MPLAB®
IDE, SIMULATOR, EDITOR
USER'S GUIDE

Table of Contents

General Information

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 Website
- 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:

- **Chapter 1: MPLAB IDE Preview** – An overview of what MPLAB IDE is and how it works.
- **Chapter 2: MPLAB IDE Installation** – How to install MPLAB IDE on your computer.
- **Chapter 3: Getting Started with MPLAB IDE – A Tutorial** – How to begin using MPLAB IDE.
- **Chapter 4: MPLAB IDE Projects Tutorial** – A tutorial on how to use MPLAB IDE projects.
- **Chapter 5: MPLAB Editor** – A discussion of the basic MPLAB Editor functions and features.
- **Chapter 6: Debugging and MPLAB-SIM** – A discussion of MPLAB IDE debugging functions and related MPLAB-SIM simulator considerations.
- **Chapter 7: 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.

- **Appendix A: MPLAB IDE Key Mapping Functions** – Lists the available MPLAB IDE key mapping functions.
- **Appendix B: Customizing MPLAB IDE After Installation** – Discusses the user configurable customizations available by modifying MPLAB IDE configuration file (MPLAB.ini).
- **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.
- **Appendix D: 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 E: MPLAB Editor Default Key Commands** – Describes the default key commands specific to the MPLAB Editor and lists the equivalent menu command (if any).
- **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
		`#define START`
		`c:\autoexec.bat`

Angle brackets: < >	Variables	`<label>, <exp>`
Square brackets [ ]	Optional arguments	`MPASMWIN [main.asm]`
Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments An OR selection	`errorlevel {0|1}`
Lower case characters in quotes	Type of data	`“filename”`
Ellipses...	Used to imply (but not show) additional text that is not relevant to the example	`list [“list_option..., “list_option”]`
0xnnn	A hexadecimal number where n is a hexadecimal digit	`0xFFFF, 0x007A`

Table: Documentation Conventions (Continued)

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

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 met 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.

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 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 directory for other README files. (In the case of MPASM, 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 website (http://www.microchip.com).

Microsoft Windows Manuals
This manual assumes that users are familiar with Microsoft Windows operating system. Many excellent references exist for this software program, 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® Communicator or Microsoft® Internet Explorer®. Files are also available for FTP download from our FTP site.

Connecting to the Microchip Internet Website

The Microchip website is available by using your favorite Internet browser to attach to:

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 is:

- 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
- Conferences for products, Development Systems, 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 to one of our list servers, you will receive email notification whenever we change, update, revise or have errata related to that product family or development tool. See the Microchip web site at www.microchip.com.

The Development Systems list names are:

- Compilers
- Emulators
- Programmers
- MPLAB
- Otools (Other Tools)

Once you have determined the names of the lists that you are interested in, you can subscribe by sending a message to:

`listserv@mail.microchip.com`

with the following as the body:

`subscribe <listname> yourname.` Here is an example:

`subscribe mplab John Doe .` To UNSUBSCRIBE from these lists, send a message to:

`listserv@mail.microchip.com`

with the following as the body:

`unsubscribe <listname> yourname`

Here is an example:

`unsubscribe mplab John Doe`

The following sections provide descriptions of the available Development Systems lists.

Compilers

The latest information on Microchip C compilers, Linkers and Assemblers. These include MPLAB-C17, MPLAB-C18, MPLINK, MPASM as well as the Librarian, MPLIB for MPLINK.

To SUBSCRIBE to this list, send a message to:

`listserv@mail.microchip.com`

with the following as the body:

`subscribe compilers yourname`

Emulators

The latest information on Microchip In-Circuit Emulators. These include MPLAB-ICE and PICMASTER® emulator.

To SUBSCRIBE to this list, send a message to:

`listserv@mail.microchip.com`

with the following as the body:

`subscribe emulators yourname`

Programmers

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

To SUBSCRIBE to this list, send a message to:

`listserv@mail.microchip.com`

with the following as the body:

`subscribe programmers yourname`

MPLAB

The latest information on Microchip MPLAB IDE, the Windows Integrated Development Environment for development systems tools. This list is focused on MPLAB IDE, MPLAB-SIM, 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.

To SUBSCRIBE to this list, send a message to:

`listserv@mail.microchip.com`

with the following as the body:

`subscribe mplab yourname`

Otools (Other Tools)

The latest information on other development system tools provided by Microchip. For specific information on MPLAB IDE and its integrated tools refer to the other mail lists.

To SUBSCRIBE to this list, send a message to:

`listserv@mail.microchip.com`

with the following as the body:

subscribe otools yourname

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) 786-7627.

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

1-480-786-7302 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®-based Integrated Development Environment (IDE) for the Microchip Technology Incorporated PICmicro® microcontroller (MCU) families. MPLAB IDE allows you to write, debug, and optimize PICmicro MUC applications for firmware product designs. MPLAB IDE includes a text editor, simulator, and project manager. MPLAB IDE also supports the MPLAB-ICE and PICMASTER® emulators, PICSTART® Plus and PRO MATE® 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 helps make 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 emulator
- Make timing measurements
- View variables in watch windows
- Program firmware with PICSTART Plus or PRO MATE II
- 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. The MPLAB IDE runs under Microsoft Windows 3.1x, Windows 95/98, Windows NT, or Windows 2000.

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

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)

The 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.

The 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

The 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 the FLASH PIC16F87X MCU family.

- **MPLAB-SIM Simulator**

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

- **MPLAB-ICE Emulator**

  The MPLAB-ICE emulator uses hardware to emulate PICmicro MCUs in real time, 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. MPLINK creates the final application by linking relocatable modules from MPASM, MPLAB-C17, and MPLAB-C18. MPLIB 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 PRO MATE II does not require MPLAB IDE to operate, programming is easier using MPLAB IDE.

- **PICMASTER and PICMASTER-CE Emulators**

  These emulators use hardware to emulate PICmicro MCUs in real time, either with or without a target system. MPLAB-ICE is the newest emulator from Microchip.

- **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 class system
- Microsoft Windows 3.1x, Windows 95/98, Windows NT, or Windows 2000
- 16 MB memory (32 MB recommended)
- 45 MB of hard disk space

Note: Not all hardware components that function under the MPLAB IDE, such as emulators and device programmer, function under all 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).

The number and names of the files vary depending on the version. Version 4.00 of MPLAB IDE, for example, would have these files:

```
MP40000.EXE
MP40000.W02
MP40000.W03
MP40000.W04
MP40000.W05
MP40000.W06
```

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:\MP41219.exe** to install version 4.12.19 of MPLAB IDE, where d: is the CD-ROM drive that contains the MPLAB IDE install.

   Windows 95/98, WIndows NT, or Windows 2000: 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:\MP41219.exe** to install version 4.12.19 of MPLAB IDE, where d: is the CD-ROM drive that contains the MPLAB IDE install. Then click **OK**.

Note: Windows NT users must have administrative privileges in order to install MPLAB-ICE.

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/MPLINK/MPLIB files

   - MPLAB-SIM Simulator Support Files

   - Help Files

   You can reinstall MPLAB IDE later to add additional components.

   Selecting the Destination Directory

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

Selecting the System Files Directory

Installing the data link libraries (DLLs) to the `\Windows\System` directory 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, you may view these files later from the `MPLAB` install directory. It is recommended that you consult the readme files before contacting technical support.

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

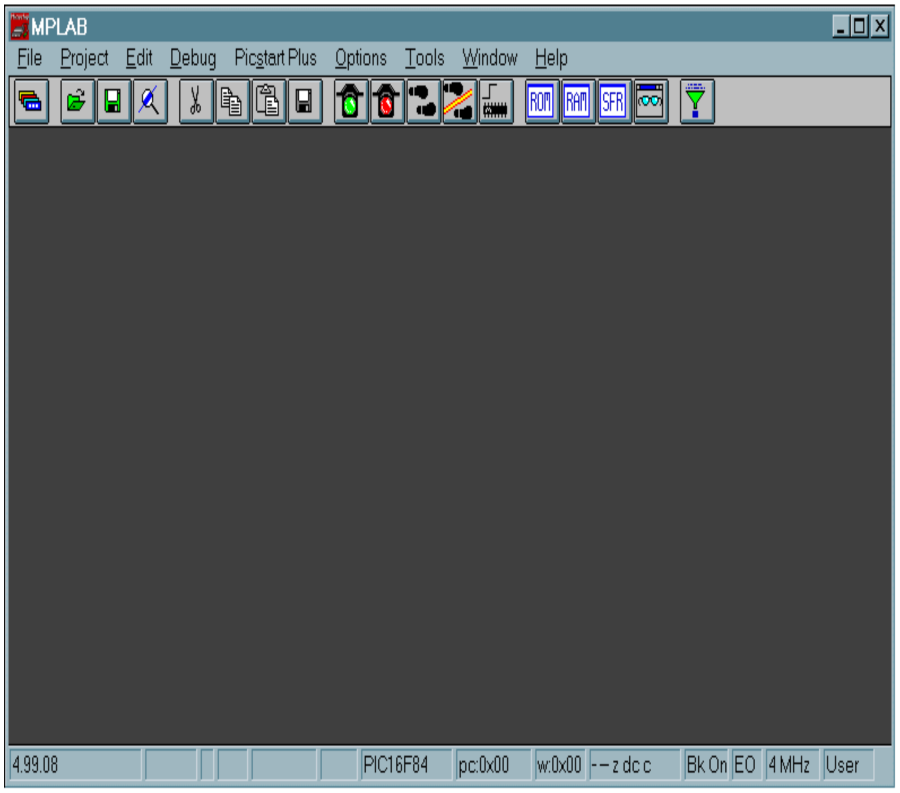


Figure 2.1: MPLAB IDE Desktop

2.6 Uninstalling MPLAB IDE

To uninstall MPLAB IDE, open Windows Explorer (or File Manager for Windows 3.1) and double-click on `unwise.exe` 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 directories.

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

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.

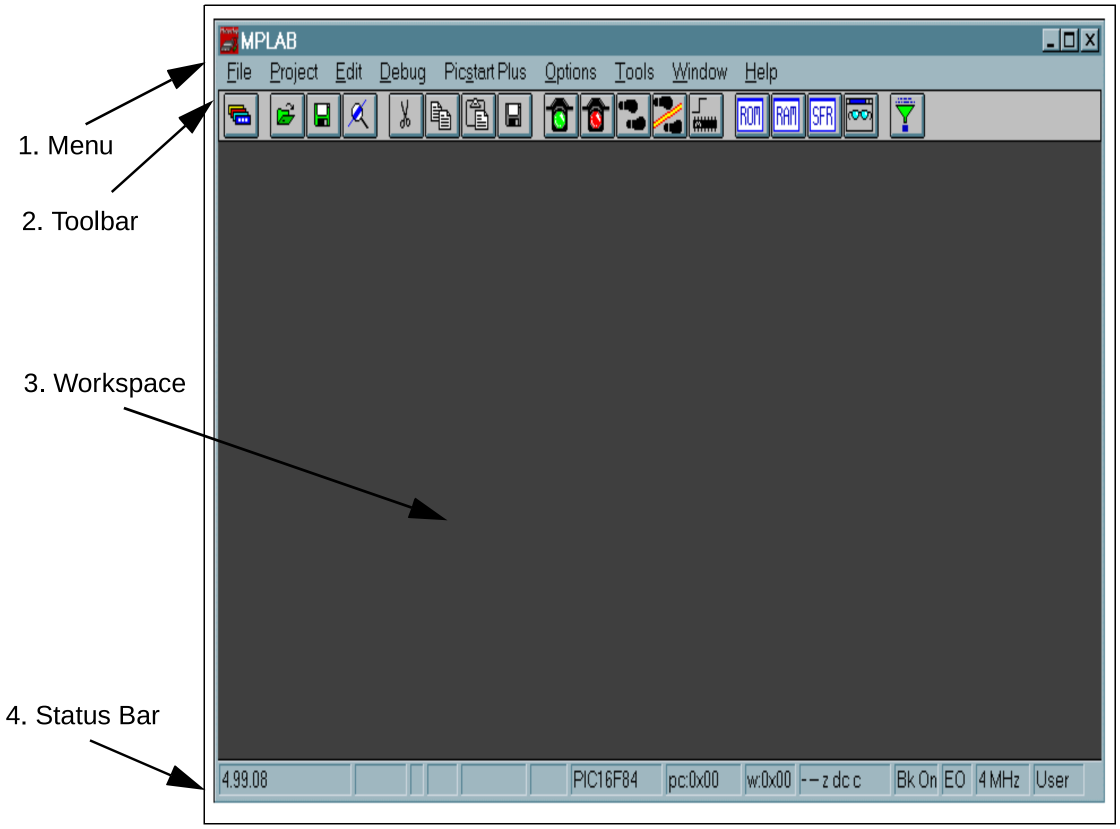
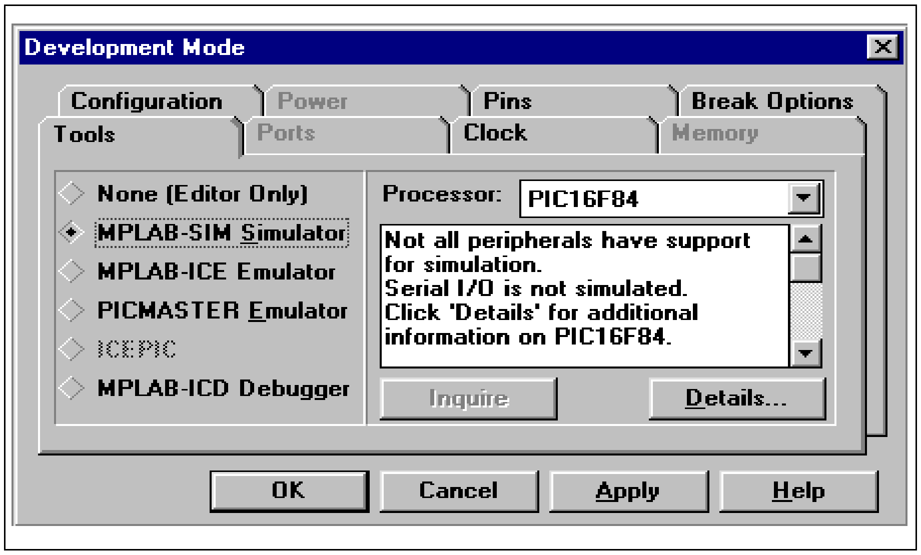


Figure 3.1: MPLAB IDE Desktop

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.



Processor: PIC16F84

Not all peripherals have support for simulation.
Serial I/O is not simulated.
Click 'Details' for additional information on PIC16F84.

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.

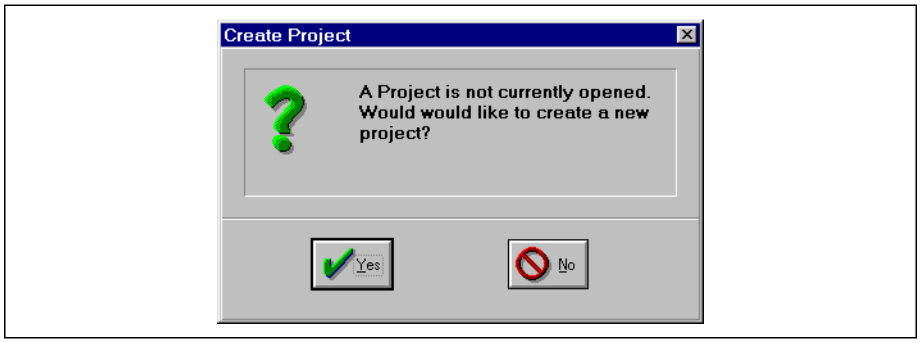

Create Project
A Project is not currently opened. Would would like to create a new project?
Yes
No

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 directory 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.

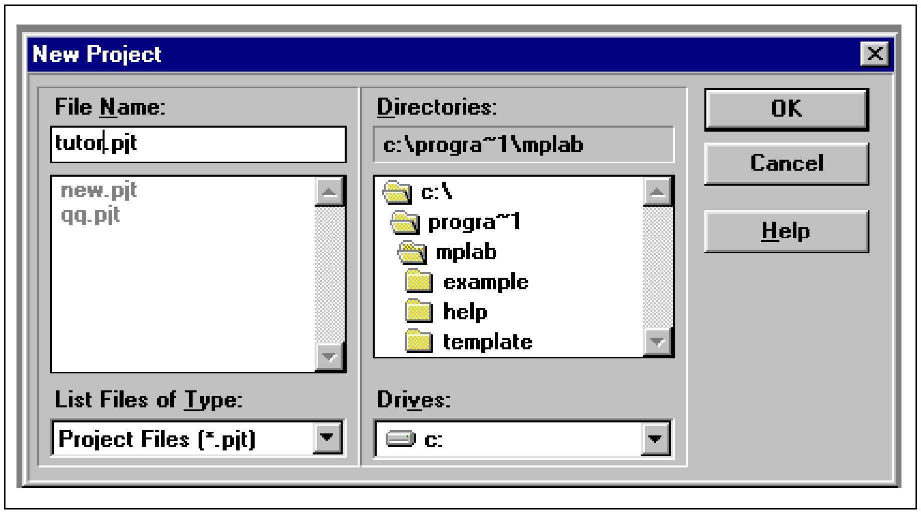
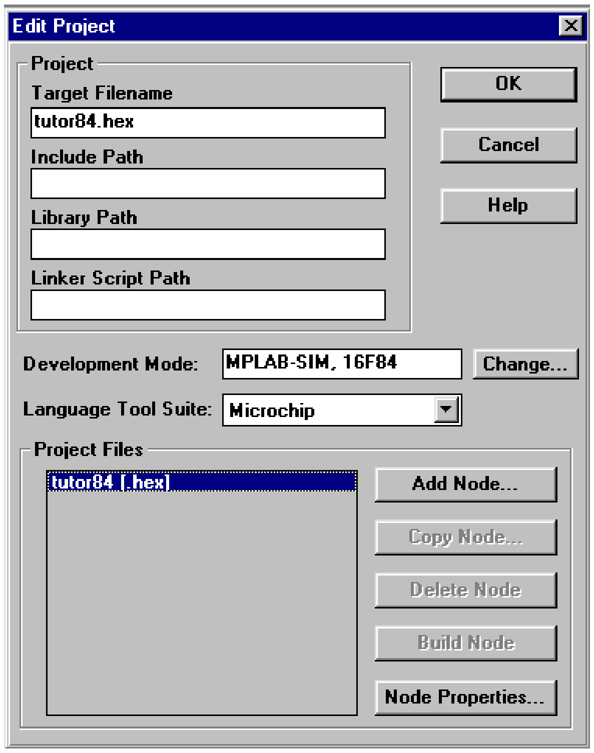


Figure 3.4: New Project Dialog

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



Target Filename: tutor84.hex
Include Path:
Library Path:
Linker Script Path:
Development Mode: MPLAB-SIM, 16F84
Language Tool Suite: Microchip
tutor84 [.hex]

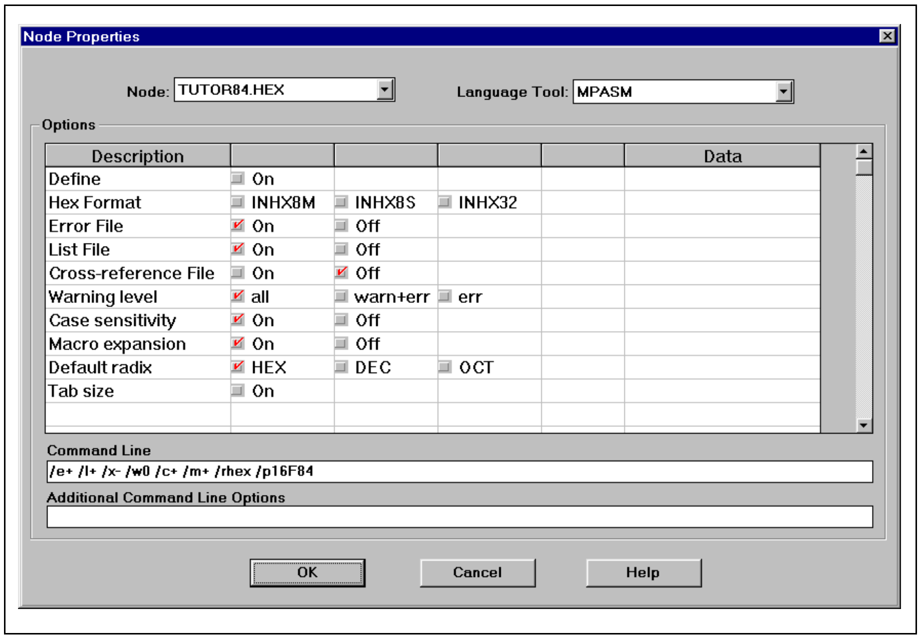
Figure 3.5: Edit Project Dialog – Node Properties Enabled

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 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 we do 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).


Node Properties

Node: TUTOR84.HEX   Language Tool: MPASM

Options

Description					Data
Define	On				
Hex Format	INHX8M	INHX8S	INHX32		
Error File	[✓ On]	Off			
List File	[✓ On]	Off			
Cross-reference File	On	[✓ Off]			
Warning level	[✓ all]	warn+err	err		
Case sensitivity	[✓ On]	Off			
Macro expansion	[✓ On]	Off			
Default radix	[✓ HEX]	DEC	OCT		
Tab size	On				

Command Line

```
/e+ /l+ /x- /w0 /c+ /m+ /rhex /p16F84
```

Additional Command Line Options

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 MPASM 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.

Clicking **OK** will apply these defaults, bring you back to the Edit Project dialog and enable the **Add Node** button (Figure 3.7).

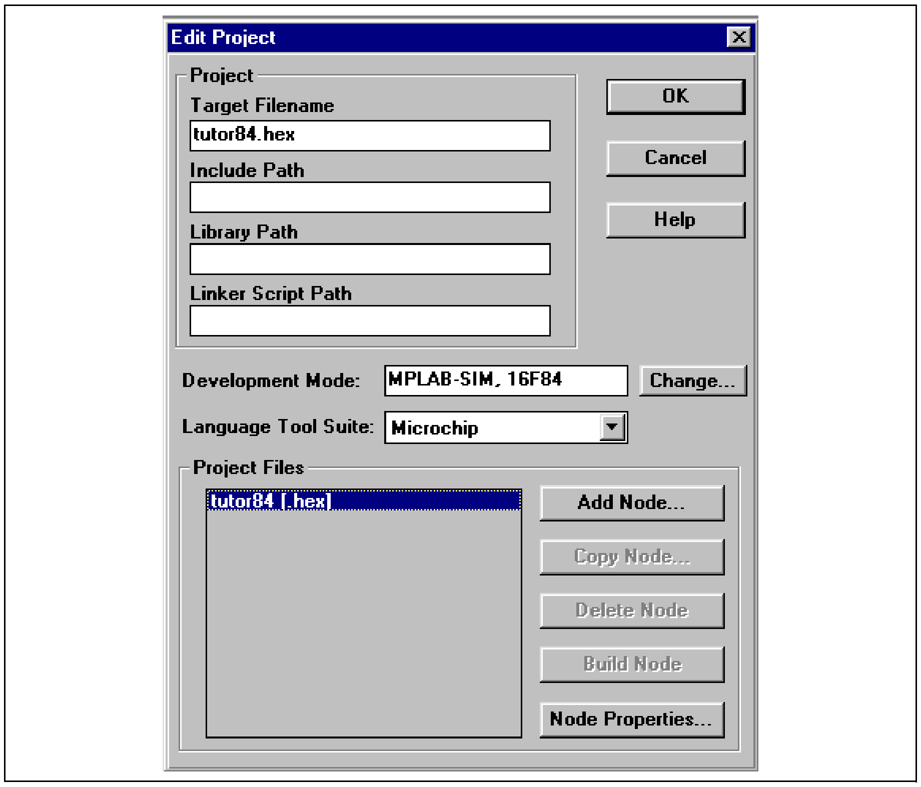


Figure 3.7: Edit Project Dialog – Add Node Enabled

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

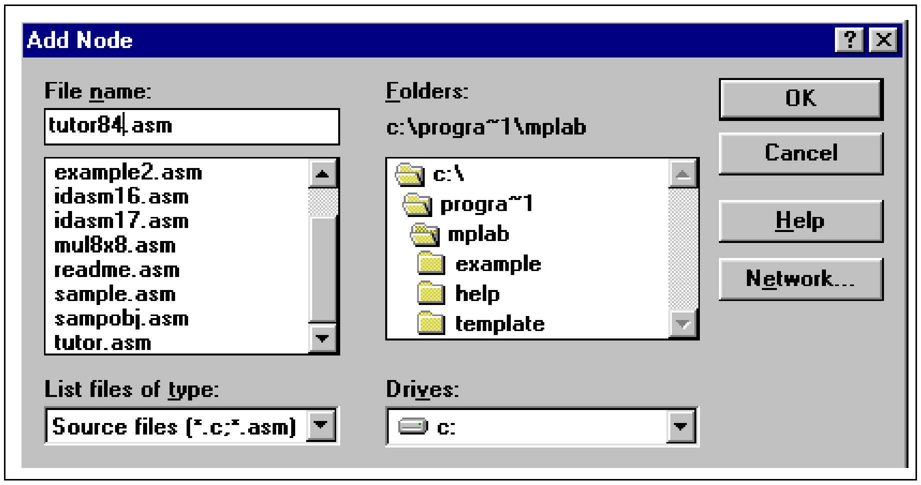
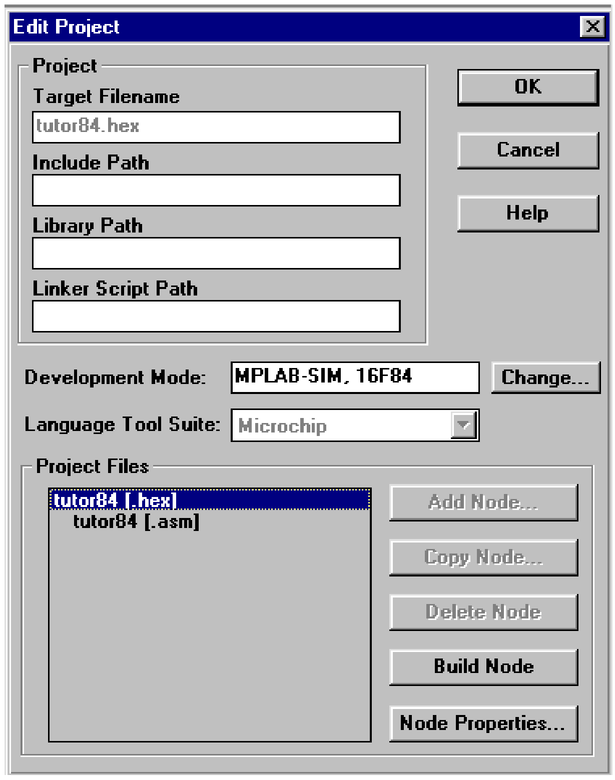


Figure 3.8: Add Node Dialog

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).



Target Filename: tutor84.hex
Include Path:
Library Path:
Linker Script Path:
Development Mode: MPLAB-SIM, 16F84
Language Tool Suite: Microchip

tutor84 [.hex]
  tutor84 [.asm]

Figure 3.9: Edit Project Dialog – Node Added

Clicking **OK** takes you back to the MPLAB IDE desktop with an open and as yet 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 directory of the project. Enter the file name and click **OK**.

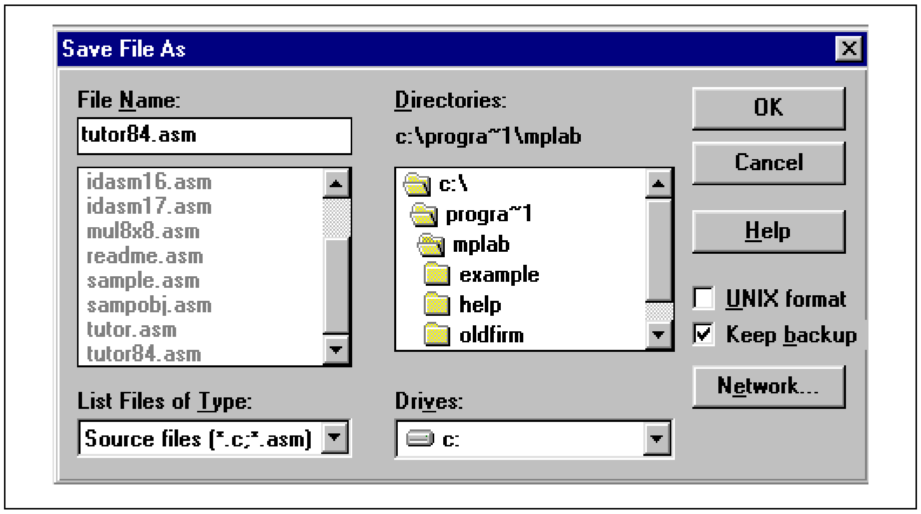


Figure 3.10: Save Source File

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, MPASM 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
```

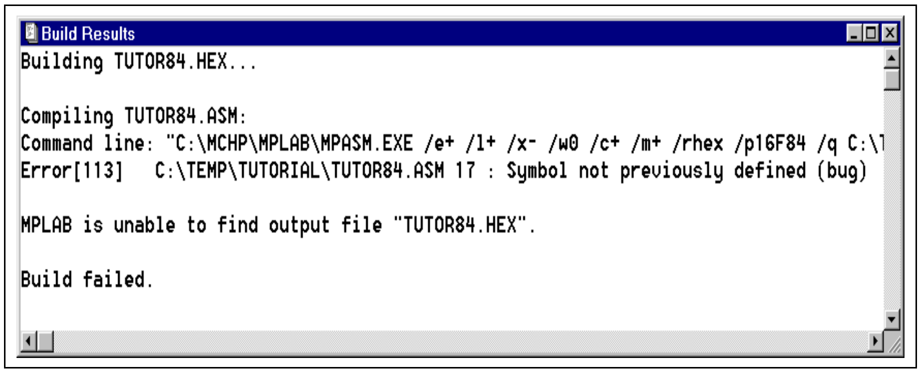
This code is a very simple program that increments a counter and resets to a predetermined value when the counter rolls over to zero.

All labels start in the first column, and the last line has an `end` directive. Refer to the *MPASM with MPLINK and MPLIB User's Guide* 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).



```
Building TUTOR84.HEX...

Compiling TUTOR84.ASM:
Command line: "C:\MCHP\MPLAB\MPASM.EXE /e+ /l+ /x- /w0 /c+ /m+ /rhex /p16F84 /q C:\T
Error[113]   C:\TEMP\TUTORIAL\TUTOR84.ASM 17 : Symbol not previously defined (bug)

MPLAB is unable to find output file "TUTOR84.HEX".

Build failed.
```

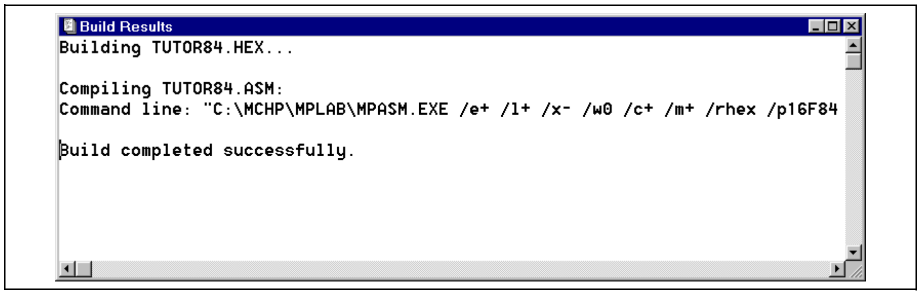
Figure 3.11: Build Results Window – Build Failed

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.



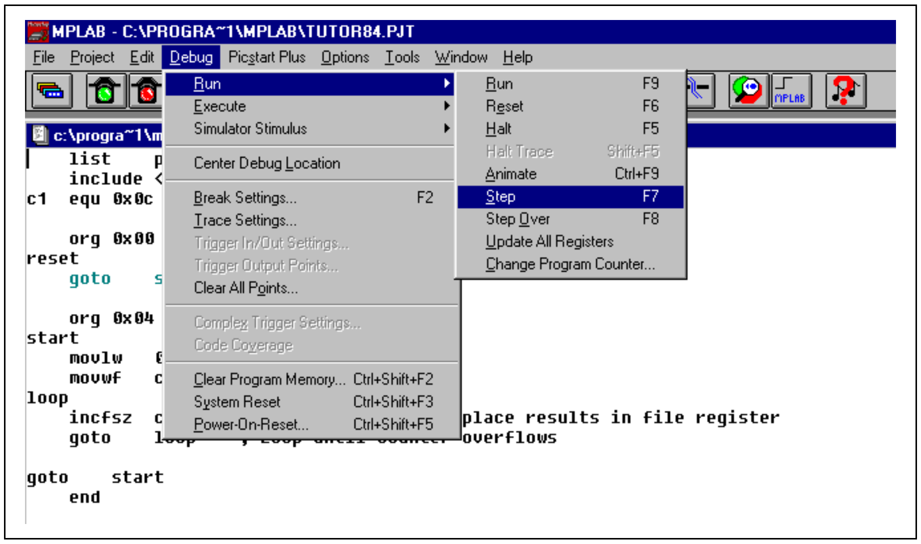
```
Building TUTOR84.HEX...

Compiling TUTOR84.ASM:
Command line: "C:\MCHP\MPLAB\MPASM.EXE /e+ /l+ /x- /w0 /c+ /m+ /rhex /p16F84

Build completed successfully.
```

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."


MPLAB - C:\PROGRA~1\MPLAB\TUTOR84.PJT
File Project Edit Debug Picstart Plus Options Tools Window Help
c:\progra~1\m
Run
Execute
Simulator Stimulus
Center Debug Location
Break Settings... F2
Trace Settings...
Trigger In/Out Settings...
Trigger Output Points...
Clear All Points...
Complex Trigger Settings...
Code Coverage
Clear Program Memory... Ctrl+Shift+F2
System Reset Ctrl+Shift+F3
Power-On-Reset... Ctrl+Shift+F5
Run F9
Reset F6
Halt F5
Halt Trace Shift+F5
Animate Ctrl+F9
Step F7
Step Over F8
Update All Registers
Change Program Counter...

```
    list     p
    include <
c1  equ  0x0c

    org  0x00
reset
    goto     s

    org  0x04
start
    movlw    0
    movwf    c
loop
    incfsz   c                               place results in file register
    goto     loop    ; Loop until counter overflows

goto    start
    end
```

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 affect 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 tool bar at the top of the screen. If you place the cursor over the items in the tool bar, you can see the name of the function in the status bar at the bottom. The left button is a standard **Change Tool bar** button that allows you to scroll through the available toolbars. Toolbars can be customized (see Section 7.8.5.1.4). 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).

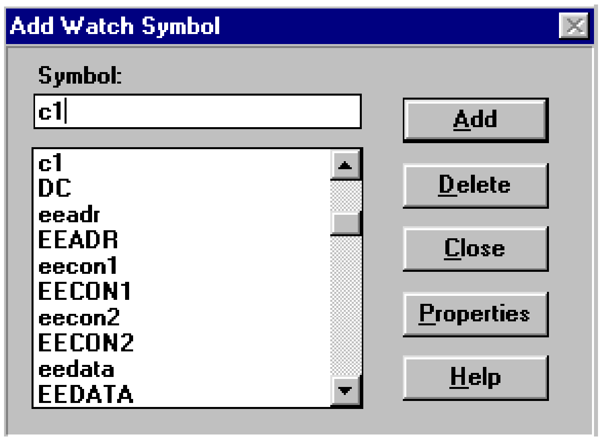
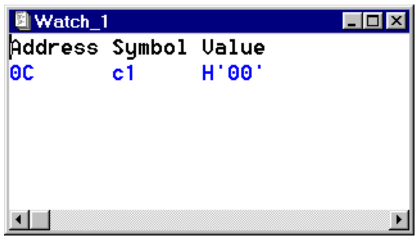


Figure 3.14: Add Watch Symbol Dialog

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.



```
Address Symbol Value
0C      c1     H'00'
```

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**.

.

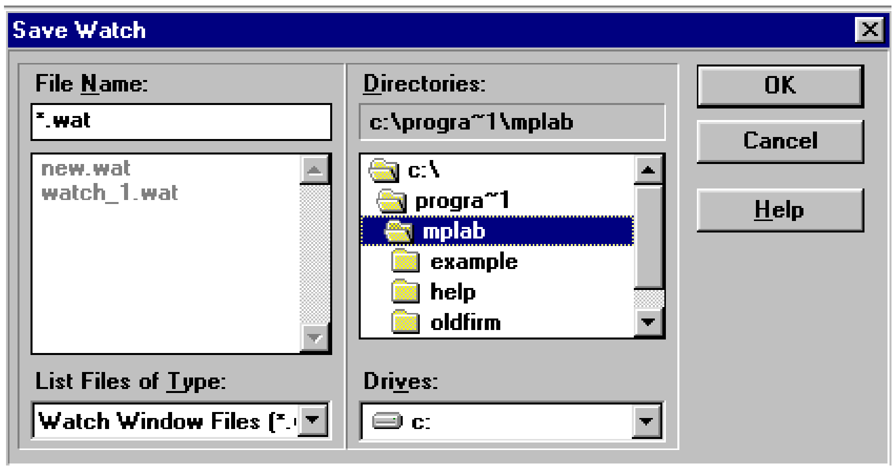


Figure 3.16: Save Watch Window Dialog

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.

Add a symbol to the Watch window	Select *Window > Watch Window > Add to Active Watch* from the MPLAB IDE menu or select **Add Watch** from the system menu inside the Watch window.
Delete a symbol from the Watch window	Click on the symbol in the Watch window, then select *Delete Watch* from the system menu.
Change the display format of symbols	Select *Window > Watch Window > Edit Active Watch* from the MPLAB IDE menu or select *Edit Watch* from the system menu inside the Watch window. Then, click **Properties**. 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).

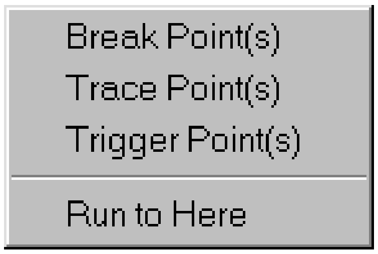


Figure 3.17: Right Mouse Button Pop-up Menu

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.

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 MPASM 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 Watch Dog Timer Enable configuration bit can be set so that when you program a device, the Watch Dog 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 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 tool bar on the side or bottom of the screen
- Modify the tool bar
- 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.

NOTES:

Chapter 4. MPLAB IDE 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 MPLAB IDE Projects
- Making a Project with One MPASM Source File
- Compiling a Single MPASM Source File Without Creating a Project
- Making a Project with Multiple MPASM Source Files using MPLINK
- Making a Project with Hi-Tech PIC C
- Making a Project with MPLAB-C17 or MPLAB-C18

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 MPLAB IDE 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 directory as the main source files.

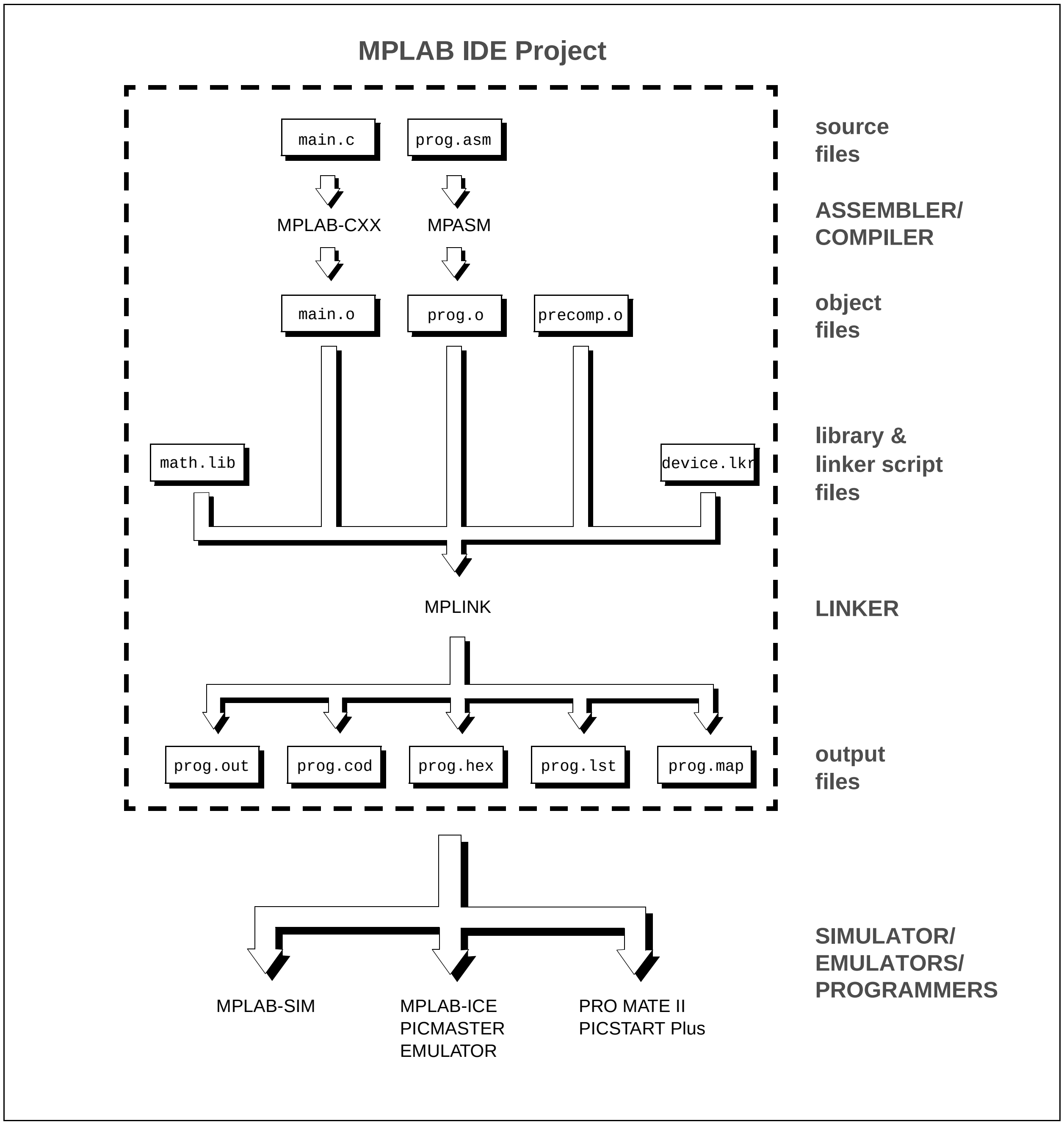


Figure 4.1: Project Relationships

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 User’s Guide* (DSXXXXX) 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 MPLINK. 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 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* for more information on using the librarian. For more information on available Microchip libraries, see the *MPLAB-CXX Reference Guide*.

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* for more information on linker script files and using the linker.

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

- **COFF file (.out).** Intermediate file used by MPLINK to generate Code file, Hex file, and Listing file.
- **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 Source File

To make a project that has only one MPASM 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.

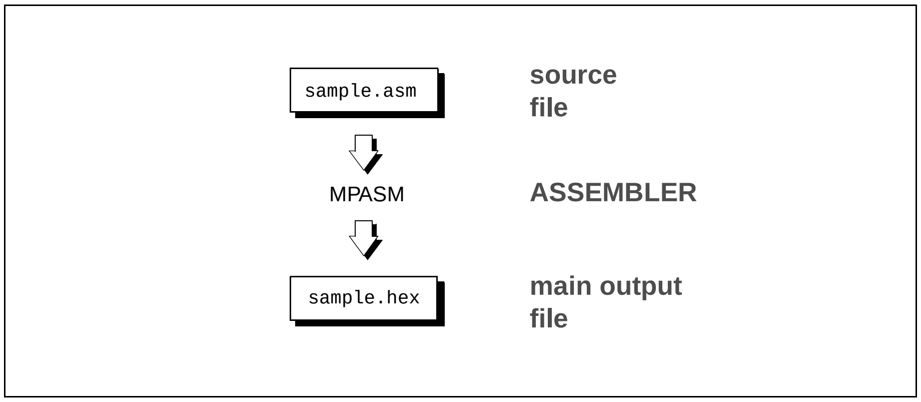
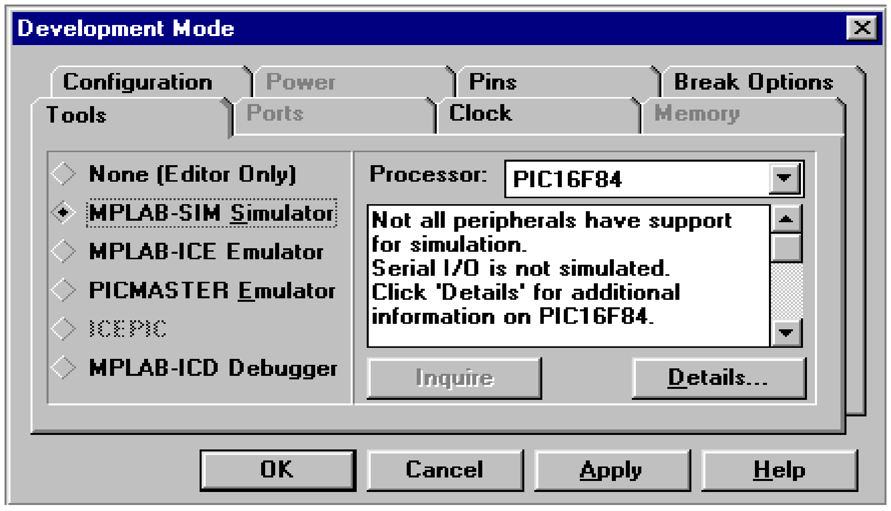


Figure 4.2: Project Relationships For One MPASM Source File

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**.


Development Mode

Configuration | Power | Pins | Break Options | Tools | Ports | Clock | Memory

None (Editor Only) [✓ MPLAB-SIM Simulator] MPLAB-ICE Emulator PICMASTER Emulator ICEPIC MPLAB-ICD Debugger

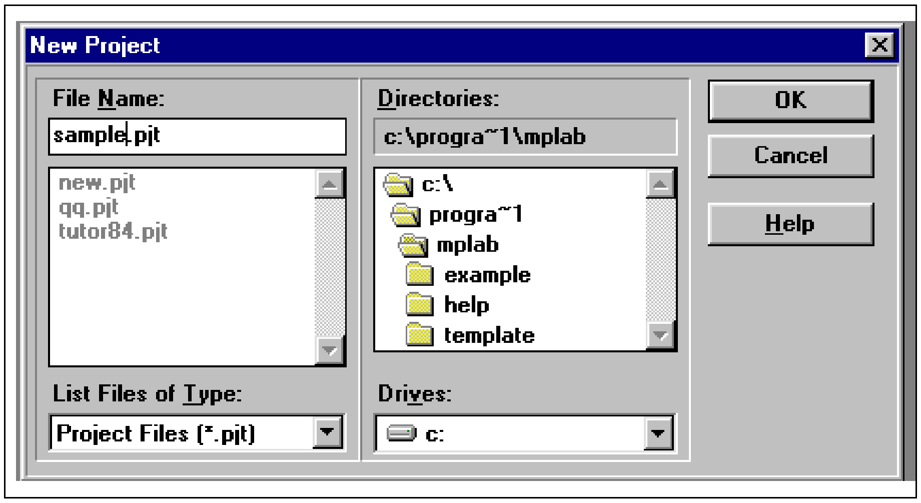
Processor: PIC16F84

Not all peripherals have support for simulation.
Serial I/O is not simulated.
Click 'Details' for additional information on PIC16F84.

Figure 4.3: Development Mode Dialog

4.4.2 New Project

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



File Name: sample.pjt

new.pjt
qq.pjt
tutor84.pjt

Directories: c:\progra~1\mplab

c:\
progra~1
mplab
example
help
template

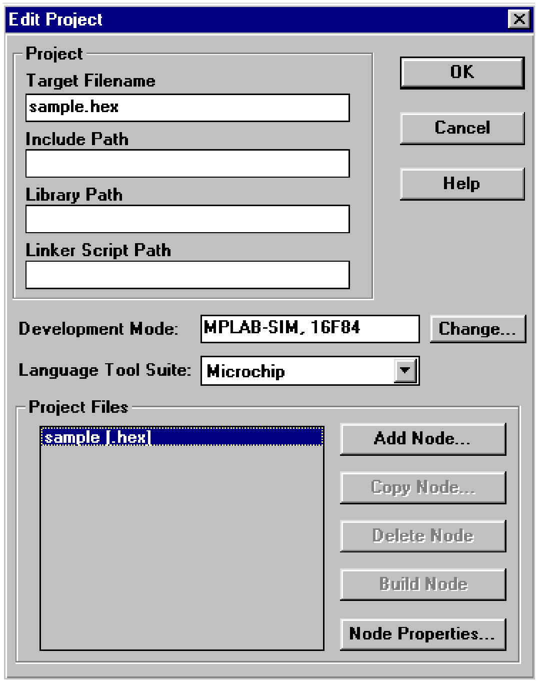
List Files of Type: Project Files (*.pjt)

Drives: c:

Figure 4.4: New Project Dialog – sample.pjt

4.4.3 Project Dialog

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



Edit Project

Project

Target Filename: sample.hex

Include Path:

Library Path:

Linker Script Path:

Development Mode: MPLAB-SIM, 16F84

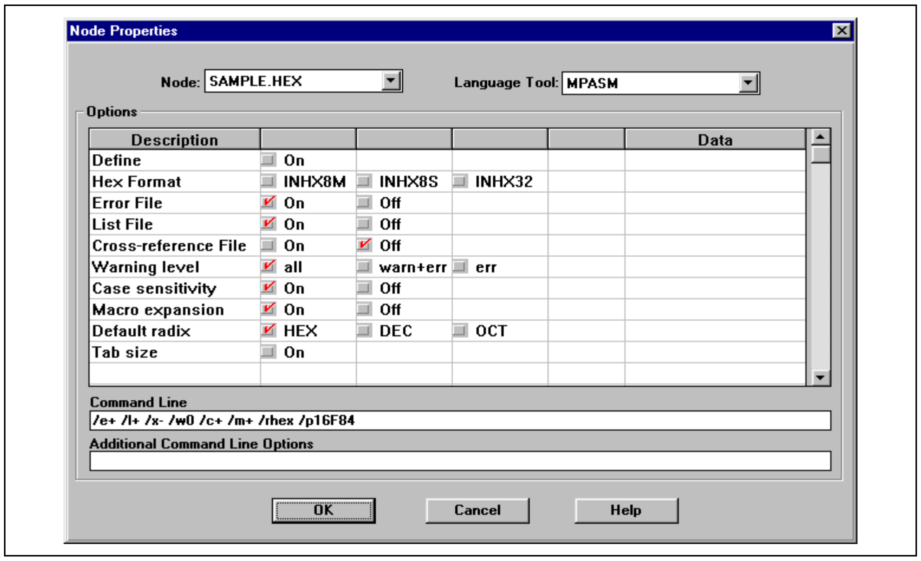
Language Tool Suite: Microchip

Project Files: sample [.hex]

Figure 4.5: Edit Project Dialog

4.4.4 Set Node Properties

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



Node: SAMPLE.HEX   Language Tool: MPASM

Options

Description					Data
Define	On				
Hex Format	INHX8M	INHX8S	INHX32		
Error File	[✓ On]	Off			
List File	[✓ On]	Off			
Cross-reference File	On	[✓ Off]			
Warning level	[✓ all]	warn+err	err		
Case sensitivity	[✓ On]	Off			
Macro expansion	[✓ On]	Off			
Default radix	[✓ HEX]	DEC	OCT		
Tab size	On				

Command Line

```
/e+ /l+ /x- /w0 /c+ /m+ /rhex /p16F84
```

Additional Command Line Options

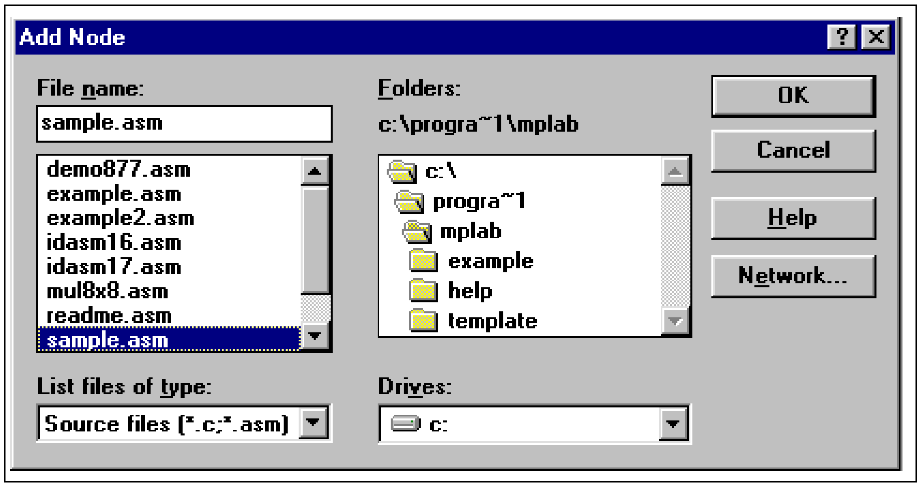
Figure 4.6: Node Properties Dialog

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 with MPLINK and MPLIB User's Guide* 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**.



Add Node

File name: sample.asm

demo877.asm
example.asm
example2.asm
idasm16.asm
idasm17.asm
mul8x8.asm
readme.asm
sample.asm

Folders: c:\progra~1\mplab

c:\
progra~1
mplab
example
help
template

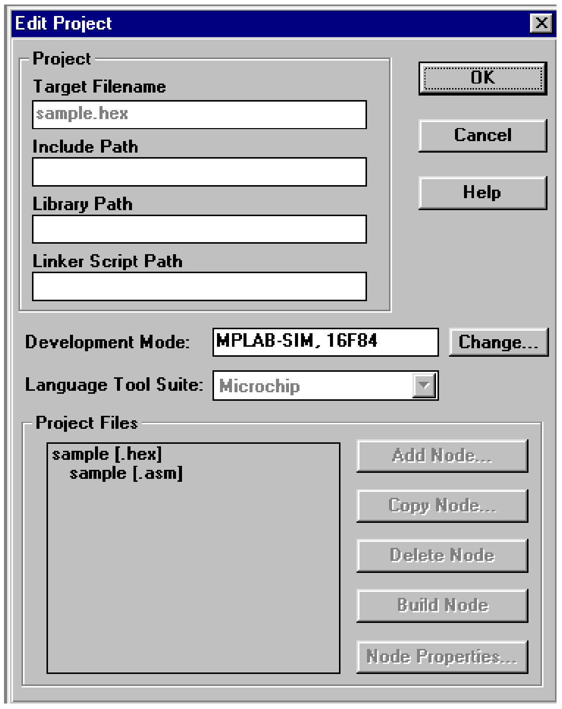
List files of type: Source files (*.c;*.asm)

Drives: c:

Figure 4.7: Add Node Dialog

MPASM 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.

The Edit Project dialog should look like this:



Target Filename: sample.hex
Include Path:
Library Path:
Linker Script Path:
Development Mode: MPLAB-SIM, 16F84
Language Tool Suite: Microchip

```
sample [.hex]
  sample [.asm]
```

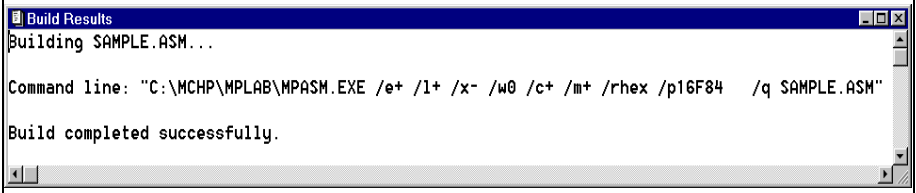
Figure 4.8: Edit Project Dialog with Node

In this simple example, no entries were made in the Path boxes. As your application becomes more complex, you may need to enter the directories 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 MPASM. A Build Results window is created that shows the command line sent to the assembler. It should look like this:



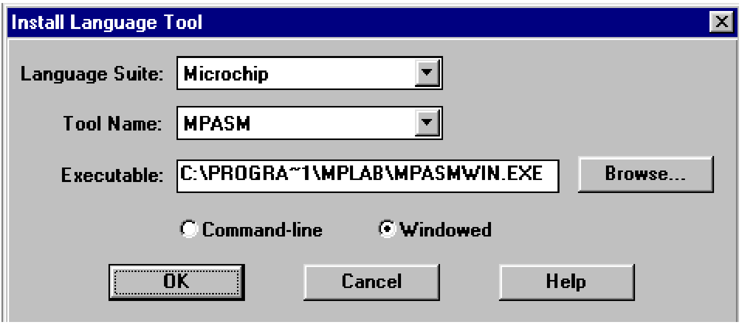
```
Building SAMPLE.ASM...

Command line: "C:\MCHP\MPLAB\MPASM.EXE /e+ /l+ /x- /w0 /c+ /m+ /rhex /p16F84   /q SAMPLE.ASM"

Build completed successfully.
```

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 MPASM is pointed to the `MPASMWIN.EXE` in the MPLAB IDE installation directory. Also, the "Windowed" option should be selected.
   Alternatively, MPASM can point to `MPASM.EXE` and the "Command-line" option selected; however this executable may not operate on Pentium 100MHz PCs and higher.



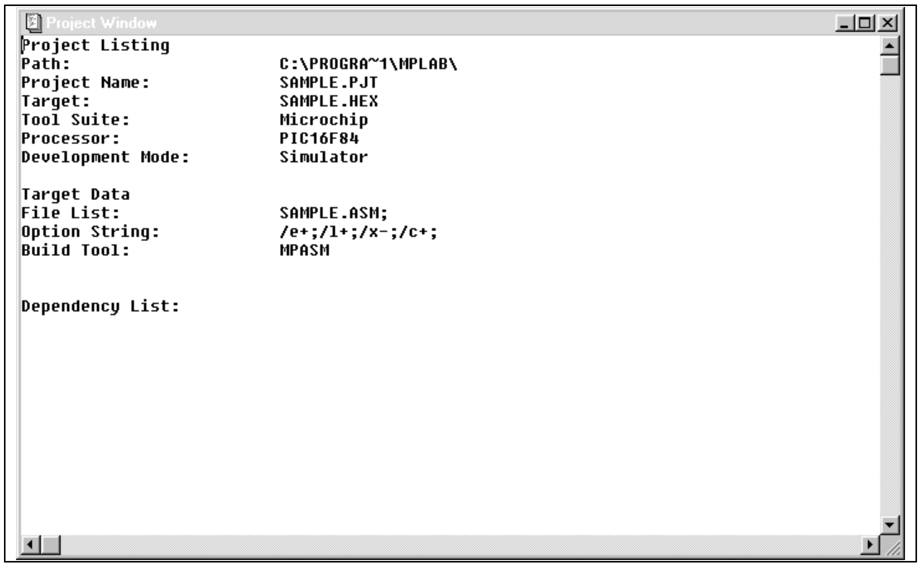
Language Suite: Microchip
Tool Name: MPASM
Executable: C:\PROGRA~1\MPLAB\MPASMWIN.EXE
Command-line [✓ Windowed]

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:



```
Project Listing
Path:                     C:\PROGRA~1\MPLAB\
Project Name:             SAMPLE.PJT
Target:                   SAMPLE.HEX
Tool Suite:               Microchip
Processor:                PIC16F84
Development Mode:         Simulator

Target Data
File List:                SAMPLE.ASM;
Option String:            /e+;/l+;/x-;/c+;
Build Tool:               MPASM


Dependency List:
```

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.

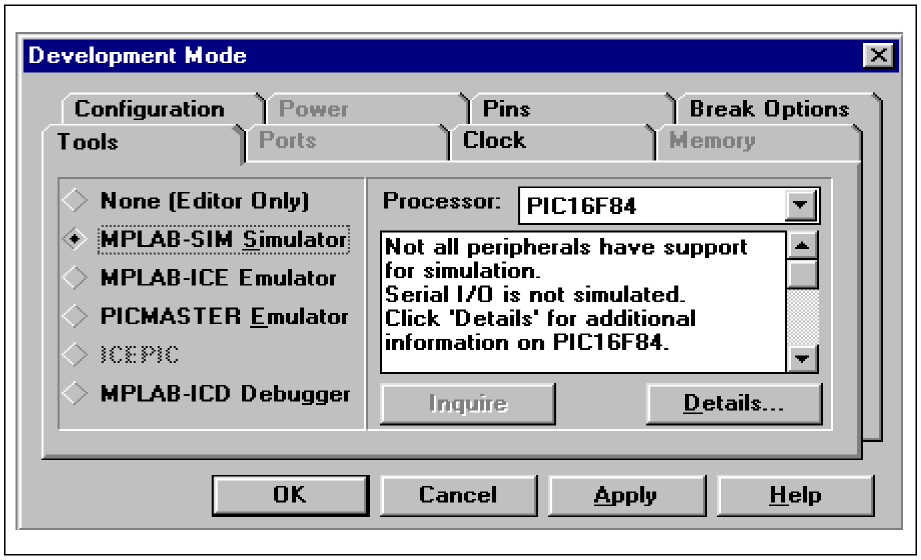
4.5 Compiling a Single MPASM Source File Without Creating a Project

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**.



None (Editor Only)
[✓ MPLAB-SIM Simulator]
MPLAB-ICE Emulator
PICMASTER Emulator
ICEPIC
MPLAB-ICD Debugger

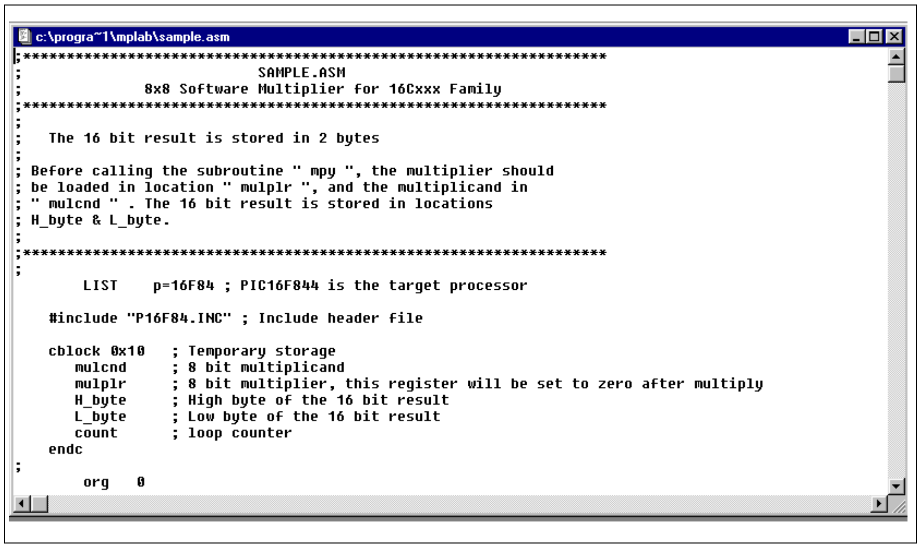
Processor: PIC16F84

Not all peripherals have support for simulation.
Serial I/O is not simulated.
Click 'Details' for additional information on PIC16F84.

Figure 4.12: Development Mode Dialog

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 directory.



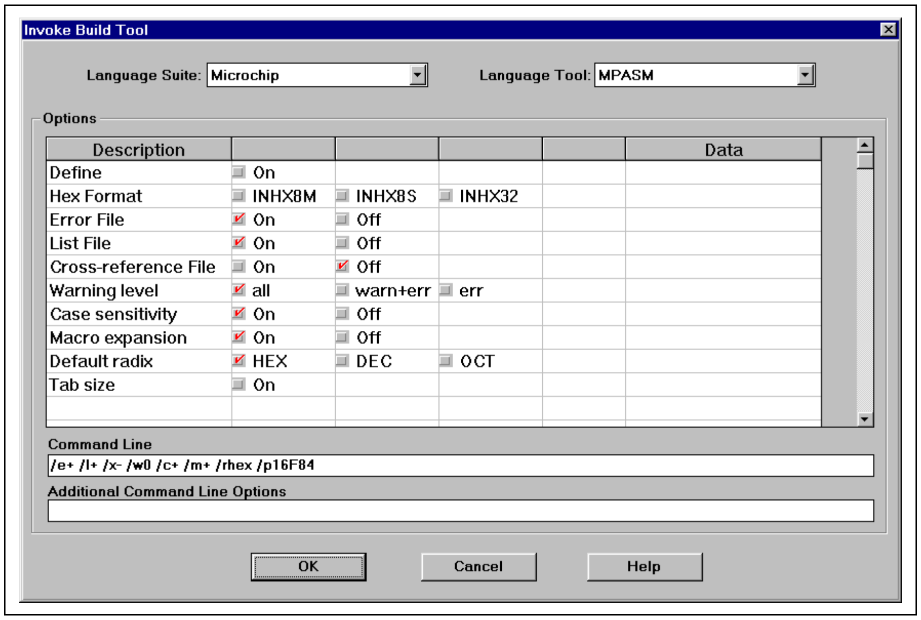
c:\progra~1\mplab\sample.asm

```
;*******************************************************************
;                          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 MPASM. MPLAB IDE opens an Invoke Build Tool Dialog that looks like this:



Invoke Build Tool

Language Suite: Microchip   Language Tool: MPASM

Options

Description					Data
Define	☐ On				
Hex Format	☐ INHX8M	☐ INHX8S	☐ INHX32		
Error File	✓ On	☐ Off			
List File	✓ On	☐ Off			
Cross-reference File	☐ On	✓ Off			
Warning level	✓ all	☐ warn+err	☐ err		
Case sensitivity	✓ On	☐ Off			
Macro expansion	✓ On	☐ Off			
Default radix	✓ HEX	☐ DEC	☐ OCT		
Tab size	☐ On				

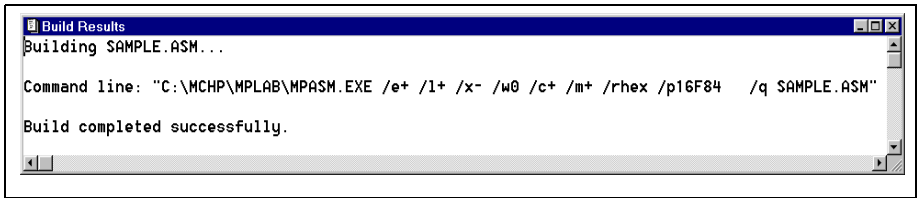
Command Line

```
/e+ /l+ /x- /w0 /c+ /m+ /rhex /p16F84
```

Additional Command Line Options

Figure 4.14: Build Tool Dialog

Verify that MPASM 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:



Build Results

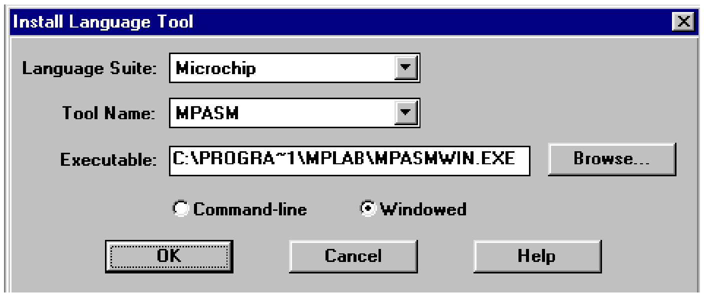
```
Building SAMPLE.ASM...

Command line: "C:\MCHP\MPLAB\MPASM.EXE /e+ /l+ /x- /w0 /c+ /m+ /rhex /p16F84   /q SAMPLE.ASM"

Build completed successfully.
```

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 is pointed to the `MPASMWIN.EXE` in the MPLAB IDE installation directory. Also, the "Windowed" option should be selected.
   Alternatively, MPASM can point to `MPASM.EXE` and the "Command-line" option selected; however this executable may not operate on Pentium 100MHz PCs and higher.



Install Language Tool

Language Suite: Microchip

Tool Name: MPASM

Executable: C:\PROGRA~1\MPLAB\MPASMWIN.EXE

( ) Command-line [✓ Windowed]

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 Source Files using MPLINK

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

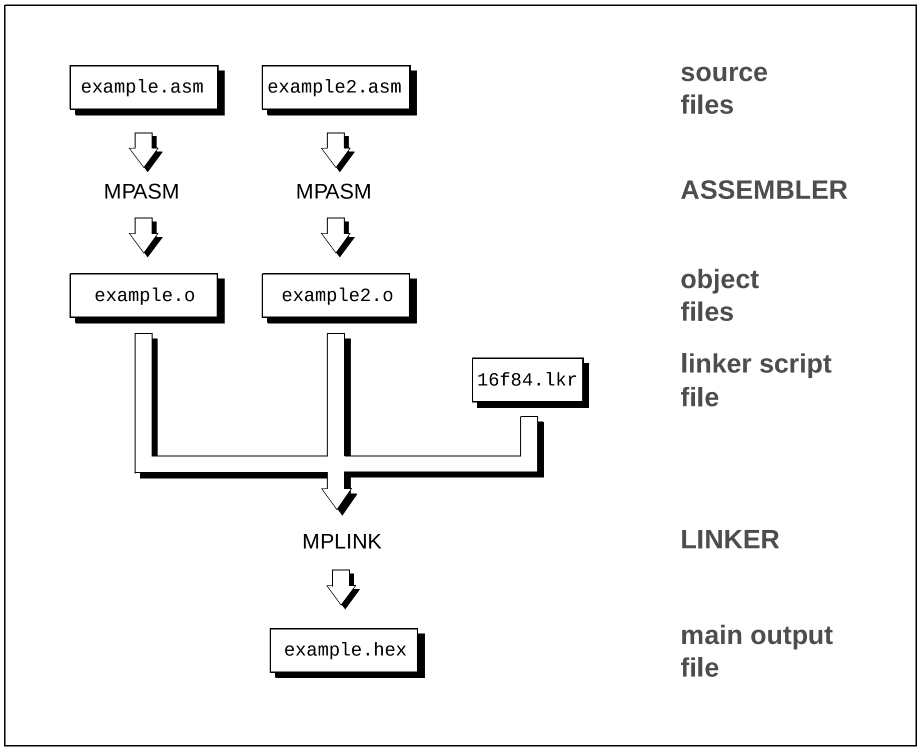
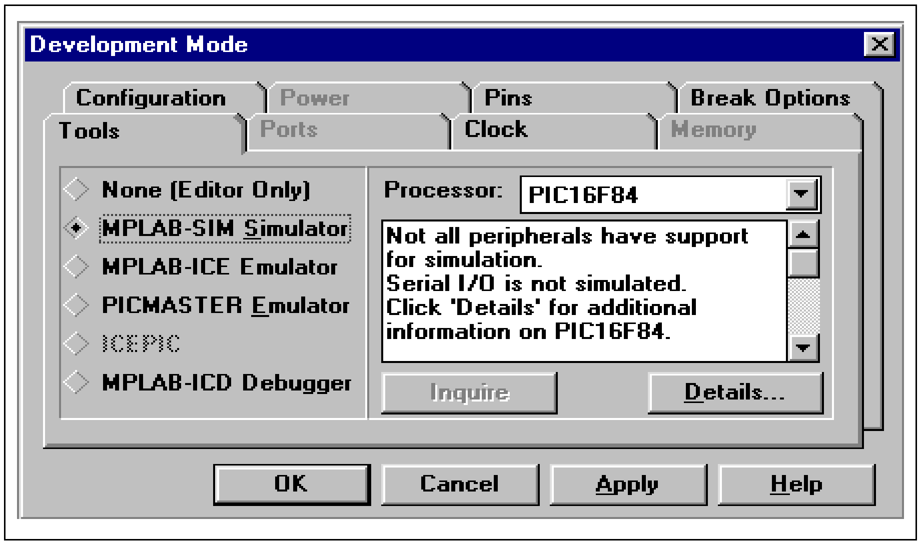


Figure 4.17: Project Relationships For Multiple MPASM Source Files

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**.



None (Editor Only)
[✓ MPLAB-SIM Simulator]
MPLAB-ICE Emulator
PICMASTER Emulator
ICEPIC
MPLAB-ICD Debugger

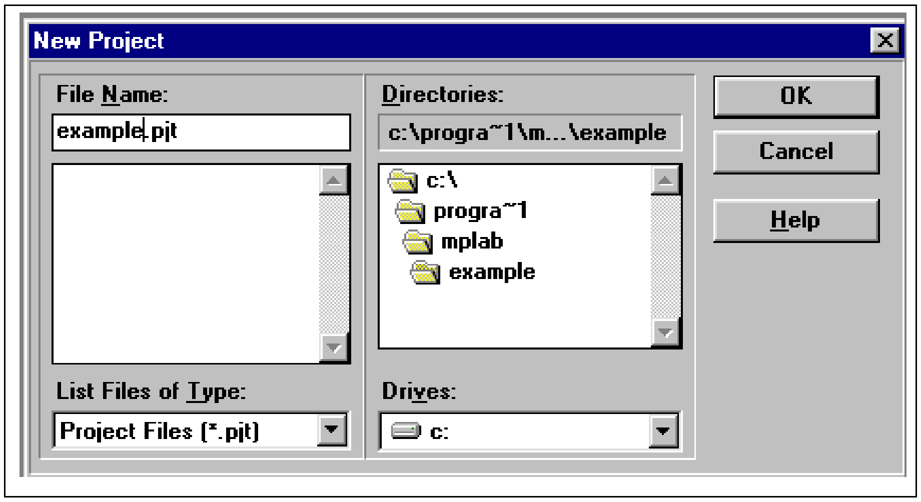
Processor: PIC16F84

Not all peripherals have support for simulation.
Serial I/O is not simulated.
Click 'Details' for additional information on PIC16F84.

Figure 4.18: Development Mode Dialog

4.6.2 New Project

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



File Name: example.pjt

Directories: c:\progra~1\m...\example

c:\
progra~1
mplab
example

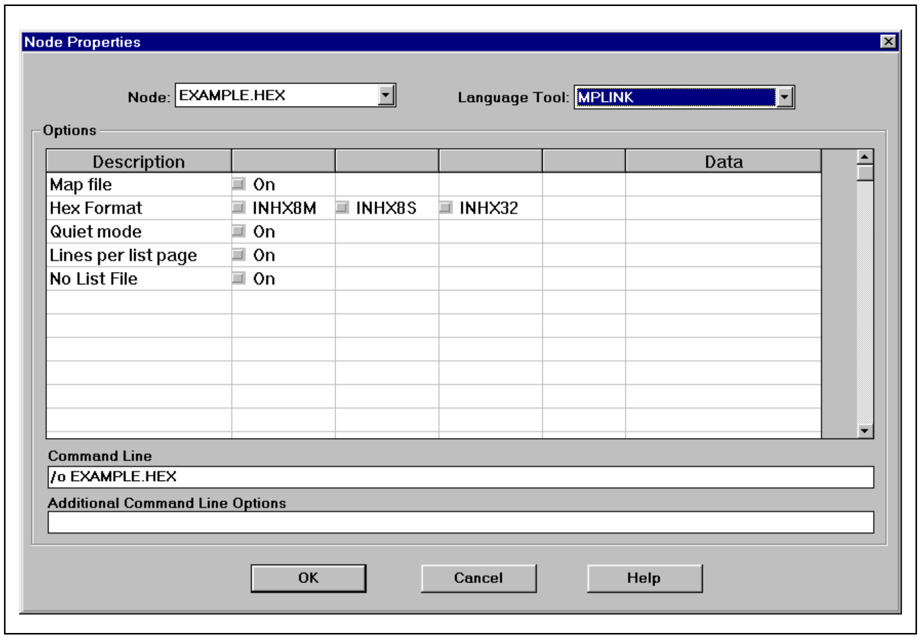
List Files of Type: Project Files (*.pjt)

Drives: c:

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.



Node: EXAMPLE.HEX
Language Tool: MPLINK

Description					Data
Map file	☐ On				
Hex Format	☐ INHX8M	☐ INHX8S	☐ INHX32		
Quiet mode	☐ On				
Lines per list page	☐ On				
No List File	☐ On				

Command Line
/o EXAMPLE.HEX

Additional Command Line Options

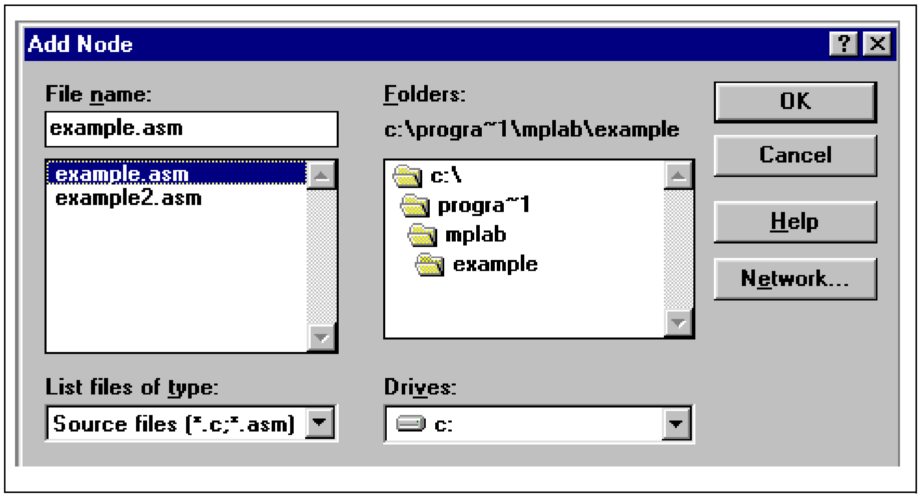
Figure 4.20: Set Node Properties Dialog

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 with MPLINK and MPLIB User’s Guide* 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` directory for this tutorial.



File name: example.asm

- example.asm
- example2.asm

Folders: c:\progra~1\mplab\example

- c:\
- progra~1
- mplab
- example

List files of type: Source files (*.c;*.asm)

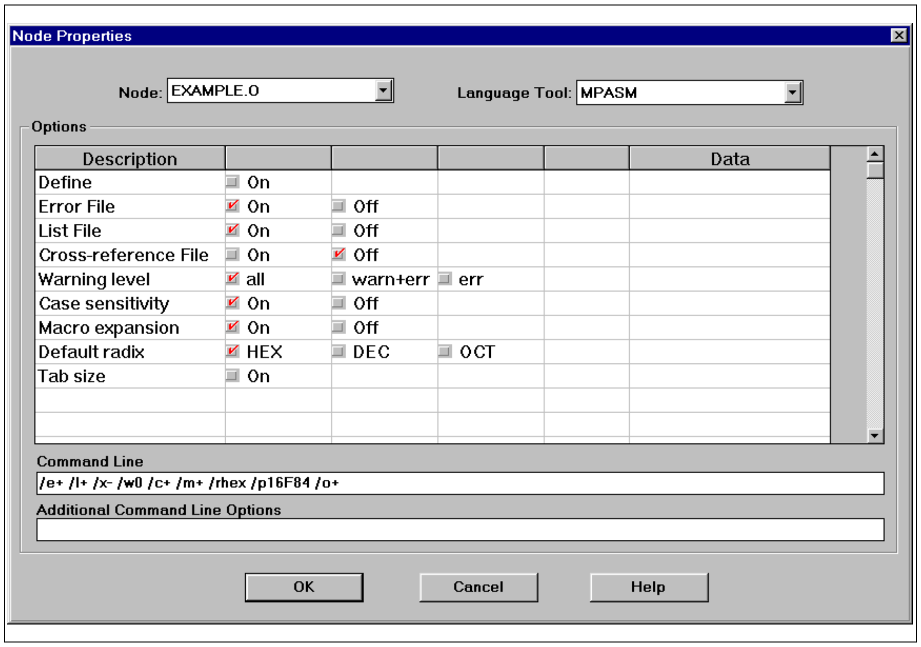
Drives: c:

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.



Node: EXAMPLE.O

Language Tool: MPASM

Options

Description					Data
Define	On				
Error File	[✓ On]	Off			
List File	[✓ On]	Off			
Cross-reference File	On	[✓ Off]			
Warning level	[✓ all]	warn+err	err		
Case sensitivity	[✓ On]	Off			
Macro expansion	[✓ On]	Off			
Default radix	[✓ HEX]	DEC	OCT		
Tab size	On				

Command Line

```
/e+ /l+ /x- /w0 /c+ /m+ /rhex /p16F84 /o+
```

Additional Command Line Options

Figure 4.22: Node Properties Dialog – Example.o

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

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

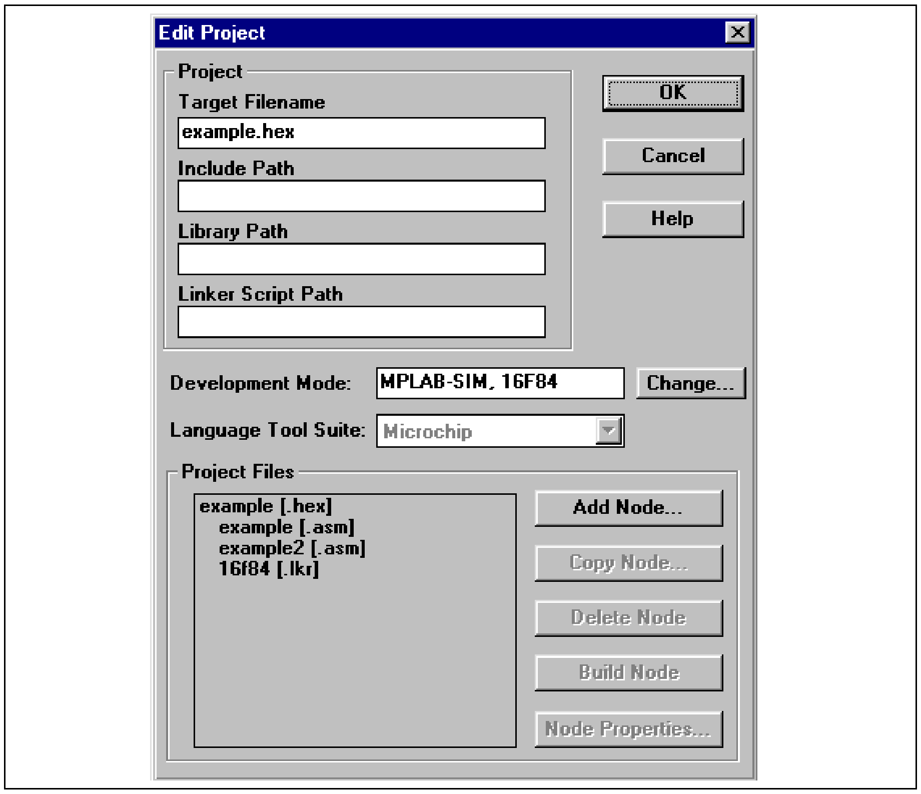
4.6.5 Adding Additional Source Files

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` directory. 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 uses to define the memory architecture of each PICmicro MCU. Standard linker scripts come with MPLINK and are in the MPLAB IDE installation directory. For this tutorial select `PIC16F84.LKR` from the `\PROGRAM FILES\MPLAB\EXAMPLE` directory. Node options can not be set for a linker script.



Edit Project

Project

Target Filename: example.hex

Include Path:

Library Path:

Linker Script Path:

Development Mode: MPLAB-SIM, 16F84

Language Tool Suite: Microchip

Project Files

```
example [.hex]
   example [.asm]
   example2 [.asm]
   16f84 [.lkr]
```

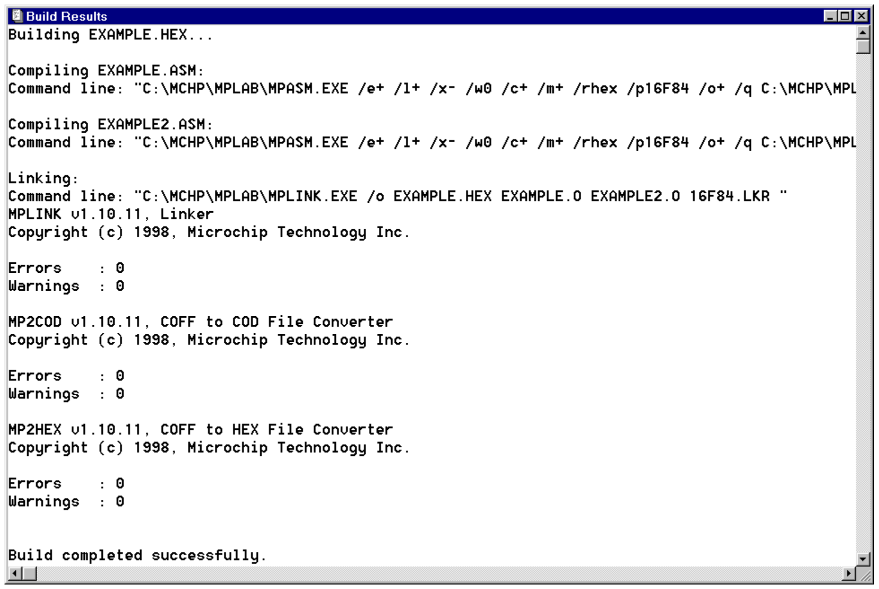
Figure 4.23: Edit Project Dialog – Linker Script

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 directories of your include files, library files, and linker script files in the appropriate box if they are not in the same directory as the project. The default language suite, paths, and nodes for all projects are set selecting *Options > Environment Setup* and clicking the **Projects** tab (see Section 7.8.5.2).

4.6.7 Make Project

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



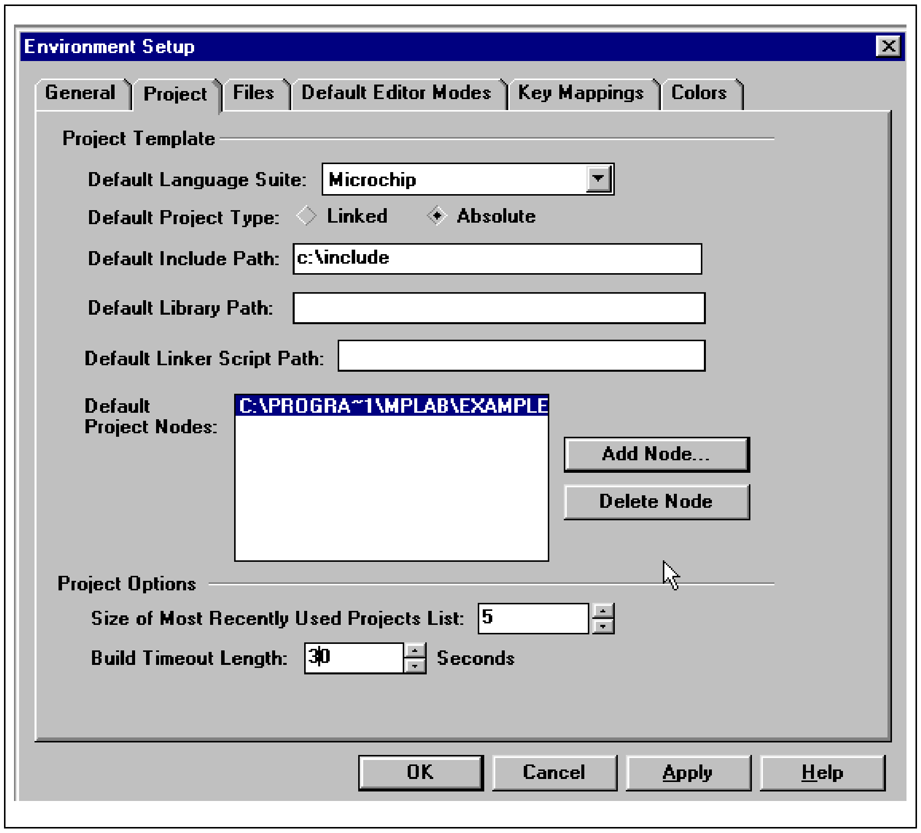
```
Building EXAMPLE.HEX...

Compiling EXAMPLE.ASM:
Command line: "C:\MCHP\MPLAB\MPASM.EXE /e+ /l+ /x- /w0 /c+ /m+ /rhex /p16F84 /o+ /q C:\MCHP\MPL

Compiling EXAMPLE2.ASM:
Command line: "C:\MCHP\MPLAB\MPASM.EXE /e+ /l+ /x- /w0 /c+ /m+ /rhex /p16F84 /o+ /q C:\MCHP\MPL

Linking:
Command line: "C:\MCHP\MPLAB\MPLINK.EXE /o EXAMPLE.HEX EXAMPLE.O EXAMPLE2.O 16F84.LKR "
MPLINK v1.10.11, Linker
Copyright (c) 1998, Microchip Technology Inc.

Errors    : 0
Warnings  : 0

MP2COD v1.10.11, COFF to COD File Converter
Copyright (c) 1998, Microchip Technology Inc.

Errors    : 0
Warnings  : 0

MP2HEX v1.10.11, COFF to HEX File Converter
Copyright (c) 1998, 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.


Environment Setup

Project Template

Default Language Suite: Microchip

Default Project Type: Linked [✓ Absolute]

Default Include Path: c:\include

Default Library Path:

Default Linker Script Path:

Default Project Nodes: C:\PROGRA~1\MPLAB\EXAMPLE

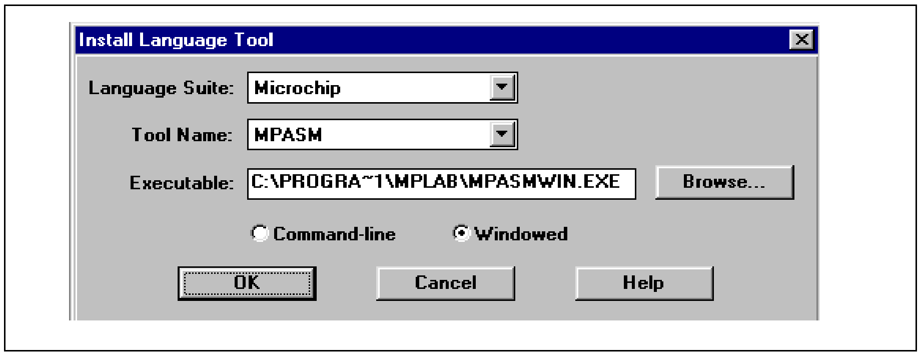
Project Options

Size of Most Recently Used Projects List: 5

Build Timeout Length: 30 Seconds

Figure 4.25: Environment Setup Dialog

5. Select *Project > Install Language Tool...* and check that MPASM is pointed to the `MPASMWIN.EXE` in the MPLAB IDE installation directory. Also, the "Windowed" option should be selected.
   Alternatively, MPASM can point to `MPASM.EXE` and the "Command-line" option selected; however this executable may not operate on Pentium 100MHz PCs and higher.


Install Language Tool

Language Suite: Microchip

Tool Name: MPASM

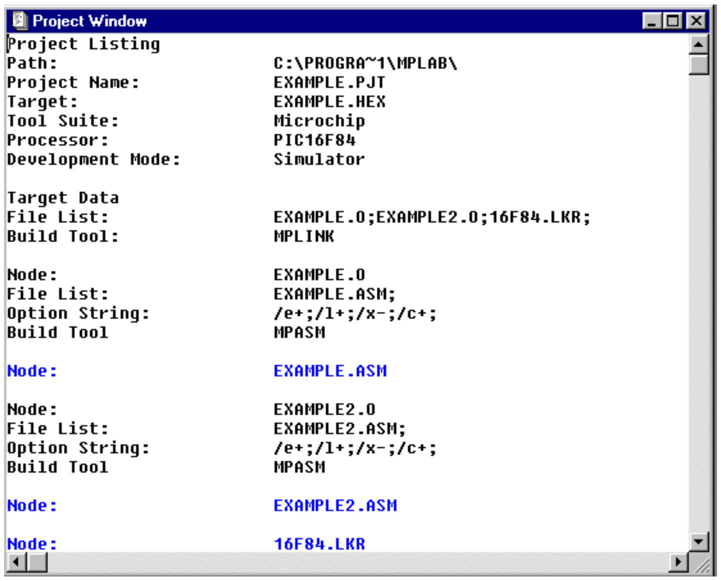
Executable: C:\PROGRA~1\MPLAB\MPASMWIN.EXE

Command-line [✓ Windowed]

Figure 4.26: Install Language Tool Dialog – MPASM

4.6.9 Project Window

Open the *Window > Project* window. It should look like this:



```
Project Listing
Path:                     C:\PROGRA~1\MPLAB\
Project Name:             EXAMPLE.PJT
Target:                   EXAMPLE.HEX
Tool Suite:               Microchip
Processor:                PIC16F84
Development Mode:         Simulator

Target Data
File List:                EXAMPLE.O;EXAMPLE2.O;16F84.LKR;
Build Tool:               MPLINK

Node:                     EXAMPLE.O
File List:                EXAMPLE.ASM;
Option String:            /e+;/l+;/x-;/c+;
Build Tool                MPASM

Node:                     EXAMPLE.ASM

Node:                     EXAMPLE2.O
File List:                EXAMPLE2.ASM;
Option String:            /e+;/l+;/x-;/c+;
Build Tool                MPASM

Node:                     EXAMPLE2.ASM

Node:                     16F84.LKR
```

Figure 4.27: Project Window

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 Hi-Tech PIC C

This tutorial will show you how to use Hi-Tech's PIC C compiler with projects in MPLAB IDE to build applications. If you have followed along the previous tutorials, select *Project > Close Project*.

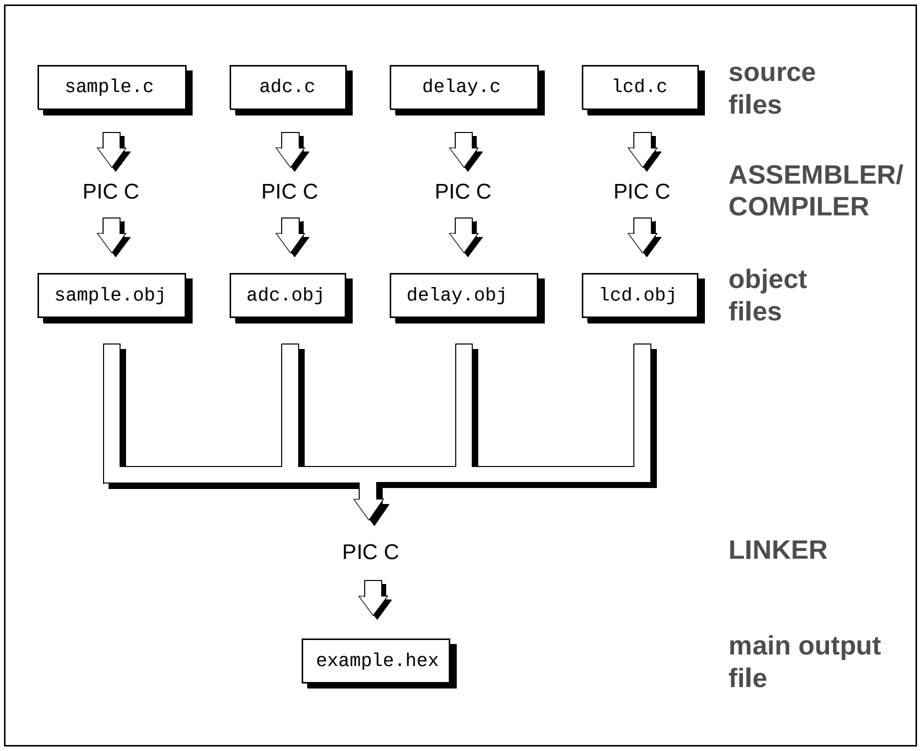
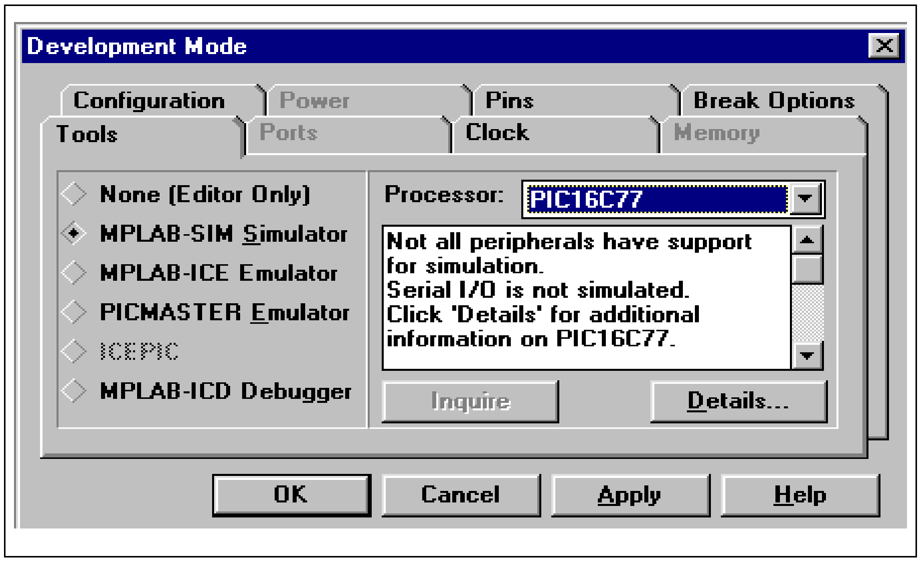


Figure 4.28: Project Relationships For PIC C Source Files

4.7.1 Set Development Mode

Set *Options > Development Mode* and click the **Tools** tab. Select MPLAB-SIM simulator and select the PIC16C77 PICmicro MCU for this example. Click **OK.**



None (Editor Only)
[✓ MPLAB-SIM Simulator]
MPLAB-ICE Emulator
PICMASTER Emulator
ICEPIC
MPLAB-ICD Debugger

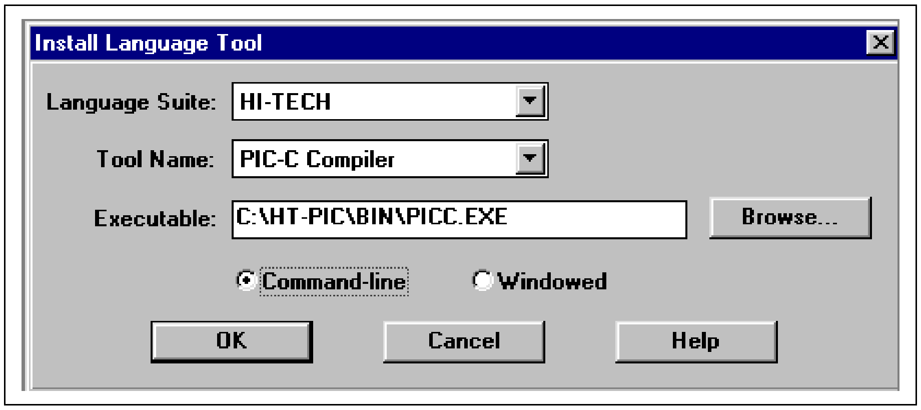
Processor: PIC16C77

Not all peripherals have support for simulation.
Serial I/O is not simulated.
Click 'Details' for additional information on PIC16C77.

Figure 4.29: Development Mode Dialog – PIC16C77

4.7.2 Install PIC C Language Tools

Make certain that PIC C is installed correctly in MPLAB IDE. The *Project > Install Language Tool* dialog should be set to something similar to this for the HI-TECH tools (your executable path may be different):



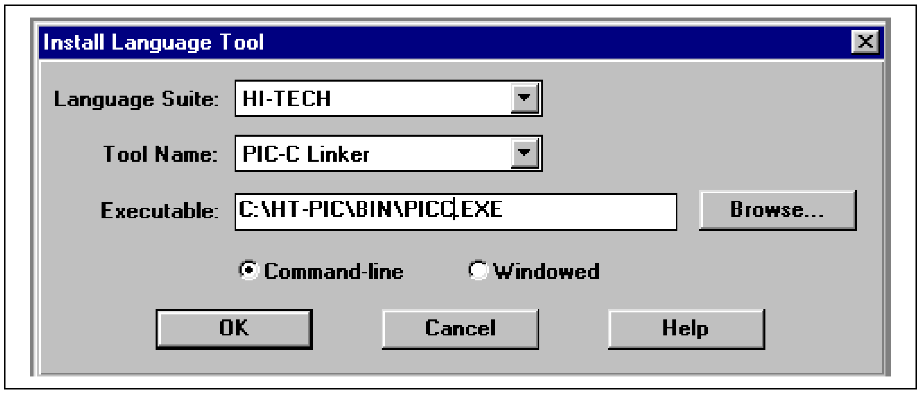
Language Suite: HI-TECH

Tool Name: PIC-C Compiler

Executable: C:\HT-PIC\BIN\PICC.EXE

[✓ Command-line] Windowed

Figure 4.30: Install Language Tool Dialog – PIC C Compiler



Install Language Tool

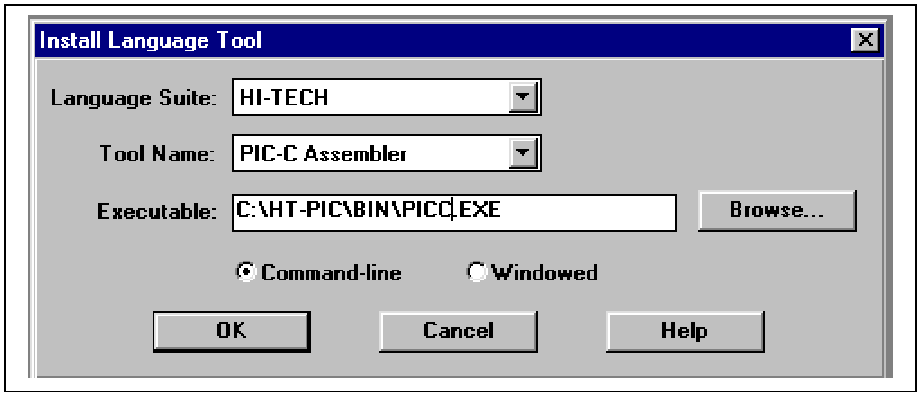
Language Suite: HI-TECH

Tool Name: PIC-C Linker

Executable: C:\HT-PIC\BIN\PICC.EXE

[✓ Command-line] Windowed

Figure 4.31: Install Language Tool Dialog – PIC C Linker



Install Language Tool

Language Suite: HI-TECH

Tool Name: PIC-C Assembler

Executable: C:\HT-PIC\BIN\PICC.EXE

[✓ Command-line] Windowed

Figure 4.32: Install Language Tool Dialog – PIC C Assembler

Use the **Browse** button to point to `PICC.EXE` on your system for the compiler, linker, and assembler.

4.7.3 New Project

Select *Project > New Project* and select a directory for a new project, then type its name. Name it `SAMPLE.PJT` in the `\HT-PIC\SAMPLES` directory.

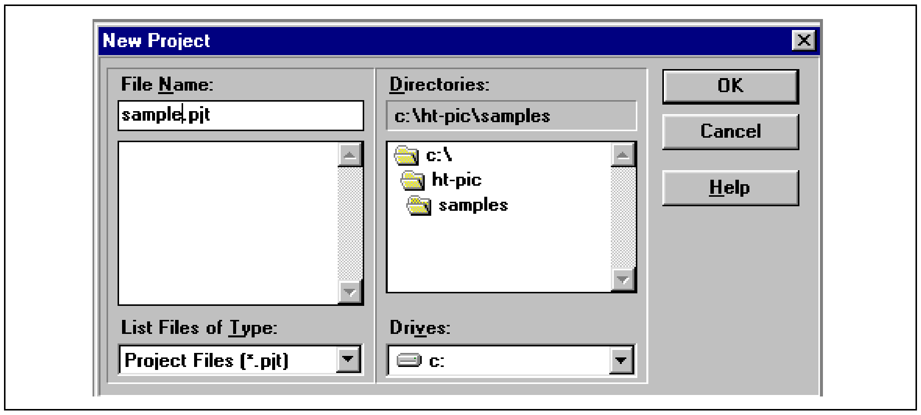


Figure 4.33: New Project Dialog – sample.pjt

After setting the project name, click **OK** and the Edit Project dialog will be shown.

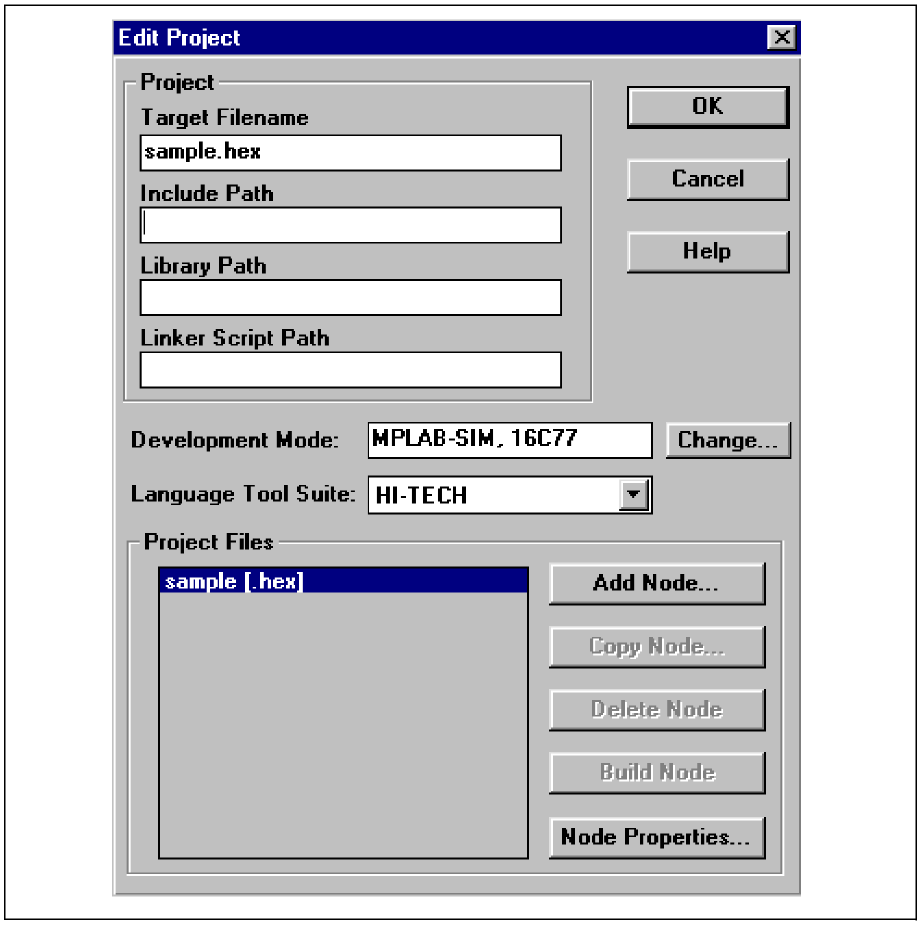


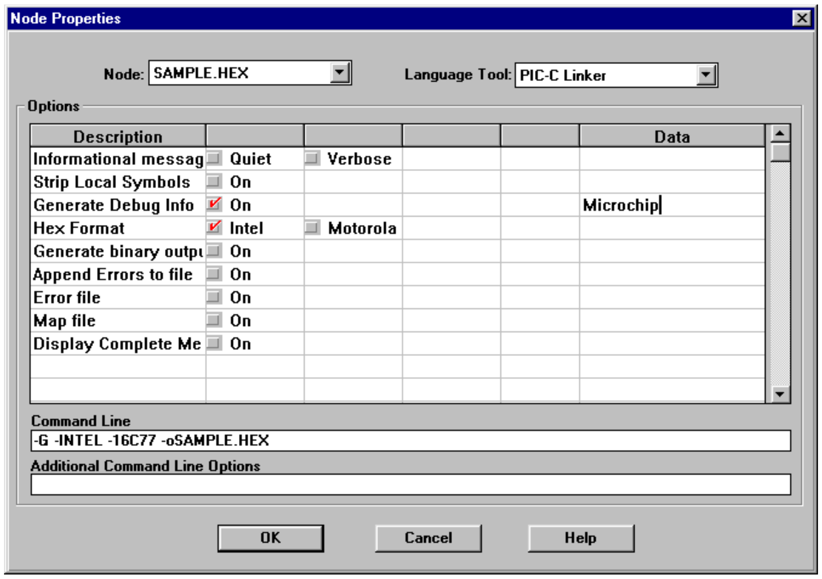
Figure 4.34: Edit Project Dialog – sample.hex

Make sure to set the Language Tool Suite to HI-TECH.

Note: Be sure you set the paths in the Edit Project window so that MPLAB IDE knows the correct compiler location. If you install your compiler in a different directory, you may see warning messages saying a source file couldn’t be opened.

4.7.4 Set Node Properties

Select the name of the project in the Project Files dialog of the Edit Project Dialog and click **Node Properties**. Set the language tool to **PIC-C Linker** and check the Generate Debug Info check box. Type in “Microchip” in the Data column as shown below:



Node: SAMPLE.HEX
Language Tool: PIC-C Linker

Description					Data
Informational messag	Quiet	Verbose			
Strip Local Symbols	On				
Generate Debug Info	✓ On				Microchip
Hex Format	✓ Intel	Motorola			
Generate binary outpu	On				
Append Errors to file	On				
Error file	On				
Map file	On				
Display Complete Me	On				

Command Line
-G -INTEL -16C77 -oSAMPLE.HEX

Additional Command Line Options

Figure 4.35: Node Properties Dialog – sample.hex

The Node Properties dialog shows the command line switches for the tool, in this case PIC C. When you first open this dialog, the checked boxes represent the default values for the tool. For this tutorial, only the Debug Info Generate setting needs to be changed. Refer to the Hi-Tech documentation for more information on these command line switches.

Click **OK** in the Node Properties dialog to return to the Edit Project dialog.

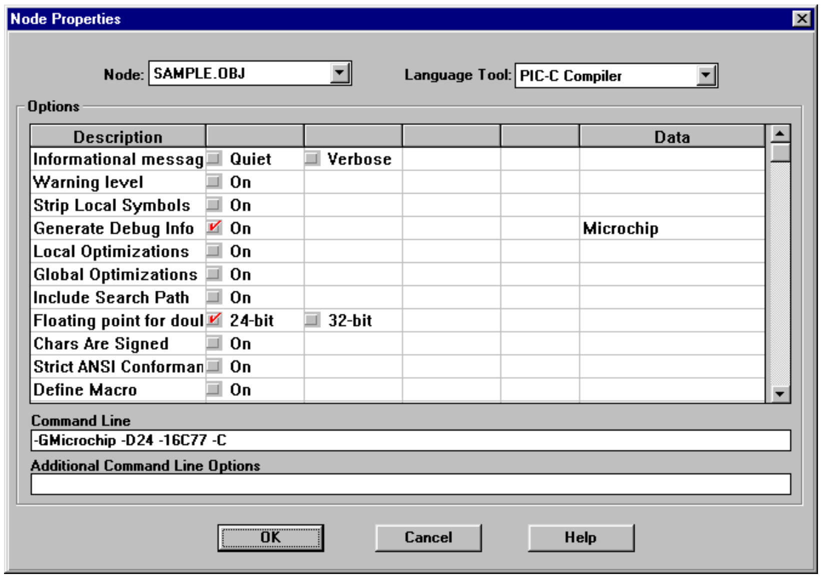
4.7.5 Add Source Files

Click **Add Node** in the Edit Project dialog. Add the source file, `SAMPLE.C` from the `\HT-PIC\SAMPLES` directory.

When the file name is shown and selected in the Add Node dialog, click **Node Properties**.

Set up this dialog this way:

- Set the Language Tool for `SAMPLE.OBJ` to PIC C Compiler
- Check the Generate Debug Info box
- Enter “Microchip” in the Generate Debug Info data column



Node: SAMPLE.OBJ

Language Tool: PIC-C Compiler

Description					Data
Informational messag	☐ Quiet	☐ Verbose			
Warning level	☐ On				
Strip Local Symbols	☐ On				
Generate Debug Info	✓ On				Microchip
Local Optimizations	☐ On				
Global Optimizations	☐ On				
Include Search Path	☐ On				
Floating point for doul	✓ 24-bit	☐ 32-bit			
Chars Are Signed	☐ On				
Strict ANSI Conforman	☐ On				
Define Macro	☐ On				

Command Line

```
-GMicrochip -D24 -16C77 -C
```

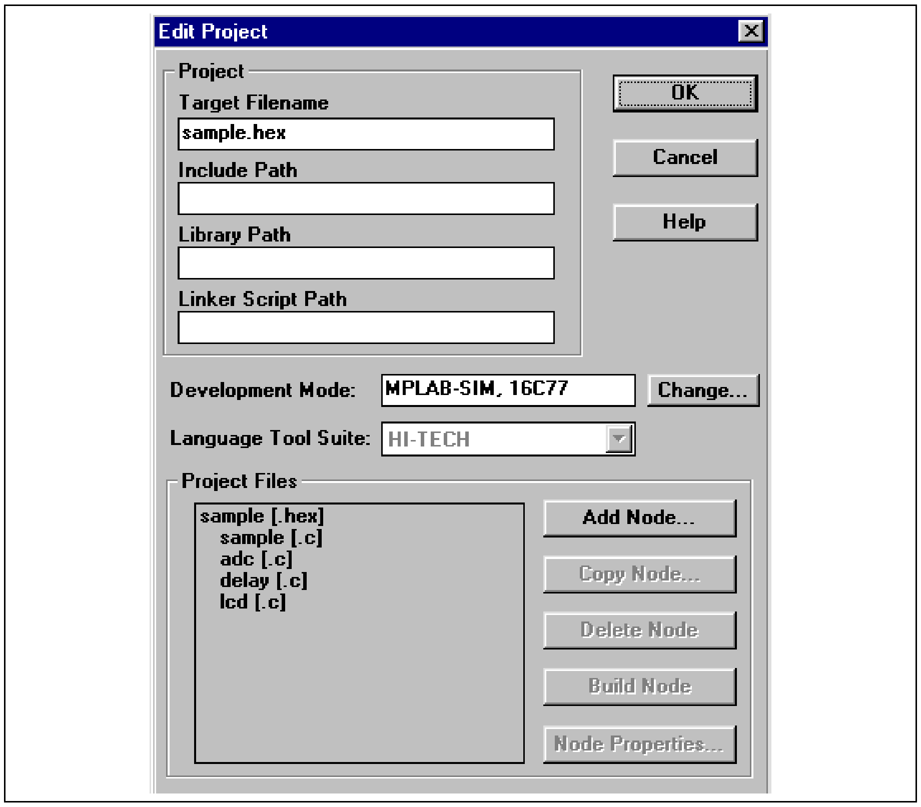
Additional Command Line Options

Figure 4.36: Node Properties Dialog – sample.obj

The Object filename is set to `SAMPLE.OBJ` automatically.

The Node Properties dialog shows the command line switches for the tool, in this case PIC C. The checked boxes represent the default values for the tool. For this tutorial, only the debug info setting needs to be changed. Refer to the Hi-Tech PIC C documentation for more information on these command line switches.

Click **OK**, select the SAMPLE.C node, and use the **Copy Node** button to add `ADC.C`, `DELAY.C`, and `LCD.C` with the same Node Properties as `SAMPLE.C`. When you are finished, the project should look like this:



Target Filename: sample.hex

Include Path:

Library Path:

Linker Script Path:

Development Mode: MPLAB-SIM, 16C77

Language Tool Suite: HI-TECH

Project Files:

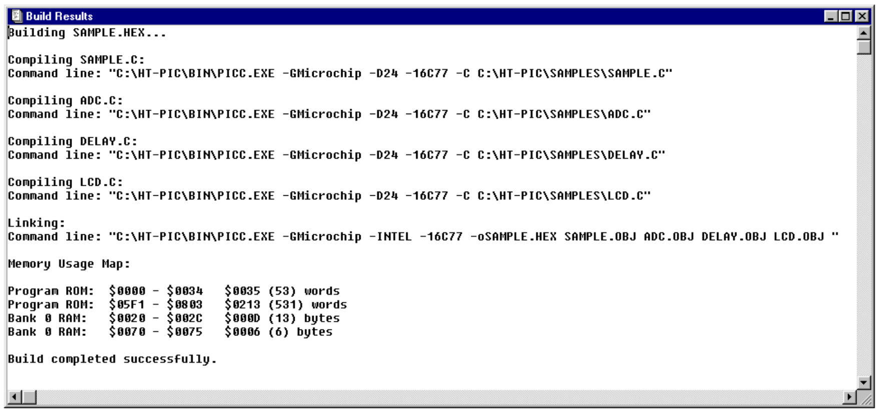
```
sample [.hex]
   sample [.c]
   adc [.c]
   delay [.c]
   lcd [.c]
```

Figure 4.37: Edit Project Dialog – sample.hex

The default language suite, paths, and nodes for all projects are set by selecting *Options > Environment Setup* and clicking the Projects tab.

4.7.6 Make Project

Select *Project > Make Project* from the menu to compile the application using the Hi-Tech compiler and linker. A Build Results window is created that shows the command lines sent to each tool. It should look like this:



```
Building SAMPLE.HEX...

Compiling SAMPLE.C:
Command line: "C:\HT-PIC\BIN\PICC.EXE -GMicrochip -D24 -16C77 -C C:\HT-PIC\SAMPLES\SAMPLE.C"

Compiling ADC.C:
Command line: "C:\HT-PIC\BIN\PICC.EXE -GMicrochip -D24 -16C77 -C C:\HT-PIC\SAMPLES\ADC.C"

Compiling DELAY.C:
Command line: "C:\HT-PIC\BIN\PICC.EXE -GMicrochip -D24 -16C77 -C C:\HT-PIC\SAMPLES\DELAY.C"

Compiling LCD.C:
Command line: "C:\HT-PIC\BIN\PICC.EXE -GMicrochip -D24 -16C77 -C C:\HT-PIC\SAMPLES\LCD.C"

Linking:
Command line: "C:\HT-PIC\BIN\PICC.EXE -GMicrochip -INTEL -16C77 -oSAMPLE.HEX SAMPLE.OBJ ADC.OBJ DELAY.OBJ LCD.OBJ "

Memory Usage Map:

Program ROM:  $0000 - $0034   $0035 (53) words
Program ROM:  $05F1 - $0803   $0213 (531) words
Bank 0 RAM:   $0020 - $002C   $000D (13) bytes
Bank 0 RAM:   $0070 - $0075   $0006 (6) bytes

Build completed successfully.
```

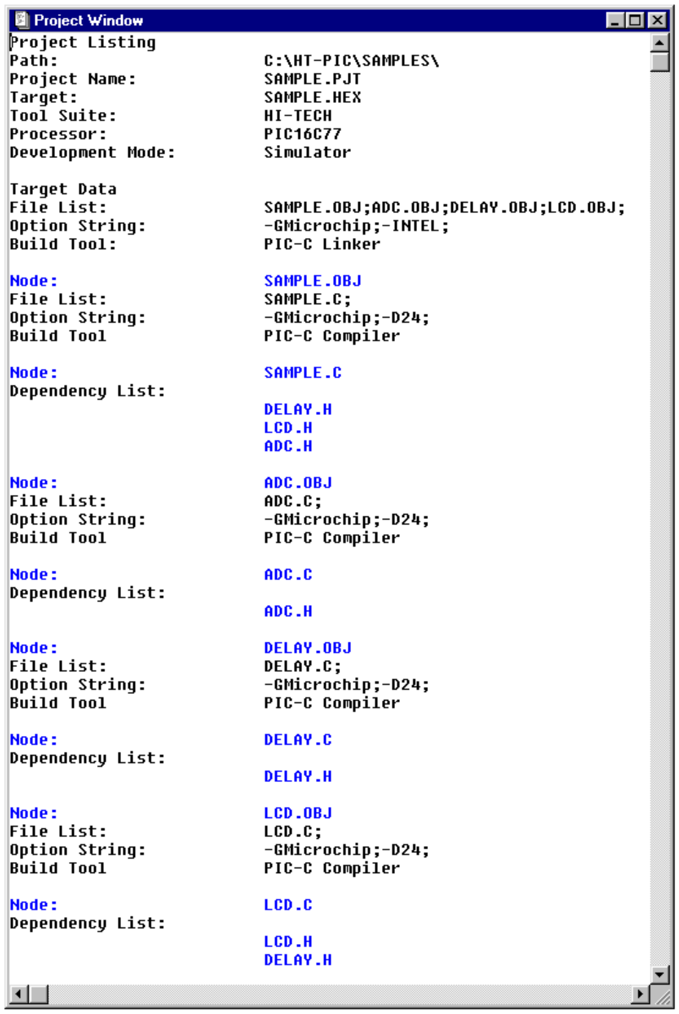
Figure 4.38: Build Results Window – sample.hex

4.7.7 Troubleshooting

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 (PIC-C Linker) 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 4.7.5.
4. Select *Project > Install Language Tool...* and check that PIC C Compiler and PIC C Linker are both pointing to the `PICC.EXE` executable.

4.7.8 Project Window

Open the *Window > Project* window. It should look like this:



Project Window

```
Project Listing
Path:                     C:\HT-PIC\SAMPLES\
Project Name:             SAMPLE.PJT
Target:                   SAMPLE.HEX
Tool Suite:               HI-TECH
Processor:                PIC16C77
Development Mode:         Simulator

Target Data
File List:                SAMPLE.OBJ;ADC.OBJ;DELAY.OBJ;LCD.OBJ;
Option String:            -GMicrochip;-INTEL;
Build Tool:               PIC-C Linker

Node:                     SAMPLE.OBJ
File List:                SAMPLE.C;
Option String:            -GMicrochip;-D24;
Build Tool                PIC-C Compiler

Node:                     SAMPLE.C
Dependency List:
                          DELAY.H
                          LCD.H
                          ADC.H

Node:                     ADC.OBJ
File List:                ADC.C;
Option String:            -GMicrochip;-D24;
Build Tool                PIC-C Compiler

Node:                     ADC.C
Dependency List:
                          ADC.H

Node:                     DELAY.OBJ
File List:                DELAY.C;
Option String:            -GMicrochip;-D24;
Build Tool                PIC-C Compiler

Node:                     DELAY.C
Dependency List:
                          DELAY.H

Node:                     LCD.OBJ
File List:                LCD.C;
Option String:            -GMicrochip;-D24;
Build Tool                PIC-C Compiler

Node:                     LCD.C
Dependency List:
                          LCD.H
                          DELAY.H
```

Figure 4.39: Project Window – sample.pjt

4.7.9 Summary

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

- Set up Language Tools for PIC C Compiler, Linker, and Assembler.
- You may need to set the Include File Directory to \HT-PIC\H (or where PIC C include files are installed on your system).
- Create new project with *Project > New Project.*
- Turn on Generate Debug Info for project node.
- Set project Node Properties to PIC C Linker.
- Add Source files, setting language tool to PIC C Compiler or Assembler.
- Turn on Generate Debug Info for each source node.
- Set Generate Debug Info Data to “Microchip” for each source node.

4.8 Making a Project with MPLAB-C17 or MPLAB-C18

For information on making a project using MPLAB-C17 or MPLAB-C18, consult the *MPLAB-CXX User’s Guide* (DS51217).

NOTES:

Chapter 5. MPLAB Editor

5.1 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 Chapter 7 along with all other MPLAB IDE functions.

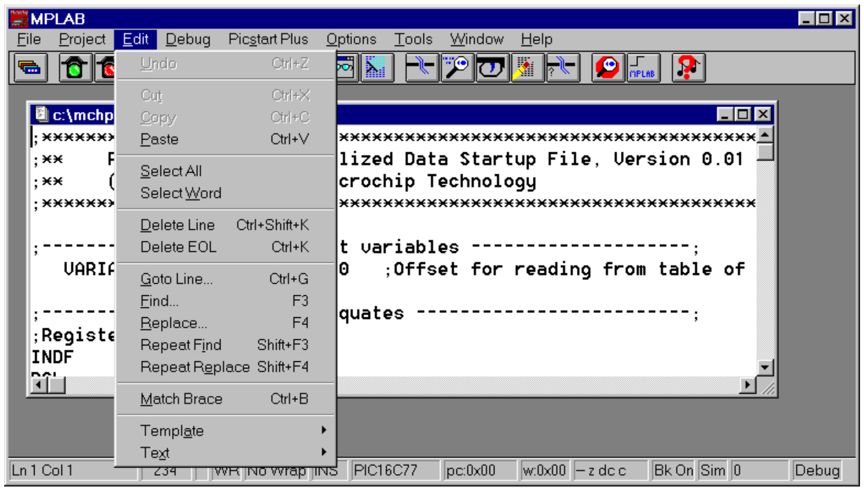
5.2 Highlights

The following topics are addressed in this chapter:

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

5.3 What is 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.



```
;*******  ************************************
;**     F  lized Data Startup File, Version 0.01
;**     (  crochip Technology
;*******  ************************************

;-------  t variables --------------------;
   VARIA  0    ;Offset for reading from table of

;-------  quates -------------------------;
;Registe
INDF
```

Figure 5.1: Using the MPLAB Editor

5.4 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.

5.5 MPLAB Editor Features

5.5.1 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.

5.5.2 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

5.5.3 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.

5.5.4 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.

5.6 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

5.6.1 File Operations

To create a file, select a file for editing, and save a file, use the File menu options (see Section 7.4). The MPLAB Editor allows you to save the file by overwriting the existing file or by saving the file to a new filename.

5.6.2 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 subdirectory of MPLAB IDE or create your own.

Once you set MPLAB IDE up 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 create 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.

5.6.3 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 7.6 for details.

5.6.3.1 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.

5.6.3.2 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.

5.6.3.3 Changing Case

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

5.6.3.4 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.

5.6.3.5 Undoing Edit Actions

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

5.6.3.6 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.

5.6.3.6.1. Language type “(none)” or “C”

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

5.6.3.6.2. 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.

5.6.4 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 Section 7.8.3 and Section 7.8.4 for more information on Editor Modes.

5.6.5 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;
    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 6. Debugging and MPLAB-SIM

6.1 Introduction

This chapter discusses MPLAB debugging functions and related MPLAB-SIM simulator considerations. You can be in the simulator (MPLAB-SIM) or in emulator mode (MPLAB-ICE, PICMASTER® emulator, ICEPIC, or MPLAB-ICD) to access debugging functions. Refer to the *MPLAB-ICE User's Guide* for information on debugging using the MPLAB-ICE emulator.

6.2 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

6.3 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 to simulate your code or you can use the MPLAB-ICE 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.

The MPLAB-ICE 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.

6.4 Real-Time Program Execution

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

6.4.1 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.

6.4.2 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.

6.5 MPLAB-SIM Simulator Environment

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

- Model operation of Microchip Technology's PICmicro MCU, e.g., PIC12CXX, PIC14000, PIC16C5X, PIC16CXX, 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 the MPLAB-ICE emulator) is designed to debug software. MPLAB-SIM 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

6.5.1 I/O Timing

External timing in MPLAB-SIM is processed only once during each instruction cycle. Transient signals, such as a spikes on $\overline{\text{MCLR}}$ 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 prior to the next instruction cycle.

6.5.2 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 attempts to provide the fastest possible simulation cycle, and depending upon the mode of operation, can operate on the order of milliseconds per instruction.

6.5.3 Cost

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

6.5.4 Debugging Tool

MPLAB-SIM 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, MPLAB-SIM 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.

6.6 Simulator Considerations

MPLAB-SIM executes on instruction cycle boundaries, and resolutions shorter than one instruction cycle (T_{CY}) can not be simulated. MPLAB-SIM 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 can not 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 does not simulate serial I/O.

6.7 Break and Trace Points

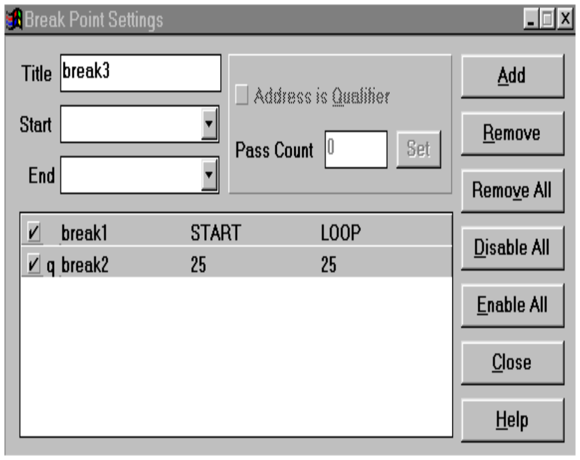
The debug functions affect execution of program instructions based upon the following elements:

- 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.

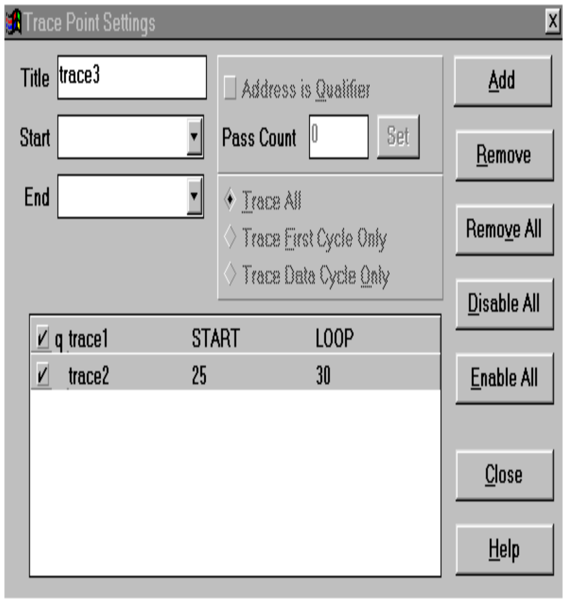


Title: break3
Start:
End:
Pass Count: 0

✓	break1	START	LOOP
✓ q	break2	25	25

Figure 6.1: Break Point Settings Dialog

Note: The MPLAB-ICD allows only one break point address to be set.



Title: trace3

Start:

End:

☐ Address is Qualifier

Pass Count: 0 Set

(● Trace All) ○ Trace First Cycle Only ○ Trace Data Cycle Only

✓ q	trace1	START	LOOP
✓	trace2	25	30

Figure 6.2: Trace Point Settings Dialog

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

6.7.1 Real-Time Break Points

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

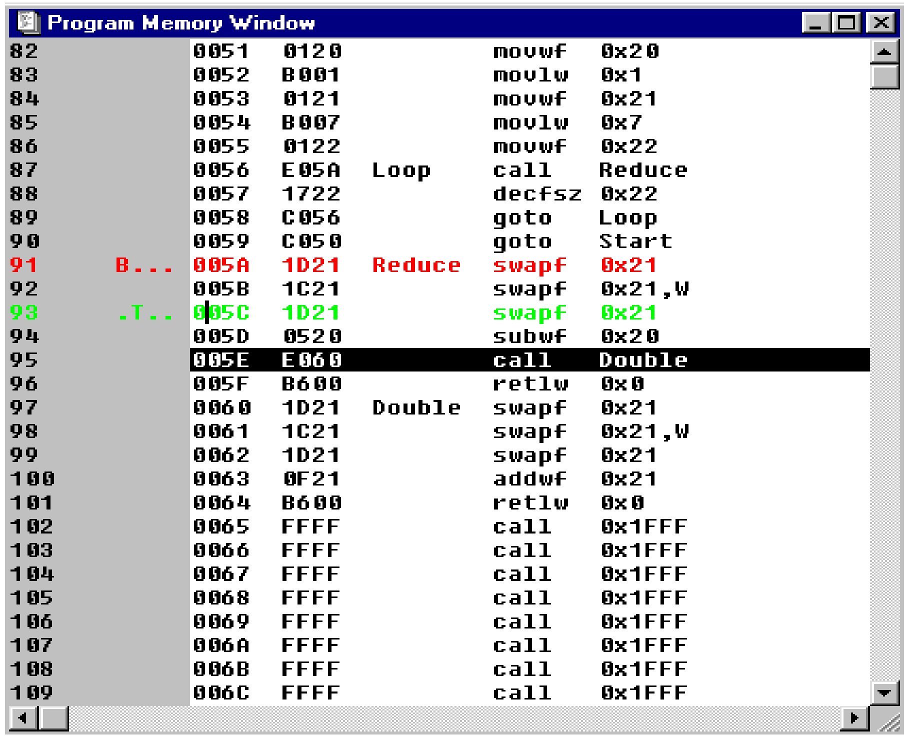
Note: If execution doesn't halt at the break point, select *Options > Development Mode* and click the **Break Options** tab. Make sure that Global Break Enable is selected (check marked).

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 Watch Dog Timer Time Out
- User Halt

The Program Memory Window shown in Figure 6.3 shows the following information:

B	Break Points
T	Trace Points
Q	Pass Counter Addresses



```
82          0051  0120          movwf   0x20
83          0052  B001          movlw   0x1
84          0053  0121          movwf   0x21
85          0054  B007          movlw   0x7
86          0055  0122          movwf   0x22
87          0056  E05A  Loop    call    Reduce
88          0057  1722          decfsz  0x22
89          0058  C056          goto    Loop
90          0059  C050          goto    Start
91   B...   005A  1D21  Reduce  swapf   0x21
92          005B  1C21          swapf   0x21,W
93   .T..   005C  1D21          swapf   0x21
94          005D  0520          subwf   0x20
95          005E  E060          call    Double
96          005F  B600          retlw   0x0
97          0060  1D21  Double  swapf   0x21
98          0061  1C21          swapf   0x21,W
99          0062  1D21          swapf   0x21
100         0063  0F21          addwf   0x21
101         0064  B600          retlw   0x0
102         0065  FFFF          call    0x1FFF
103         0066  FFFF          call    0x1FFF
104         0067  FFFF          call    0x1FFF
105         0068  FFFF          call    0x1FFF
106         0069  FFFF          call    0x1FFF
107         006A  FFFF          call    0x1FFF
108         006B  FFFF          call    0x1FFF
109         006C  FFFF          call    0x1FFF
```

Figure 6.3: Program Memory Window

6.7.1.1 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.

6.7.1.2 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).

6.7.1.3 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.

6.7.1.4 Break On Stack Overflow

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

6.7.1.5 Break On Watchdog Timer

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

6.7.1.6 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)

6.7.2 Real-Time Trace Points

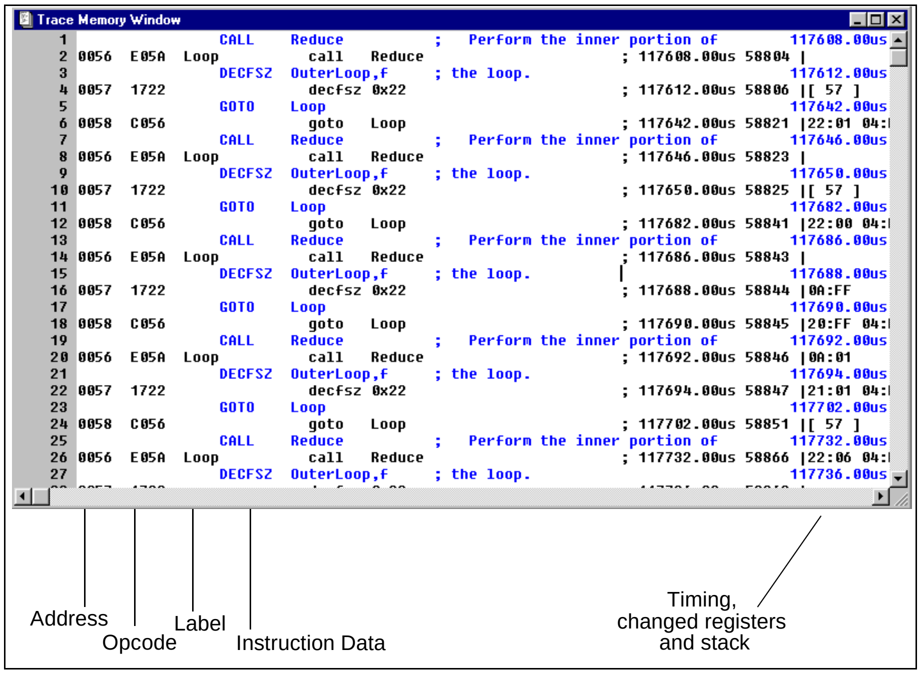
A trace is a function that logs program execution. The MPLAB-SIM simulator has an 8K real-time trace buffer that logs addresses and opcodes as they execute. This circular trace buffer continues logging data after the buffer is full, losing the oldest data (unless you have selected Break on Trace Buffer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

6.7.2.1 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



```
 1                CALL    Reduce         ;   Perform the inner portion of       117608.00us
 2 0056  E05A  Loop          call   Reduce                        ; 117608.00us 58804 |
 3                DECFSZ  OuterLoop,f    ; the loop.                           117612.00us
 4 0057  1722                decfsz 0x22                          ; 117612.00us 58806 |[ 57 ]
 5                GOTO    Loop                                                 117642.00us
 6 0058  C056                goto   Loop                          ; 117642.00us 58821 |22:01 04:
 7                CALL    Reduce         ;   Perform the inner portion of       117646.00us
 8 0056  E05A  Loop          call   Reduce                        ; 117646.00us 58823 |
 9                DECFSZ  OuterLoop,f    ; the loop.                           117650.00us
10 0057  1722                decfsz 0x22                          ; 117650.00us 58825 |[ 57 ]
11                GOTO    Loop                                                 117682.00us
12 0058  C056                goto   Loop                          ; 117682.00us 58841 |22:00 04:
13                CALL    Reduce         ;   Perform the inner portion of       117686.00us
14 0056  E05A  Loop          call   Reduce                        ; 117686.00us 58843 |
15                DECFSZ  OuterLoop,f    ; the loop.             |             117688.00us
16 0057  1722                decfsz 0x22                          ; 117688.00us 58844 |0A:FF
17                GOTO    Loop                                                 117690.00us
18 0058  C056                goto   Loop                          ; 117690.00us 58845 |20:FF 04:
19                CALL    Reduce         ;   Perform the inner portion of       117692.00us
20 0056  E05A  Loop          call   Reduce                        ; 117692.00us 58846 |0A:01
21                DECFSZ  OuterLoop,f    ; the loop.                           117694.00us
22 0057  1722                decfsz 0x22                          ; 117694.00us 58847 |21:01 04:
23                GOTO    Loop                                                 117702.00us
24 0058  C056                goto   Loop                          ; 117702.00us 58851 |[ 57 ]
25                CALL    Reduce         ;   Perform the inner portion of       117732.00us
26 0056  E05A  Loop          call   Reduce                        ; 117732.00us 58866 |22:06 04:
27                DECFSZ  OuterLoop,f    ; the loop.                           117736.00us
```

Figure 6.4: Trace Memory Window

6.7.2.2 Halting Trace from the Tool Bar

Halt Trace allows you to take a snapshot of the trace buffer and look at the captured trace without halting the processor. In the Tool Bar, 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.

6.7.2.3 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.

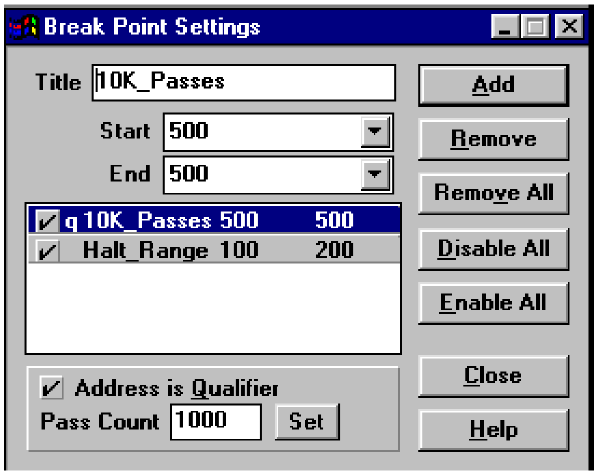
6.7.3 Assigning a Pass Count to Break or Trace Points

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.

6.7.3.1 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.



Title 10K_Passes
Start 500
End 500

✓	q 10K_Passes	500	500
✓	Halt_Range	100	200

✓ Address is Qualifier
Pass Count 1000

Figure 6.5: Break Point Settings Dialog – Pass Counter

Example 6.1: This example shows break points and pass counter addresses used in the same code (Figure 6.5). 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.

6.7.3.2 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.

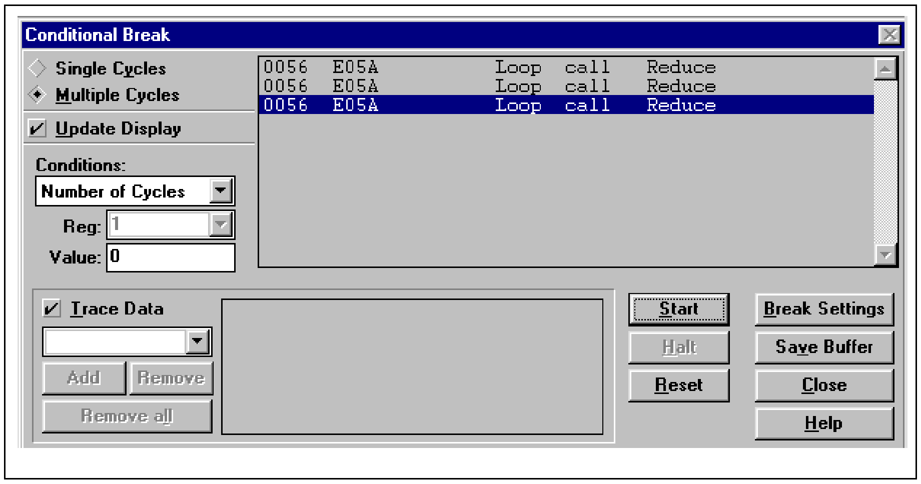
6.7.3.3 Using Pass Counter to Count Events

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

6.8 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.



Selected: Multiple Cycles; Update Display checked; Conditions: Number of Cycles; Reg: 1; Value: 0; Trace Data checked.

```
0056   E05A          Loop  call   Reduce
0056   E05A          Loop  call   Reduce
0056   E05A          Loop  call   Reduce
```

Figure 6.6: Conditional Break Dialog

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

6.8.1 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.

6.8.2 Trace Data

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

6.8.3 Single Cycle

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

6.8.4 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.

6.9 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

6.9.1 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:

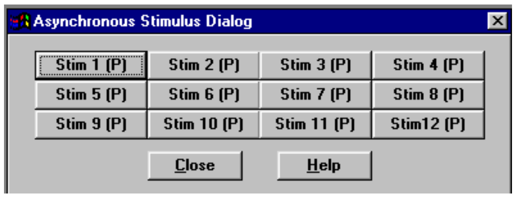


Figure 6.7: Asynchronous Stimulus Dialog

Place the cursor over the button labelled "Stim1 (P)" and click the right mouse button. A shortcut menu will appear. Scroll down and select *Toggle*.

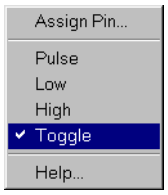


Figure 6.8: Toggle Option

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.

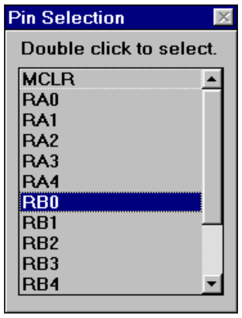


Figure 6.9: Pin Selection

Put the cursor over "RB0" and double click. The Asynchronous Stimulus dialog should now look like Figure 6.10. Note that the button now shows "RB0 (T)."

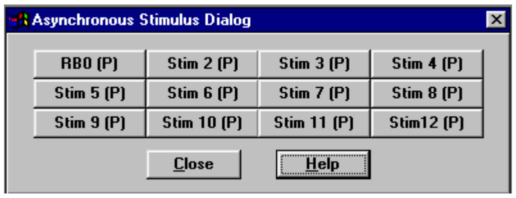


Figure 6.10: Asynchronous Stimulus Dialog – RB0(T)

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.

6.9.2 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.

6.9.2.1 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.

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 and final 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.

> **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.

6.9.2.2 Using a Pin Stimulus file

1. Select *Debug > Simulator Stimulus > Pin Stimulus > Enable* 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.

6.9.2.3 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.

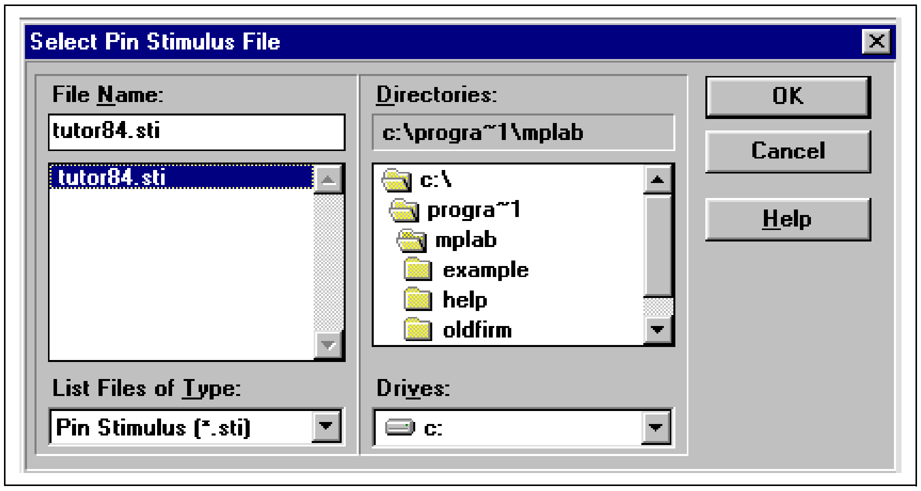
```
CYCLE    RB1     RB0
20       0        0
41       1        0   ; apply high to port b bit 1
52       0        1   ; apply high to port b bit 0, set bit 1 low
55       1        1
60       0        0
65       1        0   ; toggle bit 1, then...
76       0        1   ! ...toggle bit 0.
```

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.



File Name: tutor84.sti

Directories: c:\progra~1\mplab

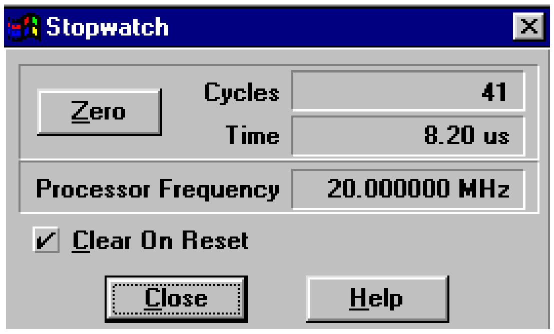
List Files of Type: Pin Stimulus (*.sti)

Figure 6.11: Pin Stimulus Enable

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.

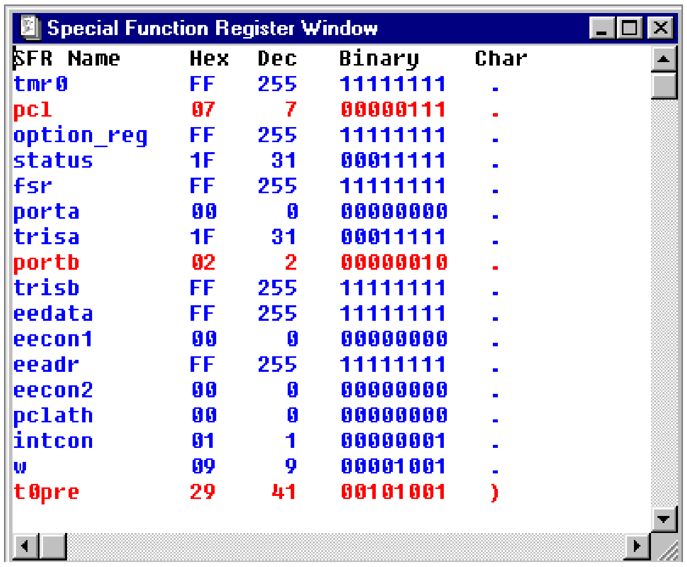
   The Stopwatch window will also show the elapsed time at each instruction, as determined from the CYCLE value and the clock frequency. If the Stopwatch is reset to 0, the pin stimulus file will also be effectively reset.

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.



Cycles 41
Time 8.20 us
Processor Frequency 20.000000 MHz
[✓] Clear On Reset

Figure 6.12: Stopwatch Window (41 Cycles)



```
SFR Name      Hex   Dec   Binary     Char
tmr0          FF    255   11111111    .
pcl           07      7   00000111    .
option_reg    FF    255   11111111    .
status        1F     31   00011111    .
fsr           FF    255   11111111    .
porta         00      0   00000000    .
trisa         1F     31   00011111    .
portb         02      2   00000010    .
trisb         FF    255   11111111    .
eedata        FF    255   11111111    .
eecon1        00      0   00000000    .
eeadr         FF    255   11111111    .
eecon2        00      0   00000000    .
pclath        00      0   00000000    .
intcon        01      1   00000001    .
w             09      9   00001001    .
t0pre         29     41   00101001    )
```

Figure 6.13: Special Function Registers Window

6.9.3 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.

6.9.3.1 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.

6.9.3.2 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 MPLAB-SIM executes.

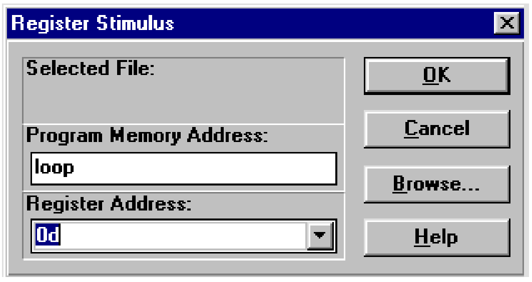
6.9.3.3 Register Stimulus example

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
```

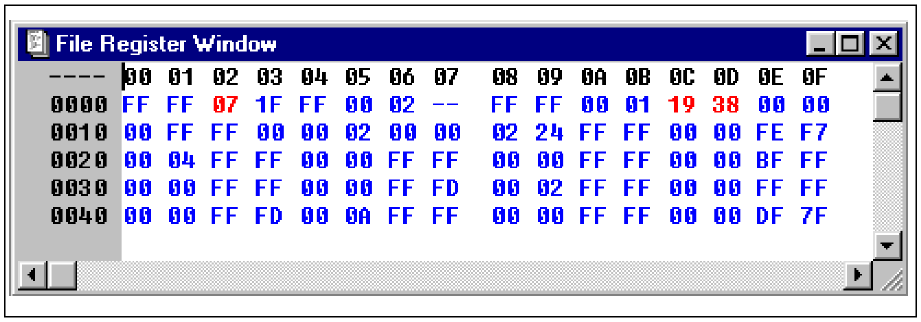
2. Select *File > Save As* to save the file and name it `tutor84.reg`. This file will be used to sequentially inject these values into a register.
3. Select *Debug > Simulator Stimulus > Register Stimulus > Enable*, then set loop to be the place in the program when values are injected, and for demonstration 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.



Selected File:
Program Memory Address: loop
Register Address: 0d

Figure 6.14: Register Stimulus Dialog

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



File Register Window

```
---- 00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F
0000 FF FF 07 1F FF 00 02 --  FF FF 00 01 19 38 00 00
0010 00 FF FF 00 00 02 00 00  02 24 FF FF 00 00 FE F7
0020 00 04 FF FF 00 00 FF FF  00 00 FF FF 00 00 BF FF
0030 00 00 FF FF 00 00 FF FD  00 02 FF FF 00 00 FF FF
0040 00 00 FF FD 00 0A FF FF  00 00 FF FF 00 00 DF 7F
```

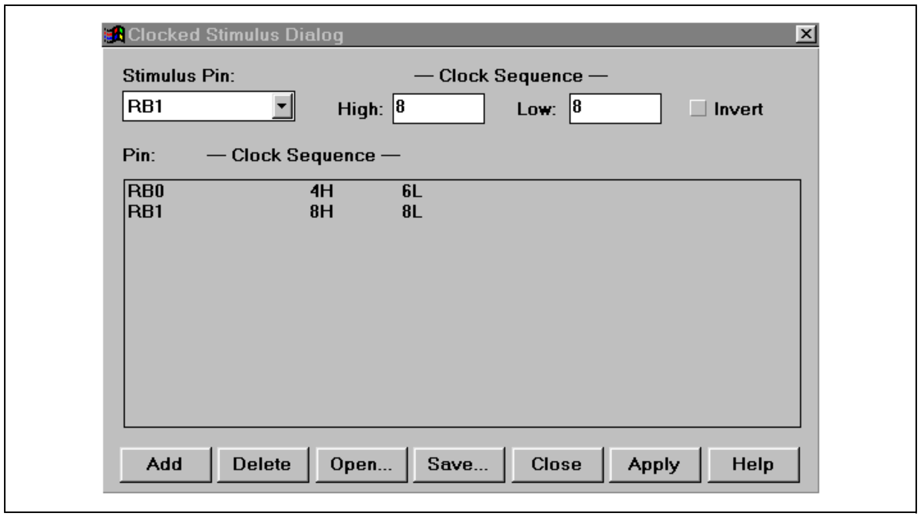
Figure 6.15: File Register Window

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 MPLAB-SIM executes.

6.9.4 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.



Clocked Stimulus Dialog

Stimulus Pin: RB1 — Clock Sequence — High: 8 Low: 8 Invert

Pin:	— Clock Sequence —	
RB0	4H	6L
RB1	8H	8L

Figure 6.16: Clock Stimulus

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 6.16, 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**.

6.10 12-Bit Core Device Simulator Issues

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

6.10.1 12-Bit Core Devices

At the time this document was generated, the following list of devices were available in the PIC16C5X family:

- PIC12C508/509
- PIC12CE518/519
- PIC16C52/54/55/56/57/58
- PIC16HV540
- PIC16C505

The above list implies all device variants; i.e., ROM versions (PIC16CR5X), and device revisions (PIC16C5XA).

6.10.2 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.

6.10.3 CPU Model

6.10.3.1 Reset and Sleep Conditions

All reset conditions are supported by the MPLAB-SIM simulator.

An 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.

6.10.3.2 Watchdog Timer

The Watchdog timer is fully simulated in the MPLAB-SIM simulator. Because it is fuse-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).

6.10.4 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 ‘T0PRE’ symbol.

Note: Because the MPLAB-SIM simulator executes on instruction cycle boundaries, resolutions below 1 T_{CY} cannot be simulated.

6.11 14-Bit Core Device Simulator Issues

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

6.11.1 14-Bit Core Devices

At the time this document was generated, the following devices were available in the 14-Bit Core family:

- PIC12C671/672
- PIC12CE673/674
- PIC140000
- PIC16C62/62A/62B/63/63A/64A/65A/65B/66/67
- PIC16C71/72/72A/73A/73B/74A/74B/76/77
- PIC16C554/558
- PIC16C620/621/622
- PIC16C642/662
- PIC16C710/711/715
- PIC16C712/716
- PIC16C717
- PIC16C770/771
- PIC16C773/774
- PIC16C923/924
- PIC16CE623/624/625
- PIC16F83/84/84A
- PIC16F872/873/874/876/877

The above list implies all device variants; i.e., ROM versions (PIC16CRXX), and device revisions (PIC16CXXA).

6.11.2 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.

6.11.3 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

6.11.4 CPU Model

6.11.4.1 Reset Conditions

All reset conditions are supported by the MPLAB-SIM simulator.

An $\overline{\text{MCLR}}$ 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.

6.11.4.2 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 the processor up 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.

6.11.4.3 Watchdog Timer

The Watchdog timer is fully simulated in the MPLAB-SIM simulator. Because it is fuse-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).

6.11.5 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
- SPIPRE – Prescaler for SPI
- SSPSR – SSP Shift register

6.11.6 Peripherals

6.11.6.1 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)

6.11.6.2 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 T_{CY} and a minimum low time of 1 T_{CY} due to stimulus requirements. The prescaler for Timer0 is made accessible as T0PRE.

6.11.6.3 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.

6.11.6.4 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.

6.11.6.5 CCP1 and CCP2

CAPTURE

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

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_{CY} only) are not supported in this version of the MPLAB-SIM simulator.

6.11.6.6 SSP

The Synchronous Serial Port is supported in SPI 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^2C™ mode.

6.11.6.7 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.

6.11.6.8 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.

6.12 16-Bit Core Device Simulator Issues

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

6.12.1 16-Bit Core Devices

At the time this document was generated, the following list of devices were available in the 16-Bit Core family:

- PIC17C42/43/44
- PIC17C752/756
- PIC17C762/766

The above list implies all device variants; i.e., ROM versions (PIC17CRXX), and device revisions (PIC17CXXA).

6.12.2 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:

- $\overline{\text{MCLR}}$
- 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.

6.12.3 Interrupts

MPLAB-SIM 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

6.12.4 CPU Model

6.12.4.1 Reset Conditions

All reset conditions are supported by the MPLAB-SIM simulator.

An $\overline{\text{MCLR}}$ 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.

6.12.4.2 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 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.

6.12.4.3 Watchdog Timer

The Watchdog Timer is fully simulated in the MPLAB-SIM simulator. Because it is fuse-selectable and fuse-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).

6.12.5 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:

- T0PRE (Prescaler for Timer0)
- WDTPRE (Prescaler for WDT)

6.12.6 Peripherals

6.12.6.1 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

6.12.6.2 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 T0PRE. It can be watched and modified.

6.12.6.3 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.

6.12.6.4 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.

6.12.6.5 PWM

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

6.12.7 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.

6.13 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).

6.13.1 16-Bit Core Devices

At the time this document was generated, the following list of devices were available in the Enhanced 16-Bit Core family:

- PIC18C242
- PIC18C252
- PIC18C442
- PIC18C452

The above list implies all device variants; that is, ROM versions (PIC18CRXX) and device revisions (PIC18CXXA). 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.

6.13.2 I/O Pins

The Enhanced 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:

- $\overline{\text{MCLR}}$
- 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.

6.13.3 Interrupts

MPLAB-SIM 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

6.13.4 CPU Model

6.13.4.1 Reset Conditions

All reset conditions are supported by the MPLAB-SIM simulator.

An $\overline{\text{MCLR}}$ 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 Watchdog timer 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.

6.13.4.2 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.

6.13.4.3 Watchdog Timer

The Watchdog timer is fully simulated in the MPLAB-SIM simulator. Because it is fuse-selectable and fuse-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).

6.13.5 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:

- T0PRE (Prescaler for Timer0)
- WDTPRE (Prescaler for WDT)

6.13.6 Peripherals

6.13.6.1 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.

6.13.6.2 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_{CY} and a minimum low time of 1 T_{CY} due to the stimulus file requirements. The prescaler for Timer0 is made accessible as T0PRE. It can be watched and modified.

6.13.6.3 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_{CY} and a minimum low time of 1 T_{CY} due to the stimulus file requirements.

6.13.6.4 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_{CY} and a minimum low time of 1 T_{CY} due to the stimulus file requirements.

6.13.6.5 PWM

Both PWM outputs are supported (resolution greater than 1 T_{CY} only) are supported in this version of the MPLAB-SIM simulator.

Chapter 7. MPLAB IDE Toolbar and Menu Options

7.1 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.

7.2 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

7.3 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 7.1 shows a maximized desktop.

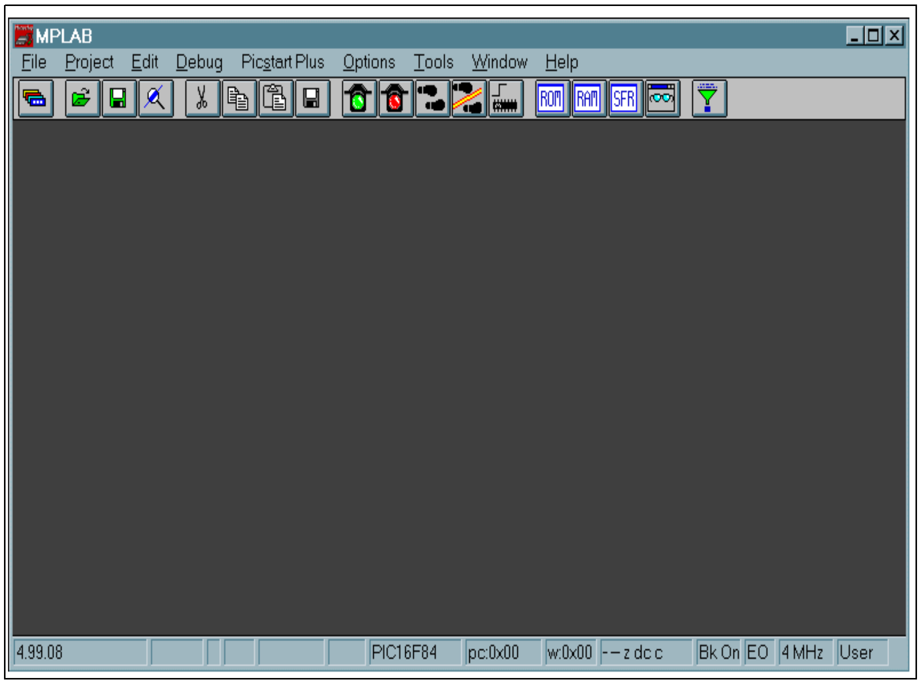

MPLAB
File Project Edit Debug Picstart Plus Options Tools Window Help
ROM RAM SFR
4.99.08 PIC16F84 pc:0x00 w:0x00 --z dc c Bk On EO 4 MHz User

Figure 7.1: 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 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.

7.3.1 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 C: 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 7.8.5.1.2 for information.

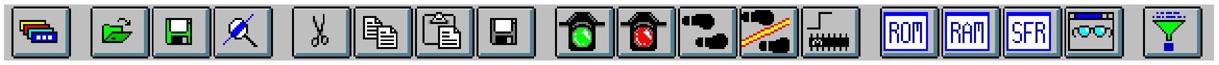


Figure 7.2: MPLAB IDE Debug Toolbar

7.3.2 Status Bar

The table below describes the information presented in the status bar (Figure 7.3). Refer to Appendix C: MPLAB IDE Toolbar and Status Bar Definitions for a description of the symbols on the status bar.

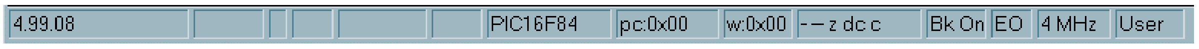


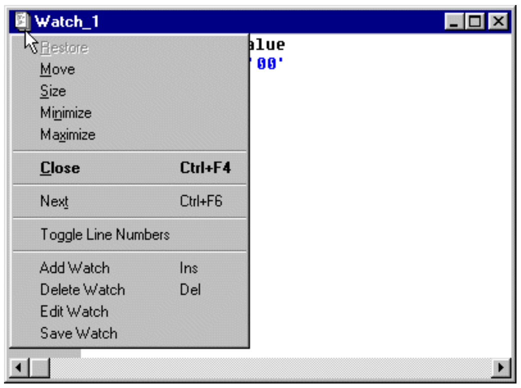
Figure 7.3: Status Bar

7.3.3 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



7.4 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)

7.4.1 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*.

7.4.2 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.

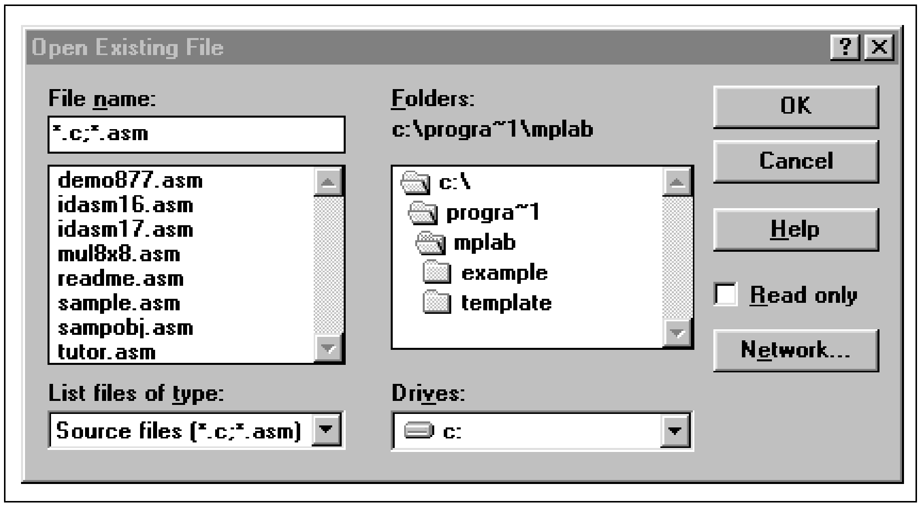


Figure 7.4: Open Existing File

1. Use the Drive and Directory list boxes to select the disk drive and the directory.
2. Select the files you want to open in the File Name list box.

   To add 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.

   OR

   Type the name of the files 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.

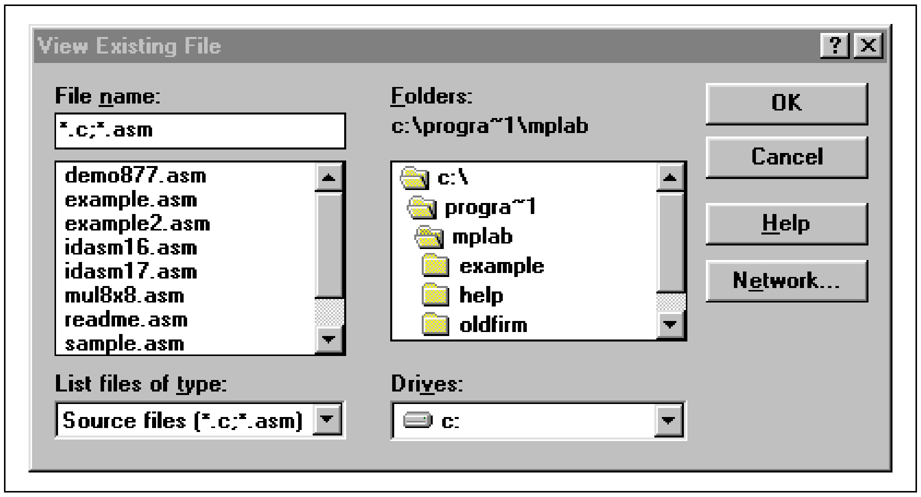
The "List Files of Type" list at lower left allows you to restrict the files shown in the list to those matching specific filename patterns. For example, "*.ASM" will list all files with the suffix ".ASM."

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

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

7.4.3 View File

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.



File name: *.c;*.asm

Folders: c:\progra~1\mplab

demo877.asm
example.asm
example2.asm
idasm16.asm
idasm17.asm
mul8x8.asm
readme.asm
sample.asm

c:\
progra~1
mplab
example
help
oldfirm

List files of type: Source files (*.c;*.asm)

Drives: c:

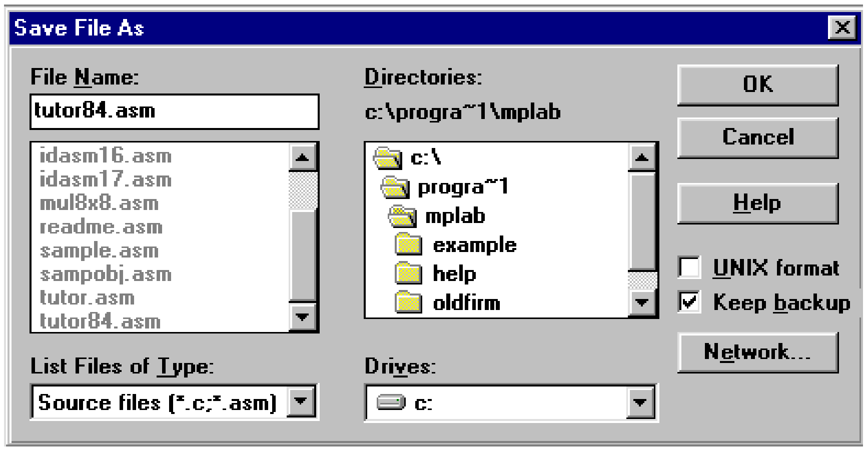
7.4.4 Save File

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 Drive and Directory list boxes to select the disk drive and the directory 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.



Save File As

File Name: tutor84.asm

idasm16.asm
idasm17.asm
mul8x8.asm
readme.asm
sample.asm
sampobj.asm
tutor.asm
tutor84.asm

Directories: c:\progra~1\mplab

c:\
progra~1
mplab
example
help
oldfirm

List Files of Type: Source files (*.c;*.asm)

Drives: c:

[ ] UNIX format
[✓] Keep backup

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.

7.4.5 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.

7.4.6 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.

7.4.6.1 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 my exporting program memory.

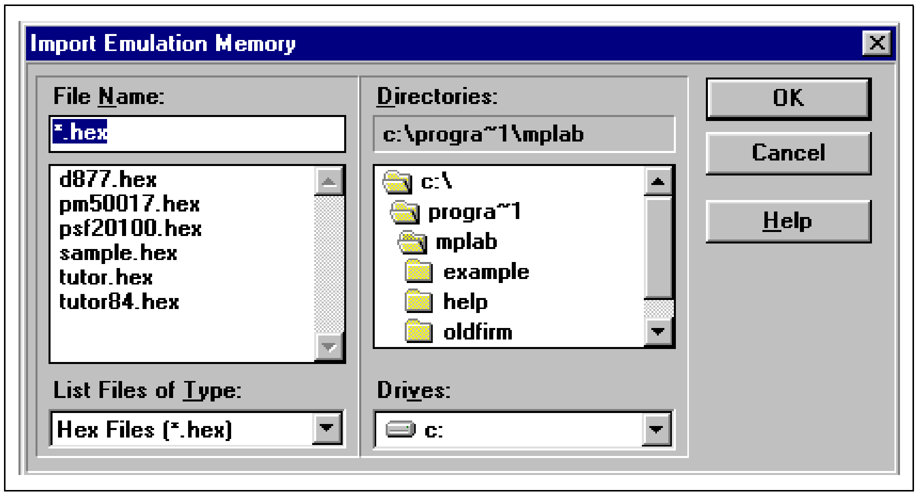


Figure 7.5: Import Emulation Memory Dialog

7.4.6.1.1. 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 (INHX32). 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.

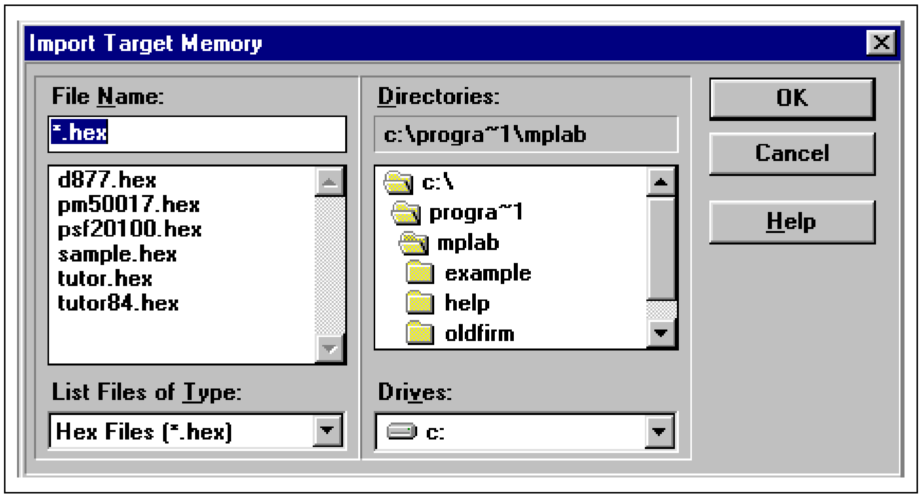
7.4.6.1.2. All Other PICmicro® MCU Devices

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 (INHX8M). 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.

7.4.6.2 Import to Target Memory

For PIC17CXXX or PIC18CXXX devices 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.



File Name: *.hex

d877.hex
pm50017.hex
psf20100.hex
sample.hex
tutor.hex
tutor84.hex

Directories: c:\progra~1\mplab

c:\
progra~1
mplab
example
help
oldfirm

List Files of Type: Hex Files (*.hex)

Drives: c:

Figure 7.6: Import Target Memory Dialog (MPLAB-ICE)

7.4.6.3 Copy from Target Memory

For PIC17CXXX or PIC18CXXX devices 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 copy target memory into a data file, select *File > Import > Copy from Target Memory* to copy the data through the emulator into a PC file. The resulting file may be debugged using MPLAB IDE.

7.4.7 Export

7.4.7.1 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 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.

7.4.7.1.1. MPLAB-ICE Trace Buffer

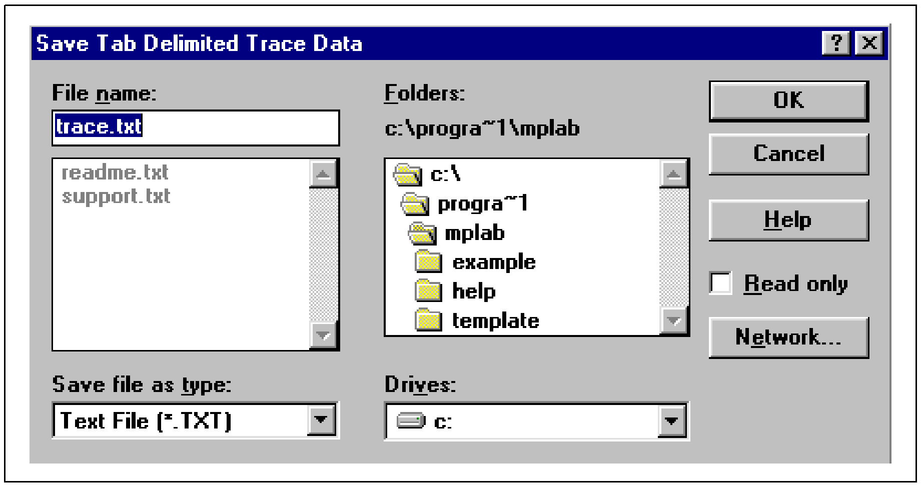
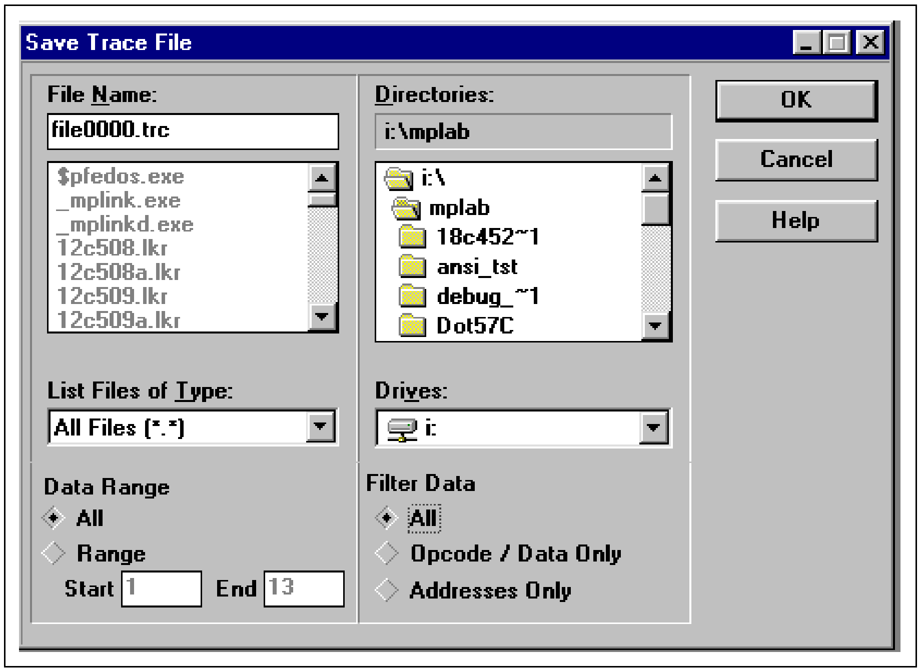


Figure 7.7: Export Trace Buffer Dialog (MPLAB-ICE)

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 directory 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.

7.4.7.1.2. MPLAB-SIM Trace Buffer


Save Trace File

File Name: file0000.trc

$pfedos.exe
_mplink.exe
_mplinkd.exe
12c508.lkr
12c508a.lkr
12c509.lkr
12c509a.lkr

Directories: i:\mplab

i:\
mplab
18c452~1
ansi_tst
debug_~1
Dot57C

List Files of Type: All Files (*.*)

Drives: i:

Data Range
[✓ All]
Range
Start 1 End 13

Filter Data
[✓ All]
Opcode / Data Only
Addresses Only

Figure 7.8: Export Trace Buffer Dialog (MPLAB-SIM)

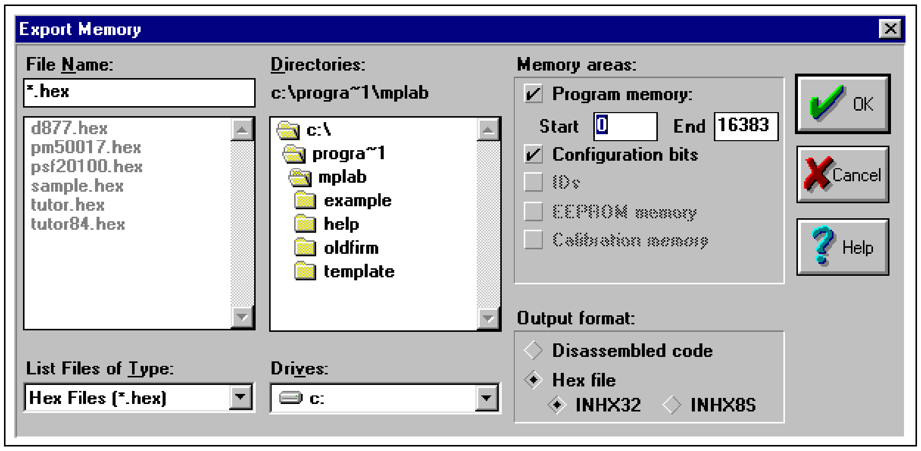
Range	Select the range (0 to 8191) of the trace buffer that you want to save. Enter the desired value for the Start line number and for the End line number.
Filter Data: All	Writes the complete trace buffer to the selected file.
Filter Data: Opcode (Data Only)	Saves Opcode / Data only. (For PIC17CXXX external read/write cycles) Available only with the MPLAB-ICE
Filter Data: Address Only	Saves Address only.

7.4.7.2 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.



Export Memory

File Name: *.hex

d877.hex
pm50017.hex
psf20100.hex
sample.hex
tutor.hex
tutor84.hex

Directories: c:\progra~1\mplab

c:\
progra~1
mplab
example
help
oldfirm
template

List Files of Type: Hex Files (*.hex)

Drives: c:

Memory areas:
[✓] Program memory: Start 0 End 16383
[✓] Configuration bits
[ ] IDs
[ ] EEPROM memory
[ ] Calibration memory

Output format:
( ) Disassembled code
(●) Hex file: (●) INHX32 ( ) INHX8S

Figure 7.9: 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.
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 INHX8X format.

7.4.8 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.

7.4.8.1 Print Current File Options

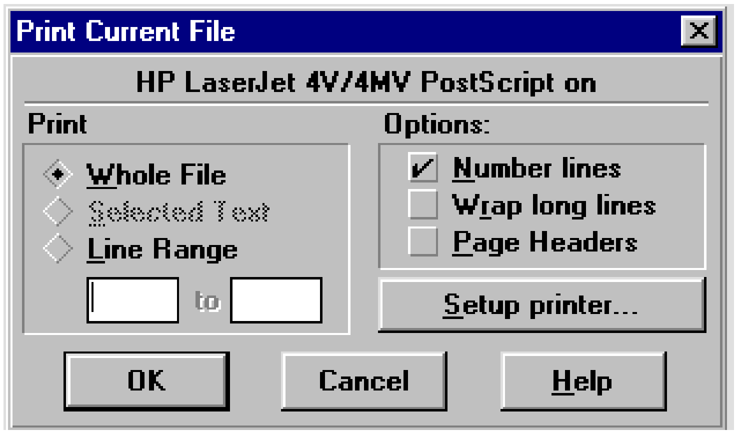

Print Current File
HP LaserJet 4V/4MV PostScript on
Print: [✓ Whole File] Selected Text, Line Range ___ to ___
Options: [✓ Number lines] Wrap long lines, Page Headers

Figure 7.10: 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.
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.

7.4.9 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.

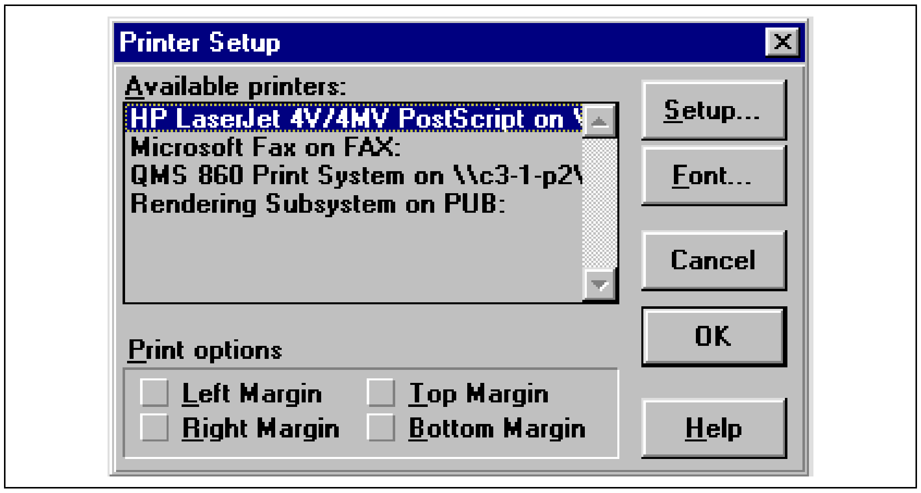


Figure 7.11: Printer Setup Dialog

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

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.

7.4.10 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.

7.4.11 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 7.8.5.3.

7.5 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.

7.6 Edit Menu

7.6.1 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.

7.6.1.1 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 7.8.5.4.

7.6.1.2 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.

7.6.1.3 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.

7.6.1.4 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.

7.6.1.5 Select All

Highlights the entire contents of the current window.

7.6.1.6 Select Word – Left Mouse Button Double-Click

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

7.6.1.7 Delete Line – Ctrl+Shift+K

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

7.6.1.8 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.

7.6.1.9 Goto Line – Ctrl+G

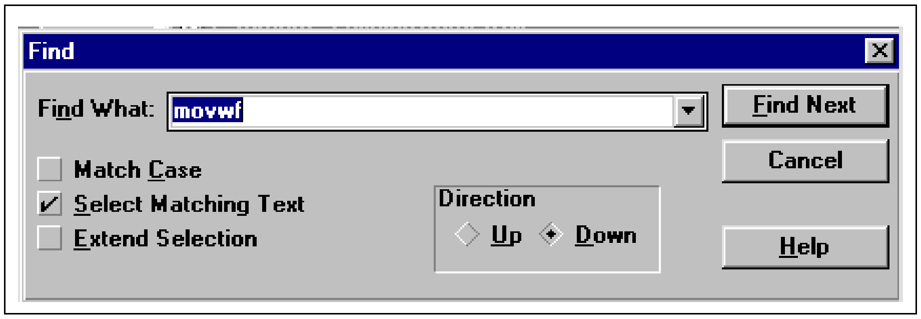
Moves the cursor to the start of a specific line.

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

7.6.1.10 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*.



Find What: movwf

[✓] Select Matching Text

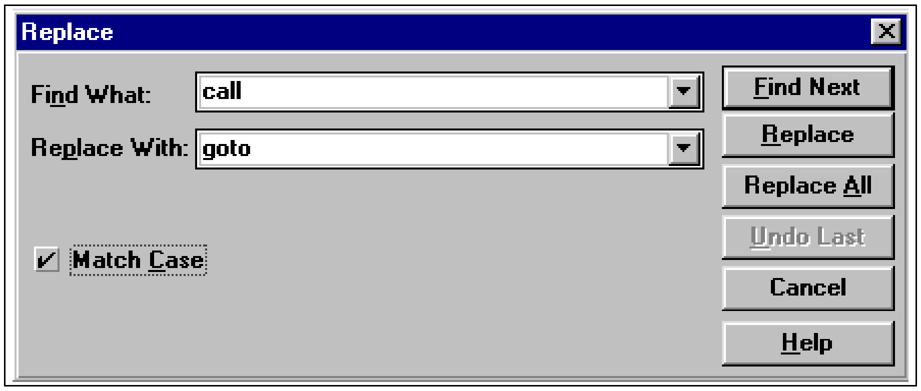
Direction: [✓ Down]

Figure 7.12: 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.

7.6.1.11 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.



Find What: call
Replace With: goto
✓ Match Case

Figure 7.13: 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.

7.6.1.12 Repeat Find – Shift+F3

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

7.6.1.13 Repeat Replace – Shift+F4

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

7.6.1.14 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.

7.6.2 Template Options

7.6.2.1 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**.

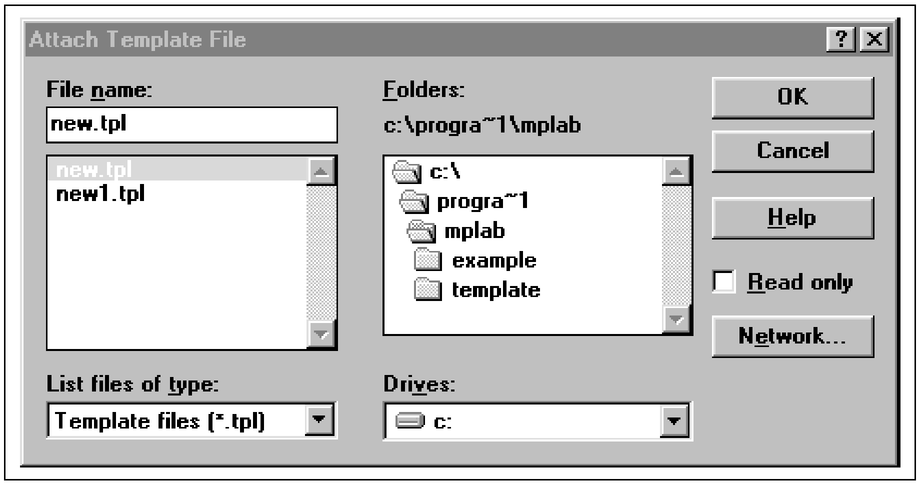
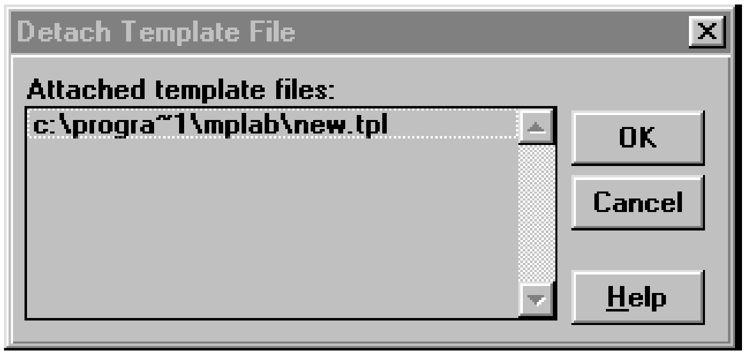


Figure 7.14: Attach Template File Dialog

7.6.2.2 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.



c:\progra~1\mplab\new.tpl

Figure 7.15: Detach Template File Dialog

7.6.2.3 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 directory. 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.

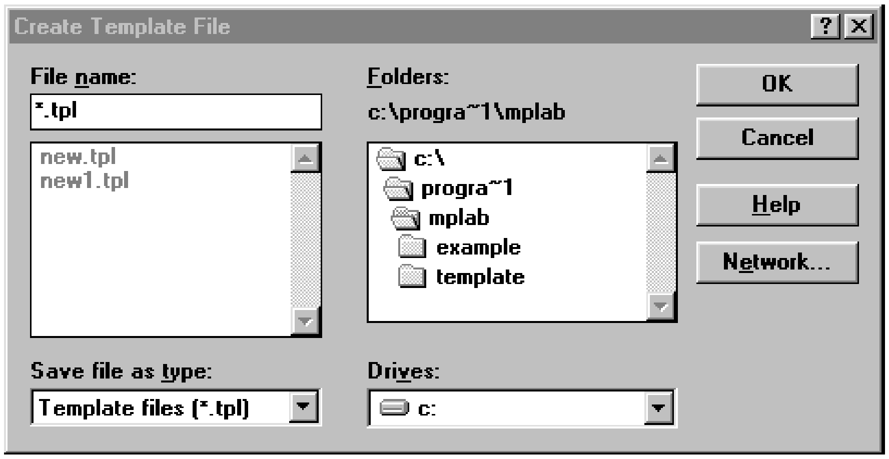


Figure 7.16: Create Template File Dialog

7.6.2.4 Save Template File

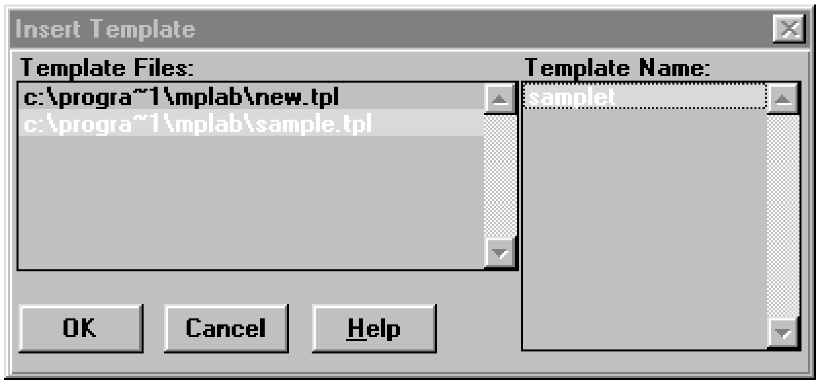
When you edit, create, or delete a template, MPLAB Editor changes a copy of the template in memory, which will be erased when you exit MPLAB IDE. These changes are not saved to disk until you use the *Edit > Template > Save File* command.

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.

7.6.2.5 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 7.6.2.1).

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**.



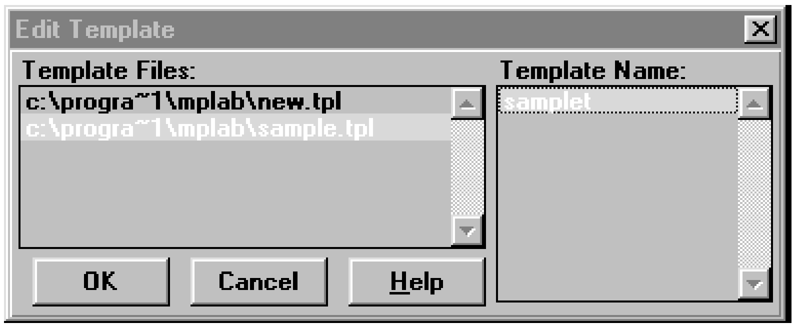
c:\progra~1\mplab\new.tpl
c:\progra~1\mplab\sample.tpl
sample1

Figure 7.17: Insert Template Dialog

7.6.2.6 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 7.6.2.1) 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.



c:\progra~1\mplab\new.tpl
c:\progra~1\mplab\sample.tpl
sample1

Figure 7.18: Edit Template Dialog

7.6.2.7 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 7.6.2.11). 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 7.6.2.8 and Section 7.6.2.9).

7.6.2.8 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.

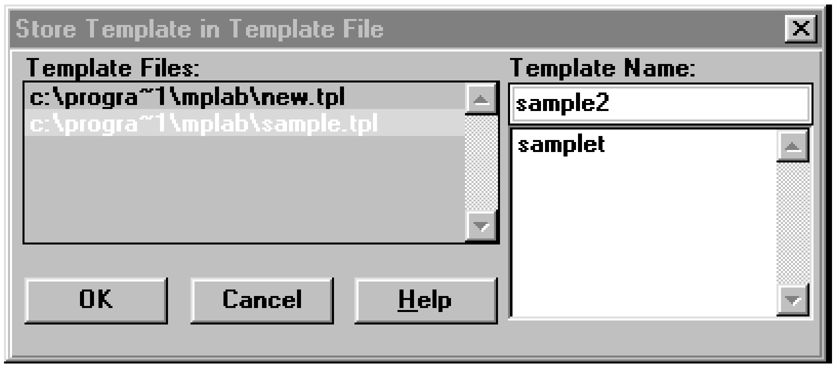
Note: Because the Store command only writes to memory, the changes will be discarded when you exit MPLAB IDE. To save your template changes for future use, use *Edit > Template > Save File*. When you exit MPLAB IDE, you may be prompted if you wish to save changes to the template (.TPL) file. Answer **Yes**.

7.6.2.9 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**.



c:\progra~1\mplab\new.tpl
c:\progra~1\mplab\sample.tpl

sample2

samplet

Figure 7.19: Store Template Dialog

7.6.2.10 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 7.6.2.1) 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**.

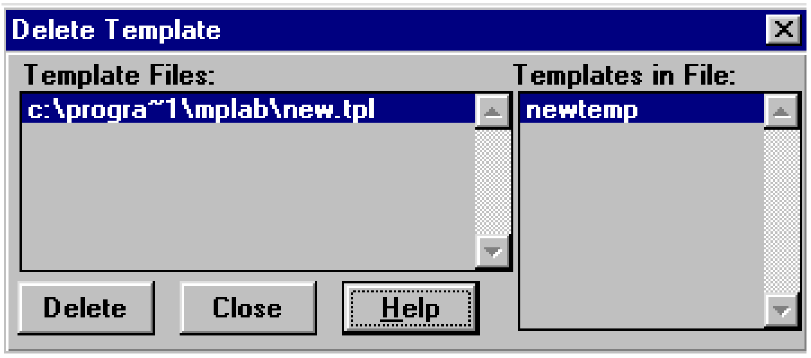


Figure 7.20: Delete Template Dialog

7.6.2.11 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.

7.6.2.12 Find Template Mark

After you have inserted the template as described in Section 7.6.2.5, 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.

7.6.3 Text Options

This menu shows a list of text-related commands.

7.6.3.1 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.

7.6.3.2 Uppercase

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

7.6.3.3 Lowercase

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

7.6.3.4 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.

7.6.3.5 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.

7.7 Debug Menu

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

7.7.1 Run

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

7.7.1.1 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.

7.7.1.2 Reset (F6)

Debug > Run > Reset issues a reset sequence to the target processor. This issues an $\overline{\text{MCLR}}$ to reset the Program Counter to the reset vector. If the processor is running it will continue running from the reset vector address.

7.7.1.3 Halt (F5)

Debug > Run > Halt forces the processor into the halt state. When you click **Halt**, the processor is forced into a Halt state (Program Counter is stopped) and the processor status information is updated.

7.7.1.4 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.

7.7.1.5 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.

7.7.1.6 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. While the processor runs in real time, MPLAB IDE ignores the Step button.

7.7.1.7 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.

7.7.1.8 Update All Registers

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

7.7.1.9 Change Program Counter

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

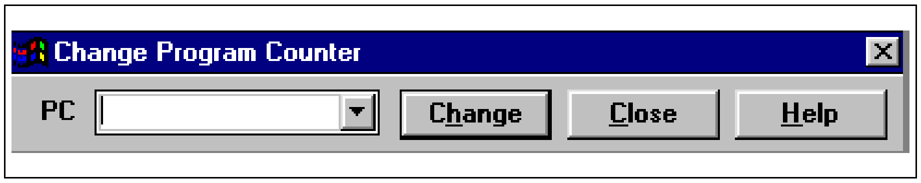


Figure 7.21: Change Program Counter Dialog

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.

7.7.2 Execute

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

7.7.2.1 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.

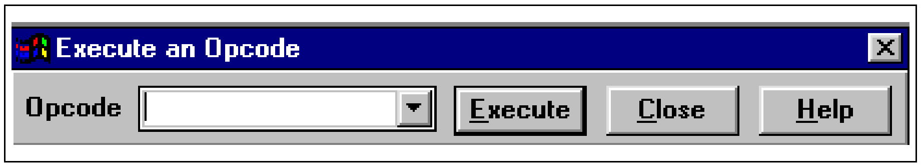


Figure 7.22: Execute an Opcode Dialog

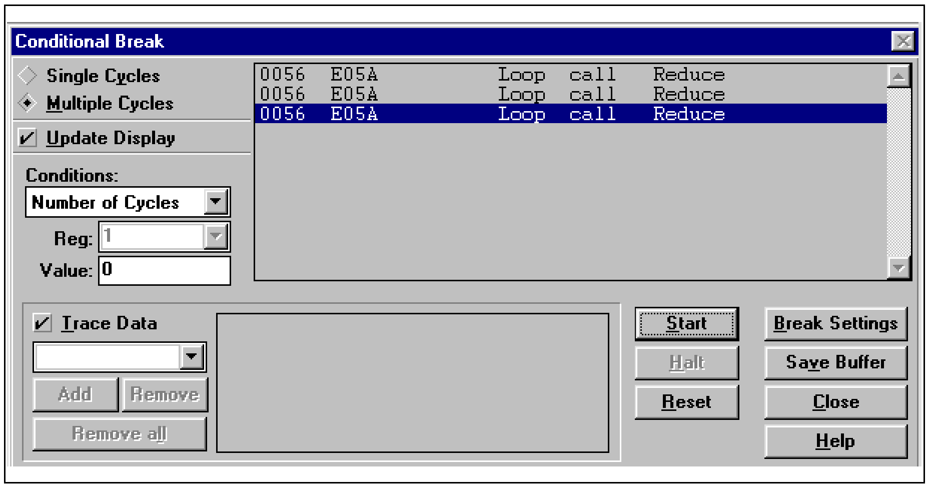
Opcode	Enter the instruction as an opcode (in hex digits) or enter a symbolic instruction (such as ADDWF 0x19). Click the Opcode list to display the last eight commands. After executing a command, MPLAB IDE highlights the command so you can type in a new instruction. (MPLAB IDE tracks the instructions 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.

7.7.2.2 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 or MPLAB-ICD development modes. For MPLAB-ICE, refer to the Complex Trigger dialog for this functionality.

For additional information on conditional breaks, see Section 6.8.



```
0056  E05A        Loop  call   Reduce
0056  E05A        Loop  call   Reduce
0056  E05A        Loop  call   Reduce
```

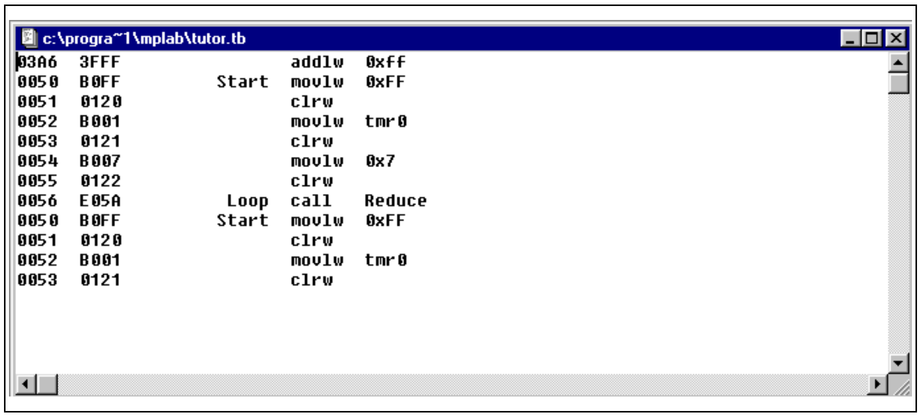
Figure 7.23: Conditional Break Dialog

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:

1. User Halt When processor is running, click **Halt** to stop the processor.
2. Number of Cycles Enter the number of cycles in the **Value** box.
3. 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

	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.



```
03A6  3FFF               addlw  0xff
0050  B0FF        Start  movlw  0xFF
0051  0120               clrw
0052  B001               movlw  tmr0
0053  0121               clrw
0054  B007               movlw  0x7
0055  0122               clrw
0056  E05A         Loop  call   Reduce
0050  B0FF        Start  movlw  0xFF
0051  0120               clrw
0052  B001               movlw  tmr0
0053  0121               clrw
```

Figure 7.24: ∗.TB File

Close	Exits the Conditional Break dialog.

7.7.3 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 directory as the current project.

Simulator functions are:

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

For a detailed discussion of simulator stimulus, see Section 6.9.

7.7.4 Center Debug Location

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

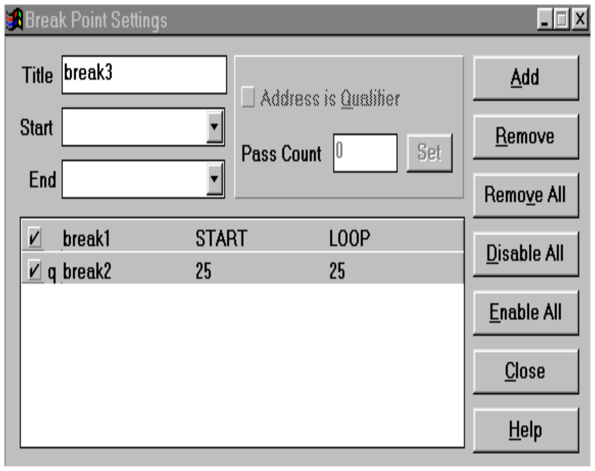
This function works on the Source Code Window, the Program Memory Window, and the Absolute Listing Window.

7.7.5 Break Point Settings

Select *Debug > Break Settings* to set breakpoints and breakpoint qualifiers.

For more functional information on break points, see Section 6.7.

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).



✓	break1	START	LOOP
✓ q	break2	25	25

Figure 7.25: Break Point Settings Dialog

7.7.5.1 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.

Note: When using the In-Circuit Debugger (ICD), only one address can be set to a break point in the Breakpoint Settings dialog.

7.7.5.2 Saving Break Point Settings

Break points are saved as part of the project.

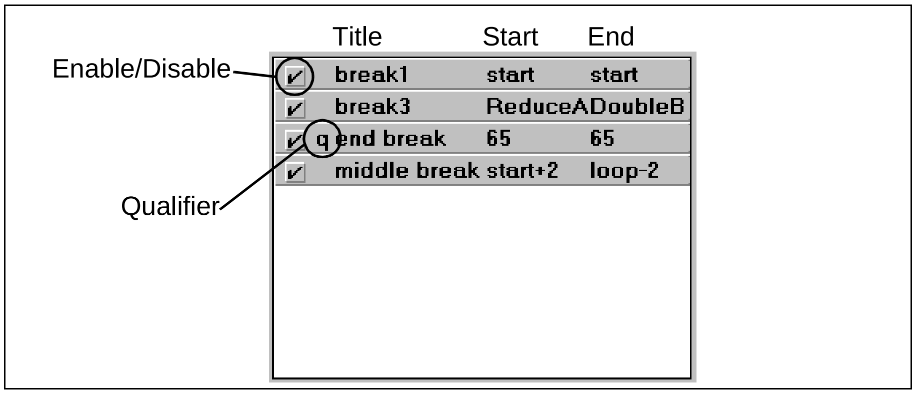
Title	Enter a unique title (up to 32 characters) for each break point range. MPLAB IDE accepts the underscore character but does not allow spaces. MPLAB IDE automatically enters a default, unique title if you choose not to enter a title. The break point range requires a title.

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.

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
✓	break1	start	start
✓	break3	ReduceADoubleB	
✓ q	end break	65	65
✓	middle break	start+2	loop-2

Figure 7.26: Break Point List

<u>Enable/Disable</u>	This switch enables/disables the break point range in program memory.
<u>Qualifier</u>	Qualifier point designator. When a range item displays the letter q, the address range is a Pass Count address.
<u>Title, Start, End</u>	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.

7.7.5.3 Break Point Qualifiers

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

1. Select a breakpoint qualifier from the dropdown list.
2. Enter a pass count qualifier for the breakpoint.

   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.

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

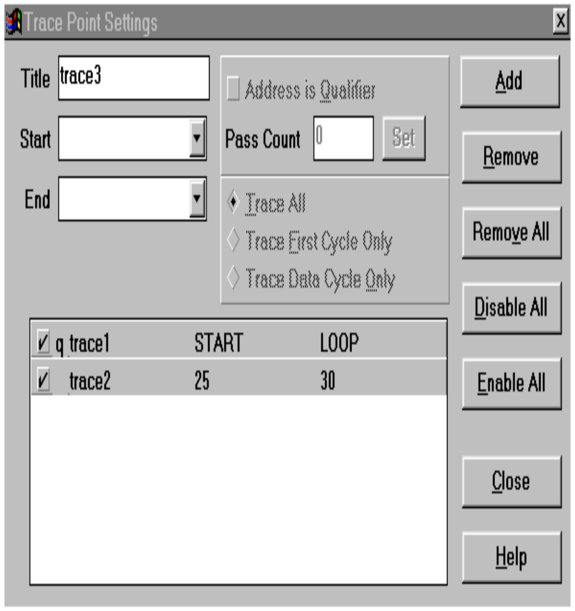
Address is Qualifier	You can assign a Pass Count qualifier address to either break logic or trace logic. You must have an element in the list selected. With an element selected, the dialog will enable the Address is Qualifier check box. When you check Address is Qualifier, you are assigning the pass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

7.7.6 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 or MPLAB-ICD development modes.

For more functional information on trace points, see Section 6.7.



Title: trace3

Pass Count: 0

✓	q trace1	START	LOOP
✓	trace2	25	30

Figure 7.27: 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 or MPLAB-ICD development modes. The MPLAB-ICE trace may be configured through the Complex Trigger dialog.

7.7.6.1 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.

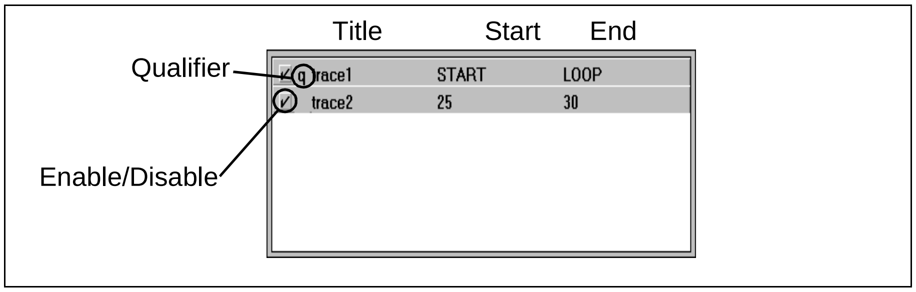
Note: MPLAB-ICD does not have trace capability.

7.7.6.2 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.

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:



✓ q	trace1	START	LOOP
✓	trace2	25	30

Figure 7.28: 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.

7.7.6.3 Trace Point Qualifiers

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

1. Select a trace point qualifier from the dropdown 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.

   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**.

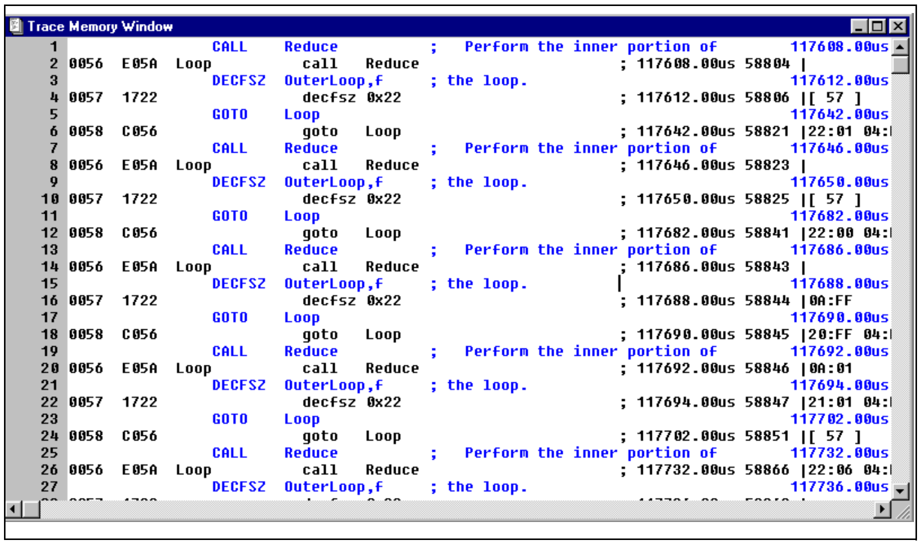
7.7.6.4 Global Trace Point Environment Options

These settings apply to the PICMASTER emulator and the simulator (MPLAB-SIM).

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 7.29. For information on emulator display, see the documentation for that emulator.

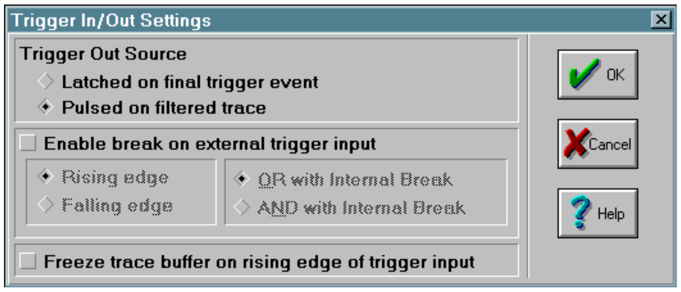


```
 1                CALL    Reduce          ;   Perform the inner portion of       117608.00us
 2 0056  E05A  Loop         call    Reduce                  ; 117608.00us 58804 |
 3                DECFSZ  OuterLoop,f     ; the loop.                            117612.00us
 4 0057  1722               decfsz 0x22                     ; 117612.00us 58806 |[ 57 ]
 5                GOTO    Loop                                                   117642.00us
 6 0058  C056               goto    Loop                    ; 117642.00us 58821 |22:01 04:
 7                CALL    Reduce          ;   Perform the inner portion of       117646.00us
 8 0056  E05A  Loop         call    Reduce                  ; 117646.00us 58823 |
 9                DECFSZ  OuterLoop,f     ; the loop.                            117650.00us
10 0057  1722               decfsz 0x22                     ; 117650.00us 58825 |[ 57 ]
11                GOTO    Loop                                                   117682.00us
12 0058  C056               goto    Loop                    ; 117682.00us 58841 |22:00 04:
13                CALL    Reduce          ;   Perform the inner portion of       117686.00us
14 0056  E05A  Loop         call    Reduce                  ; 117686.00us 58843 |
15                DECFSZ  OuterLoop,f     ; the loop.                            117688.00us
16 0057  1722               decfsz 0x22                     ; 117688.00us 58844 |0A:FF
17                GOTO    Loop                                                   117690.00us
18 0058  C056               goto    Loop                    ; 117690.00us 58845 |20:FF 04:
19                CALL    Reduce          ;   Perform the inner portion of       117692.00us
20 0056  E05A  Loop         call    Reduce                  ; 117692.00us 58846 |0A:01
21                DECFSZ  OuterLoop,f     ; the loop.                            117694.00us
22 0057  1722               decfsz 0x22                     ; 117694.00us 58847 |21:01 04:
23                GOTO    Loop                                                   117702.00us
24 0058  C056               goto    Loop                    ; 117702.00us 58851 |[ 57 ]
25                CALL    Reduce          ;   Perform the inner portion of       117732.00us
26 0056  E05A  Loop         call    Reduce                  ; 117732.00us 58866 |22:06 04:
27                DECFSZ  OuterLoop,f     ; the loop.                            117736.00us
```

Figure 7.29: Trace Memory Window

7.7.7 Trigger In/Out Settings

Select *Debug > Trigger In/Out Settings* to set trigger input and output.



Trigger In/Out Settings

Trigger Out Source
- Latched on final trigger event
- [✓ Pulsed on filtered trace]

Enable break on external trigger input

- [✓ Rising edge]
- Falling edge
- [✓ OR with Internal Break]
- AND with Internal Break

Freeze trace buffer on rising edge of trigger input

Figure 7.30: Trigger In/Out Settings Dialog

This dialog applies to the MPLAB-ICE and PICMASTER emulators.

MPLAB-ICE Options:

- Trigger Out Source
- Enable break on external trigger input
  - Rising edge
  - Falling edge
- Freeze trace buffer on rising edge of trigger input

PICMASTER Options:

- Enable break on external trigger input
  - Rising edge
  - Falling edge
  - OR with Internal Break
  - AND with Internal Break
- Freeze trace buffer on rising edge of trigger input

For more information on setting these emulator options, see either the *MPLAB-ICE User's Guide* or the *PICMASTER User's Guide*.

7.7.8 Trigger Output Points

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

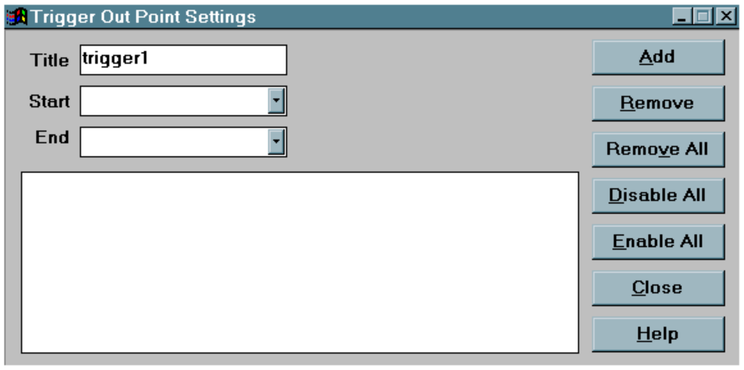
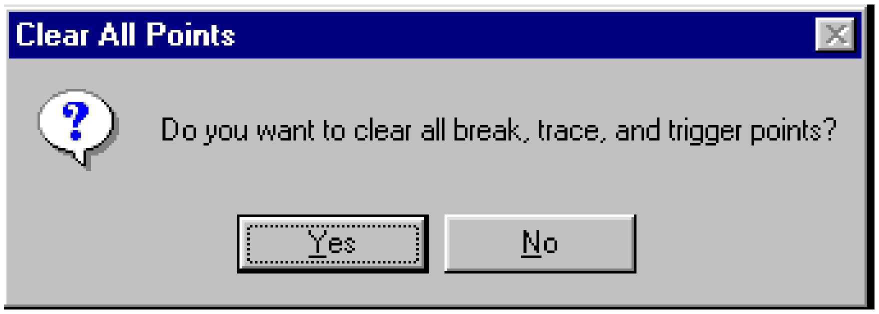


Figure 7.31: Trigger Out Point Settings Dialog

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

7.7.9 Clear All Points

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

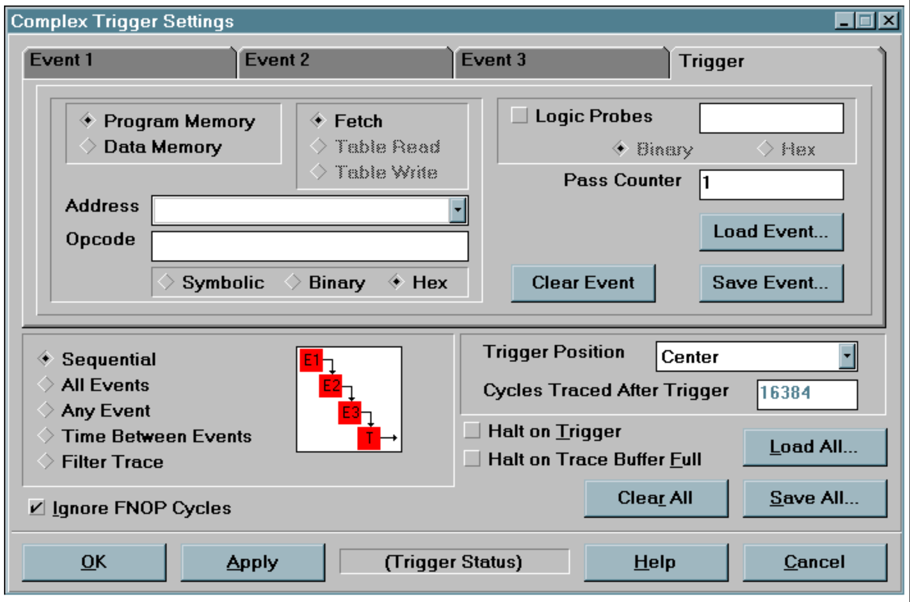


Do you want to clear all break, trace, and trigger points?

Figure 7.32: Clear All Points Message Box

7.7.10 Complex Trigger Settings

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



[✓ Program Memory] Data Memory

[✓ Fetch] Table Read Table Write

Address:

Opcode:

Symbolic Binary [✓ Hex]

Logic Probes: ☐

[✓ Binary] Hex

Pass Counter: 1

[✓ Sequential] All Events Any Event Time Between Events Filter Trace

Trigger Position: Center

Cycles Traced After Trigger: 16384

☐ Halt on Trigger

☐ Halt on Trace Buffer Full

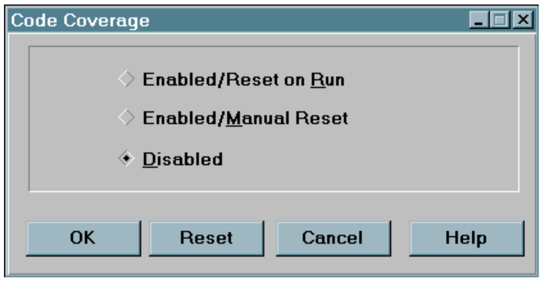
☑ Ignore FNOP Cycles

Figure 7.33: Complex Trigger Settings Dialog

For more information on setting these emulator options, see the *MPLAB-ICE User's Guide*.

7.7.11 Code Coverage

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



Enabled/Reset on Run

Enabled/Manual Reset

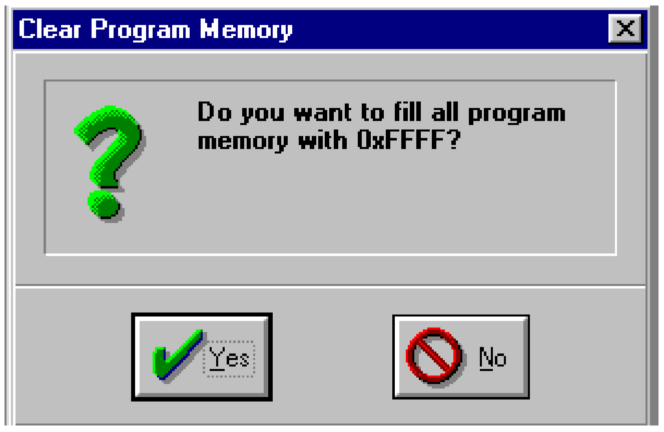
[✓ Disabled]

Figure 7.34: Code Coverage Dialog

For more information on setting these emulator options, see the *MPLAB-ICE User's Guide*.

7.7.12 Clear Program Memory (Ctrl+Shift+F2)

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



Do you want to fill all program memory with 0xFFFF?

Figure 7.35: Clear Program Memory Message Box

7.7.13 System Reset (Ctrl+Shift+F3)

Select *Debug > System Reset* to reset the entire emulator system including the MPLAB-ICE 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 ($\overline{\text{MCLR}}$), 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.

7.7.14 Power-On-Reset (Ctrl+Shift+F5)

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

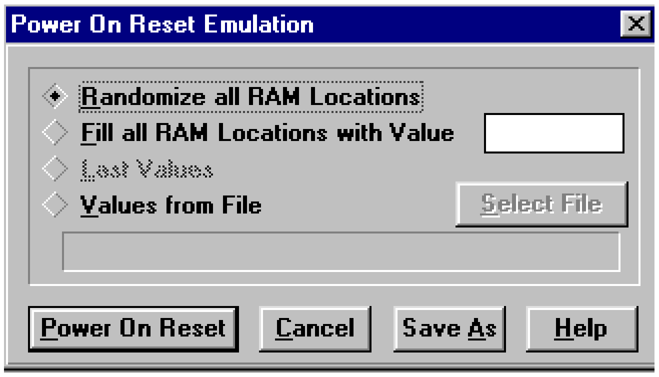


Figure 7.36: Power-On-Reset Dialog

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 V_{DD}. 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 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.

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.

7.8 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 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 device selectively (e.g., program part of program memory, or only configuration bits) or to verify that the device was programmed properly.
Read Device	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).
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 are 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	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.
Erase Program Memory	Sets all bits in the MPLAB IDE Program memory, Calibration memory, and Data memory (if applicable) to “1.”
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 PICSTART Plus. 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.
Erase Program Memory	Sets all bits in the MPLAB IDE Program memory, Calibration memory, and Data memory (if applicable) to "1."
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 Voltages	Sets V_{DDMIN}, V_{DDMAX}, and V_{PP} to their default values for the selected device.

Transfer to PRO MATE	Transfers device information to PRO MATE.
Transfer from PRO MATE	Transfers device information from PRO MATE.
Generate SQTP^{SM} File	Generate an SQTP file for device serialization.
Load SQTP File	Load an SQTP file from disk.
Download PRO MATE Operating System	Download the latest version of the PRO MATE operating system to the programmer.
Establish Communications	Establishes or reestablishes RS-232 communications with the device programmer. Use this if power has been disconnected from the PRO MATE II. This does not reset programming information in the Programmer Status Dialog, configuration bits, or IDs.

7.8.1 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

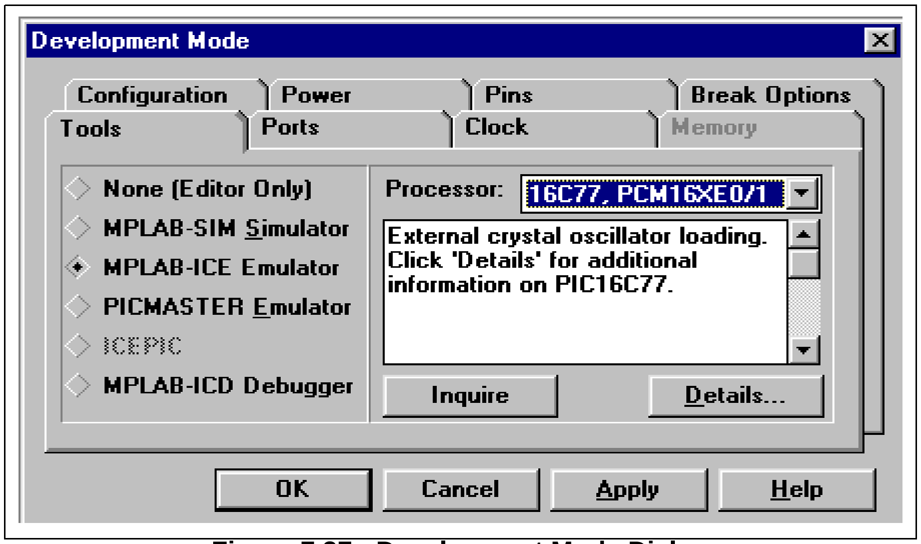
7.8.1.1 Tools

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 or PICMASTER) 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**.



Processor: 16C77, PCM16XE0/1

External crystal oscillator loading. Click 'Details' for additional information on PIC16C77.

Figure 7.37: 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 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	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.

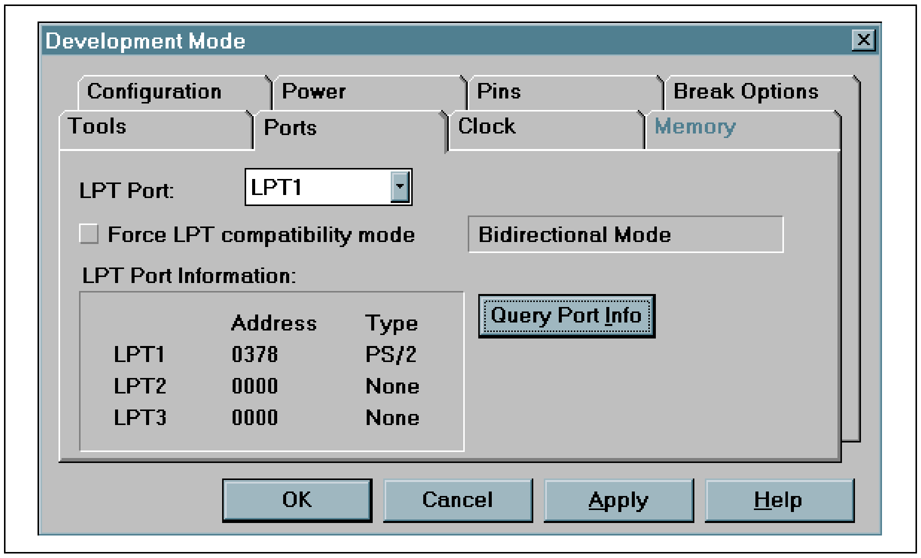
7.8.1.2 Device Ports

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 Emulator:



LPT Port: LPT1

☐ Force LPT compatibility mode   Bidirectional Mode

LPT Port Information:

```
        Address   Type
LPT1    0378      PS/2
LPT2    0000      None
LPT3    0000      None
```

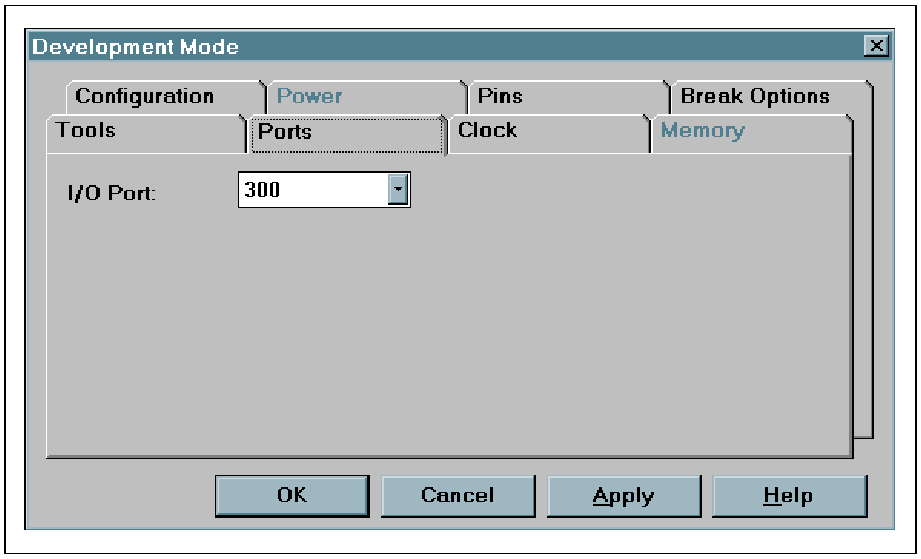
Figure 7.38: Ports Dialog - MPLAB-ICE

LPT Port	Choose the LPT port you will use to communicate with MPLAB-ICE (e.g., LPT1). The LPT port is also called the parallel or printer port.
Force Compatibility Mode	Check this item to force the MPLAB-ICE to communicate in compatibility mode rather than the usual bidirectional mode. The current operational mode appears to the right of this item.
LPT Port Information	Click **Query Port Info** to view complete information on your PC's LPT port configuration.

For more information on MPLAB-ICE communications, refer to the *MPLAB-ICE User's Guide*.

WARNING	**Do not** select the MPLAB-ICE Emulator mode if any other device (i.e., printer, scanner, zip drive) is installed on the parallel port or **permanent damage to that device** may result. See the *MPLAB-ICE User's Guide* (DS51159) for more details.

PICMASTER Emulator:



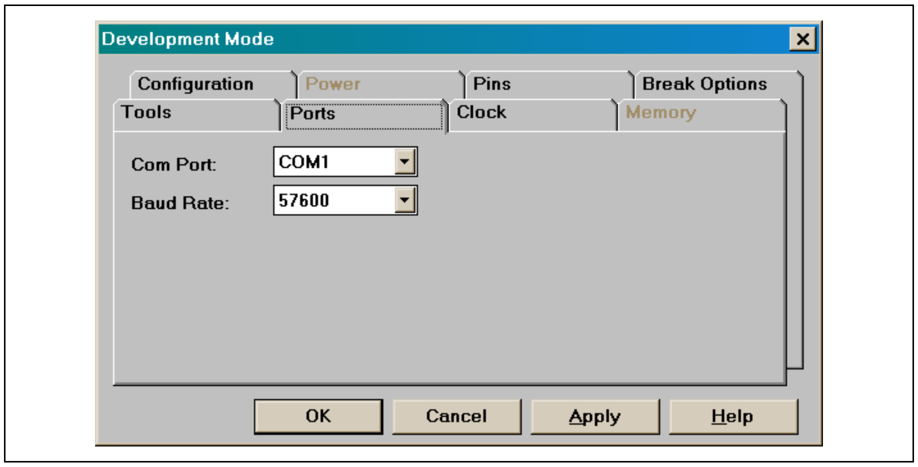
I/O Port: 300

Figure 7.39: Ports Dialog – PICMASTER

I/O Port	Select the I/O Port address that matches the base address your PICMASTER PC Interface Card.

For more information on PICMASTER communications, refer to the *PICMASTER User's Guide*.

ICEPIC Emulator:



Com Port: COM1

Baud Rate: 57600

Figure 7.40: Ports Dialog – ICEPIC

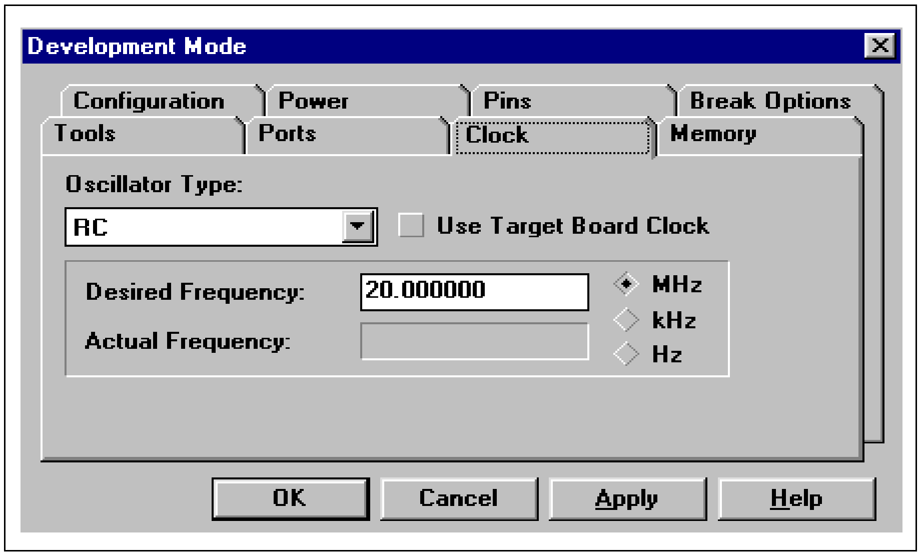
Com Port	Select the Com Port that the ICEPIC is connected to.
Baud Rate	Select the baud rate at which data will travel over the com port.

For more information on ICEPIC communications, refer to the ICEPIC documentation.

7.8.1.3 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**.



Oscillator Type: RC

Use Target Board Clock

Desired Frequency: 20.000000 [✓ MHz] kHz Hz

Actual Frequency:

Figure 7.41: 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 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 only.

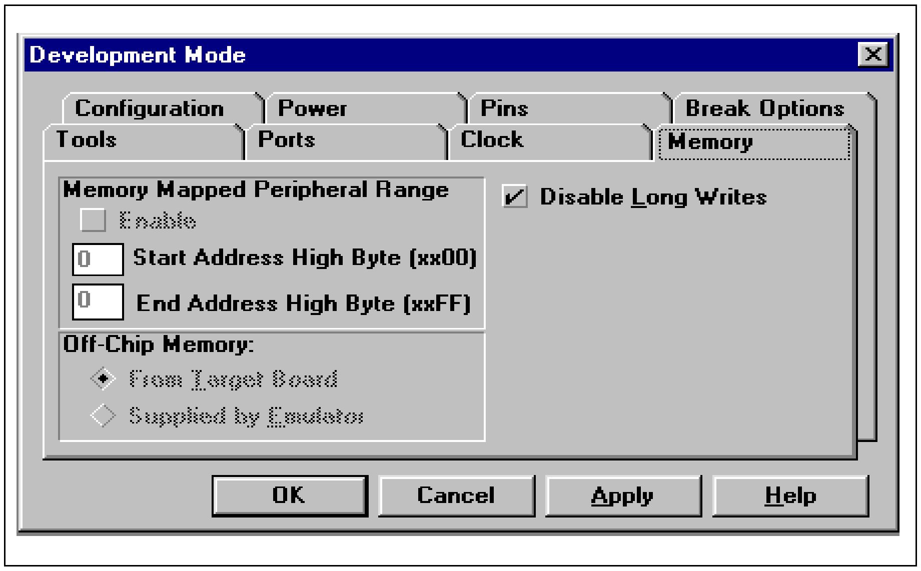
4. Click **Apply** to accept the new setting.

7.8.1.4 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**.



Memory Mapped Peripheral Range
[ ] Enable
0 Start Address High Byte (xx00)
0 End Address High Byte (xxFF)
[✓] Disable Long Writes
Off-Chip Memory:
[● From Target Board]
○ Supplied by Emulator

Figure 7.42: Memory Dialog

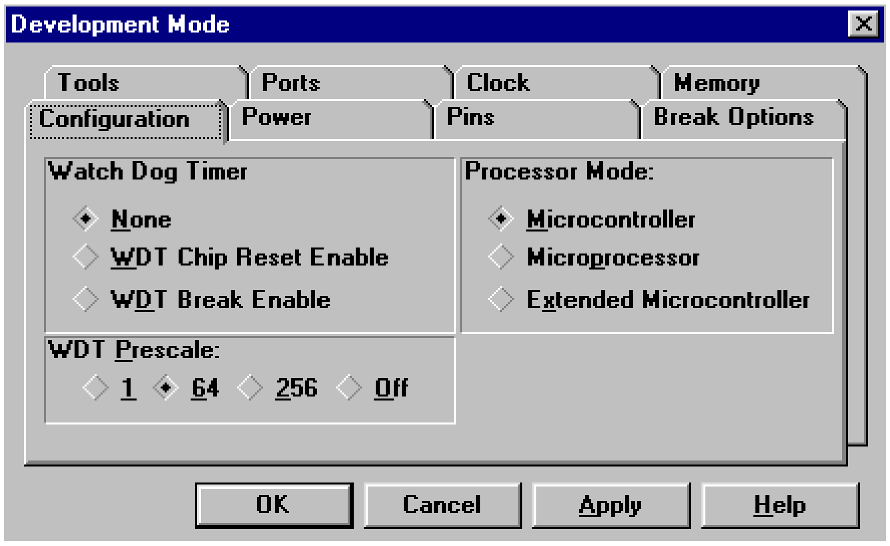
Memory Mapped Peripheral Range	Enable/Disable the memory mapped peripheral range, and specify the starting and ending high byte address. If the End Address High Byte is set to 0xFF, the program counter in MPLAB will get stuck at 0xFFFE while operating in external microcontroller mode. The highest value 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.

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 will wait for an interrupt to exit the long write. These table write instructions should be removed before a device is programmed.

7.8.1.5 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**.



Watch Dog Timer: [✓ None] WDT Chip Reset Enable / WDT Break Enable

WDT Prescale: 1 [✓ 64] 256 Off

Processor Mode: [✓ Microcontroller] Microprocessor / Extended Microcontroller

Figure 7.43: Configuration Dialog

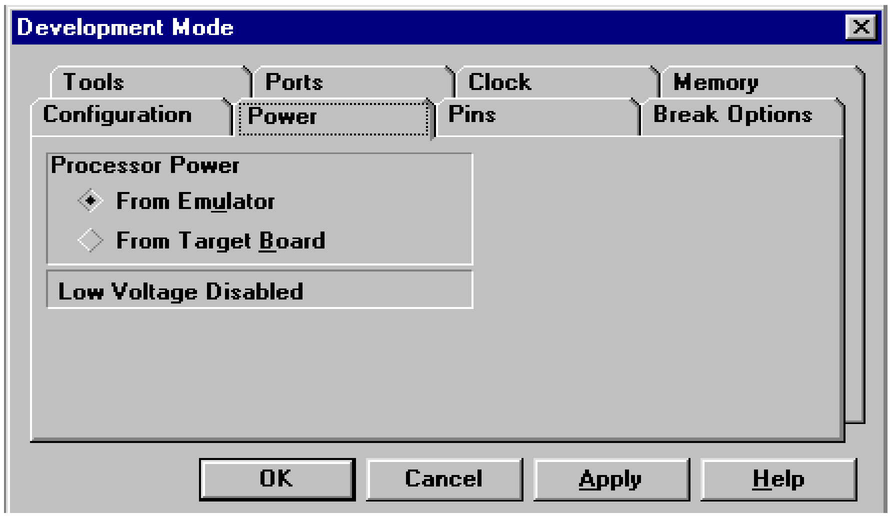
Watchdog Timer	Enable/disable the Watchdog Timer.
	None: Disables the Watchdog Timer.
	WDT Chip Reset Enable: Resets the processor when the Watchdog Timer times out.
	WDT Break Enable: Halts the processor when the Watchdog Timer times out.
	In addition to setting the Watchdog Timer, be sure to set the OPTIONS register according to the device’s data sheet.

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.

7.8.1.6 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**.



Processor Power
(●) From Emulator
( ) From Target Board

Low Voltage Disabled

Figure 7.44: 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.

7.8.1.7 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**.

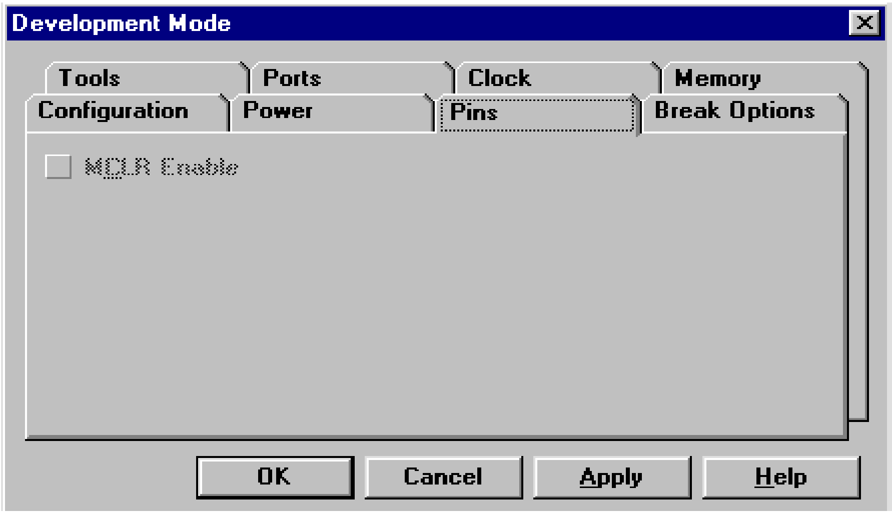


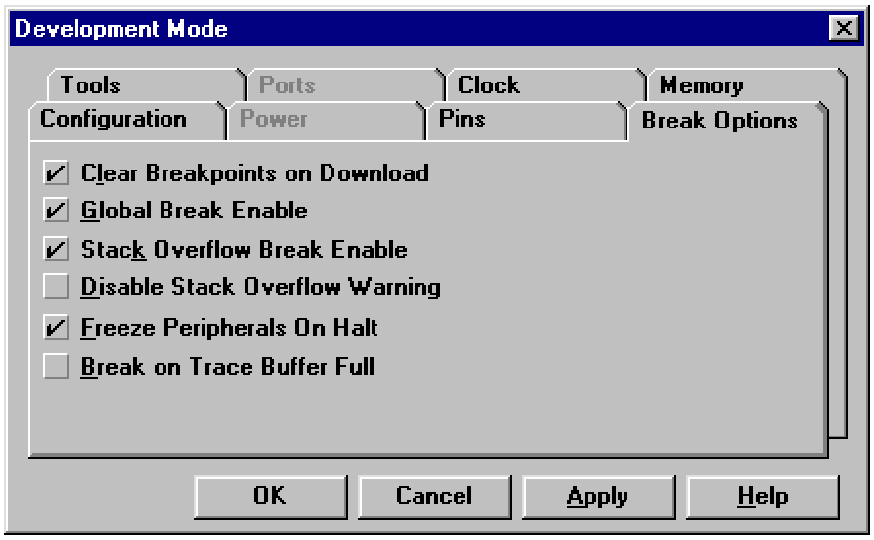
Figure 7.45: Pins Dialog

MCLR Enable	Enable or disable Master Clear.

7.8.1.8 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**.



- [✓] Clear Breakpoints on Download
- [✓] Global Break Enable
- [✓] Stack Overflow Break Enable
- [ ] Disable Stack Overflow Warning
- [✓] Freeze Peripherals On Halt
- [ ] Break on Trace Buffer Full

Figure 7.46: 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
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.
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.

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.

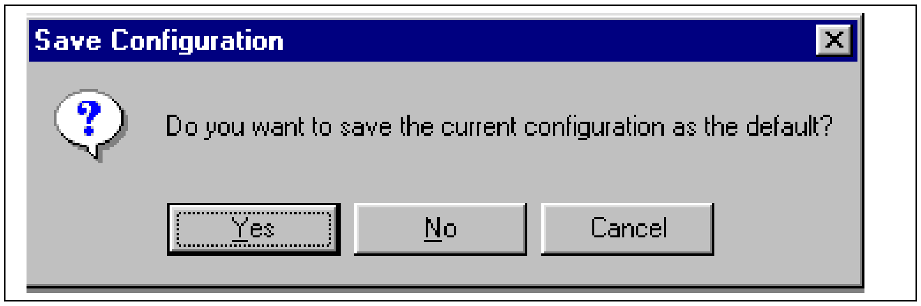
Note: If Freeze Peripherals On Halt is selected, the I/O port bits in the SFR or the watch windows will not update when single stepping.

Break on Trace Buffer Full	This option is not available for the MPLAB-ICE or the ICEPIC emulator. 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, this function is available through the Complex Trigger dialog.

7.8.2 Window Setup

7.8.2.1 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.



Do you want to save the current configuration as the default?

Figure 7.47: 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.

7.8.2.2 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.

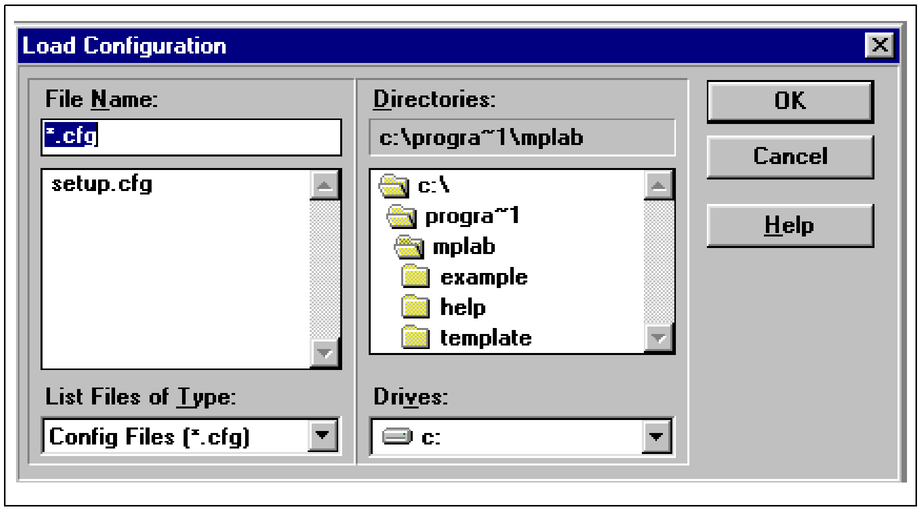


Figure 7.48: Load Setup Dialog

Note:	Setup.cfg is a configuration file used by PRO MATE II when programming Secure Data Product devices. Do not overwrite this file.

7.8.2.3 Default Configuration

Select _Options > Window Setup > Default Configuration_, to load the default user configuration the next time you start MPLAB IDE.

7.8.3 Current Editor Modes

Select *Options > Current Editor Modes* to configure the editor modes that apply to the current window.

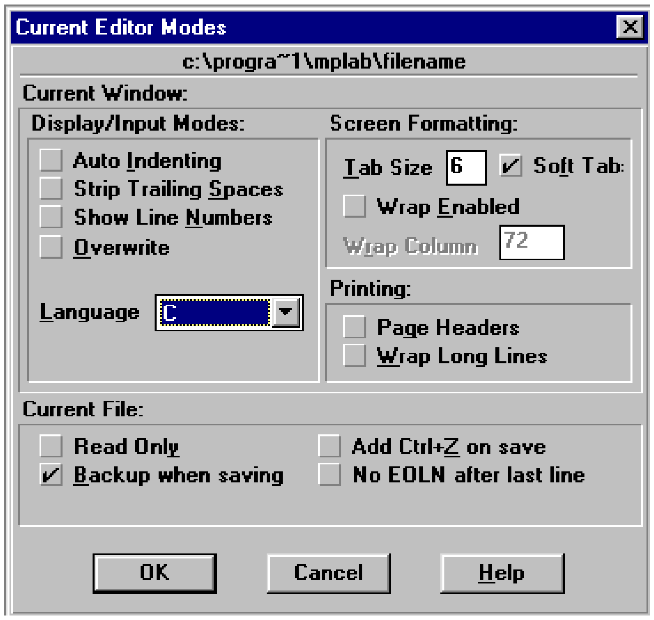


Figure 7.49: Current Editor Modes Dialog

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.

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 Enabled Wrap Column	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.

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.

Current File

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.
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.

7.8.4 Reset Editor Modes

Select *Options > Reset Editor Modes* to reset the modes that apply to the current window to the default values.

7.8.5 Environment Setup

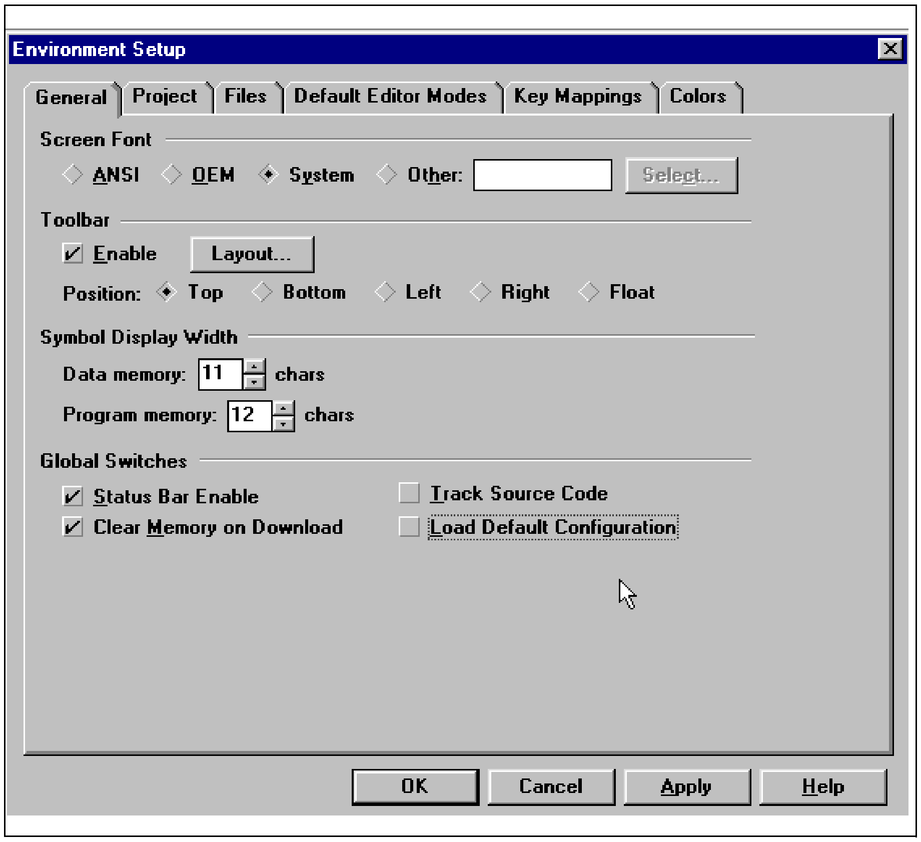
Select *Options > Environment Setup* to open a dialog box that allows you to change settings in the following areas:

- General Options
- Project Template Options
- Files
- Default Editor Modes
- Key Mappings
- Colors

7.8.5.1 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**.



Screen Font
ANSI OEM [✓ System] Other:

Toolbar
[✓ Enable]
Position: [✓ Top] Bottom Left Right Float

Symbol Display Width
Data memory: 11 chars
Program memory: 12 chars

Global Switches
[✓ Status Bar Enable]
[✓ Clear Memory on Download]
Track Source Code
Load Default Configuration

Figure 7.50: General Dialog

7.8.5.1.1. 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**.

7.8.5.1.2. 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:

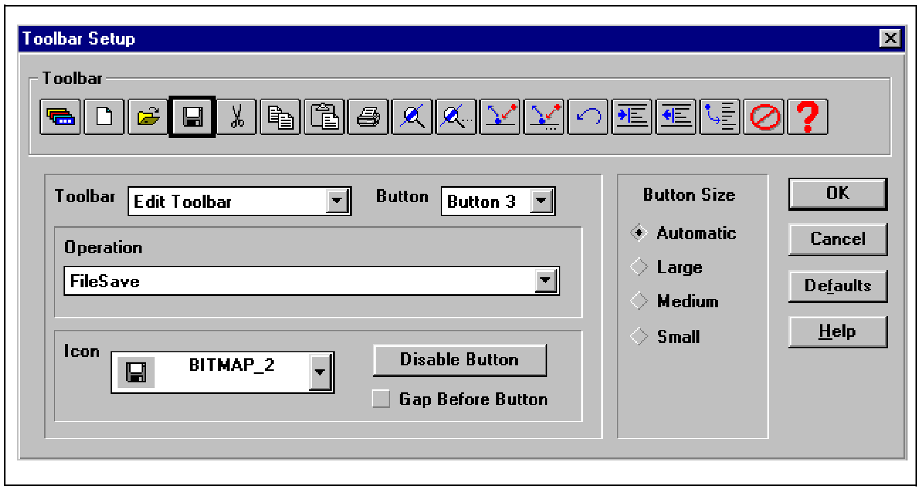


Figure 7.51: Toolbar Setup Dialog

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

Toolbar	Selects which toolbar to edit: Edit, Debug, Project, or User.
Button	Selects a toolbar button location. The toolbar has 16 available locations.
Operation	Selects the operation that MPLAB IDE will perform for the selected button location.
Icon	Selects a button to display in the selected button location.
Disable Button	Disables the toolbar button at the selected location on the toolbar.
Gap Before Button	Inserts a small gap before the toolbar button.
Button Size	Changes the toolbar button size. Button size options are: Automatic, Large, Medium, and Small.
Ok	Defines the toolbar as currently shown.
Cancel	Returns the toolbar to its previous state.
Defaults	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 and button, 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 dropdown 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.

4. From the Icon dropdown list, select the button you wish to display on the toolbar.
5. From the Operation dropdown 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 dropdown 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 dropdown list.
4. Click **Disable Button** to the right of the Icon dropdown 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 dropdown 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).

7.8.5.1.3. 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.

7.8.5.1.4. 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.

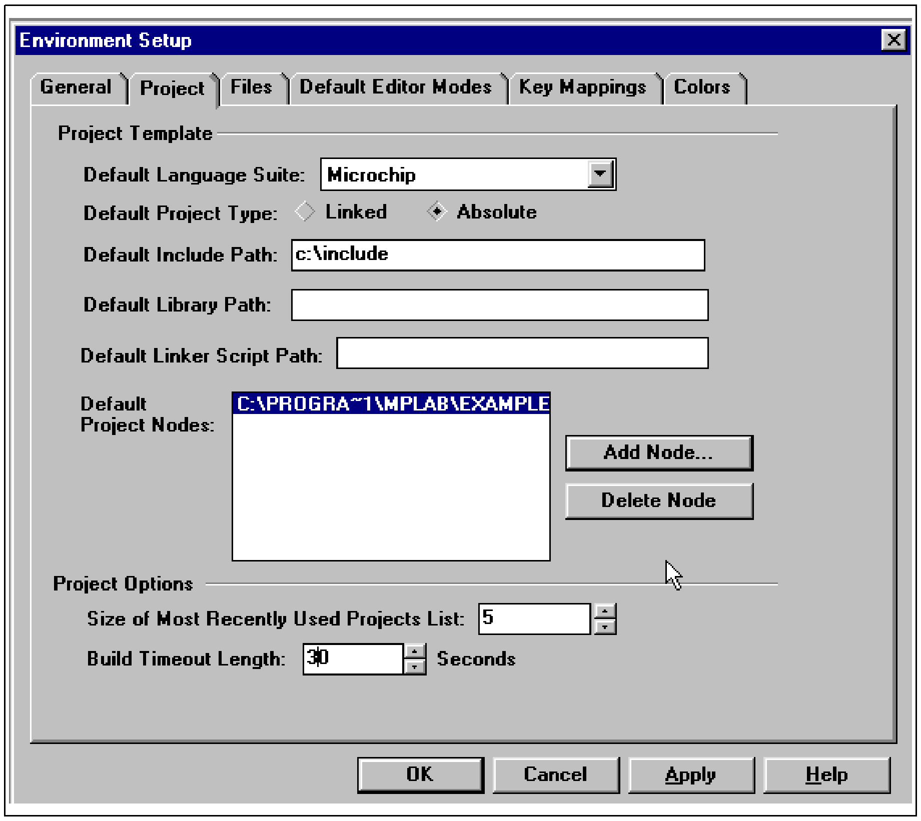
If you would like to save all the changes you have made in the Environment Setup dialog so far, click **Apply**.

7.8.5.2 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 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**.



Project Template

Default Language Suite: Microchip

Default Project Type: Linked [✓ Absolute]

Default Include Path: c:\include

Default Library Path:

Default Linker Script Path:

Default Project Nodes: C:\PROGRA~1\MPLAB\EXAMPLE

Project Options

Size of Most Recently Used Projects List: 5

Build Timeout Length: 30 Seconds

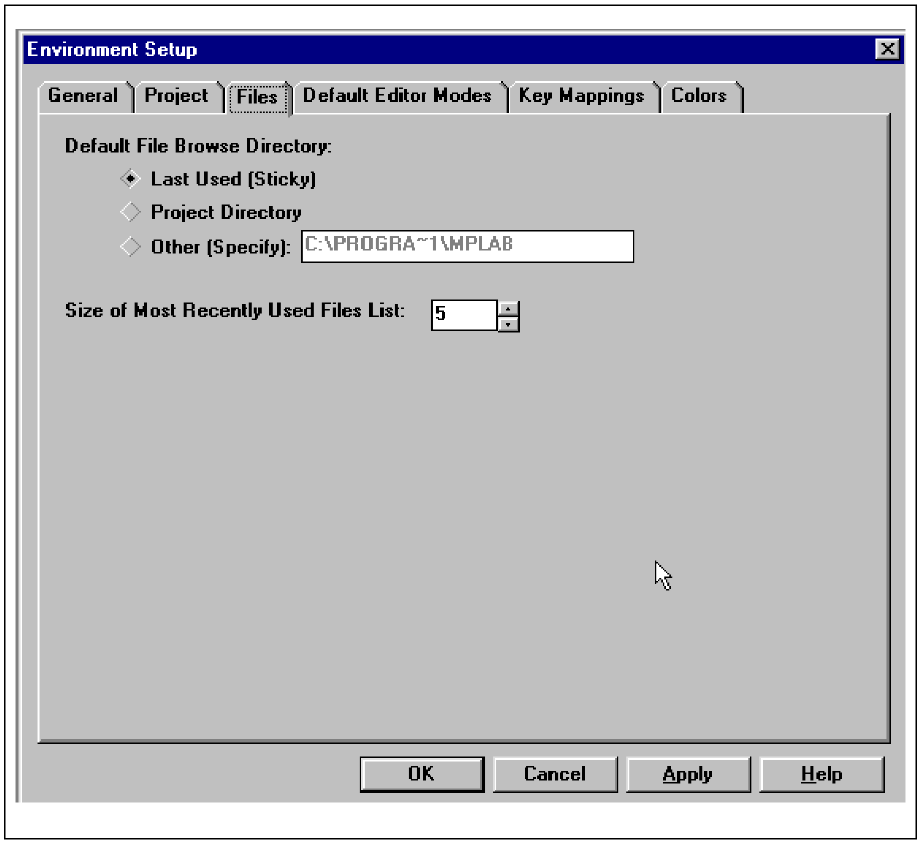
Figure 7.52: Project Template Dialog

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 directory 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 directory. 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 directory 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`

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, directory, 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 directory from which you choose your files defaults to the project template’s default browse directory. To select files from a different directory, select the Drive from the Drives dropdown 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 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 alotted time, a timeout will occur and no new hex file will result. If Build Timeout is set to “off,” no timeout will occur.

7.8.5.3 File Options

Select *Options>Environment Setup* and click the **Files** tab to set the default browse directory 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**.



Default File Browse Directory:

- [✓ Last Used (Sticky)]
- Project Directory
- Other (Specify): C:\PROGRA~1\MPLAB

Size of Most Recently Used Files List: 5

Figure 7.53: File Options Dialog

7.8.5.3.1. Default File Browse Directory

The Default File Browse Directory is the directory 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.

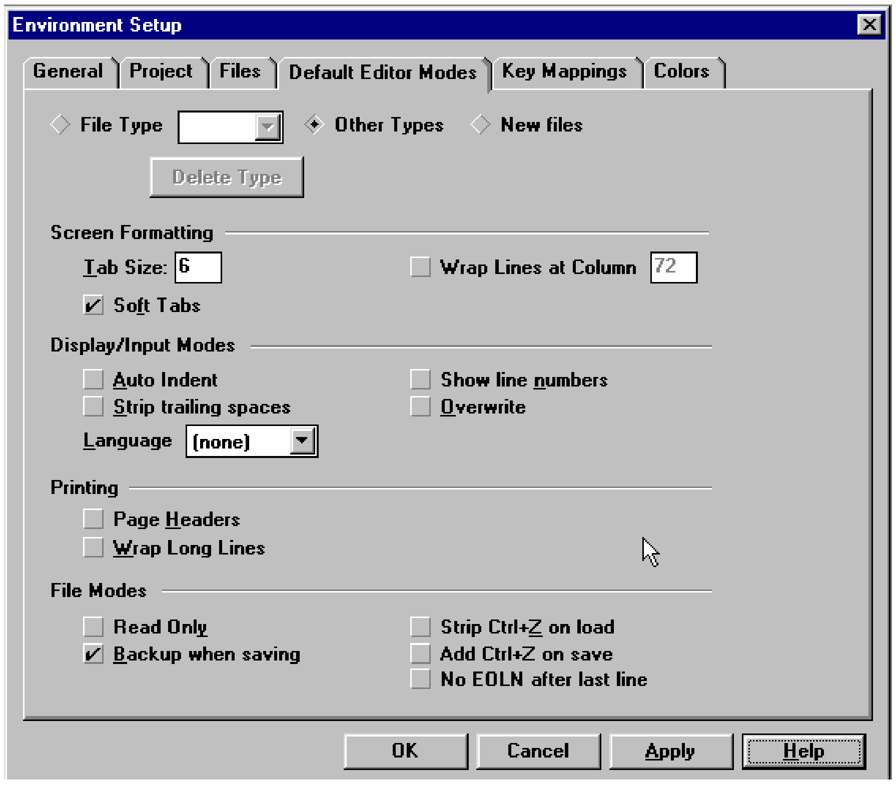
To specify a directory from which to start browsing, select Other, then enter the desired directory. This setting defaults to Project Directory.

7.8.5.3.2. 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.

7.8.5.4 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**.


Environment Setup

General | Project | Files | Default Editor Modes | Key Mappings | Colors

File Type [✓ Other Types] New files

Delete Type

Screen Formatting

Tab Size: 6

Wrap Lines at Column 72

[✓ Soft Tabs]

Display/Input Modes

Auto Indent

Show line numbers

Strip trailing spaces

Overwrite

Language (none)

Printing

Page Headers

Wrap Long Lines

File Modes

Read Only

Strip Ctrl+Z on load

[✓ Backup when saving]

Add Ctrl+Z on save

No EOLN after last line

OK | Cancel | Apply | Help

Figure 7.54: 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.

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 yo do not want lines to wrap.

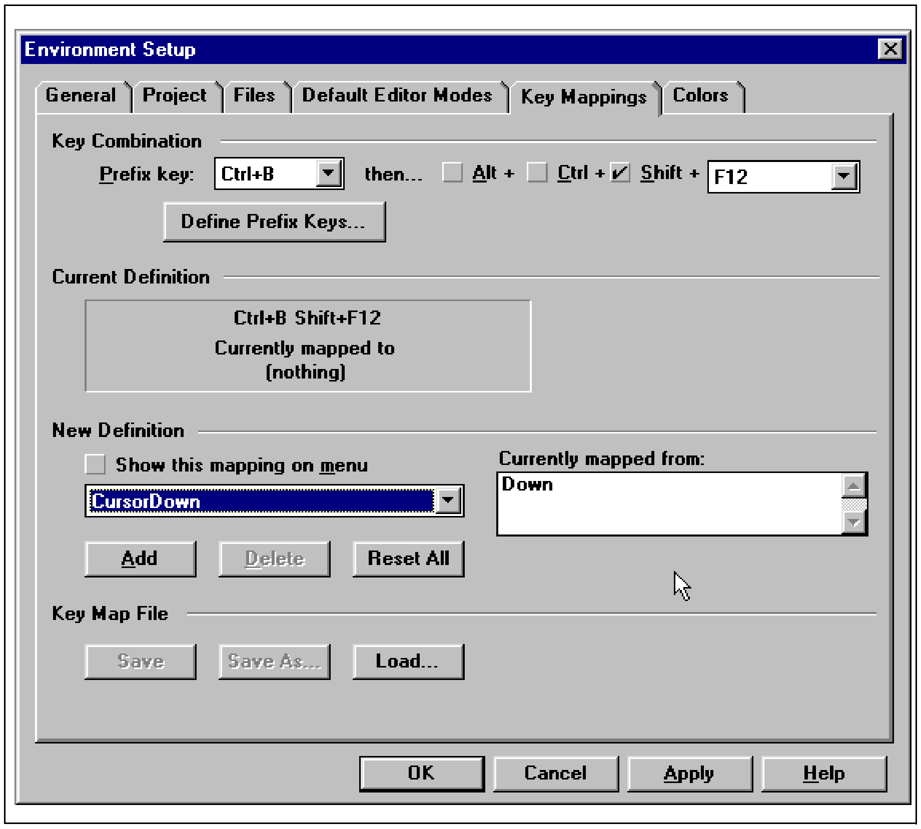
Display/Input Modes These settings control how text is displayed in the window and any special actions MPLAB Editor takes as you type.	
Auto Indent	Indents new lines to the same level as the preceding line.
Strip trailing spaces	Removes white space from the end of a line whenever you press **Enter**.
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.

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.

7.8.5.5 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 directory 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**.



Prefix key: Ctrl+B then... Alt + ☐ Ctrl + ☐ Shift + ✓ F12

Ctrl+B Shift+F12
Currently mapped to
(nothing)

☐ Show this mapping on menu
CursorDown

Currently mapped from:
Down

Figure 7.55: 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 directory to record the key mapping values that will be carried from one session to another.

To create a key mapping:

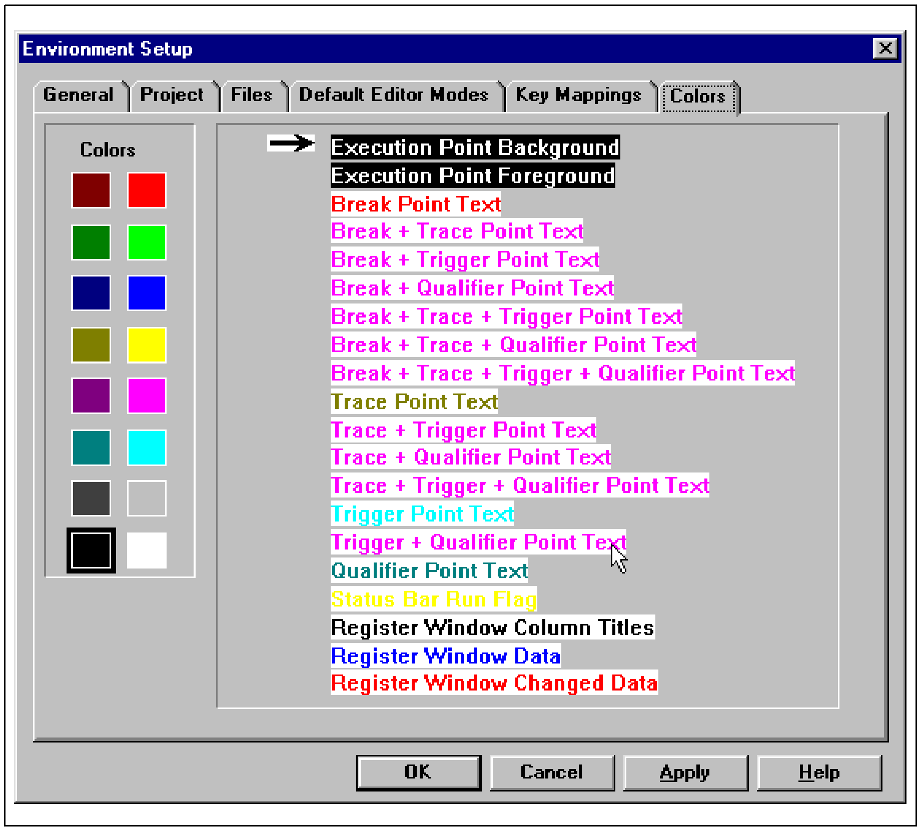
1. To modify an existing key mapping, click **Load** to select the existing key mapping file.
2. Define a prefix key (optional).
3. Click the checkboxes below the prefix key to assign the key or key sequence (e.g., Shift + F2). The Current Definition box displays the current function that is mapped to 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 dropdown 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.

7.8.5.6 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**.



Execution Point Background
Execution Point Foreground
Break Point Text
Break + Trace Point Text
Break + Trigger Point Text
Break + Qualifier Point Text
Break + Trace + Trigger Point Text
Break + Trace + Qualifier Point Text
Break + Trace + Trigger + Qualifier Point Text
Trace Point Text
Trace + Trigger Point Text
Trace + Qualifier Point Text
Trace + Trigger + Qualifier Point Text
Trigger Point Text
Trigger + Qualifier Point Text
Qualifier Point Text
Status Bar Run Flag
Register Window Column Titles
Register Window Data
Register Window Changed Data

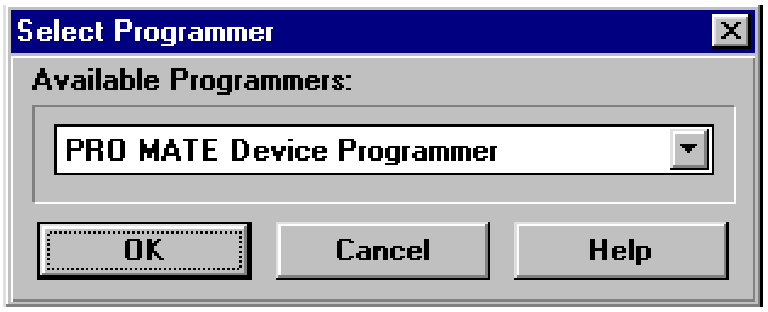
Figure 7.56: Colors Dialog

To change the color, select the text that you want to alter by clicking on it. Next, select the new color.

7.8.6 Programmer Options

7.8.6.1 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 dropdown list in the Select Programmer dialog.
3. Click **OK**.



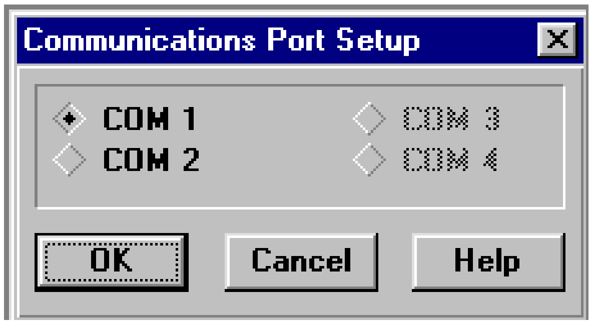
PRO MATE Device Programmer

Figure 7.57: Select Programmer Dialog

7.8.6.2 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**.



[✓ COM 1] COM 2 COM 3 COM 4

Figure 7.58: Communications Port Setup Dialog

7.9 Tools Menu

7.9.1 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.

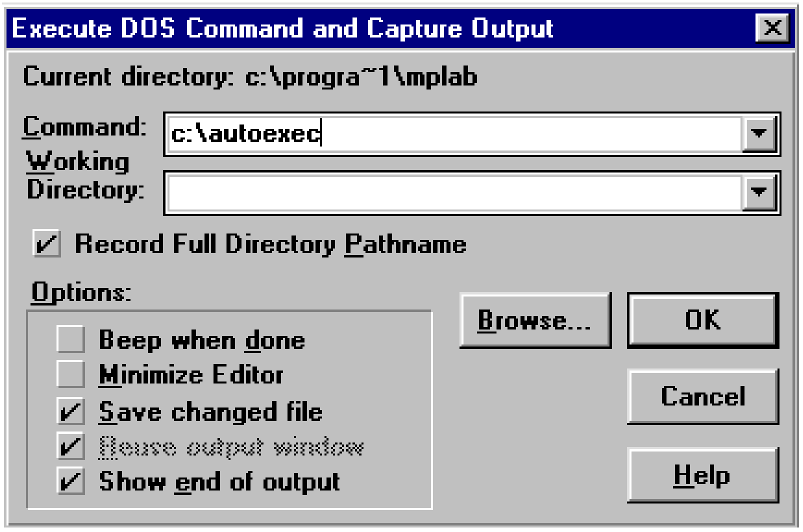
The command starts a dialog that prompts you for the command line to be executed. This can be any DOS command, including built-in ones like DIR.

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.



Current directory: c:\progra~1\mplab

Command: c:\autoexec

Working Directory:

[✓] Record Full Directory Pathname

Options:

[ ] Beep when done
[ ] Minimize Editor
[✓] Save changed file
[✓] Reuse output window
[✓] Show end of output

Figure 7.59: 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.

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.

7.9.2 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.

7.9.3 Verify PICMASTER

If you are using a PICMASTER emulator with MPLAB IDE, select *Tools > Verify PICMASTER* to verify that the PICMASTER is operating properly. Refer to the *PICMASTER User's Guide* for detailed information on verifying the PICMASTER emulator.

7.9.4 Verify MPLAB-ICE

If you are using an MPLAB-ICE with MPLAB IDE, select *Tools > Verify MPLAB-ICE* to verify that the MPLAB-ICE is operating properly. Refer to the *MPLAB-ICE User's Guide* for detailed information on verifying the MPLAB-ICE.

7.10 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

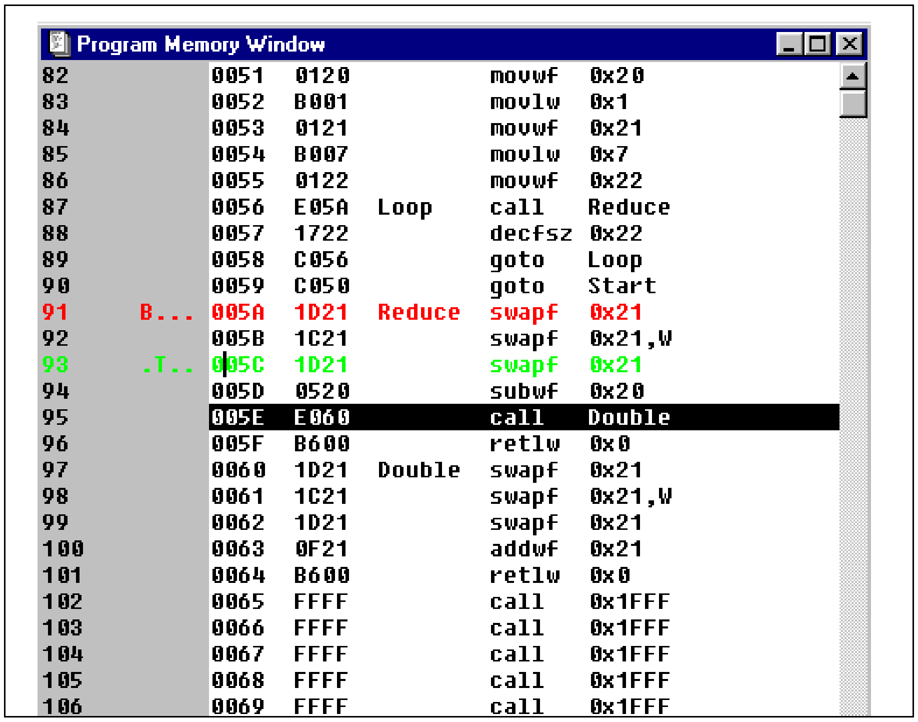
The following commands affect the arrangement and appearance of windows in MPLAB IDE.

- Tile Horizontal
- Tile Vertical
- Cascade
- Iconize All
- Arrange Icons
- (*Open Windows*)

7.10.1 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.



```
82          0051  0120          movwf   0x20
83          0052  B001          movlw   0x1
84          0053  0121          movwf   0x21
85          0054  B007          movlw   0x7
86          0055  0122          movwf   0x22
87          0056  E05A  Loop    call    Reduce
88          0057  1722          decfsz  0x22
89          0058  C056          goto    Loop
90          0059  C050          goto    Start
91    B...  005A  1D21  Reduce  swapf   0x21
92          005B  1C21          swapf   0x21,W
93    .T..  005C  1D21          swapf   0x21
94          005D  0520          subwf   0x20
95          005E  E060          call    Double
96          005F  B600          retlw   0x0
97          0060  1D21  Double  swapf   0x21
98          0061  1C21          swapf   0x21,W
99          0062  1D21          swapf   0x21
100         0063  0F21          addwf   0x21
101         0064  B600          retlw   0x0
102         0065  FFFF          call    0x1FFF
103         0066  FFFF          call    0x1FFF
104         0067  FFFF          call    0x1FFF
105         0068  FFFF          call    0x1FFF
106         0069  FFFF          call    0x1FFF
```

Figure 7.60: Program Memory Window – Machine Code Display

7.10.1.1 System Button Options

Click the system button in the upper left corner of the Program Memory screen to display the following options:

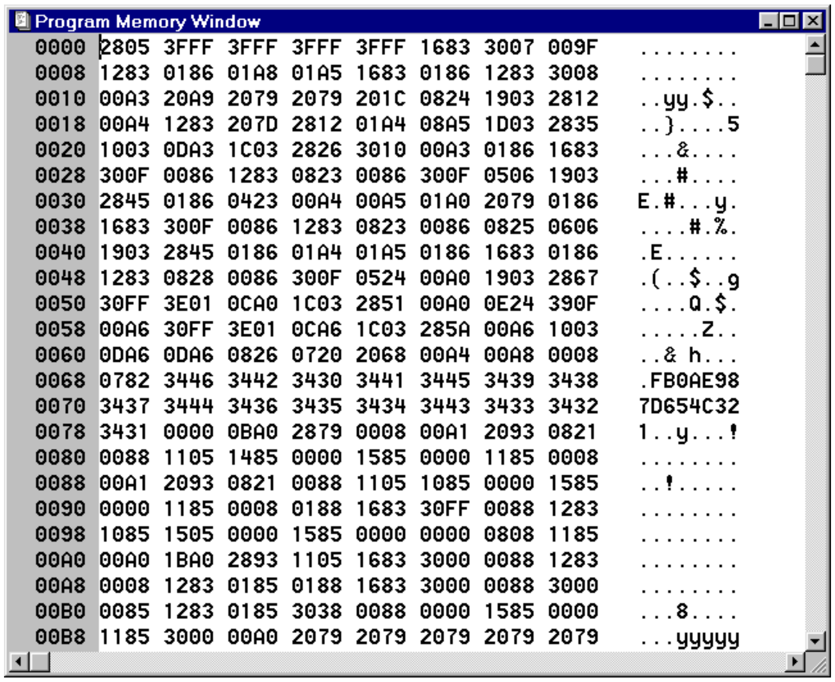
Toggle Line Numbers	Toggles field for displaying line numbers and qualifier points.
Hex Code Display	Displays program memory information as hex data.
Machine Code Display	Displays disassembled hex code with no symbolic information.
Disassembly Display	Displays disassembled hex code with symbols.

7.10.1.2 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 7.61).
- **Machine Code Display** – This displays the disassembled hex code with no symbolic information (Figure 7.60).
- **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.



```
0000 2805 3FFF 3FFF 3FFF 3FFF 1683 3007 009F   ........
0008 1283 0186 01A8 01A5 1683 0186 1283 3008   ........
0010 00A3 20A9 2079 2079 201C 0824 1903 2812   ..yy.$..
0018 00A4 1283 207D 2812 01A4 08A5 1D03 2835   ..}....5
0020 1003 0DA3 1C03 2826 3010 00A3 0186 1683   ...&....
0028 300F 0086 1283 0823 0086 300F 0506 1903   ...#....
0030 2845 0186 0423 00A4 00A5 01A0 2079 0186   E.#...y.
0038 1683 300F 0086 1283 0823 0086 0825 0606   ....#.%.
0040 1903 2845 0186 01A4 01A5 0186 1683 0186   .E......
0048 1283 0828 0086 300F 0524 00A0 1903 2867   .(..$..g
0050 30FF 3E01 0CA0 1C03 2851 00A0 0E24 390F   ....Q.$.
0058 00A6 30FF 3E01 0CA6 1C03 285A 00A6 1003   .....Z..
0060 0DA6 0DA6 0826 0720 2068 00A4 00A8 0008   ..& h...
0068 0782 3446 3442 3430 3441 3445 3439 3438   .FB0AE98
0070 3437 3444 3436 3435 3434 3443 3433 3432   7D654C32
0078 3431 0000 0BA0 2879 0008 00A1 2093 0821   1..y...!
0080 0088 1105 1485 0000 1585 0000 1185 0008   ........
0088 00A1 2093 0821 0088 1105 1085 0000 1585   ..!.....
0090 0000 1185 0008 0188 1683 30FF 0088 1283   ........
0098 1085 1505 0000 1585 0000 0000 0808 1185   ........
00A0 00A0 1BA0 2893 1105 1683 3000 0088 1283   ........
00A8 0008 1283 0185 0188 1683 3000 0088 3000   ........
00B0 0085 1283 0185 3038 0088 0000 1585 0000   ...8....
00B8 1185 3000 00A0 2079 2079 2079 2079 2079   ...yyyyy
```

Figure 7.61: Program Memory Window – Hex Code Display

7.10.1.3 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:

7.10.1.3.1. 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.

7.10.1.4 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**

7.10.2 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.

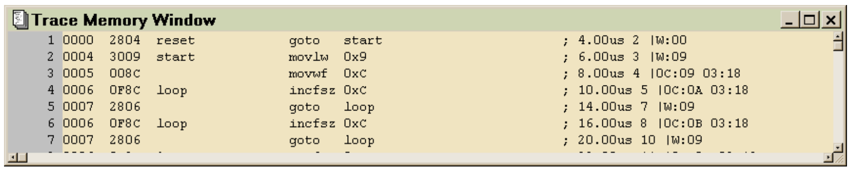
The trace memory window takes a “snapshot” of your program’s execution. For emulators that have a trace buffer, this shows how your program runs at full speed.

Some applications, such as motor control systems, can not 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:



```
 1 0000  2804  reset          goto   start              ; 4.00us 2 |W:00
 2 0004  3009  start          movlw  0x9                ; 6.00us 3 |W:09
 3 0005  008C                 movwf  0xC                ; 8.00us 4 |0C:09 03:18
 4 0006  0F8C  loop           incfsz 0xC                ; 10.00us 5 |0C:0A 03:18
 5 0007  2806                 goto   loop               ; 14.00us 7 |W:09
 6 0006  0F8C  loop           incfsz 0xC                ; 16.00us 8 |0C:0B 03:18
 7 0007  2806                 goto   loop               ; 20.00us 10 |W:09
```

Figure 7.62: 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.

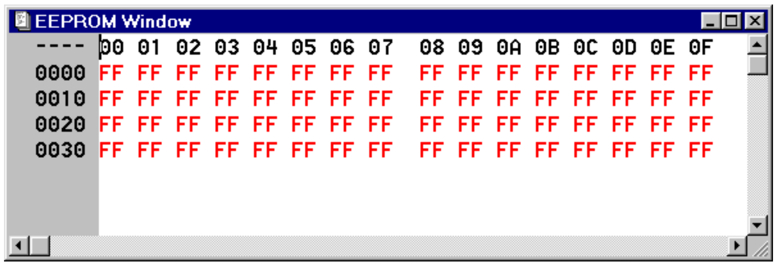
Field Three	Program Label in symbolic format. You can increase the display width of labels by selecting _Options > Environment Setup_ and clicking the **General** tab.
Field Four	Machine code, disassembled code, or source code.
Field Five	Emulator: Status on External Logic Probe Lines. The status fields are displayed 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_.

7.10.3 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.



```
---- 00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F
0000 FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0010 FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0020 FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0030 FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
```

Figure 7.63: 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.

7.10.3.0.1. System Button Options

Click the system button in the upper left corner of the EEPROM Memory screen to display the following options:

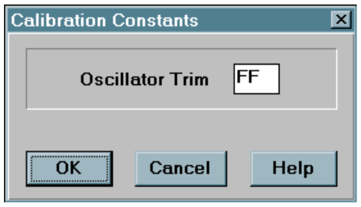
Toggle Line Numbers	Toggles field for displaying line numbers.
Hex Display	Displays program memory information as Hex data.
ASCII Display	Displays the ASCII character at each memory location.

7.10.4 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.

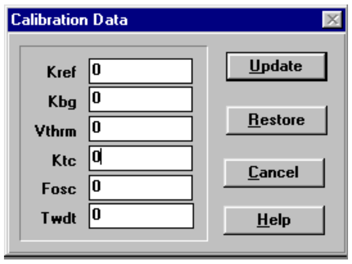
The Calibration Data dialog is for use with the PIC12CXXX or PIC14000 device families. It displays the floating point data in the Emulator Probe for alteration by the user.

Saved calibration data may be loaded into MPLAB IDE using *File>Import*.



Oscillator Trim: FF

Figure 7.64: Calibration Constants Dialog - PIC12CXXX



Kref: 0
Kbg: 0
Vthrm: 0
Ktc: 0
Fosc: 0
Twdt: 0

Figure 7.65: Calibration Data Dialog - PIC14000

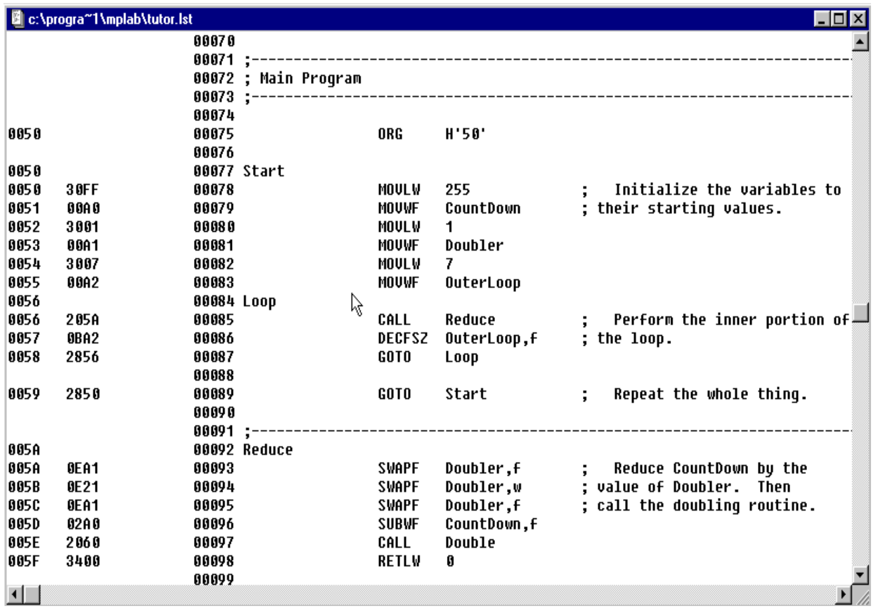
Update	Takes the information from calibration data dialog, converts it from the IEEE 754 format to the Microchip version of IEEE 754, and stores it in the memory area for use in the embedded code. Only the first four numbers KREF, KBG, VTHRM, and KTC are in floating point format. The last two, Fosc and 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.

7.10.5 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.

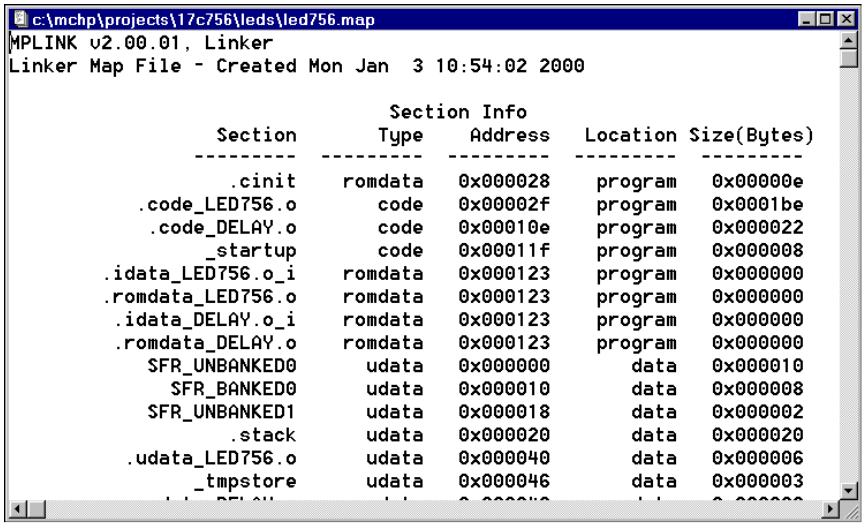


```
                        00070
                        00071 ;------------------------------------------------------------------------
                        00072 ; Main Program
                        00073 ;------------------------------------------------------------------------
                        00074
0050                    00075                 ORG     H'50'
                        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   0BA2             00086                 DECFSZ  OuterLoop,f     ; the loop.
0058   2856             00087                 GOTO    Loop
                        00088
0059   2850             00089                 GOTO    Start           ;   Repeat the whole thing.
                        00090
                        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 7.66: Absolute Listing Window

7.10.6 Map File

A map file shows the used and unused memory regions after linking. To view the map file, select *Window > Map File*.



```
MPLINK v2.00.01, Linker
Linker Map File - Created Mon Jan  3 10:54:02 2000

                                 Section Info
                  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
```

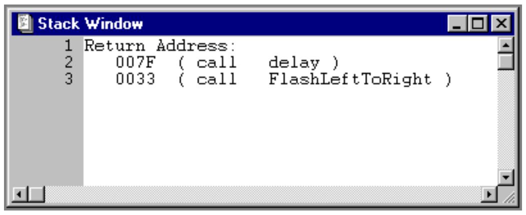
Figure 7.67: Map File Window

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*.

7.10.7 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.



```
1 Return Address:
2    007F  ( call   delay )
3    0033  ( call   FlashLeftToRight )
```

Figure 7.68: .tack 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.

7.10.7.1 Hardware Stack Levels

7.10.7.1.1. 12-Bit Core Hardware Stack – 2-Levels Deep

Devices with a 12-bit core, such as PIC12CXXX and PIC16C5X, have a 2-level deep hardware stack.

7.10.7.1.2. 14-Bit Core Hardware Stack – 8-Levels Deep

Devices with a 14-bit core, such as PIC14000 and PIC16CXX, 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.

7.10.7.1.3. 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.

7.10.7.1.4. 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.

7.10.7.2 Simulator Stack—12-Bit Core Devices

The MPLAB-SIM simulator presents an accurate simulation of the hardware stack on the PIC12CXX 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.

7.10.7.3 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.

7.10.7.4 Simulator Stack – 16-Bit Core Devices

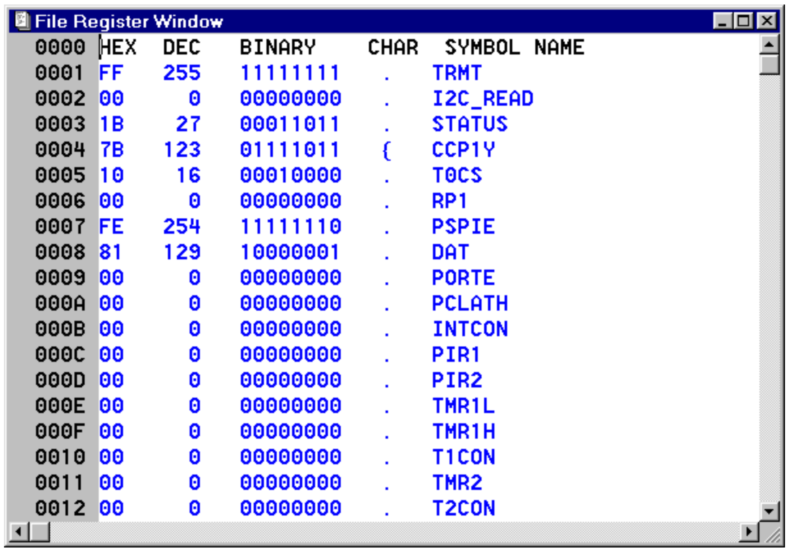
The MPLAB-SIM simulator presents an accurate simulation of the hardware stack on the PIC17CXXX, and additionally provides warning messages if an underflow or overflow condition occurs. When a CALL or LCALL instruction is encountered or when an interrupt has occurred, the value of the PC + 1 is pushed 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.

7.10.7.5 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.

7.10.8 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.



```
0000 HEX  DEC   BINARY    CHAR  SYMBOL NAME
0001 FF   255   11111111   .   TRMT
0002 00     0   00000000   .   I2C_READ
0003 1B    27   00011011   .   STATUS
0004 7B   123   01111011   {   CCP1Y
0005 10    16   00010000   .   T0CS
0006 00     0   00000000   .   RP1
0007 FE   254   11111110   .   PSPIE
0008 81   129   10000001   .   DAT
0009 00     0   00000000   .   PORTE
000A 00     0   00000000   .   PCLATH
000B 00     0   00000000   .   INTCON
000C 00     0   00000000   .   PIR1
000D 00     0   00000000   .   PIR2
000E 00     0   00000000   .   TMR1L
000F 00     0   00000000   .   TMR1H
0010 00     0   00000000   .   T1CON
0011 00     0   00000000   .   TMR2
0012 00     0   00000000   .   T2CON
```

Figure 7.69: File Registers Window – Symbolic Display

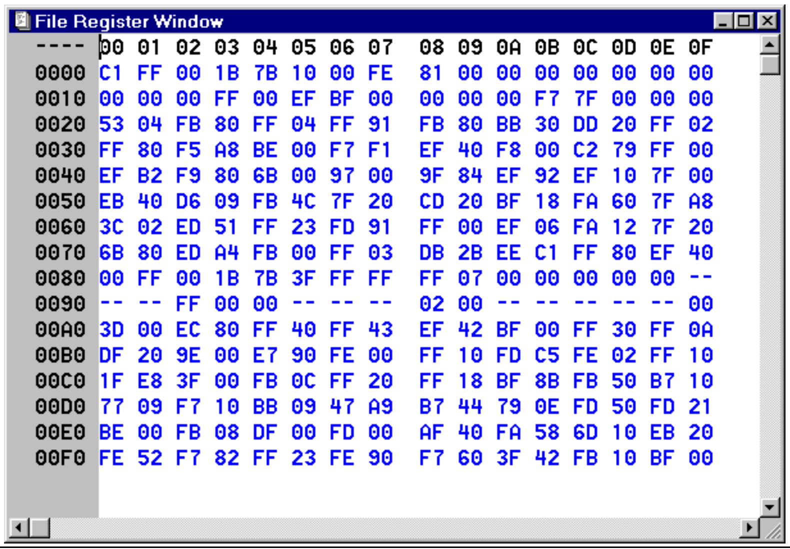
File register contents can also be modified through this window. To change a range of registers (e.g., to fill them with a constant value), click the left mouse button and drag over the values you want to change. To change one register, simply place the cursor over the register you want to change. Then click the right mouse button to display the Fill Register pop-up menu. Select **Fill Register** to display the Modify dialog (Figure 7.79) with the address range already entered.

7.10.8.1 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 7.70).
- **Symbolic Display** – This displays each file register symbolically with corresponding data in hex, decimal, binary and character formats (Figure 7.69).
- **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 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.



```
----  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F
0000  C1 FF 00 1B 7B 10 00 FE  81 00 00 00 00 00 00 00
0010  00 00 00 FF 00 EF BF 00  00 00 00 F7 7F 00 00 00
0020  53 04 FB 80 FF 04 FF 91  FB 80 BB 30 DD 20 FF 02
0030  FF 80 F5 A8 BE 00 F7 F1  EF 40 F8 00 C2 79 FF 00
0040  EF B2 F9 80 6B 00 97 00  9F 84 EF 92 EF 10 7F 00
0050  EB 40 D6 09 FB 4C 7F 20  CD 20 BF 18 FA 60 7F A8
0060  3C 02 ED 51 FF 23 FD 91  FF 00 EF 06 FA 12 7F 20
0070  6B 80 ED A4 FB 00 FF 03  DB 2B EE C1 FF 80 EF 40
0080  00 FF 00 1B 7B 3F FF FF  FF 07 00 00 00 00 00 --
0090  -- -- FF 00 00 -- -- --  02 00 -- -- -- -- -- 00
00A0  3D 00 EC 80 FF 40 FF 43  EF 42 BF 00 FF 30 FF 0A
00B0  DF 20 9E 00 E7 90 FE 00  FF 10 FD C5 FE 02 FF 10
00C0  1F E8 3F 00 FB 0C FF 20  FF 18 BF 8B FB 50 B7 10
00D0  77 09 F7 10 BB 09 47 A9  B7 44 79 0E FD 50 FD 21
00E0  BE 00 FB 08 DF 00 FD 00  AF 40 FA 58 6D 10 EB 20
00F0  FE 52 F7 82 FF 23 FE 90  F7 60 3F 42 FB 10 BF 00
```

Figure 7.70: File Register Window – Hex Display

7.10.8.2 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.

7.10.9 Special Function Registers

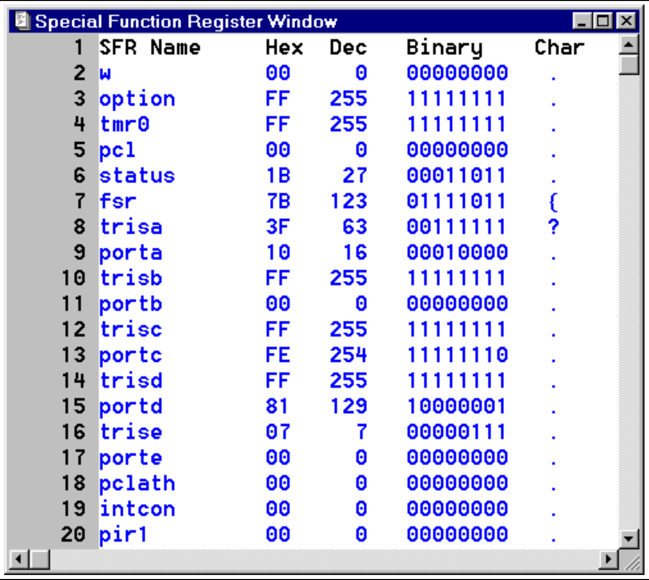
Select *Window > Special Function Registers* to display the contents of the Special Function Registers (SFR) for the processor being emulated. The format provided by this window is more useful for viewing the SFRs than the normal file register window, since each SFR name is included and several number 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.



```
 1 SFR Name     Hex  Dec  Binary    Char
 2 w            00     0  00000000   .
 3 option       FF   255  11111111   .
 4 tmr0         FF   255  11111111   .
 5 pcl          00     0  00000000   .
 6 status       1B    27  00011011   .
 7 fsr          7B   123  01111011   {
 8 trisa        3F    63  00111111   ?
 9 porta        10    16  00010000   .
10 trisb        FF   255  11111111   .
11 portb        00     0  00000000   .
12 trisc        FF   255  11111111   .
13 portc        FE   254  11111110   .
14 trisd        FF   255  11111111   .
15 portd        81   129  10000001   .
16 trise        07     7  00000111   .
17 porte        00     0  00000000   .
18 pclath       00     0  00000000   .
19 intcon       00     0  00000000   .
20 pir1         00     0  00000000   .
```

Figure 7.71: Special Function Registers

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:

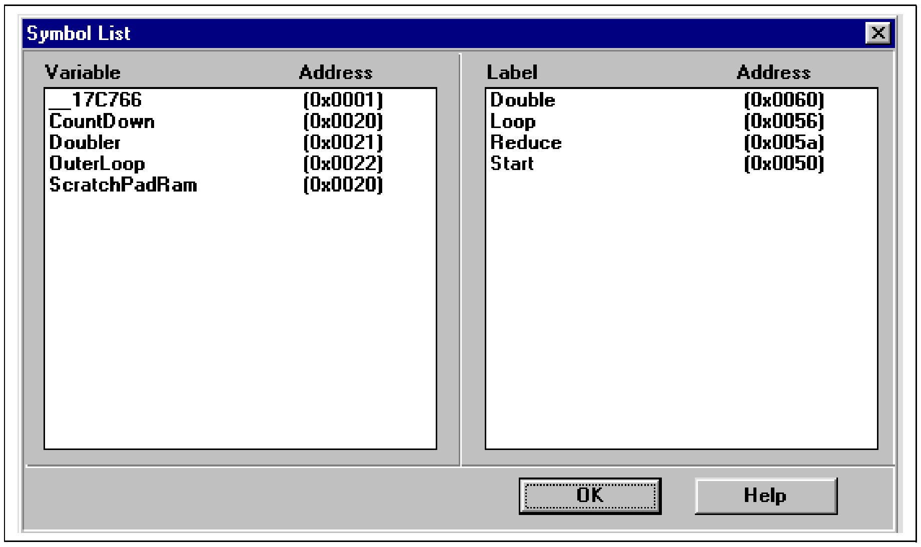
1. Double click on a register in this window to invoke the Modify dialog box with the symbol/address and data fields already filled in, or
2. Use the execute *Window > Modify* menu.

Note: The SFR names and addresses are different for every device.

7.10.10 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.



Variable	Address
__17C766	(0x0001)
CountDown	(0x0020)
Doubler	(0x0021)
OuterLoop	(0x0022)
ScratchPadRam	(0x0020)

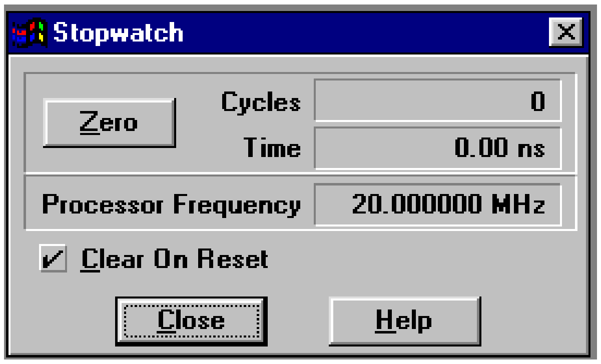
Label	Address
Double	(0x0060)
Loop	(0x0056)
Reduce	(0x005a)
Start	(0x0050)

Figure 7.72: 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.

7.10.11 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.



Cycles 0
Time 0.00 ns
Processor Frequency 20.000000 MHz
✓ Clear On Reset

Figure 7.73: Stopwatch Dialog Box

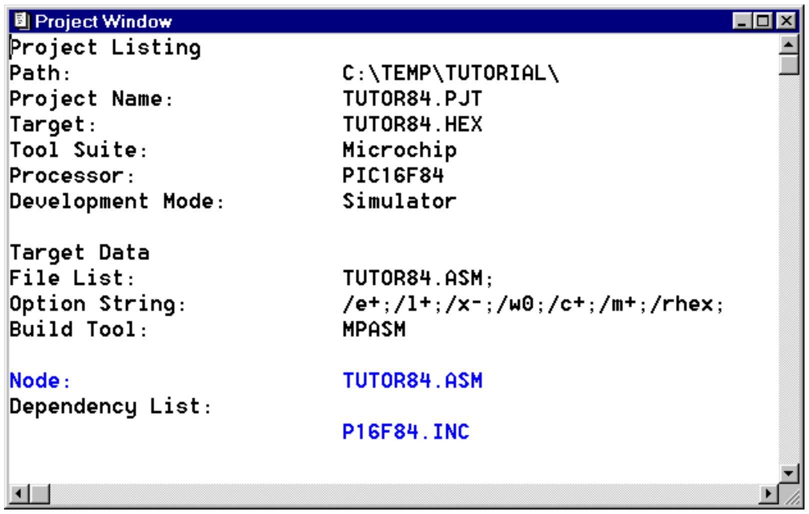
Cycles	Displays the number of cycles that the processor executes.
Time	Displays the stopwatch time in seconds. Computed from the number of cycles executed and from the processor frequency.
Zero	Click **Zero** to reset the cycle counter to zero. You can reset the timer at any time when the processor is halted.

Example 7.1: 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.

7.10.12 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.



```
Project Listing
Path:                     C:\TEMP\TUTORIAL\
Project Name:             TUTOR84.PJT
Target:                   TUTOR84.HEX
Tool Suite:               Microchip
Processor:                PIC16F84
Development Mode:         Simulator

Target Data
File List:                TUTOR84.ASM;
Option String:            /e+;/l+;/x-;/w0;/c+;/m+;/rhex;
Build Tool:               MPASM

Node:                     TUTOR84.ASM
Dependency List:
                          P16F84.INC
```

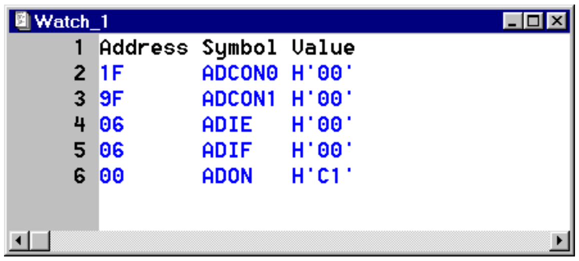
Figure 7.74: Project Window

7.10.13 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.

7.10.13.1 New Watch Window

To create a Watch window, select *Window > Watch Window > New Watch Window*. Both the Add Watch Symbol dialog (Figure 7.76) and the Watch_1 window (Figure 7.75) will open. Refer to Section 7.10.13.4 for information on editing Watch windows.



```
1 Address Symbol Value
2 1F      ADCON0 H'00'
3 9F      ADCON1 H'00'
4 06      ADIE   H'00'
5 06      ADIF   H'00'
6 00      ADON   H'C1'
```

Figure 7.75: Watch Window

7.10.13.2 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 7.10.13.4 for information on editing Watch windows.

7.10.13.3 Add Watch Window Symbols

You can add symbols to the Watch window you are active in.

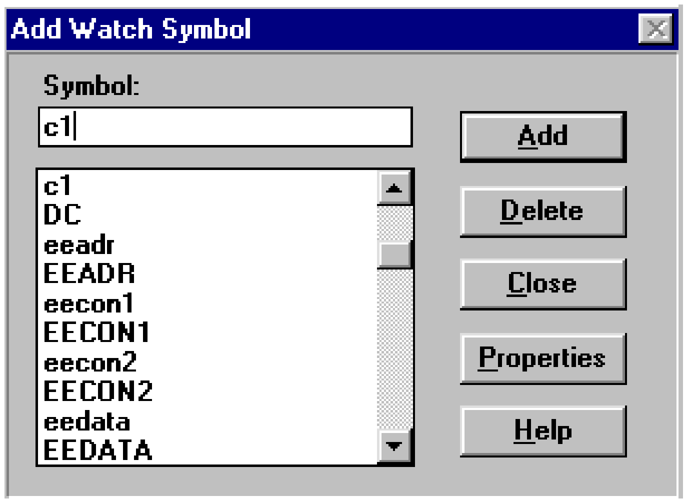


Figure 7.76: Add Watch Symbol Dialog

Select *Window > Watch Window > Add to Active Watch*. Select symbols 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.

Note:	The variable `W` represents an address of '0', where as the variable `w` represents the `W` register value.

To change how a symbol is displayed in the Watch window, select that symbol and click **Properties** to open the Properties dialog (see Section 7.10.13.5).

Click **Close** when you have finished adding symbols. The Watch window will display the current value of the variables you selected.

7.10.13.4 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.

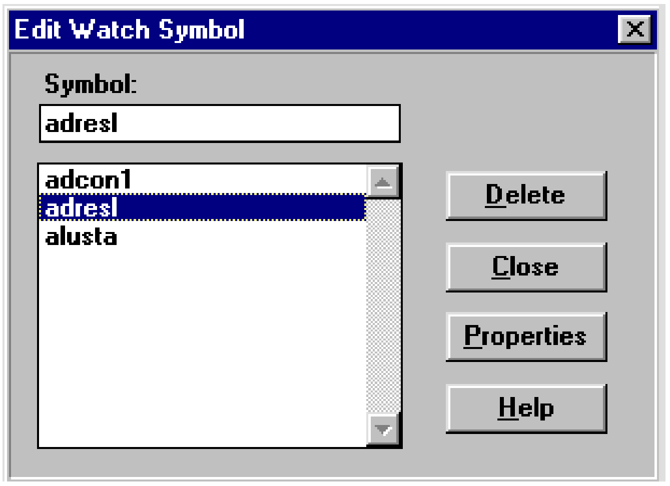


Figure 7.77: Edit Watch Symbol Dialog

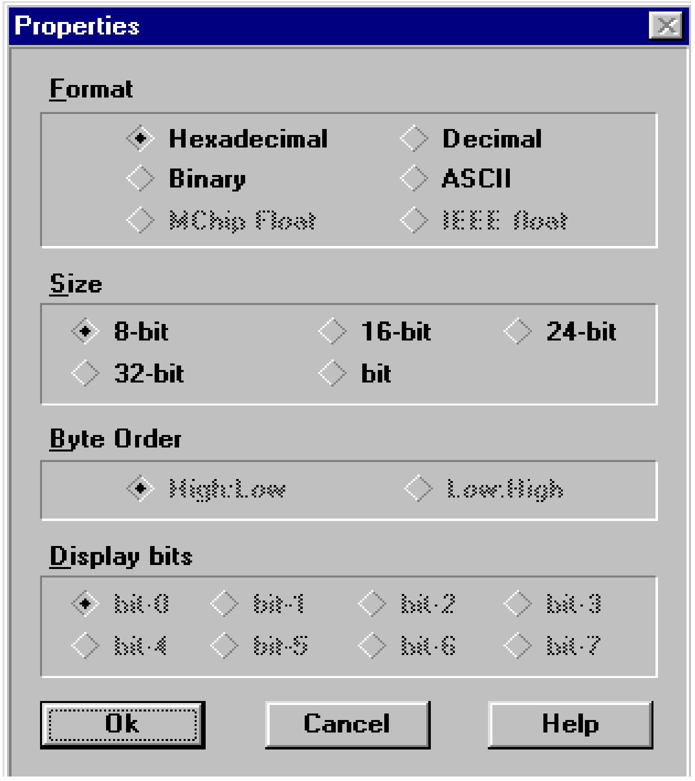
In the Symbol list of the Edit Watch Symbol dialog, scroll to the symbol of the variable you wish to change and highlight it. Click **Properties**.

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.

7.10.13.5 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.



Properties

Format
[✓ Hexadecimal] Decimal
Binary ASCII
MChip Float IEEE float

Size
[✓ 8-bit] 16-bit 24-bit
32-bit bit

Byte Order
[✓ High:Low] Low:High

Display bits
[✓ bit-0] bit-1 bit-2 bit-3
bit-4 bit-5 bit-6 bit-7

Figure 7.78: 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.

7.10.13.6 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.

7.10.14 Modify

Select *Window > Modify* to display and/or modify the contents of Data Memory, Program Memory, the Stack, or EEPROM memory.

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.

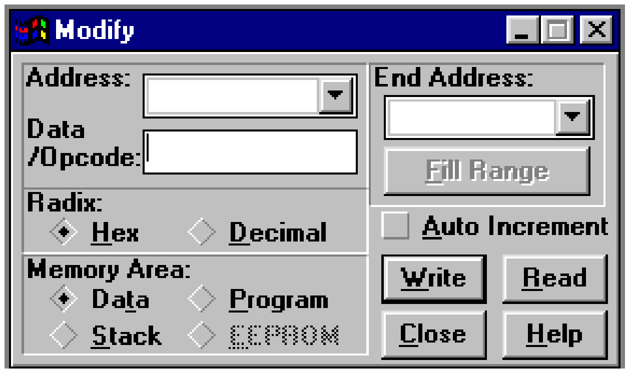


Figure 7.79: Modify Dialog Box

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	Enter the Address at which data is to be read or modified. You can enter a numerical address or a symbol. (Label)
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
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.
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: Use care when modifying the stack or the program counter register. The effect of modifying these registers is not seen until the processor is taken out of halt.

7.10.15 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.

7.10.16 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.

7.10.17 Cascade

The *Window > Cascade* command arranges all open windows in a cascade pattern.

7.10.18 Iconize All

Window > Iconize All makes all windows into icons.

7.10.19 Arrange Icons

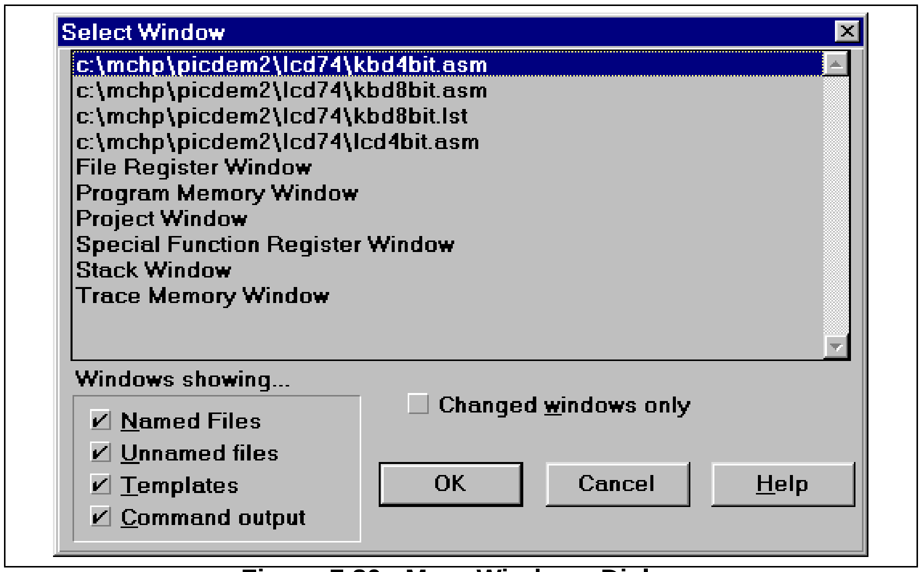
Window > Arrange Icons arranges all iconized windows so that their icons are visible in rows at the bottom of the desktop. Open windows are not affected by this command.

7.10.20 Open Windows

Window > Open Windows lists the open windows 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.

More Windows	When the Windows list contains more files than can be displayed on the Window menu, More Windows is automatically added to the Windows menu. The command opens a dialog that displays the entire list of windows and lets you select the window you want to go to. To open a window from the list, either double-click the window name in the list with the left mouse button, 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.


Select Window

c:\mchp\picdem2\lcd74\kbd4bit.asm
c:\mchp\picdem2\lcd74\kbd8bit.asm
c:\mchp\picdem2\lcd74\kbd8bit.lst
c:\mchp\picdem2\lcd74\lcd4bit.asm
File Register Window
Program Memory Window
Project Window
Special Function Register Window
Stack Window
Trace Memory Window

Windows showing...
✓ Named Files
✓ Unnamed files
✓ Templates
✓ Command output

☐ Changed windows only

Figure 7.80: 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*.

7.11 Help Menu

7.11.1 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`.

7.11.2 Tool Release Notes

If you are using additional tools, there may be an item on the Help menu on release notes for those tools.

7.11.3 MPLAB IDE Help

Help > MPLAB Help contains help on using MPLAB IDE.

7.11.4 Editor Help

Help > Editor Help contains help on using the MPLAB Editor.

7.11.5 Error Help

Help > Error Help contains help on selected MPLAB IDE error messages.

7.11.6 MPASM Help

Help > MPASM Help opens an on-line version of the *MPASM User's Guide with MPLINK and MPLIB* with MPASM-specific information. Click on the green, underlined items to get more information.

MPASM Help also contains a Quick Reference Guide that includes assembly directives and device-specific instruction sets.

7.11.7 MPLINK Help

Help > MPLINK Help opens an on-line version of the *MPASM User's Guide with MPLINK and MPLIB* with MPLINK- and MPLIB-specific information. Click on the green, underlined items to get more information.

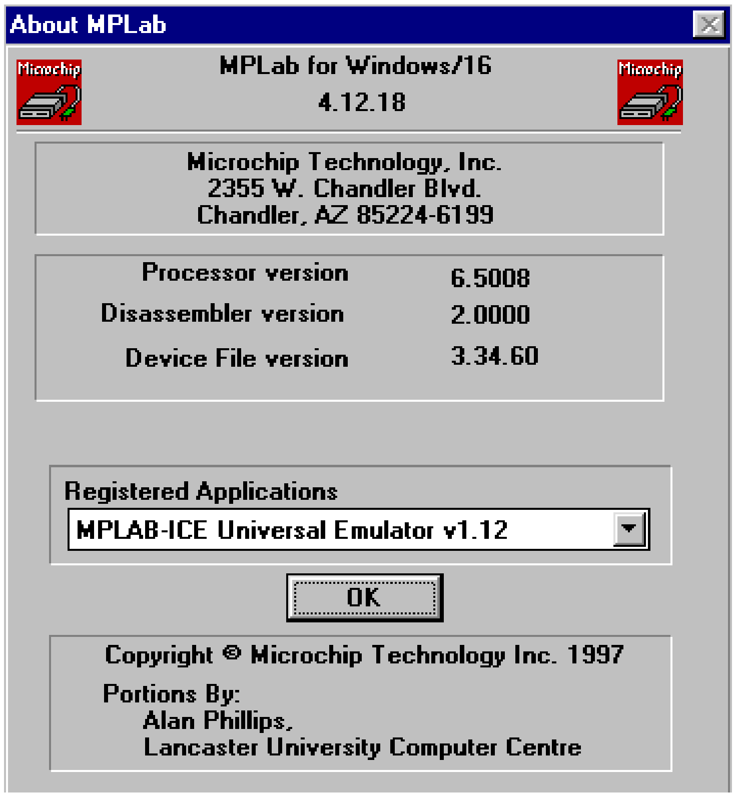
7.11.8 Tool Help

If you are using additional tools, there may be an item on the Help menu on help for those tools.

7.11.9 About

Help > About displays:

- MPLAB IDE Version
- Microchip's Address
- Processor Version
- Disassembler Version
- Information on other registered applications



MPLab for Windows/16
4.12.18

Microchip Technology, Inc.
2355 W. Chandler Blvd.
Chandler, AZ 85224-6199

Processor version	6.5008
Disassembler version	2.0000
Device File version	3.34.60

Registered Applications
MPLAB-ICE Universal Emulator v1.12

Copyright © Microchip Technology Inc. 1997
Portions By:
Alan Phillips,
Lancaster University Computer Centre

Figure 7.81: About Help Dialog

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

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

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”	

Key Map Function	Definition	Default Key Assignment
HelpMpasm	“MPASM 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 tool bar to the bottom of the window”	
OptionsToolbarFloat	“Makes the tool bar a floating window”	
OptionsToolbarHide	“Makes the tool bar invisible”	
OptionsToolbarLeft	“Moves the tool bar to the left of the window”	
OptionsToolbarRight	“Moves the tool bar to the right of the window”	
OptionsToolbarShow	“Makes the tool bar visible”	

Key Map Function	Definition	Default Key Assignment
OptionsToolbarTop	"Moves the tool bar 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

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”	

NOTES:

Appendix B. 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

NOTES:

Appendix C. MPLAB IDE Toolbar and Status Bar Definitions

C.1 MPLAB IDE Toolbars

C.1.1 Edit Toolbar

The Edit toolbar contains buttons that are commonly used when editing source code.

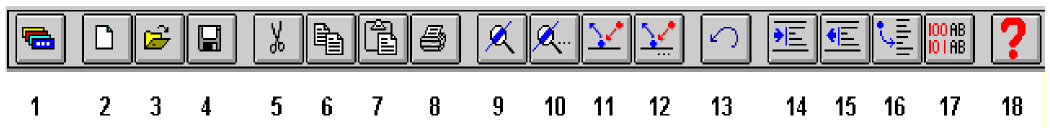


Figure 7.82: Edit Toolbar

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

C.1.2 Debug Toolbar

The Debug toolbar contains buttons that are commonly used when running and debugging code.

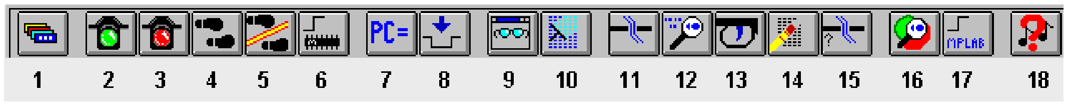


Figure 7.83: DebugToolbar

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

C.1.3 Project Toolbar

The Project toolbar contains icons that are commonly used when running and debugging code.

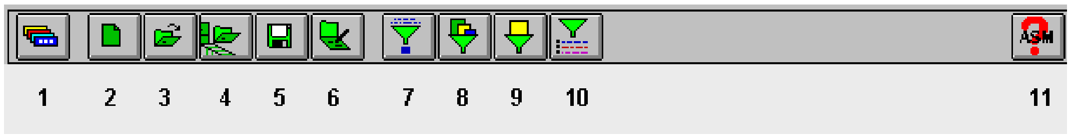


Figure 7.84: Project Toolbar

The default buttons are:

1. Change toolbar
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

C.1.4 User Defined Toolbar

The User Defined toolbar is intended to be customized to contain buttons that meet the individual user’s needs.

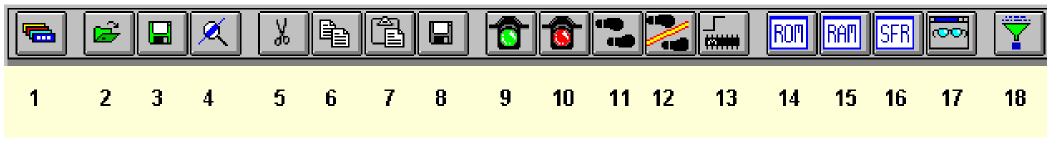


Figure 7.85: User Defined Toolbar

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

C.2 MPLAB IDE Status Bar

The Status Bar indicates such current information as cursor position, development mode and device, and active toolbar.

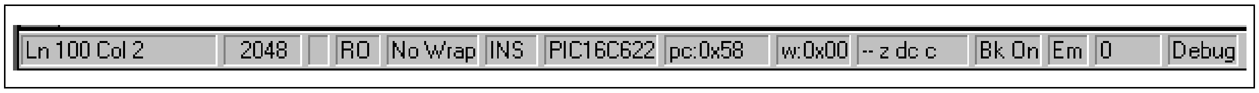


Figure 7.86: Status Bar

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 *Options > Current Editor Modes* 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
		Em = Emulator – PICMASTER emulator
			Displays Development Mode Dialog

Processor Frequency	4 MHz	Displays current processor frequency	Opens processor clock dialog
Current Tool Bar	Edit	Displays current tool bar	No Action

NOTES:

Appendix D. MPLAB Editor Default Key Commands

D.1 Introduction

This appendix describes the default key commands specific to the MPLAB Editor and lists the equivalent menu command (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.

D.2 Highlights

The categories for the default key commands are:

- Function Keys
- Movement Keys
- Control Keys
- Formatting and Editing Keys

D.3 Function Keys

F1	When pressed from a window (e.g., File Registers Window), displays the appropriate help topic.
F2	Executes the *Debug > Break Settings* command to open the Break Settings dialog.
F3	Executes the *Edit > Find* command to find strings.
F4	Executes the *Edit > Replace* command to replace strings.
F5	Executes the *Debug > Halt* command to halt debugging.
F6	Executes the *Debug > Reset* command to issue a reset to the emulated or simulated processor.
F7	Executes the *Debug > Step* command to execute a single opcode from program memory.
F8	Executes the *Debug > Step Over* command to step over a call instruction in program memory.
F9	Executes the *Debug > Run* command to issue a run to the emulated/simulated processor.
F10	Executes the *Project > Make Project* command to initiate a “make” for the current project.

F11	Executes the *Execute > DOS Command To Window* command to run a DOS command and capture its output in a window.
Shift+F3	Executes the *Edit > Repeat Find* command to repeat the last find operation.
Shift+F4	Executes the *Edit > Repeat Replace* command to repeat the last replace operation.
Shift+F5	Executes the *Debug > Halt Trace* command to halt the execution trace for the simulated processor.
Shift+F9	Executes the *File > Close All* command to close all files and windows.
Ctrl+F2	Executes the *Project > Open Project* command to open the Open Project Dialog.
Ctrl+F3	Executes the *Project > Edit Project* command to open the Edit Project Dialog.
Ctrl+F7	Executes the *Options > Environment Setup* to open the Environment Setup dialog.
Ctrl+F8	Executes the *Windows > Show Symbol List* command to open the Symbol List dialog.
Ctrl+F10	Executes the *Project > Build All* command to build all the source files for the current project.
Alt+F4	Executes the *File > Exit* command to end your MPLAB Editor session.
Alt+F10	Executes the *Project > Build Node* command to build the current node.
Shift+Ctrl+F2	Executes the *Debug > Clear Program Memory* command to clear all of program memory to an "erased" state.
Shift+Ctrl+F3	Executes the *Debug > System Reset* command to reset the entire emulator system.
Shift+Ctrl+F5	Executes the *Debug > POR Reset Emulation* command to open the POR Reset dialog.

D.4 Movement Keys

For these keys, adding Shift to the combination causes a selection to be extended.

Note that where the **Alt** key is combined with either an arrow key, or one of **Home**, **End**, **PgDn**, **PgUp**, Ins or **Del**, you must use the keys in the extended key areas, and not those in the numeric keypad.

Up	Moves the cursor up by one line
Shift+Up	Moves the cursor up by one line, extending the highlighting
Down	Moves the cursor down by one line
Shift+Down	Moves the cursor down by one line, extending the selection
Left	Moves the cursor left by one character
Shift+Left	Moves the cursor left by one character, extending the selection
Ctrl+Left	Moves the cursor left by one word
Ctrl+Shift+Left	Moves the cursor left by one word, extending the selection
Right	Moves the cursor right by one character
Shift+Right	Moves the cursor right by one character, extending the selection
Ctrl+Right	Moves the cursor right by one word
Ctrl+Shift+Right	Moves the cursor right by one word, extending the selection
PgDn	Moves the cursor down by one page
Shift+PgDn	Moves the cursor down by one page, extending the selection
Ctrl+PgDn	Moves the cursor to the start of the last line in the window
Ctrl+Shift+PgDn	Moves the cursor to the start of the last line in the window, extending the selection
PgUp	Moves the cursor up by one page
Shift+PgUp	Moves the cursor up by one page, extending the selection
Ctrl+PgUp	Moves the cursor to the start of the first line of the window
Ctrl+Shift+PgUp	Moves the cursor to the start of the first line of the window, extending the selection
Home	Moves the cursor to the start of the line
Shift+Home	Moves the cursor to the start of the line, extending the selection

Ctrl+Home	Moves the cursor to the start of the file
Ctrl+Shift+Home	Moves the cursor to the start of the file, extending the selection
Alt+Home	Moves the cursor to the first non-white-space character in the current line
Alt+Shift+Home	Moves the cursor to the first non-white-space character in the current line, extending the selection
End	Moves the cursor to the end of the line
Shift+End	Moves the cursor to the end of the line, extending the selection
Ctrl+End	Moves the cursor to the end of the file
Ctrl+Shift+End	Moves the cursor to the end of the file, extending the selection

D.5 Control Keys

Ctrl+Shift+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
Ctrl+C	Executes the *Edit > Copy* command to copy selected text to the clipboard
Ctrl+G	Executes the *Edit > Goto Line* command to move the cursor to a specific line
Ctrl+H	Deletes the character to the left of the cursor
Ctrl+I	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.
Ctrl+K	Executes the *Edit > Delete To End Of Line* command to delete everything from the cursor position to the end of the line
Ctrl+Shift+K	Executes the *Edit > Delete Line* command to delete the entire line that the cursor is in
Ctrl+N	Executes the *File > New* command to create a new, empty edit window
Ctrl+O	Executes the *File > Open* command to open an existing file
Ctrl+Shift+O	Splits the current line at the position of the cursor, leaving the cursor unmoved
Ctrl+P	Executes the *File > Print* command to print the file showing in the current window
Ctrl+Q	Executes the *Edit > Text Insert ASCII Code* command to insert a character specified by its ASCII code number
Ctrl+S	Executes the *File > Save* command to save the current file to disk
Ctrl+T	Executes the *Edit > Text Transpose Characters* command to transpose the character under the cursor with the one to its left
Ctrl+V	Executes the *Edit > Paste* command to paste data from the clipboard into the current window
Ctrl+W	Executes the *Window > Select* command to let you choose between many open windows
Ctrl+Shift+W	With the cursor between braces, this command highlights the closest (inner-most) pair of braces and all the text between them
Ctrl+X	Executes the *Edit > Cut* command to cut selected text to the clipboard
Ctrl+Z	Executes the *Edit > Undo* command to undo the last edit action

D.6 Formatting and Editing Keys

Enter	Inserts a new line
BackSpace	Deletes the character to the left of the cursor
Del	Deletes the character to the right of the cursor
Shift+Del	Executes the *Edit > Cut* command to cut selected text to the clipboard
Ins	Toggles the current window between Insert and Over-write modes
Shift+Ins	Executes the *Edit > Paste* command to paste data from the clipboard into the current window
Ctrl+Ins	Executes the *Edit > Copy* command to copy selected text to the clipboard
Tab	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.

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, MPLAB Editor
- MPASM, MPLINK, MPLIB
- MPLAB-CXX
- MPLAB-ICE, PICMASTER Emulators
- MPLAB-ICD
- 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.

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 MPLINK 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 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 MPASM permit sections of conditionally assembled code.

Data Directives

Data directives are those that control MPASM'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 device being emulated. The File Register window displays data memory.

Directives

Directives provide control of the assembler's operation by telling MPASM 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 emulator system includes the pod, processor module, device adapter, cables, and MPLAB Software. The PICMASTER emulator system includes the pod, device-specific probe, cables, and MPLAB Software.

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 MPASM'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.

Note: MPASM expressions are evaluated in 32 bit integer math. (Floating point is not currently supported.)

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 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 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.

ICE

In-Circuit Emulator. MPLAB-ICE 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 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 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 µC.

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 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

Microchip Technology's relocatable macro assembler. MPASM is a DOS or Windows-based PC application that provides a platform for developing assembly language code for Microchip's PICmicro microcontroller families. Generically, MPASM will refer to the entire development platform including the macro assembler and utility functions.

MPASM will translate source code into either object or executable code. The object code created by MPASM 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

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 Software resides on the PC host. The executable file name is MPLAB.EXE. MPLAB.EXE calls many other files.

MPLAB-SIM

Microchip's simulator that works with MPLAB IDE.

MPLIB

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 will combine multiple object files into one library file. Then MPLIB can be used to manipulate the object files within the created library.

MPLINK

MPLINK is a linker for the Microchip relocatable assembler, MPASM, and the Microchip C compilers, MPLAB-C17 or MPLAB-C18. MPLINK also may be used with the Microchip librarian, MPLIB. MPLINK is designed to be used with MPLAB IDE, though it does not have to be.

MPLINK will combine object files and libraries to create a single executable file.

MPSIM

The DOS version of Microchip's simulator. MPLAB-SIM 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 MPASM or MPLAB-C17/C18 that can be run through MPLINK to create executable code. Object code is contained in an object file.

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® 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 is the newest emulator from Microchip.

PICSTART Plus

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 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, 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

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 devices have a PWM peripheral.

Qualifier

An address or an address range used by the Pass Counter or as an event before another operation in a complex trigger.

Radix

The number base, hex, or decimal, used in specifying an address 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 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.

Shared Section

A section which resides in a shared (non-banked) region of data RAM.

Shell

The MPASM shell is a prompted input interface to the macro assembler. There are two MPASM 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 though code, one instruction at a time. After each instruction, MPLAB IDE updates register windows, watch variables, and status displays so you can analyze and debug instruction execution.

You can also single step C compiler source code, but instead of executing single instructions, MPLAB IDE will execute all assembly level instructions generated by the line of the high level C statement.

Skew

The information associated with the execution of an instruction appears on the processor bus at different times. For example, the executed 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 instructions and MPASM 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 machine code by a compiler. Machine code is suitable for use by a PICmicro MCU or Microchip development system product like MPLAB IDE.

Source File - Assembly

The ASCII text file of PICmicro instructions and MPASM 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 tool bar.

Step Into

This command is the same as Single Step. Step Into (as opposed to Step Over) follows a CALL instruction into a subroutine.

Step Over

Step Over allows you to debug code without stepping into subroutines. When stepping over a CALL instruction, the next 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.

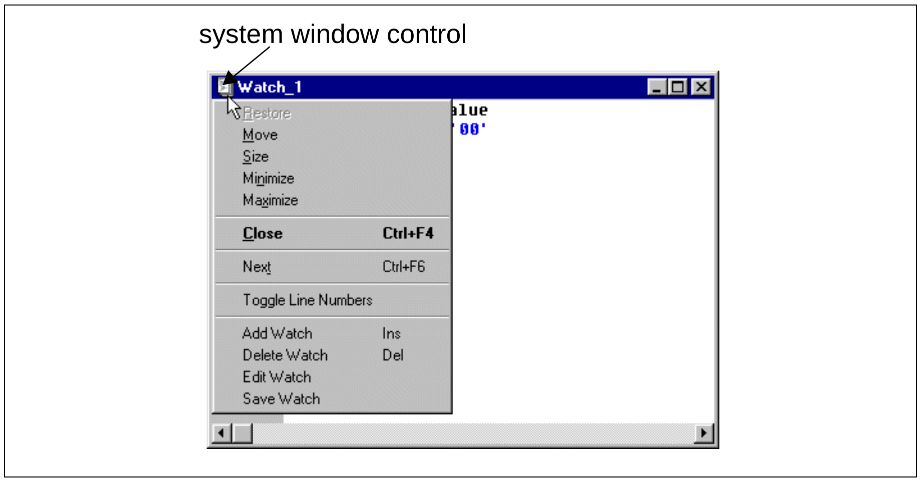


Figure G1: System Window Control Menu - Watch Window

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.

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.

Tool Bar

A row or column of icons that you can click on to execute MPLAB IDE functions.

Trace

An emulator or simulator function that logs program execution. The emulator logs program execution into its trace buffer which is uploaded to MPLAB IDE’s trace window.

Trace Memory

Trace memory contained within the emulator. Trace memory is sometimes called the trace buffer.

Trigger Output

Trigger output refers to an emulator output signal that can be generated at any address or address range, and is independent of the trace and 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.

Index

NOTES:

NOTES:

NOTES:

MICROCHIP

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-786-7200 Fax: 480-786-7277
Technical Support: 480-786-7627
Web Address: http://www.microchip.com

Atlanta
Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
Microchip Technology Inc.
4570 Westgrove Drive, Suite 160
Addison, TX 75248
Tel: 972-818-7423 Fax: 972-818-2924

Dayton
Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Detroit
Microchip Technology Inc.
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

AMERICAS (continued)

Toronto
Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Beijing
Microchip Technology, Beijing
Unit 915, 6 Chaoyangmen Bei Dajie
Dong Erhuan Road, Dongcheng District
New China Hong Kong Manhattan Building
Beijing 100027 PRC
Tel: 86-10-85282100 Fax: 86-10-85282104

Hong Kong
Microchip Asia Pacific
Unit 2101, Tower 2
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

India
Microchip Technology Inc.
India Liaison Office
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-0061 Fax: 91-80-229-0062

Japan
Microchip Technology Intl. Inc.
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
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Shanghai
Microchip Technology
Unit B701, Far East International Plaza,
No. 317, Xianxia Road
Shanghai, 200051 P.R.C
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

ASIA/PACIFIC (continued)

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan, R.O.C
Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Denmark ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Arizona Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 München, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Arizona 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
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5858 Fax: 44-118 921-5835

01/21/00

DNV Certification, Inc.
USA

ANSI • RAB
QMS
ACCREDITED

DNV

DNV MSC
The Netherlands
Accredited by the RvA

ISO 9001 / QS-9000
REGISTERED FIRM

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