDE allows you to write, debug, and optimize PICmicro MCU applications for firmware product designs. MPLAB IDE includes a text editor, simulator, and project manager. MPLAB IDE also supports the MPLAB ICE 2000 emulator, MPLAB ICD debugger, PICSTART<sup>®</sup> Plus and PRO MATE<sup>®</sup> II programmers, and other Microchip or third party development system tools.

### 1.4 How MPLAB IDE Helps You

The organization of MPLAB IDE tools by function makes pull-down menus and customizable quick keys easy to find and use. MPLAB IDE tools allow you to:

- Assemble, compile, and link source code
- Debug the executable logic by watching program flow with the simulator, or in real-time with the MPLAB ICE 2000 emulator or MPLAB ICD debugger
- Make timing measurements
- View variables in watch windows
- Program firmware with PICSTART Plus or PRO MATE II programmers
- Find quick answers to questions from the MPLAB IDE on-line Help

and much more.

## 1.5 MPLAB IDE – An Integrated Development Environment (IDE)

MPLAB IDE is an easy-to-learn and use Integrated Development Environment (IDE). The IDE provides firmware development engineers the flexibility to develop and debug firmware for Microchip's PICmicro MCU families. MPLAB IDE runs under Microsoft Windows 3.1x and higher.

MPLAB IDE provides functions that allow you to:

- Create and edit source files
- Group files into projects
- Debug source code
- Debug executable logic using the simulator or emulator(s)

MPLAB IDE allows you to create and edit source code by providing you with a full-featured text editor.

Further, you can easily debug source code with the aid of a Build Results window that displays the errors found by the compiler, assembler, and linker when generating executable files.

A Project Manager allows you to group source files, precompiled object files, libraries, and linker script files into a project format.

MPLAB IDE also provides feature-rich simulator and emulator environments to debug the logic of executables. Some of the features are:

- A variety of windows allowing you to view the contents of all data and program memory locations
- Source Code, Program Memory, and Absolute Listing windows allowing you to view the source code and its assembly-level equivalent separately and together (Absolute Listing)
- The ability to step through execution, or apply Break, Trace, Standard, or Complex Trigger Points

## 1.6 MPLAB IDE Development Tools

MPLAB IDE integrates several tools to provide a complete development environment.

- **MPLAB Project Manager**

Use the Project Manager to create a project and work with the specific files related to the project. When using a project, you can rebuild source code and download it to the simulator or emulator with a single mouse click.

- **MPLAB Editor**

Use the MPLAB Editor to create and edit text files such as source files, code, and linker script files.

- **MPLAB ICD In-Circuit Debugger**

The MPLAB ICD In-Circuit Debugger is a powerful, low-cost development and evaluation kit for many PICmicro MCU FLASH devices.

- **MPLAB SIM Simulator**

The software simulator models the instruction execution and I/O of the PICmicro MCUs.

- **MPLAB ICE 2000 In-Circuit Emulator**

The MPLAB ICE 2000 emulator uses hardware to provide real-time emulation of PICmicro MCUs, either with or without a target system.

- **MPASM™ Assembler/MPLINK™ Linker/MPLIB™ Librarian**

The MPASM assembler allows source code to be assembled without leaving MPLAB IDE. The MPLINK linker creates the final application by linking relocatable modules from MPASM, MPLAB C17 and MPLAB C18 C Compilers. The MPLIB librarian manages custom libraries for maximum code reuse.

- **MPLAB CXX C Compilers**

The MPLAB C17 and MPLAB C18 C Compilers provide ANSI-based high level source code solutions. Complex projects can use a combination of C and assembly source files to obtain the maximum benefits of speed and maintainability.

- **PRO MATE® II and PICSTART® Plus Programmers**

Develop code with the simulator or an emulator, assemble or compile it, then use one of these tools to program devices. This can all be accomplished with MPLAB IDE. Although the PRO MATE II programmer does not require MPLAB IDE to operate, programming is easier using MPLAB IDE.

- **PICMASTER® and PICMASTER CE Emulators**

MPLAB IDE provides legacy support for the PICMASTER and PICMASTER CE emulators.

# MPLAB<sup>®</sup> IDE User's Guide

---

- **Third Party Tools**

Many other companies have development tools for Microchip products that work with MPLAB IDE. Consult the *Microchip Third Party Guide* (DS00104).

---

---

## Chapter 2. MPLAB IDE Installation

---

---

### 2.1 Introduction

This chapter describes the procedure for installing MPLAB IDE.

### 2.2 Highlights

The items discussed in this chapter include:

- Host Computer System Requirements
- Obtaining the Program Files
- Installing MPLAB IDE
- Uninstalling MPLAB IDE

### 2.3 Host Computer System Requirements

The following minimum configuration is required to run MPLAB IDE:

- PC-compatible Pentium<sup>™</sup>-class system
- Microsoft Windows 3.1x or higher
- 16 MB memory (32 MB recommended)
- 45 MB of hard disk space

**Note:** Not all hardware components that function under MPLAB IDE, such as emulators and device programmers, function under all current Windows operating systems. Refer to the user's guide of the specific hardware device for details.

### 2.4 Obtaining the Program Files

The MPLAB IDE application is shipped with every Microchip Development System. Also, MPLAB IDE may be obtained by contacting any Microchip sales office and requesting the Technical Library CD-ROM or by downloading the files from the Microchip web site ([www.microchip.com](http://www.microchip.com)).

The number and names of the files vary depending on the version. Version 5.00 of MPLAB IDE, for example, would have these files:

MP50000.EXE  
MP50000.W02  
MP50000.W03  
MP50000.W04  
MP50000.W05  
MP50000.W06  
MP50000.W07

# MPLAB<sup>®</sup> IDE User's Guide

---

## 2.5 Installing MPLAB IDE

The executable file MPvvvvv.EXE installs the Microchip MPLAB Integrated Development Environment (IDE), where vvvvv is the version number of MPLAB IDE.

To install MPLAB IDE, follow these steps:

1. Enter Microsoft Windows.
2. If you are installing from the MPLAB IDE CD-ROM, place the CD-ROM into the drive now.
3. Execute the installation program:

**Windows 3.1:** From the File Manager, or from the *Program Manager > Run* option, run **X:\MPvvvvv.exe**, where **X** is the drive designation of the install files and **vvvvv** is the version of MPLAB IDE you are installing. For example, enter **d:\MP50000.exe** to install version 5.00 of MPLAB IDE, where d: is the CD-ROM drive that contains the MPLAB IDE install.

**Windows 95/98/ME, Windows NT<sup>®</sup>, or Windows 2000:** If the CD starts up automatically, follow the prompts given. Otherwise, click the **Start** button and select **Run**. Enter **X:\MPvvvvv.exe**, where **X** is the drive designation of the install files and **vvvvv** is the version of MPLAB IDE you are installing. For example, enter **d:\MP50000.exe** to install version 5.00 of MPLAB IDE, where d: is the CD-ROM drive that contains the MPLAB IDE install. Then click **OK**.

<p><b>Note:</b> Windows NT and Windows 2000 users must have administrative privileges in order to install MPLAB IDE.</p>
--

4. Step through the displayed dialogs where you may customize your MPLAB IDE installation. If you are unsure about any of the options displayed on the dialogs, simply accept the defaults as shown.

### Installation Tips:

#### Selecting the Components

If you have a limited amount of PC memory and you have not purchased a device programmer or emulator, you can install just the software tools:

- MPLAB IDE files
- MPASM Assembler/MPLINK Linker/MPLIB Librarian files
- MPLAB SIM Simulator Support Files
- Help Files

You can reinstall MPLAB IDE later to add additional components.

#### Selecting the Destination Folder

We recommend installing MPLAB IDE on a local hard drive rather than a network drive.

# MPLAB IDE Installation

## Selecting the System Files Folder

Installing the data link libraries (DLLs) to the \windows\System folder may allow better management and prevent a future installation of another version from overwriting these MPLAB IDE files.

5. Watch as your MPLAB IDE files are installed on your system. View any displayed screens for new product information.
6. View the readme files. The README files contain valuable information on new features as well as limitations and known problems.

**Note:** If you select No when asked if you want to view the README files, you may view these files later from the MPLAB install folder. It is recommended that you consult the README files before contacting technical support.

7. Start MPLAB IDE by executing MPLAB.EXE or clicking on the MPLAB IDE icon. You will see the MPLAB IDE desktop as shown in Figure 2.1.

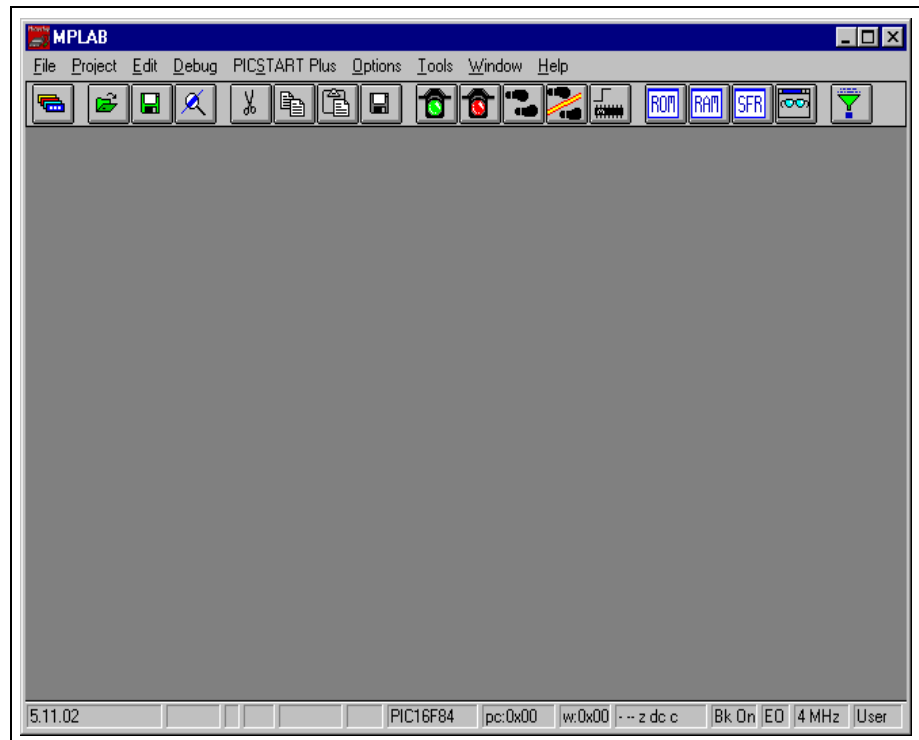


Figure 2.1: MPLAB IDE Desktop

## 2.6 Uninstalling MPLAB IDE

To uninstall MPLAB IDE, open Windows Explorer (or File Manager for Windows 3.1) and double-click on `unwise32.exe` (or `unwise.exe` if you are using Windows 3.1) in the MPLAB folder to run it. Based on the log file created during MPLAB IDE installation, the `unwise.exe` uninstall program determines which files to remove from the MPLAB IDE, Windows, and System folders.



---

---

## Chapter 3. Getting Started with MPLAB IDE – A Tutorial

---

---

### 3.1 Introduction

This tutorial is intended to be a quick introduction to the MPLAB IDE user interface. It should take about 1 to 2 hours to complete the tutorial.

This is not intended to discuss all of the details of MPLAB IDE, only to provide a beginning understanding so you can use MPLAB IDE right away.

### 3.2 Highlights

In this tutorial, you will learn about:

- Setting up the Development Mode
- Creating a Simple New Project
- Creating a Simple New Source File
- Entering Source Code
- Assembling the Source File
- Running Your Program
- Opening Other Windows for Debugging
- Creating a Watch Window
- Saving the Watch Window
- Setting a Break Point

In addition, there is an overview of other features to be discussed in later chapters.

With the operation of MPLAB IDE provided by this tutorial, you should be able to:

- Become familiar with the MPLAB IDE Desktop
- Create a new assembly source code file and enter it into a new project for the PIC16F84
- Identify and correct simple errors
- Run the built-in simulator
- Set break points
- Create Watch windows
- Become familiar with the various debugging windows

# MPLAB<sup>®</sup> IDE User's Guide

---

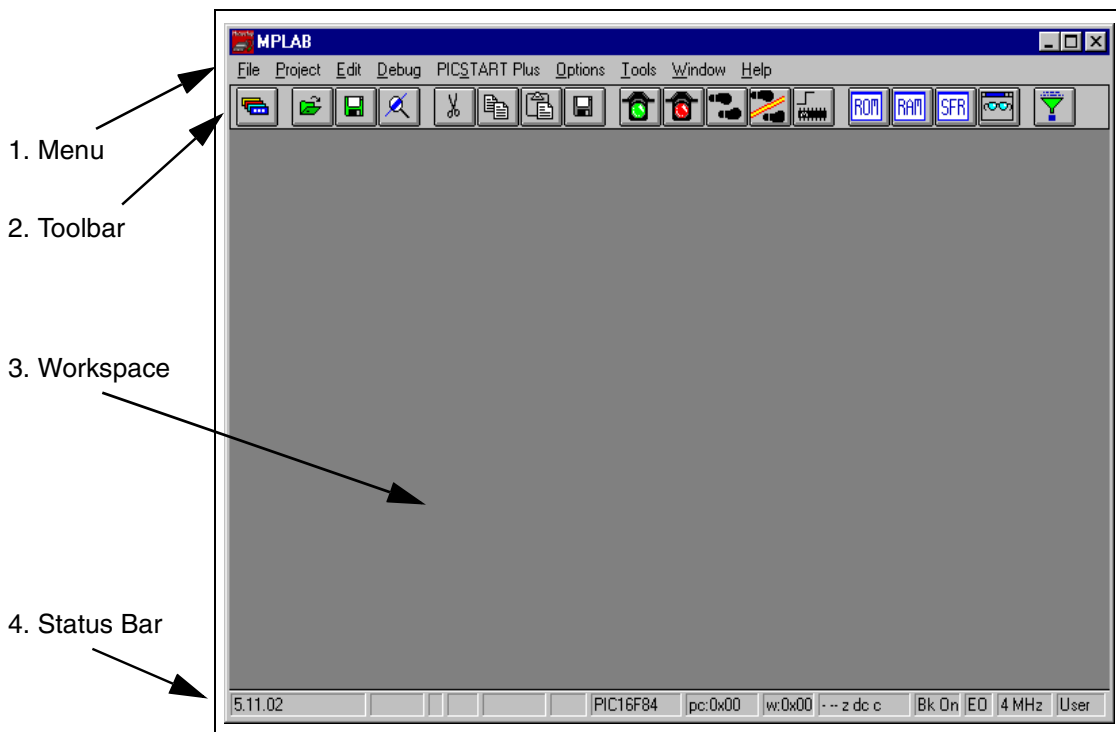
## 3.3 Setting up the Development Mode

The previous chapter discussed how to install MPLAB IDE. Now you will begin setting up the application.

The MPLAB IDE desktop (Figure 3.1) contains the following major elements:

1. A menu across the top line
2. A toolbar below the menu
3. A workspace in which various files, windows, and dialogs can be displayed
4. A status bar at the bottom

Notice that the status bar includes information about how the system is currently configured. We'll cover some of these features in more detail later. For now, let's see how to set the development mode.



**Figure 3.1: MPLAB IDE Desktop**

The development mode sets which tool, if any, will execute code. For this tutorial we will use MPLAB SIM, the software simulator. Later you may switch to one of the emulator operations if you have an emulator. Operation will be similar. "Editor Only" mode does not allow code execution, and is mainly useful if you have not installed the simulator, do not have an emulator, and are just creating code to program a PICmicro microcontroller (MCU).

Select the *Options > Development Mode* menu item and click the **Tools** tab to select the development tool and processor for your project.

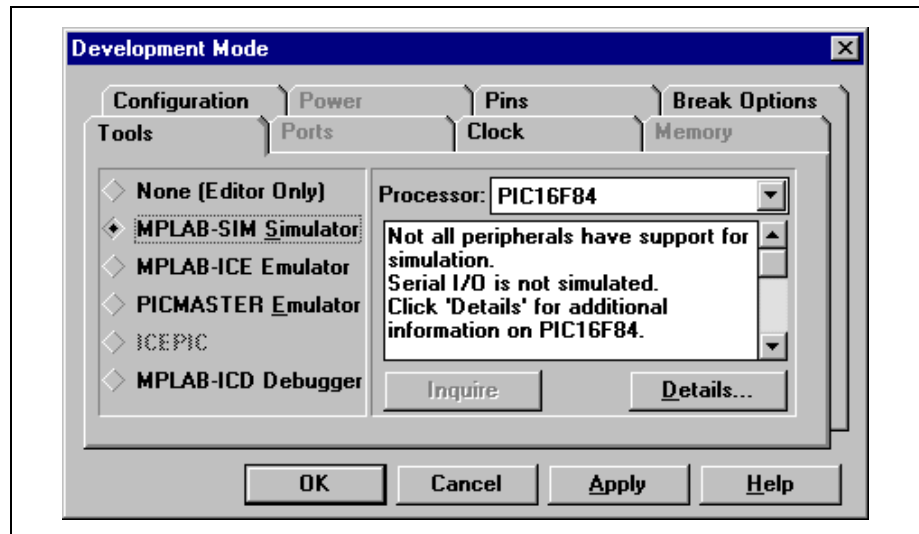


Figure 3.2: Development Mode Tools Dialog

MPLAB IDE is a constantly evolving product, so there may be subtle differences between what you see and the picture here. Select MPLAB SIM Simulator and choose the PIC16F84 from the pull-down list of available processors supported by the simulator. Click **OK**. The simulator will initialize and you should see “PIC16F84” and “Sim” in the status bar on the bottom of the MPLAB IDE desktop. You are now in simulator mode for the PIC16F84 device.

## 3.4 Creating a Simple New Project

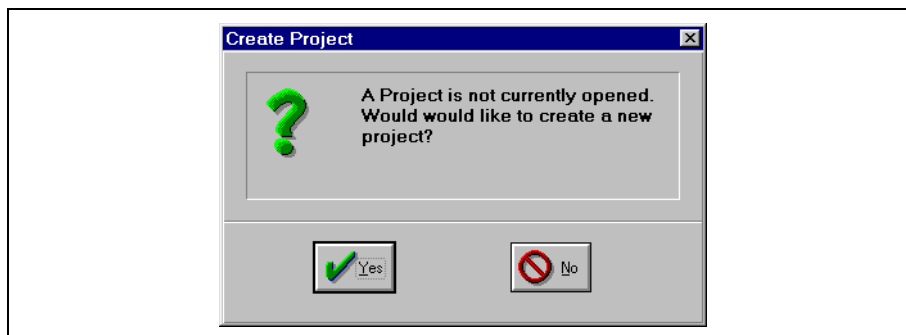
The simulator runs from the same file (a HEX file) that can be programmed into the PICmicro MCU. For the simulator to run you must first create a source code file and successfully assemble the source code.

The assembler produces, among other things, a HEX file. This file has the file extension `.HEX`. In this tutorial the file will be named `tutor84.hex`. Later this file can be loaded directly into a device programmer without using the assembler or an MPLAB IDE project. This file can also be loaded by most other third party programmers.

Select *File > New* from the menu and you will see a dialog that looks like Figure 3.3.

# MPLAB<sup>®</sup> IDE User's Guide

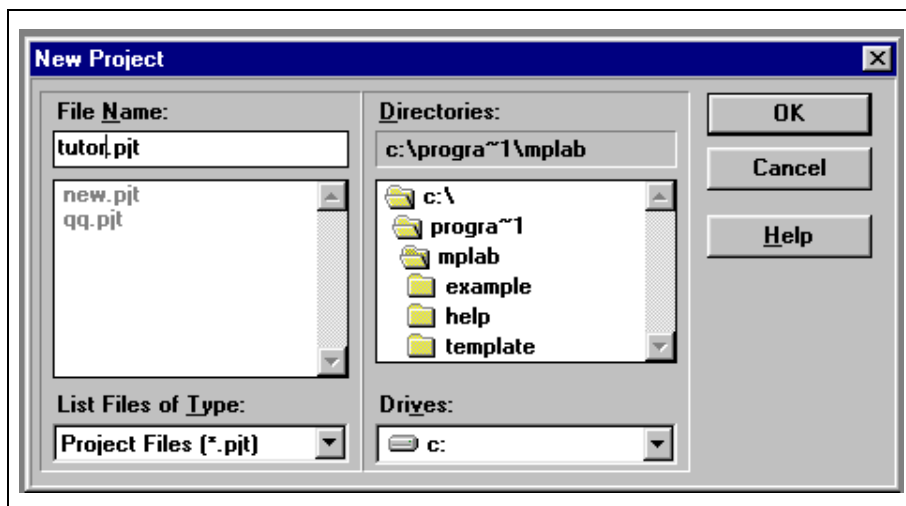
---



**Figure 3.3: Create Project Dialog**

Click **Yes**, and a standard Windows browsing dialog will appear (Figure 3.4). In this dialog, indicate the location where you want your project stored. Remember where you put it. You'll need this information later. This tutorial uses a folder in `c:\Program Files\MPLAB` and creates the project file named `tutor84.pjt`.

"PJT" is a standard suffix for MPLAB IDE project files. The prefix of the project file name, in this case `tutor84`, will become a default prefix for many of the files that MPLAB IDE will use or create for this tutorial.



**Figure 3.4: New Project Dialog**

Using the mouse to click **OK** will bring you to the Edit Project Dialog (Figure 3.5).

The simulator, programmers, and emulator systems that work with MPLAB IDE use a HEX file created by assembling, compiling, and/or linking source code. Several different tools can create HEX files, and these tools are part of each project. Projects give you flexibility to describe how the application will be built and which software tools will be used to create the `.HEX` file. We will

# Getting Started with MPLAB IDE – A Tutorial

not get into these details in this tutorial, but as you need these features you can use the Node Properties to set them. See Chapter 4 for information on more complex projects.



**Figure 3.5: Edit Project Dialog – Node Properties Enabled**

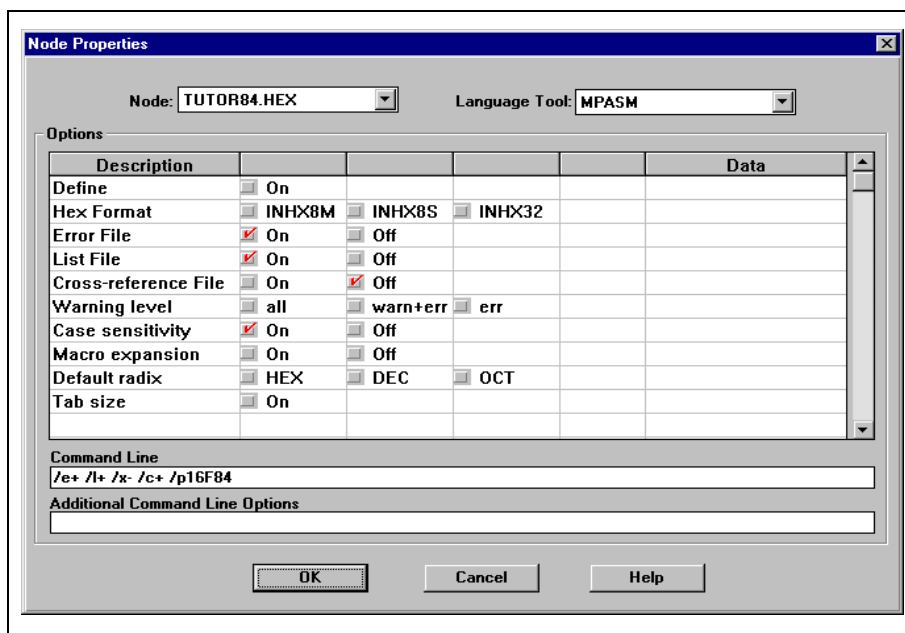
Notice that the target file name of the Edit Project dialog has been filled in for you. It uses the development mode that we set previously and defaults to using the Microchip language tool suite.

In addition, the default language suite, paths, and nodes for all projects are set by selecting *Options > Environment Setup* and clicking the **Projects** tab. These defaults appear in the Edit Project dialog for all new projects.

In the Project Files window, you will find `tutor84.[hex]`. Highlighting this name will cause the Node Properties button to become usable.

Before doing anything else, we must tell MPLAB IDE how to create the HEX file. Do this by clicking the **Node Properties** button. The Node Properties dialog will appear (Figure 3.6).

# MPLAB<sup>®</sup> IDE User's Guide



**Figure 3.6: Node Properties Dialog**

This dialog contains all of the default settings for the language tool shown in the upper right of the dialog (in this case MPASM). In the simplest form, a project contains a HEX file created from one assembly source file. This is the default as the Node Properties dialog appears.

You can see that there are a lot of rows and columns on this dialog. Each row usually corresponds to a “switch” – those things that are often set on the command line when the tool is invoked. In fact, the setting of these switches is reflected in the Command Line window near the bottom. This is the actual command line that will be issued to the MPASM assembler when it is invoked from MPLAB IDE.

For now you can use the default settings for all other entries, but as you become more familiar with building an application, you will probably find that you’ll want to change some of these.

Click **OK** to apply these defaults, return to the Edit Project dialog and enable the **Add Node** button (Figure 3.7).

# Getting Started with MPLAB IDE – A Tutorial

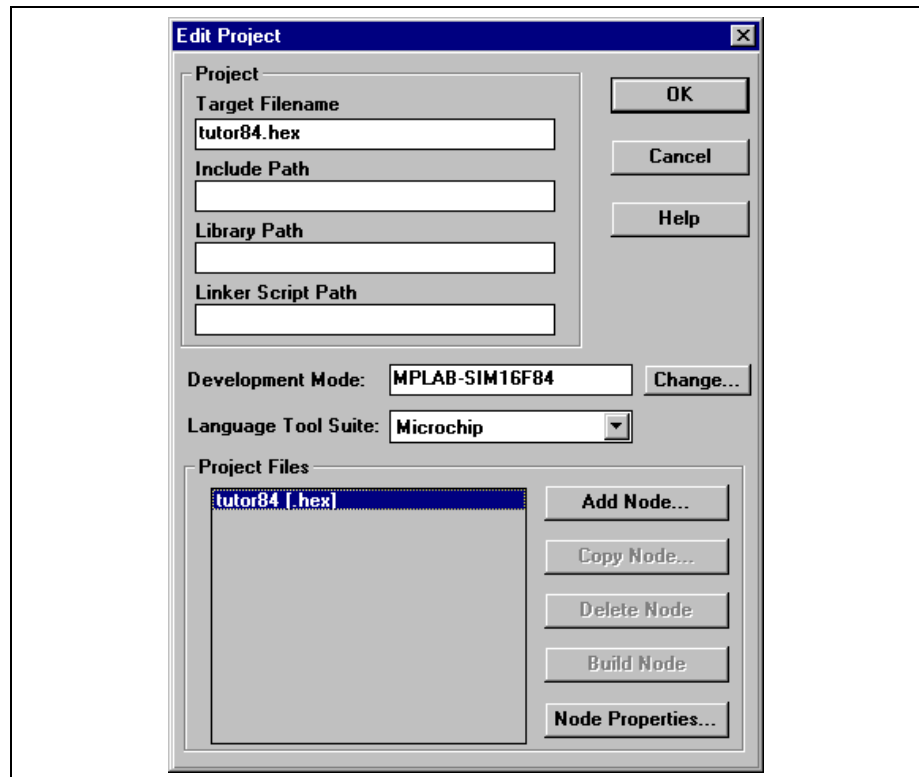


Figure 3.7: Edit Project Dialog – Add Node Enabled

Click **Add Node**. You will see the standard Windows browse dialog (Figure 3.8), and the working folder will be the same as the project folder. Enter the file name `tutor84.asm` and click **OK**.

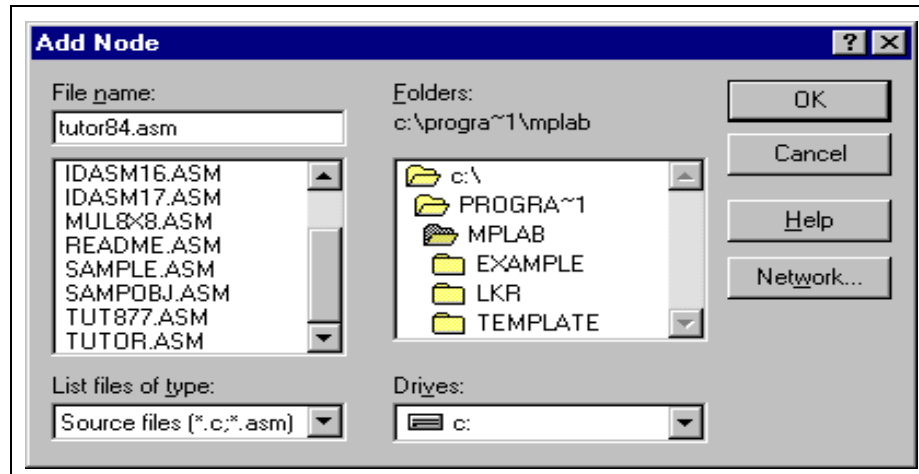
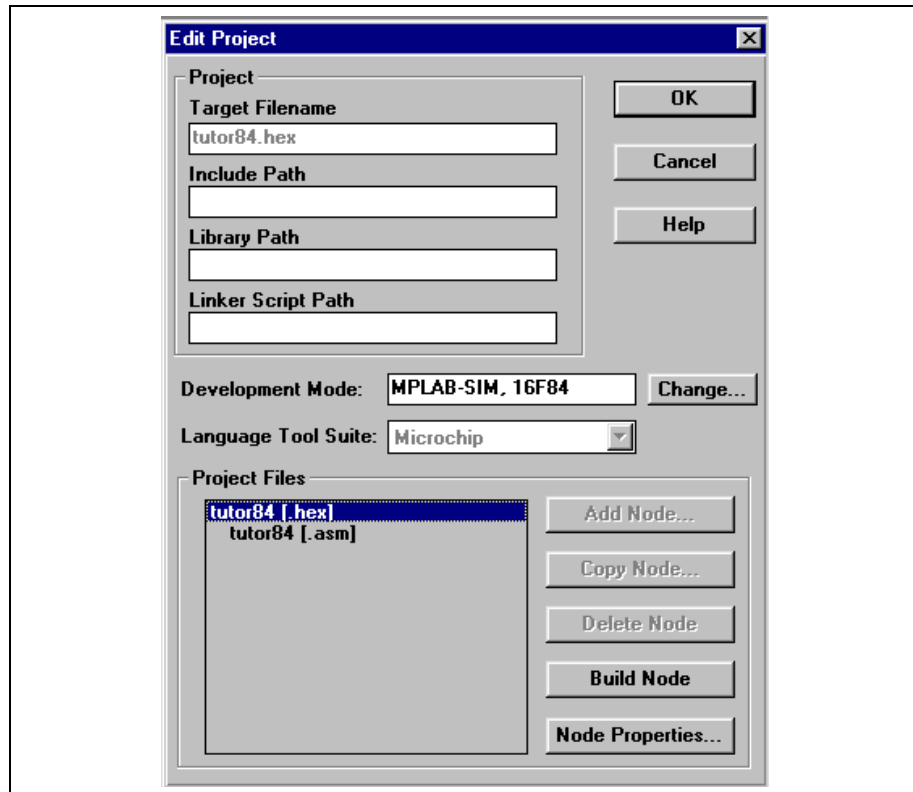


Figure 3.8: Add Node Dialog

# MPLAB® IDE User's Guide

---

You will return to the Edit Project dialog and should see `tutor84.asm` indented and below the HEX file, indicating that it is a contributing node (Figure 3.9).



**Figure 3.9: Edit Project Dialog – Node Added**

Clicking **OK** takes you back to the MPLAB IDE desktop with an open and unnamed source code file.



## 3.5 Creating a Simple New Source File

Click once in the blank space of the empty file window that has been created for you. It is probably titled Untitled. This gives the window focus. Use the *File > Save As* menu option and save the empty file as `tutor84.asm`. When the standard browse dialog opens, you will find `tutor84.asm` located in the current working folder of the project. Enter the file name and click **OK**.

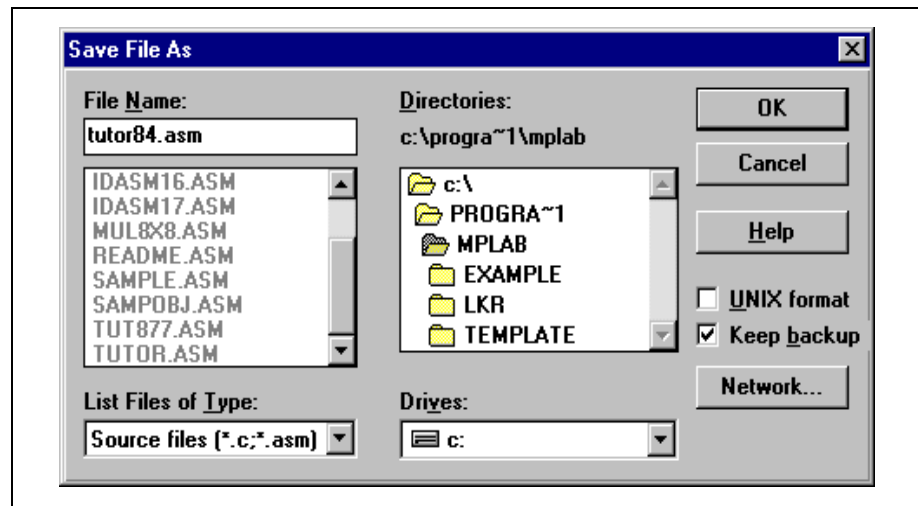


Figure 3.10: Save Source File

You will now be presented with the MPLAB IDE desktop and the empty file window, but the name of the file window will reflect its new name.

The name of the source file and the name of the project (`tutor84` in this tutorial) must be the same in this kind of project. If you change the name of the source file, you must also change the name of the project to match. Other projects that use the linker allow the output file name to be different from the input file. Section 4.6 provides a tutorial on creating projects using the linker.

**Note:** For a single source file project, the MPASM assembler creates its output HEX file with the same name as the source file. The project name, HEX file, and source file **MUST** have the same name.

## 3.6 Entering Source Code

Use the mouse to locate the cursor at the beginning of the `tutor84.asm` empty file window, and enter the following text, exactly as written, on each line. You don't have to enter the comments (the text following the semicolons).

```
list    p=16f84
include <p16F84.inc>
c1 equ  0x0c    ; Set temp variable counter c1 at address 0x0c

    org    0x00    ; Set program memory base at reset vector 0x00
reset
    goto   start   ; Go to start of the main program

    org    0x04    ; Set program memory base to beginning of user code
start
    movlw  0x09    ; Initialize counter to arbitrary value greater than zero
    movwf  c1      ; Store value in temp variable a defined above
loop
    incfsz c1,F    ; Increment counter, place results in file register
    goto   loop    ; Loop until counter overflows

    goto   bug     ; When counter overflows, got to start to re-initialize
end
```

This code is a very simple program that increments a counter and resets to a predetermined value when the counter rolls over to zero.

All labels start in the first column, and the last line has an `end` directive. Refer to the *MPASM User's Guide with MPLINK and MPLIB* (DS33014) for more information about directives. The PICmicro MCU data sheets contain full information about instructions along with examples of their use.

Save the file by using the *File > Save* menu item.

## 3.7 Assembling the Source File

Assembling the file can be accomplished in several ways. The method described here uses the *Project > Build All* menu item. This will execute the MPASM assembler in the background using the defaults saved with the project as noted before. Once the assembly process is complete, the Build Results window will appear (Figure 3.11).

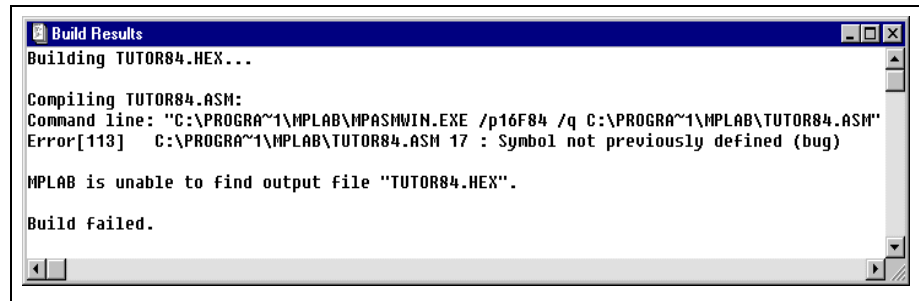


Figure 3.11: Build Results Window – Build Failed

You have intentionally entered at least one error if you entered the code as written in Section 3.6. The last `goto` in the program references a nonexistent label called `bug`. Since this label has not been defined before, the assembler reports an error. You may have other errors as well.

Using the mouse, double-click on the error message. This will bring the cursor to the line in the source code that contains the error. Change `bug` to `start`. Use the Build Results window to help find the errors, and repair any other bugs in your source code. Reassemble by executing the *Project > Build All* menu function. This process may take a couple of iterations.

**Note:** Whenever you rebuild a project all of your source files will be saved to disk.

When you've fixed all problems in the source code, the Build Results window will display "Build completed successfully" (Figure 3.12). You now have a complete project that can be executed using the simulator.



Figure 3.12: Build Results Window – Build Successful

## 3.8 Running Your Program

Use the *Debug > Run > Reset* to initialize the system. The program counter will be reset to zero, which is the reset vector on the PIC16F84. The source code line at this address may be highlighted with a dark bar. Also, you may notice that PC is set to 0x00 in the status bar at the bottom of MPLAB IDE.

Use the *Debug > Run > Step* menu item. This causes the program counter to advance to the next instruction location. The dark bar will follow the source code and the program counter displayed in the status bar should advance to “pc:0x04.”

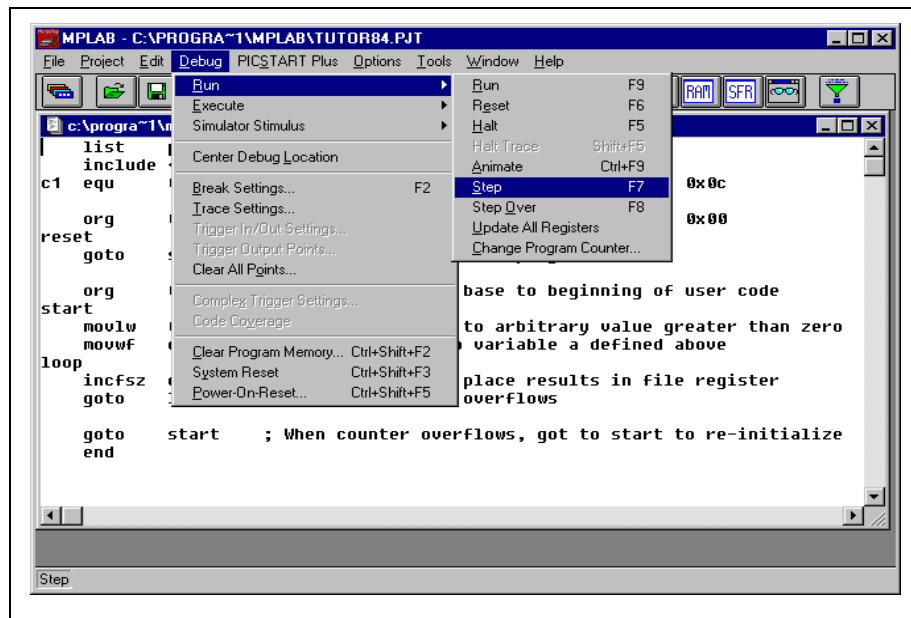


Figure 3.13: *Debug > Run > Step* Menu Item

You may notice as you execute the *Debug > Run > Step* menu item that there is text on the right side of the menu item that says <F7>. This stands for “function key seven” on your keyboard. Many MPLAB IDE functions are assigned to shortcut keys. These keys have the same effect as executing the menu item itself. Press <F7> a few times and watch the program counter and dark bar advance through the program.

Execute the *Debug > Run > Run* menu item or press <F9> to start the program running from the current location counter. The status bar will change colors indicating the program is executing instructions. None of the other fields on the status bar will be updated until the program is halted.

Stop the program by executing the *Debug > Run > Halt* menu item or by pressing <F5>. The status bar will change back to its original color, and the current program counter and other status information will be updated.

Another way to execute functions is to use the toolbar at the top of the screen. If you place the cursor over the items in the toolbar, you can see the name of the function in the status bar at the bottom. The left button is a standard **Swap Toolbar** button that allows you to scroll through the available toolbars. Toolbars can be customized (see Section 1.9.5.1.2). On the debug toolbar, the green light is equivalent to <F9> (Run) and the red light is the same as <F5> (Halt).

## 3.9 Opening Other Windows for Debugging

There are many ways to look at your program and its execution using MPLAB IDE. For example, this program is intended to increment a temporary counter, but how do you know for sure that is happening? One way is to open and inspect the file register window. Do this by executing the *Window > File Registers* menu item. A small window with all of the file registers, or RAM, of the PIC16F84 will appear.

Press <F7> (Execute Single-Step) a few times and watch the values update in the file register window. We put the counter variable at address location 0x0C. As the temporary counter is incremented, this is reflected in the file register window. File registers change colors when their value changes so that they can easily be noticed on inspection. However, in very complex programs, many values may change, making it difficult to focus on one or two variables. This problem can be solved by using a Watch window.

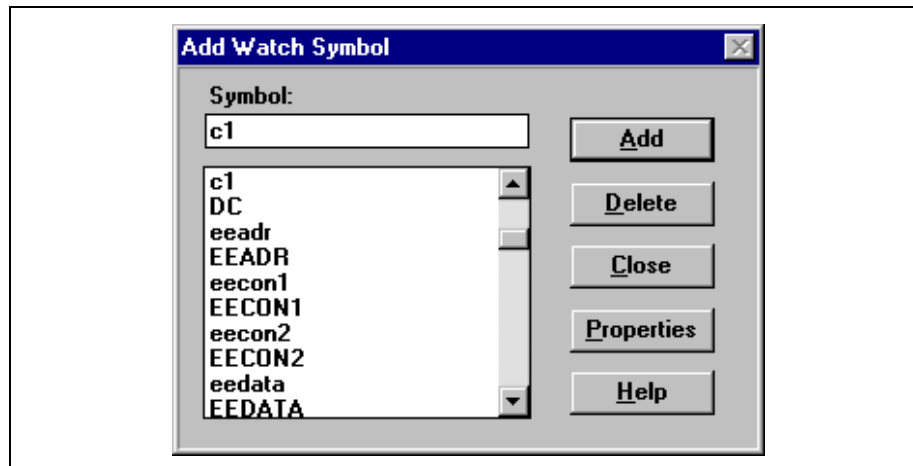
## 3.10 Using a Watch Window

MPLAB IDE allows the contents of file registers to be monitored through a Watch window.

### 3.10.1 Creating a Watch Window

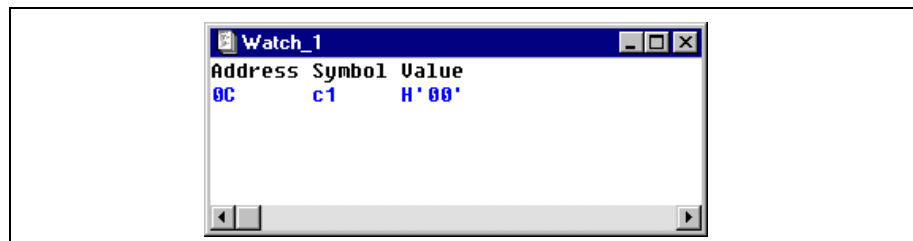
To create a Watch window, select *Window > Watch Window > New Watch Window*. If you have already created a Watch window and saved it to disk, select *Window > Watch Window > Load Watch Window*. Select the Watch window file to load and click **OK**, or double-click the desired file.

The Add Watch Symbol dialog will appear (Figure 3.14).



**Figure 3.14: Add Watch Symbol Dialog**

Typing 'c1' in the symbol name box will cause the list to scroll to the c1 symbol. Highlight the symbol and click the **Add** button, then click the **Close** button. You will be left with the Watch window on your MPLAB IDE desktop (Figure 3.15) displaying the current value of the temporary counter value 'c1'.



**Figure 3.15: Watch Window**

You can display the contents of the Watch window with or without line numbers. To change this setting, select *Toggle Line Numbers* from the system menu inside the Watch window.

Press <F7> to single step the program a few times and notice that as the counter value is incremented, the display is updated in the Watch window. If you've left the file register window open, it will update as well.

## 3.10.2 Saving the Watch Window

You can save the Watch window and its settings by selecting *Window > Watch Window > Save Active Watch* from the MPLAB IDE menu or by selecting *Save Watch* from the system menu inside the Watch window. (The system menu button is located in the upper left-hand corner of the Watch window. Clicking this button once will cause the menu underneath to cascade down.) Choose a name and click **OK**.

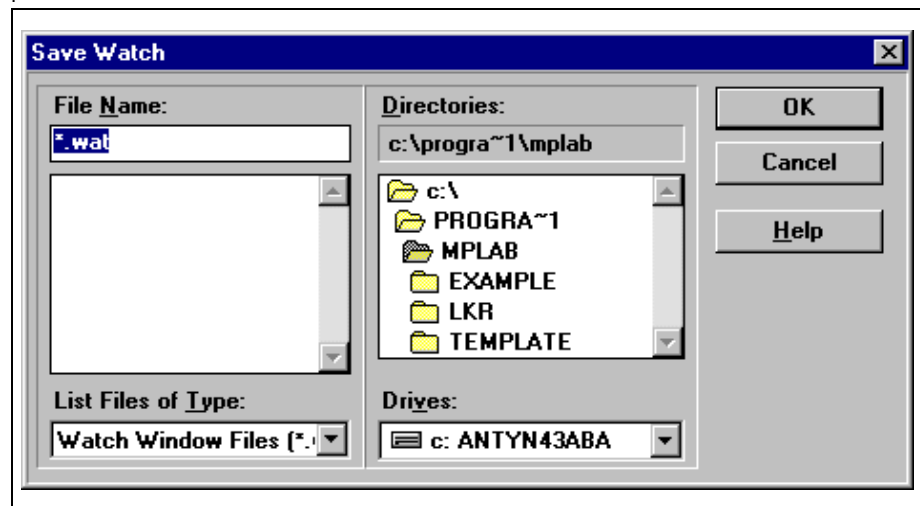


Figure 3.16: Save Watch Window Dialog

The window's open or closed status and location on the screen is saved with the project so the next time you open your project, your Watch windows will be restored as well.

### 3.10.3 Editing the Watch Window

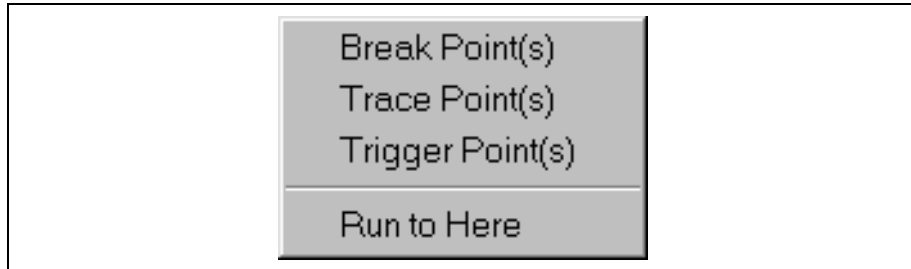
You can also edit the Watch windows after you've created them.

Use either the *Window > Watch Window* submenu or the system menu inside the Watch window to edit the information in the Watch window.

- |  |  |
|--|--|
| <i>Add a symbol to the Watch window</i>      | Select <i>Window &gt; Watch Window &gt; Add to Active Watch</i> from the MPLAB IDE menu or select <b>Add Watch</b> from the system menu inside the Watch window.   |
| <i>Delete a symbol from the Watch window</i> | Click on the symbol in the Watch window, then select <i>Delete Watch</i> from the system menu.   |
| <i>Change the display format of symbols</i>  | Select <i>Window &gt; Watch Window &gt; Edit Active Watch</i> from the MPLAB IDE menu or select <i>Edit Watch</i> from the system menu inside the Watch window. Then, click <b>Properties</b> . The Properties dialog allows you to select the format, size, byte order, and display bits for display in the Watch Window. |

## 3.11 Setting a Break Point

Press <F5> (*Debug > Run > Halt*) to make sure that the simulator processor is halted. Click in the source code window on the line immediately after the `start` label that says `movlw 0x09`. Click the right mouse button and a small shortcut menu will appear (Figure 3.17).



**Figure 3.17: Right Mouse Button Pop-up Menu**

Select the *Break Point(s)* menu item and the menu will disappear and the line where the cursor was located will change colors, indicating that a break point has been set at that location.

Press <F6> or execute the *Debug > Run > Reset* menu item to reset the system. Then run the system by pressing <F9>. The program will run and then halt at the instruction just after the break point. 'c1', as displayed in either the Watch window or the file register window (if one is still open), will reflect the reset status of zero; stepping once will execute the code and 'c1' will reflect a value of 0x09. Press <F9> a few times and notice the status bar change color while running, and will change again when the processor halts.

**Note:** If execution doesn't halt at the break point, select *Options > Development Mode* and click the **Break Options** tab. Make sure that Global Break Enable is selected (check marked).

## 3.12 Summary

This tutorial has shown you how to:

- Set up a new project
- Create and enter a source file into a project
- Assemble code
- Run your code using the simulator
- Set break points and single step your code
- Watch variables in your code

Once you are comfortable with the topics introduced here, you should look at the next section for more information on MPLAB IDE.



# Getting Started with MPLAB IDE – A Tutorial

---

Some hints and tips:

**Break Points** – You can set break points in the *Window > Program Memory* window, in the source file window (in this case `tutor84.asm`), or in the *Window > Absolute Listing* window.

**Source Files** – Use the *Window > Project Window* to bring up a list of your source files. You can double-click on the file name here to bring up that file in the editor.

**MPASM Errors** – If the MPASM assembler gives you an error, double-click on that error in the error window to go to the error in the source code. If you've got multiple errors, always choose the first error. Often one error will cause subsequent errors and fixing the first one may fix them all.

**Configuration Bits and Processor Mode** – Configuration bits in the source file will not set the mode of the processor for the simulator or emulators. For instance, the Watchdog Timer Enable configuration bit can be set so that when you program a device, the Watchdog Timer (WDT) will be turned on. You will also need to select *Options > Development Mode* and click the **Configuration** tab to enable the WDT for the simulator or emulator. This allows you to debug with it on or off without changing your source code. Use the *Options > Development Mode* Configuration tab to set the processor mode as well. Even though you can set these bits in your MPASM assembler or MPLAB CXX source file, MPLAB IDE does not automatically change modes.

**Options** – Go to *Options > Environment Setup* and click the **General** tab to do the following:

- Change the screen font or font size
- Position the toolbar on the side or bottom of the screen
- Modify the toolbar
- Change the number of characters displayed for labels

Before you close the dialog, click the **Key Mappings** tab to map European Keys to MPLAB IDE functions and special ASCII characters.

**Map Files** – Go to *Project > Edit Project* dialog and change MPASM's Node Properties to produce a MAP file named `tutor84.map`. After you've built the project, look at `tutor84.map` to see build information.

**Grayed Out Menus** – If you find menus “grayed out,” check to make sure that you haven't somehow entered the Editor Only mode. If you're sure everything is set up correctly, try exiting MPLAB IDE and restarting the program.

# MPLAB<sup>®</sup> IDE User's Guide

---

---

NOTES:

---

---

## Chapter 4. Projects Tutorial

---

---

### 4.1 Introduction

This chapter discusses in detail how to use projects in MPLAB IDE. If you completed the tutorial in Chapter 3, you may want to skip this chapter for now and return to it when you are ready to learn more about projects.

The project managers of MPLAB IDE v3.40 and later support multiple files. Previously established projects from MPLAB IDE v3.31 and earlier will be converted automatically by newer versions of MPLAB IDE when they are opened. Once a project is converted, it cannot be reopened using a previous version of MPLAB IDE.

### 4.2 Highlights

In chapter you will learn these functions of MPLAB IDE Projects:

- Overview of Projects
- Making a Project with One MPASM Assembler Source File
- Compiling a Single MPASM Assembler Source File Without Creating a Project
- Making a Project with Multiple MPASM Assembler Source Files using MPLINK Linker
- Making a Project with Other Tools

To perform these tasks, you will use the following features of MPLAB IDE:

- Install Language Tool
- New Project
- Add Nodes to a Project
- Set Project Node Properties
- Make/Build Project
- Project Window

### 4.3 Overview of Projects

A Project in MPLAB IDE is the group of files needed to build an application along with their associations to various build tools. A project is made up of a project node and one or more source nodes. The source nodes are typically assembly source files, C source files, precompiled object files, libraries and linker scripts. Usually the project is placed in the same folder as the main source files.

# MPLAB® IDE User's Guide

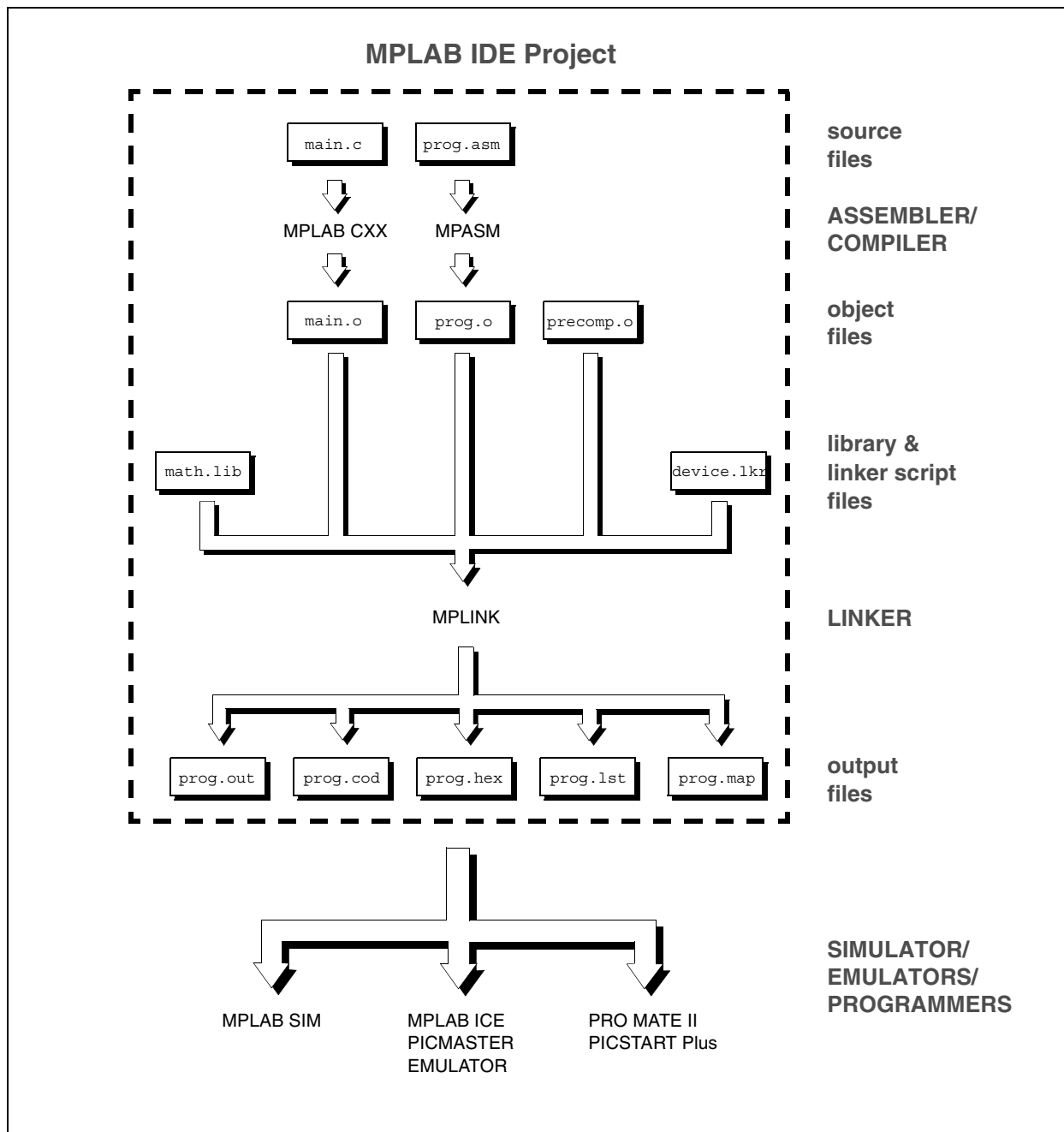


Figure 4.1: Project Relationships

In this MPLAB IDE Project, the C source file `main.c` is associated with the MPLAB CXX compiler. MPLAB IDE will use this information to generate an object file (`main.o`) for input into the linker (MPLINK). See the *MPLAB CXX C Compilers User's Guide* (DS51217) for more information on using the compiler.

An assembly source file (`prog.asm`) is shown also with its associated assembler (MPASM). MPLAB IDE will use this information to generate the object file `prog.o` for input into the MPLINK linker. See the *MPASM User's Guide with MPLINK and MPLIB* (DS33014) for more information on using the assembler.

In addition, precompiled object files (`precomp.o`) may be included in a project, with no associated tool required. Types of precompiled object files that are generally required in a project are:

- Start up code
- Initialization code
- Interrupt service routines
- Register definitions

Precompiled object files are often device and/or memory model dependent. For more information on available Microchip precompiled object files, see the *MPLAB CXX C Compilers Reference Guide* (DS51224).

Some library files (`math.lib`) are available with the compiler. Others may be built outside the project using the librarian tool (MPLIB). See the *MPASM User's Guide with MPLINK and MPLIB* (DS33014) for more information on using the librarian. For more information on available Microchip libraries, see the *MPLAB CXX C Compilers Reference Guide* (DS51224).

The object files, along with library files and a linker script file (`device.lkr`) are used to generate the project output files via the linker (MPLINK). See the *MPASM User's Guide with MPLINK and MPLIB* (DS33014) for more information on linker script files and using the linker.

The main output file generated by the MPLINK linker is the **HEX file** (`prog.hex`), used by simulators (MPLAB SIM), emulators (MPLAB ICE 2000 and PICMASTER<sup>®</sup>) and programmers (PRO MATE II and PICSTART Plus). The other output files are:

- **Code file (.cod)** – Debug file used by MPLAB IDE.
- **Listing file (.lst)** – Original source code, side-by-side with final binary code.
- **Map file (.map)** – Shows the memory layout after linking. Indicates used and unused memory regions.

The tools shown here are all Microchip development tools. However, many third party tools are available to work with MPLAB IDE Projects. Please refer to the *Third Party Guide* (DS00104) for more information.

## 4.4 Making a Project with One MPASM<sup>™</sup> Assembler Source File

To make a project that has only one MPASM assembler source file, or that uses the previous method of projects (MPLAB IDE v3.31 or earlier), wherein a single source file would `#include` other files, follow these steps.

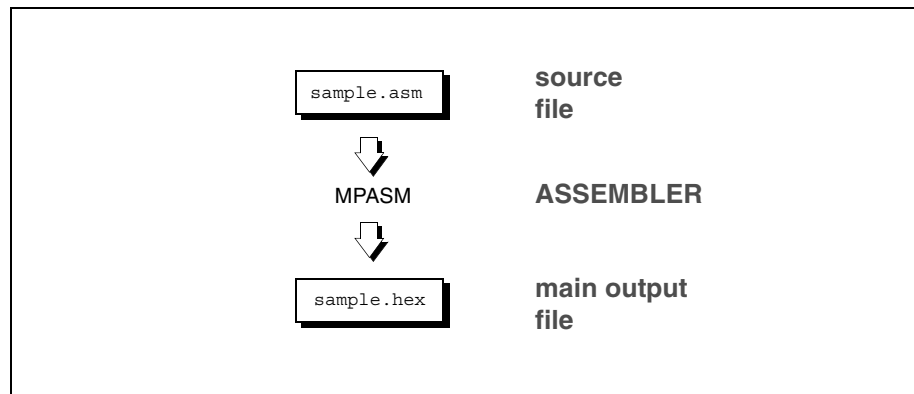


Figure 4.2: Project Relationships For One MPASM Assembler Source File

### 4.4.1 Set Development Mode

Select the proper development mode for the application. Select *Options > Development Mode* and click the **Tools** tab. For this tutorial, select MPLAB SIM simulator and select the PIC16F84 PICmicro microcontroller (MCU). Click **OK**.

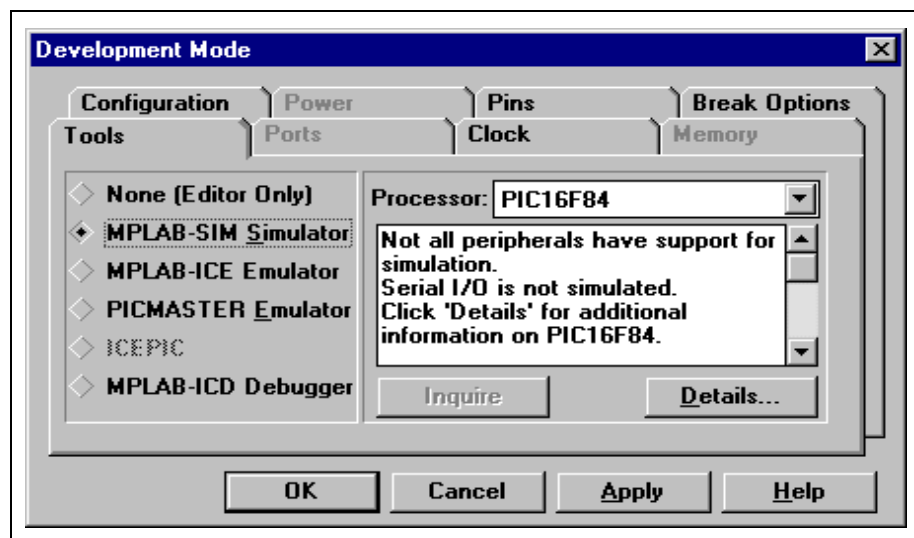


Figure 4.3: Development Mode Dialog

## 4.4.2 New Project

Select *Project > New Project*, select a folder for the new project, then type in its name. Use the \Program Files\MPLAB installation folder and name it SAMPLE.PJT for this tutorial.

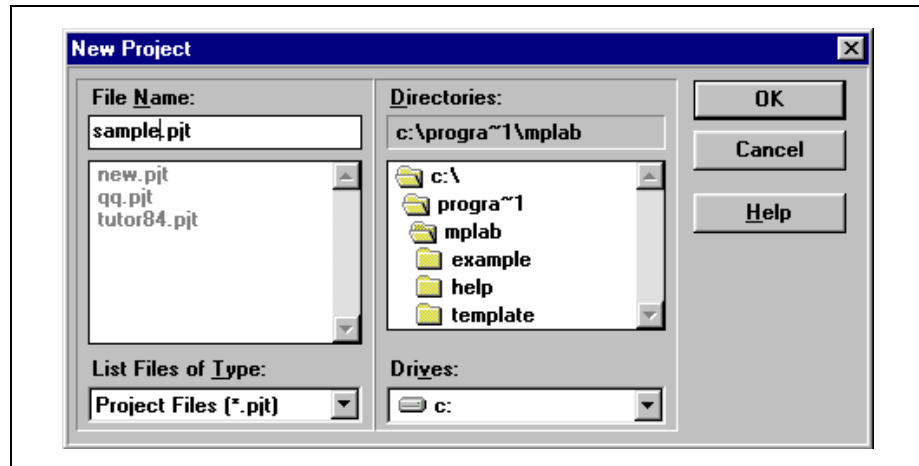


Figure 4.4: New Project Dialog – sample.pjt

## 4.4.3 Project Dialog

After clicking **OK**, you will see the Edit Project Dialog:

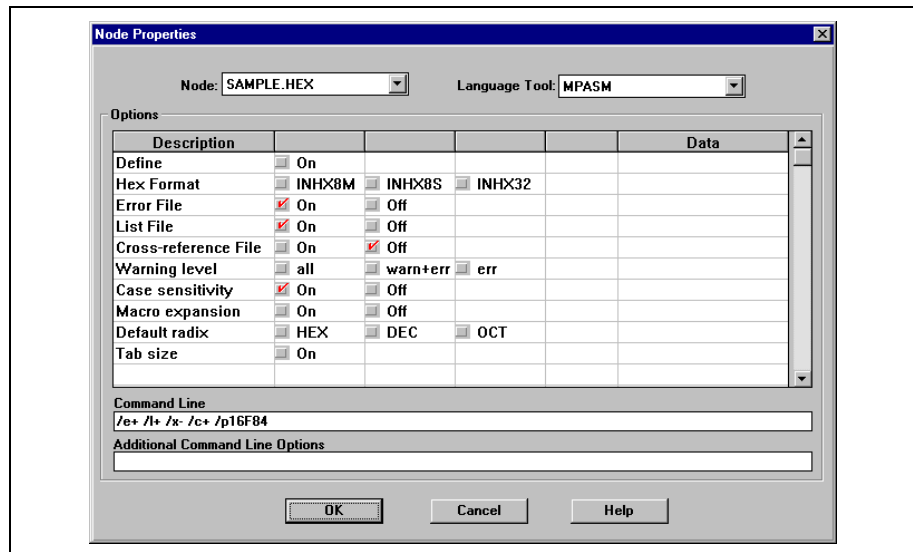


Figure 4.5: Edit Project Dialog

# MPLAB® IDE User's Guide

## 4.4.4 Set Node Properties

Select the file name, `sample.hex`, in the Project Files window, then click the **Node Properties** button.



**Figure 4.6: Node Properties Dialog**

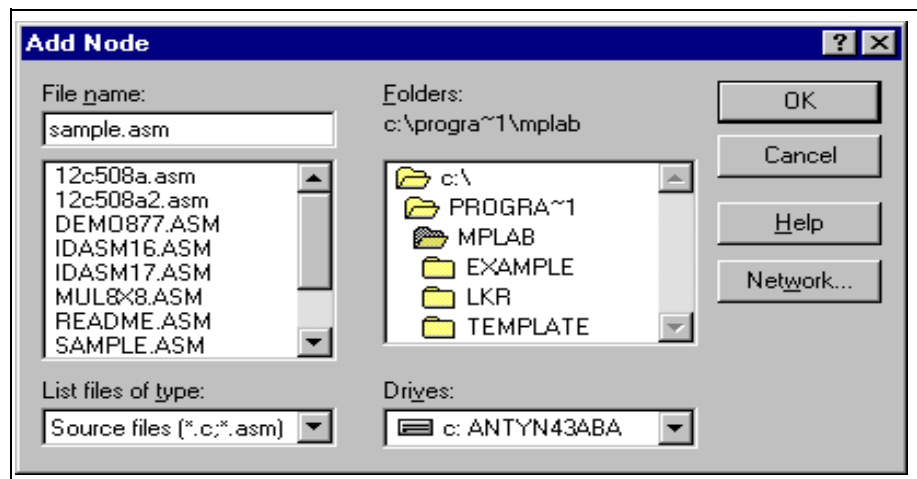
The Node Properties dialog shows the command line switches for the tool, in this case MPASM. When you first open this dialog, the checked boxes represent the default values for the tool. For this tutorial, these do not need to be changed. Refer to the *MPASM User's Guide with MPLINK and MPLIB* (DS33014) for more information on these command line switches.

Click **OK** to return to the Edit Project dialog box.



## 4.4.5 Add Node

Click **Add Node** from the Edit Project dialog. Use `sample.asm` for this tutorial. This is the browse window that pops up when you click **Add Node**.



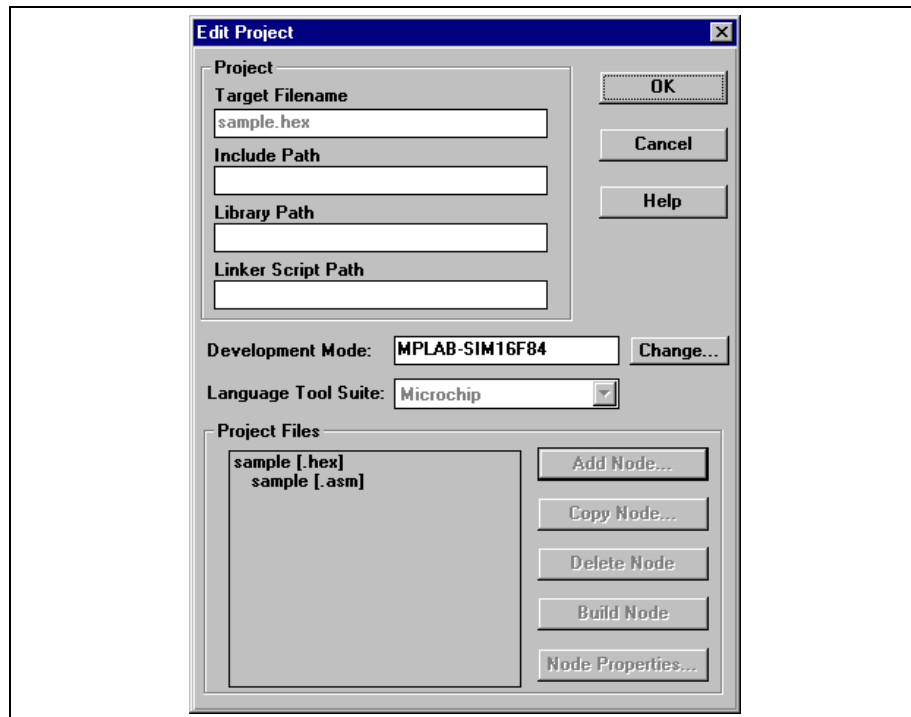
**Figure 4.7: Add Node Dialog**

The MPASM assembler always makes a `.HEX` file with the same name as the source `.ASM` file. The Project Manager will create a `sample.hex` file when the project is built.

# MPLAB<sup>®</sup> IDE User's Guide

---

The Edit Project dialog should look like this:



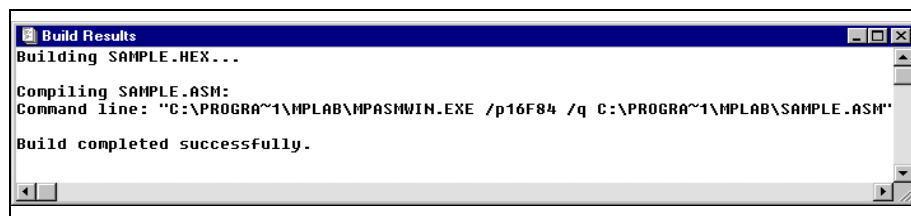
**Figure 4.8: Edit Project Dialog with Node**

In this simple example, no entries were made in the Path boxes. As your application becomes more complex, you may need to enter the folders of your include files, libraries, and linker scripts in the appropriate box. The default language suite, paths, and nodes for all projects are set selecting *Options > Environment Setup* and clicking the **Projects** tab.

Click **OK** in the Edit Project Dialog.

## 4.4.6 Make Project

Select *Project > Make Project* from the menu to compile the application using the MPASM assembler. A Build Results window is created that shows the command line sent to the assembler. It should look like this:



**Figure 4.9: Build Results Window**

## 4.4.7 Troubleshooting

If the build did not complete successfully, check these items:

1. Examine the Build Results window for syntax errors in your source file. If you find any, double-click on the error in the Build Results window to go to the line in the source file that contains the error. Correct the error, then try the build again.
2. Select *Project > Edit Project*. Select the HEX file node and click **Node Properties**. Check to see that the correct build tool (MPASM) is shown in the Node Properties dialog.
3. Select *Project > Edit Project*. Check the names of the files listed in the Project Files list. If you have accidentally added the wrong file, click on it, click **Delete Node**, then add the correct node as described in Section 4.4.5.
4. Select *Project > Install Language Tool...* and check that the MPASM assembler is pointed to MPASMWIN.EXE in the MPLAB IDE installation folder. Also, the “Windowed” option should be selected. Alternatively, the MPASM assembler can point to MPASM.EXE and the “Command-line” option selected; however this executable may not operate on Pentium 100MHz PCs and higher.

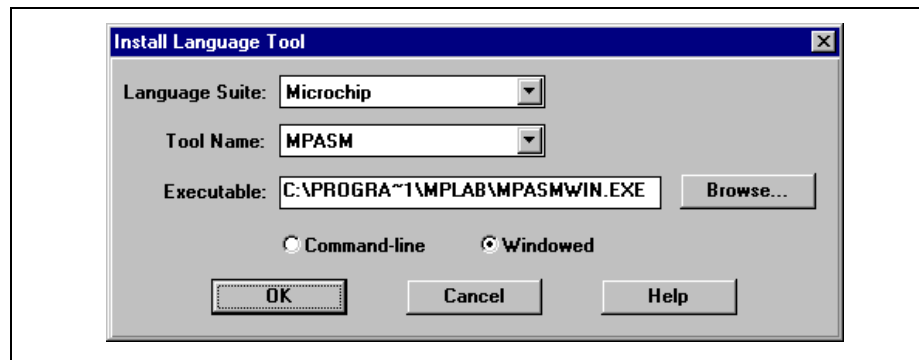


Figure 4.10: Install Language Tool Dialog

## 4.4.8 Project Window

Open the *Window > Project* window to see that the target name is set properly to match the Node source name. They will have different file extensions, .ASM and .HEX, but both are named *SAMPLE* for this tutorial.

The Project window should look like this:

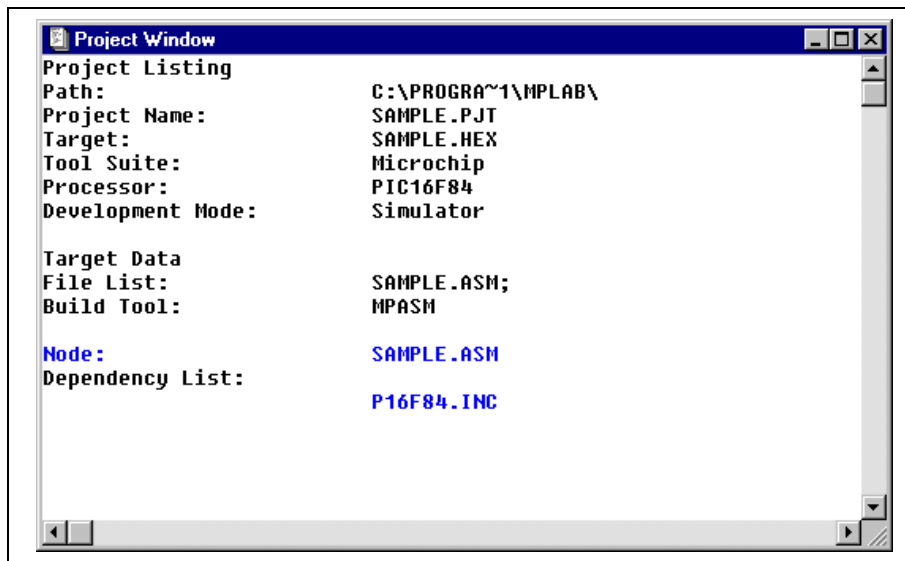


Figure 4.11: Project Window

## 4.4.9 Summary

Here is a quick list of the steps to set up a new project as described above:

- Create new project with *Project > New Project*.
- Set project Node Properties to MPASM and select the desired build options.
- Add Source file node.

## 4.5 Compiling a Single MPASM Assembler Source File Without Creating a Project

It is possible to compile a single file without opening up a project. The disadvantage of this method is that although no initial project setup is needed, you must specify options every time you compile the file. This example will use the same assembly language file used in the last example.

You must first close any open projects. To do this, select *Project > Close Project*.

### 4.5.1 Set Development Mode

Select the proper development mode for the application. For this tutorial, select *Options > Development Mode* and click the **Tools** tab. Select MPLAB SIM simulator and select the PIC16F84 PICmicro MCU. Click **OK**.

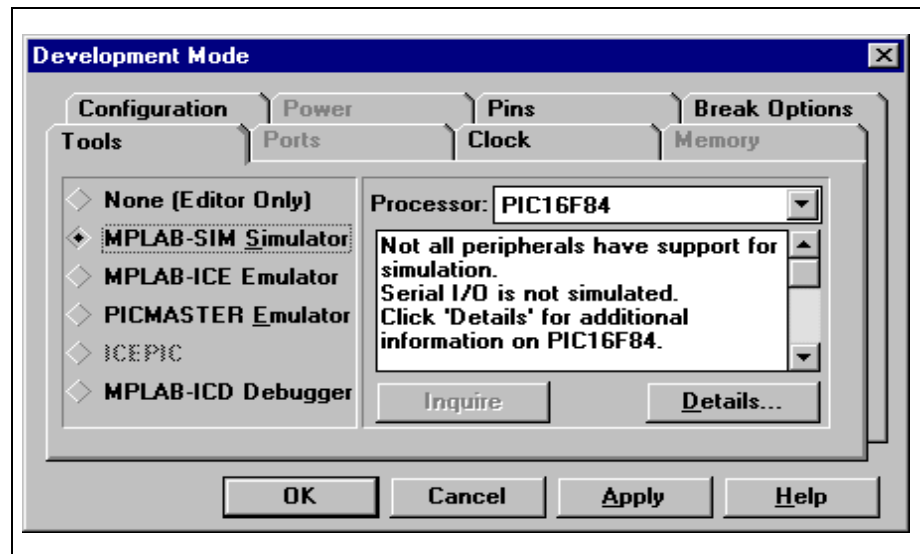
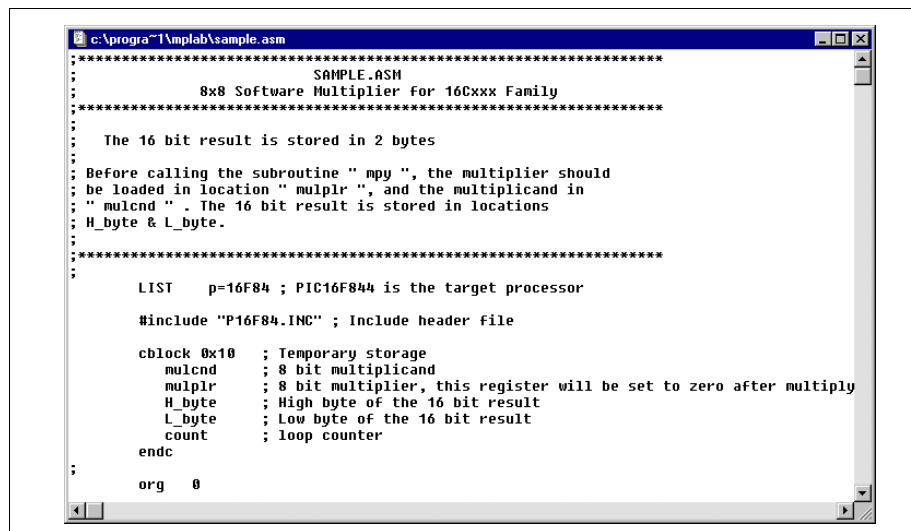


Figure 4.12: Development Mode Dialog

# MPLAB® IDE User's Guide

## 4.5.2 Open Source File

Open the source file that you wish to assemble. For this tutorial, use `sample.asm` from the MPLAB IDE installation folder.



```
c:\progra~1\mplab\sample.asm
:*****
:                               SAMPLE.ASM
:                               8x8 Software Multiplier for 16Cxxx Family
:*****
:
: The 16 bit result is stored in 2 bytes
:
: Before calling the subroutine " mpy ", the multiplier should
: be loaded in location " mulplr ", and the multiplicand in
: " mulcnd ". The 16 bit result is stored in locations
: H_byte & L_byte.
:*****
:
: LIST    p=16F84 ; PIC16F844 is the target processor
: #include "P16F84.INC" ; Include header file
:
: cblock 0x10 ; Temporary storage
:   mulcnd ; 8 bit multiplicand
:   mulplr ; 8 bit multiplier, this register will be set to zero after multiply
:   H_byte ; High byte of the 16 bit result
:   L_byte ; Low byte of the 16 bit result
:   count ; loop counter
: endc
:
: org 0
```

Figure 4.13: Source File Window

## 4.5.3 Compile Source File

Select *Project > Build Node* from the menu to compile `sample.asm` using the MPASM assembler. MPLAB IDE opens an Invoke Build Tool Dialog that looks like this:

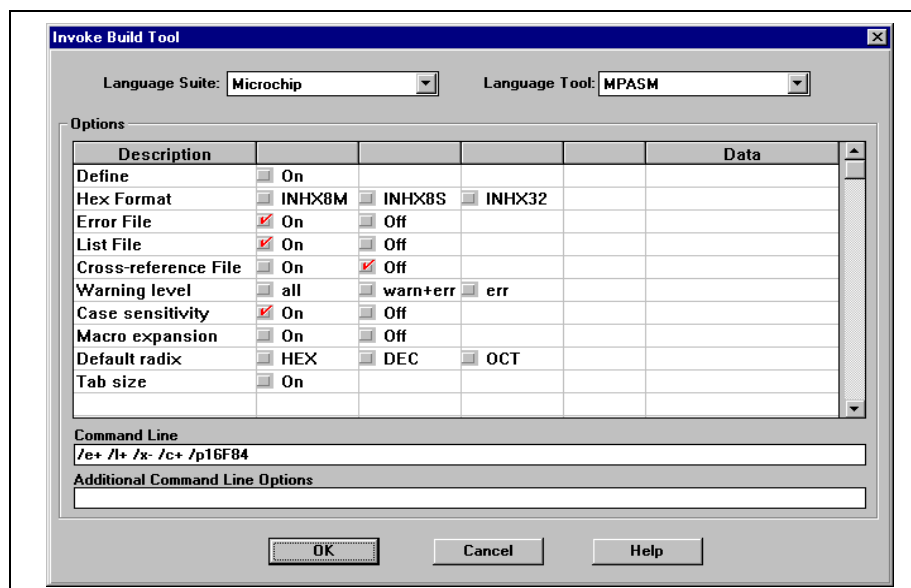


Figure 4.14: Build Tool Dialog

Verify that the MPASM assembler is selected, and set the tool options to match those shown above. Click **OK** in the Invoke Build Tool Dialog to start the build process. A Build Results window is generated that shows the command line sent to the assembler and the build output. It should look like this:

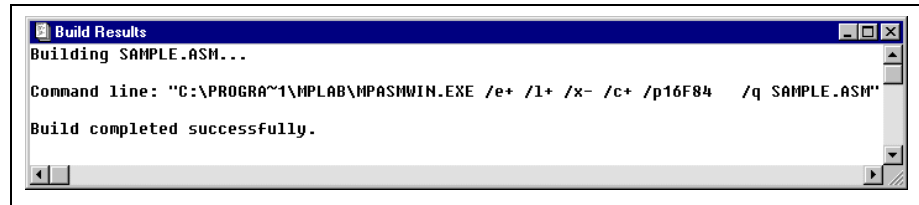


Figure 4.15: Build Results Window

## 4.5.4 Troubleshooting

If the build did not complete successfully, check these items:

1. If you modified the sample source code, examine the Build Results window for syntax errors in your source file. If you find any, double-click on the error in the Build Results window to go to the line in the source file that contains the error. Correct the error, then try the build again.
2. Select *Project > Install Language Tool...* and check that MPASM assembler is pointed to MPASMWIN.EXE in the MPLAB IDE installation folder. Also, the "Windowed" option should be selected. Alternatively, the MPASM assembler can point to MPASM.EXE and the "Command-line" option selected; however this executable may not operate on Pentium 100MHz PCs and higher.

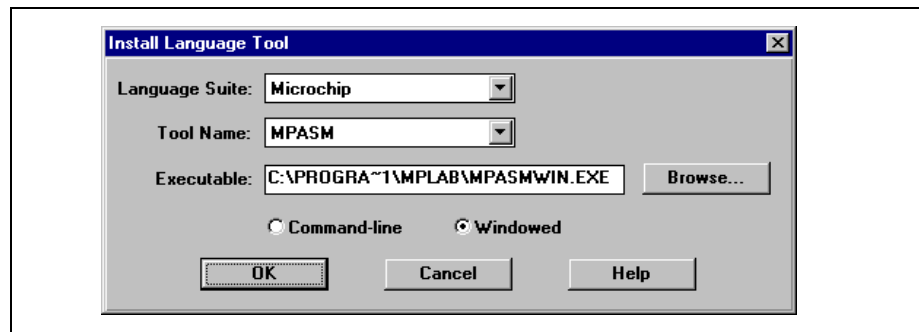


Figure 4.16: Install Language Tool Dialog

## 4.5.5 Summary

Here is a quick list of the steps to set up a new project as described above:

- If a project is open, close it using *Project > Close Project*.
- Open the source file you wish to compile.
- Select *Project > Build Node*.
- Select the desired language suite, build tool, and build options in the Invoke Build Tool dialog.



## 4.6 Making a Project with Multiple MPASM Assembler Source Files using MPLINK™ Linker

To use MPLINK linker to link two or more MPASM assembler object files, follow these steps. If you followed through the previous section, select *Project > Close Project*.

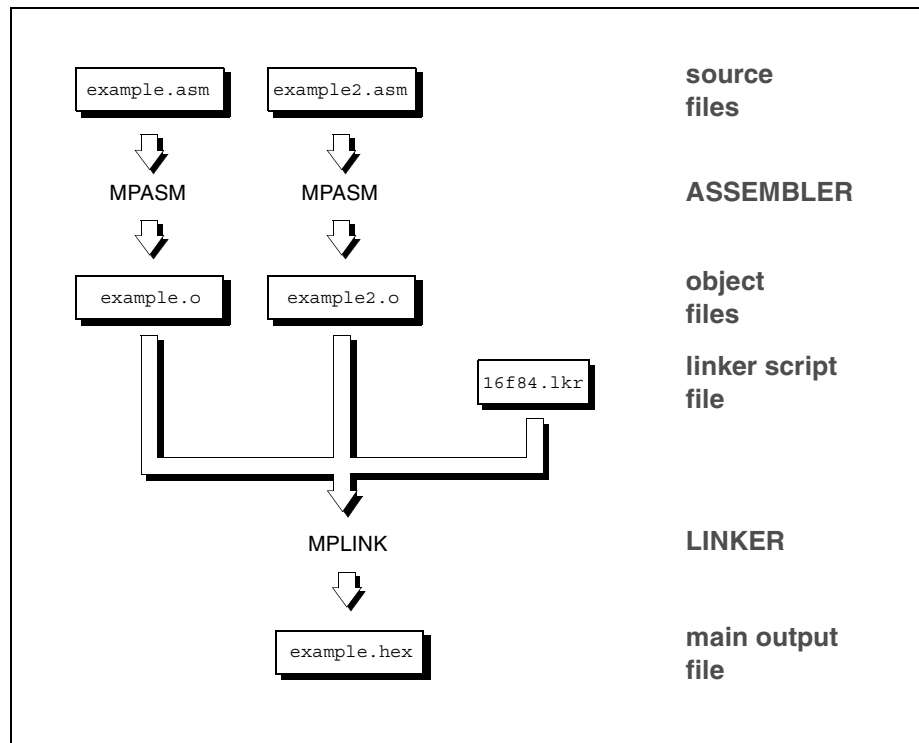


Figure 4.17: Project Relationships For Multiple MPASM Assembler Source Files

### 4.6.1 Set Development Mode

Select *Options > Development Mode* and click the **Tools** tab. Select MPLAB SIM simulator and select the PIC16F84 PICmicro MCU for this example. Click **OK**.

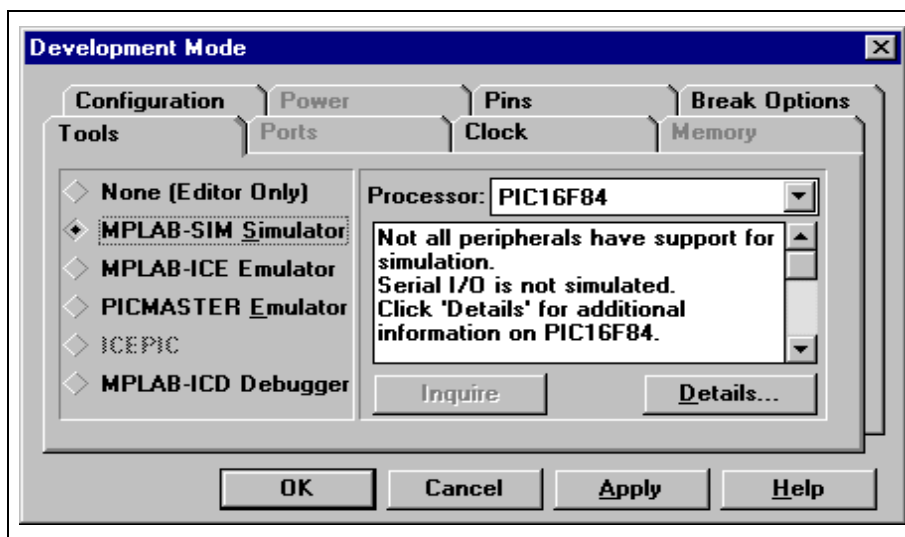


Figure 4.18: Development Mode Dialog

## 4.6.2 New Project

Select *Project > New Project*, browse to select a folder for a new project, then type in its name. Use the `\PROGRAM FILES\MPLAB\EXAMPLE` folder for this tutorial and name the file `EXAMPLE.PJT`.

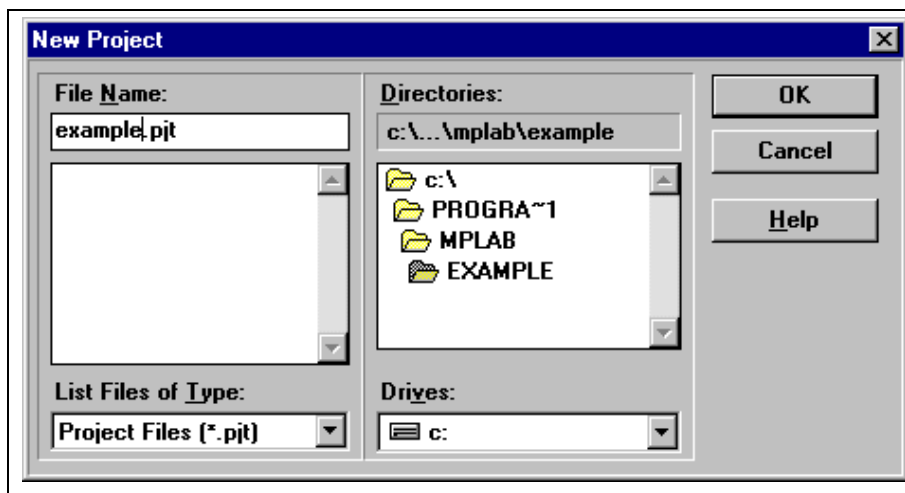


Figure 4.19: New Project Dialog – example.pjt

## 4.6.3 Set Node Properties

Select the name of the project in the Project Files dialog of the Edit Project Dialog and click **Node Properties** to open this dialog. Set the language tool to MPLINK.

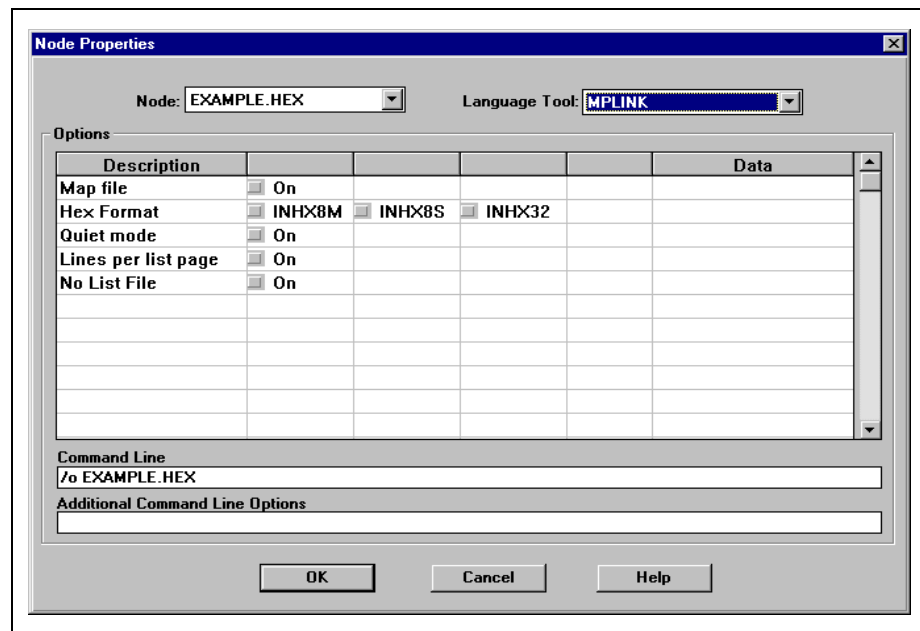


Figure 4.20: Set Node Properties Dialog

The Node Properties dialog shows the command line switches for the tool, in this case MPLINK. When you first open this dialog, the checked boxes represent the default values for the tool. For this tutorial, these do not need to be changed. Refer to the *MPASM User's Guide with MPLINK and MPLIB* (DS33014) for more information on these command line switches.

Click **OK** to return to the Edit Project Dialog box.

## 4.6.4 Add First Source File Node

Select **Add Node** from the Edit Project Dialog. Use `example.asm` in the `\PROGRAM FILES\MPLAB\EXAMPLE` folder for this tutorial.

# MPLAB<sup>®</sup> IDE User's Guide

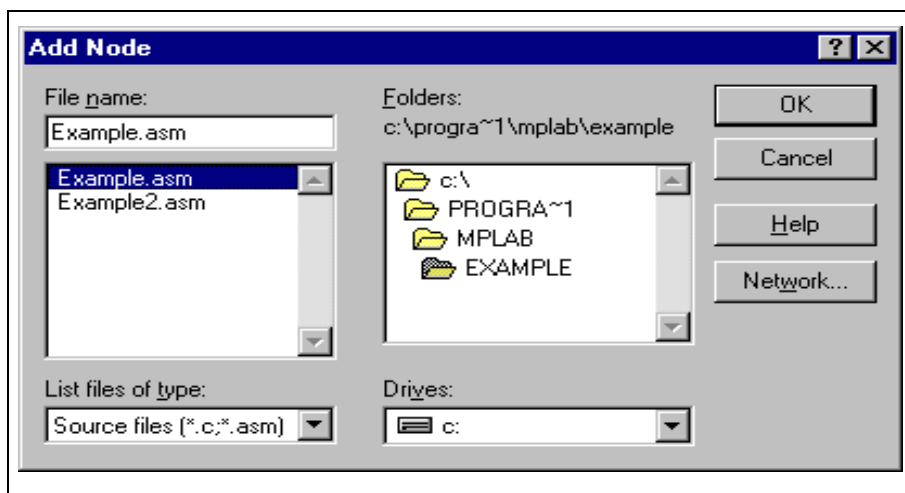


Figure 4.21: Add Node Dialog

You may select more than one file at a time from this dialog. To select several files without selecting the files between them, hold the <Ctrl> key while selecting each file. To select a range of files hold the <Shift> key and select the first and last files.

Select `example.asm` from the list of project files in the Edit Project dialog, and click **Node Properties**.

Verify that the language tool is set to MPASM.

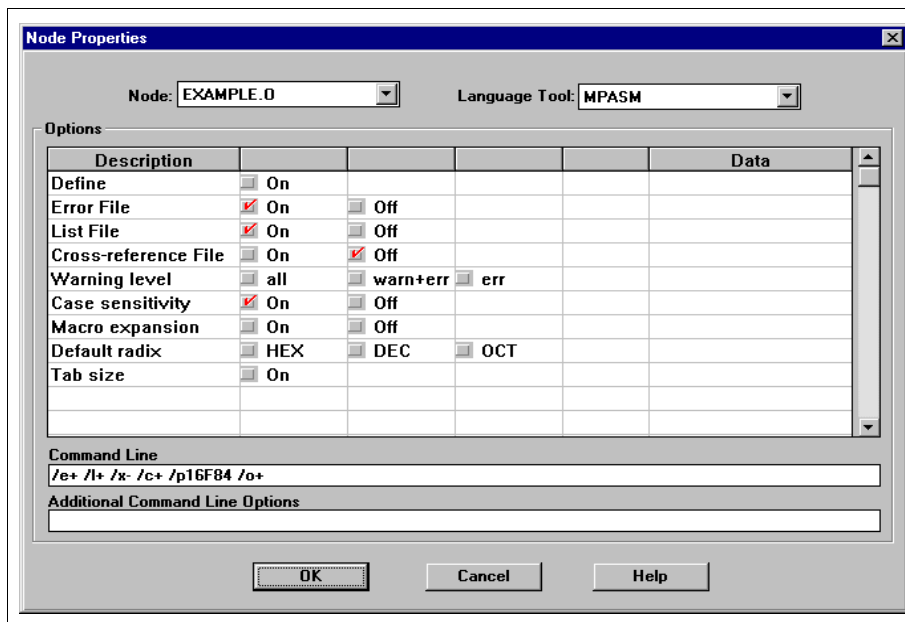


Figure 4.22: Node Properties Dialog – Example.o

The Node Properties dialog shows the command line switches for the tool, in this case the MPASM assembler. Refer to the *MPASM User's Guide with MPLINK and MPLIB* (DS33014) for more information on these command line switches.

Click **OK** to return to the Edit Project Dialog box.

## 4.6.5 Adding Additional Source Files

Follow the previous two steps to add the rest of the source files to the project. For this tutorial, select `example2.asm` from the `\PROGRAM FILES\MPLAB\EXAMPLE` folder. You can also use **Copy Node** to enter subsequent files with the same Node Options as the first source file. Make sure the Node Options are set properly on each file.

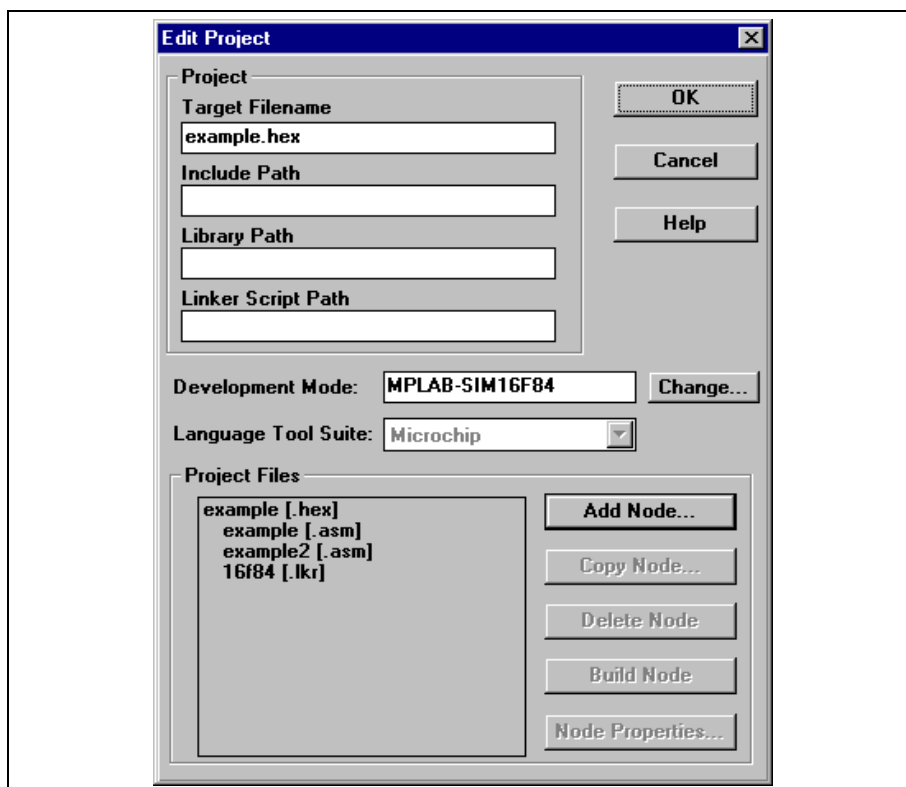
To use the Copy Node feature, select one of the source node files listed in the Project Files box. Then click **Copy Node**. In the Add Node dialog box, select one or more source files. Once the files are selected, click **OK**. This step will set up the node properties for this selected files the same as the referenced node used. This is especially useful for adding multiple source files with identical node properties.

## 4.6.6 Select Linker Script

Select a linker script using the **Add Node** button and the method described above. A linker script is a file that MPLINK linker uses to define the memory architecture of each PICmicro MCU. Standard linker scripts come with MPLINK linker and are in the MPLAB IDE installation folder. For this tutorial select `PIC16F84.LKR` from the `\PROGRAM FILES\MPLAB\EXAMPLE` folder. Node options can not be set for a linker script.

# MPLAB® IDE User's Guide

---



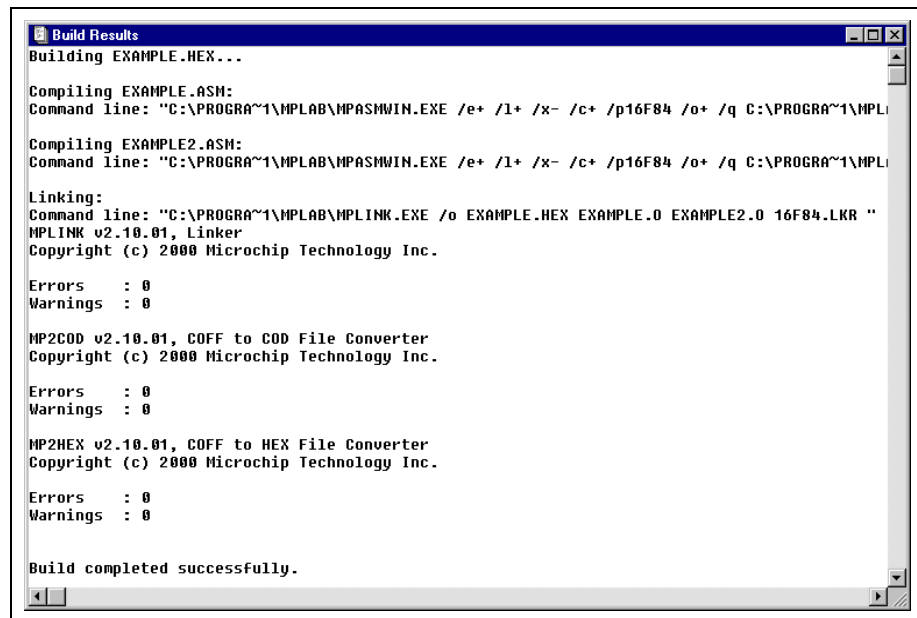
**Figure 4.23: Edit Project Dialog – Linker Script**

Click **OK** in the Edit Project dialog.

In this simple example, no entries were made in the three “Path” boxes. As your application becomes more complex, you may need to enter the folders of your include files, library files, and linker script files in the appropriate box if they are not in the same folder as the project. The default language suite, paths, and nodes for all projects are set by selecting *Options > Environment Setup* and clicking the **Projects** tab (see Section 1.9.5.2).

## 4.6.7 Make Project

Select *Project > Make Project* to compile the application using MPASM assembler and MPLINK linker. A Build Results window is created that shows the command lines sent to each tool. It should look like this:



```
Build Results
Building EXAMPLE.HEX...

Compiling EXAMPLE.ASM:
Command line: "C:\PROGRA~1\MPLAB\MPASMWIN.EXE /e+ /l+ /x- /c+ /p16F84 /o+ /q C:\PROGRA~1\MPLI

Compiling EXAMPLE2.ASM:
Command line: "C:\PROGRA~1\MPLAB\MPASMWIN.EXE /e+ /l+ /x- /c+ /p16F84 /o+ /q C:\PROGRA~1\MPLI

Linking:
Command line: "C:\PROGRA~1\MPLAB\MPLINK.EXE /o EXAMPLE.HEX EXAMPLE.O EXAMPLE2.O 16F84.LKR "
MPLINK v2.10.01, Linker
Copyright (c) 2000 Microchip Technology Inc.

Errors      : 0
Warnings   : 0

MP2COD v2.10.01, COFF to COB File Converter
Copyright (c) 2000 Microchip Technology Inc.

Errors      : 0
Warnings   : 0

MP2HEX v2.10.01, COFF to HEX File Converter
Copyright (c) 2000 Microchip Technology Inc.

Errors      : 0
Warnings   : 0

Build completed successfully.
```

Figure 4.24: Build Results Window

## 4.6.8 Troubleshooting

If the build did not complete successfully, check these items:

1. If you modified the sample source code, examine the Build Results window for syntax errors in your source file. If you find any, double-click on the error in the Build Results window to go to the line in the source file that contains the error. Correct the error, then try the build again.
2. Select *Project > Edit Project*. Select the HEX file node and click **Node Properties**. Check to see that the correct build tool (MPLINK) is shown in the Node Properties dialog. The build tool for the source files for this project should be MPASM.
3. Select *Project > Edit Project*. Check the names of the files listed in the Project Files list. If you have accidentally added the wrong file, click on it, click **Delete Node**, then add the correct node as described in Section 4.6.5.
4. If MPLAB IDE reports the message, "Time-out," click **OK** to continue. Depending on the speed of your PC and the size of your project, you may wish to configure the length of time MPLAB IDE will wait before reporting a timeout message. This value is set by selecting *Options > Environment Setup*, clicking the **Project** tab, and adjusting the Build Timeout Length in the dialog box. Set the Build Timeout Length to 0 if you never wish to see a timeout error.

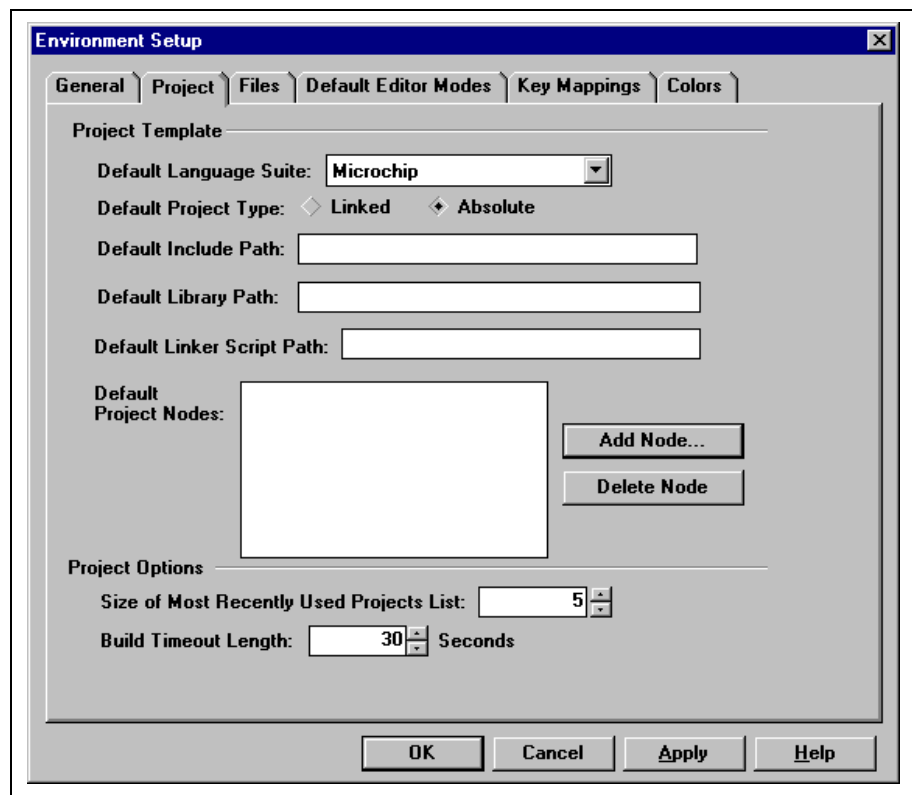


Figure 4.25: Environment Setup Dialog





## 4.6.10 Summary

Here is a quick list of the steps to set up a new project as described above:

- Create new project with *Project > New Project*.
- Set project Node Properties to MPLINK.
- Add Source file nodes, and set node properties as needed.
- Add Linker Script file node.

## 4.7 Making a Project with Other Tools

Other tools may be used with MPLAB IDE Projects to build applications. Microchip makes the MPLAB C17 and MPLAB C18 C compilers for PIC17CXXX and PIC18CXXX devices, respectively. Contact any Microchip sales office (see back cover) or our web site (<http://www.microchip.com>) for information on how to purchase these compilers. Also, see our web site for tutorials on how to use these compilers with MPLAB IDE.

Certain third party tools also work with MPLAB IDE Projects. Consult the *Third Party Guide* (DS00104) for more on these tools. Also, see our web site for tutorials on how to use these tools with MPLAB IDE.

## Chapter 1. MPLAB Editor

### 1.1 Introduction

This chapter defines what MPLAB Editor is and how it helps you, as well as introducing the features and functions of the editor. Since the MPLAB Editor is integrated into MPLAB IDE, its functions are detailed in Part 3, Chapter 1 along with all other MPLAB IDE functions.

### 1.2 Highlights

The following topics are addressed in this chapter:

- What is the MPLAB Editor
- How MPLAB Editor Helps You
- MPLAB Editor Features
- MPLAB Editor Functions

### 1.3 What is the MPLAB Editor

The MPLAB Editor is an integrated part of the MPLAB Integrated Development Environment (IDE). The editor is always available when MPLAB IDE is running. It is not a separate executable file but a set of features in MPLAB IDE.

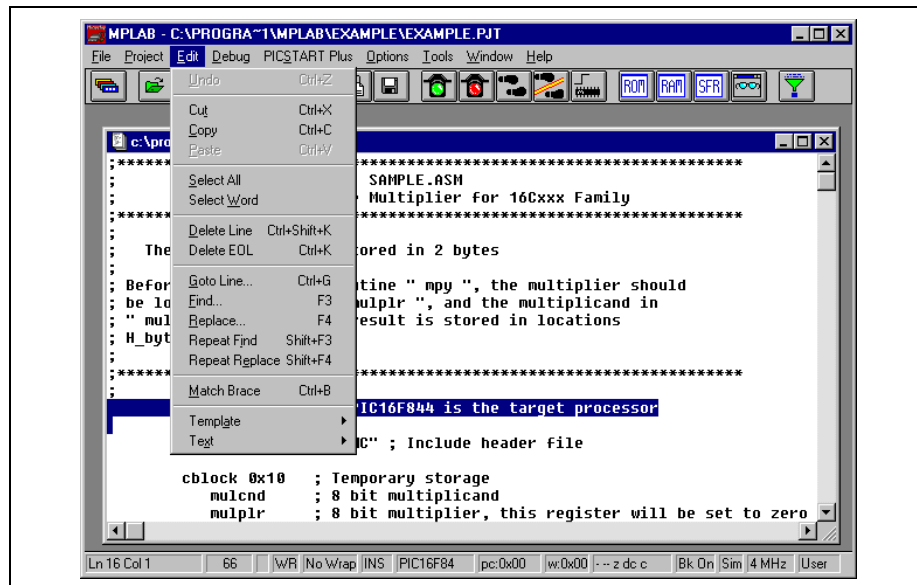


Figure 1.1: Using the MPLAB Editor

## 1.4 How MPLAB Editor Helps You

The MPLAB IDE and Editor are designed to allow PICmicro microcontroller developers an easy and quick method to develop and debug firmware for Microchip Technology Incorporated's PICmicro microcontroller product families.

## 1.5 MPLAB Editor Features

### 1.5.1 File Size

The MPLAB Editor is limited only by the total amount of available memory on your system. There are no limits on the number of editable files or on the number of open edit windows. The MPLAB Editor also has no limit on the size of file that it can open and has no limit on the number of lines that a file may contain.

### 1.5.2 Windows MDI Conventions

MPLAB Editor adheres to the Windows MDI conventions:

- Invokes most commands and facilities from menus
- Moves around with a mouse or with standard keyboard shortcuts
- Supplies cut and paste capabilities from the clipboard

### 1.5.3 Reconfigure Keyboard

You can reconfigure the keys to meet your requirements. Commands may be invoked by two-character key sequences like <Esc+G>, <Ctrl+K>, and <Ctrl+B>. You may also use key sequences such as <Alt+F> and <Alt+S>. You can map almost all keyboard keys in any combination.

### 1.5.4 Build Files Easily

MPLAB Editor allows you to:

- Define sets of templates – standard lines of text – that you can insert into the current file with just a few mouse clicks.
- Group the templates you work with into distinct files.
- Load templates for automatic usage.

## 1.6 MPLAB Editor Functions

The MPLAB Editor provides functions that allow you to perform the following operations:

- File operations
- Template operations
- Text handling
- Editor window modes
- C language awareness

### 1.6.1 File Operations

To create a file, select a file for editing, and save a file, use the File menu options (see Section 1.4). The MPLAB Editor allows you to save the file by overwriting the existing file or by saving the file to a new filename.

### 1.6.2 Template Operations

How do you take the repetition (and the waste) out of creating new source code files? You could simply paste the necessary code and text into your new source files by copying it from your previously completed source files. However, that can be an error-prone process.

MPLAB IDE provides you with templates, which are pre-built text files or sections of text that you can insert in your source files. By inserting these “canned” sections of text instead of repetitively typing it into new source files, you reduce your initial code development time. You can use the absolute code templates included in the Templates folder inside the MPLAB IDE folder or create your own.

Once you set up MPLAB IDE to use your templates, you can create your new source file (or open an existing one), insert the template text into the new source file, and search for the special markers that help you quickly locate the areas that you will need to customize during your application development.

You can repeat the above steps to create several different .tpl files if you wish. For example, you may want to a separate .tpl file for each type of application or for each device.

## 1.6.3 Text Processing

Although MPLAB Editor is intended to be used as a program text editor, it has several features that make it useful in general text editing. Refer to Section 1.6 for details.

### 1.6.3.1 Inserting, Selecting, and Deleting Text

MPLAB Editor inserts text in either insert or strikeover mode. MPLAB Editor shows the mode as either “INS” or “OVR” on the status bar.

Text selection features allow you to select a character, word, or an entire line. You can delete a character, entire line, or delete from the cursor position to the end of the line. You can also use the MPLAB Editor's built-in find and replace feature to search for and replace text or special characters.

### 1.6.3.2 Indenting and Unindenting Text

When editing program source, indenting and unindenting source is very common. The MPLAB Editor provides a facility to change the indentation level of one or more lines of text.

### 1.6.3.3 Changing Case

The MPLAB Editor allows you to change the case of selected text between uppercase and lowercase.

### 1.6.3.4 Handling Braces

MPLAB Editor allows the user to manipulate brace characters such as brackets and parentheses, which often delimit sections of text or program sources.

### 1.6.3.5 Undoing Edit Actions

The MPLAB Editor records edit actions and can reverse them with the Undo command.

### 1.6.3.6 Automatic Text Wrapping

When typing ordinary text, it may be convenient to have the program fit the text into the available line width. This would typically not be the case when editing a program source code file.

To change the text wrapping mode, double-click the left mouse button in the wrap area of the status bar. This area shows the text “No Wrap” when wrapping is not active. The double-click action turns wrapping on. For example, the status bar shows “Wr 72” when wrapping is enabled and set at column 72.

The points at which MPLAB Editor wraps a line vary with the language type defined for the window.

### 1.6.3.6.1. Language type “(none)” or “C”

MPLAB Editor breaks the line at the closest white space character or hyphen to the defined wrap column.

### 1.6.3.6.2. Language type “TeXt”

MPLAB Editor breaks the line at the closest white space character to the defined wrap column.

**Note:** MPLAB Editor wraps the line being typed only when the cursor is at the end of the line. If you move the cursor to somewhere within the line and enter text, MPLAB Editor does not wrap the line even if it extends past the wrap column.

## 1.6.4 Edit Window Modes

The MPLAB Editor associates a set of window modes with every edit window. The possible window modes affect the screen formatting, text display and input, printing, and file modes. See Part 3, Section 1.9.3 and Section 1.9.4 for more information on Editor Modes.

## 1.6.5 C Language Awareness

When editing files that have language type set to “C,” MPLAB Editor provides these facilities:

- MPLAB Editor always moves a “#” character typed in an otherwise empty line to column 1.
- MPLAB Editor moves a closing “}” brace typed in an otherwise empty line to the same column as the matching preceding opening brace “{” if the opening brace is the only character in its line.

For example:

```
//
*****
// EXAMPLE.C
//
*****
#include <PIC16C84.H>
void delay(void);
void main(void)
{
    unsigned int i,j;
    TRISB = 0xff;
    PORTB = 0;
    i = 0x1;
```

# MPLAB<sup>®</sup> IDE User's Guide

---

```
while(1)
{
    PORTB = i;
    if (i == 0x80)
        i = 0x1;
    else
        i <<= 1;
    TRISB = 0;
    delay();
    TRISB = 0xff;
    delay();
}
}
void delay(void)
{
    int x, y;
    x = 0x3f;
    y = 0xff;
    while(x--)
    {
        while(y--)
            NOP();
    }
}
```



---

---

**Chapter 2. Debugging and MPLAB SIM Simulator**

---

---

## 2.1 Introduction

This chapter discusses MPLAB IDE debugging functions and related MPLAB SIM simulator considerations. You can be in the simulator (MPLAB SIM) or in emulator mode (MPLAB ICE 2000 in-circuit emulator, PICMASTER<sup>®</sup> emulator, ICEPIC, or MPLAB ICD) to access debugging functions. Refer to the *MPLAB ICE In-Circuit Emulator User's Guide* (DS51159) for information on debugging using the MPLAB ICE 2000 emulator.

## 2.2 Highlights

This chapter covers the following information:

- MPLAB IDE Debugging Functions
- Real-Time Program Execution
- MPLAB SIM Simulator Environment
- Simulator Considerations
- Break and Trace Points
- Conditional Break Dialog
- Stimulus Functions
- Simulator Issues:
  - 12-Bit Core Device
  - 14-Bit Core Device
  - 16-Bit Core Device
  - Enhanced 16-Bit Core Device

## 2.3 MPLAB IDE Debugging Functions

After setting up and compiling projects in MPLAB IDE, you'll want to see how your code runs. If you have a device programmer, you can program a microcontroller device and plug the programmed device into your application to verify that the application runs as expected. Usually, an application will not run correctly the first time, and you'll have to debug the code. You can use MPLAB SIM simulator to simulate your code or you can use the MPLAB ICE 2000 emulator to run your firmware in the application while you debug.

Either way, you will use break and trace points as you run your code. Look at register values in the Register window or Special Function Register window to see the processor's state as you run and single-step your code.

# MPLAB<sup>®</sup> IDE User's Guide

---

The MPLAB ICE 2000 emulator runs code at the actual execution speed (real-time) on your target hardware, stopping only at specified break points. MPLAB SIM simulates the execution of any PICmicro microcontroller (MCU) and simulates I/O conditions at speeds that depend on the speed of your PC.

The following debug functions work the same with the simulator or the emulator. The main functions are:

- Emulation Memory (Program Memory Window)
- Break and Trace Points
- Single-Stepping
- Register Monitoring (Special Function Register or File Register Windows)

All of these functions use information from an MPLAB IDE project. Line labels in source code, symbolic locations in memory, and function names from code can be used to set break and trace points and to examine and modify registers.

## 2.4 Real-Time Program Execution

In this document the term “real-time” is usually applicable only to the emulators (ICEs) or in-circuit debuggers (ICDs).

### 2.4.1 Execution in MPLAB SIM Simulator Mode

When the system is said to be running in real-time in the simulator mode, instructions are executing as quickly as possible by the PC's CPU. This is usually slower than the actual PICmicro MCU would run at its rated clock speed.

The speed at which the simulator runs depends on the speed of your computer and the number of other tasks running in the background. The software simulator must update all of the simulated registers and RAM, monitor I/O, set and clear flags, check for break and trace points in software, and simulate the PICmicro MCU instruction with instructions being executed on your computer's CPU.

**Note:** Often loops are used in 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 manual says “real-time” and you are in the simulator mode, this means that the software simulation is executing simulated PICmicro MCU code as fast as your PC can simulate the instructions.

# Debugging and MPLAB SIM Simulator

---

## 2.4.2 Animate Mode

Animate Mode is a method of automatically single-stepping the processor. The simulator actually executes single steps while in Run mode, but it only updates the values of the registers when it is halted. To view the changing registers in the Special Function Register window or the Watch windows, use Animate mode. Animate mode runs slower than the Run function, but allows you to view changing register values.

## 2.5 MPLAB SIM Simulator Environment

MPLAB SIM simulator is a discrete-event simulator for the PICmicro MCU families and is integrated into MPLAB IDE. The MPLAB SIM simulator tool is designed to:

- Model operation of Microchip Technology's PICmicro MCU, e.g., PIC12CXXX, PIC14000, PIC16C5X, PIC16CXXX, PIC17CXXX, and PIC18CXXX.
- Assist users in debugging software that uses Microchip PICmicro MCU devices.

A discrete-event simulator, as opposed to an in-circuit emulator (like MPLAB ICE 2000) is designed to debug software. MPLAB SIM simulator allows you to modify object code and immediately reexecute, inject external stimuli to the simulated processor, and trace the execution of the object code. A simulator differs from an in-circuit emulator in three important areas:

- I/O timing
- Execution speed
- Cost

### 2.5.1 I/O Timing

External timing in MPLAB SIM simulator is processed only once during each instruction cycle. Transient signals, such as a spikes on  $\overline{MCLR}$  smaller than an instruction cycle, will not be simulated but may be caught by an in-circuit emulator.

**Note:** Stimulus is injected into MPLAB SIM simulator prior to the next instruction cycle.

### 2.5.2 Execution Speed

The execution speed of a discrete-event software simulator is orders of magnitude less than a hardware oriented solution. Users may view slower execution speed as a handicap or as a tool. MPLAB SIM 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.

## 2.5.3 Cost

Microchip Technology has developed the MPLAB SIM simulator to be the most cost-effective tool for debugging application firmware. The MPLAB SIM simulator does not require any external hardware to your PC, and in most respects operates exactly the same as the MPLAB ICE 2000 emulator. Unless you need to debug your application in real-time on your actual hardware, the MPLAB SIM simulator can usually be used to find and correct most coding errors.

## 2.5.4 Debugging Tool

The MPLAB SIM simulator is particularly suitable for optimizing algorithms. Unlike some emulators, the simulator makes many internal registers visible and can provide software tools that are difficult or expensive to implement in a hardware in-circuit emulator. For the most part, the MPLAB SIM simulator can be used to fully debug your system unless you run into real-time issues or peripheral device situations where an in-circuit emulator is required.

## 2.6 Simulator Considerations

The MPLAB SIM simulator executes on instruction cycle boundaries, and resolutions shorter than one instruction cycle ( $T_{CY}$ ) cannot be simulated. The MPLAB SIM simulator is a discrete-event simulator where all stimuli are evaluated, and all responses are generated, at instruction boundaries or  $T_{CY} = 4 T_{OSC}$ , where  $T_{OSC}$  is the input clock period. Therefore some physical events cannot be accurately simulated. These fall into the following categories:

- Purely asynchronous events
- Events that have periods shorter than one instruction cycle

In summary, 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 list itemizes the functions and peripherals among the entire PICmicro MCU family of microcontrollers that are affected by simulation on instruction cycle boundaries:

- Clock pulse inputs smaller than one cycle can not 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 can not be used.
- The oscillator waveform on RC0/RC1 pins can not be shown.
- MPLAB SIM simulator does not simulate serial I/O.

## 2.7 Break and Trace Points

The debug functions affect execution of program instructions based upon the following elements:

- Break Points
- Trace Points
- Pass Counter Addresses

MPLAB IDE limits the number of named address ranges to a maximum of 16 in each dialog.

Trace points and break points function totally independent of each other, and you can set them at any program memory location.

The following figures show the dialog boxes for assigning names to address ranges. Access the Break Point Settings dialog through the *Debug > Break Settings* menu item, and the Trace Point Settings dialog through the *Debug > Trace Settings* menu item.

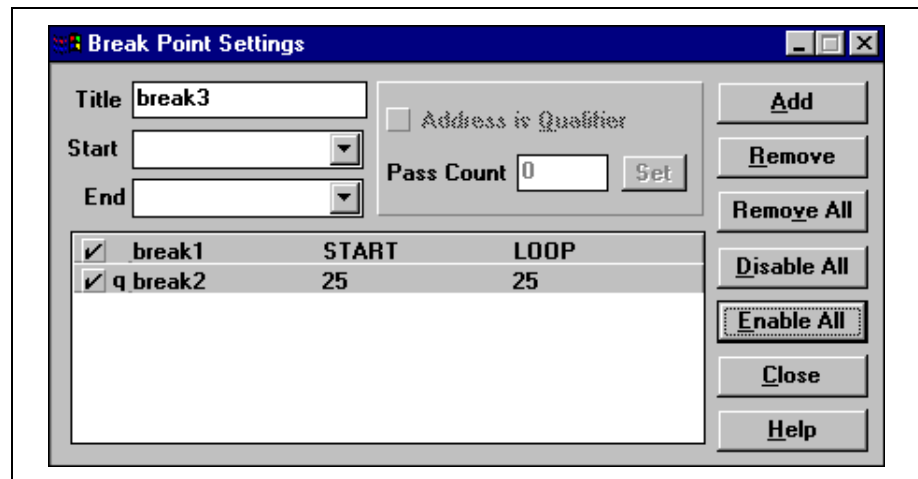


Figure 2.1: Break Point Settings Dialog

**Note:** When using the MPLAB ICD, you can only set one break point address, and no pass count may be set. When using MPLAB ICE 2000, use the Complex Trigger dialog to define any break points that require a pass counter.

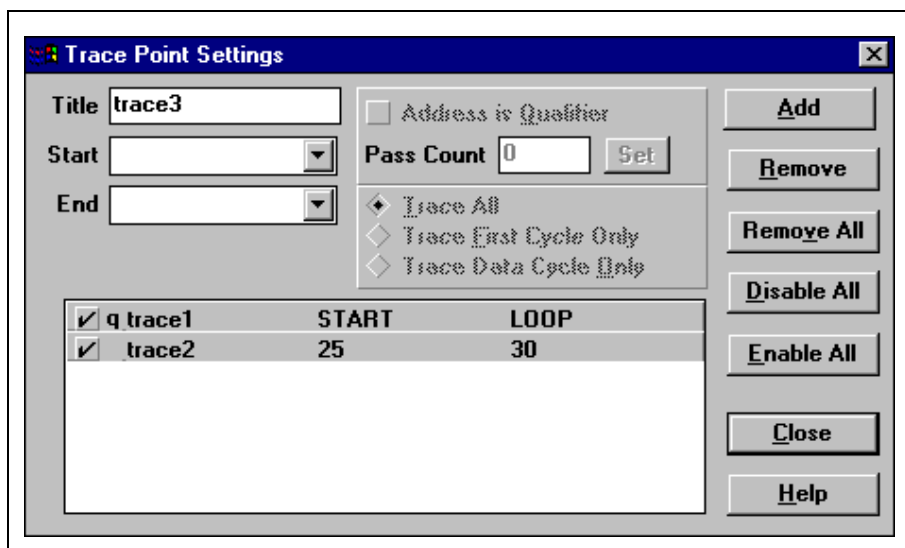


Figure 2.2: Trace Point Settings Dialog

**Note:** The Trace Point Settings dialog is not available in MPLAB ICE 2000 or MPLAB ICD. The MPLAB ICE 2000 trace may be configured through the Complex Trigger dialog.

## 2.7.1 Real-Time Break Points

A break point is a condition in which the processor executes code and halts after a certain condition is met.

**Note:** If execution doesn't halt at the break point, select *Options > Development Mode* and click the **Break Options** tab. Make sure that Global Break Enable is selected (check marked).

MPLAB IDE provides the following ways to set a break point:

- Break on Address Match
- Break on Trace Buffer Full
- Break on Pass Count Reached
- Break on Stack Overflow
- Break on Watchdog Timer Time-out
- User Halt

# Debugging and MPLAB SIM Simulator

The Program Memory Window shown in Figure 2.3 shows the following information:

- B Break Points
- T Trace Points
- Q Pass Counter Addresses

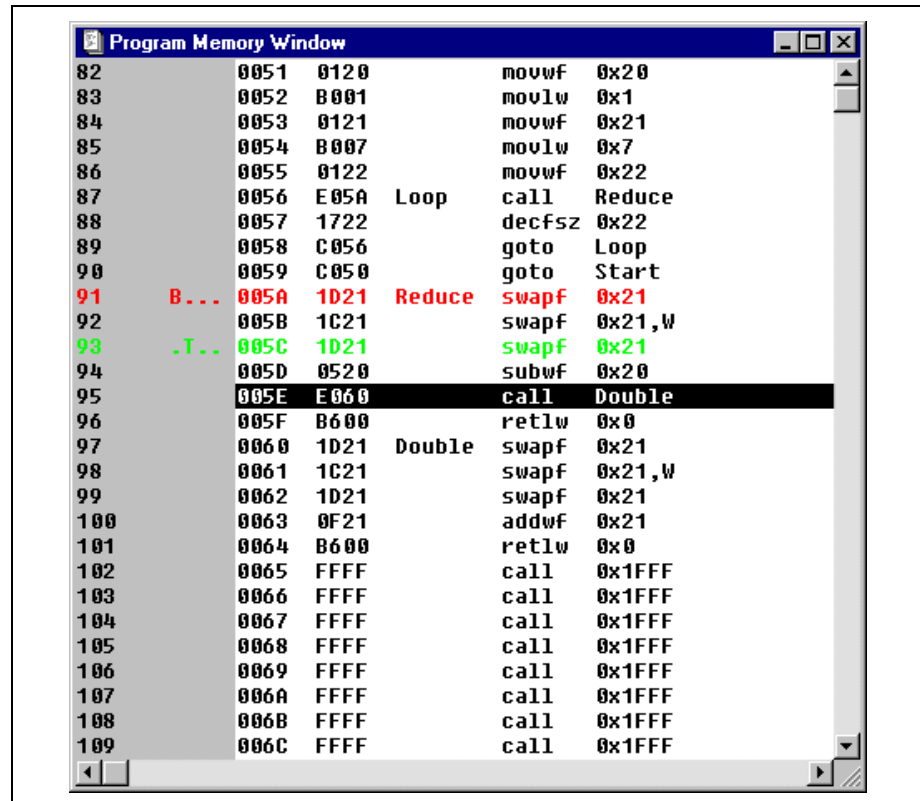


Figure 2.3: Program Memory Window

## 2.7.1.1 Break On Address Match

Break on Address Match allows you to halt the processor when the processor program counter equals a certain value. The processor breaks before the valid instruction is executed. For example, if a break point is set at address 5Ah, then the processor breaks before executing the instruction at address 5Ah.

## 2.7.1.2 Break On Trace Buffer Full

MPLAB IDE can be set to halt the processor after capturing 8K selected cycles (when the trace buffer is full).

## 2.7.1.3 Break On Pass Counter Equal to Predefined Value

MPLAB IDE has a Pass Counter switch that you can assign to either trace logic or break logic. The Pass Counter can be used to break or trace after the processor executes an address a predefined number of times.

For example, if the Pass Counter is assigned to break logic, then when the pass counter decrements to zero, the Pass Counter acts as a break point and halts the processor.

## 2.7.1.4 Break On Stack Overflow

Break on Stack Overflow causes MPLAB IDE to execute a break when the stack overflows.

## 2.7.1.5 Break On Watchdog Timer

If enabled, MPLAB IDE executes a break when a Watchdog Timer time-out generates a device RESET.

## 2.7.1.6 User Halt

MPLAB IDE provides three ways to stop at a break point any time the processor is running:

- Click *Debug > Run > Halt*.
- Click **F5**.
- Click the Halt Icon (red stop light).

## 2.7.2 Real-Time Trace Points

A trace is a function that logs program execution. The MPLAB SIM simulator has an 8K real-time trace buffer that logs addresses and opcodes as they execute. This circular trace buffer continues logging data after the buffer is full, losing the oldest data (unless you have selected Break on Trace Buffer Full in the Break Options tab of the Development Mode dialog).

### 2.7.2.1 Circular Trace Buffer

MPLAB IDE continuously captures selected bus cycles into the trace buffer.

The status information captured into the trace buffer is grouped as follows:

- 16 Bits of Address
- 16 Bits of Opcode/Data
- Time Stamp and Changed Registers



# Debugging and MPLAB SIM Simulator

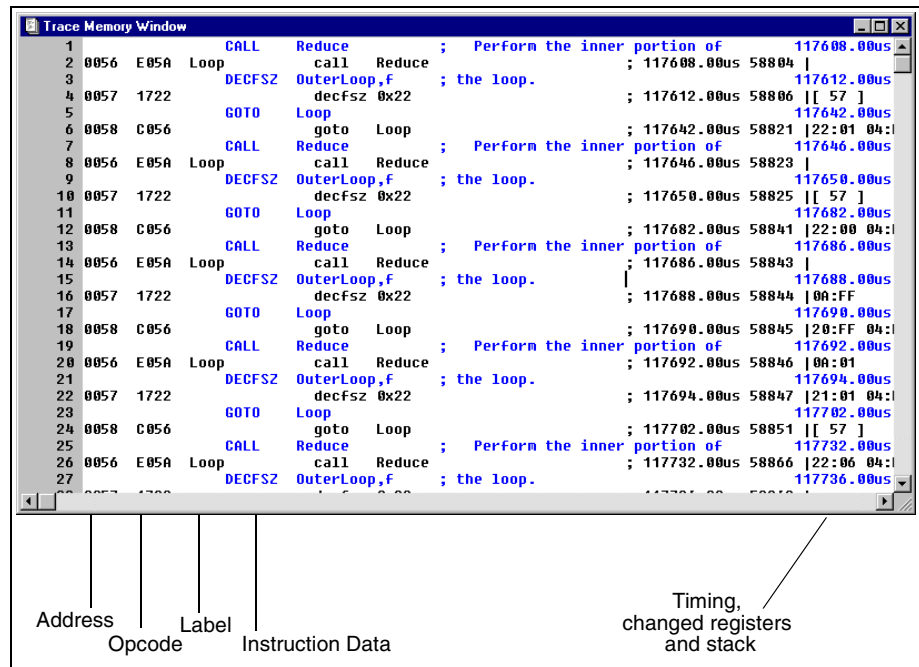


Figure 2.4: Trace Memory Window

## 2.7.2.2 Halting Trace from the Toolbar

Halt Trace allows you to take a snapshot of the trace buffer and look at the captured trace without halting the processor. In the toolbar, click **Halt Trace** to display a snapshot of the trace buffer without halting the processor. Once the trace buffer is halted, click **Halt Trace** again to take another trace snapshot.

## 2.7.2.3 MPLAB SIM Simulator Trace Display

The trace window can be used to collect executed instructions from the MPLAB SIM simulator. The trace will show the program memory address, executed code, time stamp and changes to registers. The time stamp uses the same data as the MPLAB IDE Stop Watch. You can reset the time stamp by resetting the Stop Watch.

## 2.7.3 Assigning a Pass Count to Break or Trace Points

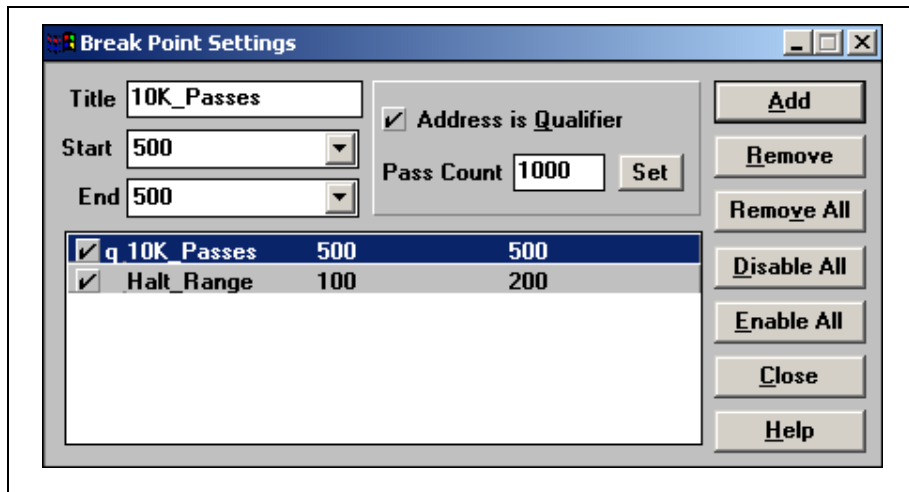
MPLAB IDE's 16-bit Pass Counter decrements by one on any address match in program memory.

When the processor is in a Halt state, you can modify the count value for the pass counter in the Break Point Settings or Trace Point Settings dialog box. To set up the Pass Counter, first set the desired address ranges, then load the counter with a desired count value (up to 16 bits). When the counter decrements to zero, the emulator will halt.

**Note:** The pass counter does not automatically reset. You must select **Set**.

## 2.7.3.1 Pass Counter Assigned to Break

If the Pass Counter is assigned to Break, the processor halts upon encountering a break point (either internal or external conditions) or when the Pass Counter reaches zero.



**Figure 2.5: Break Point Settings Dialog – Pass Counter**

**Example 2.1:** This example shows break points and pass counter addresses used in the same code (Figure 2.5). Keep in mind that break points and pass counter addresses are independent of each other.

1. Set up a named break point range from address 100 to 200.
  - Type Halt\_Range in Title box.
  - Type 100 in the Start box and 200 in the End box.
  - Click **Add** to enter the break point.
2. Set a Pass Counter Address at 500.
  - Type 10K\_Passes in Title box.
  - Type 500 in the Start box and in the End box.
  - Click **Add** to enter the break point.
3. Load the Pass Counter with a value of 1000.
  - Select (click on) the 10K\_Passes break point.
  - Click on the check box of the now-ungrayed Address is Qualifier.
  - Type 1000 in the Pass Count box and click **Set**.

The processor halts if it executes any instructions within the address range 100 to 200 or after executing 1000 instructions at address 500.

# Debugging and MPLAB SIM Simulator

## 2.7.3.2 Pass Counter Assigned to Trace

If the Pass Counter is assigned to trace, then the real-time trace buffer does not capture data until the Pass Counter decrements to zero. When the pass counter decrements to zero, the trace buffer starts capturing data on valid cycles.

## 2.7.3.3 Using Pass Counter to Count Events

The Pass Counter decrements each time an event occurs. You can use this feature to count the number of times an event happens.

## 2.8 Conditional Break Dialog

When a conditional break is set, MPLAB IDE halts when the value of a specified internal register reaches a preset value or condition.

Access the Conditional Break dialog through the *Debug > Execute > Conditional Break* menu item.

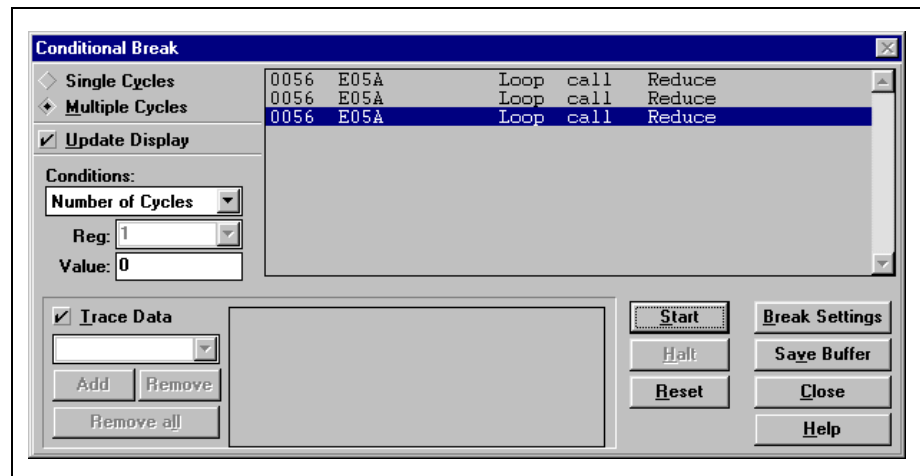


Figure 2.6: Conditional Break Dialog

**Note:** The Conditional Break dialog is not available in MPLAB ICE 2000 or MPLAB ICD development modes. For MPLAB ICE 2000, refer to the Complex Trigger dialog for this functionality.

# MPLAB<sup>®</sup> IDE User's Guide

---

## 2.8.1 Conditions

MPLAB IDE will stop at a break point in the Conditional Break dialog based on one of the following conditions:

- User Halt – MPLAB IDE executes until you click the **Halt** button on the Conditional Break dialog.
- Number of Cycles – MPLAB IDE halts after the target processor executes the specified number of cycles.
- Logic Condition satisfied.

## 2.8.2 Trace Data

Trace Data allows you to track the value of the registers in the Conditional Break dialog.

## 2.8.3 Single Cycle

In the Single Cycle mode, MPLAB IDE single steps the processor until the condition is met.

## 2.8.4 Multiple Cycles

In Multiple Cycle mode:

- Conditional Break executes instructions in real-time (in the emulator), halts at user selected break points, checks the specified condition, and continues executing instructions in real-time. The emulator or simulator only stops when meeting the specified condition.
- Break points and register conditions are only checked at the break points you specify in the Break Settings dialog.

## 2.9 Stimulus Functions

The stimulus generates signals for the simulator. You can set pins high or low, and inject values directly into registers. The four stimulus modes are:

- Asynchronous Stimulus – An interactive dialog to control signals on input pins
- Stimulus Pin File – The contents of a text file describe signals to input pins
- Stimulus Register File – The contents of a text file are used to set 8-bit values directly into a register
- Clock Stimulus – A regular, programmable, periodic source of stimulus pulses

# Debugging and MPLAB SIM Simulator

## 2.9.1 Asynchronous Stimulus Dialog

This stimulus feature provides a dialog button to simulate +5 and 0 volts being applied to input pins. As your program executes with the simulator, you can click buttons on this dialog to change levels on pins.

As an example, we'll set up a signal that will toggle the level on a pin on I/O portb of the PIC16F84.

Select *Debug > Simulator Stimulus > Asynchronous Stimulus*. This dialog will be displayed:

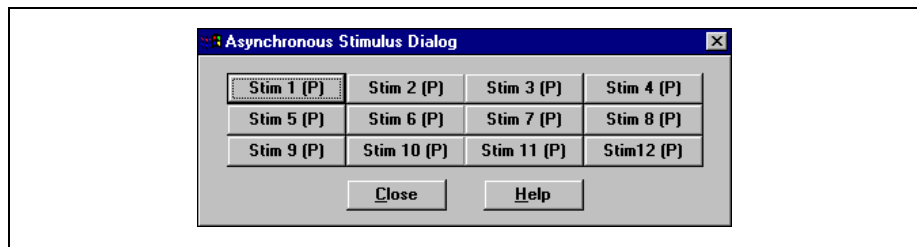


Figure 2.7: Asynchronous Stimulus Dialog

Place the cursor over the button labelled “Stim 1 (P)” and click the right mouse button. A shortcut menu will appear. Scroll down and select *Toggle*.

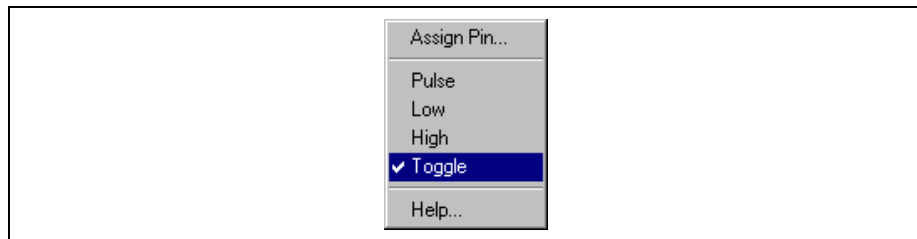
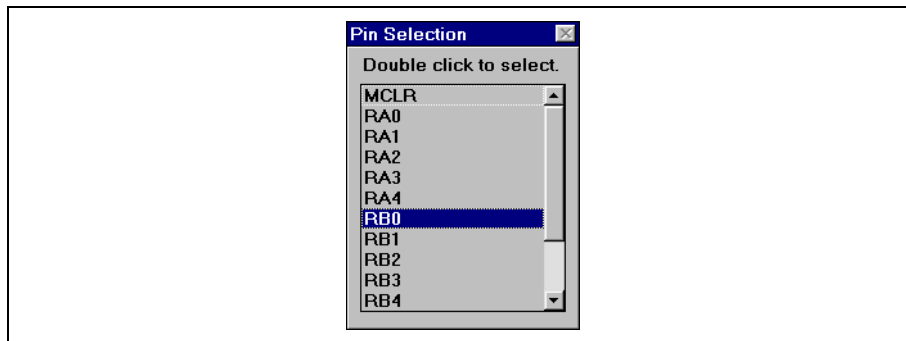


Figure 2.8: Toggle Option

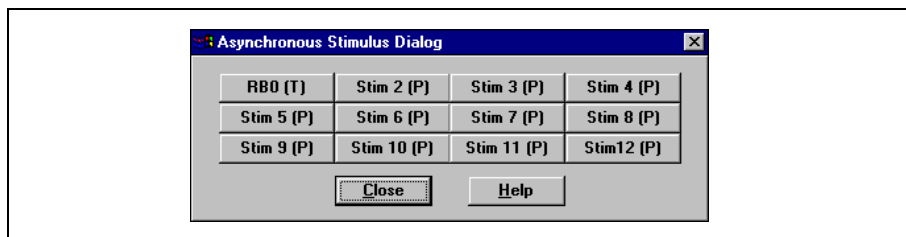
Again put the cursor over the button now labelled “Stim1 (T)” (the “P” was replaced by a “T,” meaning “Toggle”), click the right mouse button and select *Assign Pin* from the shortcut menu.

A dialog will display a list of pins on the PIC16F84.



**Figure 2.9: Pin Selection**

Put the cursor over “RB0” and double-click. The Asynchronous Stimulus dialog should now look like Figure 2.10. Note that the button now shows “RB0 (T).”



**Figure 2.10: Asynchronous Stimulus Dialog – RB0(T)**

Choose *Debug > Run > Animate* to get the processor running in a “fast single step” mode. The status bar will alternate run to stop very fast.

Click the “RB0 (T)” button on the Asynchronous Stimulus dialog. You should see the value of portb in the Special Function Register window change as you repeatedly click the button to simulate a high signal then a low signal applied to portb pin 0.

## 2.9.2 Pin Stimulus Files

A Pin Stimulus file consists of columns of input ones and zeros that will be applied to pins when the “Cycle” value in the Stopwatch matches the CYCLE column.

### 2.9.2.1 Creating a Pin Stimulus file

1. Select *File > New File*. An Untitled window will appear on your desktop. You will create the Pin Stimulus file in this window.

# Debugging and MPLAB SIM Simulator

2. Type the word CYCLE in the first line of the Untitled file window.

**Note:** For backward compatibility with earlier versions of the simulator, the first line must always start with the word CYCLE or STEP. The first column specifies the CYCLE (as determined by MPLAB IDE's Stopwatch window) where the values in the other columns will be applied.

3. To the right of the word CYCLE, type the pin name for the PICmicro MCU pin that will receive the high stimulus value. The third item on the first line of the file should be the pin to receive the low stimulus value. These names must match the Microchip PICmicro MCU pin names for the processor being simulated. You can add additional columns for additional pins.

**Note:** To see a list of supported pins, select *Debug > Simulator Stimulus > Asynchronous* and right-click on a stimulus button.

4. In the remaining lines of the file, type the cycles during which the pins are to receive the stimuli, followed by the high and low values. You can put comments on a line using the “;” or “!” character preceded and followed by at least one space.
5. Select *File > Save As...* to save your file. Select the drive and folder in which the file is to be stored, and enter the file name you would like to assign it. Give the file a `.sti` extension. Your file is now ready to use.

## 2.9.2.2 Using a Pin Stimulus file

1. Select *Debug > Simulator Stimulus > Pin Stimulus > Enable* and select the appropriate file to enable the pin stimulus file.
2. Open the Stopwatch window by selecting *Window > Stopwatch*. Also select *Window > Special Function Registers*. Watch the port the pins are on. Or, simply add the port that the pins are on to a Watch window. The Stopwatch window will show the elapsed time at each instruction, as determined from the CYCLE value and the clock frequency. If the Stopwatch is reset to 0, the pin stimulus file will also be effectively reset.
3. Reset and single step. The port will change its value as set in your stimulus file.

## 2.9.2.3 Pin Stimulus File example

**Note:** This example assumes that you have completed the simple project tutorial in Chapter 3.

1. Select *File > New File* and type in the following text. You do not have to type in the text after the “;” and “!” comment delimiters, but it is a good idea to include them in this file.

# MPLAB® IDE User's Guide

```
CYCLE   RB1   RB0
20      0     0
41      1     0   ; apply high to port b bit 1
52      0     1   ; apply high to port b bit 0, set bit 1 low
55      1     1
60      0     0
65      1     0   ; toggle bit 1, then...
76      0     1   ! ...toggle bit 0.
```

After the word CYCLE in the first line of the file are the pin names for the PICmicro MCU pins that will receive the high and low stimulus values. In this example pins RB1 and RB0, two inputs on Port B, will receive stimulus inputs.

**Note:** For backward compatibility with earlier versions of the simulator, the first line must always start with the word CYCLE or STEP. The first column specifies the CYCLE (as determined by MPLAB IDE's Stopwatch window) where the values in the other columns will be applied.

In this file, the second column contains values that will be applied to RB1 (PortB bit 1) and the third column has values for RB0 (PortB bit 0). These names must match the Microchip PICmicro MCU pin names for the processor being simulated. To see a list of all supported pins, right-click on a stimulus button and look at the pin assignment pull-down list for the Asynchronous Stimulus.

2. Select *File > Save As* to save as `tutor84.sti`.
3. Select *Debug > Simulator Stimulus > Pin Stimulus > Enable* to enable the pin stimulus file.

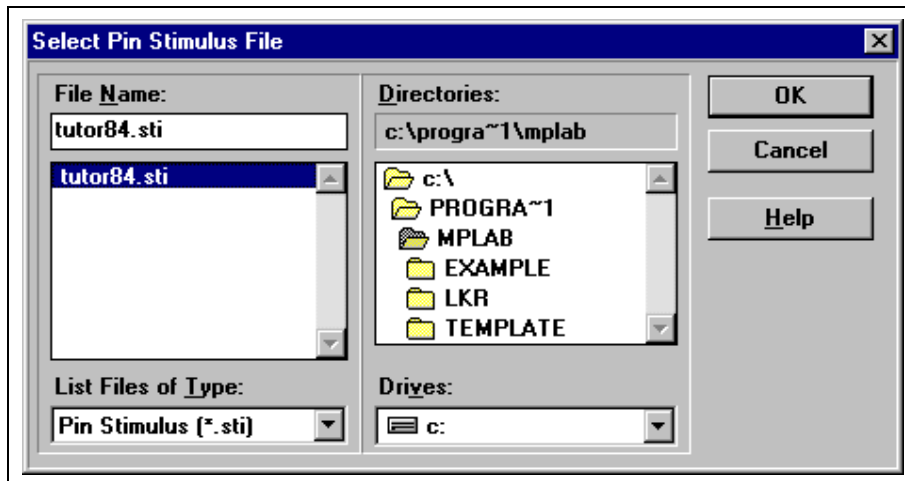


Figure 2.11: Pin Stimulus Enable



# Debugging and MPLAB SIM Simulator

4. Open the Stopwatch window by selecting *Window > Stopwatch*. Also select *Window > Special Function Registers*. Watch `Portb`. Or, simply add `Portb` to a Watch window.

The Stopwatch window will also show the elapsed time at each instruction, as determined from the CYCLE value and the clock frequency. If the Stopwatch is reset to 0, the pin stimulus file will also be effectively reset.

5. Reset and single step until you execute 41 cycles. `Portb` will change its value as set in the second line of the stimulus file.

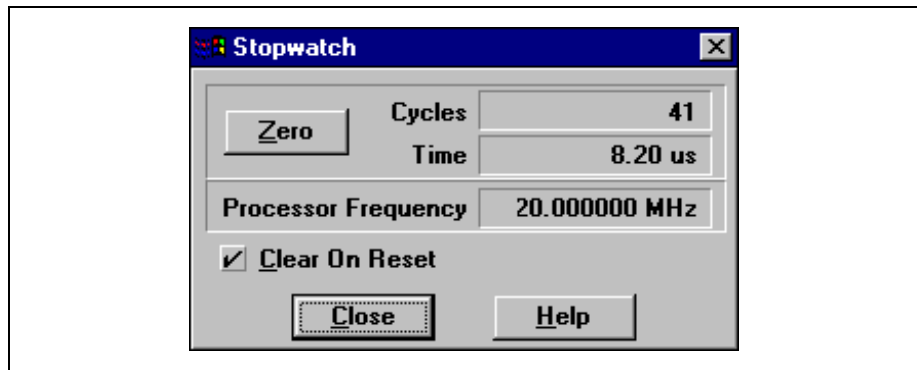


Figure 2.12: Stopwatch Window (41 Cycles)

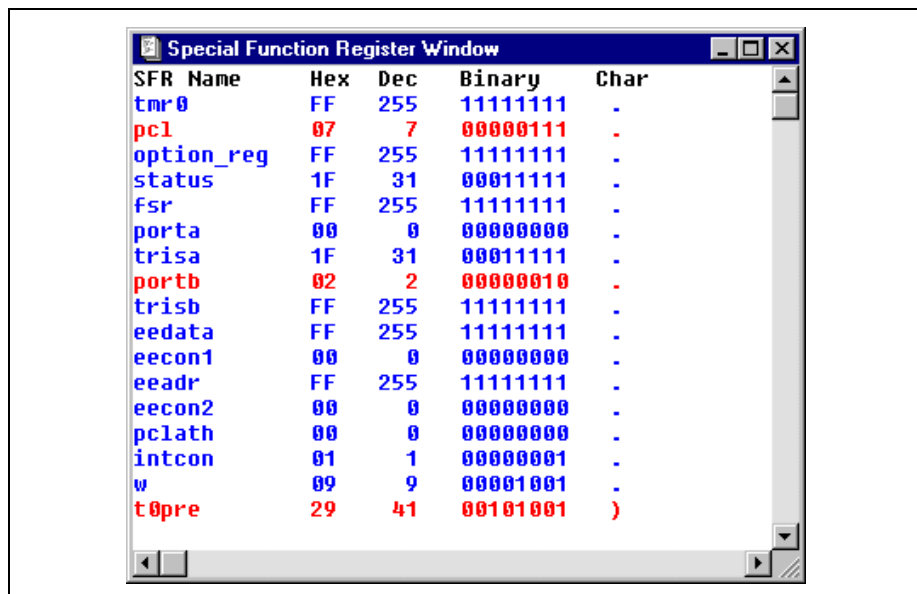


Figure 2.13: Special Function Registers Window

## 2.9.3 Register Stimulus Files

A Register Stimulus file consists of a single column of values that will be sent to a register when the program memory address reaches the location set in the Register Stimulus Dialog. This is useful for simulating an A/D conversion operation.

### 2.9.3.1 Creating a Register Stimulus file

1. Select *File > New File*. An Untitled window will appear on your desktop. You will create the Register Stimulus file in this window.
2. In the Untitled window, type the list of values you wish to insert in a register. Be sure to type them in the order you wish to have them inserted into the register.
3. Select *File > Save As* to save your file. Select the drive and folder in which the file is to be stored, and enter the file name you would like to assign it. Give the file a `.reg` extension.

### 2.9.3.2 Using a Register Stimulus file

1. Select *Debug > Simulator Stimulus > Register Stimulus > Enable* to open the Register Stimulus dialog.
2. In the Program Memory Address box, enter the address in the program where the stimulus values are to be injected.
3. In the Register Address box, select the file register address where the values are to be injected.
4. Select *Window > File Registers* to open the File Registers window and see the effect of your stimulus.
5. Reset and then single step the processor. Every time you reach the program memory address, the value in the file register at the register address you specified will change. The list of values in your register stimulus file will be sequentially injected into the selected file register.

After the last value in your Register Stimulus file is injected, the first value will be used again. The list will cycle as long as the MPLAB SIM simulator executes.

# Debugging and MPLAB SIM Simulator

## 2.9.3.3 Register Stimulus example

**Note:** This example assumes that you have completed the simple project tutorial in Chapter 3.

1. Create a new file with *File > New File* and type in the following list of numbers:  
10  
2E  
38  
41  
50  
7A  
99  
A0  
FD
2. Select *File > Save As* to save the file and name it `tutor84.reg`. This file will be used to sequentially inject these values into a register.
3. Select *Debug > Simulator Stimulus > Register Stimulus > Enable*, then set "loop" to be the place in the program when values are injected, and for demonstration purposes let's inject them into the file register at address `0x0d`. After you set `loop` and `0d` in the appropriate boxes, click *Browse* to bring up the file dialog, and then select `tutor84.reg` as the register stimulus file.

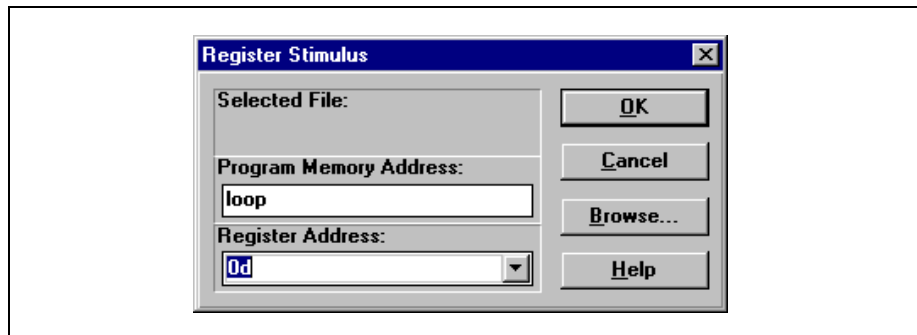


Figure 2.14: Register Stimulus Dialog

# MPLAB® IDE User's Guide

4. Select *Window > File Registers* to open the File Registers window and see the effect of this stimulus.

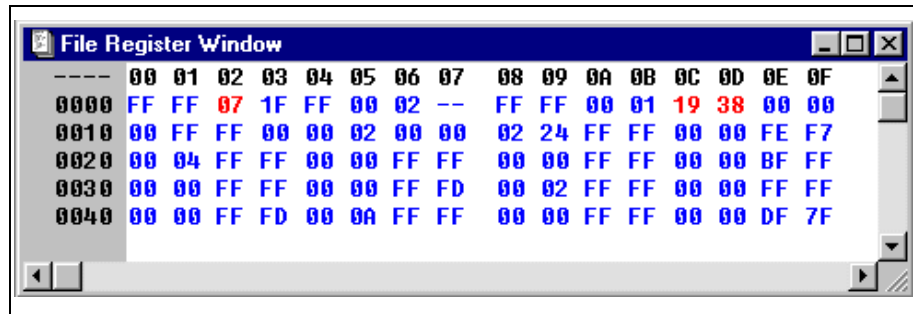


Figure 2.15: File Register Window

5. Reset and then single step the processor. Every time you get to loop the value in the file register at address 0x0D will change. The list of values in `tutor84.reg` will be sequentially injected into the selected file register. Values of 0x10, 0x2E, etc. will be injected into the register selected in the *Debug > Simulator Stimulus > Register Stimulus* dialog every time loop is executed.

After the last value is injected (0xFD in `tutor84.reg`), the first value will be used again (0x10). The list will cycle as long as the MPLAB SIM simulator executes.

## 2.9.4 Clock Stimulus

The clock stimulus generates a regular waveform on a pin with a duty cycle that is specified in terms of the processor clock cycles.

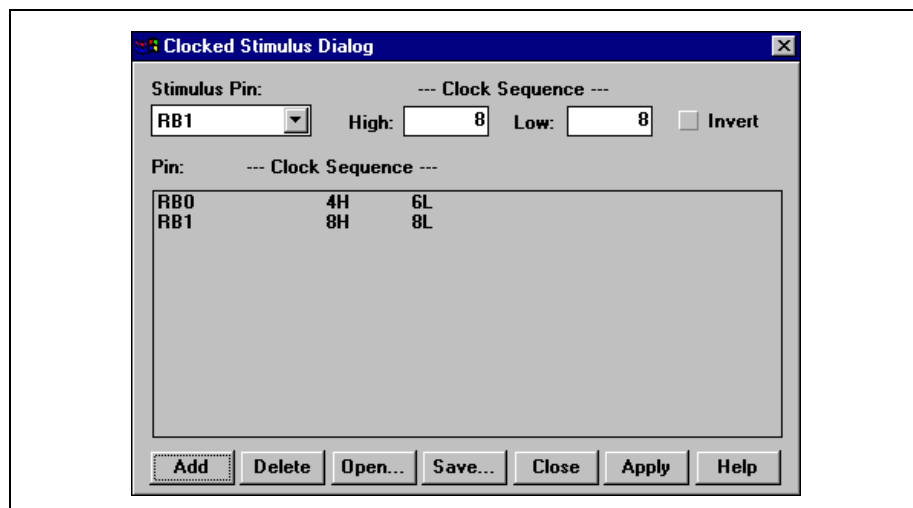


Figure 2.16: Clock Stimulus

# Debugging and MPLAB SIM Simulator

---

Select *Debug > Simulator Stimulus > Clock Stimulus* and enter clock sequences for various stimulus clocks in the Clocked Stimulus dialog. These settings will repeat until you exit MPLAB IDE or delete them using the Clocked Stimulus dialog.

When you step or run using the settings of Figure 2.16, RB0 will be high for 4 clock cycles, then go low for 6 clock cycles. RB1 will go high for 8 clock cycles then low for 8 clock cycles.

To add a stimulus, select the pin in the Stimulus Pin pull down list, set the high and low clock sequence, and click **Add**.

To delete a stimulus, highlight it by clicking on it and then click **Delete**.

## 2.10 12-Bit Core Device Simulator Issues

This section discusses I/O pins, interrupts, registers, peripherals, modes, and conditions for using 12-Bit Core devices.

### 2.10.1 12-Bit Core Devices

See the file `README.SIM` for a list of devices supported by the MPLAB SIM simulator.

### 2.10.2 I/O Pins

When modifying pins either manually or via the stimulus file, use the following pin names only. These are the only ones that the MPLAB SIM simulator recognizes as valid I/O pins. Because the pinout is device-specific, some pins (for example RC0 on a PIC16C54) are not available on all parts in this family.

- $\overline{\text{MCLR}}$
- T0CKI
- RA0-RA3
- RB0-RB7
- RC0-RC7

These pin names can be used in the Modify window (*Window > Modify*) and in stimulus files.

### 2.10.3 CPU Model

#### 2.10.3.1 Reset and Sleep Conditions

All RESET conditions are supported by the MPLAB SIM simulator.

A  $\overline{\text{MCLR}}$  Reset during normal operation or during SLEEP can easily be simulated by driving the  $\overline{\text{MCLR}}$  pin low (and then high) via the stimulus file or by using *Debug > Run > Reset*.

A WDT Time-out Reset is simulated when WDT is enabled and proper prescaler is set (by initializing OPTION register appropriately) and WDT actually overflows. WDT Time-out period (with prescale = 1) is approximated at 18 ms (to closest instruction cycle multiple).

The Time-out ( $\overline{\text{TO}}$ ) and Power-down ( $\overline{\text{PD}}$ ) 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.

# Debugging and MPLAB SIM Simulator

---

## 2.10.3.2 Watchdog Timer

The Watchdog timer is fully simulated in the MPLAB SIM simulator. Because it is configuration bit-selectable on the device, it must be enabled from the Configuration tab of the Development Mode dialog, accessed by *Options > Development Mode* in the MPLAB SIM simulator. The period of the WDT is determined by the prescaler settings in the OPTION register. The basic period (with prescaler = 1) is approximated at 18 ms (to closest instruction cycle multiple).

## 2.10.4 Peripherals

Along with providing core support, the TIMER0 timer/counter module is fully supported in both internal and external clock modes. The prescaler is made readable and writable as the 'TOPRE' symbol.

**Note:** Because the MPLAB SIM simulator executes on instruction cycle boundaries, resolutions below 1 Tcy cannot be simulated.

## 2.11 14-Bit Core Device Simulator Issues

This section discusses I/O pins, interrupts, registers, peripherals, modes, and conditions for using 14-Bit Core devices.

### 2.11.1 14-Bit Core Devices

See the file `README.SIM` for a list of devices supported by the MPLAB SIM simulator.

### 2.11.2 I/O Pins

The 14-Bit Core devices have I/O pins multiplexed with other peripherals (and therefore referred by more than one name). When modifying pins either manually or via the stimulus file, use the following pin names only. These pin names are the only ones that the MPLAB SIM simulator recognizes as valid I/O pins. (Pins are available only as described in the data sheet of the specific device.)

- $\overline{\text{MCLR}}$
- RA0-RA5
- RB0-RB7
- RC0-RC7
- RD0-RD7
- RE0-RE7

These pin names can be used in the Modify window (*Window > Modify*) and in stimulus files.



## 2.11.3 Interrupts

The MPLAB SIM simulator supports all interrupts of 14-Bit Core devices. (Peripherals are available only as described in the data sheet of the particular device.)

- Timer0 overflow
- Timer1 overflow
- Timer2
- CCP1
- CCP2
- SSP (in SPI mode ONLY)
- Change on Port RB <7:4 >
- External interrupt from RB0/INT pin
- Parallel Slave Port
- Comparators
- A/D complete
- EEPROM write complete

## 2.11.4 CPU Model

### 2.11.4.1 Reset Conditions

All RESET conditions are supported by the MPLAB SIM simulator.

A  $\overline{\text{MCLR}}$  Reset during normal operation or during SLEEP can easily be simulated by driving the  $\overline{\text{MCLR}}$  pin low (and then high) via the stimulus file or by using MPLAB IDE *Debug > Run > Reset*.

The Time-out ( $\overline{\text{TO}}$ ) and Power-down ( $\overline{\text{PD}}$ ) 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.

### 2.11.4.2 Sleep

The MPLAB SIM simulator simulates the SLEEP instruction, and will appear “asleep” until a wake-up from sleep condition occurs. For example, if the Watchdog timer has been enabled, it will wake up the processor from sleep when it times out (depending upon the prescaler setting in the OPTION register).

Another 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.

## 2.11.4.3 Watchdog Timer

The Watchdog timer is fully simulated in the MPLAB SIM simulator. Because it is configuration bit-selectable on the device, it must be enabled from the Configuration tab of the Development Mode dialog, accessed by *Options > Development Mode* in the MPLAB SIM simulator. The period of the WDT is determined by the prescaler settings in the OPTION register. The basic period (with prescaler = 1) is approximated at 18 ms (to closest instruction cycle multiple).

## 2.11.5 Special Registers

To aid in debugging this device, certain items that are normally not observable have been declared as “special” registers. Prescalers and postscalers cannot be declared in your code as “registers,” so special labels appear in the Special Function Registers window.

The following are special symbols that are available for the processors in the 14-bit core family. (Consult the data sheet for the particular device you are using to see which symbols are implemented.)

- T0PRE – Prescaler for timer0
- T1PRE – Prescaler for timer1
- T2PRE – Prescaler for timer2
- T2POS – Postscaler for timer2
- CCP1PRE – Prescaler for CCP1
- SPIPRES – Prescaler for SPI
- SSPSR – SSP Shift register

## 2.11.6 Peripherals

### 2.11.6.1 Peripherals supported

Along with providing core support, the following peripheral modules (in addition to general-purpose I/O) are supported. (Consult the data sheet for the particular device you are using to see which symbols are implemented.)

- Timer0
- Timer1
- Timer2
- CCP1
- CCP2
- Parallel Slave Port
- SSP (in SPI Mode only)
- Comparators
- A/D (Limited)

### 2.11.6.2 TIMER0

Timer0 (and the interrupt it can generate on overflow) is fully supported by the MPLAB SIM simulator, 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. The prescaler for Timer0 is made accessible as T0PRE.

### 2.11.6.3 TIMER1

Timer1 in its various modes is supported by the MPLAB SIM simulator, except when running in counter mode by an external crystal. The MPLAB SIM simulator supports timer1 interrupts generated on overflow, and interrupts generated by wake-up from sleep. The prescaler for Timer1 is viewable as T1PRE in the Special Function Registers window. The external oscillator on RC0/RC1 is not simulated, but a clock stimulus can be assigned to those pins.

### 2.11.6.4 TIMER2

Timer2 and the interrupt that can be generated on overflow are fully supported by the MPLAB SIM simulator, and both the prescaler and postscaler for Timer2 are viewable as T2PRE and T2POS.

### 2.11.6.5 CCP1 and CCP2

#### CAPTURE

The MPLAB SIM simulator fully supports capture and the interrupt generated. The prescaler for the CCP module is viewable CCP1PRE.

# MPLAB<sup>®</sup> IDE User's Guide

---

## **COMPARE**

Compare mode, its interrupt, and the special event trigger (resetting Timer1 by CCP1) are supported in this version of the MPLAB SIM simulator.

## **PWM**

PWM output (resolution greater than 1 T<sub>CY</sub> only) are not supported in this version of the MPLAB SIM simulator.

### **2.11.6.6 SSP**

The Synchronous Serial Port is supported in SPI<sup>™</sup> mode only. The shift register (SSPSR) can be added to the view screen, observed, and modified. The MPLAB SIM simulator currently does not support the I<sup>2</sup>C<sup>™</sup> mode.

### **2.11.6.7 A/D Converter**

All the registers, timing function, and interrupt generation are implemented. However, the simulator does not load any meaningful value into A/D result register (ADRES) at the end of a conversion.

### **2.11.6.8 EEPROM Data Memory**

The EEPROM data memory (for PIC16F8X devices) 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.

## 2.12 16-Bit Core Device Simulator Issues

This section discusses I/O pins, interrupts, registers, peripherals, modes, and conditions for using 16-Bit Core devices.

### 2.12.1 16-Bit Core Devices

See the file `README.SIM` for a list of devices supported by the MPLAB SIM simulator.

### 2.12.2 I/O Pins

The 16-Bit Core devices have I/O pins multiplexed with other peripherals (and therefore referred by more than one name). When modifying pins either manually or via the stimulus file, use the following pin names only. These are the only ones that the MPLAB SIM simulator recognizes as valid I/O pins:

- MCLR
- RA0-RA5
- RB0-RB7
- RC0-RC7
- RD0-RD7
- RE0-RE2

These pin names can be used in the Modify window (*Window > Modify*) and in stimulus files.

### 2.12.3 Interrupts

The MPLAB SIM simulator supports all interrupts on 16-Bit Core devices:

- 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

## 2.12.4 CPU Model

### 2.12.4.1 Reset Conditions

All RESET conditions are supported by the MPLAB SIM simulator.

A  $\overline{\text{MCLR}}$  Reset during normal operation or during SLEEP can easily be simulated by driving the  $\overline{\text{MCLR}}$  pin low (and then high) via the stimulus file, by clicking the **Reset** button on the toolbar or by selecting *Debug > Run > Reset*.

A WDT Time-out Reset is simulated when the WDT is enabled (*Options > Development Mode* dialog, Configuration tab), the proper prescaler is set, and the WDT actually overflows. WDT time-out period is approximated at 12 ms (to closest instruction cycle multiple) but can be changed by using the dialog.

The Time out ( $\overline{\text{TO}}$ ) and Power-Down ( $\overline{\text{PD}}$ ) bits in the ALUSTA register reflect appropriate RESET condition. This feature is useful for simulating various power-up and time-out forks in the user code.

### 2.12.4.2 Sleep

The MPLAB SIM simulator simulates the SLEEP instruction and will appear “asleep” until a wake-up from sleep condition occurs. For example, if the Watchdog timer has been enabled, it will wake up the processor from sleep when it times out. Another 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.

### 2.12.4.3 Watchdog Timer

The Watchdog Timer is fully simulated in the MPLAB SIM simulator. Because it is configuration bit-selectable and configuration bit-configurable on the device, it must be enabled from the Configuration tab of the Development Mode dialog, accessed by *Options > Development Mode* in the MPLAB SIM simulator. The basic period of the WDT (with prescaler = 1) is approximated at 12ms (to closest instruction cycle multiple).

## 2.12.5 Special Registers

To aid in debugging this device, certain items that are normally not observable have been declared as “special” registers. Prescalers cannot be declared in user code as “registers,” so the following special symbols are available in the Special Function Registers window:

- TOPRE (Prescaler for Timer0)
- WDTPRE (Prescaler for WDT)

## 2.12.6 Peripherals

### 2.12.6.1 Peripherals Supported

Along with providing core support, the following peripheral modules (in addition to general-purpose I/O) are supported:

- Timer0 in both internal and external clock modes
- Timer1 and Timer2 (and their respective period registers)
- Timer3
- Two Capture Modules
- Two PWM Modules

### 2.12.6.2 TIMER0

Timer0 (and the interrupt it can generate on overflow) is fully supported by the MPLAB SIM simulator, 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. The prescaler for Timer0 is made accessible as TOPRE. It can be watched and modified.

### 2.12.6.3 TIMER1 and TIMER2

Timer1 and Timer2 in its various modes is fully supported by the MPLAB SIM simulator. Delay 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.

### 2.12.6.4 TIMER3 and Capture

The MPLAB SIM 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.

### 2.12.6.5 PWM

Both PWM outputs are supported (resolution greater than 1 Tcy only) are supported in this version of the MPLAB SIM simulator.

## 2.12.7 Memory Modes

The following memory modes are supported by the MPLAB SIM simulator:

- Microcontroller Mode
- Extended Microcontroller Mode
- Microprocessor Mode

The default is Microcontroller mode. If you would like to use any of the other modes, you must use the Configuration tab of the Development Mode dialog, accessed by *Options > Development Mode* in the MPLAB SIM simulator.

The extended memory is only available on PIC17CXXX devices.



## 2.13 Enhanced 16-Bit Core Device Simulator Issues

This section discusses I/O pins, interrupts, registers, peripherals, modes, and conditions for using Enhanced 16-Bit Core devices (PIC18CXXX).

### 2.13.1 16-Bit Core Devices

See the file `README.SIM` for a list of devices supported by the MPLAB SIM simulator.

Note that this family of PICmicro MCU devices has a byte-organized program memory space rather than word addressed as in the other PICmicro MCU families. 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.

### 2.13.2 I/O Pins

The Enhanced 16-Bit Core devices have I/O pins multiplexed with other peripherals (and therefore referred to by more than one name). When modifying pins either manually or via the stimulus file, use the following pin names only. These are the only ones that the MPLAB SIM simulator recognizes as valid I/O pins:

- $\overline{\text{MCLR}}$
- RA0-RA5
- RB0-RB7
- RC0-RC7
- RD0-RD7
- RE0-RE2

These pin names can be used in the Modify window (*Window > Modify*) and in stimulus files.

## 2.13.3 Interrupts

The MPLAB SIM simulator supports all interrupts on Enhanced 16-Bit Core devices:

- 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

## 2.13.4 CPU Model

### 2.13.4.1 Reset Conditions

All RESET conditions are supported by the MPLAB SIM simulator.

A  $\overline{\text{MCLR}}$  Reset during normal operation or during SLEEP can easily be simulated by driving the  $\overline{\text{MCLR}}$  pin low (and then high) via the stimulus file, by clicking on the Reset button on the toolbar or selecting *Debug > Run > Reset*.

A WDT Time-out Reset is simulated when the watchdog timer is enabled in hardware (*Options > Development Mode, Configuration tab*) or disabled in hardware and enabled in software through the WDTCON register. When the Watchdog timer actually overflows during normal operation, the chip will either Reset and continue execution or will break (*Options > Development Mode, Configuration tab*). The Watchdog timer time-out period depends on the postscaler settings and is approximated at 18 ms (to closest instruction cycle multiple) when configured with no postscaler.

### 2.13.4.2 Sleep

The MPLAB SIM simulator simulates the SLEEP instruction and 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. Another 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.

## 2.13.4.3 Watchdog Timer

The Watchdog timer is fully simulated in the MPLAB SIM simulator. Because it is configuration bit-selectable and configuration bit-configurable on the device, it must be enabled from the Configuration tab of the Development Mode dialog, accessed by *Options > Development Mode* in the MPLAB SIM simulator. The basic period of the WDT (with prescaler = 1) is approximated at 12 ms (to closest instruction cycle multiple).

## 2.13.5 Special Registers

To aid in debugging this device, certain items that are normally not observable have been declared as “special” registers. Prescalers cannot be declared in user code as “registers,” so the following special symbols are available in the Special Function Registers window:

- TOPRE (Prescaler for Timer0)
- WDTPRE (Prescaler for WDT)

## 2.13.6 Peripherals

### 2.13.6.1 Peripherals Supported

Along with providing core support, the following peripheral modules (in addition to general-purpose I/O) are supported:

- Timer0 in both internal and external clock modes
- Timer1 and Timer2 (and their respective period registers)
- Timer3
- Two Capture Modules
- Two PWM Modules

The delays are implemented on all peripherals, but the interrupt latency is not.

### 2.13.6.2 TIMER0

Timer0 (and the interrupt it can generate on overflow) is fully supported by the MPLAB SIM simulator, 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 T<sub>CY</sub> and a minimum low time of 1 T<sub>CY</sub> due to the stimulus file requirements. The prescaler for Timer0 is made accessible as TOPRE. It can be watched and modified.

## 2.13.6.3 TIMER1 and TIMER2

Timer1 and Timer2 in its various modes is fully supported by the MPLAB SIM simulator. 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 T<sub>CY</sub> and a minimum low time of 1 T<sub>CY</sub> due to the stimulus file requirements.

## 2.13.6.4 TIMER3 and Capture

The MPLAB SIM 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 T<sub>CY</sub> and a minimum low time of 1 T<sub>CY</sub> due to the stimulus file requirements.

## 2.13.6.5 PWM

Both PWM outputs are supported (resolution greater than 1 T<sub>CY</sub> only) are supported in this version of the MPLAB SIM simulator.

---

---

**Chapter 1. MPLAB IDE Toolbar and Menu Options**

---

---

**1.1 Introduction**

This chapter gives detailed information on using the MPLAB IDE desktop toolbars and menu options. The chapter organization follows the entries on the pull-down menus.

**1.2 Highlights**

This chapter will discuss the following:

- MPLAB IDE Desktop
- File Menu
- Project Menu
- Edit Menu
- Debug Menu
- Programmer Menu
- Options Menu
- Tools Menu
- Window Menu
- Help Menu

# MPLAB<sup>®</sup> IDE User's Guide

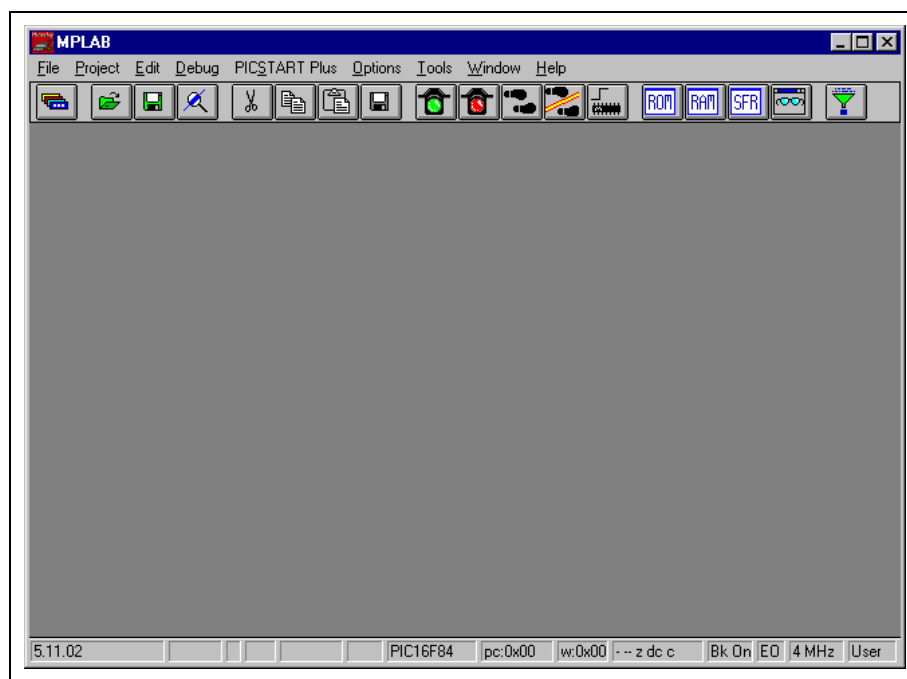
---

## 1.3 MPLAB IDE Desktop

The MPLAB IDE desktop is a resizable window that operates independently of the rest of the menu items.

To reduce the size of the MPLAB IDE window on your desktop, click the Maximize button in the upper right corner of the desktop. To maximize the size of the MPLAB IDE window again, click the Maximize button again. The next time you start MPLAB IDE, the MPLAB IDE window will automatically open in the size you last set it to. When not using MPLAB IDE, you can iconize the window by clicking the Minimize button. MPLAB IDE will remain in your computer's memory, but the desktop will be free for you to use.

Figure 1.1 shows a maximized desktop.



**Figure 1.1: MPLAB IDE Desktop**

MPLAB IDE dialog boxes behave as normal Windows applications and allow you to access standard Windows functions (as well as MPLAB IDE-specific functions) through the Microsoft Windows system button in the upper left hand corner. Other standard Windows features include window size buttons, icon buttons, vertical and horizontal scroll bars, and elevator buttons.

All MPLAB IDE functions are accessible through the menu bar located across the top of the desktop. MPLAB IDE menus that pull down from the menu bar allow you to access the emulator functions. Underlined characters on the pull-down menus are keyboard shortcuts. To use a shortcut, press and hold down the <Alt> key and press the shortcut key. For example, press and hold down

# MPLAB IDE Toolbar and Menu Options

the <Alt> key and press <F> to display the file menu. With the File menu displayed and the <Alt> key still held down, press <O> to display the Open File dialog.

## 1.3.1 Toolbars

For your convenience, MPLAB IDE contains four toolbars to provide you with shortcuts for performing routine tasks. The four toolbars are:

- Edit Toolbar
- Debug Toolbar
- Project Toolbar
- User Defined Toolbar

Click the button at the far left of the toolbar to display the desired toolbar. Refer to “Appendix 2: MPLAB IDE Toolbar and Status Bar Definitions” for complete descriptions of all four toolbars.

The buttons on each toolbar can be reconfigured for your specific needs. Refer to Section 1.9.5.1.2 for information.



Figure 1.2: MPLAB IDE Debug Toolbar

## 1.3.2 Status Bar

The figure below shows the status bar (Figure 1.3) that appears at the bottom of the MPLAB IDE desktop. Refer to “Appendix 2: MPLAB IDE Toolbar and Status Bar Definitions” for a description of the symbols on the status bar.

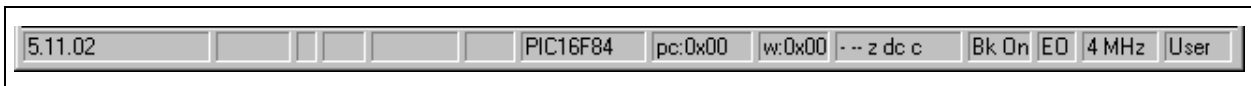


Figure 1.3: Status Bar

# MPLAB® IDE User's Guide

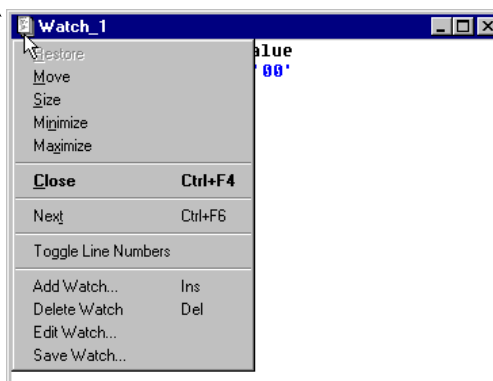
---

## 1.3.3 System Menu

MPLAB IDE provides windows for viewing various information.

**Note:** Use the system window control to change how data is displayed in the window.

system window control



## 1.4 File Menu

Options under the File menu are:

- New
- Open...
- View...
- Save
- Save As...
- Save All
- Close
- Close All
- Import
- Export
- Print
- Print Setup
- Exit
- (Most-Recently-Used File List)



# MPLAB IDE Toolbar and Menu Options

## 1.4.1 New File

The *File > New* command opens a new, empty window in which you can type. The window does not initially have a file name. To save the new file's contents, select *File > Save As*.

When the MPLAB Editor creates the window, it applies the set of modes that are defined for new files.

The tab size and other settings are set for new files in the Editor Modes tab after selecting *Options > Environment Setup*.

## 1.4.2 Open Existing File

To edit one or more existing files, select *File > Open*. It opens a standard dialog from which you select the files to edit. If a selected file is already open, MPLAB Editor activates the window that is currently showing the file.

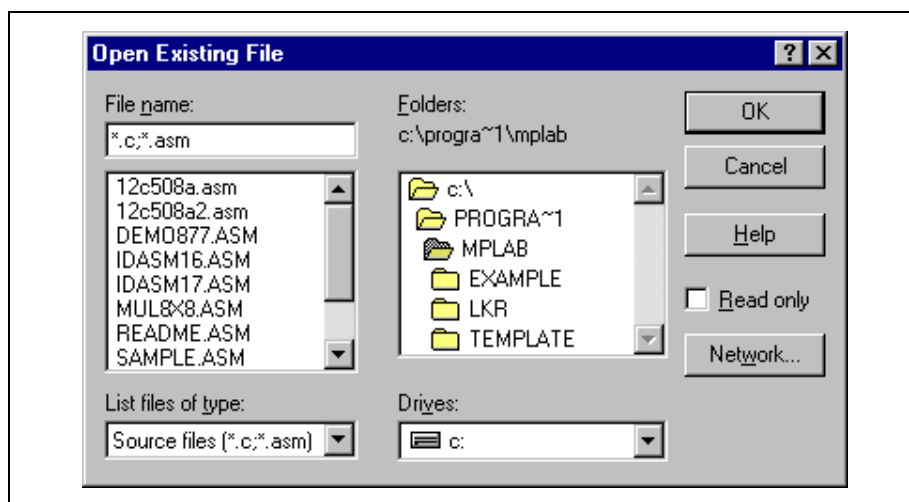


Figure 1.4: Open Existing File

1. Use the Drives and Folders list boxes to select the disk drive and the folder.
2. Select the files you want to open in the File Name list box.

To add multiple single files from the list, hold down the <Ctrl> key and click on the desired files.

To add a list of files, either hold the <Shift> key and click on the first and last file in the desired range, or click on the first file and drag down to the last file.

You can also type the name of the file(s) to open in the File Name field.

3. If you want to open the files in read only mode, check the Read Only box. This affects all the files you open in this operation.
4. Click **OK** to open the files.

# MPLAB<sup>®</sup> IDE User's Guide

---

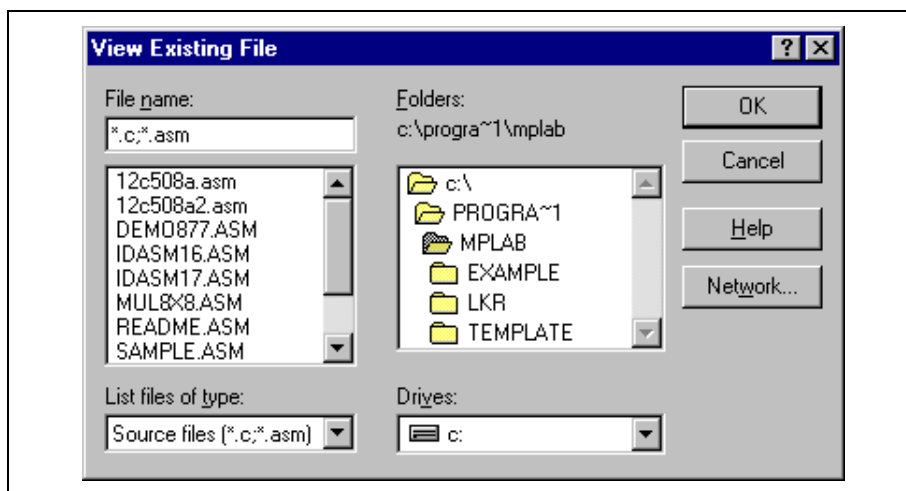
The “List Files of Type” list at lower left allows you to restrict the files shown in the list to those matching specific filename patterns. For example, “\*.ASM” will list all files with the suffix “.ASM.”

Whether you close the dialog with the OK or the Cancel buttons, MPLAB Editor makes its current working folder the one in the dialog.

If a file is selected that is already being edited, MPLAB Editor activates the window showing the file.

## 1.4.3 View File

The *File > View* command opens one or more existing files in read only mode. You can examine their contents, but not alter them. The *File > View* action is exactly as if you used the *File > Open* command and checked the Read Only box.



# MPLAB IDE Toolbar and Menu Options

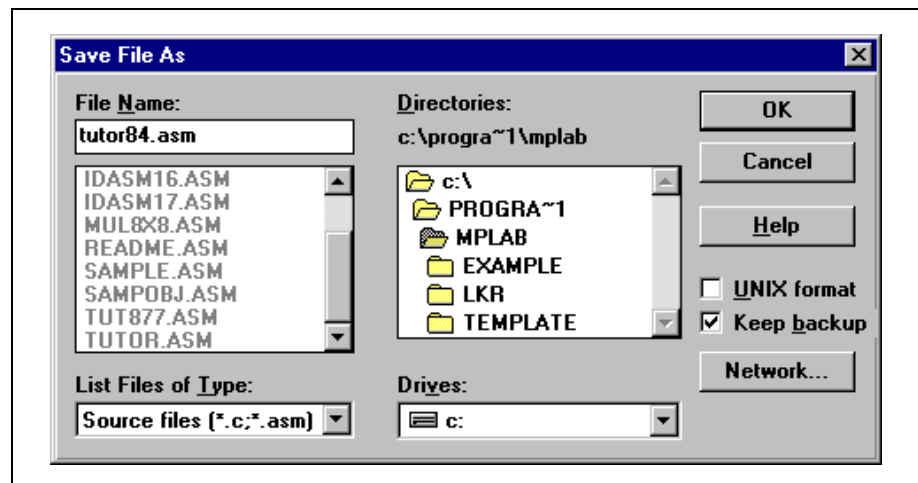
## 1.4.4 Save File

To save files to disk you have three options:

Select *File > Save* to save the contents of the current window to disk. The MPLAB Editor replaces the file with the contents of the current window. If the current file is unnamed, MPLAB Editor prompts for a new filename. If a file of the same name exists, MPLAB Editor makes that the backup copy and saves the current file.

Select *File > Save As* to save the contents of a file to disk, allowing you to specify the file name. The MPLAB Editor confirms overwrites of existing files.

1. Use the Drives and Directories list boxes to select the disk drive and the folder where you want to save the file.
2. Either type the name of the file into in the File Name edit control, or select the name of an existing file you want to overwrite from the list box.
3. Click **OK** to save the data to the file. If you specify the name of a file that already exists, MPLAB Editor confirms the overwrite.
4. Use the List Files of Type list at the lower left to restrict the files shown in the list to those matching specific filename patterns.



Select *File > Save All* to save all altered files, stores all altered templates into template files, and saves all altered template files in a single operation.

**Note:** Clicking **Cancel** in any of the dialogs that occur in this process cancels the entire Save operation.

# MPLAB® IDE User's Guide

---

## 1.4.5 Close File

To close the file being shown in the current window, select *File > Close*.

Select *File > Close All* to close all the open files that you're working on.

If you have changed any of the files and haven't saved the changes to disk, MPLAB Editor prompts you to save the changes, discard them, or cancel the entire operation.

## 1.4.6 Import

The *File > Import* functions allow you to move data from a PC file to the emulator and into target memory or into the simulator memory. This function also allows you to transfer data from the target into emulator memory for debugging.

### 1.4.6.1 Import to Memory

Select *File > Import > Import to Memory* to display the dialog box for selecting a file to import (download). The file you select is imported to the emulator memory or simulator memory. The file must be a valid HEX file. If you have a HEX file (ex; `code.hex`) ready for programming the PICmicro MCU device, use this option to load your HEX code into the MPLAB IDE Program Memory window. The file can be created by building an MPLAB IDE project or by exporting program memory.

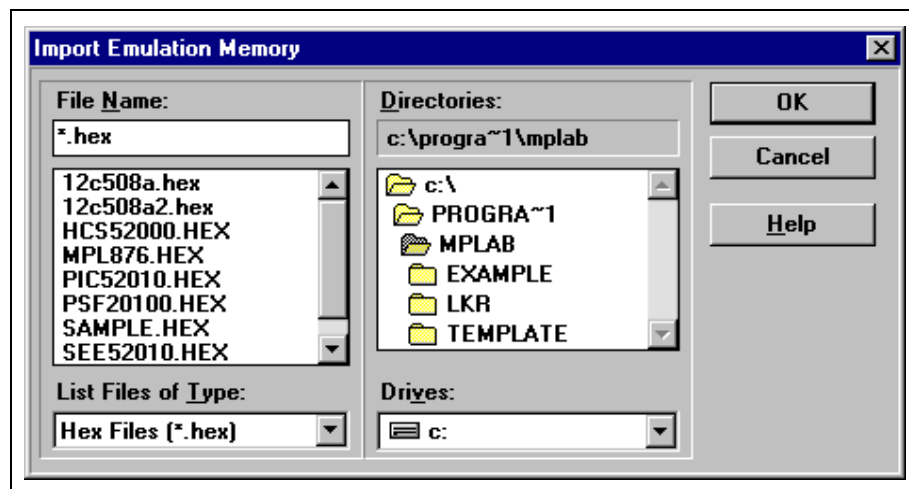


Figure 1.5: Import Emulation Memory Dialog

# MPLAB IDE Toolbar and Menu Options

---

## 1.4.6.1.1. PIC17CXXX and PIC18CXXX Devices

If the current target processor belongs to the PIC17CXXX or PIC18CXXX device family, then the file should be in the Intel extended hex file format (INH32). If the object file is successfully imported, then the symbols are loaded automatically from the file \*.COD. The default extension for the object code file is .HEX.

If you do not have a \*.COD file, you may want to turn off source tracking. Select *Options > Environment Setup*, click the **General** tab, and locate the source tracking setting in the Global Switches section of the dialog.

**Note:** Use INHX32 if your application addresses memory beyond 64 KB (32K words for the PIC17CXXX) or contains configuration bit information.

## 1.4.6.1.2. All Other PICmicro® MCU Devices

If the current target processor belongs to a PICmicro MCU family other than the PIC17CXXX or PIC18CXXX device family, then the file should be in the Intel 8-bit hex file format (INH8M) or the Intel extended hex file format (INH32). If the object file is successfully imported, then the symbols are loaded automatically from the corresponding .COD file if it exists. The default extension for the object code is .HEX.

## 1.4.6.2 Import to Target Memory

For PIC17CXXX or PIC18CXXX devices (that support extended memory) operating with an emulator in either microprocessor or extended microcontroller mode, you may select off-chip memory from a target board. Select *Options > Development Mode* and click the **Configuration** tab to access the Processor Mode. Select *Options > Development Mode* and click the **Memory** tab to access Off-Chip Memory.

To import (download) a data file to target memory, select *File > Import > Import to Target Memory* to display the dialog box for selecting a file to import through the emulator to target memory. The file must be a valid HEX file in the Intel extended hex file format (INHX32). The file can be created by building an MPLAB IDE project or by exporting program memory.

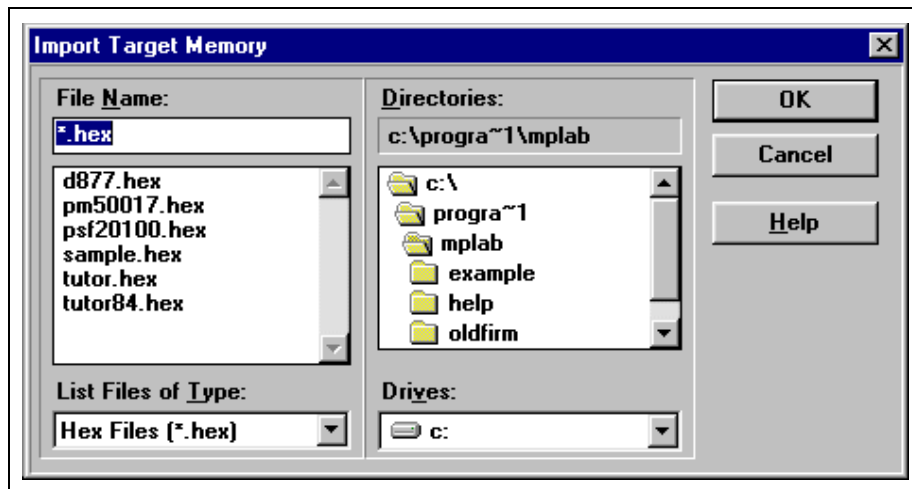


Figure 1.6: Import Target Memory Dialog (MPLAB ICE 2000)

Downloading data to off-chip memory on a target board does not automatically update that memory range in the Program Memory window. To perform this function, use the *Window > Modify > Read Range* feature. See Section 1.11.14.

# MPLAB IDE Toolbar and Menu Options

## 1.4.7 Export

### 1.4.7.1 Export Trace Buffer

Select *File > Export > Export Trace Buffer* to display a dialog box that will allow you to save the emulator or simulator trace buffer to the selected file.

**Note:** If you have a full trace buffer with maximum length labels, saving the trace for the MPLAB ICE 2000 emulator could require over 7 MB. Saving the complete trace buffer as it appears in the Trace Window (address, data, disassembled code and external logic probe lines) can exceed 1 MB for the MPLAB SIM simulator.

#### 1.4.7.1.1. MPLAB ICE 2000 Trace Buffer

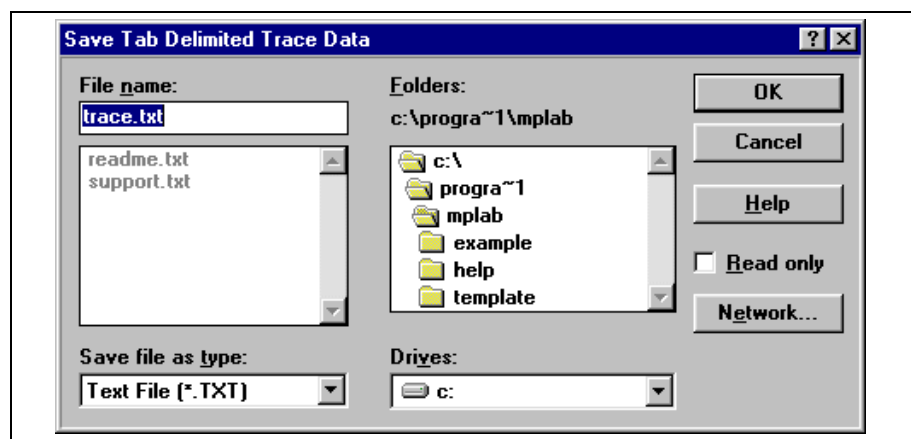


Figure 1.7: Export Trace Buffer Dialog (MPLAB ICE 2000)

*File name* Type the name of the file into in the File Name box or select the name of an existing file you want to overwrite from the list.

*Save file as type* Specify the type of file the trace buffer will be saved as. The list of files shown will be restricted to those of the type you specify.

*Folders* Select the folder where you want to save the file.

*Drives* Select the disk drive where you want to save the file.

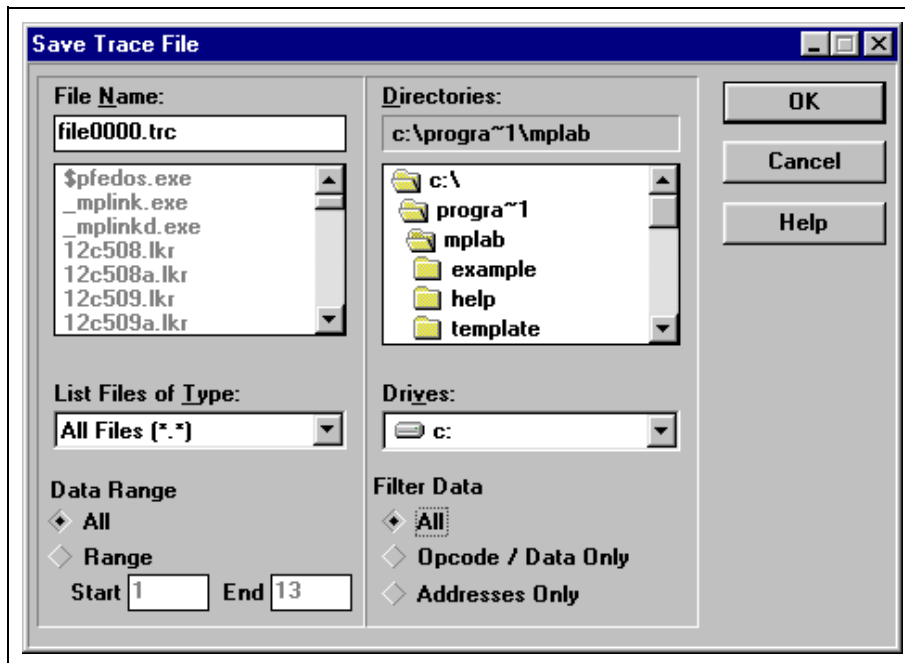
*OK* Click **OK** to save the data to the file. If you specify the name of a file that already exists, MPLAB Editor confirms the overwrite.

*Cancel* **Note:** Clicking Cancel cancels the entire Save operation.

*Read Only* Select **Read Only** if you wish to prevent the file from being overwritten later.

# MPLAB® IDE User's Guide

## 1.4.7.1.1. MPLAB SIM Trace Buffer



**Figure 1.8: Export Trace Buffer Dialog (MPLAB SIM)**

*Range* Select the range (0 to 8191) of the trace buffer that you want to save. Enter the desired value for the Start line number and for the End line number.

*Filter Data: All* Writes the complete trace buffer to the selected file.

*Filter Data: Opcode/ Data Only* Saves Opcode / Data only (for PIC17CXXX external read/write cycles). Available only with the MPLAB ICE 2000.

*Filter Data: Address Only* Saves Address only.



# MPLAB IDE Toolbar and Menu Options

## 1.4.7.2 Export Memory

Select *File > Export > Export Memory* to display the Export Memory dialog box and export memory contents to a file.

If you have a programmed PICmicro MCU device that you may wish to copy to other devices later, you can save the device's program memory by reading it into MPLAB IDE (using the Read Device option from the device programmer menu), then exporting the results of the Read to a HEX file using the *File > Export > Export Memory* option. Later, you can program another device by importing the file into MPLAB IDE and using the Program/Verify option from the device programmer menu.

**Note:** All locations (including the empty locations) in the range you select when you export will overwrite program memory when you import later. In contrast, the Build Project process ignores empty locations when creating a HEX file.

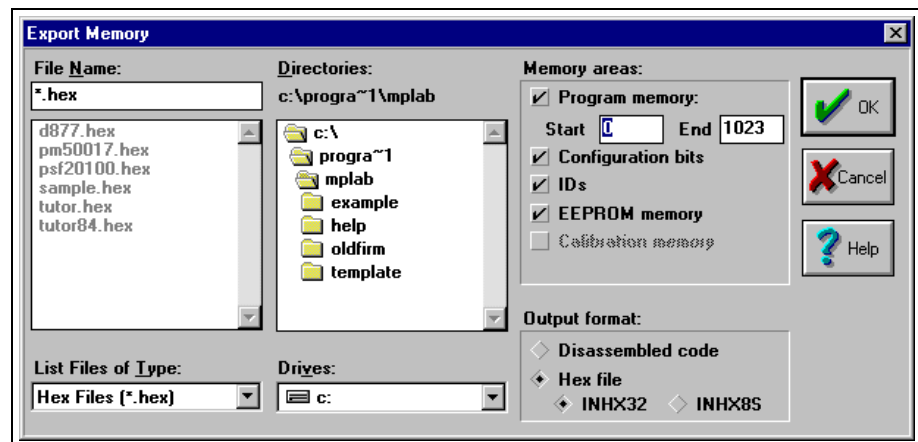


Figure 1.9: Export Memory Dialog

*Memory areas* Determine which memory areas you want to save in the build output file.

*Program Memory* Select to save program memory. Also state the Start and End address of program memory to save (entire range is the default).

*Configuration Bits* Select to save configuration bit settings.

*IDs* Select to save ID information, if applicable.

*EEPROM Memory* Select to save EEPROM memory, if applicable.

*Calibration Memory* Select to save the calibration memory, if applicable.

# MPLAB<sup>®</sup> IDE User's Guide

---

*Output format* Determine the format of the output file.

*Disassembled Code* Select to save in disassembled code format.

*HEX Code* Select either hex code format INHX32 or INHX85 (Intel split hex) format.

## 1.4.8 Print (Ctrl+P)

Select *File > Print* to print some or all of the current file on your currently selected printer.

The Print dialog allows you to specify the details of how the file is to be printed and on which printer.

By default, MPLAB IDE uses the same printer that you specified the last time you printed a file. The default details of the printing, such as line wrapping, page headers, tab size, and line numbering are taken from the window edit modes set on the current window. You can configure both these settings with the *Options > Current Editor Modes* menu command.

The dialog box shows the name of the printer that MPLAB IDE is currently using—if you haven't specified otherwise, this will be your system default printer.

### 1.4.8.1 Print Current File Options

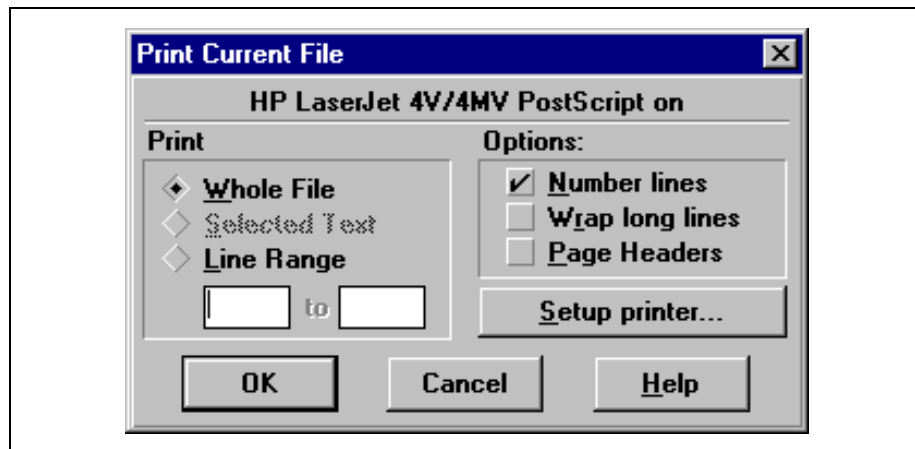


Figure 1.10: Print Current File Dialog

*Setup Printer* Click **Setup Printer** to change printers, printer font, or page margins.

*Whole File* Click **Whole File** (the default) to print entire file.

# MPLAB IDE Toolbar and Menu Options

- Selected Text* Click **Selected Text** to print only the text that is highlighted. This option is available only if you have text highlighted.
- Line Range* Click **Line Range** and fill in the start and end line numbers to print a range of lines. You can use the words “start” and “end” to represent the first and last lines of the file.
- Number Lines* Select **Number Lines** to print lines with numbering.
- Wrap Long Lines* Select **Wrap Long Lines** to fold lines too long to fit the page rather than being truncated.
- Page Headers* Select **Page Headers** to start each page with a header giving the file name and other information.

## 1.4.9 Print Setup

Select *File > Print Setup* to set up details of the printer that MPLAB IDE will use, run an individual printer’s setup dialog, and select the font. MPLAB IDE records the values you set with the Print Setup option for the selected printer. Thus, you can have different settings for different printers. These values become the defaults.

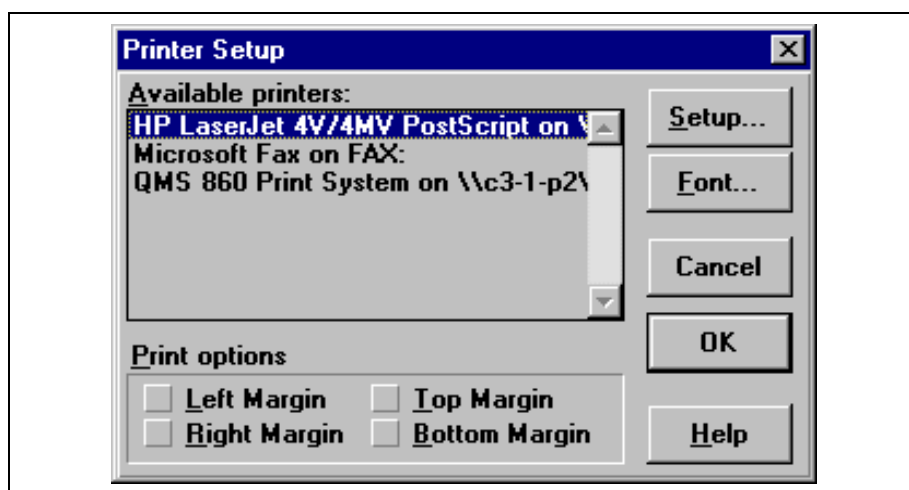


Figure 1.11: Printer Setup Dialog

*File > Print Setup* lets you specify:

- Which printer to use
- What page margins are to be applied when you print on this device
- What font to use with this printer

# MPLAB<sup>®</sup> IDE User's Guide

---

*File > Print Setup* also allows you to run the printer's own setup dialog to set device-specific information.

*Available Printers* To use a different printer than the one that is highlighted in the Available Printers list, scroll the list and click the left button on the printer name.

*Print Options* To change the margins used on each page, check or uncheck the boxes in the Print Options area.

*Setup* To run the highlighted printer's own setup dialog, click **Setup**.

*Font* To change the printer font that MPLAB IDE will use for the highlighted printer, click **Font**.

Because MPLAB IDE is a text editor rather than a word processor, you're restricted to choosing fixed pitchfonts, where the characters are all the same width.

If you run your printer's own setup dialog after selecting a font, you may find that the font is no longer available; some printers offer different fonts in different operating modes.

## 1.4.10 Exit (Alt+F4)

Select *File > Exit* to terminate your MPLAB IDE session.

If any of the files you're working with have changed and you haven't saved the changes to disk, MPLAB IDE will prompt you to save each one in turn. You can choose to save the changes, discard them, or cancel the exit operation. You will also be prompted to save the current project.

## 1.4.11 Most-Recently-Used File List

The MPLAB Editor adds the Most-Recently-Used (MRU) Files list to the end of the File Menu. Whenever you open a file, MPLAB Editor records the file name in the list, arranging the list so that the files that have been most recently used appear at the top. Any file in the list can be reopened simply by clicking on the menu item. This is a user configurable option. See Section 1.9.5.3.

# MPLAB IDE Toolbar and Menu Options

---

## 1.5 Project Menu

The following options are available on the Project menu:

- New Project: Creates a new project
- Open Project: Opens an existing MPLAB IDE project
- Close Project: Closes the currently open MPLAB IDE project
- Save Project: Saves the MPLAB IDE project
- Edit Project: Allows you to adjust many settings, add files, etc.
- Make Project: Builds all changed nodes in the project
- Build All: Builds all nodes in the project
- Build Node: Builds the selected node
- Install Language Tool: Allows you to configure MPLAB IDE to recognize a language tool
- (Most Recently Used Projects): Displays the most recently used projects

For detailed information on Project menu items, refer to "Chapter 3. Getting Started with MPLAB IDE – A Tutorial."

## 1.6 Edit Menu

### 1.6.1 General Edit Options

MPLAB Editor inserts text in either insert or strikeover mode. MPLAB Editor shows the mode as either “INS” or “OVR” on the status bar.

#### 1.6.1.1 Undo – Ctrl+Z

Select *Edit > Undo* to undo the last edit action. When there is no edit action to undo, the menu command shows Can't Undo, and you cannot select the command.

You can configure the number of recent edit actions that can be undone. See Section 1.9.5.4.

#### 1.6.1.2 Cut – Ctrl+X

Deletes the highlighted text in the current window, placing it on the clipboard. After this operation, you can paste the deleted text into another MPLAB Editor window or into another Windows application.

#### 1.6.1.3 Copy – Ctrl+C

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

#### 1.6.1.4 Paste – Ctrl+V

Pastes the contents of the clipboard into the current window at the position of the cursor. 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.

#### 1.6.1.5 Select All

Highlights the entire contents of the current window.

#### 1.6.1.6 Select Word – Left Mouse Button Double-Click

Double-click the left mouse button to highlight the word that the cursor is on.

#### 1.6.1.7 Delete Line – Ctrl+Shift+K

Deletes the entire line containing the cursor and moves the cursor to the start of the next line.

# MPLAB IDE Toolbar and Menu Options

## 1.6.1.8 Delete EOL – Ctrl+K

Deletes the text from the position of the cursor to the end of the line. If the cursor is at the start of a line that is completely empty, the next line is moved up to close the gap.

## 1.6.1.9 Goto Line – Ctrl+G

Moves the cursor to the start of a specific line.

This menu command allows you to specify either an absolute or a relative line number.

## 1.6.1.10 Find – F3

This command searches the current window for the text string using the search parameters you specify. If there is highlighted text in the current window, the Find operation uses it as the search string.

To search for text in a window, place the cursor at the start point of the search, and select *Edit > Find*.

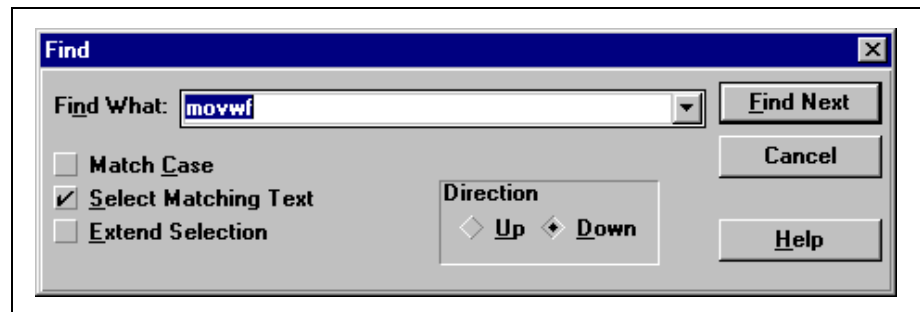


Figure 1.12: Find Dialog

In the Find dialog you can search for special characters by using the escape sequence for the character. You can use “\n” for carriage returns, “\t” for tabs, and “\” for backslashes.

## 1.6.1.11 Replace – F4

This command searches the current window for a text string, and optionally replaces occurrences with another string. This command lets you specify search parameters. If there is highlighted text in the current window, the Replace operation uses it as the search string.

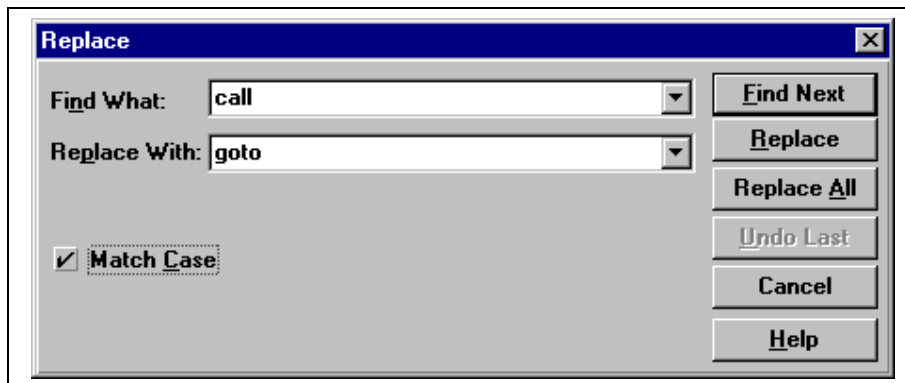


Figure 1.13: Replace Dialog

To replace text or special characters in a window, place the cursor at the start point, and select Edit > Replace. In the Replace dialog you can replace special characters by using the escape sequence for the character. You can use “\n” for carriage returns, “\t” for tabs, and “\” for backslashes.

## 1.6.1.12 Repeat Find – Shift+F3

To repeat a search without using the Find Dialog, select Edit > Repeat Find.

## 1.6.1.13 Repeat Replace – Shift+F4

To repeat the last replace operation without prompting for details, select Edit > Repeat Replace.



# MPLAB IDE Toolbar and Menu Options

## 1.6.1.14 Match Brace – Ctrl+B

MPLAB Editor allows the user to manipulate brace characters such as brackets and parentheses, which often delimit sections of text or program sources.

The definition of a brace character varies, depending on the language type set in a window's window modes. For C, braces are defined as the characters that have syntactic meaning – opening braces are { [ or (, and closing braces are } ] or ).

For the language type “none,” brace characters are defined as those commonly used in text or many other languages – opening braces are { [ ( or <, and closing braces are } ] ) or > .

To locate matching braces, place the cursor on one of the braces and use *Edit > Match Brace*. The cursor moves to the matching brace, respecting the level of nesting in the code.

## 1.6.2 Template Options

### 1.6.2.1 Attach Template File

The *Edit > Template > Attach File* command attaches an existing template (.TPL) file, loading it into memory so the templates within it can be used. Select the file you wish to attach, and click **OK**.

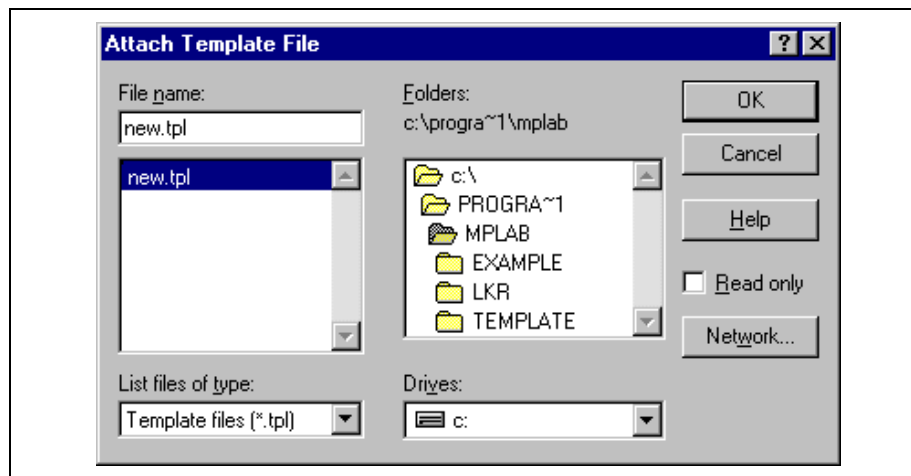


Figure 1.14: Attach Template File Dialog

## 1.6.2.2 Detach Template File

Use the *Edit > Template > Detach File* command to remove a template file from memory. Once a template file is detached, you cannot use the templates it contains.

Select *Edit > Template > Detach File*, select the template file you wish to detach, and click **OK**. You cannot detach the file if you're editing one of its individual templates.

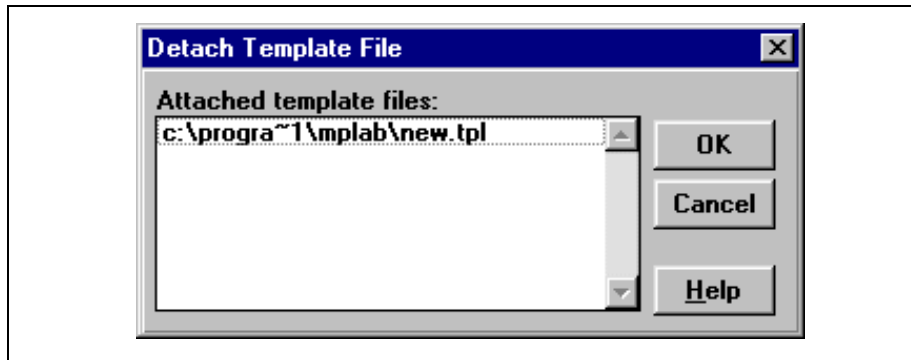


Figure 1.15: Detach Template File Dialog

# MPLAB IDE Toolbar and Menu Options

## 1.6.2.3 Create Template File

The first step in preparing MPLAB IDE templates for application developers' use is to create the file that will contain the templates.

Use the *Edit > Template > Create File* command to have MPLAB Editor create the template (.TPL) file that will contain the individual templates. A template (\*.TPL) file is a binary file that must be created before storing a template. Supply the filename, drive and folder. Click **OK**.

If you want this template file to automatically be available for all MPLAB IDE projects, name the file `auto.tpl`. The templates in `auto.tpl` are always available while you are editing source files in MPLAB IDE.

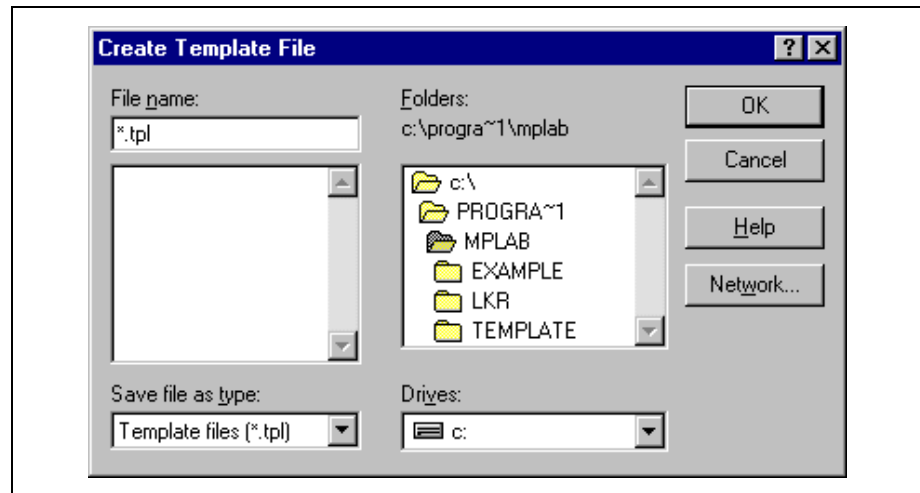


Figure 1.16: Create Template File Dialog

## 1.6.2.4 Save Template File

When you edit, create, or delete a template, MPLAB Editor changes a copy of the template in memory, which will be erased when you exit MPLAB IDE. These changes are not saved to disk until you use the *Edit > Template > Save File* command.

If you attempt to exit MPLAB IDE without saving the template file, you will be prompted if you wish to save changes to the template (.TPL) file. Answer **Yes**.

# MPLAB<sup>®</sup> IDE User's Guide

---

## 1.6.2.5 Insert Template

The *Edit > Template > Insert* command inserts an existing individual template into a source file so that it can be used in application development. Before you can insert the template, the file in which it resides must already be attached in MPLAB IDE (see Section 1.6.2.1).

Position the cursor at the location in your source file where you want to insert the template text. Select *Edit > Template > Insert*, select the template file that contains the template, select the individual template you wish to insert, and click **OK**.

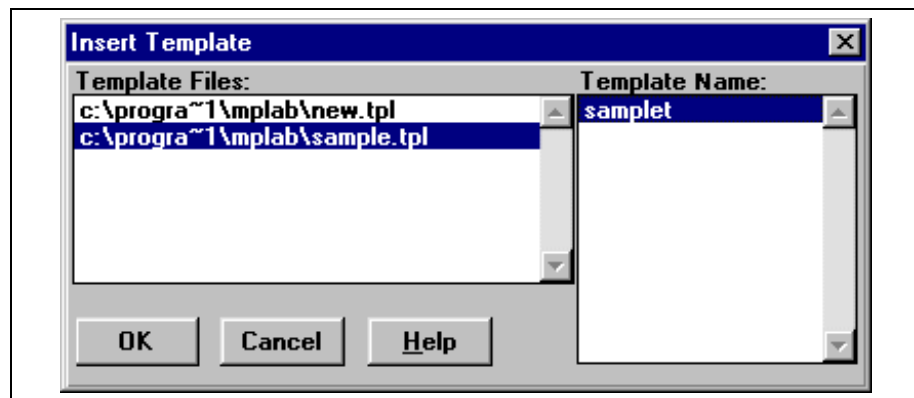


Figure 1.17: Insert Template Dialog

## 1.6.2.6 Edit Template

To change a template in a template file, use the *Template > Edit* command, select the template file and individual template you wish to modify, and click **OK**. The template file must already be attached in MPLAB IDE (see Section 1.6.2.1) in order for you to edit one of the files it contains.

When you select a template, MPLAB Editor places it in an edit window. The window is the same as for a normal text file, but the caption shows that it is actually a template. After editing the template use the *Edit > Template > Store* command to update the template.

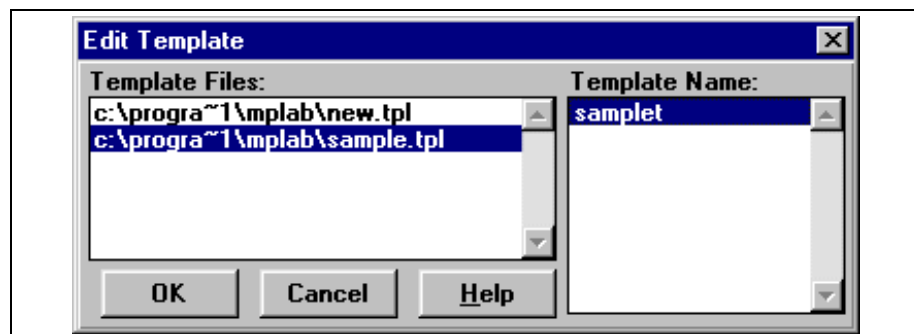


Figure 1.18: Edit Template Dialog

# MPLAB IDE Toolbar and Menu Options

---

## 1.6.2.7 New Template

Once you have created the .TPL file that will contain your templates, create the source and/or text content for the individual templates. You may wish to use the MPLAB IDE “canned” templates instead of creating your own from scratch.

Use the Edit > Template > New command to create a template. MPLAB Editor opens an edit window whose caption displays that it is a template.

To create your own source, simply type the text in the untitled template window.

To use one of the MPLAB IDE template files, open the MPLAB IDE template file in MPLAB\Templates\Code or MPLAB\Templates\Object. Select the text you wish to copy (or use Edit > Select All), and then copy and paste it into the untitled template window using the Edit > Copy and Edit > Paste commands.

You may wish to insert marks into your template to indicate any custom source code or text that developers will have to complete (see Section 1.6.2.11). When you are through creating your template, be sure to save it using the Edit > Template > Store or Edit > Template > Store Template As commands (see Sections Section 1.6.2.8 and Section 1.6.2.9).

## 1.6.2.8 Store

Select Edit > Template > Store to store a template in the template file. This command overwrites the previous version of the template in the template file.

**Note:** Because the Store command only writes to memory, the changes will be discarded when you exit MPLAB IDE. To save your template changes for future use, use Edit > Template > Save File. When you exit MPLAB IDE, you may be prompted if you wish to save changes to the template (.TPL) file. Answer **Yes**.

## 1.6.2.9 Store As

Once you are through creating the template and inserting marks, use the *Edit > Template > Store As* command to store the individual template source code/text in the template file. You can also use this command to save the changes to a new individual template file with a different name (for example, to create a variation of the original template source/text).

Select *Edit > Template > Store As*, select the template file, specify the template name, and click **OK**. If you select the name of an existing template, this command overwrites the previous template's contents. MPLAB Editor confirms before executing this command.

**Note:** Because the Store As command only writes to memory, the changes or additions will be destroyed when you exit MPLAB IDE. To save your template changes for future use, use *Edit > Template > Save File*. When you exit MPLAB IDE, you may be prompted if you wish to save changes to the template (.TPL) file. Answer **Yes**.

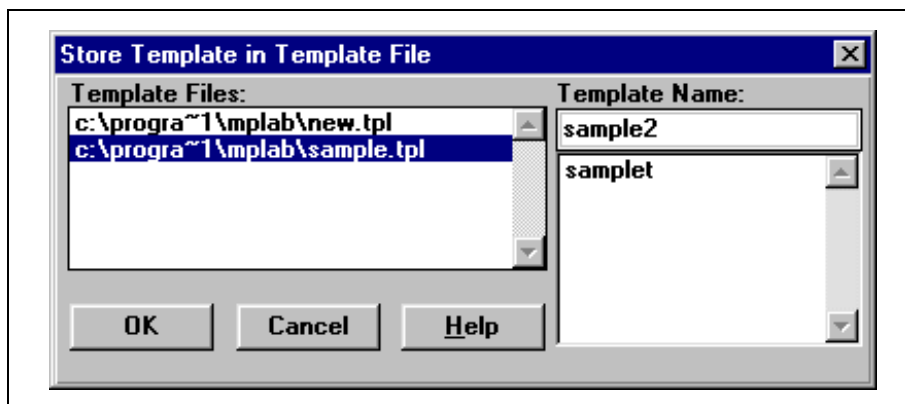


Figure 1.19: Store Template Dialog

# MPLAB IDE Toolbar and Menu Options

## 1.6.2.10 Delete Template

To delete a template from a template file, use the *Edit > Template > Delete* command. Select the template file containing the individual template you wish to delete, select the individual template, and click **OK**. The template must already be attached in MPLAB IDE (see Section 1.6.2.1) in order for you to select it for deletion.

**Note:** The Delete command only deletes the template from memory. To permanently remove the template from the template file, use *Edit > Template > Save File*. When you exit MPLAB IDE, you may be prompted if you wish to save changes to the template (.TPL) file. Answer **Yes**.

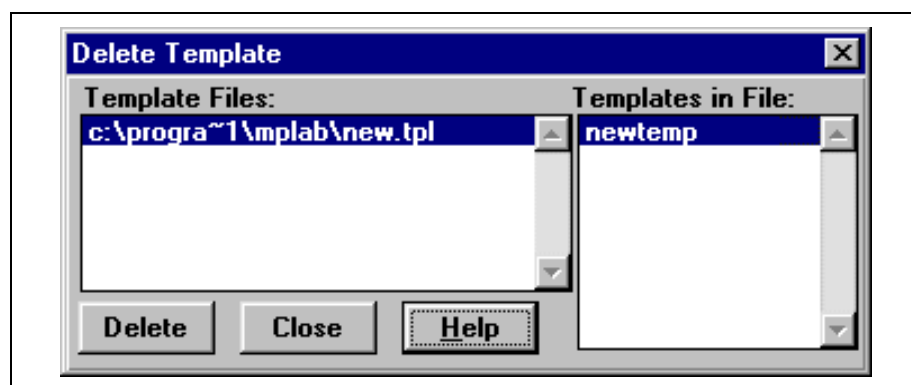


Figure 1.20: Delete Template Dialog

## 1.6.2.11 Insert Template Mark

While you are creating your template, you may wish to insert marks to indicate locations where developers will have to customize or insert special code for their application.

Position the cursor at each location in the template that developers will need to add their code and select *Edit > Template > Insert Mark* to insert marks (<???) at these locations. These marks will appear in their source file to facilitate the developers' work.

## 1.6.2.12 Find Template Mark

After you have inserted the template as described in Section 1.6.2.5, use the *Edit > Template > Find Mark* command to search for template marks. As described in the previous section, template marks identify locations where you will want to customize or insert special code for your application. Enter the desired text in place of each mark.

## 1.6.3 Text Options

This menu shows a list of text-related commands.

### 1.6.3.1 Transpose – Ctrl+T

Select *Edit > Text > Transpose* to transpose the character to the right of the cursor and the character to the left of the cursor. This command has no effect if the cursor is positioned at the start or end of a line.

### 1.6.3.2 Uppercase

Select *Edit > Text > Uppercase* to change all lowercase characters in the currently highlighted text to uppercase characters.

### 1.6.3.3 Lowercase

Select *Edit > Text > Lowercase* to change all uppercase characters in the currently highlighted text to lowercase characters.

### 1.6.3.4 Indent

To indent a single line, place the cursor anywhere within it and select *Edit > Text Indent*. The MPLAB Editor moves the entire text of the line to the right by one tab stop. The MPLAB Editor indents all highlighted lines. If no lines are highlighted, the MPLAB Editor indents only the line that the cursor is in.

### 1.6.3.5 Unindent

To un-indent a single line, place the cursor within it and select *Edit > Text UnIndent*. The MPLAB Editor moves the text left by one tab stop. The MPLAB Editor does not alter a line that does not start with either a tab or a space. Lines that do not start with white space are not affected.



# MPLAB IDE Toolbar and Menu Options

---

## 1.7 Debug Menu

The Debug menu contains all the options you would use when debugging your code.

### 1.7.1 Run

The Run menu options allow you to control the execution of your firmware in the target processor.

#### 1.7.1.1 Run (F9)

Debug > Run > Run takes the processor out of the halt state and puts the processor into execution until a break point is encountered or until you halt the processor.

Execution starts at the current program counter (as displayed in the status bar). The current program counter location is also highlighted in the Program Memory window. While the processor is running, the Step and Run buttons are disabled.

#### 1.7.1.2 Reset (F6)

Debug > Run > Reset issues a RESET sequence to the target processor. This issues a MCLR to reset the Program Counter to the RESET vector. If the processor is running it will continue running from the RESET vector address.

#### 1.7.1.3 Halt (F5)

Debug > Run > Halt forces the processor into the halt state. The Program Counter is stopped and the processor status information is updated.

#### 1.7.1.4 Halt Trace (Shift+F5)

Debug > Run > Halt Trace halts the trace buffer from capturing data but allows the processor to continue running. Refer to your emulator documentation for more information on Halt Trace.

#### 1.7.1.5 Animate

Debug > Run > Animate causes the simulator to actually execute single steps while in run mode, updating the values of the registers as it runs.

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

# MPLAB<sup>®</sup> IDE User's Guide

---

## 1.7.1.6 Step (F7)

*Debug > Run > Step* single steps the processor. This command executes one processor instruction (single or multiple cycle instructions) and then puts the processor back into halt state. After execution of one instruction, all the windows are updated with the current state of the processor. When the processor runs in real-time, MPLAB IDE ignores the Step button.

## 1.7.1.7 Step Over (F8)

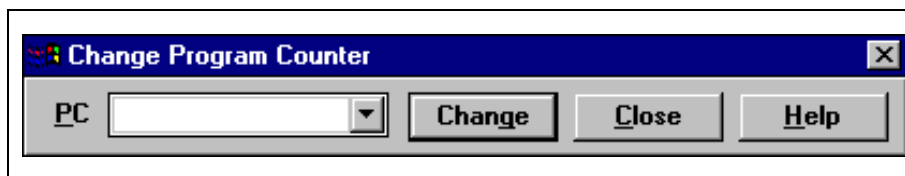
Select *Debug > Run > Step Over* to execute the instruction at the current program counter location. At a call instruction, Step Over executes the called subroutine, and halts at the address following the call.

## 1.7.1.8 Update All Registers

*Debug > Run > Update All Registers* updates all registers for the current instruction.

## 1.7.1.9 Change Program Counter

*Debug > Run > Change Program Counter* allows you to change the current program counter.



**Figure 1.21: Change Program Counter Dialog**

*PC* Enter desired Program Counter address.

*Change* Click **Change** to change to the new program counter address. The processor must be halted for the change to take effect.

*Close* Exits from the Change Program Counter dialog box.

# MPLAB IDE Toolbar and Menu Options

## 1.7.2 Execute

The Execute menu options allow you to control the polled execution of your firmware in the target processor.

### 1.7.2.1 Execute an Opcode

Select *Debug > Execute > Execute an Opcode* to execute a single instruction or a series of instructions without modifying the object code or program memory. After executing the instruction, you may resume executing from the current program memory location. Executing an opcode is not like single stepping through code; executing an opcode doesn't increment timers, cause cycle counts to occur, etc.

**Note:** Two-word instructions in the PIC18CXXX family cannot be executed in this manner.



Figure 1.22: Execute an Opcode Dialog

*Opcode* Enter the instruction as an opcode (in HEX digits) or enter a symbolic instruction (such as ADDWF 0x19). Click the Opcode list to display the last eight commands. After executing a command, MPLAB IDE highlights the command so you can type in a new instruction. (MPLAB IDE tracks the instructions you enter so you don't get two copies of the same instruction in the opcode list.)

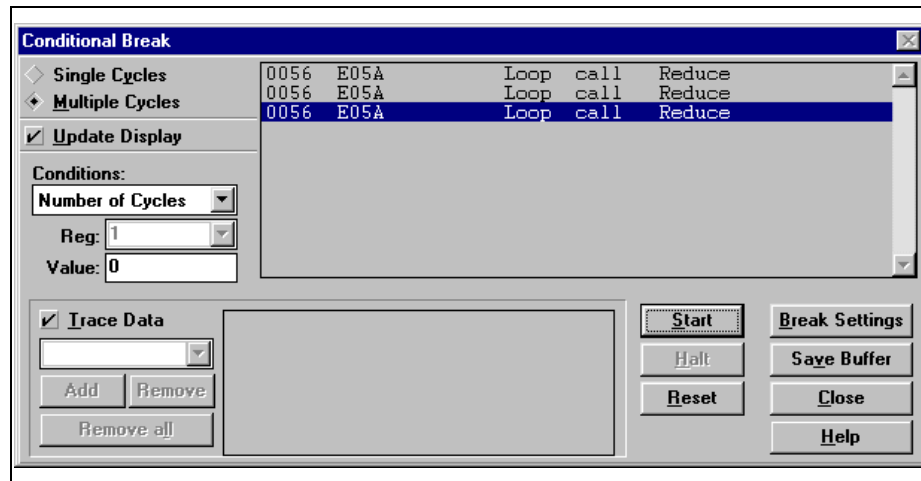
*Execute* Click **Execute** to execute an instruction without modifying the current location of the program counter.

### 1.7.2.2 Conditional Break

Select *Debug > Execute > Conditional Break* to display a dialog box that performs an automated single stepping of the processor. Execution starts when you click **Start** and continues until the condition presented in the dialog is met or until you click **Halt**.

**Note:** The Conditional Break dialog is not available in MPLAB ICE 2000 or MPLAB ICD development modes. For MPLAB ICE 2000, refer to the Complex Trigger dialog for this functionality.

For additional information on conditional breaks, see Section 2.8.



**Figure 1.23: Conditional Break Dialog**

- Single Cycle* Checks condition at every instruction. The Single Cycle option samples conditions after every instruction, allowing you to catch a particular condition.
- Multiple Cycles* Checks condition only at user-defined break points. The Multiple Cycles option runs at real time except for the halt at break points. This option allows interrupts to be serviced.
- Update Display* Executes the conditional break but does not update the disassembled code in the window. MPLAB IDE stores the last 1000 lines.
- Conditions* The condition that you set up is tested on any register location and an 8-bit constant value that you enter. You can test for the following conditions:  
User Halt When processor is running, click **Halt** to stop the processor.  
Number of Cycles Enter the number of cycles in the **Value** box.  
Register Value Conditions  
RAM Addr Data Value == Equals Value Entered  
RAM Addr Data Value < > Not Equals Value Entered  
RAM Addr Data Value > Greater Than Value Entered  
RAM Addr Data Value < Less Than Value Entered  
RAM Addr Data Value >= Greater or Equal Value Entered  
RAM Addr Data Value <= Less or Equal Value Entered

# MPLAB IDE Toolbar and Menu Options

If the tested condition is true, execution stops before executing the next instruction. The next instruction in the Program Memory window will be highlighted.

**Caution:** All register values are treated as 8-bit unsigned values. Therefore, the condition  $<0$  will never be true.

**Reg** Register Condition. Enter a RAM address location where you want to test against the data value at that location. The location that you enter must be a file register location.

**Value** Enter an 8-bit value in the Value box that you want to test against.

*Trace Data* Trace Data samples specific registers at each time the processor is halted and displays the register data in the list.

*Add, Remove, Remove All* Edits the list of data variables sampled at each break point.

*Start* Starts execution and continues to execute single steps until the condition is met or until you press the **Halt** button.

*Halt* Halts execution of the Conditional Break.

*Reset* Resets the processor.

*Break Settings* Opens the Break Point Settings dialog.

*Save Buffer* Opens the Save File dialog to save information from the list box in a \*.TB file.

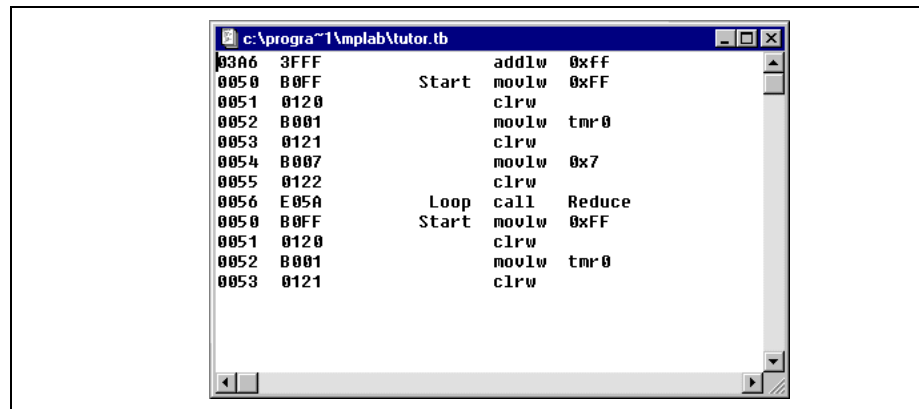


Figure 1.24: \*.TB File

*Close* Exits the Conditional Break dialog.

## 1.7.3 Simulator Stimulus

MPLAB SIM simulator functions allow you to set up regular clock stimulus signals and allow the simulator to respond to events from files on your PC. The files can be written with the MPLAB Editor or any other suitable text editor or word processor and should be saved in the same folder as the current project.

Simulator functions are:

- Asynchronous Stimulus
- Pin Stimulus
- Clock Stimulus
- Register Stimulus

For a detailed discussion of simulator stimulus, see Section 2.9.

## 1.7.4 Center Debug Location

Select *Debug > Center Debug Location* to move the current program counter to the middle of the debugging window.

This function works on the Source Code Window, the Program Memory Window, and the Absolute Listing Window.

## 1.7.5 Break Point Settings

Select *Debug > Break Settings* to set break points and break point qualifiers.

For more functional information on break points, see Section 2.7.

**Note:** If execution doesn't halt at the break point, select *Options > Development Mode* and click the **Break Options** tab. Make sure that Global Break Enable is selected (check marked).

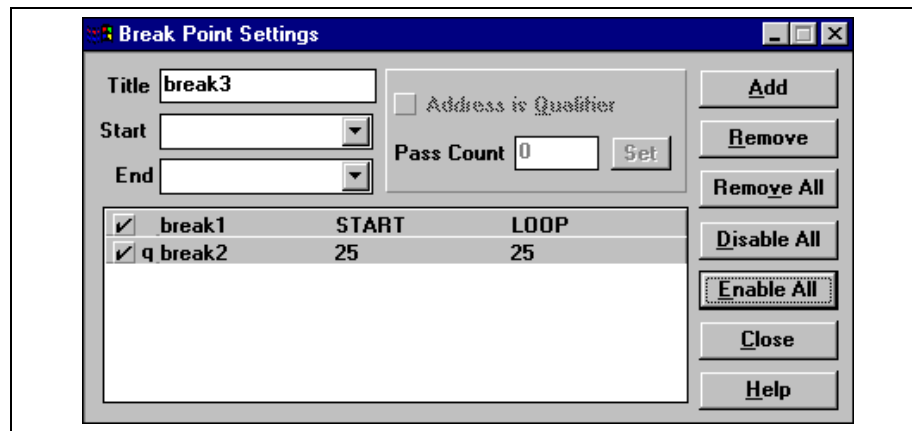


Figure 1.25: Break Point Settings Dialog

# MPLAB IDE Toolbar and Menu Options

---

## Operational Notes

- If execution doesn't halt at the break point, select *Options>Development Mode* and click the **Break Options** tab. Make sure that Global Break Enable is selected (checkmarked).
- When using MPLAB ICD, only one address can be set to a break point in the Break Point Settings dialog.
- If a software break point is set, then checksums calculated on program memory at run time will be incorrect. Programmer checksums are not affected.
- Manually resetting the timestamp when using software break points will cause the timestamp to be incorrect.

### 1.7.5.1 General Break Point Settings

You can define up to 16 named break point ranges. After entering a break point title, start address, and end address (optional), click **Add** to accept the break point range definition.

### 1.7.5.2 Saving Break Point Settings

Break points are saved as part of the project.

*Title* Enter a unique title (up to 32 characters) for each break point range. MPLAB IDE accepts the underscore character but does not allow spaces. MPLAB IDE automatically enters a default, unique title if you choose not to enter a title. The break point range requires a title.

*Start, End* Enter a Start and End address in HEX or as a label for the break point range. The address range is restricted to the valid address range of the target processor. You can enter the start address and MPLAB IDE will fill in the same end address for a break point on a single location (rather than a range).

You can enter the Start and End values as addresses or labels. If you use labels, MPLAB IDE allows you to modify the labels by using offsets: "MAIN+2" or "EXECTIMR-10." When you use labels and recompile a project (and the label moves due to the compilation) MPLAB IDE assigns the break points to the new address range.

You can use an existing break point range item as the starting point for entering a new break point range. Click on a desired item in the list box. Type in a new title. Then click **Add** to accept the defined break point range.

# MPLAB® IDE User's Guide

---

**Break Point List** The list box allows you to enter up to 16 break point ranges. On selecting a range, the break point Settings dialog box displays Start, End, and Title to allow you to edit the start and end address. The list box contains the following elements:

	Title	Start	End
<input checked="" type="checkbox"/>	break1	start	start
<input checked="" type="checkbox"/>	break3	ReduceADoubleB	
<input checked="" type="checkbox"/>	q end break	65	65
<input checked="" type="checkbox"/>	middle break	start+2	loop-2

**Figure 1.26: Break Point List**

**Enable/Disable** This switch enables/disables the break point range in program memory.

**Qualifier** Qualifier point designator. When a range item displays the letter q, the address range is a Pass Count address.

**Title, Start, End** Click an item in the list box to display the title, start, and end data. You can then edit the start and end address or change the title to enter a new break point range.

**Add** Click **Add** to accept the selected range and add the range to the list of ranges. Entering a range enables break points in that range in program memory. Disabled break point ranges will not clear break points included in other ranges.

**Remove** Removes the selected break point range. If an item is not selected, Remove does nothing.

**Remove All** Removes all break point ranges from the list.

**Disable All** Disables all break point ranges.

**Enable All** Enables all break point ranges.

**Close** Accepts your break point ranges and closes the Break Point Settings dialog.

**Help** Displays this information.



# MPLAB IDE Toolbar and Menu Options

---

## 1.7.5.3 Break Point Qualifiers

This function applies only to the PICMASTER Emulator and Simulator development modes.

1. Select a break point qualifier from the drop-down list.
2. Enter a pass count qualifier for the break point.

MPLAB IDE has a 16-bit Pass Counter that decrements by one on any address match in program memory.

When the processor is in a Halt state, you can modify the count value in the Break Point Settings dialog box. To set up the Pass Counter, first set the desired address ranges and then load the counter with a desired count value (up to 16 bits). When the counter decrements to zero, the emulator will halt.

**Note:** The pass counter does not automatically reset. You must select **Set**.

The Pass Counter decrements each time an event occurs. You can use this feature to count the number of times an event happens.

*Address is Qualifier* You can assign a Pass Count qualifier address to either break logic or trace logic. You must have an element in the list selected. With an element selected, the dialog will enable the Address is Qualifier check box.

When you check Address is Qualifier, you are assigning the pass counter to all addresses in the selected range. Usually it is useful to set only a single address for the pass counter. After selecting Address is Qualifier, MPLAB IDE enables the Pass Count Edit Box and the Set button. With the Pass Count Edit Box enabled, you can set the pass counter to a desired value (up to 65,534).

*Pass Count* Type in a pass count value. The pass count value defines the number of times the program can pass a qualifier address before halting the processor. Each time the program encounters an address that has been set as a qualifier, it decrements the pass count. When the pass count reaches 0, it halts the processor.

*Set* Click **Set** to enter a pass count value. The displayed Pass Count value does not download the value to the emulator's pass counter until you click **Set**.

## 1.7.6 Trace Settings

Select *Debug > Trace Settings* to display the Trace Point Settings dialog box for defining up to 16 named trace point ranges. The Trace Point Settings dialog is not available in MPLAB ICE 2000 or MPLAB ICD development modes.

For more functional information on trace points, see Section 2.7.

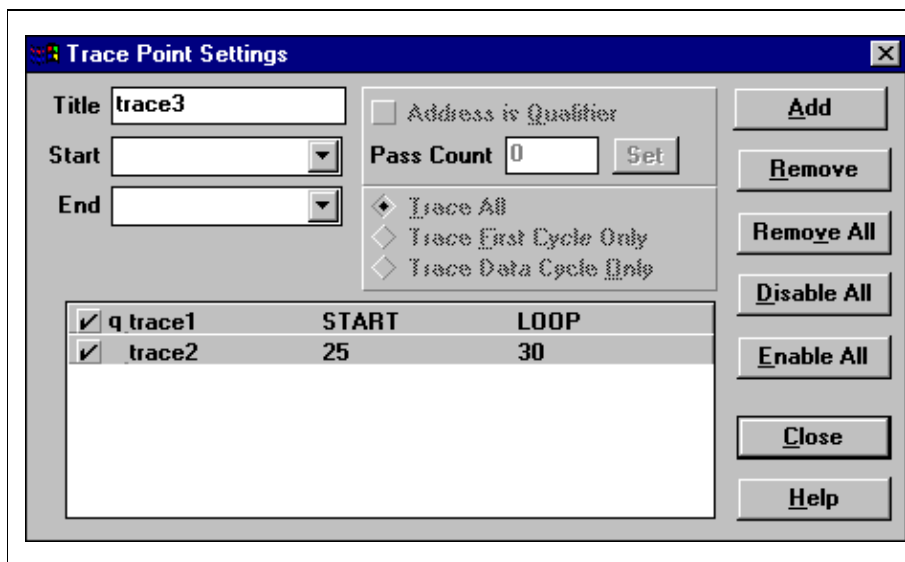


Figure 1.27: Trace Point Settings Dialog

- Note 1:** The pass counter qualifier address can be assigned to trace logic or break logic.
- 2:** The Trace Point Settings dialog is not available in MPLAB ICE 2000 or MPLAB ICD development modes. The MPLAB ICE 2000 trace may be configured through the Complex Trigger dialog.

### 1.7.6.1 General Trace Settings

You can define up to 16 named trace point ranges. After entering a trace point title, start address, and end address (optional), click **Add** to accept the trace point range definition.

**Note:** MPLAB ICD does not have trace capability.

# MPLAB IDE Toolbar and Menu Options

---

## 1.7.6.2 Saving Trace Point Settings

Trace points are saved as part of the project.

*Title* Enter a unique title (up to 32 characters) for each trace point range. MPLAB IDE accepts the underscore character but does not allow spaces. MPLAB IDE automatically enters a default, unique title if you choose not to enter a title. The trace point range requires a title.

*Start, End* Enter a Start and End address in HEX or as a label for the trace point range. The address range is restricted to the valid address range of the target processor. You can enter the start address and MPLAB IDE will fill in the same end address for a trace point on a single location (rather than a range).

You can enter the Start and End values as addresses or labels. If you use labels, MPLAB IDE allows you to modify the labels by using offsets: "MAIN+2" or "EXECTIMR-10." When you use labels and recompile a project (and the label moves due to the compilation) MPLAB IDE assigns the trace points to the new address range.

You can use an existing trace point range item as the starting point for entering a new trace point range. Click on a desired item in the list box. Type in a new title. Then click **Add** to accept the defined trace point range.

# MPLAB® IDE User's Guide

---

*Trace Point List* The list box allows you to enter up to 16 trace point ranges. On selecting a range, the Trace Settings dialog box displays Start, End, and Title to allow you to edit the start and end address. The list box contains the following elements:

	Title	Start	End
Qualifier	q trace1	START	LOOP
Enable/Disable	trace2	25	30

**Figure 1.28: Trace Point List**

Enable/Disable This switch enables/disables the trace point range in program memory.

Qualifier Qualifier point designator. When a range item displays the letter q, the address range is a Pass Count address.

Title, Start, End Click an item in the list box to display the title, start, and end data. You can then edit the start and end address or change the title to enter a new trace point range.

*Add* Click **Add** to accept the selected range and add the range to the list of ranges. Entering a range enables trace points in that range in program memory. Disabled trace point ranges will not clear trace points included in other ranges.

*Remove* Removes the selected trace point range. If an item is not selected, Remove does nothing.

*Remove All* Removes all trace point ranges from the list.

*Disable All* Disables all trace point ranges.

*Enable All* Enables all trace point ranges.

*Close* Accepts your trace point ranges and closes the Trace Point Settings dialog.

*Help* Displays this information.

# MPLAB IDE Toolbar and Menu Options

---

## 1.7.6.3 Trace Point Qualifiers

This function applies only to the PICMASTER Emulator and Simulator development modes.

1. Select a trace point qualifier from the drop-down list.
2. Enter a pass count qualifier for the trace point.

MPLAB IDE has a 16-bit Pass Counter that decrements by one on any address match in program memory.

When the processor is in a Halt state, you can modify the count value in the Trace Point Settings dialog box. To set up the Pass Counter, first set the desired address ranges and then load the counter with a desired count value (up to 16 bits). When the counter decrements to zero, the trace will start.

**Note:** The pass counter does not automatically reset. You must select **Set**.

The Pass Counter decrements each time an event occurs. You can use this feature to count the number of times an event happens.

*Address is Qualifier* You can assign a Pass Count qualifier address to either break logic or trace logic. You must have an element in the list selected. With an element selected, the dialog will enable the Address is Qualifier check box.

When you check Address is Qualifier, you are assigning the pass counter to all addresses in the selected range. Usually it is useful to set only a single address for the pass counter. After selecting Address is Qualifier, MPLAB IDE enables the Pass Count Edit Box and the Set button. With the Pass Count Edit Box enabled, you can set the pass counter to a desired value (up to 65,534).

*Pass Count* Type in a pass count value. The pass count value defines the number of times the program can pass a qualifier address before halting the processor. Each time the program encounters an address that has been set as a qualifier, it decrements the pass count. When the pass count reaches 0, it halts the processor.

*Set* Click **Set** to enter a pass count value. The displayed Pass Count value does not download the value to the emulator's pass counter until you click **Set**.

## 1.7.6.4 Global Trace Point Environment Options

Trace All applies to the PICMASTER emulator and the simulator (MPLAB SIM). Trace First Cycle Only and Trace Data Cycle Only apply to the PICMaster emulator only.

*Trace All* Valid for all PICmicro MCU devices. Lets you trace each enabled trace address location. This option displays the trace buffer exactly the way the processor fetches and executes instructions at each clock cycle.

*Trace First Cycle Only* Valid for all PICmicro MCU devices. Filters cycles from the trace buffer of all extra forced NOP cycles (or data cycles) of two-cycle instructions. This allows the trace buffer to display only the exact instruction flow sequence. Eliminating extra cycles also conserves valuable trace buffer space and provides more room to capture meaningful data.

*Trace Data Cycle Only* Valid for PIC17CXXX devices only. Captures only the data cycle (second cycle) of the two-cycle table write (TABLWT) and table read (TABLRD) instructions. Use this feature to read an array of tables, or to capture data memory values from a desired RAM location and table write the values to an unused program memory location in real-time for debugging purposes.

To save captured data cycles into a file for plotting or analysis, use *File > Export > Export Trace Buffer*.

A simulator trace memory buffer looks like Figure 1.29. For information on emulator display, see the documentation for that emulator.

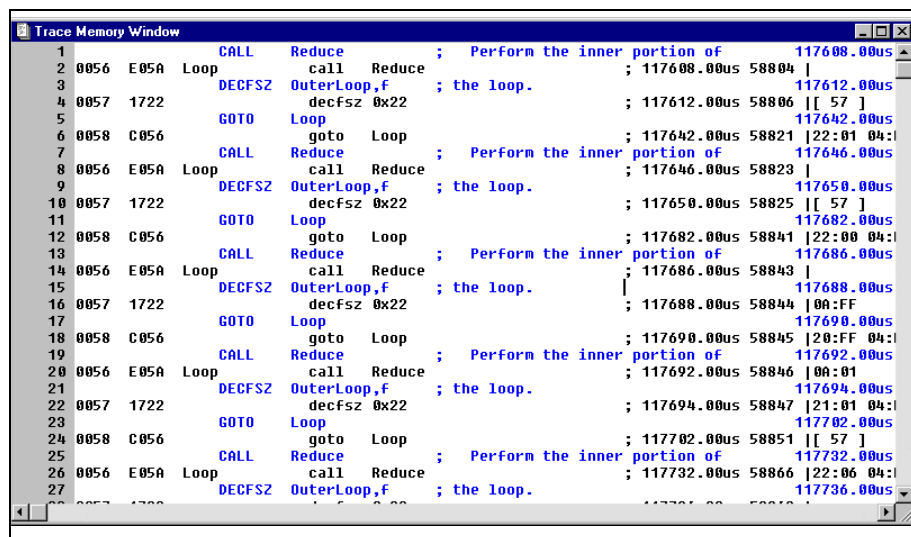


Figure 1.29: Trace Memory Window

# MPLAB IDE Toolbar and Menu Options

## 1.7.7 Trigger In/Out Settings

Select *Debug > Trigger In/Out Settings* to set trigger input and output. This dialog applies to the MPLAB ICE 2000 and PICMASTER emulators.

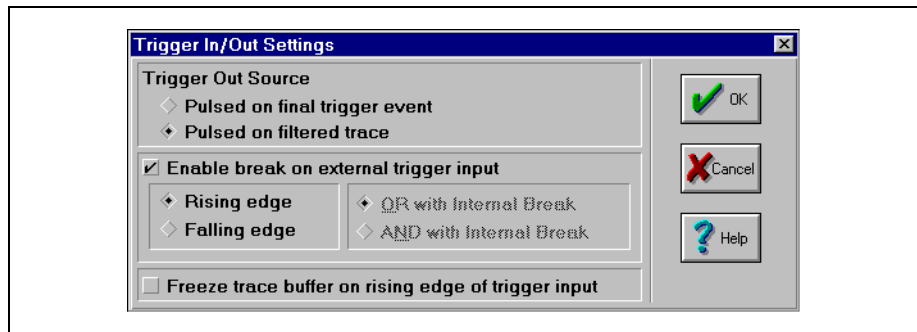


Figure 1.30: Trigger In/Out Settings Dialog – MPLAB ICE 2000

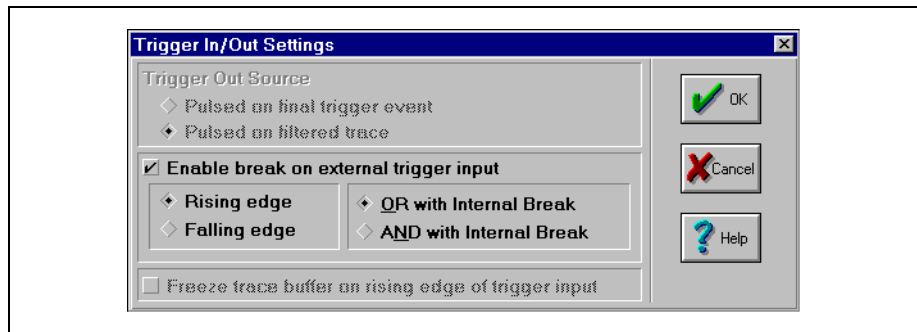


Figure 1.31: Trigger In/Out Settings Dialog – PICMASTER Emulator

For more information on setting these emulator options, see either the *MPLAB ICE In-Circuit Emulator User's Guide* (DS51159) or the *PICMASTER User's Guide* (DS51037).

## 1.7.8 Trigger Output Points

Select *Debug > Trigger Output Points* to set trigger output for the PICMASTER emulator.

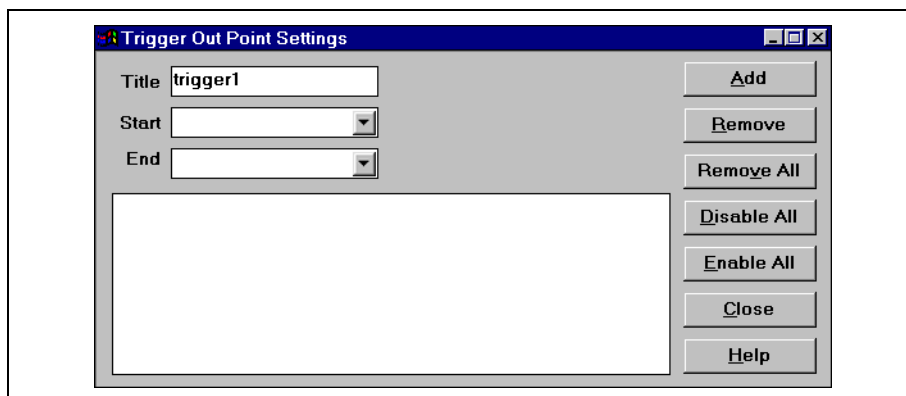


Figure 1.32: Trigger Out Point Settings Dialog

For more information on setting these emulator options, see the *PICMASTER User's Guide* (DS51037).

## 1.7.9 Clear All Points

Select *Debug > Clear All Points* to clear all break and trace points.

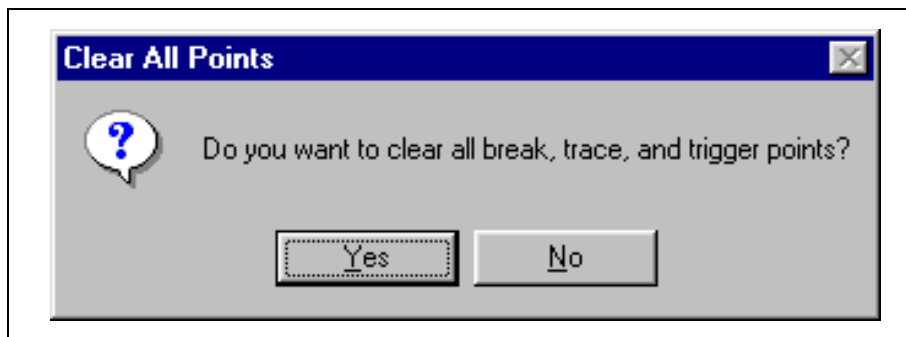


Figure 1.33: Clear All Points Message Box



# MPLAB IDE Toolbar and Menu Options

## 1.7.10 Complex Trigger Settings

Select *Debug > Complex Trigger Settings* to set a complex trigger for the MPLAB ICE 2000 emulator.

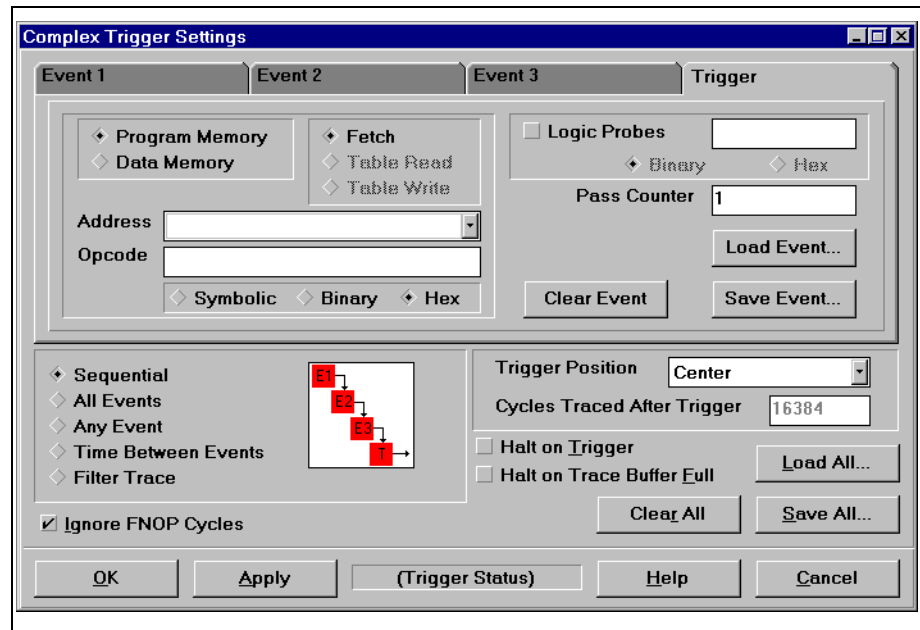


Figure 1.34: Complex Trigger Settings Dialog

For more information on setting these emulator options, see the *MPLAB ICE In-Circuit Emulator User's Guide (DS51159)*.

## 1.7.11 Code Coverage

Select *Debug > Code Coverage* to enable/disable code coverage for the MPLAB ICE 2000 emulator.

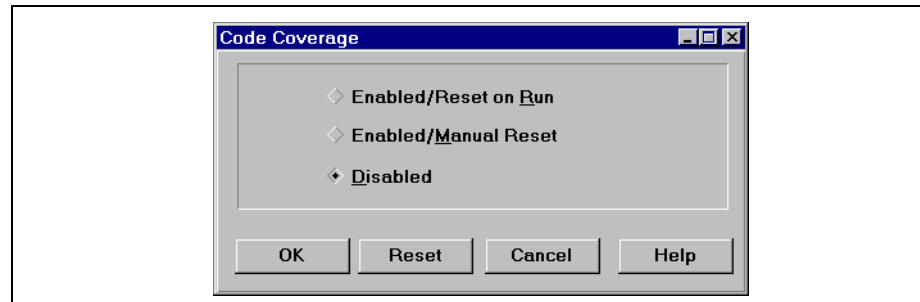


Figure 1.35: Code Coverage Dialog

For more information on setting these emulator options, see the *MPLAB ICE In-Circuit Emulator User's Guide (DS51159)*.

## 1.7.12 Clear Program Memory (Ctrl+Shift+F2)

Select *Debug > Clear Program Memory* to set all program memory bits to one.



Figure 1.36: Clear Program Memory Message Box

## 1.7.13 System Reset (Ctrl+Shift+F3)

Select *Debug > System Reset* to reset the entire emulator system including the MPLAB ICE 2000 emulator hardware (if connected), software and the target processor. System Reset performs the same initialization that is performed when MPLAB IDE is first entered.

- Note 1:** To perform a processor Reset ( $\overline{MCLR}$ ), select *Debug > Run > Reset*.
- 2:** Always power down the emulator pod when changing probes or processor modules, and then perform a system Reset. If you do not perform a system Reset, MPLAB IDE will not be properly configured for the new probe or processor module.

# MPLAB IDE Toolbar and Menu Options

## 1.7.14 Power-On-Reset (Ctrl+Shift+F5)

Select *Debug > Power-On-Reset* to display the Power-On-Reset dialog box for selecting a POR option.

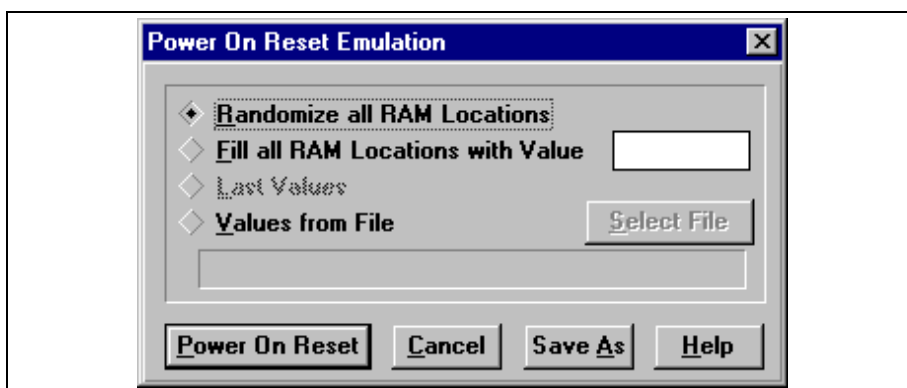


Figure 1.37: Power-On-Reset Dialog

Power-On-Reset allows you to fill RAM locations with random or determined values.

Often, uninitialized registers can lead to a program malfunction that is hard to track down. POR will simulate the action of randomizing registers when the application first starts up. If the application misbehaves sometimes, the Power-On-Reset function may help you isolate the problem.

**Note:** Power-On-Reset is implemented on a device by tying the  $\overline{\text{MCLR}}$  pin to VDD. The POR does not necessarily reset all SFRs, especially for the emulator.

The Power-On-Reset dialog can be used for the following functions:

- Randomize registers that have an unknown value at POR.
- Fill registers with a value, or clear registers.
- Set registers to the POR Condition shown in the Microchip Databook for the respective part (MPLAB SIM simulator only). The Fill with Value POR function will not affect the registers that have specifically defined reset values.
- Save current POR values to a file.
- Load POR values from a file.

**Note:** Program memory and break points are undisturbed when using the POR dialog.

# MPLAB<sup>®</sup> IDE User's Guide

---

- Randomize* Select **Randomize** to enter random values into registers that have an unknown value at POR.
- Fill with Value* Type a fill value in the Enter Value box that you want to enter into registers at POR.
- Last Values* Select **Last Values** to enter the last randomized or filled values into the device at POR.
- Power-On-Reset* Click **Power-On-Reset** to reset and set selected register values.
- Cancel* Click **Cancel** to close the Power-On-Reset dialog box without performing a POR.
- Save As* Opens a dialog box to enter the name of a file (\*.POR) to save Power-On-Reset settings.
- Values from File* To load data values from a file, click **Values from File** and then click **Select** to open a dialog box to enter the name of an \*.POR file containing values to load at power-on-reset.

# MPLAB IDE Toolbar and Menu Options

---

## 1.8 Programmer Menu

To select a programmer, open the Select Programmer dialog from the *Options > Programmer Options > Select Programmer* menu item. Once you change programmers, MPLAB IDE will shut down. The programmer change will take effect when you restart MPLAB IDE. A menu specific to the programmer you selected will appear on the menu bar.

For more information on the operation of individual programmers, refer to the documentation for that programmer.

### PICSTART Plus Programmer Menu

- |                                  |  |
|----------------------------------|--|
| <i>Enable/Disable Programmer</i> | Enable or disable the programmer. Once the programmer is enabled, this menu item changes to Disable Programmer.  |
| <i>Program/Verify</i>            | Opens the Program/Verify dialog which allows you to program the device selectively (e.g., program part of program memory, or only configuration bits) or to verify that the device was programmed properly.  |
| <i>Read Device</i>               | Opens the Read Device dialog which allows you to read the device selectively (e.g., part of program memory, or only configuration bits of the device).   |
| <i>Blank Check All</i>           | Checks that the device is completely blank (all bits are set to a "1"). This will also check that all configuration bits are set to a "1" (unprogrammed state).  |
| <i>Blank Check OTP</i>           | This function is intended for use with OTP devices that come with factory programmed configuration bits. Before using this function, set the displayed configuration bits to match the factory programmed settings. The function verifies that all program memory bits are set to "1" and that the configuration bits match the settings displayed in the Configuration Bits window. |
| <i>Display Error Log</i>         | Displays the error log on the screen if any errors occur. The error log is generated when programming or verification results in a mismatch of device data and data in the Program Memory window and the Programmer Status dialog.   |
| <i>Erase Program Memory</i>      | Sets all bits in the MPLAB IDE Program memory, Calibration memory, and Data memory (if applicable) to "1."   |

# MPLAB<sup>®</sup> IDE User's Guide

---

*Erase Configuration Bits* Sets all available configuration bits and ID location bits to "1." If you then reload your HEX file which has configuration data or your project which has configuration bits defined, these values will change in the Programmer Status dialog. You can use this to override the values in your code by selecting this after you have loaded your HEX file or rebuilt your project.

*Reset Programmer* Resets the PICSTART Plus hardware and reestablishes RS-232 communications. Use this option if power has been disconnected from the programmer. This option does not reset programming information in the Program Memory window, configuration bits, or IDs.

## PRO MATE Programmer Menu

*Enable/Disable Programmer* Enable or disable the programmer. Once the programmer is enabled, this menu item changes to Disable Programmer.

*Program/Verify* Opens the Program/Verify dialog which allows you to program the data as shown in the Program Memory window or verify that the data in the device in the PRO MATE socket matches data in the Program Memory window. You can select to program or verify only the program memory or other memory areas on the target device.

*Read Device* Opens the Read Device dialog which allows you to read the program and configuration bits of the device. You can set the program memory range and the other read options.

*Blank Check All* Checks that the device is completely blank (all bits are set to a "1"). This will also check that all configuration bits set to a "1" (unprogrammed state).

*Blank Check OTP* This function is intended for use with OTP devices that come with factory programmed configuration bits. Before using this function, set the displayed configuration bits to match the factory programmed settings. The function verifies that all program memory bits are set to "1" and that the configuration bits match the settings displayed in the Configuration Bits window.

*Display Error Log* When you have programmed a device or verified a device, an error window will show you data from memory in the device that do not match the corresponding memory in MPLAB IDE.

# MPLAB IDE Toolbar and Menu Options

---

<i>Erase Program Memory</i>	Sets all bits in the MPLAB IDE Program memory, Calibration memory, and Data memory (if applicable) to “1.”
<i>Erase Configuration Bits</i>	Sets all available configuration bits and ID location bits to “1.” If you then reload your HEX file which has configuration data or your project which has configuration bits defined, these values will change in the Programmer Status dialog. You can use this to override the values in your code by selecting this after you have loaded your HEX file or rebuilt your project.
<i>Reset Voltages</i>	Sets VDDMIN, VDDMAX, and VPP to their default values for the selected device.
<i>Transfer to PRO MATE</i>	Transfers device information to the PRO MATE programmer.
<i>Transfer from PRO MATE</i>	Transfers device information from the PRO MATE programmer.
<i>Generate SQTP<sup>SM</sup> File</i>	Generate an SQTP file for device serialization.
<i>Load SQTP File</i>	Load an SQTP file from disk.
<i>Download PRO MATE Operating System</i>	Download the latest version of the operating system to the programmer.
<i>Establish Communications</i>	Establishes or reestablishes RS-232 communications with the device programmer. Use this if power has been disconnected from the programmer. This does not reset programming information in the Programmer Status Dialog, configuration bits, or IDs.

## 1.9 Options Menu

### 1.9.1 Development Mode Options

Select *Options > Development Mode* to open a dialog box that allows you to change settings in the following areas:

- Tools
- Ports
- Clock
- Memory
- Configuration
- Power
- Pins
- Break options

#### 1.9.1.1 Tools

Select *Options > Development Mode* and click the **Tools** tab to change the current Development Mode setting and select a processor module or device.

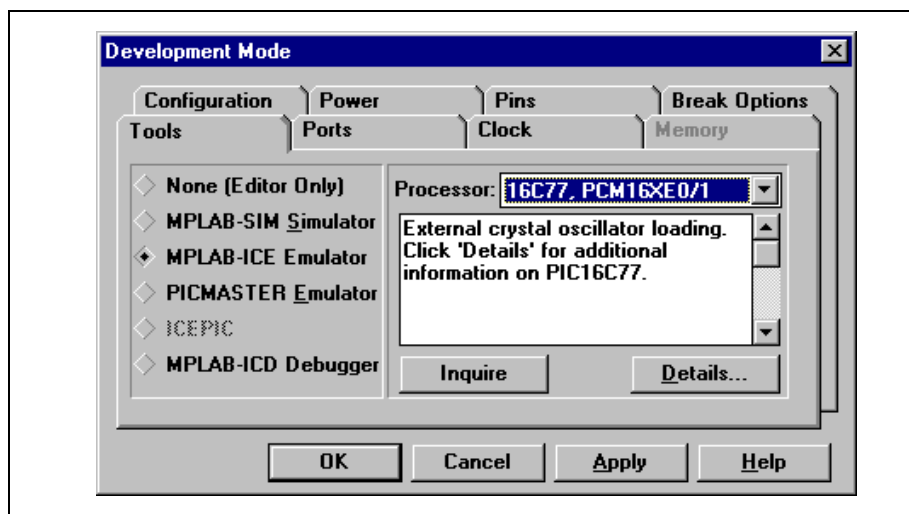
Click **Details** and double-click the highlighted item in the list area to view complete information on the inherent limitations for your selected device.

If you want to see which device your emulator is currently configured for, make sure that the correct tool (MPLAB ICE 2000 or PICMASTER emulator) is selected and click **Inquire**. The device will appear in the Processor box.

If you would like to save all of the changes you have made in the Development Mode dialog so far, click **Apply**.



# MPLAB IDE Toolbar and Menu Options



**Figure 1.38: Development Mode Dialog**

- None (Editor Only)* Selects the Editor Only mode and displays EO on the Status Bar. In Editor Only mode, all emulator and simulator functions are disabled. You can only edit, compile, and perform project management operations.  
Processor: Selects the processor that you are using for development.
- MPLAB SIM Simulator* Selects the MPLAB SIM simulator mode.  
Processor: Selects the processor that you are simulating.
- MPLAB ICE Emulator* Selects the MPLAB ICE 2000 Emulator mode (if connected) and displays Ice on the Status Bar.  
Processor: Selects the processor that you are emulating.
- PICMASTER Emulator* Selects the PICMASTER Emulator mode (if connected) and displays Em on the Status Bar.  
Processor: Selects the processor that you are emulating.
- ICEPIC Emulator* Selects the ICEPIC Emulator (if connected).  
Processor: Selects the processor that you are emulating.
- MPLAB ICD Debugger* Selects the MPLAB ICD in-circuit debugger (if connected) and displays ICD on the Status Bar.  
Processor: Selects the processor that you are debugging.
- Cancel* Click **Cancel** to cancel your selection and exit this display.
- OK* Click **OK** to accept your selection and exit this display.

# MPLAB<sup>®</sup> IDE User's Guide

## 1.9.1.2 Device Ports

Select *Options > Development Mode* and click the **Ports** tab to set the port for current Development Mode tool operation.

If you would like to save all of the changes you have made in the Development Mode dialog so far, click **Apply**.

*(Development Mode) Port* Select the port over which the tool will communicate with the PC in this development mode. The items in this box are specific to the development mode you have selected on the Tools tab of the Development Mode dialog.

### MPLAB ICE 2000 Emulator:

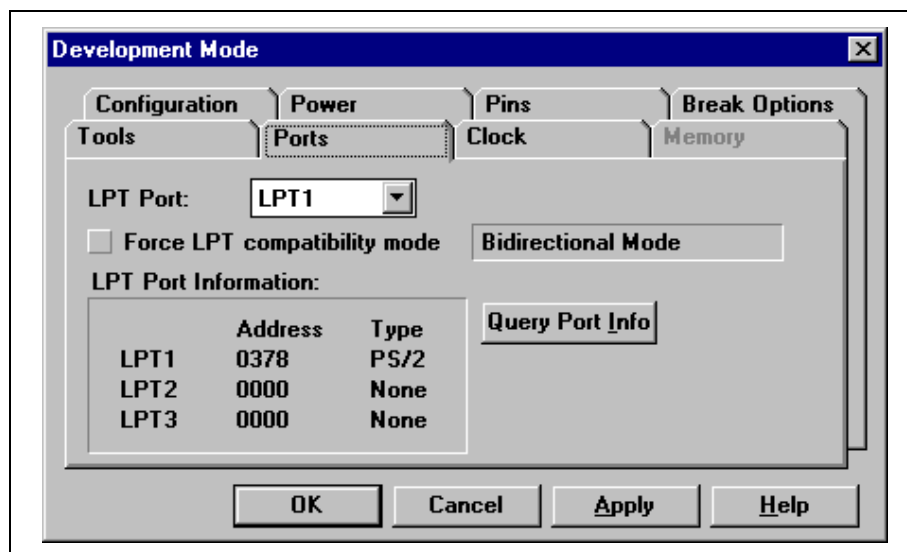


Figure 1.39: Ports Dialog – MPLAB ICE 2000

*LPT Port* Choose the LPT port you will use to communicate with MPLAB ICE 2000 (e.g., LPT1). The LPT port is also called the parallel or printer port. We recommend that your LPT port be configured for bidirectional operation (ECP preferred; EPP or PS/2 if no ECP). This may require changing the LPT configuration in your BIOS.

*Force LPT Compatibility Mode* Check this item to force the MPLAB ICE 2000 to communicate in compatibility mode rather than the usual bidirectional mode. The current operational mode appears to the right of this item.

# MPLAB IDE Toolbar and Menu Options

*LPT Port Information* Click **Query Port Info** to view complete information on your PC's LPT port configuration.

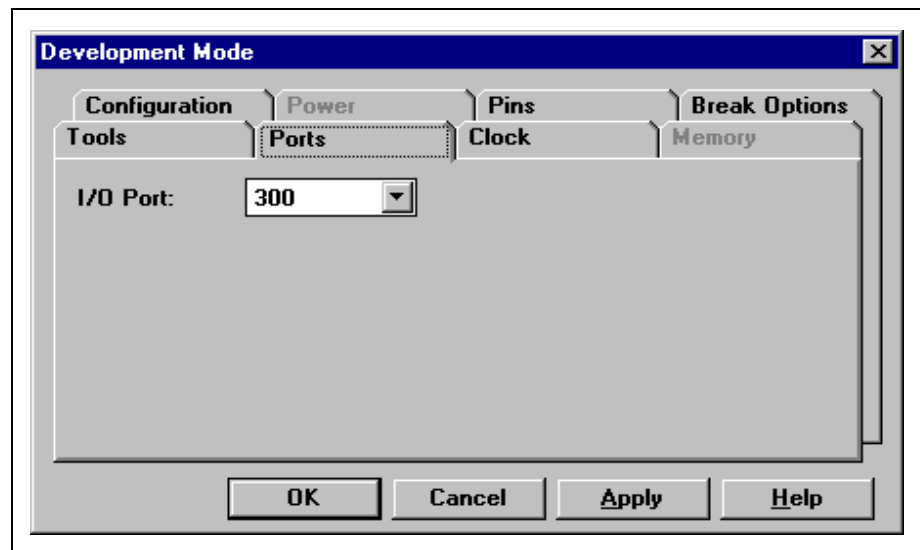
For more information on MPLAB ICE 2000 communications, refer to the *MPLAB ICE In-Circuit Emulator User's Guide* (DS51159).

## WARNING



**Do not** select the MPLAB ICE emulator mode if any other device (e.g., printer, scanner, zip drive) is installed on the parallel port or **permanent damage to that device** may result. See the *MPLAB ICE In-Circuit Emulator User's Guide* (DS51159) for more details.

## PICMASTER Emulator:

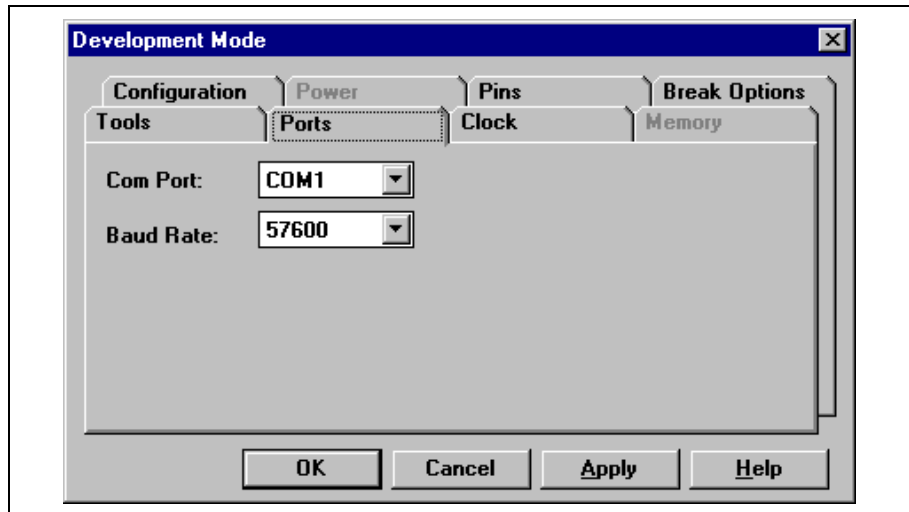


**Figure 1.40: Ports Dialog – PICMASTER Emulator**

*I/O Port* Select the I/O Port address that matches the base address your PICMASTER PC Interface Card.

For more information on PICMASTER emulator communications, refer to the *PICMASTER User's Guide* (DS51037).

## ICEPIC Emulator:



**Figure 1.41: Ports Dialog – ICEPIC Emulator**

*Com Port* Select the Com Port that the ICEPIC emulator is connected to.

*Baud Rate* Select the baud rate at which data will travel over the com port.

For more information on ICEPIC emulator communications, refer to the *ICEPIC User's Guide* (DS51103).

# MPLAB IDE Toolbar and Menu Options

## 1.9.1.3 Clock Frequency

Select *Options > Development Mode* and click the **Clock** tab to set the clock frequency. You can set the frequency in MHz, kHz, and Hz. Refer to the specific device's data sheet to determine the supported frequency range.

If you would like to save all of the changes you have made in the Development Mode dialog so far, click **Apply**.

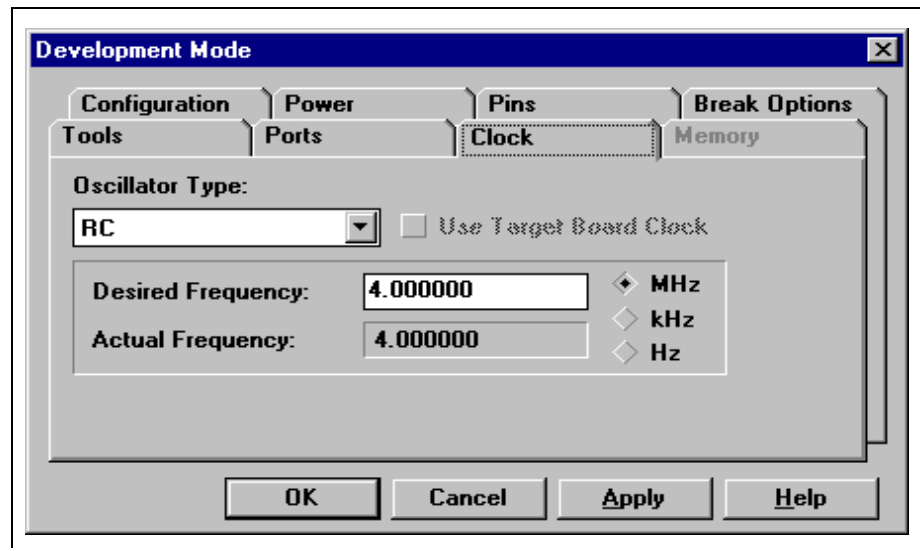


Figure 1.42: Processor Clock Dialog

1. Select the type of oscillator for an emulation system. The available oscillator selections are dependent on the selected processor.
2. Select Use Target Board Clock if appropriate for your MPLAB ICE 2000 application.
3. Enter the desired frequency and select MHz, kHz, or Hz. Refer to the specific device's data sheet to determine the supported frequency range.

**Note:** The Actual Frequency is for use with MPLAB ICE 2000 only.

4. Click **Apply** to accept the new setting.

# MPLAB<sup>®</sup> IDE User's Guide

## 1.9.1.4 Memory

Select *Options > Development Mode* and click the **Memory** tab to set the memory configuration being used.

**Note:** This option is not available for all processors.

If you would like to save all of the changes you have made in the Development Mode dialog so far, click **Apply**.

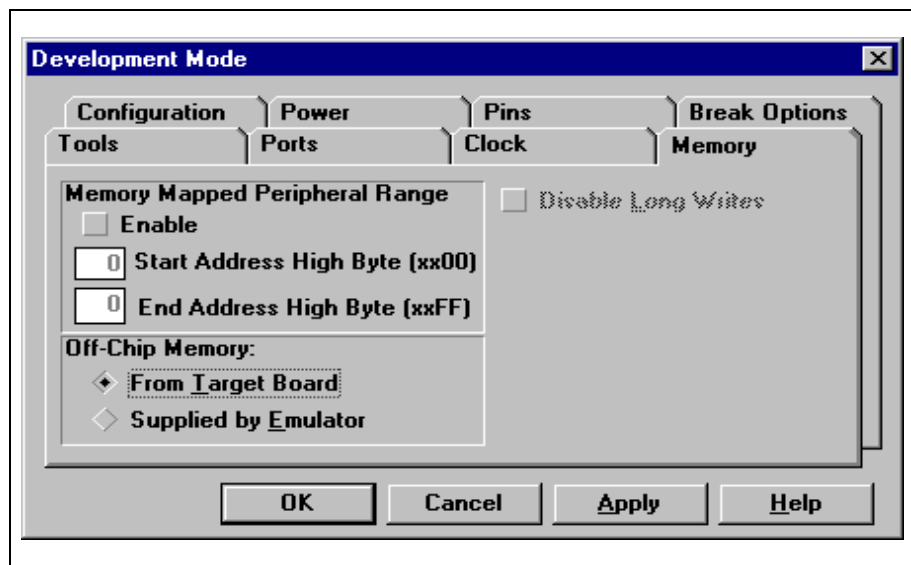


Figure 1.43: Memory Dialog

*Memory Mapped Peripheral Range* Enable/Disable the memory mapped peripheral range, and specify the starting and ending high byte address. If the End Address High Byte is set to 0xFF, the program counter in the MPLAB IDE will get stuck at 0xFFFF while operating in external microcontroller mode. The highest value that can be entered for the high byte is 0xFE. To recover, change the high byte from 0xFF to 0xFE or lower, power the emulator pod off and on, and select *Debug>System Reset*.

*Off-Chip Memory* Select off-chip memory From Target Board or Supplied By Emulator.

**Note:** The Processor Mode (*Options> Development Mode*, Configuration tab) must be set to Microprocessor or Extended Microcontroller in order for off-chip memory to be used.

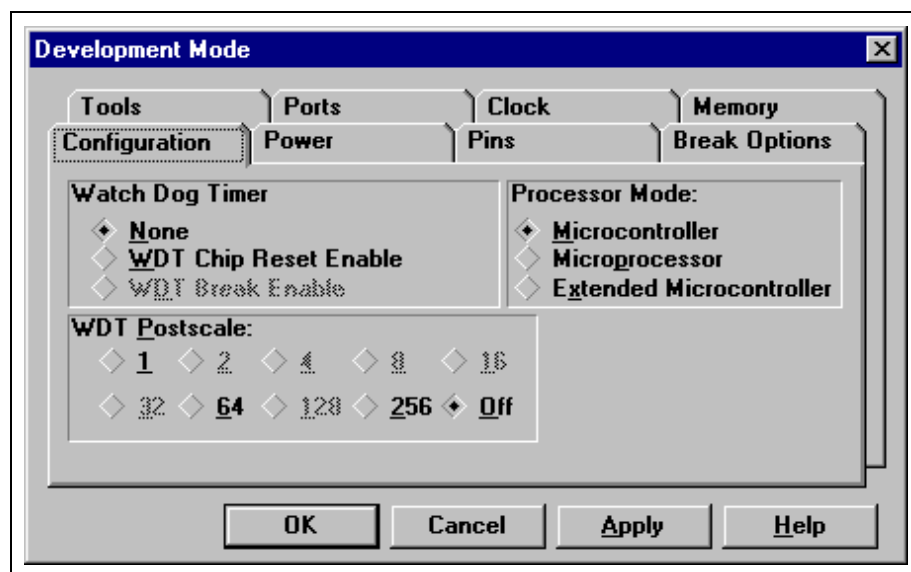
# MPLAB IDE Toolbar and Menu Options

*Disable Long Writes* Disable/enable long writes. It is possible to use the table write instructions to trace values in Data RAM with the PIC17C42; however, if the table write is to internal memory, you should select Disable Long Writes. If this is not selected, then MPLAB IDE will wait for an interrupt to exit the long write. These table write instructions should be removed before a device is programmed.

## 1.9.1.5 Configuration

Select *Options > Development Mode* and click the **Configuration** tab to change the settings for the watchdog timer and processor mode.

If you would like to save all of the changes you have made in the Development Mode dialog so far, click **Apply**.



**Figure 1.44: Configuration Dialog**

*Watchdog Timer* Enable/disable the Watchdog Timer.  
None: Disables the Watchdog Timer.  
WDT Chip Reset Enable: Resets the processor when the Watchdog Timer times out.  
WDT Break Enable: Halts the processor when the Watchdog Timer times out.  
In addition to setting the Watchdog Timer, be sure to set the OPTIONS register according to the device's data sheet.

# MPLAB<sup>®</sup> IDE User's Guide

---

*WDT Prescale* For PIC17CXXX and PIC18CXXX processors: If you are using the Watchdog Timer, remember to set its prescaler. This option is used for selecting the prescaler value.

For all other processors, the WDT prescaler is set under software control. For PIC12CXXX, PIC14000 and PIC16CXXX devices, this is set in the OPTIONS register. See your device data sheet for more information.

*Processor Mode* Select the processor mode, if available for the selected processor. The processor mode must be set to Microprocessor or Extended Microcontroller in order for off-chip memory to be used. Refer to individual data sheets for more information on each mode.

Microcontroller: Accesses internal program memory only.

Microprocessor: Accesses external program memory only.

Extended Microcontroller: Accesses internal and external program memory.



# MPLAB IDE Toolbar and Menu Options

## 1.9.1.6 Power

Select *Options > Development Mode* and click the **Power** tab to set the processor's power source and voltage.

If you would like to save all of the changes you have made in the Development Mode dialog so far, click **Apply**.

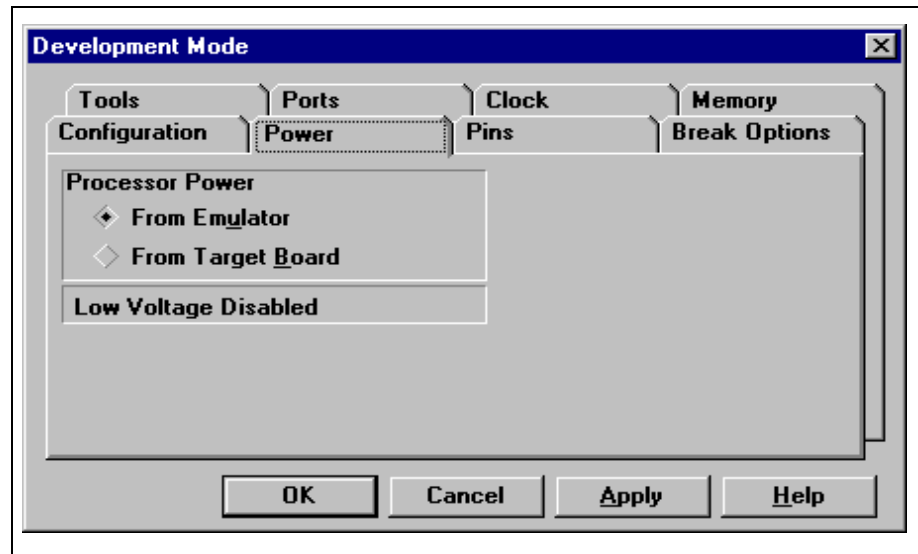


Figure 1.45: Power Dialog

*Processor Power* For an emulator system, state whether the processor gets its power From Emulator or From Target Board.

*Low Voltage Enabled/Disabled* This line indicates whether low voltage is enabled or disabled. The emulator determines whether low voltage is enabled or disabled.

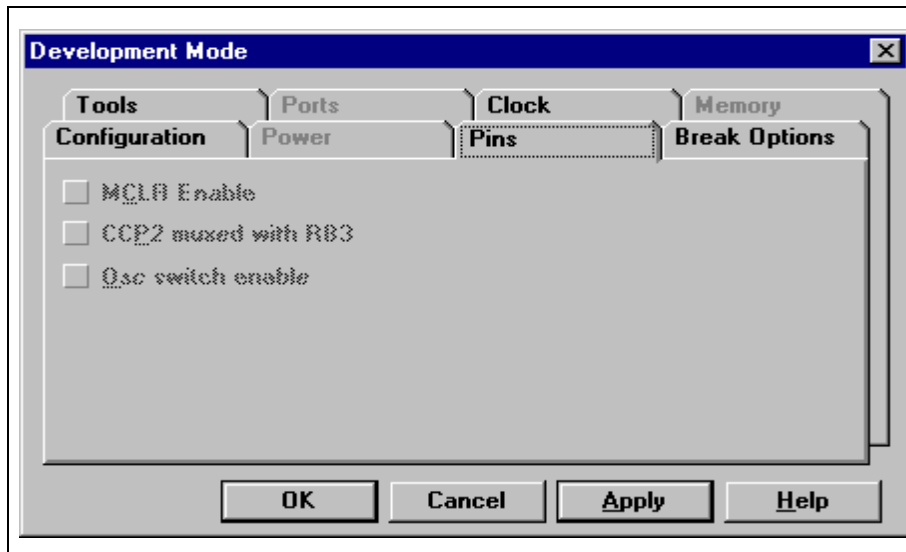
# MPLAB<sup>®</sup> IDE User's Guide

---

## 1.9.1.7 Pins

Select *Options > Development Mode* and click the **Pins** tab to adjust pin settings.

If you would like to save all of the changes you have made in the Development Mode dialog so far, click **Apply**.



**Figure 1.46: Pins Dialog**

$\overline{MCLR}$  Enable    Enable or disable Master Clear.

CCP2 muxed with RB3    Mux CCP2 with RB3 or RB3 as I/O.

Osc switch enable    Enable or disable oscillator switch.

These options are only available for some devices. If they are not available, they will be grayed-out.

# MPLAB IDE Toolbar and Menu Options

## 1.9.1.8 Break Options

Select *Options > Development Mode* and click the **Break Options** tab to change the global break and trace point environment options.

If you would like to save all of the changes you have made in the Development Mode dialog so far, click **Apply**.

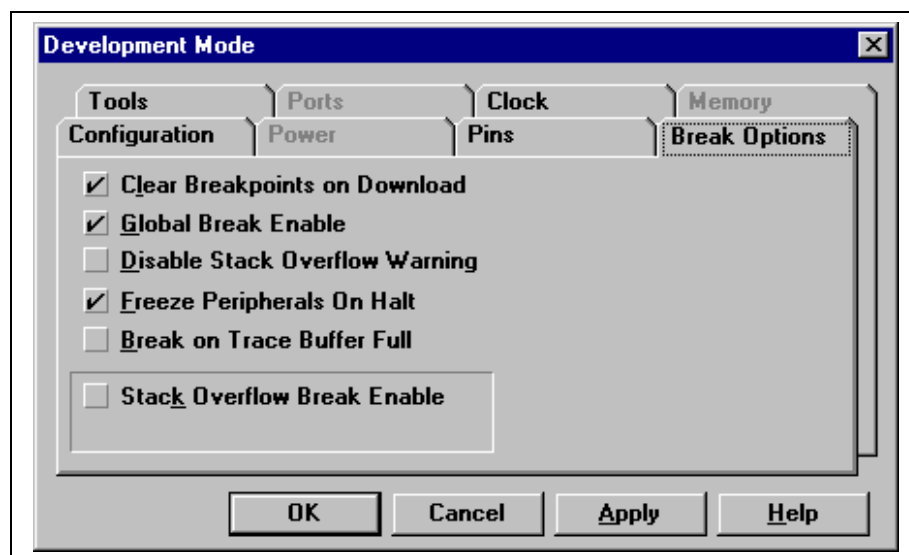


Figure 1.47: Break Options Dialog

The following options apply to all development modes except Editor Only, except as otherwise indicated.

*Clear Break Points on Download* When selected, removes all break points, trace points, trigger points, and pass count addresses on download to the emulator.

*Global Break Enable* When selected, enables all break points. If Global Break Enable is not selected, then all break points are disabled. Global Break Enable is also available from the Status Bar

*Disable Stack Overflow Warning* When selected, prevents a warning message from appearing if a stack overflow or underflow occurs. If you want to see a warning in the event of a stack overflow or underflow, clear this switch.

# MPLAB<sup>®</sup> IDE User's Guide

---

*Freeze Peripherals on Halt* This option freezes peripherals (timers, ports, PWM, etc.) on a halt operation. By default, this option is on so that you will get a true picture of the status. When Freeze Peripherals on Halt is on, you can write to a port after a halt, but you cannot read from the port until you perform a single-step. You may wish to turn this option off if a peripheral must continue to run after a halt. This option facilitates troubleshooting of timers in situations when timers are continuing to get set/reset and cleared after a halt.

**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.

*Break on Trace Buffer Full* This option is not available for the MPLAB ICE 2000 or the ICEPIC emulator.

*Full* When selected, causes the processor to halt when the trace buffer is full. The trace buffer is full when it captures 8K instructions/cycles. For MPLAB ICE 2000, this function is available through the Complex Trigger dialog.

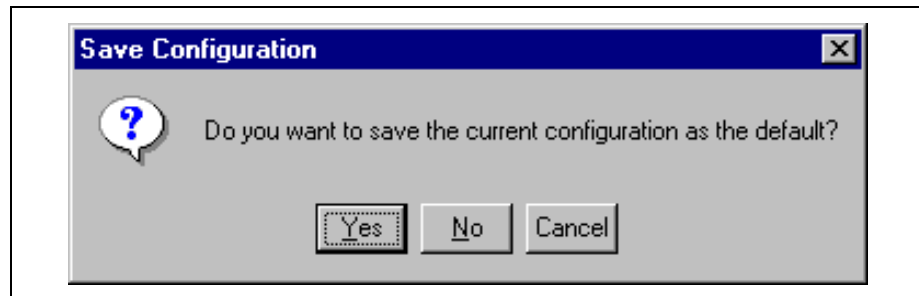
*Stack Overflow Break Enable* When selected, causes the processor to halt as soon as a stack overflow or underflow occurs. If you do not wish to stop processing specifically for a stack overflow or underflow, clear this switch. The setting of Disable Stack Overflow Warning determines whether a message will be displayed due to the overflow or underflow.

# MPLAB IDE Toolbar and Menu Options

## 1.9.2 Window Setup

### 1.9.2.1 Save Setup

Select *Options > Window Setup > Save Setup* to save the current configuration to a file. The default extension of the configuration file is \*.CFG.



**Figure 1.48: Save Setup as Default Message Box**

Save Setup only saves the currently open windows—not the break, trace, and trigger points.

- Yes** If you answer **Yes** to the question, “Do you want to save the current configuration as the default?” MPLAB IDE will save the current configuration as the default user configuration that MPLAB IDE loads during start-up.
- No** If you answer **No**, MPLAB IDE displays the Save Configuration dialog box. At this point, enter the name and path where you want to save the setup.
- Cancel** Select **Cancel** to exit the Save Configuration dialog without saving.

# MPLAB<sup>®</sup> IDE User's Guide

---

## 1.9.2.2 Load Setup

Select *Options > Window Setup > Load Setup* to load a configuration setup from a file previously stored by the *Options > Window Setup > Save Setup* command. Select the file that you want to read from the dialog box and press **Enter**. The default extension of the setup file is \*.CFG.

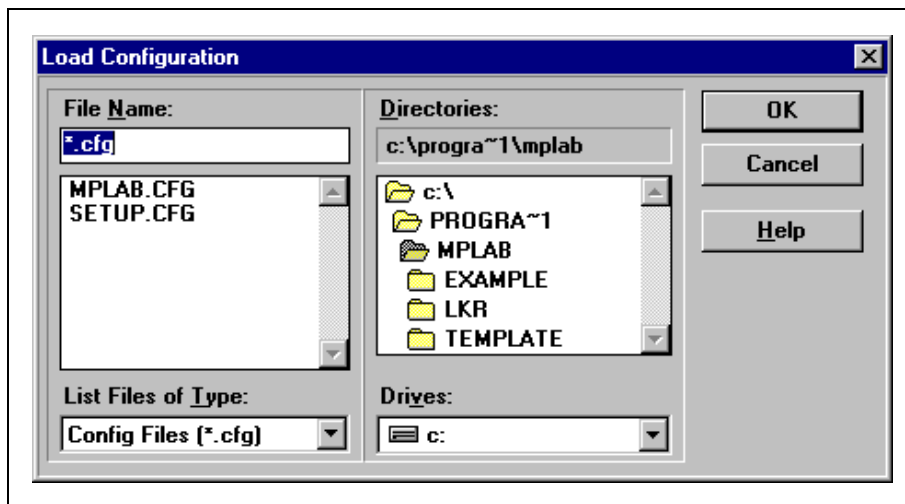


Figure 1.49: Load Setup Dialog

**Note:** Setup.cfg is a configuration file used by the PRO MATE II programmer when programming Secure Data Product devices. Do not overwrite this file.

## 1.9.2.3 Default Configuration

Select *Options > Window Setup > Default Configuration*, to load the default user configuration the next time you start MPLAB IDE.

## 1.9.3 Current Editor Modes

Select *Options > Current Editor Modes* to configure the editor modes that apply to the current window.

# MPLAB IDE Toolbar and Menu Options

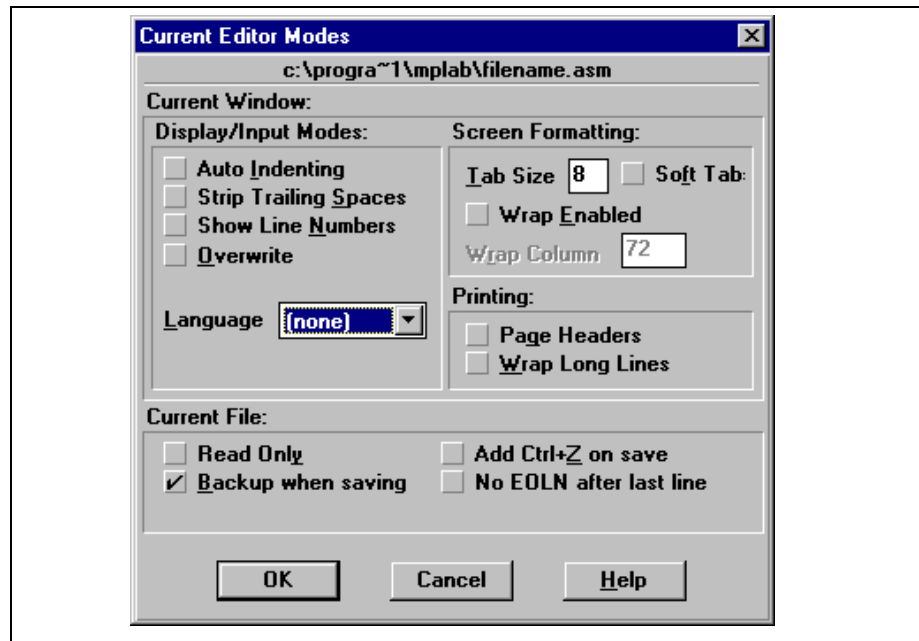


Figure 1.50: Current Editor Modes Dialog

## Display/Input Modes

These settings control how text is displayed in the window and any special actions MPLAB Editor takes as you type.

*Auto Indenting* Indents new lines to the same level as the preceding line.

*Strip Trailing Spaces* Removes white space from the end of a line whenever you press **Enter**.

*Show Line Numbers* Displays line numbers in the window.

*Overwrite* Overwrites the characters under the cursor with the characters you type. Otherwise MPLAB Editor inserts characters at the cursor.

*Language* Select from this list to associate a language (such as C or Pascal) with the file type. This associates certain formatting settings (like indenting) with files of the type you are setting modes for.

# MPLAB<sup>®</sup> IDE User's Guide

---

## Screen Formatting

These settings affect tabs and text wrapping.

<i>Tab Size</i>	Defines the width of a tab character.
<i>Soft Tabs</i>	Inserts spaces instead of tabs when Soft Tabs is checked.
<i>Wrap Enabled</i> <i>Wrap Column</i>	To use line wrapping, place a checkmark to the left of Wrap Enabled. In the Wrap Column box, specify the column at which you want lines to wrap. Clear the checkmark if you do not want lines to wrap.

## Printing

These settings affect how MPLAB Editor prints the file. These settings become the defaults for the Print File command, but can be overridden at the time of printing.

<i>Page Headers</i>	Prints each page headed with the file name, the page number and the date.
<i>Wrap Long Lines</i>	Wraps lines that are too wide to fit the page to the next line during printing.

## Current File

These settings are the file modes that apply to file.

<i>Read Only</i>	Prevents users from saving files of this type.
<i>Backup When Saving</i>	Makes a backup copy of any existing file of the same name when you write the file to disk.
<i>Add Ctrl+Z On Save</i>	Automatically adds a Ctrl+Z character to the end of the file on disk when it saves the file.
<i>No EOLN after last line</i>	Writes an end-of-line terminator (CR-LF or LF) after the last character of the last line when it saves the file.

## 1.9.4 Reset Editor Modes

Select Options > Reset Editor Modes to reset the modes that apply to the current window to the default values.



# MPLAB IDE Toolbar and Menu Options

## 1.9.5 Environment Setup

Select *Options > Environment Setup* to open a dialog box that allows you to change settings in the following areas:

- General Options
- Project Template Options
- Files
- Default Editor Modes
- Key Mappings
- Colors

### 1.9.5.1 General Options

When you select *Options > Environment Setup* and click the **General** tab, a dialog displays settings for screen font, toolbar setup, symbol display width, and global switches.

If you would like to save all of the changes you have made in the Environment Settings dialog so far, click **Apply**.

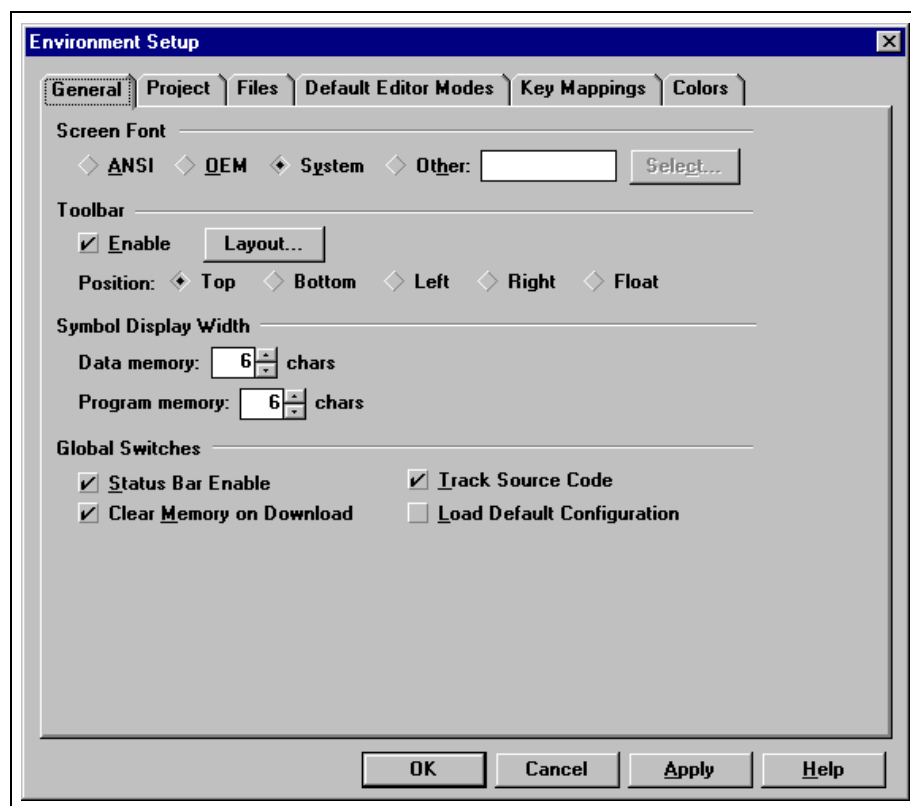


Figure 1.51: General Dialog

# MPLAB<sup>®</sup> IDE User's Guide

---

## 1.9.5.1.1. Screen Font

When you select *Options > Environment Setup* and click the **General** tab, the Screen Font area allows you to select a fixed point font for MPLAB IDE screen displays.

1. Select ANSI, OEM, System or Other in the Screen Font area of the Environment Settings dialog.
2. If you select Other, click **Select** to display the Font dialog.
3. In the Font dialog, select the desired screen font, then click **OK** to return to the Environment Settings dialog.

If you would like to save all the changes you have made in the Environment Setup dialog so far, click **Apply**.

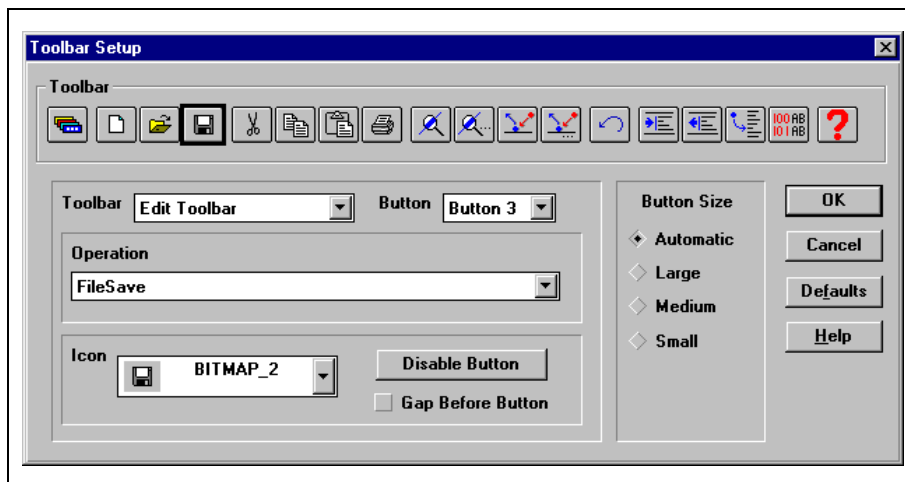
## 1.9.5.1.2. Toolbar Setup

Select *Options > Environment Setup* and click the **General** tab to set up the MPLAB IDE toolbars.

On this tab you can:

- Select a location for the toolbar on the screen and change other display settings. Available locations are: Top, Bottom, Left, Right, and Float.
- Select the Enable checkmark to have MPLAB IDE display the toolbar.

Click **Layout** in the General tab of the Environment Setup dialog to display the Toolbar Setup dialog:



**Figure 1.52: Toolbar Setup Dialog**

From the Toolbar Setup dialog you can:

- Add a button to the toolbar or edit an existing button's operation
- Remove a button from the toolbar
- Group toolbar buttons

# MPLAB IDE Toolbar and Menu Options

---

<i>Toolbar</i>	Selects which toolbar to edit: Edit, Debug, Project, or User.
<i>Button</i>	Selects a toolbar button location. The toolbar has 16 available locations.
<i>Operation</i>	Selects the operation that MPLAB IDE will perform for the selected button location.
<i>Icon</i>	Selects a button to display in the selected button location.
<i>Disable Button</i>	Disables the toolbar button at the selected location on the toolbar.
<i>Gap Before Button</i>	Inserts a small gap before the toolbar button.
<i>Button Size</i>	Changes the toolbar button size. Button size options are: Automatic, Large, Medium, and Small.
<i>Ok</i>	Defines the toolbar as currently shown.
<i>Cancel</i>	Returns the toolbar to its previous state.
<i>Defaults</i>	Restores ALL toolbars to their default settings, undoing any changes you have made to any toolbar.

If you would like to save all the changes you have made in the Environment Setup dialog so far, click **Apply**.

## Adding and Editing Toolbar Buttons

If you wish to be able to perform an operation from the Toolbar for which there is no toolbar button (such as FileSave), you can either add a button and associate it with the desired operation, or modify an existing button.

1. Select *Options > Environment Setup* and click the **General** tab to view the existing toolbar button setup. Click **Layout** to display the Toolbar Setup dialog.
2. In the Toolbar drop-down list, select the toolbar to which you wish to add a button. The toolbar you selected appears in the Toolbar area at the top of the Toolbar Setup dialog.
3. In the Toolbar area at the top of the Toolbar Setup dialog, click the position to which you wish to assign the new button.

**Note:** If there is not currently a red circle with a slash in the position, the button and operation you select in the following steps will replace the current button and operation. To preserve the currently assigned buttons, add the desired button to the right of the currently assigned buttons. You cannot add a button beyond the 17th position on the toolbar.

# MPLAB<sup>®</sup> IDE User's Guide

---

4. From the Icon drop-down list, select the button you wish to display on the toolbar.
5. From the Operation drop-down list, select the operation you wish to have performed when users click on the button (e.g., EditTextIndent).
6. Click **OK** to save your changes (or click **Cancel** to discard them) and return to the Environment Setup dialog.

## Disabling (Removing) a Toolbar Button

If you do not wish a certain operation to be available from a toolbar you can disable its toolbar button position. This removes it from the toolbar.

1. Select *Options > Environment Setup* and click the **General** tab to view the existing toolbar button setup. Click **Layout** to display the Toolbar Setup dialog.
2. In the Toolbar drop-down list, select the toolbar that contains the button you wish to disable (remove). The toolbar you selected appears in the Toolbar area at the top of the Toolbar Setup dialog.
3. In the Toolbar area at the top of the Toolbar Setup dialog, click the button you wish to disable (remove). As you select the toolbar button, the operation that is performed when the user clicks the button appears in the Operation drop-down list.
4. Click **Disable Button** to the right of the Icon drop-down list. A red circle with a slash appears in the button's position, replacing the button.
5. Click **OK** to save your changes (or click **Cancel** to discard them).

When you return to the MPLAB IDE desktop, the position where the button was located will be blank.

## Grouping the Buttons on a Toolbar

You can group several buttons on a together to make it easier to perform logically related operations. For example, you may wish to group the file operation buttons together on the Edit toolbar, and separate the other buttons from them by a small space. Groups of buttons are separated from other buttons by a small space on the toolbar.

1. Select *Options > Environment Setup* and click the **General** tab to view the existing toolbar button setup. Click **Layout** to display the Toolbar Setup dialog.
2. In the Toolbar drop-down list, select the toolbar you wish to customize. The toolbar you selected appears in the Toolbar area at the top of the Toolbar Setup dialog.
3. In the Toolbar area at the top of the Toolbar Setup dialog, click the first (left most) button in each group, and select **Gap Before**. Click the remaining buttons in the group and deselect (clear) **Gap Before**.
4. Click **OK** to save your changes (or click **Cancel** to discard them).

# MPLAB IDE Toolbar and Menu Options

---

## 1.9.5.1.1. Symbol Display Width

When you select *Options > Environment Setup* and click the **General** tab, the Symbol Display Width area allows you to specify how many character spaces MPLAB IDE allocates when displaying symbolic information.

*Register Variables* Allows you to select a width of 6 characters wide to 32 characters.

*Address Labels* Allows you to select a width of 6 characters wide to 32 characters.

## 1.9.5.1.2. Global Switches

When you select *Options > Environment Setup* and click the **General** tab, the Global Switches area allows you to turn the following user selections on or off:

*Status Bar Enable* Turns the status bar on and off.

*Clear Memory on Download* When selected, this global switch clears memory before MPLAB IDE downloads to the emulator. This function sets all program memory bits to one.

*Load Default Configuration* When selected, MPLAB IDE loads the default user configuration at start-up. To change the default window setup, open the windows you want to load at start-up, select *Options > Window Setup > Save Setup* and click **Yes**. The MPLAB IDE window setup may contain any available MPLAB IDE window. MPLAB.CFG is the default user configuration file.

*Track Source Code* When selected, MPLAB IDE updates the current line in the source code when single stepping. You may wish to turn this feature off if you have a \*.HEX file, but no \*.COD file.

If you would like to save all the changes you have made in the Environment Setup dialog so far, click **Apply**.

# MPLAB<sup>®</sup> IDE User's Guide

## 1.9.5.2 Project Template Options

Select *Options > Environment Setup* and click the **Project** tab to display and edit the project language and path default options. The defaults you set here will be used in new projects you subsequently create. Existing projects will not be affected.

**Note:** To change the settings for an individual MPLAB IDE project, select *Project > Edit Project* and follow the instructions in Section 1, Chapter 3: Getting Started with MPLAB IDE – A Tutorial.

If you would like to save all of the changes you have made in the Environment Settings dialog so far, click **Apply**.

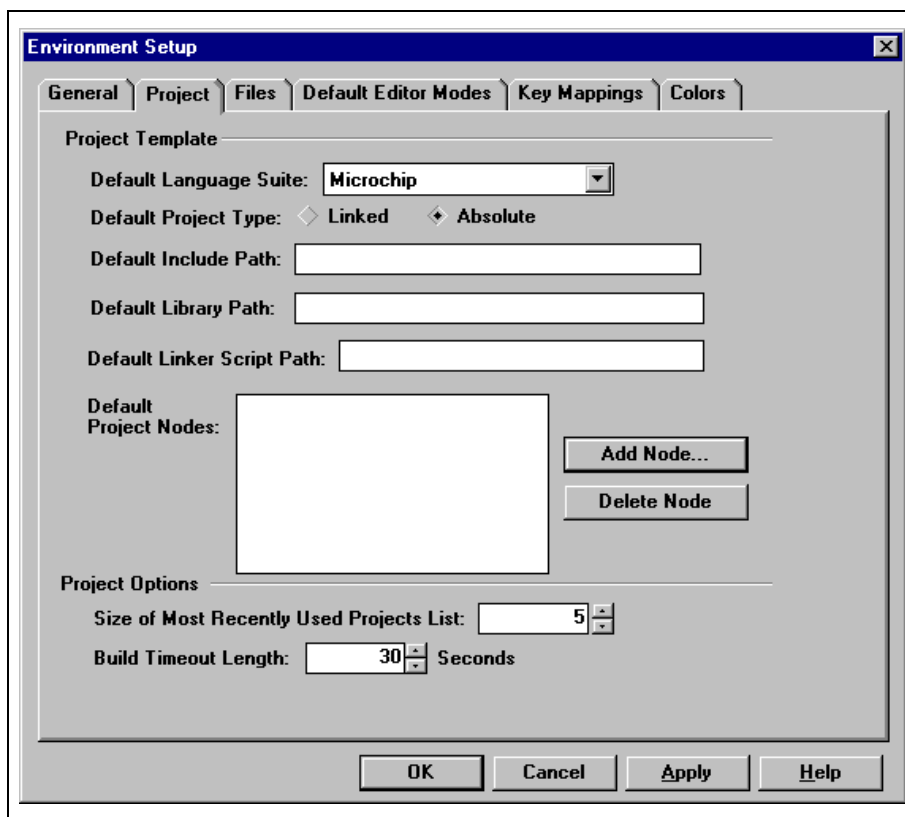


Figure 1.53: Project Template Dialog

# MPLAB IDE Toolbar and Menu Options

---

*Default Language Suite* You can select a Language Tool Suite (e.g., assembler, compiler) from the drop-down list at the top of the Project Templates tab of the Environment Setup dialog. You may need to install the language tools for this suite. You can select a suite that is not installed on your PC, but the project will not build.

If you installed your language tool before installing MPLAB IDE, you must give MPLAB IDE information about the language tool in the Install Language Tool dialog.

*Default Project Type* If you selected Microchip as the default language suite, indicate whether your projects will consist of multiple linked files or a single, absolute file.

*Default Include Path* In the Include Path box, enter the path to the default folder of included project files, e.g., Header files.

If the include path is not set for an individual project, Make Project will not be able to find the include files and will proceed to build the entire project. The build will be successful since the assembler/compiler will find the files via the MCC\_INCLUDE variable in the AUTOEXEC.BAT file. However, this is an inefficient way to make a project.

*Default Library Path* In the Library box of the Edit Project dialog, enter the path to the default project library file folder. This path may be set only if you have selected the Linked project type.

*Default Linker Script Path* In the Linker Script box of the Edit Project dialog, enter the path to the default folder of project linker script files. This path may be set only if you have selected the Linked project type.

Each Path box can contain one or more absolute or relative paths separated by semicolons. For example:

```
c:\mplab\projects\mpproj\include;  
c:\include\h;..\sys
```

# MPLAB<sup>®</sup> IDE User's Guide

---

*Default Project Nodes* Indicate any non-compiled (object, library, and linker script) nodes you wish to include in your project template. Click **Add Node** to specify the drive, folder and filename of each node you wish to add. To remove an existing node from your project template, select the node and click **Delete Node**. The folder from which you choose your files defaults to the project template's default browse folder. To select files from a different folder, select the drive from the Drives drop-down list at the bottom right of the Add Node dialog, double-click the folder in the Folders hierarchy at the top right of the Add Node dialog, and be sure that Source Files is selected in the Show files of type box below the list of files.

*Size of Most Recently Used Projects List* The Size of Most Recently Used Projects List allows you to select the length of the most recently used project list at the bottom of the MPLAB IDE Project menu. When you decrease this value, file names will be removed from the list. These names will not be remembered (redisplayed) when you increase this value later. File names that you use after you increase this value will appear in this list. The default size of this list is five.

*Build Timeout Length* Set the maximum time allowed for a build. If a build takes longer than the allotted time, a timeout will occur and no new HEX file will result. If Build Timeout is set to "off," no timeout will occur.



# MPLAB IDE Toolbar and Menu Options

## 1.9.5.3 File Options

Select *Options>Environment Setup* and click the **Files** tab to set the default browse folder for MPLAB IDE and adjust the number of files displayed in the Most Recently Used File List. If you would like to save all of the changes you have made in the Environment Settings dialog so far, click **Apply**.

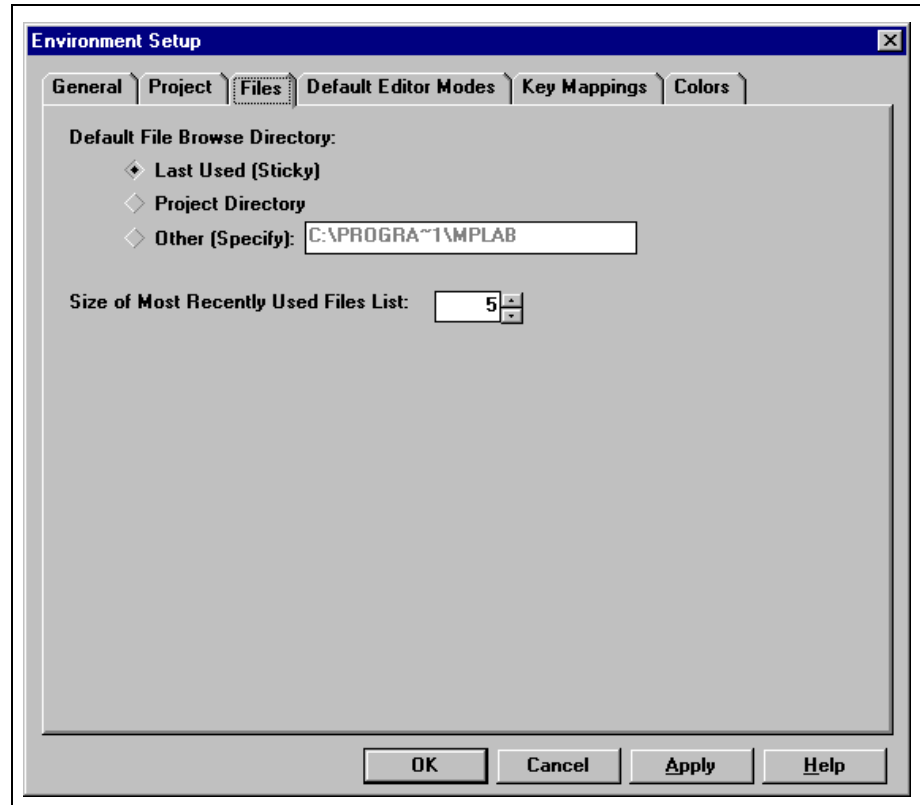


Figure 1.54: File Options Dialog

### 1.9.5.3.1 Default File Browse Directory

The Default File Browse Directory is the directory (folder) in which MPLAB IDE starts a file browse window.

To start from the last location a file was added, select Last Used. If the directory no longer exists, MPLAB IDE will use the MPLAB IDE directory.

To start in the root of the project directory, select Project Directory. If this directory doesn't exist, MPLAB IDE will use the highest parent directory that exists. If no project is open, MPLAB IDE will use the MPLAB IDE directory.

To specify a directory from which to start browsing, select Other, then enter the desired directory. This setting defaults to Project Directory.

# MPLAB<sup>®</sup> IDE User's Guide

## 1.9.5.3.2. Size of Most Recently Used Files List

The Size of Most Recently Used Files List allows you to select the length of the most recently used file list at the bottom of the MPLAB IDE File menu. When you decrease this value, file names will be removed from the list. These names will not be remembered (redisplayed) when you increase this value later. File names that you use after you increase this value will appear in this list. The default size of this list is five.

## 1.9.5.4 Default Editor Modes

MPLAB Editor associates a set of Window Modes with every edit window for existing and new files. The modes can be set with the Default Editor Modes tab from *Options > Environment Setup*. If you would like to save all of the changes you have made in the Environment Settings dialog so far, click **Apply**.

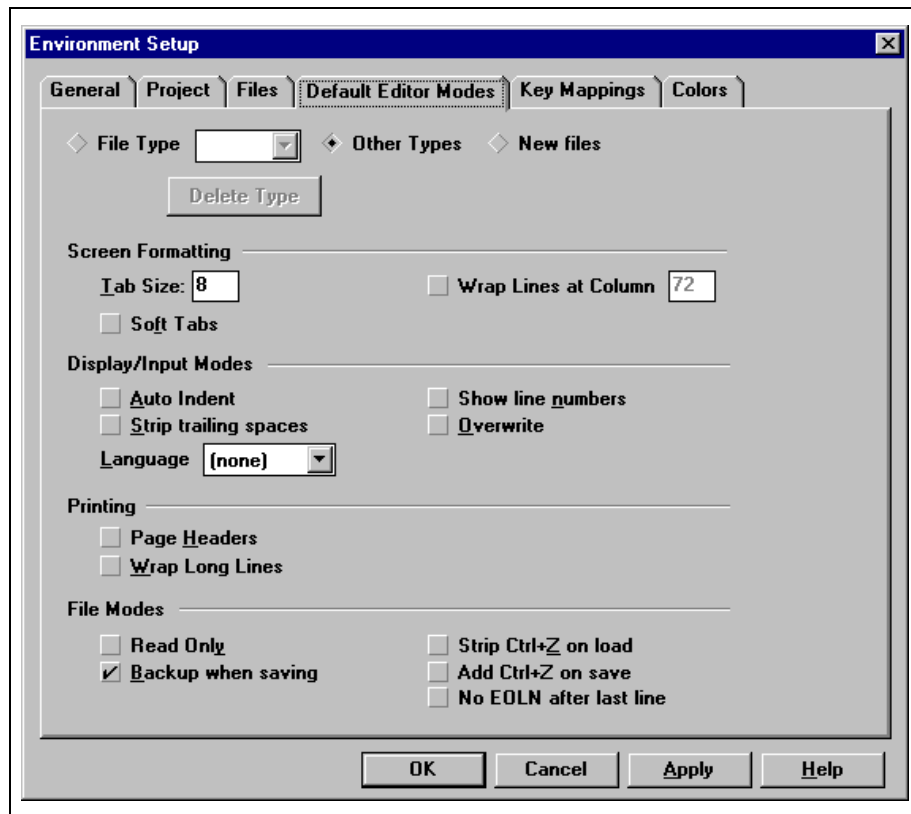


Figure 1.55: Editor Modes Dialog

To set the default editor modes for a given type of files, click the **File Type** button and either select the file type from the adjoining list or type a new one (with a leading “.”) into the edit control. This mode is applied when you use the *File > Open Source*, *File > View*, and *File > Save As* commands.

# MPLAB IDE Toolbar and Menu Options

---

---

1. To set the default editor modes for files created with the New File command, select New Files.  
Or, click the **Other Types** button to set default modes for file types not in the File Type list.
2. Check the mode boxes in the dialog to specify screen formatting, display/input, printing, and file save modes.
3. Click **Apply Settings** to save the settings for this file type.
4. Repeat steps 1 through 3 to set editor modes for other file types.

The modes you can set in the dialog are:

## Screen Formatting

These settings affect tabs and text wrapping.

*Tab Size* Defines the width of a tab character.

*Soft Tabs* Inserts spaces instead of tabs when Soft Tabs is checked.

*Wrap Lines at Column* Place a checkmark to the left of this option and specify the column at which you want lines to wrap. Clear the checkmark if you do not want lines to wrap.

## Display/Input Modes

These settings control how text is displayed in the window and any special actions MPLAB Editor takes as you type.

*Auto Indent* Indents new lines to the same level as the preceding line.

*Strip trailing spaces* Removes white space from the end of a line whenever you press **Enter**.

*Show line numbers* Displays line numbers in the window.

*Overwrite* Overwrites the characters under the cursor with the characters you type. Otherwise MPLAB Editor inserts characters at the cursor.

*Language* Select from this list to associate a language (such as C or Pascal) with the file type. This associates certain formatting settings (like indenting) with files of the type you are setting modes for.

# MPLAB<sup>®</sup> IDE User's Guide

---

## Printing

These settings affect how MPLAB Editor prints the text in the current window. These settings become the defaults for the *File > Print* command, but can be overridden at the time of printing.

*Page Headers* Prints each page headed with the file name, the page number and the date.

*Wrap Long Lines* Wraps lines that are too wide to fit the page to the next line during printing.

## File Modes

These settings are the file modes that apply to file.

*Read only* Prevents users from saving files of this type.

*Backup when saving* Makes a backup copy of any existing file of the same name when you write the file to disk.

*Strip Ctrl+Z on load* Automatically removes any **Ctrl+Z** character at the end of the file when it is loaded.

*Add Ctrl+Z on save* Automatically adds a **Ctrl+Z** character to the end of the file on disk when it saves the file.

*No EOLN after last line* Writes an end-of-line terminator (**CR-LF** or **LF**) after the last character of the last line when it saves the file.

# MPLAB IDE Toolbar and Menu Options

## 1.9.5.5 Key Mappings

Select *Options > Environment Setup* and click the **Key Mappings** tab to display and edit the MPLAB IDE key mappings. By mapping keyboard keys to MPLAB IDE functions, you can perform common operations quickly. You may use the existing key mappings or modify the key mappings to meet your specific needs for your current project. MPLAB IDE uses the binary initialization file, MPLAB.KEY, in the MPLAB IDE folder to record the key mapping values that will be carried from one session to another. If you would like to save all of the changes you have made in the Environment Settings dialog so far, click **Apply**.

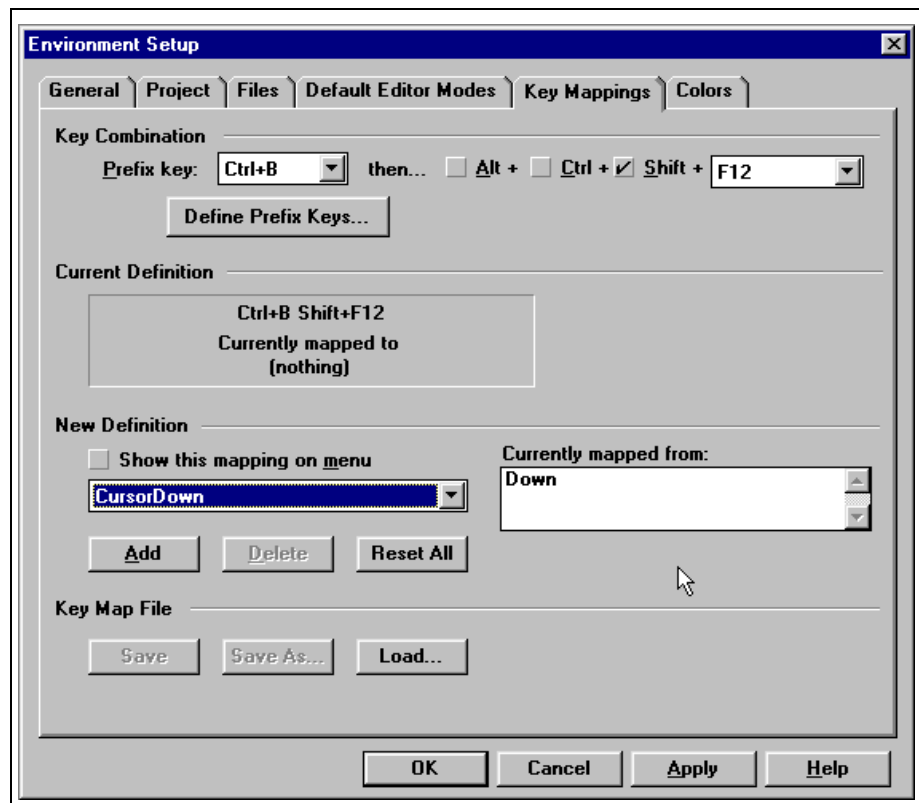


Figure 1.56: Key Mappings Dialog

By default, no prefix keys are enabled. This means that only single keys are available for mapping.

MPLAB IDE uses the binary initialization file, MPLAB.KEY, in the MPLAB IDE folder to record the key mapping values that will be carried from one session to another.

# MPLAB<sup>®</sup> IDE User's Guide

---

To create a key mapping:

1. To modify an existing key mapping, click **Load** to select the existing key mapping file.
2. Define a prefix key (optional).
3. Click the checkboxes below the prefix key to assign the key or key sequence (e.g., Shift + F2). The Current Definition box displays the current function that is mapped to the key or key sequence.
4. Select the "Show this mapping on menu" checkbox if you want this key mapping shortcut to appear to the right of the menu option on the MPLAB IDE menu.
5. Select the function that you want this key or key sequence to perform from the drop-down New Definition list. (Refer to "Appendix A: MPLAB IDE Key Mapping Functions" for a list of available MPLAB IDE key mapping functions.)
6. Click **Add** to add this mapping to the key mapping file.  
To change the function performed by an existing key mapping, select the key or key sequence, select the desired function, then click **Change**.  
To delete a key mapping, locate the key sequence and click **Delete**.
7. Click **Save** or **Save As** to save your key mapping.

# MPLAB IDE Toolbar and Menu Options

## 1.9.5.6 Colors

Select *Options > Environment Setup* and click the **Colors** tab to change the colors assigned to displayed data. If you would like to save all of the changes you have made in the Environment Settings dialog so far, click **Apply**.

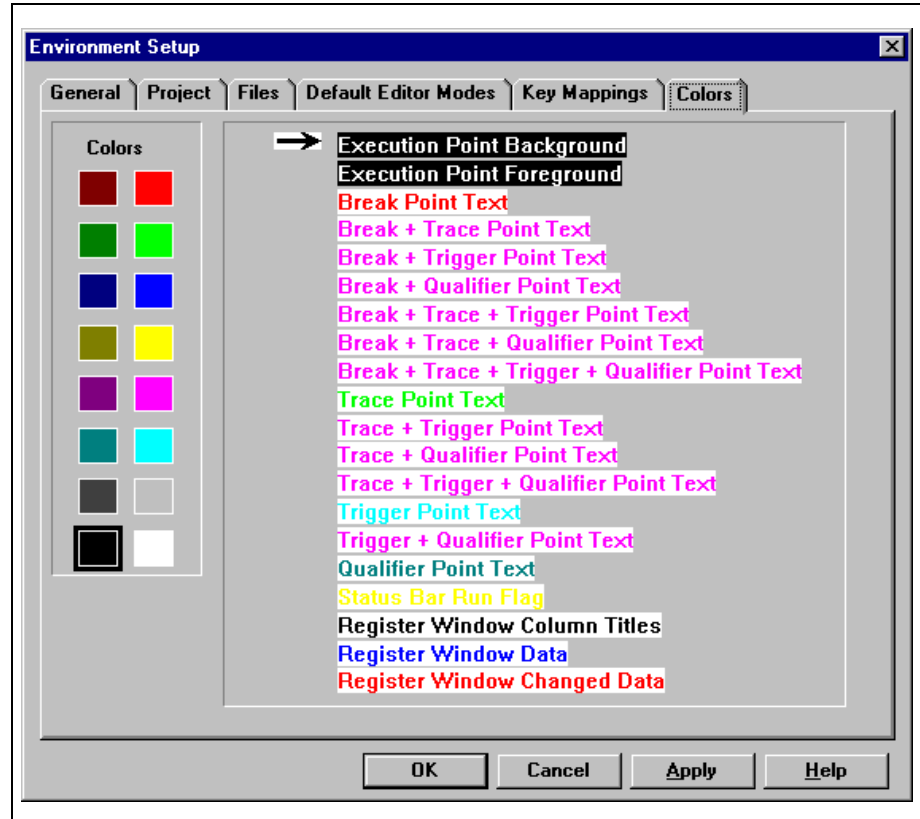


Figure 1.57: Colors Dialog

To change the color, select the text that you want to alter by clicking on it. Next, select the new color.

## 1.9.6 Programmer Options

### 1.9.6.1 Select Programmer

1. Select *Options > Programmer Options > Select Programmer* to select a programmer from the list of available programmers.
2. Select the programmer from the drop-down list in the Select Programmer dialog.
3. Click **OK**.



Figure 1.58: Select Programmer Dialog

### 1.9.6.2 Communications Port Setup

Select *Options > Programmer Options > Communications Port Setup* to select the comm port for the programmer you are using.

1. Select *Options > Programmer Options > Communications Port Setup* from the menu bar.
2. Select the comm port in the Communications Port Setup dialog.
3. Click **OK**.

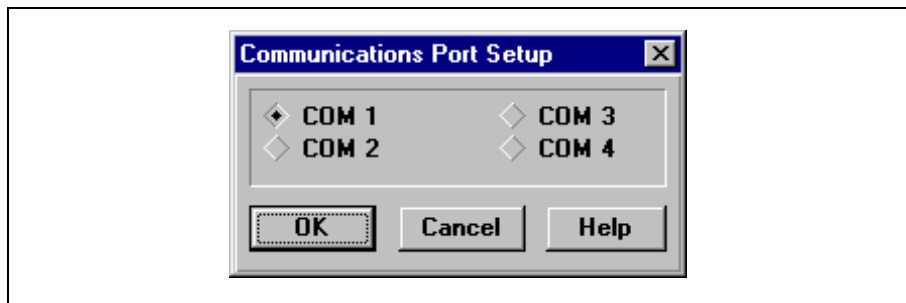


Figure 1.59: Communications Port Setup Dialog



# MPLAB IDE Toolbar and Menu Options

## 1.10 Tools Menu

### 1.10.1 DOS™ Command to Window (F11)

*Tools > DOS Command to Window* allows you to run a DOS program such as a compiler, or an internal command such as DIR, and capture the output into an edit window.

The command starts a dialog that prompts you for the command line to be executed. This can be any DOS command, including built-in ones like DIR.

By clicking the **Browse** button, you can specify the working directory that the command is to run in. This affects only the DOS command. MPLAB IDE will still use its previous working directory.

When you start the dialog, MPLAB IDE sets the command string and the proposed working directory name to the values you set the last time you used it.

You may execute only one DOS command at a time. Until the command completes, MPLAB IDE will reject attempts to start another.

**Note:** You cannot start a Windows application with this dialog.

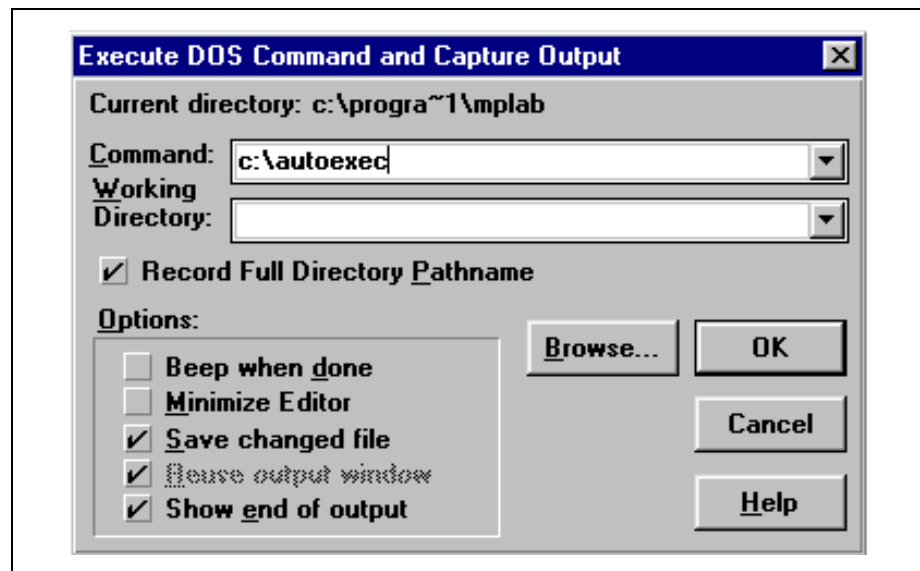


Figure 1.60: Execute DOS Command Dialog

1. Set the various options you want to apply:

Beep when done causes the Editor to sound the standard system beep when the DOS program finishes.

Minimize Editor causes the Editor to minimize into an icon before running the DOS program.

# MPLAB<sup>®</sup> IDE User's Guide

---

Save changed files causes the Editor to see if any of the files you're editing have changed and gives you the opportunity to save the changes before running the DOS program. If you agree to save the changes, all the files you've altered will be written to disk.

Reuse output window causes the Editor to place the DOS program's output into the window used the last time you used this dialog. If not, a new output window will be created.

Show end of output causes the Editor to automatically scroll the window showing the output to show you the end of it rather than the start.

2. Click **OK** to run the DOS program.

The command line, working directory path, and options are recorded and become the default the next time you use this dialog.

## 1.10.2 Repeat DOS Command to Window (Ctrl+F11)

Tools > Repeat DOS Command to Window exactly repeats the last DOS command you ran with Execute DOS Command To Window and shows the command output in a window when it completes.

If you have not previously run a command, the Editor will act as if you selected Execute DOS Command To Window and will open the Execute DOS Command and Capture Output dialog.

## 1.10.3 Verify PICMASTER Emulator

If you are using a PICMASTER emulator with MPLAB IDE, select Tools > Verify PICMASTER to verify that the PICMASTER emulator is operating properly. Refer to the *PICMASTER User's Guide* (DS51037) for detailed information on verifying the PICMASTER emulator.

## 1.10.4 Verify MPLAB ICE

If you are using an MPLAB ICE 2000 with MPLAB IDE, you must exit MPLAB IDE and run `VERIFY.EXE`, located in the main MPLAB IDE folder, to verify that the MPLAB ICE 2000 is operating properly. Refer to the *MPLAB ICE In-Circuit Emulator User's Guide* (DS51159) for detailed information.

# MPLAB IDE Toolbar and Menu Options

---

## 1.11 Window Menu

All Window options are available in simulator mode or emulator mode.

In Editor Only mode, Absolute Listing and Show Symbol List are available. In addition, the window positioning options and the Open Windows selections are also available.

Available windows are:

- Program Memory
- Trace Memory
- EEPROM Memory (device dependent)
- Calibration Data (device dependent)
- Absolute Listing
- Map File
- Stack
- File Registers
- Special Function Registers
- Show Symbol List
- Stopwatch
- Project
- Watch Window
- Modify

The following commands affect the arrangement and appearance of windows in MPLAB IDE.

- Tile Horizontal
- Tile Vertical
- Cascade
- Iconize All
- Arrange Icons
- (*Open Windows*)

# MPLAB® IDE User's Guide

## 1.11.1 Program Memory

Select *Window > Program Memory* to display program memory. The program memory window can display locations in the range of program memory for the currently selected processor. You can leave the Program Memory window open at all times and move and resize the window.

The Program Memory window is only available in Emulator and Simulator mode.

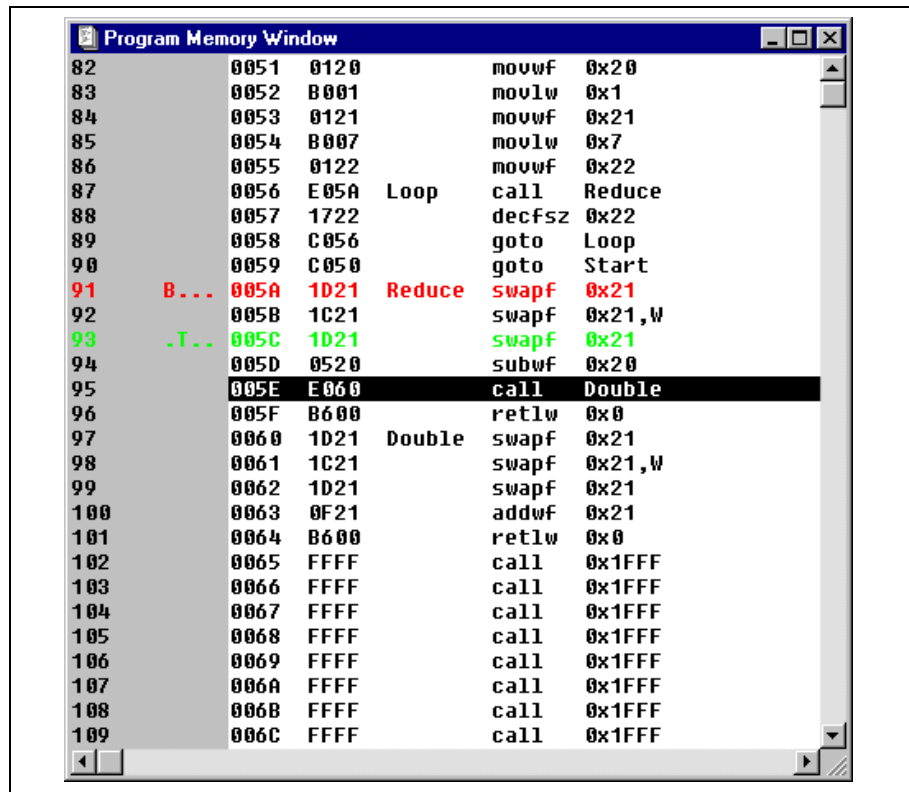


Figure 1.61: Program Memory Window – Machine Code Display

### 1.11.1.1 System Button Options

Click the system button in the upper left corner of the Program Memory screen to display the following options:

- Toggle Line Numbers* Toggles field for displaying line numbers and qualifier points.
- HEX Code Display* Displays program memory information as HEX data.

# MPLAB IDE Toolbar and Menu Options

*Machine Code Display* Displays disassembled HEX code with no symbolic information.

*Disassembly Display* Displays disassembled HEX code with symbols.

## 1.11.1.2 Program Memory Display Modes

The program memory can be displayed three ways. The desired format is chosen through the system menu.

- **HEX Code Display** – This displays the program memory as HEX data. This option is most useful when using a device programmer (Figure 1.62).
- **Machine Code Display** – This displays the disassembled HEX code with no symbolic information (Figure 1.61).
- **Disassembly Display** – This displays the disassembled HEX code with symbols.

When this window is in Machine Code Display mode or Disassembly Display mode, the instruction at the current program counter address will be highlighted. Other features of MPLAB IDE can alter the display of the program memory window.

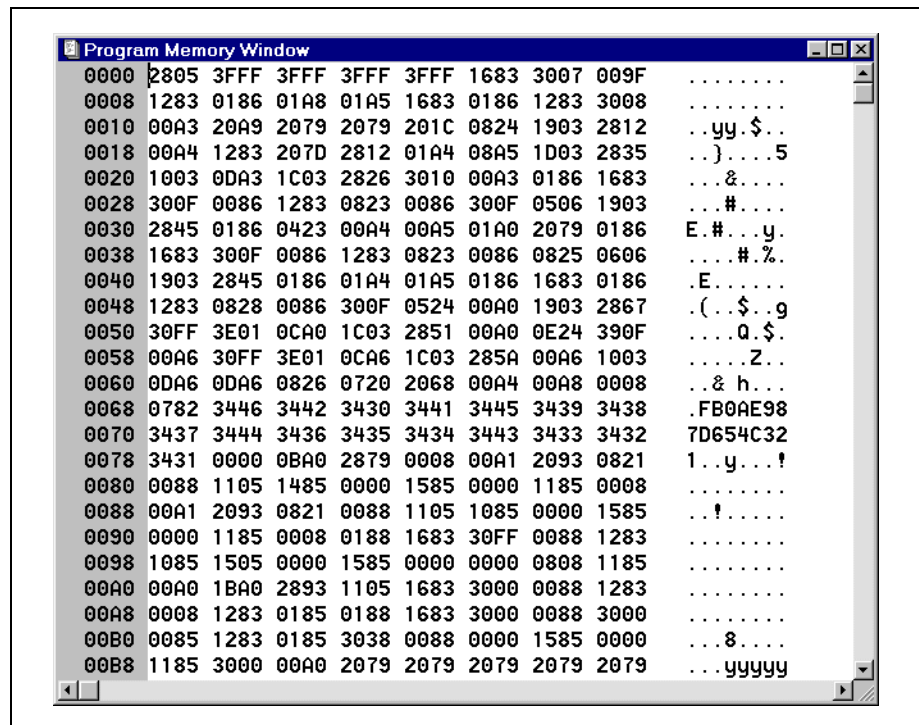


Figure 1.62: Program Memory Window – HEX Code Display

# MPLAB® IDE User's Guide

---

## 1.11.1.3 Program Memory Field Descriptions

MPLAB IDE displays data in the Program Memory window that it reads directly from emulation memory. Program Memory fields contain the following information:

*Field One* Address in HEX.

*Field Two* Opcode (or data) in HEX.

*Field Three* Program Label in symbolic format. You can increase the display width of labels by selecting Options > Environment Setup and clicking the **General** tab.

*Field Four* Machine code, disassembled code, or source code.

*Highlight Bar* Current location of the program counter.

Program Memory locations display break, trace, and trigger out status at each memory location as follows:

### 1.11.1.3.1. Selecting Points

Symbol	Point Type	Menu Selection (RMB = Right Mouse Button)
B	Break Points	RMB > Break Point(s)
T	Trace Points	RMB > Trace Point(s)
O	Trigger Output	RMB > Trigger Point(s)
Q	Pass Count Address	Set from Break or Trace Point Settings Dialog

Available options depend on the development tool in use.

MPLAB IDE uses a combination of color and notations in the Program Memory window to show break, trace, and trigger points. If no points have been set at a particular address, the text will display normally. If a point is set, the color of the text will change and the width of the line number window will increase to show the active points. Unset points are displayed as periods.

## 1.11.1.4 Creating a Temporary Real-Time Break Point

To set up a temporary real-time break point from the Program Memory Window, double-click the left mouse button anywhere on a valid address line. The processor runs in real time until one of the following occurs:

- The line containing the temporary break point is executed
- A break point is encountered
- You click on **Halt**

# MPLAB IDE Toolbar and Menu Options

## 1.11.2 Trace Memory

Select *Window > Trace Memory* to display the contents of the trace buffer. This window can be left open at all times, moved around, and resized.

The trace memory window takes a “snapshot” of your program’s execution. For emulators that have a trace buffer, this shows how your program runs at full speed.

Some applications, such as motor control systems, cannot be halted. Some bugs may appear only when the application is running; i.e., they don’t occur when single-stepping through the code. The trace buffer gives you another tool for testing such applications. Check your emulator user’s guide for more information on the information collected in its hardware trace buffer.

In the simulator, the trace buffer is useful for collecting a long record of program execution so you can analyze it later. The simulator will show slightly different information than the emulator’s trace.

To use the simulator’s trace buffer, first you must select code to trace. If you click and drag across the program memory window you can select instructions to trace, then press the right mouse button. This will bring up a shortcut menu where you can select “Trace Point(s).” Now reset and run the code, then halt it after it runs for a few seconds. Select *Window > Trace* to see the collected trace:

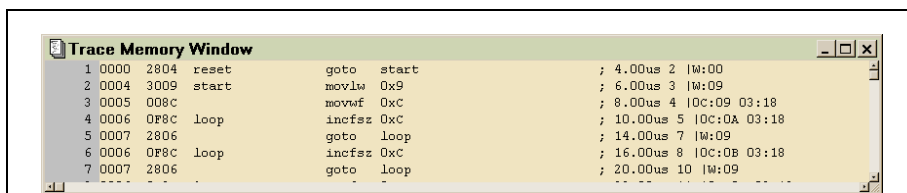


Figure 1.63: Trace Memory Window

The simulator puts a time stamp on each line and also shows any registers that were changed along with their values.

*Field One* Address in HEX.

*Field Two* Opcode (or data) in HEX.

*Field Three* Program Label in symbolic format. You can increase the display width of labels by selecting *Options > Environment Setup* and clicking the **General** tab.

*Field Four* Machine code, disassembled code, or source code.

*Field Five* Emulator: Status on External Logic Probe Lines. The status fields are shown on the extreme right-hand side of the window. Simulator: Time stamp and changed register information.

To save the contents of the trace buffer to a file, select *File > Export > Export Trace Buffer*.

## 1.11.3 EEPROM Memory

Select *Window > EEPROM Memory* to display the EEPROM data memory window for a microcontroller device that has EEPROM data memory. The PIC16F84 is an example of a device that supports EEPROM memory.

The EEPROM window can be left open at all times, moved around, and resized. This window is for information only and you cannot change values from this window. To change the EEPROM values, select *Window > Modify* and use the Modify dialog.

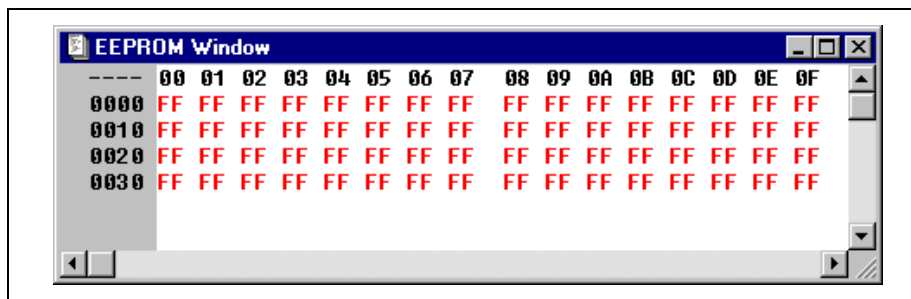


Figure 1.64: EEPROM Memory Window

The EEPROM window displays the data/opcode HEX information of the particular processor being emulated. When an EEPROM register value changes or the processor is halted, the data in the EEPROM window is updated.

### System Button Options

Click the system button in the upper left corner of the EEPROM Memory screen to display the following options:

*Toggle Line Numbers* Toggles field for displaying line numbers.

*HEX Display* Displays program memory information as HEX data.

*ASCII Display* Displays the ASCII character at each memory location.

## 1.11.4 Calibration Data

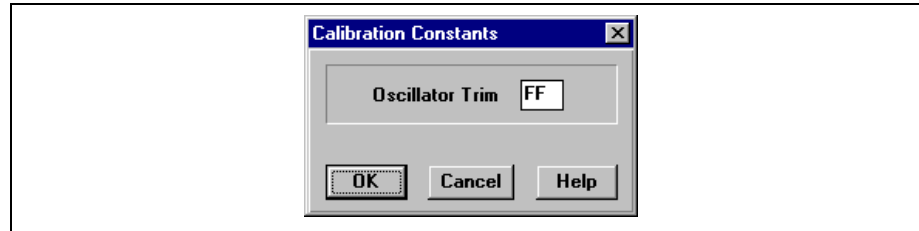
If the emulated device contains calibration memory, the calibration memory can be viewed by selecting *Window > Calibration Data*. The appearance of this window will depend on the emulated device.

The Calibration Data dialog is for use with the PIC12CXXX or PIC14000 device families. It displays the floating point data in the Emulator Probe for alteration by the user.

Saved calibration data may be loaded into MPLAB IDE using *File>Import*.



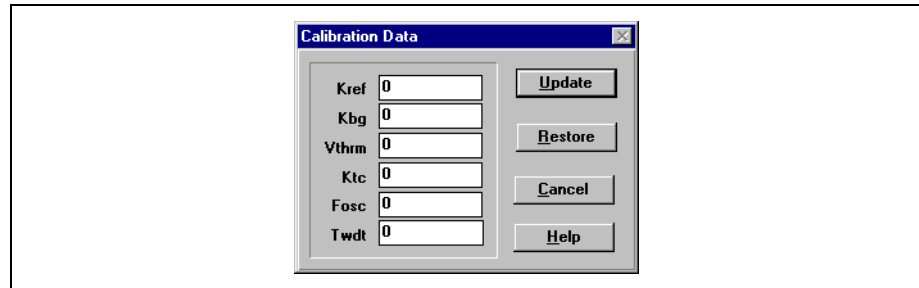
# MPLAB IDE Toolbar and Menu Options



**Figure 1.65: Calibration Constants Dialog – PIC12CXXX**

*OK* Closes the dialog and modifies the calibration value as specified.

*Cancel* Closes the dialog and does not modify the values in the memory area.



**Figure 1.66: Calibration Data Dialog – PIC14000**

*Update* Takes the information from calibration data dialog, converts it from the IEEE 754 format to the Microchip version of IEEE 754, and stores it in the memory area for use in the embedded code. Only the first four numbers KREF, KBG, VTHRM, and KTC are in floating point format. The last two, FOSC and TWDT, are 8-bit, unsigned numbers with values from 0 to 255.

*Restore* Takes the original calibration data uploaded from the probe during system Reset and resets the values in the memory area for use in the embedded code.

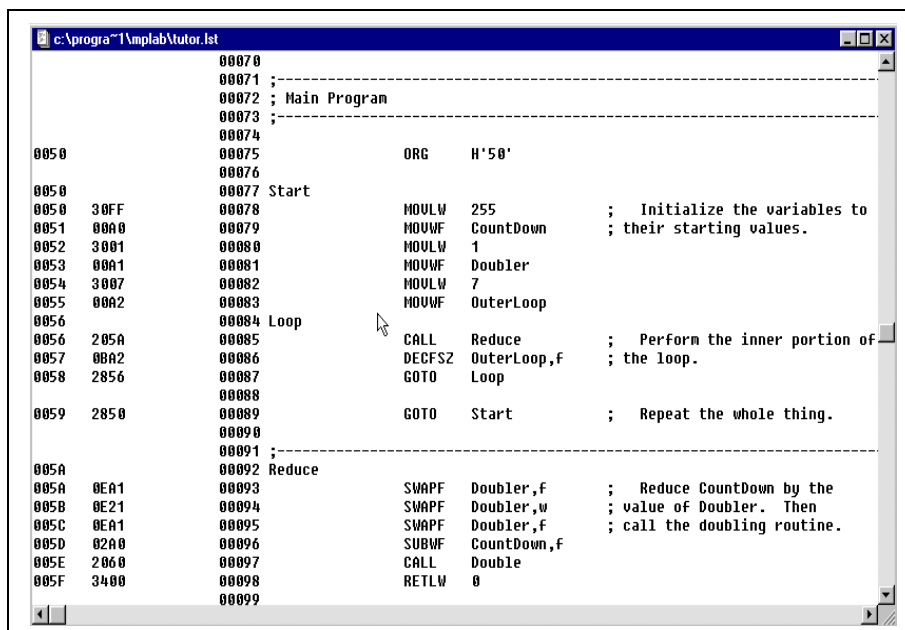
*Cancel* Closes the dialog and does not modify the values in the memory area.

## 1.11.5 Absolute Listing

Select *Window > Absolute Listing* to display and single step through the list (\*.LST) file generated by the MPASM assembler or a compatible C compiler.

Use this function to display both C code and the assembly code that corresponds to the C code. The Absolute Listing gives you a better idea how the C compiler implemented your code. If you are not using C, an assembler (e.g., MPASM) will generate a similar listing file.

The Absolute Listing Window shows the list file generated by the assembler or compiler. The Absolute Listing displays source code with the generated object code.



The screenshot shows a window titled "c:\progra~1\mplab\tutor.lst" containing assembly code. The code is organized into sections: a main program section, an initialization section, a loop section, and a reduce section. Each assembly instruction is followed by its object code (hexadecimal) and a comment. The assembly code includes instructions like ORG, MOVLW, MOVWF, CALL, DECFSZ, GOTO, SWAPF, SUBWF, CALL, and RETLW. Comments provide context for the code, such as "Initialize the variables to their starting values." and "Perform the inner portion of the loop."

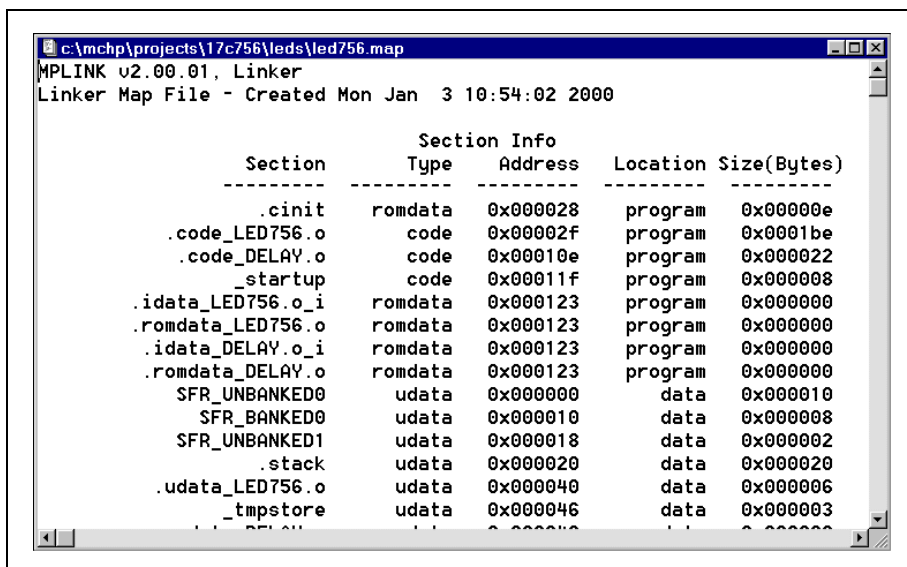
```
c:\progra~1\mplab\tutor.lst
00070
00071 ;-----
00072 ; Main Program
00073 ;-----
00074
0050          ORG      H'50'
00075
00076
0050          00077 Start
0050 30FF      00078          MOVLW 255      ; Initialize the variables to
0051 00A0      00079          MOVWF  Countdown ; their starting values.
0052 3001      00080          MOVLW 1
0053 00A1      00081          MOVWF  Doubler
0054 3007      00082          MOVLW 7
0055 00A2      00083          MOVWF  OuterLoop
0056          00084 Loop
0056 205A      00085          CALL  Reduce      ; Perform the inner portion of
0057 00A2      00086          DECFSZ OuterLoop,f ; the loop.
0058 2856      00087          GOTO   Loop
0059          00088
0059 2850      00089          GOTO   Start      ; Repeat the whole thing.
0059          00090
0059          00091 ;-----
005A          00092 Reduce
005A 0EA1      00093          SWAPF  Doubler,f      ; Reduce Countdown by the
005B 0E21      00094          SWAPF  Doubler,w      ; value of Doubler. Then
005C 0EA1      00095          SWAPF  Doubler,f      ; call the doubling routine.
005D 02A0      00096          SUBWF  Countdown,f
005E 2060      00097          CALL  Double
005F 3400      00098          RETLW 0
00099
```

Figure 1.67: Absolute Listing Window

# MPLAB IDE Toolbar and Menu Options

## 1.11.6 Map File

A map file shows the used and unused memory regions after linking. To view the map file, select *Window > Map File*.



The screenshot shows a window titled "c:\mchp\projects\17c756\leds\led756.map" with the following content:

```
MPLINK v2.00.01, Linker
Linker Map File - Created Mon Jan 3 10:54:02 2000
```

Section	Type	Address	Location	Size(Bytes)
.cinit	romdata	0x000028	program	0x00000e
.code_LED756.o	code	0x00002f	program	0x0001be
.code_DELAY.o	code	0x00010e	program	0x000022
_startup	code	0x00011f	program	0x000008
.idata_LED756.o_i	romdata	0x000123	program	0x000000
.romdata_LED756.o	romdata	0x000123	program	0x000000
.idata_DELAY.o_i	romdata	0x000123	program	0x000000
.romdata_DELAY.o	romdata	0x000123	program	0x000000
SFR_UNBANKED0	udata	0x000000	data	0x000010
SFR_BANKED0	udata	0x000010	data	0x000008
SFR_UNBANKED1	udata	0x000018	data	0x000002
.stack	udata	0x000020	data	0x000020
.udata_LED756.o	udata	0x000040	data	0x000006
_tmpstore	udata	0x000046	data	0x000003

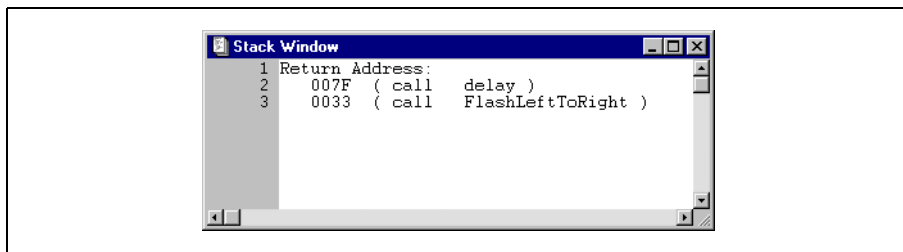
Figure 1.68: Map File Window

To generate a map file for a linked project, select *Project > Edit Project* to bring up the Edit Project dialog. Then select/highlight the main node [.hex] in the Project Files section of the dialog. Click **Node Properties** to bring up the Node Properties dialog. Select the Map file checkbox to turn on the creation of the map file.

For more information on map files, see the *MPASM User's Guide with MPLINK and MPLIB* (DS33014).

## 1.11.7 Stack

Select *Window > Stack* to open a window displaying the contents of the stack. The number of available levels depends on the processor type being emulated. The Stack Window can be left open at all times, moved around, and resized.



**Figure 1.69: Stack Window**

**Note:** If Stack Overflow Break Enable is set (*Options > Development Mode > Break Options*), MPLAB IDE will display stack overflow and underflow warnings when they occur.

The contents of the stack may be displayed with (shown) or without line numbers. The desired format is chosen through the system menu.

**Note:** The system menu is accessed by clicking the top left corner of the program memory window.

### 1.11.7.1 Hardware Stack Levels

#### 1.11.7.1.1. 12-Bit Core Hardware Stack – 2-Levels Deep

Devices with a 12-bit core, such as PIC12C5XX and PIC16C5X, have a 2-level deep hardware stack.

#### 1.11.7.1.2. 14-Bit Core Hardware Stack – 8-Levels Deep

Devices with a 14-bit core, such as PIC14000 and PIC16CXXX, have an 8-level deep hardware stack.

**Note:** If you clear Disable Stack Overflow Warning (*Options > Development Mode, Break Options* tab) and push the stack beyond its limit, MPLAB IDE will display an underflow or overflow message.

# MPLAB IDE Toolbar and Menu Options

---

## 1.11.7.1.3. 16-Bit Core Hardware Stack – 16-Levels Deep

Devices with a 16-bit core, such as PIC17CXXX, have a 16-level deep hardware stack.

**Note:** If you clear Disable Stack Overflow Warning (*Options > Development Mode*, Break Options tab) and push the stack beyond its limit, MPLAB IDE will display an underflow or overflow message.

## 1.11.7.1.4. 16-Bit Enhanced Hardware Stack – 31-Levels Deep

Devices with an enhanced 16-bit core, such as PIC18CXXX, have a 31-level deep hardware stack.

**Note:** If you clear Disable Stack Overflow Warning (*Options > Development Mode*, Break Options tab) and push the stack beyond its limit, MPLAB IDE will display an underflow or overflow message.

## 1.11.7.2 Simulator Stack—12-Bit Core Devices

The MPLAB SIM simulator presents an accurate simulation of the hardware stack on the PIC12C5XX and PIC16C5X devices and additionally provides warning messages if an underflow or overflow condition occurs. When a CALL instruction is encountered, or when an interrupt has occurred, the value of the PC + 1 is pushed to the stack, and the stack is popped when a RETLW instruction is executed. If more than two values are pushed to the stack before it is popped, the value will be pushed to the stack, but a warning message will be issued indicating a stack overflow condition. An error message will also be generated if the user attempts to pop an empty stack. Popping an empty stack will cause the last value popped to be put in the PC.

## 1.11.7.3 Simulator Stack – 14-Bit Core Devices

The MPLAB SIM simulator presents an accurate simulation of the hardware stack on the PIC14000 and PIC16CXX devices, and additionally provides warning messages if an underflow or overflow condition occurs. When a CALL instruction is encountered, or when an interrupt has occurred, the value of the PC + 1 is pushed to the stack, and the stack is popped when a RETLW, RETURN, or RETFIE instruction is executed. If more than eight values are pushed to the stack before it is popped, the value will be pushed to the stack, but a warning message will be issued indicating a stack overflow condition. An error message will also be generated if the user attempts to pop an empty stack. Popping an empty stack will cause the stack pointer to point to the top of a full stack, and will not generate an error message if another pop is initiated.

## 1.11.7.4 Simulator Stack – 16-Bit Core Devices

The MPLAB SIM simulator presents an accurate simulation of the hardware stack on the PIC17CXXX, and additionally provides warning messages if an underflow or overflow condition occurs. When a CALL or LCALL instruction is encountered or when an interrupt has occurred, the value of the PC + 1 is pushed to the stack. The stack is popped when a RETLW, RETURN, or RETFIE instruction is executed. If more than sixteen values are pushed to the stack before it is popped, the value will be pushed to the stack, a warning message will be issued indicating a stack overflow condition, and the STAKAVL bit will be cleared until a RESET condition occurs.

## 1.11.7.5 Simulator Stack – Enhanced 16-Bit Core Devices

The MPLAB SIM simulator presents an accurate simulation of the hardware stack on the PIC18CXXX, and additionally provides warning messages if an underflow or overflow condition occurs. When a CALL or LCALL instruction is encountered or when an interrupt has occurred, the address of the next instruction following the call is pushed to the stack. The stack is popped when a RETLW, RETURN, or RETFIE instruction is executed. If more than 31 values are pushed to the stack before it is popped, the value will be pushed to the stack, a warning message will be issued indicating a stack overflow condition, and the STAKAVL bit will be cleared until a RESET condition occurs.

# MPLAB IDE Toolbar and Menu Options

## 1.11.8 File Registers

Select *Window > File Registers* to display a window of all the File Registers of the particular processor being emulated. When a file register value changes, or the processor is interrogated, the data in the File Register window is updated. The File Register window can be left open at all times, moved around, and resized.

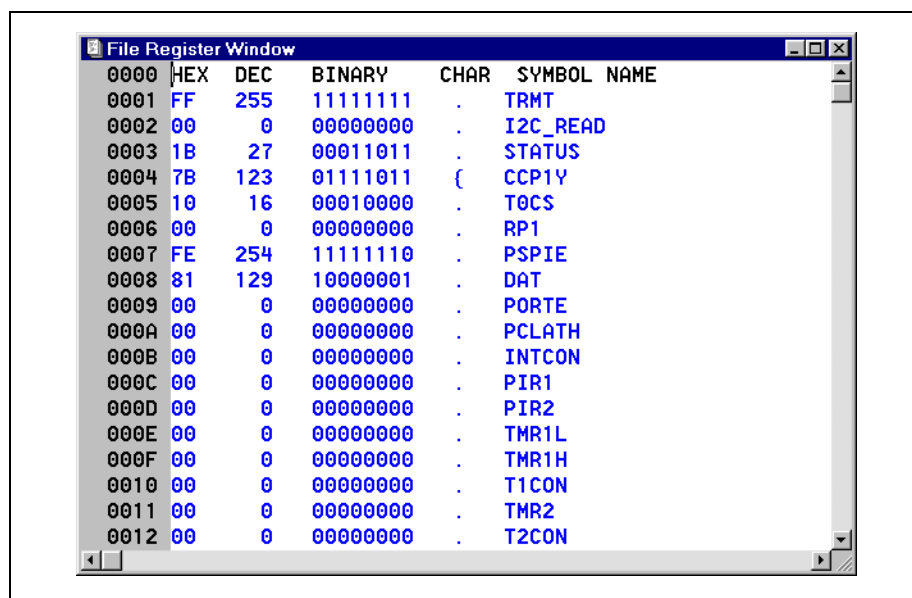


Figure 1.70: File Registers Window – Symbolic Display

File register contents can also be modified through this window. To change a range of registers (i.e., to fill them with a constant value), click the left mouse button and drag over the values you want to change. To change one register, simply place the cursor over the register you want to change. Then click the right mouse button to display the Fill Register pop-up menu. Select **Fill Register** to display the Modify dialog (Figure 1.80) with the address range already entered.

## 1.11.8.1 File Registers Window Display Modes

File registers can be displayed three ways. The desired format is chosen through the system menu.

- **HEX Display** – This displays the file registers as HEX data (Figure 1.71).
- **Symbolic Display** – This displays each file register symbolically with corresponding data in HEX, decimal, binary and character formats (Figure 1.70).
- **ASCII Display** – This displays the file registers as ASCII data.

**Note 1:** The system menu is accessed by clicking the top left corner of the File Register window.

**2:** If Freeze Peripherals On Halt is selected in the Break Options tab under *Options > Development Mode*, 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.

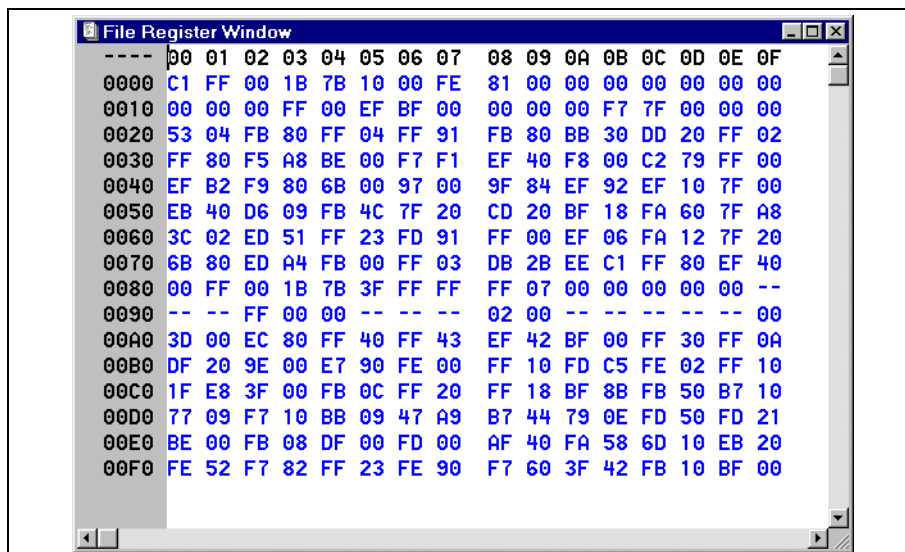


Figure 1.71: File Register Window – HEX Display



# MPLAB IDE Toolbar and Menu Options

---

## 1.11.8.2 System Button Options

Click the system button in the upper left corner of the File Register Memory screen to display the following options:

*Toggle Line Numbers* Toggles field for displaying line numbers.

*HEX Display* Displays file registers information as HEX data.

*Symbolic Display* Displays data at each memory location in the following formats: HEX, Decimal, Binary, ASCII Character, Symbol, and Name.

*ASCII Display* Displays the ASCII character at each memory location.

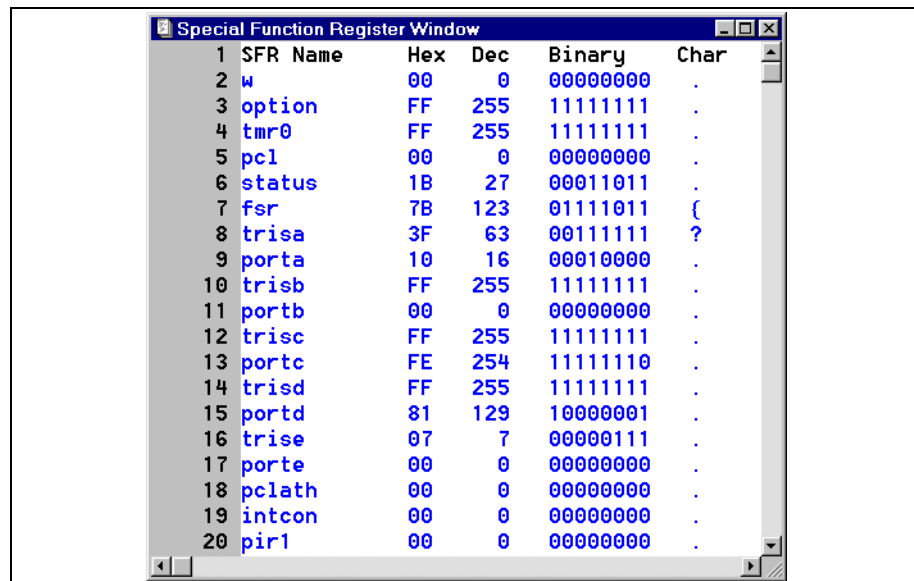
## 1.11.9 Special Function Registers

Select *Window > Special Function Registers* to display the contents of the Special Function Registers (SFR) for the processor being emulated. 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 SFRs are updated.

This window can be left open at all times, moved around and resized. A sample of this window is shown below.

SFRs may be displayed with (shown) or without line numbers. The desired format is chosen through the system menu.

- Note 1:** The system menu is accessed by clicking the top left corner of the program memory window.
- 2:** If Freeze Peripherals On Halt is selected in the Break Options tab under *Options > Development Mode*, 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.



SFR Name	Hex	Dec	Binary	Char
w	00	0	00000000	.
option	FF	255	11111111	.
tmr0	FF	255	11111111	.
pcl	00	0	00000000	.
status	1B	27	00011011	.
fsr	7B	123	01111011	{
trisa	3F	63	00111111	?
porta	10	16	00010000	.
trisb	FF	255	11111111	.
portb	00	0	00000000	.
trisc	FF	255	11111111	.
portc	FE	254	11111110	.
trisd	FF	255	11111111	.
portd	81	129	10000001	.
trise	07	7	00000111	.
porte	00	0	00000000	.
pclath	00	0	00000000	.
intcon	00	0	00000000	.
pir1	00	0	00000000	.

Figure 1.72: Special Function Registers

# MPLAB IDE Toolbar and Menu Options

---

---

*Field One* Name of SFR.

*Field Two* Data as a hexadecimal value.

*Field Three* Data as a decimal value.

*Field Four* Data as a binary number.

*Field Five* Data as ASCII characters.

To modify the contents of a particular SFR:

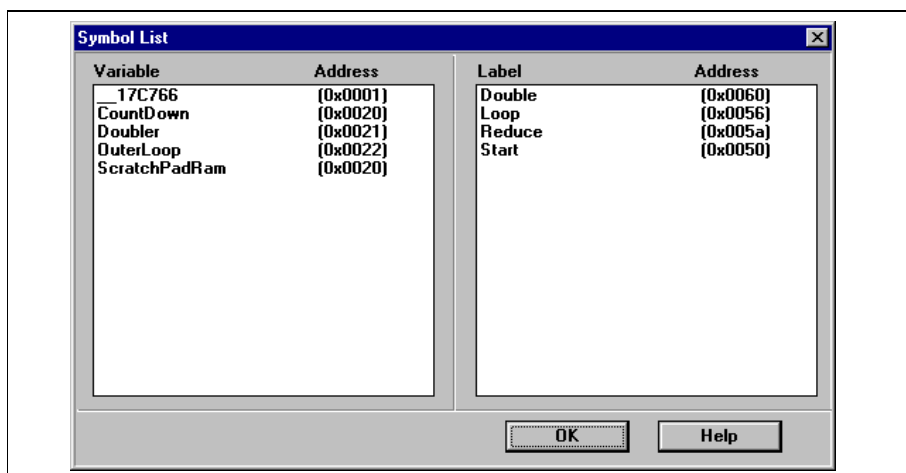
1. Double-click on a register in this window to invoke the Modify dialog box with the symbol/address and data fields already filled in, or
2. Use the execute *Window > Modify* menu.

**Note:** The SFR names and addresses are different for every device.

## 1.11.10 Show Symbol List (Ctrl+F8)

Select *Window > Show Symbol List* to display all symbols known to MPLAB IDE. Symbols include constants and labels. Show Symbol List is an information only dialog box. The symbols displayed in this dialog box represent the symbols imported from your source code after compiling or assembling. These symbols are from the \*.COD file in your project.

**Note:** A project must be open and built in order for you to display a symbol list.



**Figure 1.73: Show Symbol List**

*Variable, Address* Displays variables from the File Register memory and the address of each variable.

*Label, Address* Displays labels from program memory and the address of each label.

*Constants* Constants defined in the source code can be used in executing opcode and as operands for instructions in Modify.

# MPLAB IDE Toolbar and Menu Options

## 1.11.11 Simulator Stopwatch

Select *Window > Stopwatch* to display the current value of the Cycle counter. The system Stopwatch counts the number of clock cycles that the processor executes. The counting occurs with real-time execution and with polled execution. The timer triggers on every cycle of an instruction. The stopwatch allows you to measure code execution time. It is not always accurate while single stepping. The stopwatch calculates time based upon the clock frequency of the PICmicro MCU device. To set the clock frequency, select *Options > Development Mode* and click the **Clock** tab.

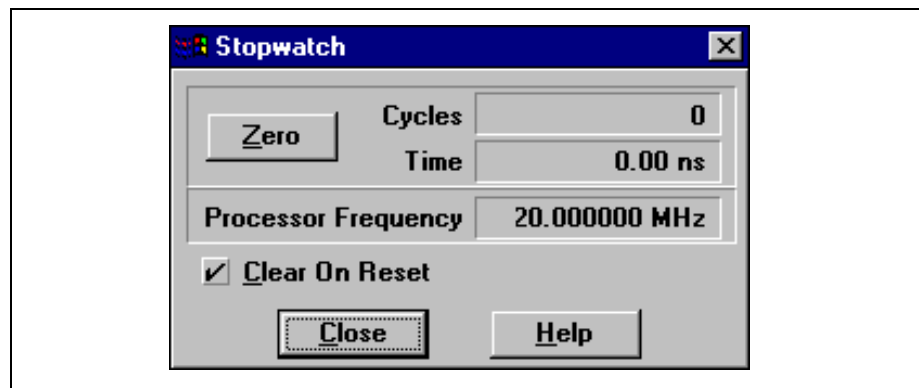


Figure 1.74: Simulator Stopwatch Dialog Box

- Cycles* Displays the number of cycles that the processor executes.
- Time* Displays the stopwatch time in seconds. Computed from the number of cycles executed and from the processor frequency.
- Zero* Click **Zero** to reset the cycle counter to zero. You can reset the timer at any time when the processor is halted.

**Example 1.1:** You can use this timer for precise timing measurements. If you need to measure the exact time a subroutine takes to execute, then simply reset the timer before entering the subroutine and put a break point at the end of the subroutine. The timer displays the total number of cycles executed in the subroutine and also displays the execution time.

*Processor Frequency* Displays the selected Processor Frequency. To change the frequency, you must select *Options > Development Mode* and click the **Clock** tab.

**Note:** The stopwatch is not available with MPLAB ICD or MPLAB ICE 2000. For MPLAB ICE 2000, refer to the *MPLAB ICE Emulators User's Guide* (DS51159) for an example of a "Time Between Events" Complex Trigger.

# MPLAB<sup>®</sup> IDE User's Guide

---

## 1.11.12 Project Window

The Project Window is available only when a project is open. It displays the list of files currently in the project. If the project has been compiled, the project window displays a list of all included files in the project. Otherwise, the Project Window only displays the main project file. Double-click on any file displayed in the Project Window to open that file for editing.

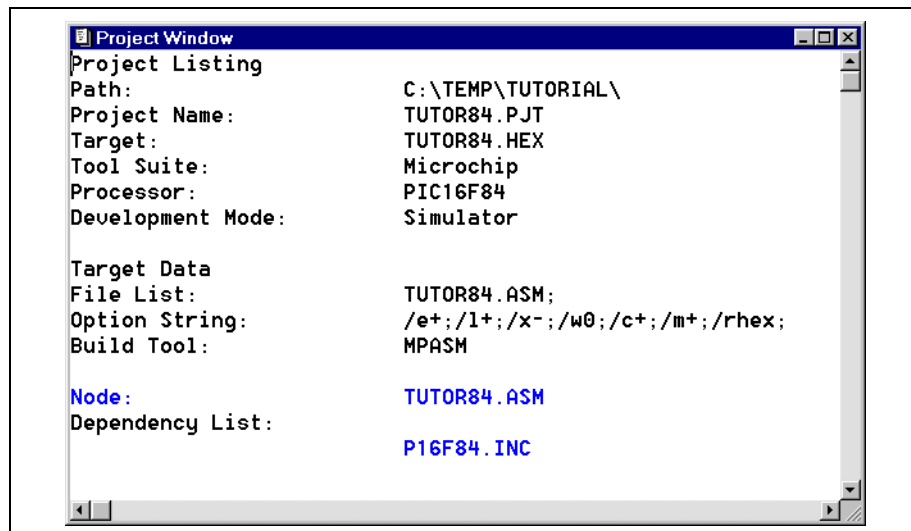


Figure 1.75: Project Window

# MPLAB IDE Toolbar and Menu Options

## 1.11.13 Watch Window

MPLAB IDE allows the contents of file registers to be monitored through a Watch window. Using either the MPLAB IDE menu or a system menu inside the Watch window, you can add and remove symbols and change their display properties. The contents of the Watch window may be displayed with or without line numbers. To choose the desired format, select *Toggle Line Numbers* on the system menu inside the Watch window.

### 1.11.13.1 New Watch Window

To create a Watch window, select *Window > Watch Window > New Watch Window*. Both the Add Watch Symbol dialog (Figure 1.77) and the Watch\_1 window (Figure 1.76) will open. Refer to Section 1.11.13.4 for information on editing Watch windows.

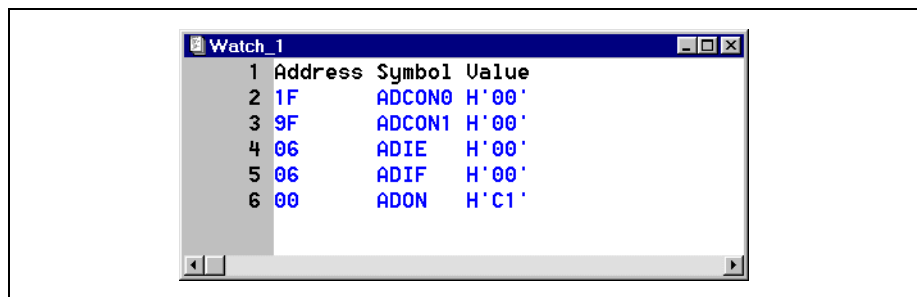


Figure 1.76: Watch Window

### 1.11.13.2 Load Watch Window

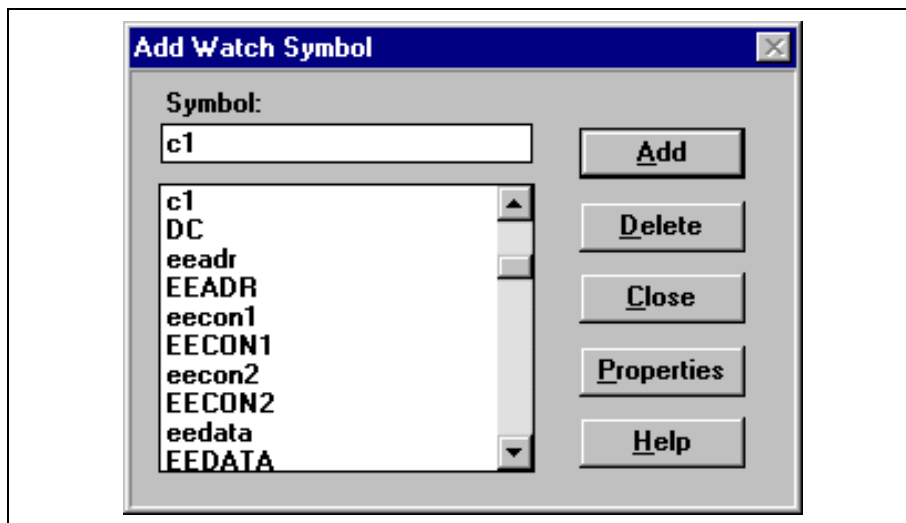
Select *Window > Watch Window > Load Watch Window* to load a Watch window that you previously created and saved to disk. Select a Watch window file to load and click **OK**, or double-click the desired file.

In the Load Watch dialog, select the Drive from the Drives pull down list at the bottom right of the dialog. Double-click the folder in the Folders hierarchy at the top right of the Load Watch dialog to specify the path on the selected drive. In the list of files below the File Name box at the left of the Load Watch dialog, select the Watch window file you wish to open. Click **OK**. Both the Add Watch Symbol dialog and the Watch window will open.

Select *Add Watch Symbol* from the system menu to add symbols to the Watch window. Refer to Section 1.11.13.4 for information on editing Watch windows.

## 1.11.13.3 Add Watch Window Symbols

You can add symbols to the active Watch window.



**Figure 1.77: Add Watch Symbol Dialog**

Select *Window > Watch Window > Add to Active Watch*. Select symbols (see Section 1.11.13.7) to add to the open Watch window by clicking on them in the Add Watch Symbol list box. Or, you can enter the address of the symbol you want to watch (e.g., 0x40). Click **Add** to add them to the Watch window.

To change how a symbol is displayed in the Watch window, select that symbol and click **Properties** to open the Properties dialog (see Section 1.11.13.5).

Click **Close** when you have finished adding symbols. The Watch window will display the current value of the variables you selected.



# MPLAB IDE Toolbar and Menu Options

## 1.11.13.4 Edit Watch Window Symbols

You can use either the MPLAB IDE menu or the system menu inside the Watch window to change the display properties of symbols that are already in the Watch window, or delete the symbols from the Watch window.

After creating or opening a Watch window, select *Window > Watch Window > Edit Active Watch* from the MPLAB IDE menu to display the Edit Watch Symbol dialog. Or, select *Edit Watch* from the system menu inside the Watch window. The dialog lists all symbols you have added in the Watch window.

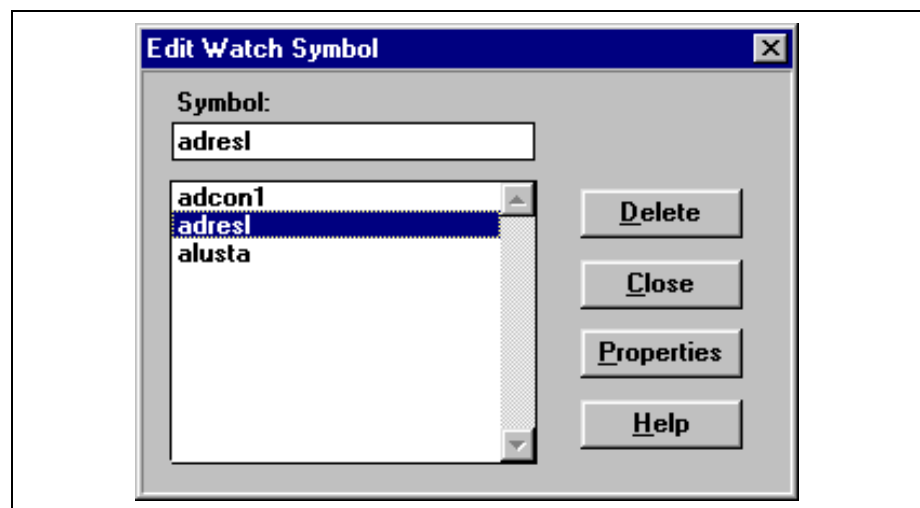


Figure 1.78: Edit Watch Symbol Dialog

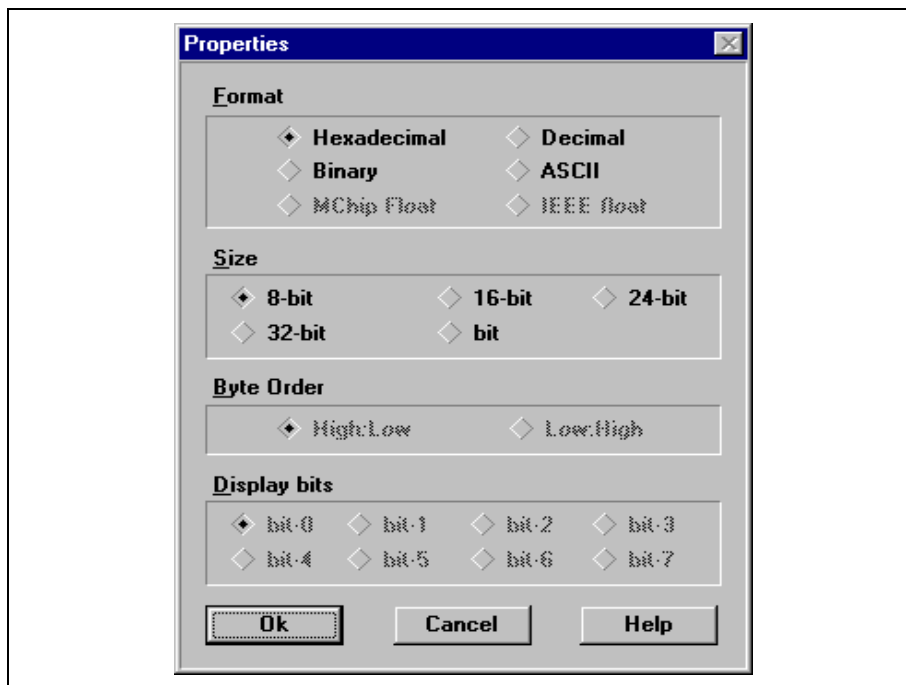
In the Symbol list of the Edit Watch Symbol dialog, scroll to the symbol of the variable you wish to change and highlight it. Click **Properties**.

To delete a symbol from the Watch window in the Edit Watch Symbol dialog, scroll to the symbol of the variable you wish to delete and click **Delete**. Or, simply place the cursor on the symbol in the Watch window and select *Delete > Watch* from the system menu inside the Watch window.

Once you are through editing or deleting watch symbols, click **Close** to close the Edit Watch dialog and return to the Watch window.

## 1.11.13.5 Changing Watch Symbol Properties

The Properties dialog allows you to select the format in which the symbols will be displayed in the Watch window. You may display the Properties dialog box by clicking **Properties** from the Add Watch Symbol and Edit Watch Symbol dialogs.



**Figure 1.79: Properties Dialog**

- Format* Determines what type of number to display: Hexadecimal, Binary, Mchip Float, Decimal, ASCII, or IEEE Float. Mchip Float and IEEE Float are available only for 24- and 32-bit size selections.
- Size* Determines how many bits will be included in the display of the number.
- Byte Order* Determines the display order of each byte. Available for 16-, 24-, and 32-bit numbers.
- Display Bits* To display a specific bit, select Bit in the Size area, then select the bit to be displayed here.
- OK* Click **OK** to accept the changes and return to the Edit Watch Symbol dialog.
- Cancel* Click **Cancel** to return to the Edit Watch Symbol dialog without accepting any changes.

# MPLAB IDE Toolbar and Menu Options

---

## 1.11.13.6 Save Watch Window

Select *Window > Watch Window > Save Watch Window* to save the Watch window that you are active in.

In the Save Watch dialog, select the Drive from the Drives pull-down list at the bottom right of the dialog. Double-click the folder in the Folders hierarchy at the top right of the Save Watch dialog to specify the path on the selected drive. In the File Name box at the top left of the Save Watch dialog, replace the "\*" with the name you want to give your Watch window. Click **OK** to save the Watch window to disk.

## 1.11.13.7 Watch Window Symbols Defined

Watch window symbols are represented differently depending on their application.

Symbol	Example	Definition
lower case	portb	Always available; internal to MPLAB IDE
upper case	PORTB	Loaded from header file

**Note:** The variable  $w$  represents an address of '0', where as the variable  $w$  represents the  $w$  register value.

## 1.11.14 Modify

Select *Window > Modify* to display and/or modify the contents of Data Memory, Program Memory, the Stack, or EEPROM memory. You can use this feature to write input pin states.

Modify allows you to Read/Write to a specific address, Read/Write while incrementing to the next address, or fill an address block. MPLAB IDE allows you to leave the Modify window open at all times and move it around.

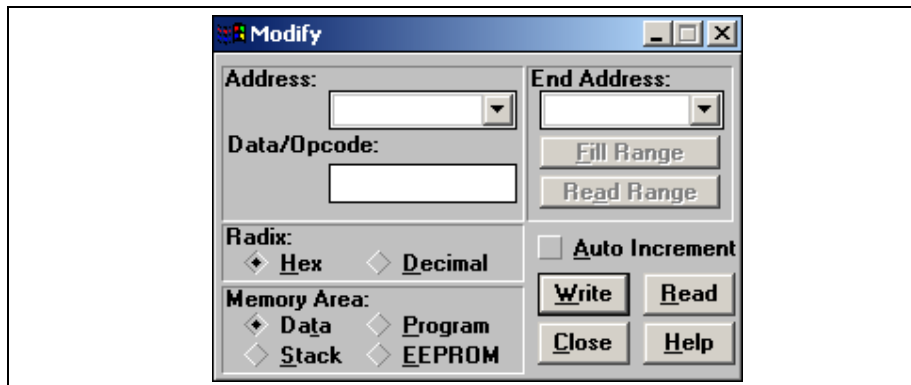


Figure 1.80: Modify Dialog Box

MPLAB IDE provides four ways to open the Modify dialog box:

- Select *Window > Modify*.
- Double-click on an item in the Special Functions Register window.
- Double-click an item in a Watch Window.
- Select an address or range in the File Register Window and click the right mouse button to display a popup menu which contains the Fill Register(s) option. Select *Fill Register(s)* from the popup menu to display the Modify dialog box.

*Address* Select a label from the drop-down list or enter the address at which data is to be read or modified. You can enter a numeric address or a symbol (label).

**Note:** Numeric constants must begin with a numeric digit. Therefore, HEX values may need to be preceded by "0"; e.g. "0FF" instead of "FF."

*Data/Opcode* Click **Read** to display data value/opcode at a selected address and memory area. Click **Write** to write data value/Opcode to the selected address and memory area.

*Radix* HEX or Decimal

# MPLAB IDE Toolbar and Menu Options

---

*Memory Area* Select the Memory Area that you want to modify:  
Data Memory:RAM Memory  
Program Memory:ROM Memory in the emulator  
Stack:Stack Memory on the Device  
EEPROM:EE Data Memory

*End Address* The ending address for Fill Range.

*Fill Range* Fills the range defined by the two addresses with the value entered in Data/Opcode.

*Read Range* Reads the memory range defined by the two addresses. This is used to read off-chip memory when using devices that support extended memory, such as the PIC17CXXX.

*Auto Increment* Select Auto Increment to increment to the next address after each Read/Write.

**Note:** Auto increment increments to the next address, displays the next address, and reads the contents at the address. If you are using Auto Increment to read a range, enter the address of the memory area minus one because the first read will increment the address.

*Write* Enter new data in the Data/Opcode field, and click **Write** to modify the data at the specified address. (You can enter data in symbolic format.) When data is modified, all the appropriate windows are updated with the new information.

*Read* Click **Read** to read the data at a specified address.

*Close* Click **Close** to exit the Modify dialog.

**Caution:** The Program Counter should not be changed through the Modify dialog. Instead, use the Change Program Counter dialog.

## 1.11.15 Tile Horizontal

The *Window > Tile Horizontal* command sizes open windows in a horizontal format making each window as wide as possible to allow you to see as much of each line in as many windows as possible. The command arranges all open windows in a tile pattern, placing the windows above one another. Excess windows are tiled in a horizontal pattern in the lower part of the screen.

Windows containing the output from commands run by the *Tools > DOS Command To Window* command are arranged preferentially at the top of the screen.

## 1.11.16 Tile Vertical

Window > Tile Vertical command sizes open windows vertically in columns to allow you to see as many lines as possible in each window.

The Tile Vertical command arranges all open windows in a tile pattern, placing them side by side so that each window is as deep as possible.

## 1.11.17 Cascade

The Window > Cascade command arranges all open windows in a cascade pattern.

## 1.11.18 Iconize All

Window > Iconize All makes all windows into icons.

## 1.11.19 Arrange Icons

Window > Arrange Icons arranges all iconized windows so that their icons are visible in rows at the bottom of the desktop. Open windows are not affected by this command.

# MPLAB IDE Toolbar and Menu Options

## 1.11.20 Open Windows

Open windows are listed at the end of the Window commands.

Whenever you open a window, MPLAB IDE records the name in the list, ordering it so that the windows you have used most recently always appear at the top. Click on a window name to jump to that window.

*More Windows* When the Windows list contains more files than can be displayed on the menu, More Windows is automatically added to the Windows menu. The command opens a dialog that displays the entire list of windows. To open a window from the list, either double-click the window name in the list, or select the name and press the **Open** button. The More Windows dialog box also provides additional options. The settings of the additional options are remembered and become the default for the next time you use this dialog.

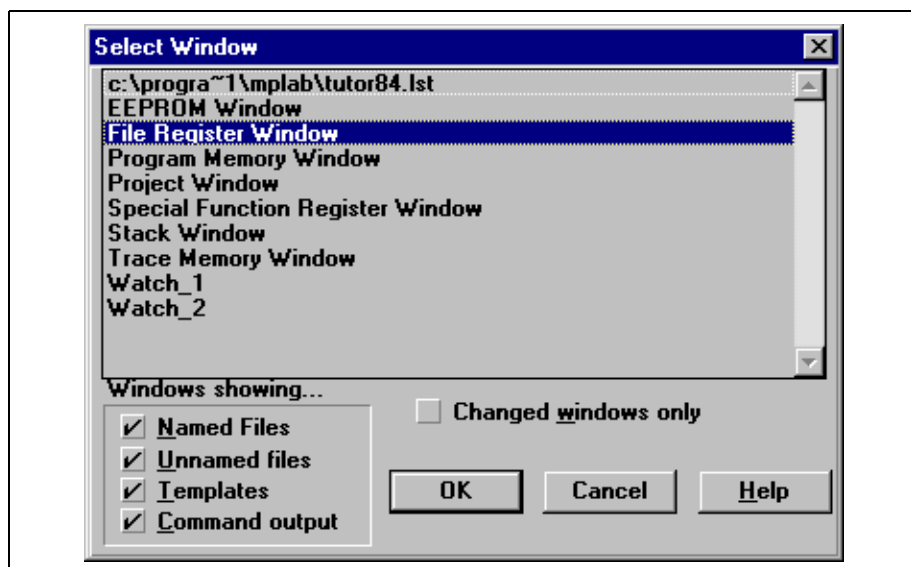


Figure 1.81: More Windows Dialog

*Changed Windows Only* Displays only windows that have been changed since opening.

*Named Files* Lists all windows having an associated file name.

*Unnamed Files* Lists all windows that do not have an associated file name.

*Templates* Lists all template windows.

*Command Output* Lists all windows containing command output obtained by selecting *Tools > DOS Command to Window*.

## 1.12 Help Menu

### 1.12.1 Release Notes (Shift+F1)

*Help > Release Notes* opens and displays the recent change history of MPLAB IDE software. The Release Notes are contained in the file `README.LAB`.

### 1.12.2 Tool Release Notes

If you are using additional tools, there may be an item on the Help menu on release notes for those tools.

### 1.12.3 MPLAB IDE Help

*Help > MPLAB IDE Help* contains help on using MPLAB IDE.

### 1.12.4 Editor Help

*Help > MPLAB Editor Help* contains help on using the MPLAB Editor.

### 1.12.5 Error Help

*Help > MPLAB Error Help* contains help on selected MPLAB IDE error messages.

### 1.12.6 MPASM Assembler Help

*Help > MPASM Assembler Help* opens an on-line version of the *MPASM User's Guide with MPLINK and MPLIB* (DS33014) with MPASM assembler-specific information. Click on the green, underlined items to get more information.

MPASM Assembler Help also contains a Quick Reference Guide that includes assembly directives and device-specific instruction sets.

### 1.12.7 MPLINK Linker Help

*Help > MPLINK Linker Help* opens an on-line version of the *MPASM User's Guide with MPLINK and MPLIB* (DS33014) with MPLINK linker- and MPLIB librarian-specific information. Click on the green, underlined items to get more information.

### 1.12.8 Tool Help

If you are using additional tools, there may be an item on the Help menu on help for those tools.



# MPLAB IDE Toolbar and Menu Options

## 1.12.9 About

*Help > About* displays:

- MPLAB IDE Version
- Microchip's Address
- Processor Version
- Disassembler Version
- Information on other registered applications

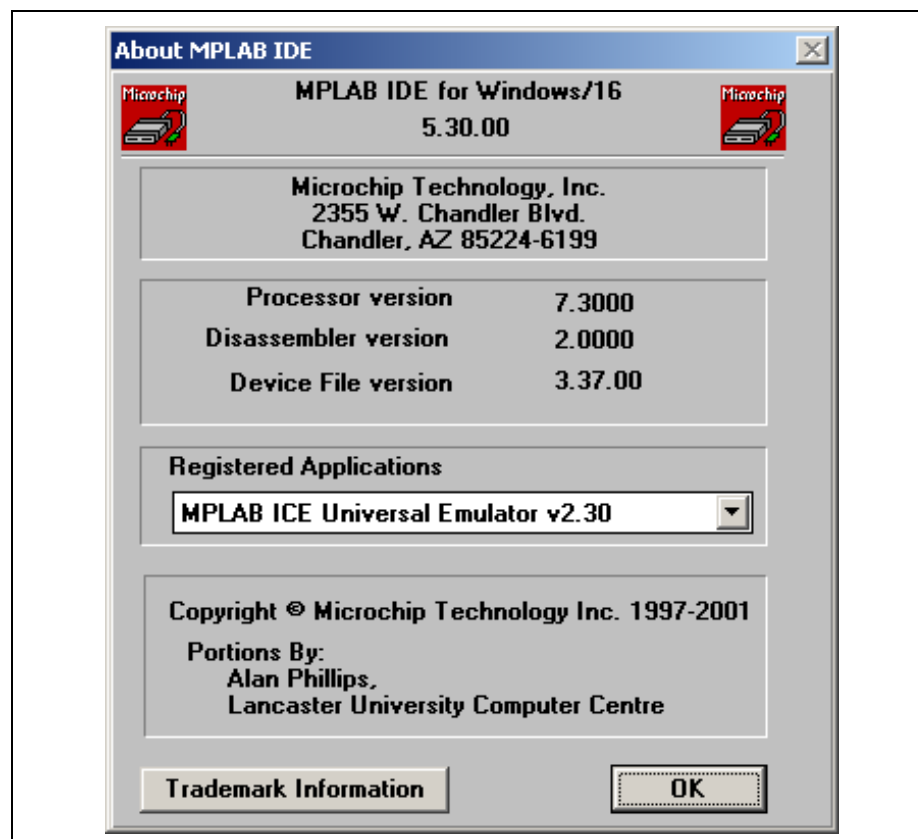


Figure 1.82: About Help Dialog

# MPLAB<sup>®</sup> IDE User's Guide

---

NOTES:

---

---

**Chapter 2. MPLAB IDE Toolbar and Status Bar Definitions**

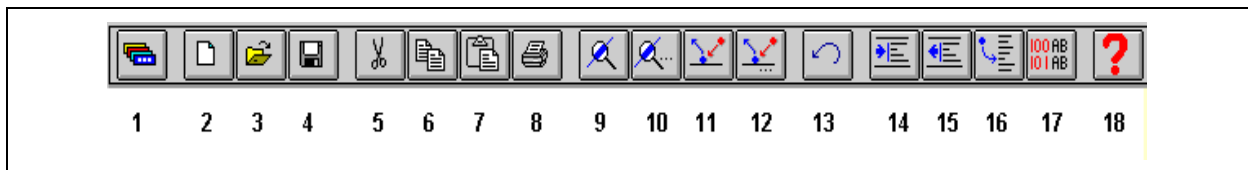
---

---

## 2.1 MPLAB IDE Toolbars

### 2.1.1 Edit Toolbar

The Edit toolbar contains buttons that are commonly used when editing source code.



**Figure 2.1: Edit Toolbar**

The default buttons are:

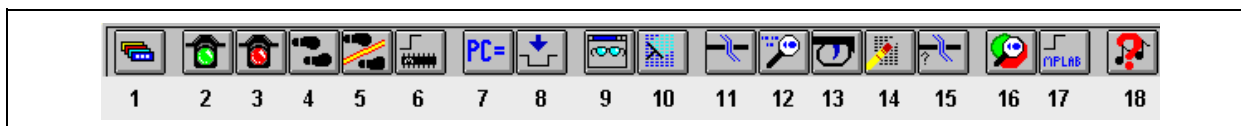
1. Change toolbar
2. File New
3. File Open
4. File Save
5. Cut
6. Copy
7. Paste
8. Print
9. Find
10. Repeat Last Find
11. Replace
12. Repeat Last Replace
13. Undo Edit
14. Indent
15. Unindent
16. Goto Line
17. Toggle Line Numbers
18. Help Contents

# MPLAB® IDE User's Guide

---

## 2.1.2 Debug Toolbar

The Debug toolbar contains buttons that are commonly used when running and debugging code.



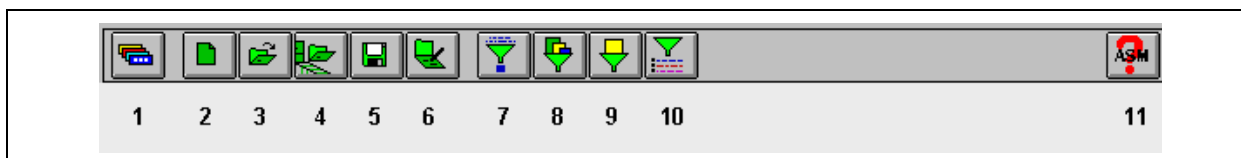
**Figure 2.2: DebugToolbar**

The default buttons are:

1. Change toolbar
2. Run Program
3. Halt Program
4. Step Through Program
5. Step Over
6. Reset System
7. Change PC
8. Execute Opcode
9. New Watch Window
10. Modify Window
11. Break Point
12. Trace Point
13. Trigger
14. Clear All Breaks
15. Conditional Break
16. Halt Trace
17. System Reset
18. Help Release Notes

## 2.1.3 Project Toolbar

The Project toolbar contains icons that are commonly used when running and debugging code.



**Figure 2.3: Project Toolbar**

The default buttons are:

1. Change toolbar

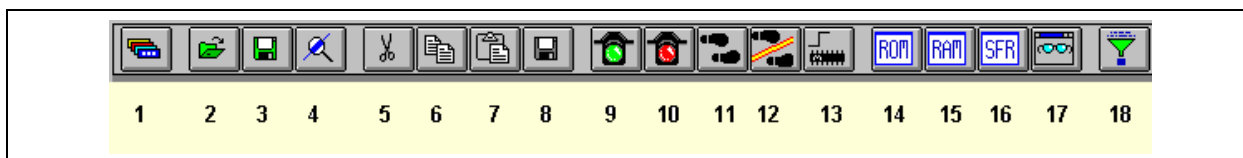
# MPLAB IDE Toolbar and Status Bar Definitions

---

2. New Project
3. Open Project
4. Close Project
5. Save Project
6. Edit Project
7. Make Project
8. Build All
9. Build Node
10. Install Tools
11. Help

## 2.1.4 User Defined Toolbar

The User Defined toolbar is intended to be customized to contain buttons that meet the individual user's needs.



**Figure 2.4: User Defined Toolbar**

It initially contains the following buttons:

1. Change Toolbar
2. Open Project
3. Save Project
4. Find
5. Cut
6. Copy
7. Paste
8. Save File
9. Run
10. Halt
11. Step
12. Step Over
13. Reset
14. Program Memory Window
15. File Registers Window
16. Special Function Registers Window
17. New Watch Window
18. Make Project

# MPLAB<sup>®</sup> IDE User's Guide

## 2.2 MPLAB IDE Status Bar

The Status Bar indicates such current information as cursor position, development mode and device, and active toolbar.

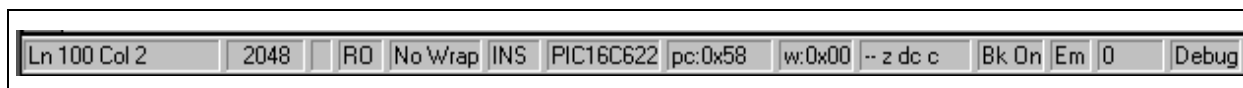


Figure 2.5: Status Bar

Title	Typical Entry	Description	Result from Double Clicking
Line No., Column–Windows Open  Or, displays MPLAB IDE Version Number when no windows are open	Ln 1 Col 1  2.00.00	Displays current line number and column in file MPLAB IDE Version Number	Opens Goto Line Dialog  No Action
Lines in File	72	Displays number of lines in current text file	No Action
File Modified	#	Displays # Symbol if file has been changed since opening	No Action
Write/Read Only	WR	Displays Write/Read Only Status. WR = Editable File RO = Read Only File	Toggles between write and read only for files that you have access to
Text Wrap	No Wrap	Displays current wrap mode and wrap column if text wrap is on Example 1: NoWrap Example 2: WR 72 Useful for text files. Use <i>Options &gt; Current Editor Modes</i> to change wrap column.	Toggles between wrap and no wrap No Wrap Wrap at Column 72
Insert/Strikeover	INS	Toggles typing mode between insert and strikeover INS = Insert Characters OVR = Type over characters	Toggles between INS and OVR
Current Processor	PIC16C61	Displays the currently selected processor	No Action
Current Program Counter	pc:0x5f	Displays the current program counter	Opens Change Program Counter dialog
Current w Register Value	W:0x00	Displays current w register value	No Action
Status Bits	ov Z dc c	Upper Case = Set (1) Lower Case = Reset (0)	No Action
Global Break Enable	Bk On	Displays current status of Global Break Enable	Toggles Global Break Enable On and Off

# MPLAB IDE Toolbar and Status Bar Definitions

---

---

Title	Typical Entry	Description	Result from Double Clicking
Current Development Mode	Sim	Displays Current Development Mode. Examples: EO = Editor Only Sim = Simulator – MPLAB SIM Si = Simulator – SIMICE ICE = Emulator – MPLAB ICE 2000 Em = Emulator – PICMASTER emulator	Displays Development Mode Dialog
Processor Frequency	4 MHz	Displays current processor frequency	Opens processor clock dialog
Current Toolbar	Edit	Displays current toolbar	No Action

# MPLAB<sup>®</sup> IDE User's Guide

---

---

NOTES:



**Appendix A. MPLAB IDE Key Mapping Functions**

**A.1 INTRODUCTION**

This appendix lists the available MPLAB IDE key mapping functions.

**A.2 MPLAB IDE KEY MAPPING FUNCTIONS**

Key Map Function	Definition	Default Key Assignment
CursorBottomOfWindow	"Move Cursor to Bottom of Window"	Ctrl+PgDn
CursorBottomOfWindowSelect	"Move Cursor to Bottom of Window Selecting"	Ctrl+Shift+PgDn
CursorDown	"Move Cursor Down"	Down
CursorDownSelect	"Move Cursor Down Selecting"	Shift+Down
CursorEndOfFile	"Goto End of File"	Ctrl+End
CursorEndOfFileSelect	"Goto End of File Selecting"	Ctrl+Shift+End
CursorEndOfLine	"Move Cursor to End of Line"	End
CursorEndOfLineSelect	"Move Cursor to End of Line Selecting"	Shift+End
CursorLeft	"Move Cursor Left"	Left
CursorLeftSelect	"Move Cursor Left Selecting"	Shift+Left
CursorLeftWord	"Move Cursor Left by Word"	Ctrl+Left
CursorLeftWordSelect	"Move Cursor Left by Word Selecting"	Ctrl+Shift+Left
CursorPageDown	"Page Down"	PgDn
CursorPageDownSelect	"Page Down Selecting"	Shift+PgDn
CursorPageUp	"Page Up"	PgUp
CursorPageUpSelect	"Page Up Selecting"	Shift+PgUp
CursorRight	"Cursor Right"	Right
CursorRightSelect	"Cursor Right Selecting"	Shift+Right
CursorRightWord	"Cursor Right by Word"	Ctrl+Right
CursorRightWordSelect	"Cursor Right by Word Selecting"	Ctrl+Shift+Right
CursorStartOfFile	"Cursor Start of File"	Ctrl+Home
CursorStartOfFileSelect	"Cursor Start of File Selecting"	Ctrl+Shift+Home
CursorStartOfLine	"Cursor Start of Line"	Home
CursorStartOfLineSelect	"Cursor Start of Line Selecting"	Shift+Home
CursorStartOfText	"Cursor Start of Text" Alt Home	Alt+Home
CursorStartOfTextSelect	"Cursor Start of Text Selecting"	Alt+Shift+Home
CursorTopOfWindow	"Cursor Top of Window"	Ctrl+PgUp

# MPLAB<sup>®</sup> IDE User's Guide

Key Map Function	Definition	Default Key Assignment
CursorTopOfWindowSelect	"Cursor Top of Window Selecting"	Ctrl+Shift+PgUp
CursorUp	"Cursor Up"	Up
CursorUpSelect	"Cursor Up Selecting"	Shift+Up
DebugAnimate	"Animate"	Ctrl+F9
DebugBreak	"Set Break Settings"	F2
DebugCenterDebug	"Center Debug Location"	
DebugChangePC	"Change Program Counter"	
DebugClearAll	"Clear All Qualifiers"	
DebugClearMemory	"Clear Program Memory"	Ctrl+Shift+F2
DebugConditionalBreak	"Conditional Break"	
DebugExecuteOpcode	"Execute an Opcode"	
DebugHalt	"Halt the Processor"	F5
DebugHaltTrace	"Halt the Trace"	Shift+F5
DebugPerformanceAnalysis	"Performance Analysis"	
DebugPORReset	"Power-On-Reset"	Ctrl+Shift+F5
DebugStep	"Step"	F7
DebugReset	"Reset Processor"	F6
DebugRun	"Run"	F9
DebugStepOver	"Step Over"	F8
DebugStepTrace	"Step The Trace Window"	Shift+F7
DebugAsyncStim	"Asynchronous Stimulus"	
DebugClockStim	"Clock Stimulus"	
DebugPinStim	"Pin Stimulus"	
DebugRegStim	"Register Stimulus"	
DebugSystemReset	"System Reset"	Ctrl+Shift+F3
DebugTrace	"Trace Settings"	
DebugTrigger	"Trigger Settings"	
DebugUpdateRegisters	"Update Registers"	
EditCancelSelection		keypad5
EditClearUndo	"Forgets details of all stored undo actions"	
EditCopy	"Copies highlighted text to the clipboard"	Ctrl+C Ctrl+Ins
EditCut	"Cuts highlighted text to the clipboard"	Ctrl+X Shift+Del

# MPLAB IDE Key Mapping Functions

Key Map Function	Definition	Default Key Assignment
EditDeleteBackwards	“Delete Character Backwards”	Backspace Ctrl+H
EditDeleteForwards	“Delete Forwards”	Del
EditDeleteLine	“Deletes the entire line containing the cursor”	Ctrl+Shift+K
EditDeleteSelection	“Delete Selection”	
EditDeleteToEndOfLine	“Deletes from the cursor to end of line”	Ctrl+K
EditFind	“Searches for a text string”	F3
EditGotoLine	“Moves the cursor to a specific line”	Ctrl+G
EditInsertHardTab	“Insert Hard Tab”	
EditInsertSoftTab	“Insert Soft Tab”	
EditInsertTab	“Insert Tab”	Tab Ctrl+I
EditMarkUnchanged	“Mark File As Unchanged”	
EditNewLine	“Insert New Line”	Enter Ctrl+M
EditPaste	“Pastes the clipboard at the cursor position”	Ctrl+V Shift+Ins
EditRepeatLastFind	“Repeats the last search action exactly”	Shift+F3
EditRepeatLastReplace	“Repeats the last replace action exactly”	Shift+F4
EditReplace	“Replaces a text string”	F4
EditSelectAll	“Highlights all the text in the current window”	
EditSelectLine	“Select Line”	
EditSelectWord	“Highlights the word containing the cursor”	
EditShowCursor	“Scrolls the window to bring the cursor into view”	
EditShowNextLine	“Show Next Line”	
EditShowNextPage	“Show Next Page”	
EditShowPreviousLine	“Show Previous Line”	
EditShowPreviousPage	“Show Previous Page”	
EditSplitLine	“Split Line”	Ctrl+Shift+O
EditTextIndent	“Moves text right by one tab stop”	
EditTextInsertASCIIcode	“Inserts an arbitrary character code”	Ctrl+Q
EditTextLowercaseSelection	“Converts the highlighted text to lower case”	
EditTextMatchBrace	“Moves to a matching brace character”	Ctrl+B
EditTextMatchBraceSelect	“Moves to a matching brace character and highlights”	Shift+Ctrl+B

# MPLAB<sup>®</sup> IDE User's Guide

---

Key Map Function	Definition	Default Key Assignment
EditTextTransposeCharacters	"Swaps the characters to left and right of the cursor"	Ctrl+T
EditTextUnIndent	"Moves text left by one tab stop"	
EditTextUppercaseSelection	"Converts the highlighted text to upper case"	
EditTextWidenBraceSelect	"Highlights the next largest braced area of text"	Shift+Ctrl+W
EditUndo	"Undoes the last edit action"	Ctrl+Z
ExecDosCommand	"Runs a DOS command and captures output"	F11
ExecRepeatDosCommand	"Repeats the last DOS command-with-capture"	Ctrl+F11
FileAbandon	"Abandon File"	
FileClose	"Closes the file in the current window"	
FileCloseAll	"Closes all open files"	Shift+F9
FileExit	"Ends your MPLAB IDE session"	Alt+F4
FileImportDownloadToMemory	"Download A Hex file to the Engine"	
FileImportDownloadToTarget	"Download A Hex file to the Target"	
FileImportReadTarget	"Copy Engine Memory to Target"	
FileInsert	"Inserts a file at the position of the cursor"	
FileName	"Changes the file name for the current window"	
FileNew	"Creates a new, empty edit window"	Ctrl+N
FileOpen	"Opens an existing file"	Ctrl+O
FilePrint	"Prints the current file"	Ctrl+P
FilePrintSetup	"Changes details of the current printer"	
FileSave	"Saves the current file to disk"	Ctrl+S
FileSaveAll	"Saves all open files to disk"	
FileSaveAs	"Saves the current file to disk"	
FileSaveHex	"Save Hex File"	
FileSaveTrace	"Save Trace File"	
FileSimulatorStimulus	"Load Simulator Stimulus File"	
FileView	"Opens an existing file in read-only mode"	
FileWrite	"Writes the current file to disk"	
HelpAbout	"Gives information about this MPLAB IDE version"	
HelpBugs	"MPLAB IDE Bug List"	
HelpCommands	"Gives help on MPLAB IDE commands"	
HelpContents	"Enters the MPLAB IDE help file at the Contents screen"	F1
HelpEditor	"Editor Help"	

# MPLAB IDE Key Mapping Functions

Key Map Function	Definition	Default Key Assignment
HelpMpasm	"MPASM Assembler Help"	
HelpMpc	"MPLAB C Help"	
HelpOnHelp	"Gives help on using the help system"	
HelpPICmicro	"PICmicro Users Guide"	
HelpReleaseNotes	"Release Notes"	Shift+F1
OptionsColors	"Edit Color Options"	
OptionsCommunicationsPort	"Set Communications Address"	
OptionsCurrent	"Sets modes for the current window and file"	
OptionsDefault	"Sets default modes for files"	
OptionsDevelopmentMode	"Select Development Mode"	
OptionsEnvironmentSetup	"Setup Environment"	Ctrl+F7
OptionsHardware	"Select Hardware Options"	
OptionsKeyMapping	"Changes the mapping of keys to commands"	
OptionsLoadSetup	"Load Setup File"	
OptionsMultiProcessor	"Setup Multi-Processor"	
OptionsPreferences	"Sets MPLAB IDE configuration options"	
OptionsResetModes	"Sets the current file/window modes to default values"	
OptionsSaveSetup	"Save Setup File"	
OptionsScreenFontANSI	"Sets the screen font to the standard ANSI fixed-pitch font"	
OptionsScreenFontOEM	"Sets the screen font to the standard OEM font"	
OptionsScreenFontOther	"Selects the screen font from all available fixed-pitch fonts"	
OptionsScreenFontSystem	"Sets the screen font to the standard system font"	
OptionsSimulatorIOSetup	"Setup Simulator I/O"	
OptionsToggleInsertMode	"Toggle Insert Mode"	Ins
OptionsToggleLineNumbers	"Toggle Line Numbers"	
OptionsToggleStatusBar	"Hides or shows the status bar"	
OptionsToolbarBottom	"Moves the toolbar to the bottom of the window"	
OptionsToolbarFloat	"Makes the toolbar a floating window"	
OptionsToolbarHide	"Makes the toolbar invisible"	
OptionsToolbarLeft	"Moves the toolbar to the left of the window"	
OptionsToolbarRight	"Moves the toolbar to the right of the window"	
OptionsToolbarShow	"Makes the toolbar visible"	

# MPLAB<sup>®</sup> IDE User's Guide

---

Key Map Function	Definition	Default Key Assignment
OptionsToolbarTop	"Moves the toolbar to the top of the window"	
ProjectBuildAll	"Build Full Project"	Ctrl+F10
ProjectBuildNode	"Build Node"	Alt+F10
ProjectCloseProject	"Close Project"	
ProjectCompileSingle	"Compile Single File"	Alt+F10
ProjectEditProject	"Edit the Project Definition"	Ctrl+F3
ProjectMakeProject	"Make the Current Project"	F10
ProjectMakeSetup	"Setup the Project Make"	
ProjectNewProject	"Create a New Project"	
ProjectOpenProject	"Open a Project"	Ctrl+F2
ProjectSaveProject	"Save the Current Project"	
SwapToolbar	"Swap Toolbar"	
SysSetMenuMode	"Set Menu Mode"	
TemplateDelete	"Deletes a template from a template file"	
TemplateEdit	"Edits a template from a template file"	
TemplateFileAttach	"Loads a template file for use"	
TemplateFileCreate	"Creates an empty template file"	
TemplateFileDetach	"Releases an attached template file"	
TemplateFileSave	"Saves an altered template file to disk"	
TemplateFindMark	"Searches for a template marker in the current window"	
TemplateInsert	"Inserts a template at the position of the cursor"	
TemplateInsertMark	"Inserts a template marker at the position of the cursor"	
TemplateNew	"Creates a new window for a template"	
TemplateStore	"Saves a template into a template file"	
TemplateStoreAs	"Saves a template into a template file"	
ToolsEmulatorConfiguration	"Setup Emulator Configuration"	
ToolsProgramHeader	"Program Emulator Header"	
ToolsProgramPod	"Program Emulator Pod"	
ToolsVerifyEmulator	"Verify Emulator Components"	
WindowAbsoluteListing	"Absolute Listing"	
WindowArrangeIcons	"Arranges all iconic windows neatly"	
WindowCascade	"Arranges windows in a cascade pattern"	
WindowClose	"Closes the current window"	Ctrl+F4

## MPLAB IDE Key Mapping Functions

Key Map Function	Definition	Default Key Assignment
WindowDuplicate	“Makes a duplicate of the current window”	
WindowEeprom	“EEPROM Memory Window”	
WindowFileRegisters	“File Register Memory”	
WindowIconize	“Iconize Window”	
WindowIconizeAll	“Makes all windows into icons”	
WindowLoadWatch	“Load a Watch Window”	
WindowMaximize	“Maximize Window”	
WindowModify	“Modify Window”	
WindowNewWatch	“Create New Watch Window”	
WindowNext	“Activates the next non-iconic window”	
WindowProgramMemory	“Program Memory Window”	
WindowRestore	“Restore Window”	
WindowSelect	“Chooses a window to activate from a list”	Ctrl+W
WindowSpecialFunctionRegisters	“Special Functions Register Window”	
WindowStack	“Stack Window”	
WindowStopwatch	“Stopwatch Window”	
WindowSymbolList	“Symbol List Window”	Ctrl+F8
WindowTileHorizontal	“Tiles windows to maximize height”	
WindowTileVertical	“Tiles windows to maximize width”	
WindowTrace	“Trace Window”	
WindowWiden	“Maximizes the width of the current window”	

# MPLAB<sup>®</sup> IDE User's Guide

---

---

NOTES:



## Appendix B. MPLAB Editor Default Key Commands

### B.1 INTRODUCTION

This appendix describes the default key commands specific to the MPLAB Editor and lists the equivalent menu commands (if any).

The key commands perform the most common operations quickly. These key commands can be modified to suit individual preferences. By default, no prefix keys are enabled, so that only single keys are available for mapping.

For a table that lists the keys with the names of the functions used in the key mapping dialog, see Appendix A: MPLAB IDE Key Mapping Functions.

### B.2 HIGHLIGHTS

The categories for the default key commands are:

- Function Keys
- Movement Keys
- Control Keys
- Formatting and Editing Keys

### B.3 FUNCTION KEYS

<b>F1</b>	Displays the appropriate help topic when pressed from a window (e.g., File Registers Window).
<b>F2</b>	Executes the <i>Debug &gt; Break Settings</i> command to open the Break Settings dialog.
<b>F3</b>	Executes the <i>Edit &gt; Find</i> command to find strings.
<b>F4</b>	Executes the <i>Edit &gt; Replace</i> command to replace strings.
<b>F5</b>	Executes the <i>Debug &gt; Halt</i> command to halt debugging.
<b>F6</b>	Executes the <i>Debug &gt; Reset</i> command to issue a reset to the emulated or simulated processor.
<b>F7</b>	Executes the <i>Debug &gt; Step</i> command to execute a single opcode from program memory.
<b>F8</b>	Executes the <i>Debug &gt; Step Over</i> command to step over a call instruction in program memory.
<b>F9</b>	Executes the <i>Debug &gt; Run</i> command to issue a run to the emulated/simulated processor.
<b>F10</b>	Executes the <i>Project &gt; Make Project</i> command to initiate a “make” for the current project.

# MPLAB<sup>®</sup> IDE User's Guide

---

<b>F11</b>	Executes the <i>Execute &gt; DOS Command To Window</i> command to run a DOS command and capture its output in a window.
<b>Shift+F3</b>	Executes the <i>Edit &gt; Repeat Find</i> command to repeat the last find operation.
<b>Shift+F4</b>	Executes the <i>Edit &gt; Repeat Replace</i> command to repeat the last replace operation.
<b>Shift+F5</b>	Executes the <i>Debug &gt; Halt Trace</i> command to halt the execution trace for the simulated processor.
<b>Shift+F9</b>	Executes the <i>File &gt; Close All</i> command to close all files and windows.
<b>Ctrl+F2</b>	Executes the <i>Project &gt; Open Project</i> command to open the Open Project Dialog.
<b>Ctrl+F3</b>	Executes the <i>Project &gt; Edit Project</i> command to open the Edit Project Dialog.
<b>Ctrl+F7</b>	Executes the <i>Options &gt; Environment Setup</i> to open the Environment Setup dialog.
<b>Ctrl+F8</b>	Executes the <i>Windows &gt; Show Symbol List</i> command to open the Symbol List dialog.
<b>Ctrl+F10</b>	Executes the <i>Project &gt; Build All</i> command to build all the source files for the current project.
<b>Alt+F4</b>	Executes the <i>File &gt; Exit</i> command to end your MPLAB Editor session.
<b>Alt+F10</b>	Executes the <i>Project &gt; Build Node</i> command to build the current node.
<b>Shift+Ctrl+F2</b>	Executes the <i>Debug &gt; Clear Program Memory</i> command to clear all of program memory to an "erased" state.
<b>Shift+Ctrl+F3</b>	Executes the <i>Debug &gt; System Reset</i> command to reset the entire emulator system.
<b>Shift+Ctrl+F5</b>	Executes the <i>Debug &gt; POR Reset Emulation</i> command to open the POR Reset dialog.

# MPLAB Editor Default Key Commands

## B.4 MOVEMENT KEYS

Adding <Shift> to cursor movement keys causes a text selection to be extended.

Note that where the <Alt> key is combined with either an arrow key, or <Home>, <End>, <PgDn>, <PgUp>, <Ins> or <Del>, you must use the keys in the extended key areas, and not those in the numeric keypad.

<b>Up</b>	Moves the cursor up by one line
<b>Shift+Up</b>	Moves the cursor up by one line, extending the highlighting
<b>Down</b>	Moves the cursor down by one line
<b>Shift+Down</b>	Moves the cursor down by one line, extending the selection
<b>Left</b>	Moves the cursor left by one character
<b>Shift+Left</b>	Moves the cursor left by one character, extending the selection
<b>Ctrl+Left</b>	Moves the cursor left by one word
<b>Ctrl+Shift+Left</b>	Moves the cursor left by one word, extending the selection
<b>Right</b>	Moves the cursor right by one character
<b>Shift+Right</b>	Moves the cursor right by one character, extending the selection
<b>Ctrl+Right</b>	Moves the cursor right by one word
<b>Ctrl+Shift+Right</b>	Moves the cursor right by one word, extending the selection
<b>PgDn</b>	Moves the cursor down by one page
<b>Shift+PgDn</b>	Moves the cursor down by one page, extending the selection
<b>Ctrl+PgDn</b>	Moves the cursor to the start of the last line in the window
<b>Ctrl+Shift+PgDn</b>	Moves the cursor to the start of the last line in the window, extending the selection
<b>PgUp</b>	Moves the cursor up by one page
<b>Shift+PgUp</b>	Moves the cursor up by one page, extending the selection
<b>Ctrl+PgUp</b>	Moves the cursor to the start of the first line of the window
<b>Ctrl+Shift+PgUp</b>	Moves the cursor to the start of the first line of the window, extending the selection
<b>Home</b>	Moves the cursor to the start of the line
<b>Shift+Home</b>	Moves the cursor to the start of the line, extending the selection

# MPLAB<sup>®</sup> IDE User's Guide

---

---

<b>Ctrl+Home</b>	Moves the cursor to the start of the file
<b>Ctrl+Shift+Home</b>	Moves the cursor to the start of the file, extending the selection
<b>Alt+Home</b>	Moves the cursor to the first non-white-space character in the current line
<b>Alt+Shift+Home</b>	Moves the cursor to the first non-white-space character in the current line, extending the selection
<b>End</b>	Moves the cursor to the end of the line
<b>Shift+End</b>	Moves the cursor to the end of the line, extending the selection
<b>Ctrl+End</b>	Moves the cursor to the end of the file
<b>Ctrl+Shift+End</b>	Moves the cursor to the end of the file, extending the selection

# MPLAB Editor Default Key Commands

## B.5 CONTROL KEYS

<b>Ctrl+Shift+B</b>	Moves the cursor to the brace character matching the brace the cursor is currently on, and highlights all the text between and including the brace characters
<b>Ctrl+C</b>	Executes the <i>Edit &gt; Copy</i> command to copy selected text to the clipboard
<b>Ctrl+G</b>	Executes the <i>Edit &gt; Goto Line</i> command to move the cursor to a specific line
<b>Ctrl+H</b>	Deletes the character to the left of the cursor
<b>Ctrl+I</b>	Inserts a TAB character, or the required number of spaces to bring the cursor to the next TAB stop, depending on whether the current window's Window Mode is set for hard or soft tabs.
<b>Ctrl+K</b>	Executes the <i>Edit &gt; Delete To End Of Line</i> command to delete everything from the cursor position to the end of the line
<b>Ctrl+Shift+K</b>	Executes the <i>Edit &gt; Delete Line</i> command to delete the entire line that the cursor is in
<b>Ctrl+N</b>	Executes the <i>File &gt; New</i> command to create a new, empty edit window
<b>Ctrl+O</b>	Executes the <i>File &gt; Open</i> command to open an existing file
<b>Ctrl+Shift+O</b>	Splits the current line at the position of the cursor, leaving the cursor unmoved
<b>Ctrl+P</b>	Executes the <i>File &gt; Print</i> command to print the file showing in the current window
<b>Ctrl+Q</b>	Executes the <i>Edit &gt; Text Insert ASCII Code</i> command to insert a character specified by its ASCII code number
<b>Ctrl+S</b>	Executes the <i>File &gt; Save</i> command to save the current file to disk
<b>Ctrl+T</b>	Executes the <i>Edit &gt; Text Transpose Characters</i> command to transpose the character under the cursor with the one to its left
<b>Ctrl+V</b>	Executes the <i>Edit &gt; Paste</i> command to paste data from the clipboard into the current window
<b>Ctrl+W</b>	Executes the <i>Window &gt; Select</i> command to let you choose between many open windows
<b>Ctrl+Shift+W</b>	With the cursor between braces, this command highlights the closest (inner-most) pair of braces and all the text between them
<b>Ctrl+X</b>	Executes the <i>Edit &gt; Cut</i> command to cut selected text to the clipboard
<b>Ctrl+Z</b>	Executes the <i>Edit &gt; Undo</i> command to undo the last edit action

## B.6 FORMATTING AND EDITING KEYS

<b>Enter</b>	Inserts a new line
<b>BackSpace</b>	Deletes the character to the left of the cursor
<b>Del</b>	Deletes the character to the right of the cursor
<b>Shift+Del</b>	Executes the <i>Edit &gt; Cut</i> command to cut selected text to the clipboard
<b>Ins</b>	Toggles the current window between Insert and Overwrite modes
<b>Shift+Ins</b>	Executes the <i>Edit &gt; Paste</i> command to paste data from the clipboard into the current window
<b>Ctrl+Ins</b>	Executes the <i>Edit &gt; Copy</i> command to copy selected text to the clipboard
<b>Tab</b>	Inserts a tab character, or the required number of spaces to bring the cursor to the next tab stop, depending on whether the current window's Window Mode is set for hard or soft tabs.

---

---

**Appendix C. File Extensions Used by MPLAB IDE**

---

---

The default extensions of files used by the MPLAB IDE are listed below:

*.ASM	Assembly language source file
*.C	C source file
*.CFG	Configuration/setup files
*.CSV	Trace save files (MPLAB ICE 2000 only)
*.COD	Contains symbolic information and object code
*.DAT	Simulator data file
*.ERR	Error file generated by assembler/compiler
*.H	C include file
*.HEX	PICmicro machine code in hex format
*.HLP	Help file
*.INC	Assembly language include file
*.INI	MPLAB IDE and language tool configuration file
*.KEY	MPLAB IDE key mappings
*.LKR	MPLINK linker script
*.LST	Absolute listing file generated by assembler/compiler
*.MTC	Language tool configuration file
*.PJT	Contains most of the information related to a project
*.REG	Stimulus register file
*.STI	Stimulus pin file
*.TB	Conditional break trace file
*.TBR	Toolbar file
*.TPL	Template file
*.TRC	Trace save files
*.TXT	Trace save file other than MPLAB ICE 2000
*.WAT	Watch window file

# MPLAB<sup>®</sup> IDE User's Guide

---

NOTES:



---

---

## Glossary

---

---

### Introduction

To provide a common frame of reference, this glossary defines the terms for several Microchip tools.

### Highlights

This glossary contains terms and definitions for the following tools:

- MPLAB IDE, MPLAB SIM Simulator, MPLAB Editor
- MPASM Assembler, MPLINK Linker, MPLIB Librarian
- MPLAB CXX C Compilers
- MPLAB ICE 2000 In-Circuit Emulator, PICMASTER Emulator
- MPLAB ICD In-Circuit Debugger
- PICSTART Plus, PRO MATE programmer

### Terms

#### **Absolute Section**

A section with a fixed (absolute) address which can not be changed by the linker.

#### **Access RAM (PIC18CXXX Devices Only)**

Special general purpose registers on PIC18CXXX devices that allow access regardless of the setting of the bank select bit (BSR).

#### **Alpha Character**

Alpha characters are those characters, regardless of case, that are letters of the alphabet: (a, b, ..., z, A, B, ..., Z).

#### **Alphanumeric**

Alphanumeric characters include alpha characters and numbers: (0,1, ..., 9).

#### **Application**

A set of software and hardware developed by the user, usually designed to be a product controlled by a PICmicro microcontroller.

#### **Assemble**

What an assembler does. See assembler.

#### **Assembler**

A language tool that translates a user's assembly source code (.asm) into machine code. MPASM is Microchip's assembler.

# MPLAB<sup>®</sup> IDE User's Guide

---

---

## **Assembly**

A programming language that is once removed from machine language. Machine languages consist entirely of numbers and are almost impossible for humans to read and write. Assembly languages have the same structure and set of commands as machine languages, but they enable a programmer to use names (mnemonics) instead of numbers.

## **Assigned Section**

A section which has been assigned to a target memory block in the linker command file. The linker allocates an assigned section into its specified target memory block.

## **Break Point – Hardware**

An event whose execution will cause a halt.

## **Break Point – Software**

An address where execution of the firmware will halt. Usually achieved by a special break opcode.

## **Build**

A function that recompiles all the source files for an application.

## **C**

A high level programming language that may be used to generate code for PICmicro MCUs, especially high-end device families.

## **Calibration Memory**

A special function register or registers used to hold values for calibration of a PICmicro microcontroller on-board RC oscillator.

## **COFF**

Common Object File Format. An intermediate file format generated by the MPLINK linker that contains machine code and debugging information.

## **Command Line Interface**

Command line interface refers to executing a program on the DOS command line with options. Executing MPASM assembler with any command line options or just the file name will invoke the assembler. In the absence of any command line options, a prompted input interface (shell) will be executed.

## **Compile**

What a compiler does. See compiler.

## **Compiler**

A language tool that translates a user's C source code into machine code. MPLAB C17 and MPLAB C18 are Microchip's C compilers for PIC17CXXX and PIC18CXXX devices, respectively.

---

---

## **Configuration Bits**

Unique bits programmed to set PICmicro microcontroller modes of operation. A configuration bit may or may not be preprogrammed. These bits are set in the *Options > Development Mode* dialog for simulators or emulators and in the `_ _ CONFIG MPASM` directive for programmers.

## **Control Directives**

Control directives in the MPASM assembler permit sections of conditionally assembled code.

## **Data Directives**

Data directives are those that control the MPASM assembler's allocation of memory and provide a way to refer to data items symbolically; that is, by meaningful names.

## **Data Memory**

General purpose file registers (GPRs) from RAM on the PICmicro MCU device being emulated. The File Register window displays data memory.

## **Directives**

Directives provide control of the assembler's operation by telling the MPASM assembler how to treat mnemonics, define data, and format the listing file. Directives make coding easier and provide custom output according to specific needs.

## **Download**

Download is the process of sending data from the PC host to another device, such as an emulator, programmer or target board.

## **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 the firmware resided on the microcontroller device under development.

## **Emulation Memory**

Program memory contained within the emulator.

## **Emulator**

Hardware that performs emulation.

## **Emulator System**

The MPLAB ICE 2000 emulator system includes the pod, processor module, device adapter, cables, and MPLAB IDE software. The PICMASTER emulator system includes the pod, device-specific probe, cables, and MPLAB IDE software.

# MPLAB<sup>®</sup> IDE User's Guide

---

---

## **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 and break points.

## **Executable Code**

See HEX Code.

## **Export**

Send data out of the MPLAB IDE in a standardized format.

## **Expressions**

Expressions are used in the operand field of the MPASM assembler's source line and may contain constants, symbols, or any combination of constants and symbols separated by arithmetic operators. Each constant or symbol may be preceded by a plus or minus to indicate a positive or negative expression.

<p><b>Note:</b> MPASM assembler expressions are evaluated in 32-bit integer math. (Floating point is not currently supported.)</p>
--

## **Extended Microcontroller Mode (PIC17CXXX and PIC18CXXX Devices Only)**

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 PIC17CXXX or PIC18CXXX device.

## **External Input Line (MPLAB ICE 2000 only)**

An external input signal logic probe line (TRIGIN) for setting an event based upon external signals.

## **External Linkage**

A function or variable has external linkage if it can be accessed from outside the module in which it is defined.

## **External RAM (PIC17CXXX and PIC18CXXX Devices Only)**

Off-chip Read/Write memory.

## **External Symbol**

A symbol for an identifier which has external linkage.

## **External Symbol Definition**

A symbol for a function or variable defined in the current module.

## **External Symbol Reference**

A symbol which references a function or variable defined outside the current module.

---

---

## **External Symbol Resolution**

A process performed by the linker in which external symbol definitions from all input modules are collected in an attempt to update all external symbol references. Any external symbol references which do not have a corresponding definition cause a linker error to be reported.

## **File Registers**

On-chip general purpose and special function registers.

## **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 MCU 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**

See Data Memory.

## **Halt**

A function that stops the emulator. Executing Halt is the same as stopping at a break point. The program counter stops, and the user can inspect and change register values, and single step through code.

## **HEX Code**

Executable instructions assembled or compiled from source code into standard hexadecimal format code. Also called executable or machine 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. This format is readable by a device programmer.

## **High Level Language**

A language for writing programs that is of a higher level of abstraction from the processor than assembler code. High level languages (such as C) employ a compiler to translate statements into machine instructions that the target processor can execute.

## **ICD**

In-Circuit Debugger. MPLAB ICD is Microchip's in-circuit debugger for PIC16F87X devices. MPLAB ICD works with MPLAB IDE.

# MPLAB<sup>®</sup> IDE User's Guide

---

## **ICE**

In-Circuit Emulator. MPLAB ICE 2000 is Microchip's in-circuit emulator that works with MPLAB IDE.

## **IDE**

Integrated Development Environment. An application that has multiple functions for firmware development. The MPLAB IDE integrates a compiler, an assembler, a project manager, an editor, a debugger, a simulator, and an assortment of other tools within one Windows application. A user developing an application can write code, compile, debug, and test an application without leaving the MPLAB IDE desktop.

## **Identifier**

A function or variable name.

## **Import**

Bring data into the MPLAB Integrated Development Environment (IDE) from an outside source, such as from a HEX file.

## **Initialized Data**

Data which is defined with an initial value. In C, `int myVar=5;` defines a variable which will reside in an initialized data section.

## **Internal Linkage**

A function or variable has internal linkage if it can not be accessed from outside the module in which it is defined.

## **Librarian**

A language tool that creates and manipulates libraries. MPLIB is Microchip's librarian.

## **Library**

A library is a collection of relocatable object modules. It is created by assembling multiple source files to object files, and then using the librarian to combine the object files into one library file. A library can be linked with object modules and other libraries to create executable code.

## **Link**

What a linker does. See Linker.

## **Linker**

A language tool that combines object files and libraries to create executable code. Linking is performed by Microchip's linker, MPLINK.

## **Linker Script Files**

Linker script files are the command files of MPLINK (.LKR). They define linker options and describe available memory on the target platform.

---

---

## **Listing Directives**

Listing directives are those directives that control the MPASM 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, MPASM 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 connected to the emulator. The logic probes provide external trace inputs, trigger output signal, +5V, and a common ground.

## **Machine Code**

Either object or executable code.

## **Macro**

A collection of assembler instructions that are included in the assembly code when the macro name is encountered in the source code. Macros must be defined before they are used; forward references to macros are not allowed.

All statements following a `MACRO` directive and prior to an `ENDM` directive are part of the macro definition. Labels used within the macro must be local to the macro so the macro can be called repetitively.

## **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  $\mu$ C.

## **Memory Models**

Versions of libraries and/or precompiled object files based on a device's memory (RAM/ROM) size and structure.

## **Microcontroller**

A highly integrated chip that contains all the components comprising a controller. Typically this includes a CPU, RAM, some form of ROM, I/O ports, and timers. Unlike a general-purpose computer, which also includes all of

# MPLAB® IDE User's Guide

---

these components, a microcontroller is designed for a very specific task – to control a particular system. As a result, the parts can be simplified and reduced, which cuts down on production costs.

## **Microcontroller Mode (PIC17CXXX and PIC18CXXX Devices Only)**

One of the possible program memory configurations of the PIC17CXXX and PIC18CXXX families of microcontrollers. In microcontroller mode, only internal execution is allowed. Thus, only the on-chip program memory is available in microcontroller mode.

## **Microprocessor Mode (PIC17CXXX and PIC18CXXX Devices Only)**

One of the possible program memory configurations of the PIC17CXXX and PIC18CXXX families of microcontrollers. In microprocessor mode, the on-chip program memory is not used. The entire program memory is mapped externally.

## **Mnemonics**

Instructions that are translated directly into machine code. Mnemonics are used to perform arithmetic and logical operations on data residing in program or data memory of a microcontroller. They can also move data in and out of registers and memory as well as change the flow of program execution. Also referred to as Opcodes.

## **MPASM Assembler**

Microchip Technology's relocatable macro assembler. The MPASM assembler is a DOS or Windows-based PC application that provides a platform for developing assembly language code for Microchip's PICmicro microcontroller families. Generically, the MPASM assembler will refer to the entire development platform including the macro assembler and utility functions.

The MPASM assembler will translate source code into either object or executable code. The object code created by the MPASM assembler may be turned into executable code through the use of the MPLINK linker.

## **MPLAB CXX**

Refers to MPLAB C17 and MPLAB C18 C compilers.

## **MPLAB ICD**

Microchip's in-circuit debugger for PIC16F87X devices. MPLAB ICD works with MPLAB IDE. The MPLAB ICD system consists of a module, header, demo board (optional), cables, and MPLAB Software.

## **MPLAB ICE 2000**

Microchip's in-circuit emulator that works with MPLAB IDE.

## **MPLAB IDE**

The name of the main executable program that supports the IDE with an Editor, Project Manager, and Emulator/Simulator Debugger. The MPLAB IDE software resides on the PC host. The executable file name is MPLAB.EXE. MPLAB.EXE calls many other files.



---

---

### **MPLAB SIM Simulator**

Microchip's simulator that works with MPLAB IDE.

### **MPLIB Librarian**

MPLIB is a librarian for use with COFF object modules (`filename.o`) created using either MPASM v2.0, MPASMWIN v2.0, or MPLAB C v2.0 or later.

MPLIB librarian will combine multiple object files into one library file. Then the MPLIB librarian can be used to manipulate the object files within the created library.

### **MPLINK Linker**

MPLINK is a linker for the Microchip relocatable assembler, MPASM, and the Microchip C compilers, MPLAB C17 or MPLAB C18. MPLINK linker also may be used with the Microchip librarian, MPLIB. MPLINK linker is designed to be used with MPLAB IDE, though it does not have to be.

MPLINK linker will combine object files and libraries to create a single executable file.

### **MPSIM**

The DOS version of Microchip's simulator. MPLAB SIM simulator is the newest simulator from Microchip.

### **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. Macros can be nested to 16 levels deep.

### **Non Real-Time**

Refers to the processor at a break point or executing single step instructions or MPLAB IDE being run in simulator mode.

### **Node**

MPLAB IDE project component.

### **NOP**

No Operation. An instruction that has no effect when executed except to advance the program counter.

### **Object Code**

The intermediate code that is produced from the source code after it is processed by an assembler or compiler. Relocatable code is code produced by the MPASM assembler or MPLAB C17 and C18 C compilers that can be run through the MPLINK linker to create executable code. Object code is contained in an object file.

# MPLAB<sup>®</sup> IDE User's Guide

---

## Object File

A module which may contain relocatable code or data and references to external code or data. Typically, multiple object modules are linked to form a single executable output. Special directives are required in the source code when generating an object file. The object file contains object code.

## Object File Directives

Directives that are used only when creating an object file.

## Off-Chip Memory (PIC17CXXX and PIC18CXXX Devices Only)

Off-chip memory refers to the memory selection option for the PIC17CXXX or PIC18CXXX device where memory may reside on the target board, or where all program memory may be supplied by the Emulator. The Memory tab accessed from *Options > Development Mode* provides the Off-Chip Memory selection dialog box.

## Opcodes

Operational Codes. See Mnemonics.

## Operators

Arithmetic symbols, like the plus sign '+' and the minus sign '-', that are used when forming well-defined expressions. Each operator has an assigned precedence.

## 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<sup>®</sup> or compatible Personal Computer running Windows 3.1x or Windows 95/98, Windows NT, or Windows 2000. MPLAB IDE runs on 486 or higher machines.

## PICmicro MCUs

PICmicro microcontrollers (MCUs) refers to all Microchip microcontroller families.

## PICMASTER Emulator

The hardware unit that provides tools for emulating and debugging firmware applications. This unit contains emulation memory, break point logic, counters, timers, and a trace analyzer among some of its tools. MPLAB ICE 2000 is the newest emulator from Microchip.

---

---

### **PICSTART Plus Programmer**

A device programmer from Microchip. Programs 8, 14, 28, and 40 pin PICmicro microcontrollers. Must be used with MPLAB Software.

### **Pod**

The external emulator box that contains emulation memory, trace memory, event and cycle timers, and trace/break point logic. Occasionally used as an abbreviated name for the MPLAB ICE 2000 emulator.

### **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.

### **Precedence**

The concept that some elements of an expression are evaluated before others; i.e., \* and / before + and -. In MPASM assembler, operators of the same precedence are evaluated from left to right. Use parentheses to alter the order of evaluation.

### **Program Counter**

A register that specifies the current execution address.

### **Program Memory**

The memory area in a PICmicro microcontroller where instructions are stored. Memory in the emulator or simulator containing the downloaded target application firmware.

### **Programmer**

A device used to program electrically programmable semiconductor devices such as microcontrollers.

### **Project**

A set of source files and instructions to build the object and executable code for an application.

### **PRO MATE Programmer**

A device programmer from Microchip. Programs all PICmicro microcontrollers and most memory and Keeloq devices. Can be used with MPLAB IDE or stand-alone.

### **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.

# MPLAB<sup>®</sup> IDE User's Guide

---

---

## **Radix**

The number base, HEX, or decimal, used in specifying an address and for entering data in the *Window > Modify* command.

## **RAM**

Random Access Memory (Data Memory).

## **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 2000 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 break point 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.

## **Relocatable Section**

A section whose address is not fixed (absolute). The linker assigns addresses to relocatable sections through a process called relocation.

## **Relocation**

A process performed by the linker in which absolute addresses are assigned to relocatable sections and all identifier symbol definitions within the relocatable sections are updated to their new addresses.

## **ROM**

Read Only Memory (Program Memory).

## **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.

## **Section**

An portion of code or data which has a name, size, and address.

## **SFR**

Special Function Registers of a PICmicro microcontroller.

## **Shared Section**

A section which resides in a shared (non-banked) region of data RAM.

---

---

## 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 the PICmicro microprocessor.

## Single Step

This command steps through code, one instruction at a time. After each instruction, MPLAB IDE updates register windows, watch variables, and status displays so you can analyze and debug instruction execution.

You can also single step C compiler source code, but instead of executing single instructions, MPLAB IDE will execute all assembly level instructions generated by the line of the high level C statement.

## Skew

The information associated with the execution of an instruction appears on the processor bus at different times. For example, the executed opcode 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 opcode 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 break point 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 break point is referred to as the skid.

## Source Code – Assembly

Source code consists of PICmicro MCU instructions and MPASM assembler directives and macros that will be translated into machine code by an assembler.

## Source Code – C

A program written in the high level language called “C” which will be converted into PICmicro MCU machine code by a compiler. Machine code is suitable for use by a PICmicro MCU or Microchip development system product like MPLAB IDE.

# MPLAB<sup>®</sup> IDE User's Guide

---

## Source File – Assembly

The ASCII text file of PICmicro MCU instructions and MPASM assembler directives and macros (source code) that will be translated into machine code by an assembler. It is an ASCII file that can be created using any ASCII text editor.

## Source File – C

The ASCII text file containing C source code that will be translated into machine code by a compiler. It is an ASCII file that can be created using any ASCII text editor.

## Special Function Registers

Registers that control I/O processor functions, I/O status, timers, or other modes or peripherals.

## Stack – Hardware

An area in PICmicro MCU memory where function arguments, return values, local variables, and return addresses are stored; i.e., a “Push-Down” list of calling routines. Each time a PICmicro MCU executes a `CALL` or responds to an interrupt, the software pushes the return address to the stack. A return command pops the address from the stack and puts it in the program counter.

The PIC18CXXX family also has a hardware stack to store register values for “fast” interrupts.

## Stack – Software

The compiler uses a software stack for storing local variables and for passing arguments to and returning values from functions.

## 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 toolbar.

## Step Into

This command is the same as Single Step. Step Into (as opposed to Step Over) follows a `CALL` instruction into a subroutine.

## Step Over

Step Over allows you to debug code without stepping into subroutines. When stepping over a `CALL` instruction, the next break point 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 break point will never be reached.

The Step Over command is the same as Single Step except for its handling of `CALL` instructions.

---

---

### Stimulus

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.

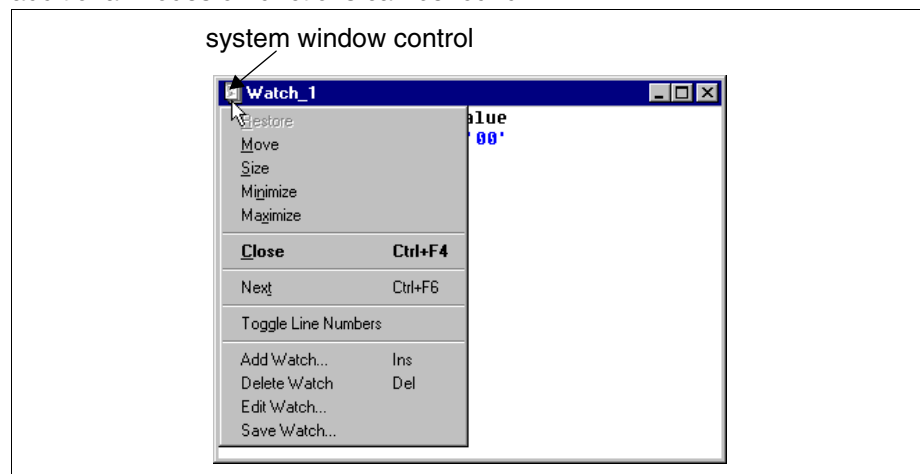
Symbols in MPLAB IDE refer mainly to variable names, function names and assembly labels.

### System Button

The system button is another name for the system window control. Clicking on the system button pops up the system menu.

### 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.” In some MPLAB IDE windows, additional modes or functions can be found.



**Figure G1: System Window Control Menu – Watch Window**

### Target

Refers to user hardware.

### Target Application

Firmware residing on the target board.

### Target Board

The circuitry and programmable device that makes up the target application.

# MPLAB<sup>®</sup> IDE User's Guide

---

## **Target Processor**

The microcontroller device on the target application board that is being emulated.

## **Template**

Lines of text that you build for inserting into your files at a later time. The MPLAB Editor stores templates in template files.

## **Toolbar**

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 break point settings. Any number of trigger output points can be set.

## **Unassigned Section**

A section which has not been assigned to a specific target memory block in the linker command file. The linker must find a target memory block in which to allocate an unassigned section.

## **Uninitialized Data**

Data which is defined without an initial value. In C, `int myVar;` defines a variable which will reside in an uninitialized data section.

## **Upload**

The Upload function transfers data from a tool, such as an emulator or programmer, to the host PC or from the target board to the emulator.

## **Warning**

An alert that is provided to warn you of a situation that would cause physical damage to a device, software file, or equipment.

## **WatchDog Timer (WDT)**

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.

## **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 break point.

# MPLAB<sup>®</sup> IDE User's Guide

---

NOTES:

**Index**

**Symbols**

.asm .....	37
.c .....	37
.cod .....	37
.hex .....	37
.lib .....	37
.lkr .....	37
.lst .....	37
.map .....	37
.o .....	37

**Numerics**

12-Bit Core Devices .....	86
I/O Pins .....	86
Peripherals .....	87
Reset and Sleep Conditions .....	86
Watchdog Timer .....	87
14-Bit Core Devices .....	88
I/O Pins .....	88
Interrupts .....	89
Peripherals .....	91
Reset Conditions .....	89
Sleep .....	89
Special Registers .....	90
Watchdog Timer .....	90
16-Bit Core Devices .....	93
I/O Pins .....	93
Interrupts .....	93
Memory Modes .....	96
Peripherals .....	95
Reset Conditions .....	94
Sleep .....	94
Special Registers .....	94
Watchdog Timer .....	94

**A**

Absolute Listing Window .....	194
Absolute Section .....	241
Access RAM .....	241
Add Watch Window Symbols .....	208
Adding Toolbar Buttons .....	171
Address Labels .....	173
Arrange Icons .....	214
ASCII Display .....	192, 201
Assembler .....	241

Assigned Section .....	242
------------------------	-----

**B**

Break and Trace Points .....	69
Break Options .....	163
Break Point Settings .....	134
Break Point, Hardware .....	242
Break Point, Software .....	242
Break Points .....	69
Real-Time .....	70
Break Settings .....	133, 134
Break on Address Match .....	71
Break on Stack Overflow .....	72
Break on Trace Buffer Full .....	71, 164
Break on Watchdog Timer .....	72
Clear Breakpoints on Download .....	163
Global Break Enable .....	163
Pass Counter .....	72
Saving Breakpoint Settings .....	135, 139
Stack Overflow Break Enable .....	164

**C**

Calibration Data Window .....	192
Calibration Memory .....	113, 242
Calibration Memory Window .....	192
Cascade .....	214
CD-ROM	
Microchip Technology Library .....	4
Center Debug Location .....	134
Change Program Counter .....	130
Change Watch Window Symbol Properties ..	210
Clear All Points .....	144
Clear Memory on Download .....	173
Clear Program Memory .....	146
Clock Frequency .....	157
Code	
Start Up .....	37
Code File .....	37
Colors .....	183
Command Line Interface .....	242
Communications Port Setup .....	184
Compiler .....	242
Conditional Break .....	75, 131
Configuration .....	159
Configuration Bits .....	113, 243

# MPLAB<sup>®</sup> IDE User's Guide

---

---

Creating a Watch Window .....	207	Editor Modes .....	178
Customer Notification Service .....	6	Display/Input Modes .....	179
<b>D</b>		File Modes .....	180
Data Memory .....	243	Printing .....	180
Debug .....	129	Screen Formatting .....	179
Break PointSettings .....	134	EEPROM .....	243
Center Debug Location .....	134	EEPROM Memory .....	113, 192
Clear All Points .....	144	Emulator .....	243
Execute .....	131	Enhanced 16-Bit Core Devices .....	97
Conditional Break .....	131	I/O Pins .....	97
Execute an Opcode .....	131	Interrupts .....	98
Power-On Reset .....	147	Peripherals .....	99
Run .....	129	Reset Conditions .....	98
Change Program Counter .....	130	Sleep .....	98
Halt Trace .....	129	Special Registers .....	99
Reset .....	129	Watchdog Timer .....	99
Run .....	129	Environment Setup .....	169
Step .....	130	Colors .....	183
Step Over .....	130	Editor Modes .....	178
System Reset .....	146	Display/Input Modes .....	179
Default Configuration .....	166	File Modes .....	180
Default Editor Modes .....	169	Printing .....	180
Default File Browse Directory .....	177	Screen Formatting .....	179
Development Mode Options .....	152	File Options .....	177
Break Options .....	163	Default File Browse Directory .....	177
Clock Frequency .....	157	Size of MRU File List .....	178
Configuration .....	159	General Options .....	169
Device Ports .....	154	Clear Memory on Download .....	173
Memory .....	158	Load Default Configuration .....	173
Pin Settings .....	162	Status Bar Enable .....	173
Processor Power .....	161	Track Source Code .....	173
Tools .....	152	Global Switches .....	173
Device Ports .....	154	Key Mappings .....	181
Directives .....	243	Project Template Options .....	174
Control .....	243	Screen Font .....	170
Data .....	243	Symbol Display Width .....	173
Listing .....	247	Toolbar Setup .....	170
Macro .....	247	Executable	
Object File .....	250	Files .....	37
Disabling Toolbar Buttons .....	172	Executable Code .....	244
Disassembled Code .....	114	Execute	
DOS		Conditional Break .....	131
Command to Window .....	185	Execute an Opcode .....	131
Repeat Command to Window .....	186	Execution	
<b>E</b>		Non Real-Time .....	67
Edit .....	101	Exit .....	116
Edit Watch Window Symbols .....	209	Export .....	244
Editing Toolbar Buttons .....	171	Export Memory .....	113
		Export Trace Buffer .....	111

# Index

Expressions .....	244	HEX Code .....	245
Extended Microcontroller Mode .....	244	HEX Display .....	192
External RAM .....	244	HEX File .....	37
<b>F</b>			
File		<b>I</b>	
Exit .....	116	ICD .....	245
Export		ICE .....	246
Export Trace Buffer .....	111	ICEPIC .....	153
Import		Iconize All .....	214
Import to Memory .....	108	IDE .....	9, 246
Import to Target Memory .....	110	IDs .....	113
Listing .....	247	Import .....	108, 246
Print .....	114	Import to Memory .....	108
File Extensions Used by MPLAB .....	239	Import to Target Memory .....	110
File Options .....	177	Import to Memory .....	108
Default File Browse Directory .....	177	Import to Target Memory .....	110
Size of MRU File List .....	178	Index .....	259
File Register Window .....	199	Initialization	
File Registers		Data .....	37
ASCII Display .....	201	Initialized Data .....	246
HEX Display .....	201	Internet Address .....	5
Symbolic Display .....	201	Interrupts	
<b>G</b>			
Global Break Enable .....	163	Handler Code .....	37
Global Switches .....	173	<b>K</b>	
Break on Trace Buffer Full .....	164	Key Mapping Functions .....	225
Clear Breakpoints on Download .....	163	Key Mappings .....	181
Global Break Enable .....	163	Keyboard Shortcut .....	102
Stack Overflow Break Enable .....	164	<b>L</b>	
Glossary .....	241	Librarian .....	246
Grouping Toolbar Buttons .....	172	Library .....	246
<b>H</b>			
Hardware		Files .....	37
Stack Levels .....	196	Linker .....	246
Help .....	216	Script .....	37
About .....	217	Linker Script Files .....	246
Editor Help .....	216	Listing File .....	37, 247
Error Help .....	216	Load Default Configuration .....	173
MPASM Help .....	216	Load Setup .....	166
MPLAB Help .....	216	Load Watch Window .....	207
MPLIB Help .....	216	Local Label .....	247
MPLINK Help .....	216	Logic Probes .....	247
Release Notes .....	216	<b>M</b>	
Tool Help .....	216	Map File .....	37
Tool Release Notes .....	216	MCU .....	247

# MPLAB<sup>®</sup> IDE User's Guide

---

---

Memory .....	96, 158
Calibration .....	242
Data .....	243
Export .....	113
Program .....	251
Trace .....	256
Memory Models .....	247
Menu Shortcut .....	102
Microchip Internet Web Site .....	5
Microcontroller Mode .....	248
Microprocessor Mode .....	248
Mnemonics .....	248
Modify	
Address .....	212
Auto Increment .....	213
Data/Opcode .....	212
End All .....	213
Fill Range .....	213
Memory Area .....	213
Radix .....	212
Read .....	213
Read Range .....	213
Modify Window .....	212
Most-Recently-Used File List .....	116, 178
MPASM Assembler.....	11, 241, 248
MPASM Assembler Help .....	216
MPLAB	
IDE .....	9
MPLAB Editor .....	11
MPLAB File Extensions .....	239
MPLAB IDE .....	248
MPLAB Project Manager .....	11
MPLAB/PICMASTER Help .....	216
MPLAB CXX .....	248
MPLAB CXX C Compiler .....	11
MPLAB ICD .....	248
MPLAB ICE .....	153, 248
Verify .....	186
MPLAB ICE Emulator .....	11
MPLAB SIM Simulator .....	153, 249
MPLAB SIM Simulator .....	11
MPLIB Librarian.....	11, 37, 246, 249
MPLINK Linker.....	11, 37, 246, 249
MPLINK Linker Help .....	216
MRU .....	116
Multiple Cycles .....	76

<b>N</b>	
New Project .....	67
<b>O</b>	
Object Code .....	249
Object Files, Precompiled .....	37
Off-Chip Memory .....	250
Opcodes .....	250
Open Windows .....	215
Operators .....	250
Options .....	150
Development Mode .....	152
Development Mode Options .....	152, 163
Break Options .....	163
Clock Frequency .....	157
Configuration .....	159
Device Ports .....	154
Memory .....	158
Pin Settings .....	162
Processor Power .....	161
Tools .....	152
Environment Setup .....	169
Colors .....	183
Editor Modes .....	178
File Options .....	177
Global Switches .....	173
Key Mappings .....	181
Project Template Options .....	174
Screen Font .....	170
Symbol Display Width .....	173
Toolbar Setup .....	170
Programmer Options .....	184
Communications Port Setup .....	184
Select Programmer .....	184
Window Setup .....	165
Default Configuration .....	166
Load Setup .....	166
Save Setup .....	165
<b>P</b>	
Pass Counter .....	72, 250
Assigned to Break .....	74
Assigned to Trace .....	75
Points .....	73
Set Up .....	73
Using to Count Events .....	75
PICMASTER Emulator.....	153, 250
Verify .....	186
PICmicro MCU .....	253, 254
PICSTART Plus Programmer .....	251

Pin Settings .....	162	Show Symbol List Window .....	204
Power-On-Reset .....	147	Simulator .....	253
Precedence .....	251	Single Cycle .....	76
Print .....	114	Single Step .....	253
Print Options .....	114	Skew .....	253
PRO MATE Programmer.....	251	Skid .....	253
Processor Power .....	161	Sleep .....	86, 89, 94, 98
Program Counter .....	251	Source Code .....	37
Program Memory .....	251	Source Code, Assembly .....	253
Creating Temporary Breakpoint .....	190	Source Code, C .....	253
Load .....	108	Special Function Registers Window .....	202
Save .....	113	Stack	
Window .....	188	12-Bit Core Devices .....	197
Programmer .....	251	14-Bit Core Devices .....	197
Programmer Options .....	184	16-Bit Core Devices .....	198
Communications Port Setup .....	184	Enhanced 16-Bit Core Devices .....	198
Select Programmer .....	184	Levels .....	196
Project .....	251	Overflow .....	72
Project Template Options .....	174	Window .....	196
Project Window .....	206	Stack Overflow Break Enable .....	164
<b>Q</b>		Stack, Hardware .....	254
Qualifier .....	251	Stack, Software .....	254
<b>R</b>		Start Up Code .....	37
Radix .....	252	Status Bar .....	103
Real-Time .....	252	Current Development Mode .....	223
Register Definitions .....	37	Current Processor .....	222
Register Variables .....	173	File Modified .....	222
Release Notes .....	216	Insert/Strikeover .....	222
Relocatable Section .....	252	Line No. ....	222
Removing Toolbar Buttons .....	172	Lines in File .....	222
Repeat DOS Command to Window .....	186	MPLAB Version .....	222
Reset .....	129	Write/Read Only .....	222
Conditions .....	86, 89, 94, 98	Status Bar Enable .....	173
Run .....	129	Step Over .....	130
<b>S</b>		Stimulus .....	134, 255
Save Setup .....	165	Stopwatch .....	255
Save Watch Window .....	211	Target Frequency .....	205
Screen Font .....	170	Window .....	205
Section .....	252	Symbol .....	255
Absolute .....	241	Symbol Display Width .....	173
Assigned .....	242	Address Labels .....	173
Relocatable .....	252	Register Variables .....	173
Shared .....	252	Symbolic Display .....	201
Unassigned .....	256	System Button .....	255
Select Programmer .....	184	System Reset .....	146
Shared Section .....	252	System Window Control .....	255
Shortcut .....	102		

# MPLAB<sup>®</sup> IDE User's Guide

---

---

## T

Target .....	255
Tile Horizontal .....	213
Tile Vertical .....	214
Tool Release Notes .....	216
Toolbar .....	103
Debug .....	220
Edit .....	219
Project .....	220
User Defined .....	221
Toolbar Setup .....	170, 171, 172
Tools .....	152, 185
DOS Command to Window .....	185
Repeat DOS Command to Window .....	186
Verify MPLAB ICE .....	186
Verify PICMASTER Emulator .....	186
Trace .....	256
Circular Trace Buffer .....	72
Halting from the Toolbar .....	73
Trace Buffer Full .....	71
Trace Data .....	76
Trace Memory .....	73, 256
Window .....	191
Trace Points .....	69
Real-Time .....	72
Trace Settings .....	138
Track Source Code .....	173

## U

Unassigned Section .....	256
Uninitialized Data .....	256

## V

Verify MPLAB ICE .....	186
Verify PICMASTER Emulator .....	186

## W

Watchdog Timer .....	87, 90, 99, 256
Watch Window .....	257
Add Symbols .....	208
Creating .....	207
Edit Symbols .....	209
Load Watch Window .....	207
Save Watch Window .....	211
Symbol Properties .....	210
Watchdog Timer .....	72, 94, 159
WDT .....	256
Window .....	187
Absolute Listing .....	194
Arrange Icons .....	214

Calibration Data .....	192
Calibration Memory .....	192
Cascade .....	214
EEPROM Memory .....	192
File Registers .....	199
Iconize All .....	214
Modify .....	212
Open Windows .....	215
Program Memory .....	146, 188
Project .....	206
Show Symbol List .....	204
Special Function Registers .....	202
Stack .....	196
Stopwatch .....	205
Tile Horizontal .....	213
Tile Vertical .....	214
Trace Memory .....	142, 191
Watch Window .....	207
Add Symbols .....	208
Edit .....	209
Load Watch Window .....	207
Save Watch Window .....	211
Symbol Properties .....	210
Window Setup .....	165
Default Configuration .....	166
Load Setup .....	166
Save Setup .....	165
WWW Address .....	5



**NOTES:**

# MPLAB<sup>®</sup> IDE User's Guide

---

---

NOTES:

**NOTES:**

# MPLAB<sup>®</sup> IDE User's Guide

---

---

NOTES:

**NOTES:**

# MPLAB<sup>®</sup> IDE User's Guide

---

---

NOTES:

**NOTES:**



# MICROCHIP

## 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: 480-792-7627  
Web Address: <http://www.microchip.com>

#### Rocky Mountain

2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7966 Fax: 480-792-4338

#### Atlanta

500 Sugar Mill Road, Suite 200B  
Atlanta, GA 30350  
Tel: 770-640-0034 Fax: 770-640-0307

#### Boston

2 Lan Drive, Suite 120  
Westford, MA 01886  
Tel: 978-692-3848 Fax: 978-692-3821

#### Chicago

333 Pierce Road, Suite 180  
Itasca, IL 60143  
Tel: 630-285-0071 Fax: 630-285-0075

#### Dallas

4570 Westgrove Drive, Suite 160  
Addison, TX 75001  
Tel: 972-818-7423 Fax: 972-818-2924

#### Detroit

Tri-Atria Office Building  
32255 Northwestern Highway, Suite 190  
Farmington Hills, MI 48334  
Tel: 248-538-2250 Fax: 248-538-2260

#### Kokomo

2767 S. Albright Road  
Kokomo, Indiana 46902  
Tel: 765-864-8360 Fax: 765-864-8387

#### Los Angeles

18201 Von Karman, Suite 1090  
Irvine, CA 92612  
Tel: 949-263-1888 Fax: 949-263-1338

#### San Jose

Microchip Technology Inc.  
2107 North First Street, Suite 590  
San Jose, CA 95131  
Tel: 408-436-7950 Fax: 408-436-7955

#### Toronto

6285 Northam Drive, Suite 108  
Mississauga, Ontario L4V 1X5, Canada  
Tel: 905-673-0699 Fax: 905-673-6509

### ASIA/PACIFIC

#### Australia

Microchip Technology Australia Pty Ltd  
Suite 22, 41 Rawson Street  
Epping 2121, NSW  
Australia  
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

#### China - Beijing

Microchip Technology Consulting (Shanghai)  
Co., Ltd., Beijing Liaison Office  
Unit 915  
Bei Hai Wan Tai Bldg.  
No. 6 Chaoyangmen Beidajie  
Beijing, 100027, No. China  
Tel: 86-10-85282100 Fax: 86-10-85282104

#### China - Chengdu

Microchip Technology Consulting (Shanghai)  
Co., Ltd., Chengdu Liaison Office  
Rm. 2401, 24th Floor,  
Ming Xing Financial Tower  
No. 88 TIDU Street  
Chengdu 610016, China  
Tel: 86-28-86766200 Fax: 86-28-86766599

#### China - Fuzhou

Microchip Technology Consulting (Shanghai)  
Co., Ltd., Fuzhou Liaison Office  
Unit 28F, World Trade Plaza  
No. 71 Wusi Road  
Fuzhou 350001, China  
Tel: 86-591-7503506 Fax: 86-591-7503521

#### China - Shanghai

Microchip Technology Consulting (Shanghai)  
Co., Ltd.  
Room 701, Bldg. B  
Far East International Plaza  
No. 317 Xian Xia Road  
Shanghai, 200051  
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

#### China - Shenzhen

Microchip Technology Consulting (Shanghai)  
Co., Ltd., Shenzhen Liaison Office  
Rm. 1315, 13/F, Shenzhen Kerry Centre,  
Renminnan Lu  
Shenzhen 518001, China  
Tel: 86-755-82350361 Fax: 86-755-82366086

#### China - Hong Kong SAR

Microchip Technology Hongkong Ltd.  
Unit 901-6, Tower 2, Metroplaza  
223 Hing Fong Road  
Kwai Fong, N.T., Hong Kong  
Tel: 852-2401-1200 Fax: 852-2401-3431

#### India

Microchip Technology Inc.  
India Liaison Office  
Divyasree Chambers  
1 Floor, Wing A (A3/A4)  
No. 11, O'Shaugnessey Road  
Bangalore, 560 025, India  
Tel: 91-80-2290061 Fax: 91-80-2290062

### Japan

Microchip Technology Japan K.K.  
Benex S-1 6F  
3-18-20, Shinyokohama  
Kohoku-Ku, Yokohama-shi  
Kanagawa, 222-0033, Japan  
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

### Korea

Microchip Technology Korea  
168-1, Youngbo Bldg. 3 Floor  
Samsung-Dong, Kangnam-Ku  
Seoul, Korea 135-882  
Tel: 82-2-554-7200 Fax: 82-2-558-5934

### Singapore

Microchip Technology Singapore Pte Ltd.  
200 Middle Road  
#07-02 Prime Centre  
Singapore, 188980  
Tel: 65-6334-8870 Fax: 65-6334-8850

### Taiwan

Microchip Technology (Barbados) Inc.,  
Taiwan Branch  
11F-3, No. 207  
Tung Hua North Road  
Taipei, 105, Taiwan  
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

### EUROPE

#### Austria

Microchip Technology Austria GmbH  
Durisolstrasse 2  
A-4600 Wels  
Austria  
Tel: 43-7242-2244-399  
Fax: 43-7242-2244-393

#### Denmark

Microchip Technology Nordic ApS  
Regus Business Centre  
Lautrup høj 1-3  
Ballerup DK-2750 Denmark  
Tel: 45 4420 9895 Fax: 45 4420 9910

#### France

Microchip Technology SARL  
Parc d'Activite du Moulin de Massy  
43 Rue du Saule Trapu  
Batiment A - 1er Etage  
91300 Massy, France  
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

#### Germany

Microchip Technology GmbH  
Steinheilstrasse 10  
D-85737 Ismaning, Germany  
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

#### Italy

Microchip Technology SRL  
Centro Direzionale Colleoni  
Palazzo Taurus 1 V. Le Colleoni 1  
20041 Agrate Brianza  
Milan, Italy  
Tel: 39-039-65791-1 Fax: 39-039-6899883

#### United Kingdom

Microchip Ltd.  
505 Eskdale Road  
Winnersh Triangle  
Wokingham  
Berkshire, England RG41 5TU  
Tel: 44 118 921 5869 Fax: 44-118 921-5820

10/18/02