MICROCHIP

PRO MATE® II

USER'S GUIDE

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

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable".
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

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

Trademarks

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

Total Endurance, ICSP, In-Circuit Serial Programming, FilterLab, MXDEV, microID, *Flex*ROM, *fuzzy*LAB, MPASM, MPLINK, MPLIB, PICC, PICDEM, PICDEM.net, ICEPIC, Migratable Memory, FanSense, ECONOMONITOR, Select Mode and microPort are trademarks of Microchip Technology Incorporated in the U.S.A.

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

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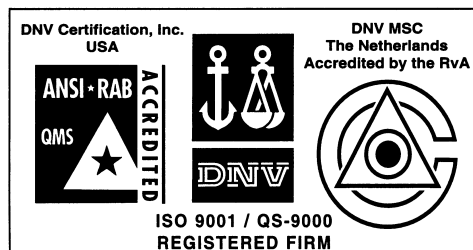
Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices,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

PRO MATE II USER'S GUIDE

Table of Contents

PRO MATE II User's Guide

Table of Contents

PRO MATE II User's Guide

Quick Start

Introduction

This chapter provides the limited information that experienced users of PCs and embedded application development tools need in order to quickly start using PRO MATE II.

Highlights

Topics covered in this chapter:

- Installing PRO MATE II
  - Installing MPLAB® IDE and PRO MATE II
  - Installing PRO MATE II Command Line Interface
- Connecting the PRO MATE II to your PC
- Using PRO MATE II with MPLAB IDE
  - Enabling (starting) PRO MATE II
  - Setting up the Device Programmer and Configuration Bits
  - Programming, Verifying, and Reading a Device
- Using PROCMD
- Using PRO MATE II in Stand-Alone Mode

Installing PRO MATE II

Installing MPLAB® IDE and PRO MATE II

1. PRO MATE II is a component of the MPLAB Integrated Development Environment (IDE). To install the MPLAB IDE software, insert the MPLAB IDE installation CD and run `MPXXXX.EXE`, where `XXXX` represents the version of the MPLAB IDE software. This executable is in the root directory of the CD.

2. When the component selection window of the installation program appears, make sure PRO MATE II is checkmarked in the list of components to install. As a minimum, install the MPLAB IDE software and MPASM™/MPLINK™. You should also install the MPLAB-SIM and help files. Continue with the installation by following the instructions displayed by the installation program. Under Language Components, select All.

If you previously installed the MPLAB IDE software but did not install PRO MATE II, you must repeat the installation. You can clear all the checkmarks except PRO MATE II.

> **Note:** The `setup.cfg` file is used in programming KEELOQ® devices. It is created whenever you install the MPLAB IDE software. If you have to reinstall the MPLAB IDE software, create a backup of your `setup.cfg` file first in order to save your KEELOQ programming settings.

Installing PRO MATE II Command Line Interface

- If you have installed the MPLAB IDE software, execute `procmd.exe` from DOS or a Windows DOS shell. It is in the MPLAB directory.

- If you have not installed the MPLAB IDE software, you will need to get a copy of PROCMD and its associated files. You may obtain PROCMD from the MPLAB directory of the MPLAB IDE installation CD-ROM or from our Internet Web Site at http://www.microchip.com. Select Development Tools, PRO MATE II, Command Line Interface.

Connecting PRO MATE II to Your PC

Follow these steps to connect the PRO MATE II to your PC.

> **Note:** Always make sure that the device programmer is powered OFF before installing the socket module.

1. First, install a socket module on your PRO MATE II. (For more information, see Chapter 2 or Appendix A.) Align the tip of the arrow on the socket module with the tip of the arrow on the PRO MATE II. Tighten the two socket module thumbscrews evenly and, if possible, simultaneously. Avoid over tightening them; they should be finger-tight only.

> **Note:** Socket modules do not come with the device programmer. You must order your socket module(s) separately. Socket modules are available to accommodate each device package. The Development System Ordering Guide (DS30177) describes the available socket modules.

2. Use the RS-232 communications cable to connect the PRO MATE II to an available COM port on your PC.
3. Make sure that the power switch on the back of the unit is in the OFF position. Install the power supply.
4. Turn the power switch on the back of the PRO MATE II to ON.

If the message "Socket Not Supported" appears, ensure that the module is properly installed into the PRO MATE II.

You are now ready to start using PRO MATE II with the MPLAB IDE.

PRO MATE II User's Guide

Using PRO MATE II with MPLAB

Starting PRO MATE II

1. Start the MPLAB IDE by double-clicking the MPLAB desktop icon or by selecting _Programs > Microchip MPLAB > MPLAB_ from your Start menu.
2. Select _Options > Programmer Options > Communications Port Setup_ from the MPLAB menu. Select the COM port that the PRO MATE II is on.
3. If the PRO MATE II menu item is not on the MPLAB menu, you must select the PRO MATE II programmer. In the MPLAB desktop, select _Options > Programmer Options > Select Programmer_. Select PRO MATE Device Programmer and click **OK**. A message will prompt you to restart MPLAB in order for the change to take effect. Click **Yes**. MPLAB will shut down, and the PRO MATE menu will appear on the MPLAB menu when you restart MPLAB.
4. Select _Options > Development Mode_. Select the MPLAB-SIM or Editor Only development mode and select the device you are going to program. Click **OK**.
5. If you are going to use the Hex file from a project to program the device, open that project now (_Project > Open Project_.)
6. Select _PRO MATE > Enable Programmer_ from the MPLAB menu. The MPLAB IDE will attempt to establish communications with the PRO MATE, and the PRO MATE Device Programmer and Configuration Bits dialogs will appear.
7. If you get a message stating that there is a newer PRO MATE operating system available, refer to Section 7.12 for instructions on how to update your operating system.

Setting Up the Device Programmer and Configuration Bits

The PRO MATE Device Programmer dialog is open whenever the programmer is enabled. Closing this dialog disables the programmer.

1. Make sure the correct device is displayed. If the device you want to program is not listed, you must upgrade your PRO MATE II operating system. See Section 7.12.
2. If you want to identify the programming on a device (e.g., version control), click **Device ID** and edit the device ID or select unprotected checksum. Click **OK**.
3. If your application runs at the extreme voltage operating range, select the correct voltages.
4. Set the configuration bits in the Configuration Bits dialog. If you set configuration bits in your source code, this dialog will be updated with those values when you rebuild your project. The values in this dialog will be programmed into the device you program.

   If the Configuration Bits dialog is not visible, click **Configuration Bits** in the PRO MATE Device Programmer dialog to reopen it.

Loading a Hex File

If you are not using an MPLAB project, but have a hex file ready for programming into a device, select *File > Import > Import to Memory* to load the hex code into the MPLAB Program Memory window. Select *Window > Program Memory* to view the contents of program memory.

Checking for a Blank Device

Insert the device to be programmed into the PRO MATE II socket.

Click **Blank** in the PRO MATE Device Programmer dialog to verify that the device is blank (all bits set to '1').

If you are programming a one-time programmable (OTP) device, use the Configuration Bits dialog to set the configuration bits to their factory settings. Select *PRO MATE > Blank Check OTP* to make sure that all program memory bits are set to '1' and that the configuration bits match the value in the Configuration Bits dialog.

Programming, Verifying, and Reading a Device

If you have followed the previous sections, you are ready to program a device. To program the entire device, click **Program** in the PRO MATE Device Programmer dialog.

> **Note:** If you are programming a windowed calibration device, be sure to store its calibration data as described in Section 7.10 to ensure proper operation of the device.

To program selectively (part of program memory or configuration bits) select *PRO MATE > Program/Verify*. Select the memory range and items to be programmed, and click **Program** in the Program/Verify dialog.

When programming has completed, "Success" or "Failure" will appear to the right of the Start Address. An error window displaying the expected and actual data will appear if programming failed. If the bad data shows 0000, try reseating the socket module and doing a blank check before trying to program the device again.

Using PRO MATE II with PROCMD

You can use PRO MATE II on an IBM-80386 compatible PC without the MPLAB IDE. The user interface is through DOS or a DOS shell from Windows.

Starting PROCMD

Execute the file `procmd.exe` to start the PRO MATE II command line interface. For example, use the command `c:\promate\procmd.exe` if you created a directory called `promate`.

Programming a Device

You can write the contents of a hex file directly to the device in the PRO MATE II socket module from the DOS prompt. You do not need to send the hex file every time a device is programmed. Instead, you can use one command to transfer the hex file and set up any voltages, and then use subsequent command line executions to program.

To program a device, make sure the programmer is set up and turned on. Place the device in the socket module. Then execute the following:

PROCMD /<number> /p<partname> /f<filename> /m

where:

   `<number>` is 1, 2, 3, or 4 to indicate the COM port you are using
   `<partname>` is the name of the device
   `<filename>` is the name of the hex file
   `/m` is the command to program the part

Example:

PROCMD /1 /p16c74a /f16c74a.hex /m

will program a PIC16C74A device with the contents of the file `16c74a.hex`, using COM1 to communicate with PRO MATE II. Only the addresses specified in `16c74a.hex` are programmed. All values not specified in the hex file are set to blank (erased) values.

To program the entire contents of the PRO MATE II device programmer into the device, you do not need to specify a hex file.

Using PRO MATE II in Stand-Alone Mode

Starting PRO MATE II in Stand-Alone Mode

When you power-up the PRO MATE II device programmer, the unit automatically detects the type of socket module installed and initializes the PRO MATE II function buttons. The device programmer then displays the device options for the currently installed socket module. If you power-on the device programmer without a valid socket module installed, the unit displays the message "Socket Not Supported."

> **Caution:** Ensure the device programmer is powered OFF before changing a socket module.

After you select a device, the device programmer displays the Command menu. The function buttons allow you to perform the basic tasks for programming a microcontroller device: Program, Verify, and Read. Main will return you to the main menu.

Setting Up PRO MATE II for Stand-Alone Mode

PRO MATE II operating in stand-alone mode allows you to read, program, and verify a device without using a PC. Stand-Alone mode is useful in situations where a PC may not be required, such as in the field or in a lab production environment. However, you will need a PC to download the hex file into PRO MATE II memory in order to set up the PRO MATE II for stand-alone mode.

To download a file to the PRO MATE II, use the following command:

PROCMD /<number> /p<partname> /f<filename>

where:

    <number> = 1, 2, 3 or 4, depending on the COM port you are using
    <partname> is the name of the device
    <filename> is the name of the file (/f for hex, /s for SQTP)

Example:

PROCMD /1 /p16c74a /f16c74a.hex /m

will send the file 16c74a.hex, used to program PIC16C74A devices, via COM1 to the PRO MATE II. The /m command is to program the device.

Now you are ready to use PRO MATE II in stand-alone mode.

PRO MATE II User's Guide

Programming the Device

With the programmer on, place the device to be programmed into the socket module and press <F2> on the PRO MATE II. The device programmer programs the contents of its memory into the microcontroller device loaded in the socket module.

After programming a device without errors, the device programmer performs a check to verify the data programmed into the device, and returns the results of the verification. For the installed device, the device programmer performs the verification at the V_{DD} Minimum and V_{DD} Maximum voltages.

Verifying the Programming

If you want to compare the contents of the programmer's internal memory to the contents of the programmed microcontroller device in the socket module, press <F3>. If the data and configuration bit settings are correct, VERIFIED will display on the LCD. The device programmer reports errors according to which part of the device failed.

The Verify function also confirms that erased parts are blank. If all programmable locations are blank for a device loaded in the socket module, the device programmer displays ERASED.

Copying a Device's Programming

To copy the contents of a programmed device onto other devices, press <F1> to read the contents of the programmed device into the PRO MATE II's memory. You can then replace the programmed device with an unprogrammed device and press <F2> to program it.

PRO MATE II USER'S GUIDE

General Information

Introduction

This chapter contains general information that will be useful to know before using PRO MATE II.

Highlights

Topics covered in this chapter:

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

About This Guide

Document Layout

This document describes how to use PRO MATE II as a development tool to program firmware to a target device. The manual layout is as follows:

- **Chapter 1: PRO MATE II Preview** – Describes the PRO MATE II and how it works.

- **Chapter 2: Installing and Setting Up PRO MATE II** – Describes how to install PRO MATE II hardware and MPLAB software. Explains how to set up the MPLAB IDE and PRO MATE II to work together and how to start PRO MATE II from MPLAB.

- **Chapter 3: Programming Examples** – Contains several examples (tutorials) for programming calibration memory devices, memory devices, and other PICmicro® MCU devices.

- **Chapter 4: Using PRO MATE II with MPLAB** – Provides step-by-step instructions on using PRO MATE II with the MPLAB IDE to program, read, and verify devices.

- **Chapter 5: Using PRO MATE II with PROCMD** – Provides step-by-step instructions on using the PRO MATE II command line interface to program, read, and verify devices.

PRO MATE II User's Guide

- **Chapter 6: Using PRO MATE II in Stand-Alone Mode** – Provides step-by-step instructions on using the PRO MATE II without a PC to program, read, and verify devices.

- **Chapter 7: PRO MATE II – MPLAB Reference** – Describes PRO MATE II dialogs and menu options.

- **Chapter 8: PROCMD Reference** – Describes the commands available through the PRO MATE II command line interface as well as error messages.

- **Chapter 9: PRO MATE II Stand-Alone Mode Reference** – Describes the operations you can perform using PRO MATE II without a PC.

- **Appendix A: Connecting to a 25-Pin Serial Port** – Describes how to connect PRO MATE II to a 25-pin serial port.

- **Appendix B: Socket Module Cleaning** – Provides instructions on cleaning PRO MATE II socket modules.

- **Appendix C: Troubleshooting** – Provides information on solving common problems.

- **Index** – Provides a cross-reference listing of terms, features, and sections of this document.

- **Worldwide Sales and Service** – Lists 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 the MPLAB IDE, PRO MATE II, and other Microchip tools are constantly evolving to meet customer needs, some MPLAB dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site to obtain the latest documentation available.

Documentation Numbering Conventions

Documents are numbered with a "DS" number. The number is located on the bottom of each page, in front of the page number. The numbering convention for the DS Number is: DSXXXXXA,

where:

XXXXX	=	The document number.
A	=	The revision level of the document.

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 PRO MATE II. You may also find the data sheets for specific microcontroller devices informative in developing firmware.

README.PRO, README.CMD

For the latest information on using PRO MATE II, read the `README.PRO` file (ASCII text file) included with the PRO MATE II software. For the latest information on using PROCMD, the DOS command-line control program for PRO MATE II, read the `README.CMD` file (ASCII text file) included with the PRO MATE II software. These README files contain updated information that may not be included in this document.

ICSP Socket Module User's Guide (DS51113)

The In-circuit Serial Programming (ICSP) socket module allows you to use the PRO MATE II to perform in-circuit serial programming of microcontroller. This document describes how to use this socket module.

MPLAB IDE User's Guide (DS51025)

Comprehensive guide that describes installation and features of Microchip's MPLAB Integrated Development Environment, as well as the editor and simulator functions in the MPLAB environment.

MPASM User's Guide with MPLINK & MPLIB™ (DS33014)

Describes how to use Microchip Universal PICmicro Microcontroller Assembler (MPASM), Linker (MPLINK) and Librarian (MPLIB).

Technical Library CD-ROM (DS00161)

This CD-ROM contains comprehensive data sheets for Microchip PICmicro devices available at the time of print. To obtain this disk, contact the nearest Microchip Sales and Service location (see back page) or download individual data sheet files from the Microchip website (http://www.microchip.com).

Embedded Control Handbook Vol.1 & 2 (DS00092 & DS00167)

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.

Troubleshooting

See Appendix B for information on common problems.

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 Web Site

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

http://www.microchip.com

The file transfer site is available by using an FTP program/client to connect to:

ftp://ftp.microchip.com

The website 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 information includes:

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

Development Systems Customer Notification Service

Microchip started the 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 WWW page for other Microchip list servers.

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 programmers 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 programmers 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®.

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 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's MPLAB IDE, the Windows Integrated Development Environment for development systems tools. This list is focused on MPLAB, MPLAB-SIM, MPLAB's Project Manager and general editing and debugging features. For specific information on MPLAB compilers, linkers and assemblers, subscribe to the COMPILERS list. For specific information on MPLAB emulators, subscribe to the EMULATORS list. For specific information on MPLAB 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

The latest information on other development system tools provided by Microchip. For specific information on MPLAB 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. PRO MATE II Preview

1.1 Introduction

This chapter presents an overview of the features and requirements of PRO MATE II.

1.2 Highlights

Topics covered in this chapter:

- What PRO MATE II Is
- What PRO MATE II Does
- Components of PRO MATE II System
- PRO MATE II CE Compliance
- PRO MATE II and PRO MATE
- How PRO MATE II Helps You
- PRO MATE II Operating with a PC
- PRO MATE II Operating without a PC (Stand-Alone)
- MPLAB Integrated Development Environment
- MPLAB Development Tools

1.3 What PRO MATE II Is

PRO MATE II is a Microchip microcontroller device programmer. Through interchangeable programming socket modules PRO MATE II enables you to quickly and easily program the entire line of Microchip PICmicro microcontroller devices, many of the Microchip memory parts, and the KEELOQ® Code Hopping Encoders.

PRO MATE II may be used with the Microsoft Windows operating systems running the MPLAB Integrated Development Environment (IDE), with DOS using the command-line controller PROCMD, or as a stand-alone programmer.

1.4   What PRO MATE II Does

You can set up PRO MATE II on any serial communications port on your PC. With PRO MATE II you can do the following operations:

- Program memory, configuration bits, ID locations, and calibration data into PICmicro devices.
- Verify that PICmicro microcontrollers are blank.
- Verify that code in the target microcontroller matches your firmware.
- Read code from an unprotected PICmicro microcontroller into the MPLAB IDE's program memory window for debugging and programming into other PICmicro devices.
- Program unique serialized ID numbers into your firmware using Serial Quick Turn Programming (SQTP) files.
- Program many Microchip Memory parts.
- Program KEELOQ Code Hopping devices.

1.5   Components of the PRO MATE II System

The PRO MATE II device programmer system consists of the following:

1. PRO MATE II device programmer
2. Socket module(s)

> **Note:**   A complete line of socket modules is available. You may order the socket modules for the device that you will be programming separately.

3. RS-232 interface cable to connect to any standard PC serial port
4. 9V power supply (not shown)
5. MPLAB software, an Integrated Development Environment including a text editor, project manager, MPASM assembler, and MPLAB-SIM debugger (not shown)
6. Blank chips for programming
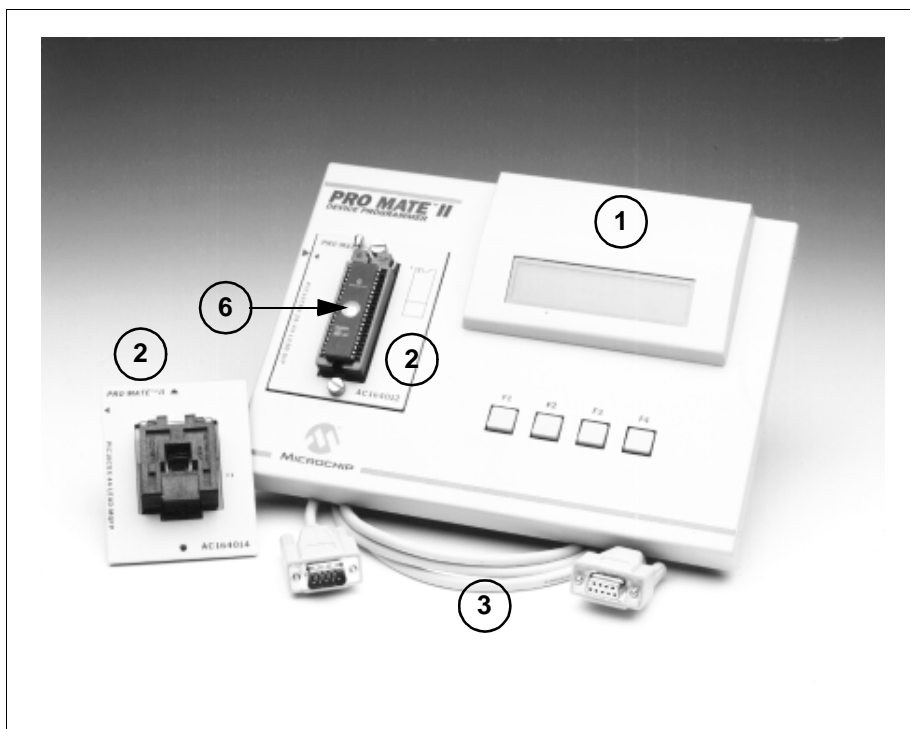
Figure 1.1: PRO MATE II System

1.6   PRO MATE II CE Compliance

The PRO MATE II development system is designed, tested, and certified to meet the Electromagnetic Compatibility requirements known as the CE compliance directives. These standards, set by the European Union (EU) countries, include limiting radiated emission, reducing susceptibility to radiated emission, and reducing susceptibility to Electrostatic Discharge (ESD).

1.7   PRO MATE II and PRO MATE

PRO MATE II is the second generation PRO MATE device programmer from Microchip. Information in this document pertaining to the PRO MATE II is equally valid for the PRO MATE programmer, unless otherwise specified.

1.8 How PRO MATE II Helps You

With the PRO MATE II device programmer, you can program Microchip devices from a PC Host, or you can use the device programmer as a stand-alone unit.

- PRO MATE II is easy to use and flexible in programming various Microchip devices and package types.

- PRO MATE II will expand to support future Microchip devices, always providing the latest programming algorithms to support Microchip PICmicro microcontroller devices and other Microchip parts via the Microchip Internet Web Site (http://www.microchip.com).

1.9 PRO MATE II Operating with a PC

When connected to a PC, PRO MATE II provides two methods of control.

1.9.1 MPLAB Windows IDE

Using MPLAB Integrated Development System (IDE) as the interface, PRO MATE II becomes another tool in the MPLAB IDE, allowing you to quickly compile, test, and debug your firmware, then download it into PRO MATE II to be programmed into your device.

1.9.2 DOS Command Line Interface

The PROCMD program is a DOS command line interface to the PRO MATE II device programmer. This interface is designed for programming devices in a production environment with limited resource PCs (80286 or better).

1.10 PRO MATE II Operating without a PC (Stand-Alone)

Without a PC connection to PRO MATE II, the unit operates as a stand-alone device programmer. However, a PC connection is required for initial stand-alone setup (hex code download) or operating system updates. The main programmer features of the PRO MATE II are available, including Read, Program, Verify.

1.11 MPLAB Integrated Development Environment

The MPLAB desktop provides an environment for developing and debugging your application. PRO MATE II is integrated into the MPLAB IDE, but you do not need the MPLAB IDE to use PRO MATE II.

This document covers the basic setup and operation of the PRO MATE II device programmer, but it does not cover all functions of the MPLAB IDE. Read the *MPLAB IDE User's Guide* (DS51025) to get a full understanding of the features and debug capabilities of the MPLAB IDE.

1.12   MPLAB 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, source code is rebuilt and downloaded 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-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 Universal Assembler/MPLINK Relocatable Linker/ MPLIB Librarian**

    The MPASM assembler allows source code to be assembled without leaving the 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-C17 and MPLAB-C18 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, and then use one of these tools to program devices. This can all be accomplished with the MPLAB IDE. Although PRO MATE II does not require MPLAB IDE to operate, programming is easier using the MPLAB IDE.

- **Third Party Tools**

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

PRO MATE II User's Guide

NOTES:

Chapter 2. Installing and Setting Up PRO MATE II

2.1   Introduction

This chapter describes how to install PRO MATE II hardware and software.

2.2   Highlights

Topics covered in this chapter:

- Host Computer System Requirements
- Installing PRO MATE II Hardware
- Installing MPLAB Software
- Installing PROCMD Software
- Powering Up PRO MATE II
- Configuring the Serial Port for PRO MATE II
- Selecting the PRO MATE II Programmer
- Setting Up the MPLAB Development Mode
- Enabling (Starting) PRO MATE II

2.3   Host Computer System Requirements

To use PRO MATE II with a PC, you may use either the MPLAB IDE (the Windows program, recommended) or PROCMD (the DOS command line program).

The minimum configuration that is required to run MPLAB IDE is described in the *MPLAB IDE, Simulator, Editor User's Guide*.

To run `procmd.exe`, you must have:

- MS-DOS® 5.0 or later
- EGA (or better) monitor
- 1 MB of memory
- 1.44 Megabyte floppy disk drive, 3.5"
- Hard drive
- Serial port

2.4   Installing PRO MATE II Hardware

The PRO MATE II hardware is simple to set up. First, install the socket module, then attach the communications cable. Finally, connect the power supply.

2.4.1   Installing a Socket Module

Socket modules are not provided with the device programmer. You must order your socket module(s) separately. Socket modules are available to accommodate each device package. The *Development System Ordering Guide* (DS30177) describes the available socket modules. Also, the `readme.pro` file lists socket module support for each device.

Once you have the required socket module, insert it as described below. (For more information on installation, see Appendix A.)

> **Note:** Always make sure that the device programmer is powered OFF before installing the socket module.

- Align the tip of the arrow on the socket module (◀) with the tip of the arrow on the PRO MATE II (▶).

> **Note:** PRO MATE II socket modules will not work with the PRO MATE programmer. You must obtain an adapter kit. See the *Development System Ordering Guide* (DS30177).

- Tighten the two socket module thumbscrews evenly and, if possible, simultaneously. Avoid overtightening them; they should be finger-tight only.

> **Note:** The gold connector strips on PRO MATE II are relatively fragile. Avoid touching them with the socket module screws, and avoid over-tightening the screws.

- After changing a socket module, turn on the PRO MATE II and install a device in the socket module, matching pin 1 of the device to the "1" on the socket module. Perform a blank check to ensure that the socket module is making proper contact. If the device is blank, the PRO MATE II display will indicate that the device is erased.

2.4.2   Installing the Communications Cable

PRO MATE II provides communications with the host PC via an RS-232 9-pin D-type connector. PRO MATE II is data communication equipment (DCE), and hardware handshaking is via clear-to-send (CTS) and request-to-send (RTS).

A 6-foot data cable with DB-9 connectors is supplied with PRO MATE II. All lines on the data cable are wired straight through. This cable is not a null modem cable.

- Connect one end of the cable to an available COM port on your PC. Check your PC setup to see which communications port is available. Usually a mouse device is connected to COM1 or COM2. If you have a modem, you might not have a third serial port on your PC.
- Connect the cable from the PC COM port to the RS-232 connector on the back of the PRO MATE II.

2.4.3   Installing the Power Supply

PRO MATE II comes with a 9V power supply. The earlier PRO MATE programmer came with a 5V power supply.

The PRO MATE II requires +9 volts at 750 mA. The power connector is a 2.5 mm DC power jack (Switchcraft P/N RAPC-712). A 0.75A fuse (Littlefuse Nano2Smf, P/N 451.750) is located near the +9V input on the PRO MATE II motherboard for circuit protection.
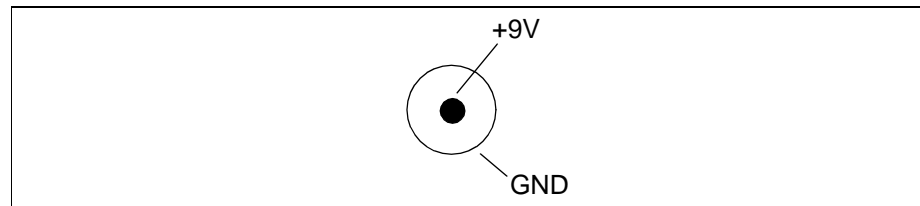

Figure 2.1: Back of PRO MATE II

The PRO MATE device programmer requires +5 volts ±5% at 750 mA on the 5-pin DIN connector (5 VDC IN). Pin 3 is +5 Volts, pins 1, 2, and 4 are ground, and pin 5 is not connected.

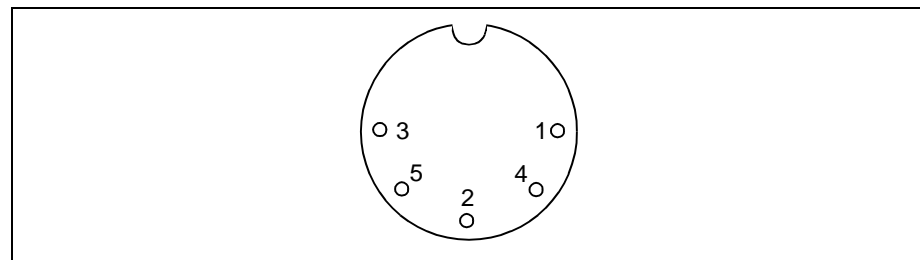**Figure 2.2: Back of PRO MATE Programmer**

- Make sure that the power switch on the back of the unit is in the "OFF" position.
- Plug the power supply into a power socket and connect the power supply cable to the unit.

2.5   Installing MPLAB Software

If you are going to use PRO MATE II with a PC running Windows, we recommend that you install the MPLAB software. You should install the MPLAB software by following the instructions in the *MPLAB IDE User's Guide* (DS51025). A brief summary of this procedure is discussed next.

- Insert the MPLAB CD-ROM into your CD-ROM drive (Example: drive D).
- Run the Setup program.
  - **For Windows 3.x/NT 3.51:** From the Program Manager Run option, type D:\setup.exe.
  - **For Windows 95/98, Windows NT 4.0, Windows 2000:** From the Start menu, select **Run**. In the Run dialog, type D:\setup.exe, where D is the CD-ROM drive.

> **Note:** The setup.cfg file is used in programming KEELOQ devices. It is created whenever you install MPLAB. If you have to reinstall the MPLAB software, create a backup of your setup.cfg file first in order to save your KEELOQ programming settings.

- Follow the on-screen instructions to install the MPLAB software. Be sure the checkbox to install PRO MATE Support Files is checked.
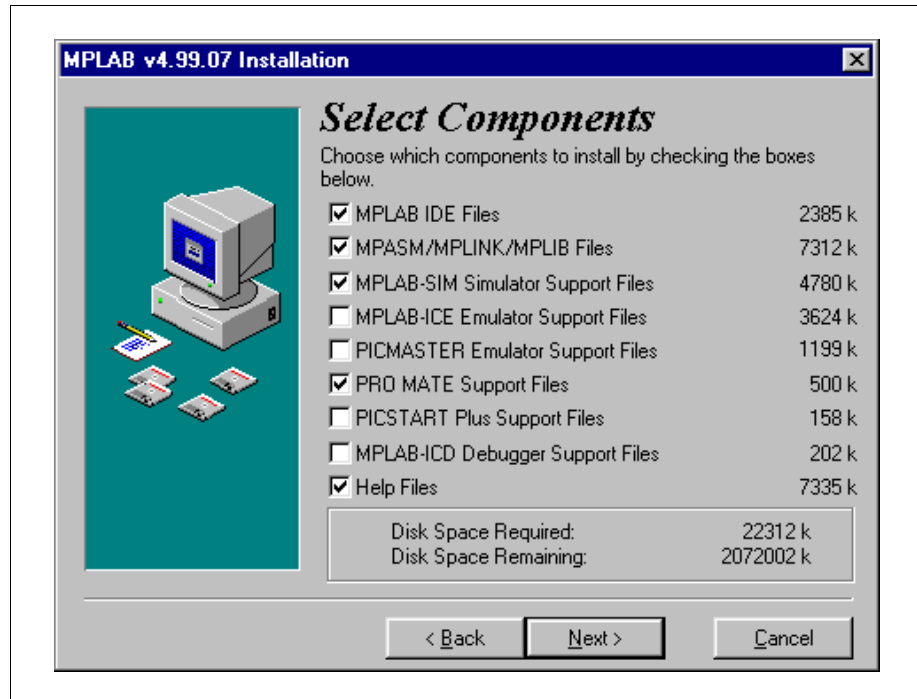
Figure 2.3: Select Components Dialog

2.6 Installing PROCMD Software

If you are going to use PRO MATE II with a PC running DOS or from a DOS shell in the Microsoft Windows operating system, you will need to install PROCMD software.

- If you have installed the MPLAB software, you already have PROCMD installed in the MPLAB directory. Execute `procmd.exe` from DOS or a Windows DOS shell.

- If you have not installed the MPLAB software, you will need to get a copy of PROCMD and its associated files. You may obtain PROCMD from the MPLAB directory of the MPLAB IDE installation CD-ROM or from our Internet Web Site at http://www.microchip.com. The required files are:

    - `procmd.exe`
    - `mplab.dev`

Create a directory in which to place these files and run them from there. For example, use the command `c:\promate\procmd.exe` if you created a directory called `promate`.

2.7   Powering Up PRO MATE II

Once you have connected the hardware and installed the software, you are ready to turn on PRO MATE II. Turn the power switch on the back of the PRO MATE II to ON. You should see the following types of messages appear on the LCD panel on the front of the PRO MATE II:

1. PRO MATE II Hardware Self Check
2. PRO MATE II Device Programmer
3. Version number and copyright dates
4. Type of socket module or family selection options
5. Device selection options

If the message "Socket Not Supported" appears, make sure the socket module is properly installed into PRO MATE II (see Section 2.4.1). If the installation is correct, you may need to update the PRO MATE II operating system for support of the installed socket module (see Section 7.12). If the operating system is the latest update, you may have a bad socket module. Please contact Microchip support (see General Information).

If any of the following messages appear, power off the PRO MATE II, realign the socket module (see Section 2.4.1), then turn it back on:

- Tighten Socket Top

- Select Socket

- Socket Not Supported

At this point, you are ready to use PRO MATE II. If you are going to use PRO MATE II with MPLAB (Windows), please refer to Chapter 4. If you are going to use PRO MATE II with PROCMD (DOS), please refer to Chapter 5. If you are going to use PRO MATE II in stand-alone mode, please refer to Chapter 6.

2.8   Configuring the Serial Port for PRO MATE II

Once MPLAB is installed on your PC, you run it by one of the following methods:

- **For Windows 3.x/NT 3.51:** Double-click on the MPLAB icon.

- **For Windows 95/98, Windows NT 4.0, or Windows 2000:** From the Start menu, select *Programs > Microchip MPLAB > MPLAB*.

The MPLAB desktop should look like Figure 2.4.
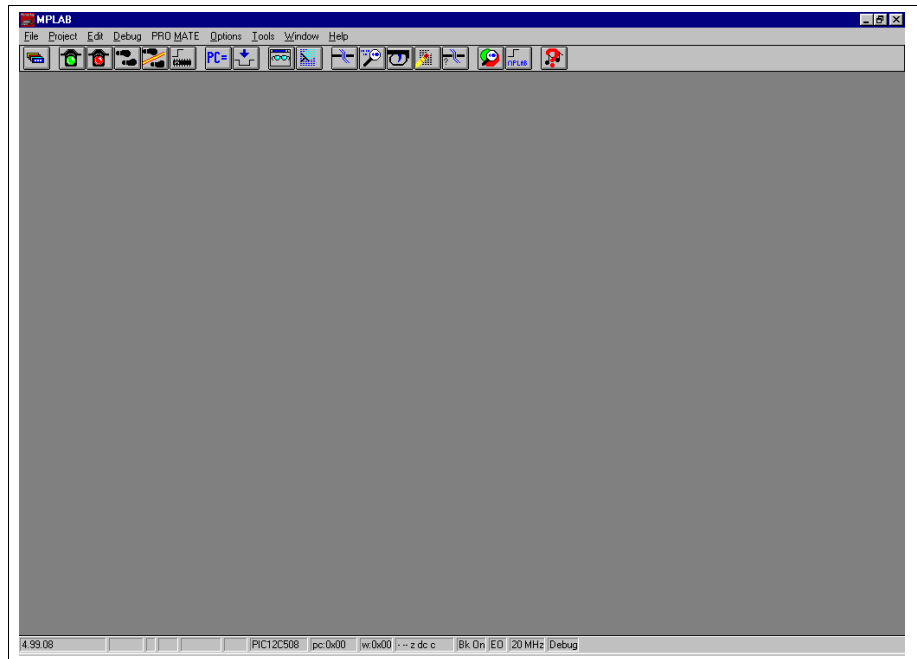
Figure 2.4: MPLAB Desktop

From the MPLAB Options menu, select *Options > Programmer Options > Communications Port Setup*. A dialog similar to the one shown in Figure 2.5 will appear.
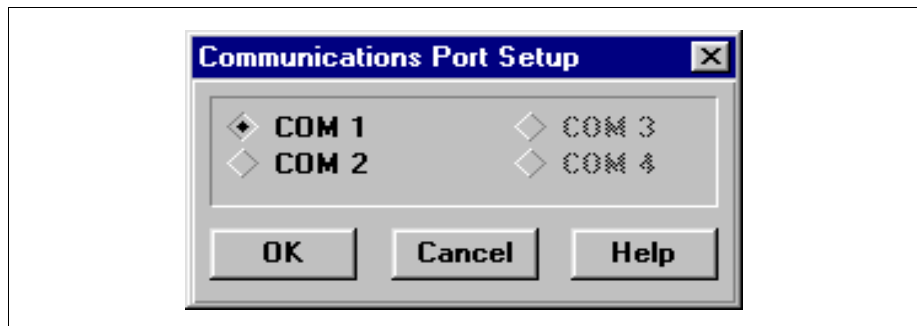
Figure 2.5: Communications Port Setup Dialog

The Communications Port Setup Dialog shows the possible PC serial communication ports. **OK** sets the options and tries to contact PRO MATE II to verify the port. **Cancel** will ignore the changes and close the dialog.

If PRO MATE II is not found on the selected COM port, the dialog of Figure 2.6 appears.

Figure 2.6: Communications Error Dialog

If you cannot establish communications between the PC and PRO MATE II, please make sure you have installed the hardware and software correctly. If you still cannot establish communications between the PC and PRO MATE II, refer to Appendix B.

2.9 Selecting PRO MATE II as the Programmer

In addition to PRO MATE II, the PICSTART Plus device programmer is supported under the MPLAB IDE. However, only one programmer can be used at a time. To change programmers, select *Options > Programmer Options > Select Programmer* to display the Select Programmer dialog. Next, select PRO MATE Device Programmer from the pull-down menu, and click **OK** (Figure 2.7). The Changing Selected Programmer dialog appears requesting that "You must restart MPLAB for the new programmer to take effect." Click **Yes** to close the MPLAB program. When you restart the MPLAB IDE, it will be reconfigured for the new programmer and the menu for the new programmer will appear on the menu bar.

Figure 2.7: Select Programmer Dialog

2.10 Setting Up the MPLAB Development Mode

Select *Options > Development Mode* to open the Development Mode dialog (Figure 2.8) and select a processor module or device.

To view complete information on the inherent limitations of the device in the development mode you've selected, click **Details**. Scroll to the device and double-click on it to view the details.

If you want to see which device your emulator is 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.
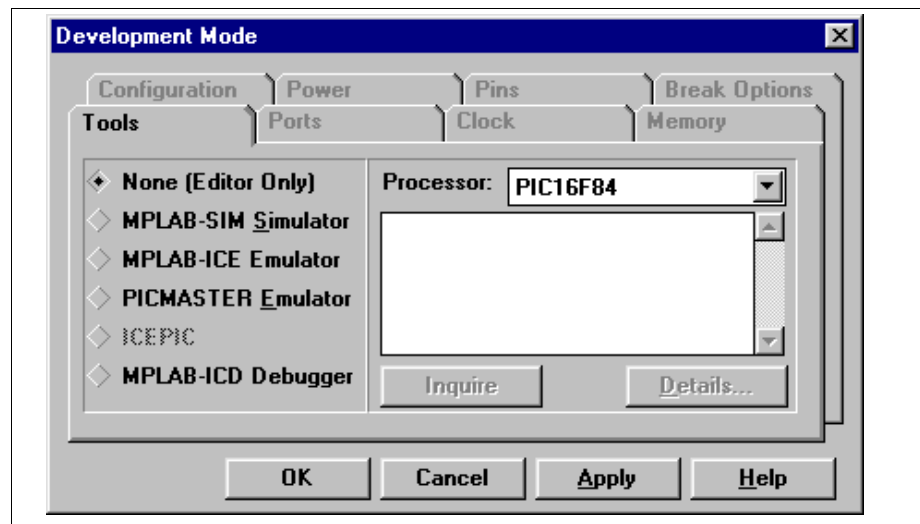
Figure 2.8: Development Mode Dialog

If you are going to use the MPLAB IDE to develop your firmware, you may choose from the following modes:

- None (Editor Only) – to write your source code and then assemble or compile it into a hex file. Select the relevant Processor from the list.

- MPLAB-SIM Simulator – to write your source code, assemble or compile it into a hex file, and then simulate PICmicro program execution via the simulator. Select the Processor from the list.

- Emulator (MPLAB-ICE, PICMASTER, or ICEPIC™) – to write your source code, assemble or compile it into a hex file, and then emulate PICmicro program execution via an emulator. MPLAB will select the processor for the hardware. Click the **Ports** tab to select the I/O port for emulator communication.

- MPLAB-ICD Debugger – to write your source code, assemble or compile it into a hex file, and then debug PICmicro code via a debugger. Select the processor from the list. Click the **Ports** tab to select communication port options.

If you already have hex code that you wish to program into a device, select Editor Only for the development mode and choose your desired processor (device) for programming from the pull-down list. You can click **Apply** to save the changes you have made up to this point.

Refer to the *MPLAB IDE User's Guide* for more information on developing firmware using the MPLAB IDE. When you have finished your development mode selections, click **OK**.

2.11   Starting PRO MATE II

To enable PRO MATE II, select the Programmer menu and click **Enable Programmer** (Figure 2.9). The PRO MATE Device Programmer dialog and the Configuration Bits dialog will appear when the programmer is enabled. If these dialogs appear, refer to the next chapter for PRO MATE II programming examples. Or, read Chapter 4 for information on using the PRO MATE II programmer.
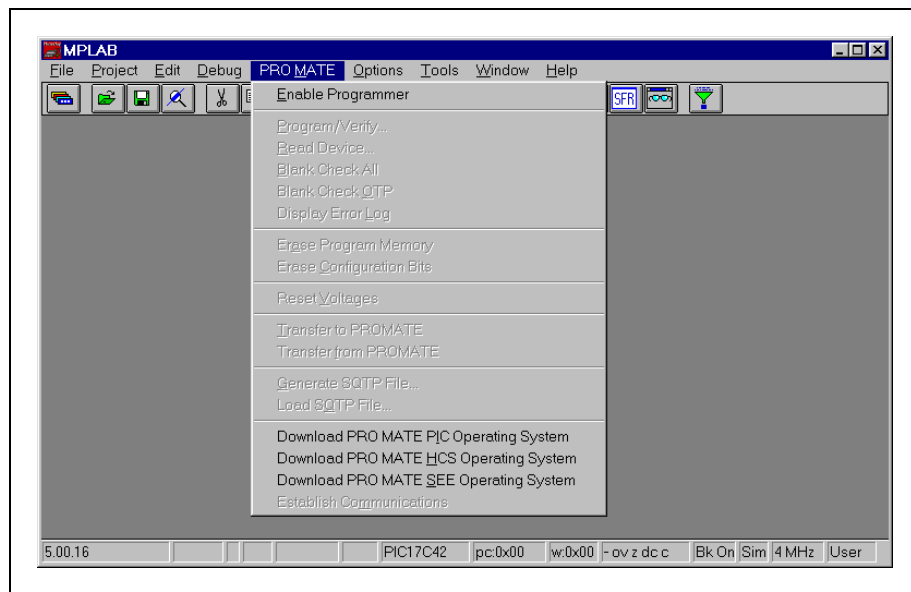
Figure 2.9: PRO MATE II Programmer Pull-Down Menu

If you have been using another programmer (e.g., PICSTART Plus), the PRO MATE menu might not be available. From the Options menu, select *Options > Programmer Options > Select Programmer* to bring up the Select Programmer dialog, and choose PRO MATE from the list. After you select the programmer, the MPLAB IDE will display a message and shut itself down. You must restart the MPLAB IDE before the programmer options are available. The PRO MATE menu will appear on the menu bar when you restart the MPLAB IDE.

Once the programmer is enabled, the first menu item will change to **Disable Programmer** and can be used to disable PRO MATE II.

If the device you selected when setting up the development mode in MPLAB is not supported by the PRO MATE II operating system, a message box will pop up stating this when you try to enable the programmer.

Where possible, new device support on the old operating system is allowed. However, some new devices require updates to the PRO MATE II operating system. Instructions on upgrading the operating system are provided in Section 7.12.

2.11.1   PRO MATE II Dialogs

The PRO MATE Device Programmer dialog and the Configuration Bits dialog are displayed whenever the programmer is enabled. Closing the PRO MATE Device Programmer dialog will disable the programmer. The options on the screen will show the current values. If the option is unavailable, the item appears in gray (not black) text. The bits listed in the Configuration Bits dialog depend on the device you have selected.

Section 4.5 will discuss how these dialogs are used to program devices.

PRO MATE II User's Guide

NOTES:

Chapter 3. Programming Examples

3.1   Introduction

The tutorials in this chapter lead you through the steps involved in programming the following PICmicro devices:

- PIC16F84
- PIC12C508A
- 24AA02
- 24AA65

3.2   Example: Programming a Mid-Range PICmicro Device

The PRO MATE II supports many of Microchip's Memory devices, including the Smart Serial^{TM} EEPROMs.

This example leads you through the steps involved in programming the PIC16F84, one of Microchip's most popular PICmicro devices.

3.2.1   Programming Overview – Mid-Range PICmicro Devices

Programming a mid-range PICmicro device involves the following steps:

- Configuring the Serial Port for PRO MATE II
- Setting up the MPLAB Development Mode
- Creating the MPLAB Project and Assembly Code
- Building the Hex File
- Enabling PRO MATE II
- Setting up the PRO MATE Device Programmer Dialog
- Setting the Configuration Bits
- Programming the Device
- Verifying the Programming

3.2.2   Before You Begin

Before you can begin this tutorial, you must install the PRO MATE II hardware (Section 2.4) and MPLAB IDE software (Section 2.5). Make sure you have read and completed the instructions in Section 2.8 through Section 2.11.

PRO MATE II User's Guide

Make sure that your PC and PRO MATE are communicating and the PRO MATE menu item appears on the MPLAB menu before you begin this tutorial.

3.2.3 Setting Up the MPLAB Development Mode

Select *Options > Development Mode* to open the Development Mode dialog (Figure 3.1) and select a processor module or device.

Under Tools, select None (Editor Only). Select the PIC16F84 device and click **OK**.
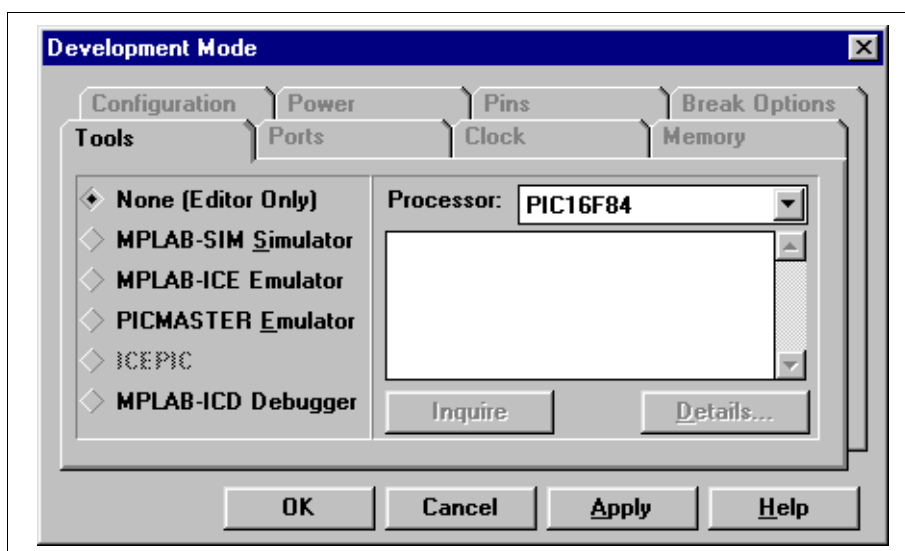
Figure 3.1: Setting the Development Mode

3.2.4 Creating the MPLAB Project and Assembly Code

In order to program the device, you'll need a hex file. In this example, you'll create your own source code and compile it into a hex file using MPLAB Projects.

3.2.4.1 Creating the Project

Select *Project > New Project*. Name the new project ex16f84 and select the drive and directory you want to store it in. In this example, use the MPLAB directory. Click **OK**.
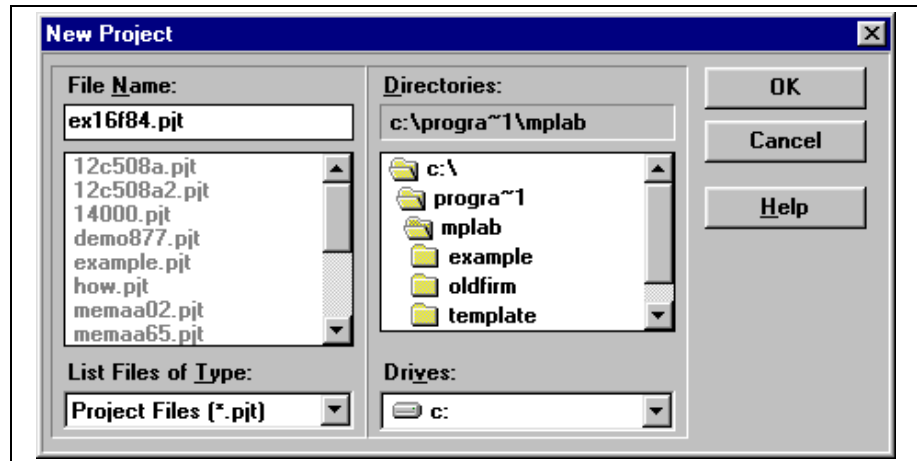
Figure 3.2: Creating a Project

The Edit Project dialog will appear. It should show the Editor Only development mode and the PIC16F84 device. The name of the hex file that this project will create appears in the Project Files area.
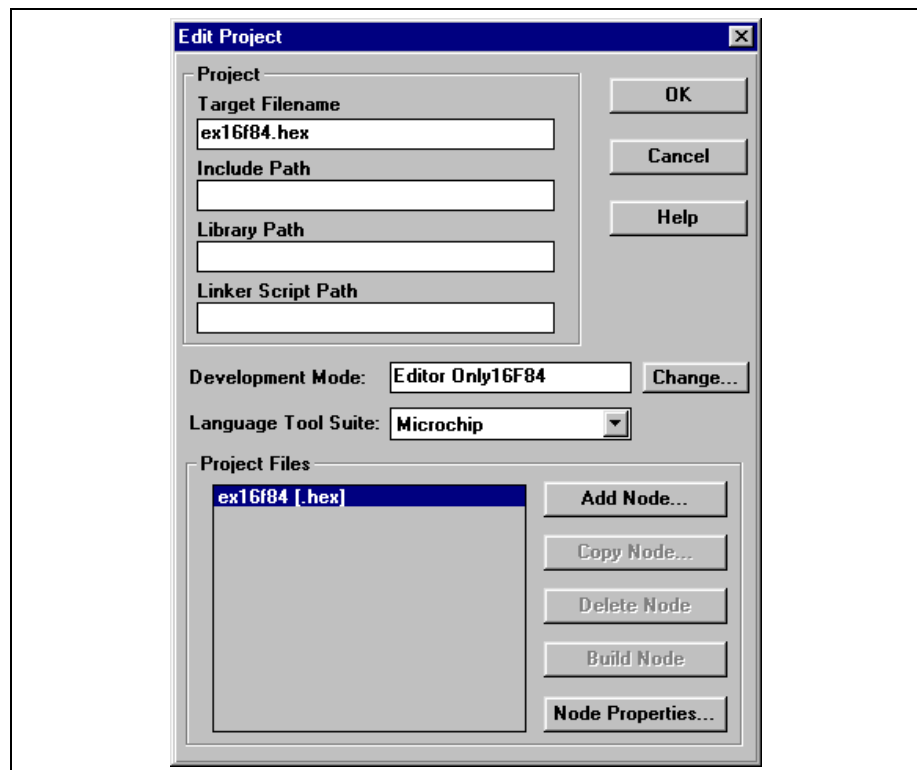
Figure 3.3: Edit Project Dialog

Select the hex file ex16f84.hex and click **Add Node**.

PRO MATE II User's Guide
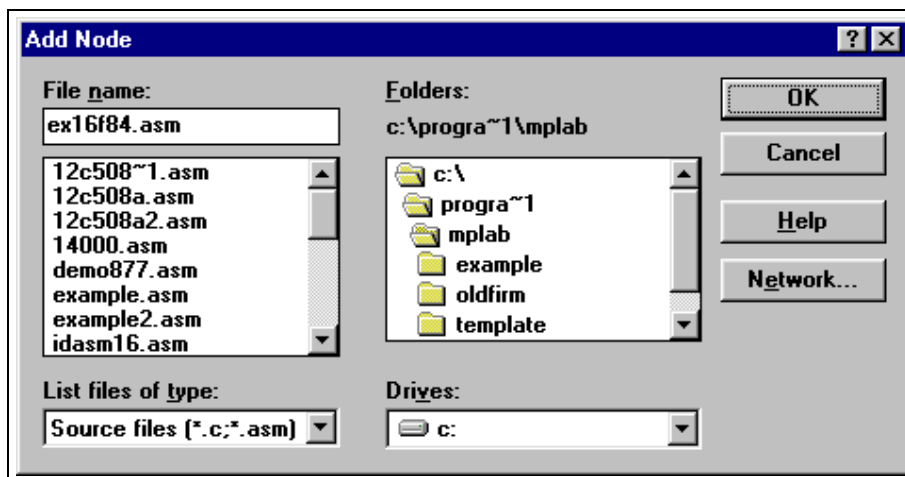
Figure 3.4: Adding the Assembly File

Type `ex16f84.asm` in the File name box and click **OK**. The file
`ex16f84.asm` appears below `ex16f84.hex` in the Project Files list in the
Edit Project dialog.
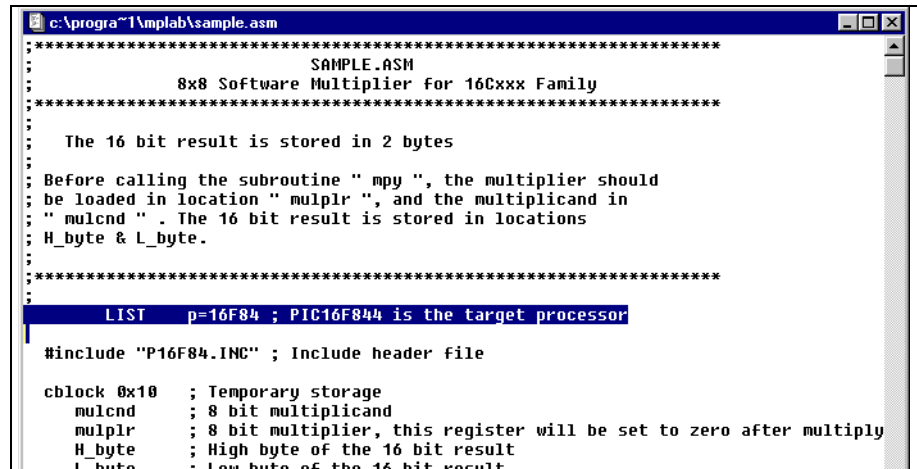
Click **OK** to close the Edit Project dialog. Select *Project > Save Project* to
save your project.

3.2.4.2   Creating the Assembly Code

At this point, you still need to create the source code that you will assemble to
create your hex file. Select *File > New*. An Untitled source code window will
open on your MPLAB desktop.

For this example, you will use the same source code that is in the file
`sample.asm`. Select *File > Open* and open `sample.asm`. It is included with
the standard MPLAB installation and is located in the MPLAB directory.

Figure 3.5: Copying Source Code from `sample.asm`

Use the MPLAB Editor to copy the source code in the `sample.asm` window. Then, select the Untitled source code window and paste the code into it.

When you are finished placing the source code in the Untitled window, save your file. Select *File > Save As* and specify the name `ex16f84.asm`.
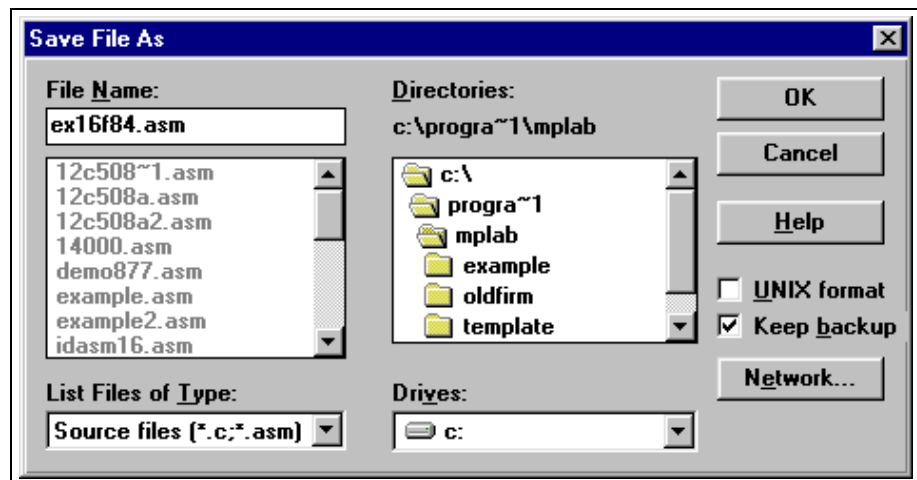
Figure 3.6: Saving Your Source File

3.2.5 Building the Project's Hex Code

Now, build the project and create your hex file. Select *Project > Build All*.
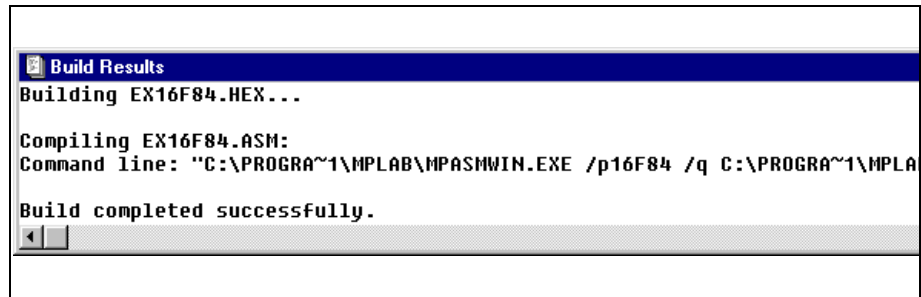
Figure 3.7: Build Results Window

3.2.6 Starting PRO MATE II

To enable PRO MATE II, select *PRO MATE > Enable Programmer* from the MPLAB menu. The PRO MATE Device Programmer dialog and Configuration Bits dialog will appear when the programmer is enabled. The Program Memory window will display the hex file you built. If the Program Memory window did not open, select *Window > Program Memory* to open it. You can resize or move the Program Memory window on your display. You may wish to close the Build Results window.
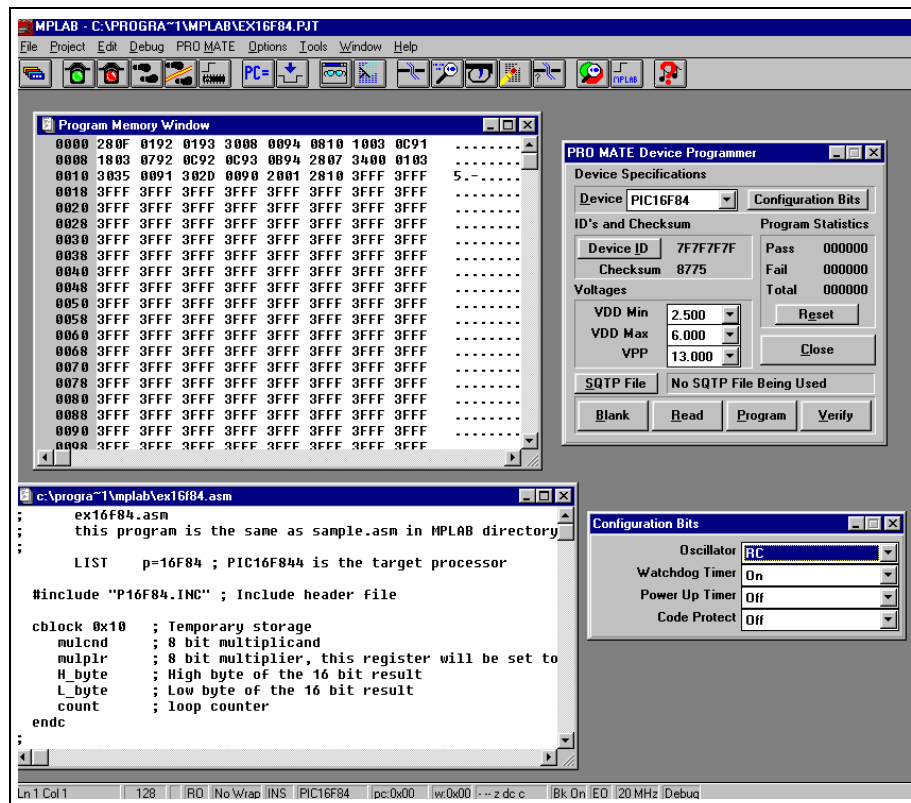
Figure 3.8: Desktop with PRO MATE II Enabled

3.2.6.1   The PRO MATE Device Programmer Dialog

The PRO MATE Device Programmer dialog must remain open during your programming session.
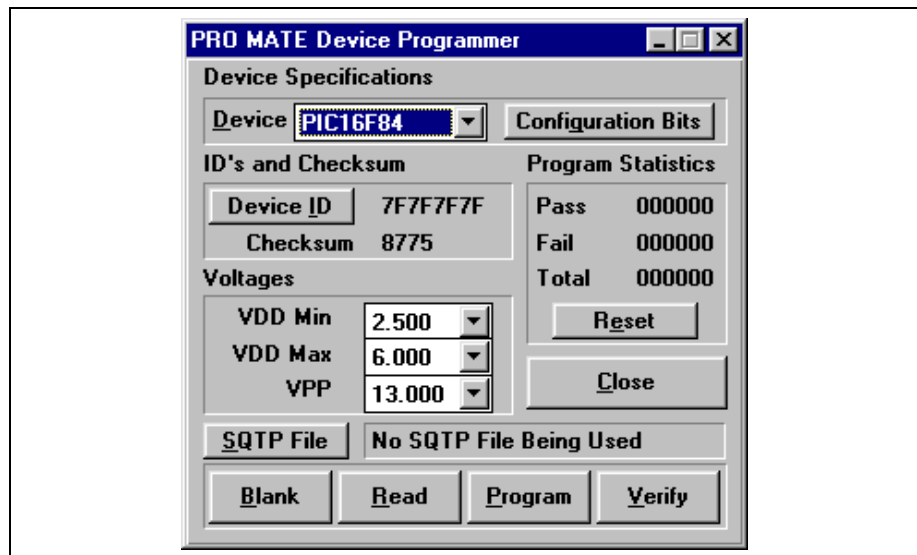
Figure 3.9: Device Programmer Dialog - PIC16F84

The Device ID is useful for tracking the version of firmware on a device once it is code protected. The calibration memory tutorial incorporates instructions on using this feature (see Section 3.3).

Although you can adjust voltages on this dialog, you rarely need to.

You can use the **Read**, **Blank** Check, **Program**, and **Verify** buttons to perform those functions on an entire device.

3.2.6.2   Configuration Bits Dialog

The Configuration Bits dialog opens when you enable the programmer. If you close the Configuration Bits dialog, you can click **Configuration Bits** in the PRO MATE Device Programmer dialog to reopen it.
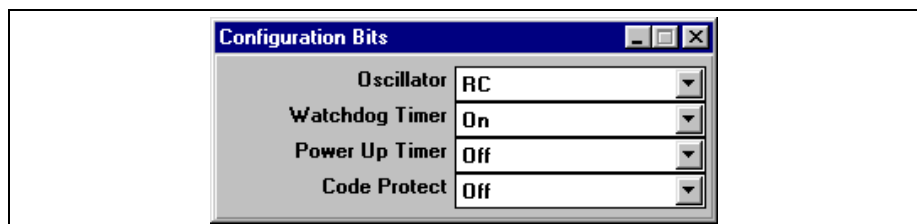
Figure 3.10: Setting Configuration Bits

3.2.6.3 The Program Memory Window

The MPLAB Program Memory window displays the hex code that you will program onto your PIC16F84 device. If the Program Memory window is not open, select *Window > Program Memory* to open it. If your Program Memory window does not look similar to Figure 3.11, click the system icon (to the left of the window's title) and select Hex Code Display. The window displays the hex values and the ASCII text from the source code.
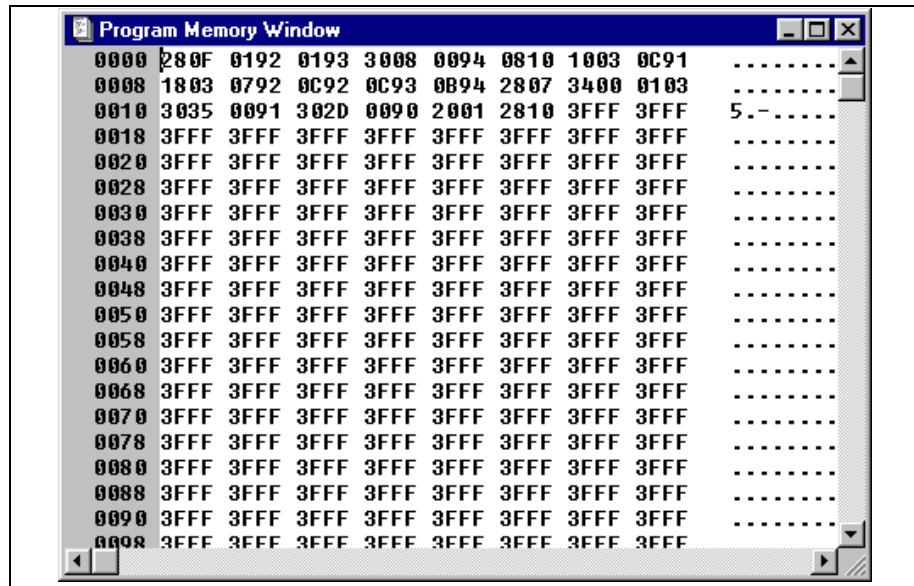
Figure 3.11: PIC16F84 Code in Program Memory

3.2.7 Programming the Device

Now that you have your data in program memory, you can program the device. Click **Program** in the PRO MATE Device Programmer dialog. A window will indicate the progress, and will indicate success or failure when finished.

Figure 3.12: Programming the PIC16F84

3.2.8 Verifying the Programming

Click **Verify** to double-check the programming in the device. If any address locations on the device do not match program memory, an error log will display the discrepancies. If the bad data in the error log is 0000, try reseating the socket module.

3.3   Example: Programming Calibration Devices

The two-part tutorial in this section leads you through the steps involved in programming devices that have been supplied by the factory with calibration memory. Programming devices with calibration memory include:

- PIC12C508/508A
- PIC12C509/509A
- PIC12CE518
- PIC12CE519
- PIC12C671
- PIC12C672
- PIC12EC673
- PIC12EC674
- PIC14000
- PIC16C505

This two-part example provides step-by-step instructions for initially programming and reprogramming a windowed PIC12C508A calibration memory device.

3.3.1   Calibration Programming Process Overview

The steps required to initially program a new, unprogrammed calibration memory device are:

- Configuring the Serial Port for PRO MATE II
- Setting Up the MPLAB Development Mode
- Enabling PRO MATE II
- Setup for Programming a Device
  - Storing Calibration Data to a File
  - Checking the Device and Setting the Device ID or Checksum
  - Checking the Configuration Bit Settings
  - Loading the Hex File to be Programmed
- Programming the Device
- Verifying the Programming

The steps required to reprogram an erased windowed calibration memory device are:

- Configuring the Serial Port for PRO MATE II
- Setting Up the MPLAB Development Mode
- Enabling PRO MATE II

- Setup for Programming a Device

    - Checking the Device and Setting the Device ID or Checksum
    - Checking the Configuration Bit Settings
    - Loading the Hex File to be Programmed
    - Restoring the Calibration Data from a File

- Re-Programming the Device

- Verifying the Programming

3.3.2   Before You Begin

Before you can begin this tutorial, you must install the PRO MATE II hardware (Section 2.4) and MPLAB IDE software (Section 2.5). Make sure you have read and completed the instructions in Section 2.8 through Section 2.11. Make sure that your PC and PRO MATE II are communicating and the PRO MATE II menu item appears on the MPLAB menu before you begin this tutorial.

3.3.3   PIC12C508A Example Part 1: Initial Programming of a Windowed Calibration Memory Device

3.3.3.1   Setting Up the MPLAB Development Mode

Select *Options > Development Mode* to open the Development Mode dialog (Figure 3.13).
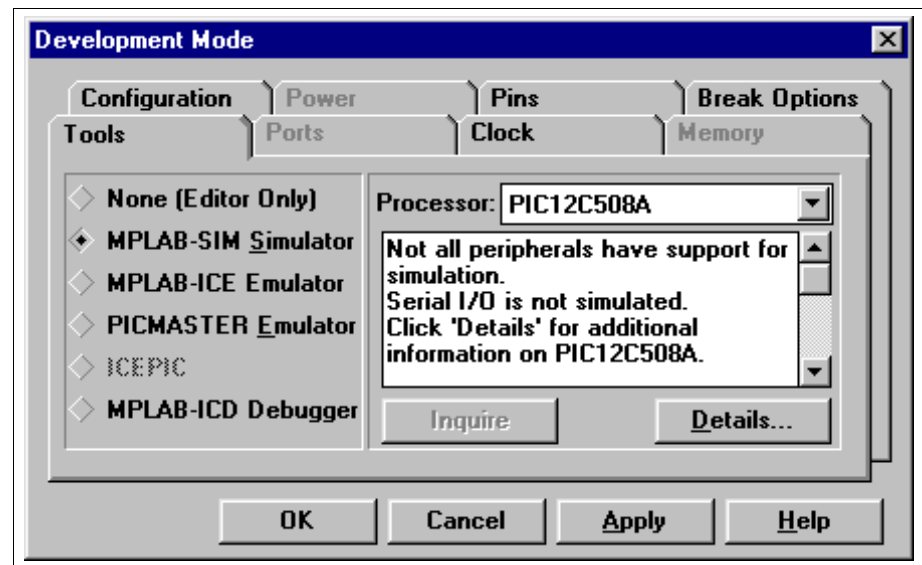
Figure 3.13: Development Mode Dialog

PRO MATE II User's Guide

Select the MPLAB-SIM Simulator development mode. Select the
PIC12C508A processor. Click **OK**.

> **Note:** When working with calibration memory, it is important that the
> MPLAB IDE *not* be in Emulator mode. This ensures that the cali-
> bration data is correctly loaded from the device or file rather than
> from the emulator probe.

3.3.3.2   Changing the Environment Settings

A default setting in the MPLAB IDE clears all memory on download. You don't
want to do that when working with a device with calibration memory, because
you want to preserve that memory in the device. Select *Options >
Environment Setup*. Locate the Global Switches area in the **General** tab of
the Development Mode dialog. Remove the check mark in the Clear Memory
on Download check box. Click **OK**.

3.3.3.3   Starting PRO MATE II

To enable PRO MATE II, select the PRO MATE menu and click **Enable
Programmer** (Figure 3.14).
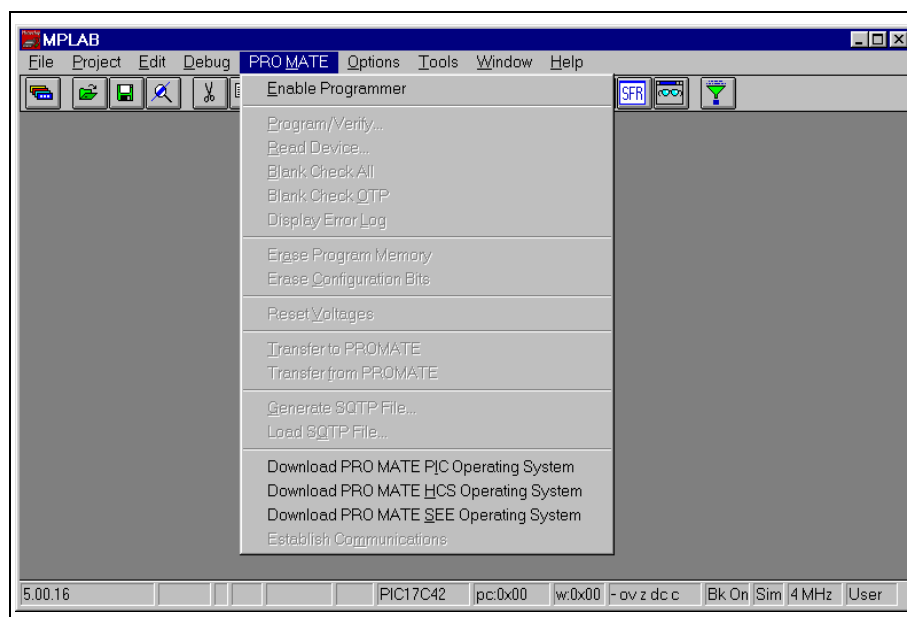
Figure 3.14: PRO MATE II Programmer Pull-Down Menu

The PRO MATE Device Programmer dialog and the Configuration Bits dialog
are displayed whenever the programmer is enabled.

3.3.3.4 Storing Calibration Data

Calibration parts such as the PIC12C508A are calibrated at the factory with the calibration parameters stored in a special section of program memory. When you erase a windowed device in order to reprogram it, you also erase its calibration data. In order for the device to operate properly once it is re-programmed, you must restore that individual device's calibration data. The device might not operate properly if you use another individual device's calibration data.

Before you program a windowed device for the first time, store its calibration data in a hex file on your PC.

This tutorial will use a PIC12C508A device. Number it with a small label. For this tutorial, label the part A.

1. Before you place your device in the PRO MATE II socket, select *PRO MATE > Erase Program Memory*. This sets all bits in the Program Memory, Data Memory, and Calibration Memory windows to 1.
2. Insert the part that you labeled A in the PRO MATE II.
3. Click **Read** in the PRO MATE Device Programmer dialog to read the device. If the socket module is not tightly fastened to the PRO MATE II, a message will indicate that the device is code-protected. Tighten the socket module and attempt to read the device again. When the code protection warning no longer occurs and PRO MATE II reads the device, continue.
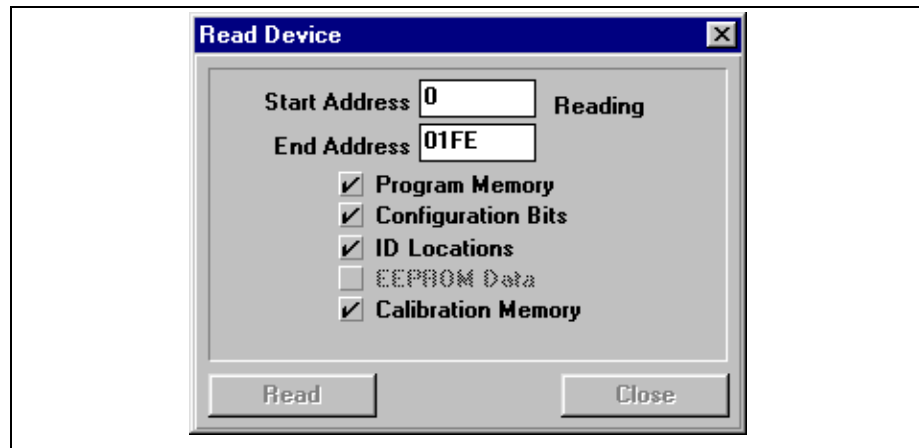
Figure 3.15: Reading the Device

4. Select *Window > Calibration Data* to see the calibration information that was uploaded from the device to the MPLAB IDE. Click **OK**.
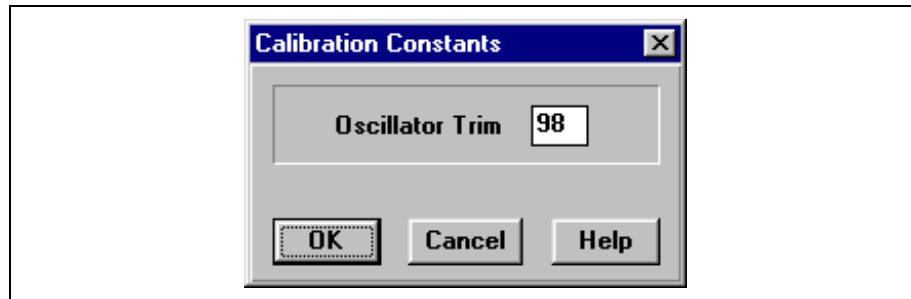
Figure 3.16: Calibration Data

5. Now that the device's calibration data is in memory, make a note of the setting and save it to a file for later use. Select *File > Export > Export Memory*. Check only the checkbox labeled Calibration memory. Clear the Program memory, Configuration bits, and IDs check boxes. Name the file `calib-a.hex` so that you will be able to match the file to the part that is now in the socket module. Click **OK**.
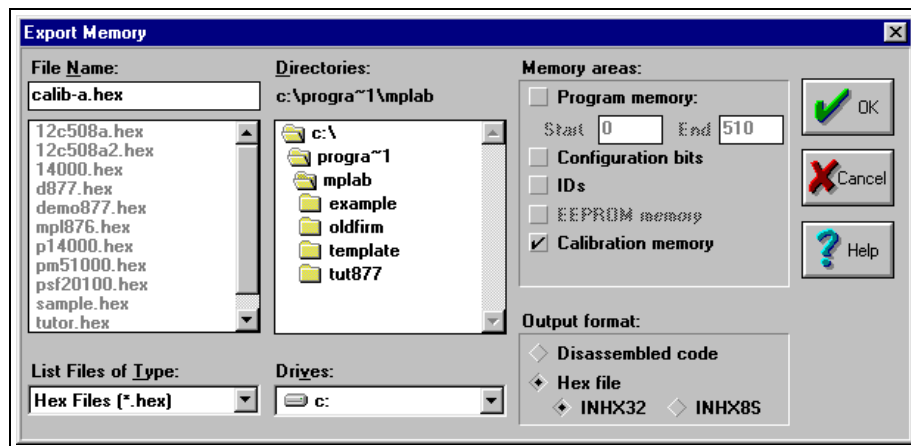
Figure 3.17: Saving Calibration Data

If you had several new windowed parts that you wanted to program, you would repeat this procedure for each one, being careful to name each calibration hex file after the correct part.

3.3.3.5 Setting Up the Device Programmer Dialog

The PRO MATE Device Programmer dialog (Figure 3.18) is always open when PRO MATE II is enabled. Closing this dialog will disable the programmer.
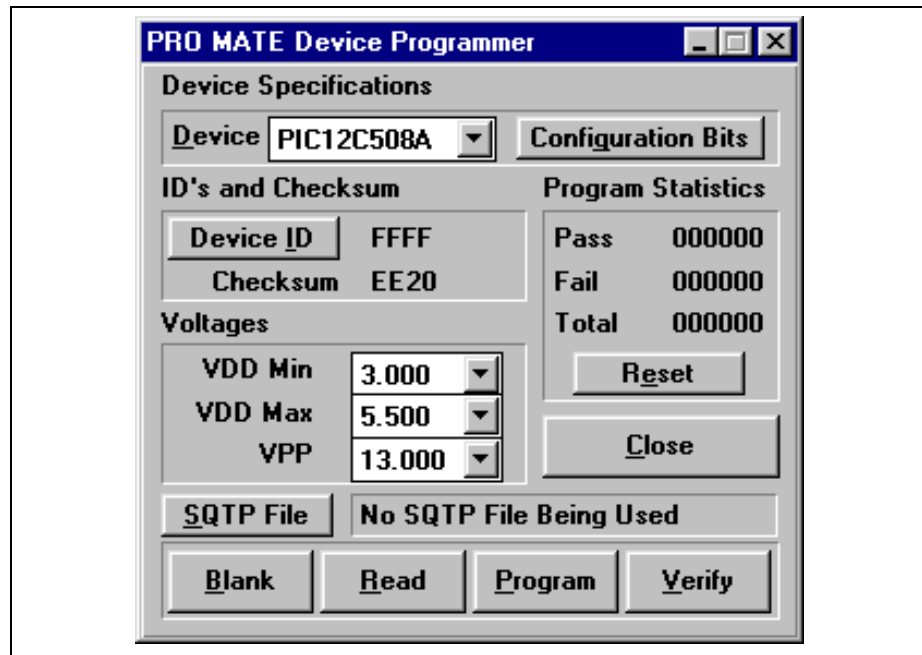
Figure 3.18: PRO MATE Device Programmer Dialog

The Device box shows the PIC12C508A device that you selected when you set up the development mode.

Click **Device ID**. Enter 0001. This will allow you to identify the version of firmware on the device after it is programmed and code-protected. Click **OK**.

Figure 3.19: Using Device ID to Identify Firmware or Other Information

3.3.3.6 Setting Up the Configuration Bits Dialog

The Configuration Bits dialog (Figure 3.20) opens when PRO MATE II is enabled. If you closed the Configuration Bits dialog, reopen it by clicking **Configuration Bits** in the PRO MATE Device Programmer dialog.

Figure 3.20: Configuration Bits Dialog – PIC16F84

Use the default values for the configuration bits. Make sure Code Protect is Off.

3.3.3.7 Loading the Hex File to be Programmed

Now that you have saved the calibration data to a hex file, load the file to be programmed. Select *File > Import > Import to Memory* to load the hex code into the MPLAB Program Memory window. Select `12c508a.hex`. It is included in the MPLAB installation. Click **OK**.
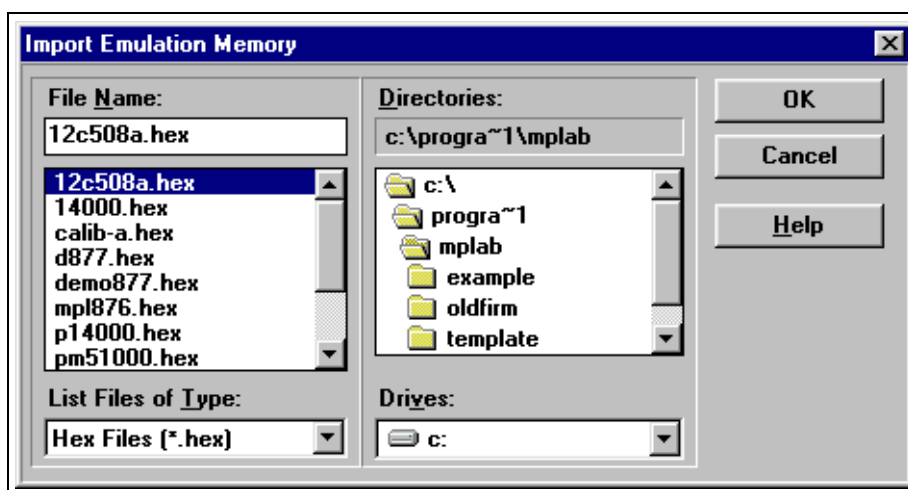
Figure 3.21: Download Emulation Memory Dialog

The Program Memory window should now contain the hex code from the hex file (Figure 3.22). If the Program Memory window is not open, select *Window > Program Memory* to open it.

Figure 3.22: Program Memory — Hex Code Display

3.3.3.8   Programming a Calibration Device

Click **Program** in the PRO MATE Device Programmer dialog to program the entire device (all of program memory, configuration bits, etc.).

Figure 3.23: Program/Verify Dialog

PRO MATE II User's Guide

The Program/Verify dialog will appear while the device is being programmed. The areas being programmed are automatically selected based on the device.

When the programming is finished, "Success" or "Failure" will appear to the right of the Start Address. If the programming failed, an error window will appear showing the good (expected) data and the bad (actual) data for each address it attempted to program. If the bad data shows 0000, try reseating the socket module. Then repeat the steps in Section 3.3.3.4 and return to this section.

3.3.3.9   Verifying the Programming

Now that you have programmed a device, click **Verify** in the PRO MATE Device Programmer dialog to verify that the contents of the device match the values shown in the Program Memory window and PRO MATE Device Programmer dialog.
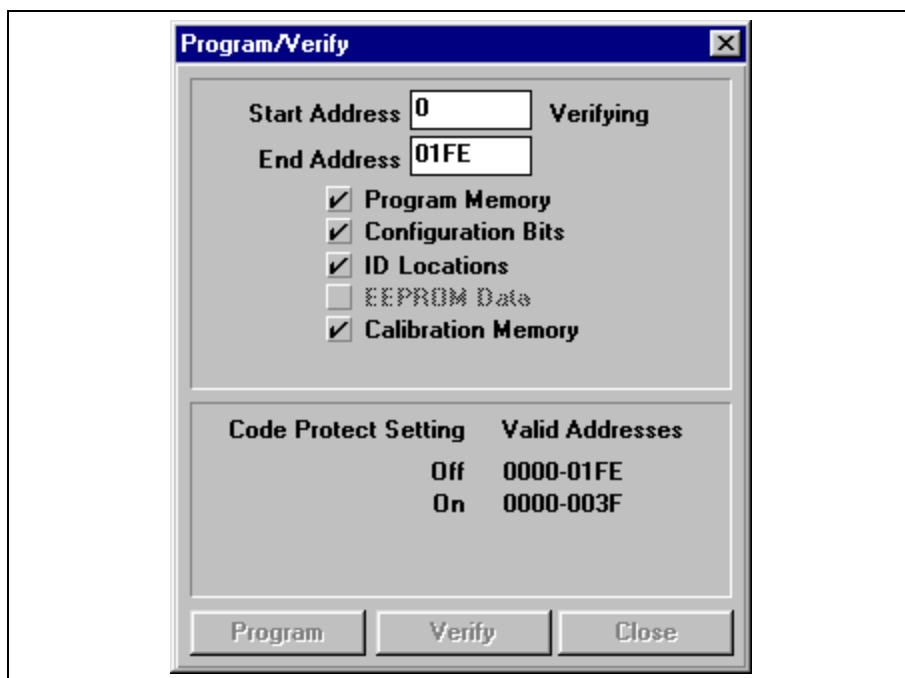
Figure 3.24: Verifying the Programming

3.3.4 PIC12C508A Example Part 2: Re-Programming A Windowed Calibration Memory Device

3.3.4.1 Before You Begin

This tutorial is intended to be used after you have completed the tutorial on initially programming a windowed device in Section 3.3.3.

Before you can reprogram the windowed part, you must erase it in a UV eraser. Remove the covering on the device's window. Place the part in a UV eraser. The amount of time required to completely erase a UV erasable device depends on: the wavelength of the light, its intensity, distance from UV source, and the process technology of the device (the size of the memory cells). This erases the entire device, including its calibration memory. In order to work properly, the calibration memory for that individual device must be restored by importing the calibration file you created for the chip earlier.

3.3.4.2 Setting Up the Device Programmer Dialog

When you first programmed the part in our previous tutorial (Section 3.3.3), you established a device ID to keep track of the version of firmware you programmed into it. You'll update that number for this step of the tutorial. Click **Device ID** in the PRO MATE Device Programmer dialog. Enter 0002. This is to indicate that you are about to program version 2 of our code. This will allow you to identify the version of firmware on the device after it is programmed and code-protected.
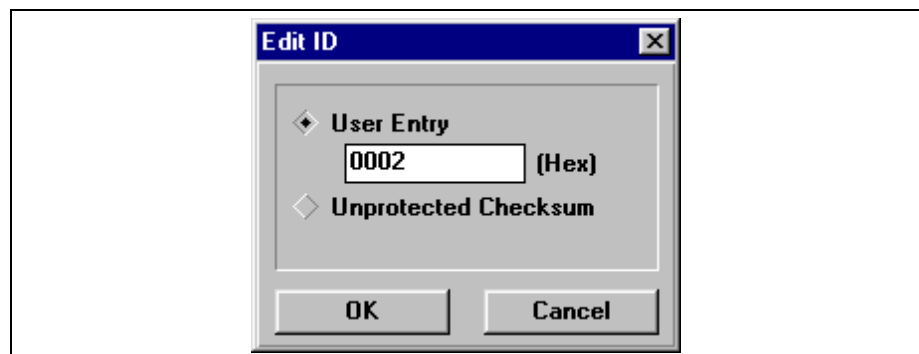
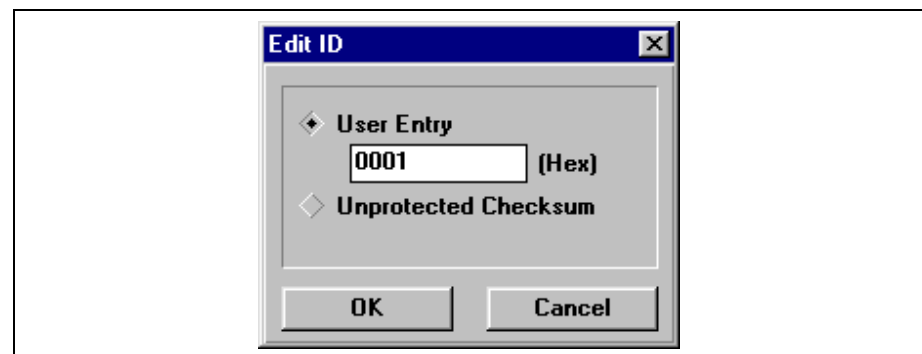**Figure 3.25: Using Device ID to Identify Firmware or Other Information**

3.3.4.3   Setting Up the Configuration Bits Dialog

The Configuration Bits dialog opens when PRO MATE II is enabled. If you closed the Configuration Bits dialog, reopen it by clicking **Configuration Bits** in the PRO MATE Device Programmer dialog.
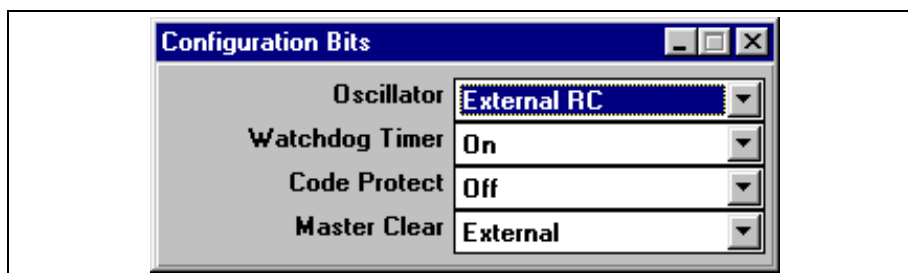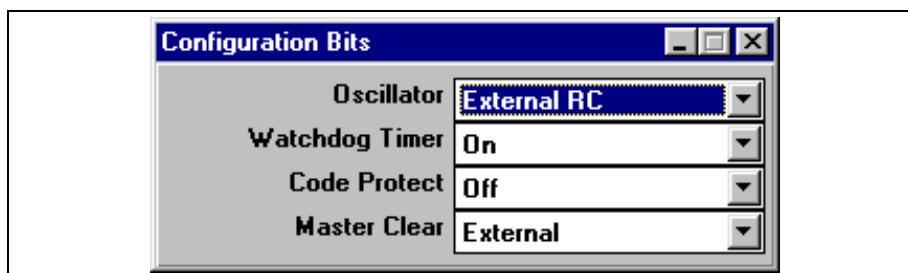

Figure 3.26: Configuration Bits Dialog – PIC16F84

Use the default values for the configuration bits. Make sure Code Protect is Off.

3.3.4.4   Loading the Hex File to be Programmed

Before you load the device's calibration data into program memory, load the file to be programmed.

Select *File > Import > Import to Memory* to load the hex code into the MPLAB Program Memory window. Select `12c508a2.hex`. This is similar to the program you originally used to program the part, but it has been modified slightly for this exercise. Click **OK**.

The Program Memory window should now contain the hex code from the hex file. If the Program Memory window is not open, select *Window > Program Memory* to open it.

```
Program Memory Window                              _ □ ✕
  0000 A05 FFF FFF FFF FFF 066 C00 006    .....f..   ▲
  0008 506 90D 406 90D A08 CFF 028 029    ......()
  0010 030 2E8 A11 2E9 A11 2F0 A11 800    0.......
  0018 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0020 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0028 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0030 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0038 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0040 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0048 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0050 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0058 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0060 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0068 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0070 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0078 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0080 FFF FFF FFF FFF FFF FFF FFF FFF    .........
  0088 FFF FFF FFF FFF FFF FFF FFF FFF    .........   ▼
 ◀ □                                              ▶
```
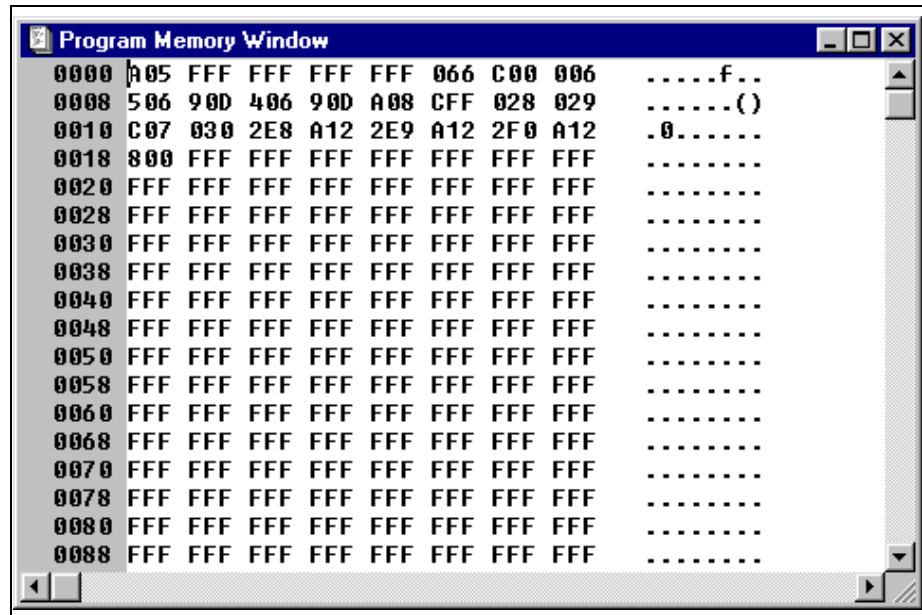
Figure 3.27: Program Memory - Hex Code Display

3.3.4.5 Restoring Calibration Values to the Device

Earlier you saved the part's calibration data to the hex file `calib-a.hex`.
Because the UV eraser erased that data from the device, you must rewrite it
to the device in order for the reprogrammed part to operate correctly.

1. A default setting in the MPLAB IDE clears all memory on download. You
   don't want to do that when working with a device with calibration mem-
   ory, because you want to preserve that memory in the device. Select
   Options > Environment Setup. Locate the Global Switches area in the
   General tab of the Development Mode dialog. Make sure there is *no*
   check mark in the Clear Memory on Download check box. Click **OK**.

2. Select *File > Import > Import to Memory* and select the saved calibration parameters file (`calib-a.hex`).
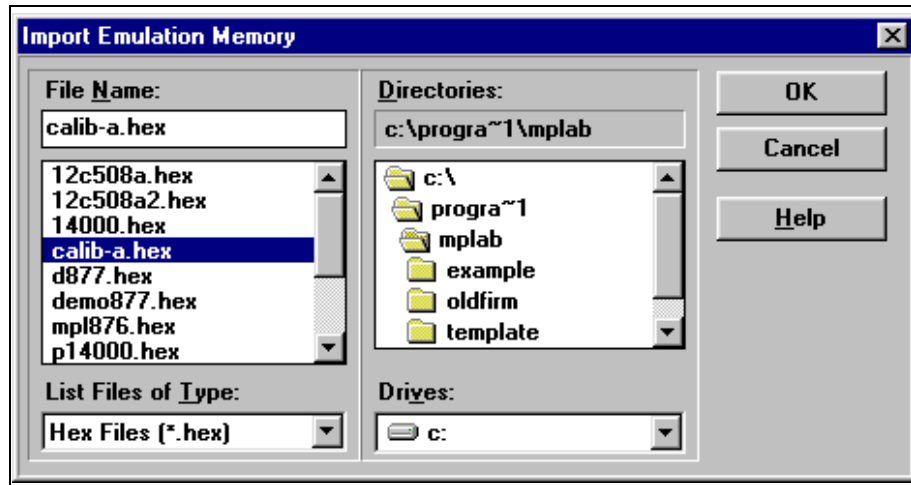
Figure 3.28: Restoring Calibration Data to an Erased Device

3.3.4.6 Programming a Calibration Device

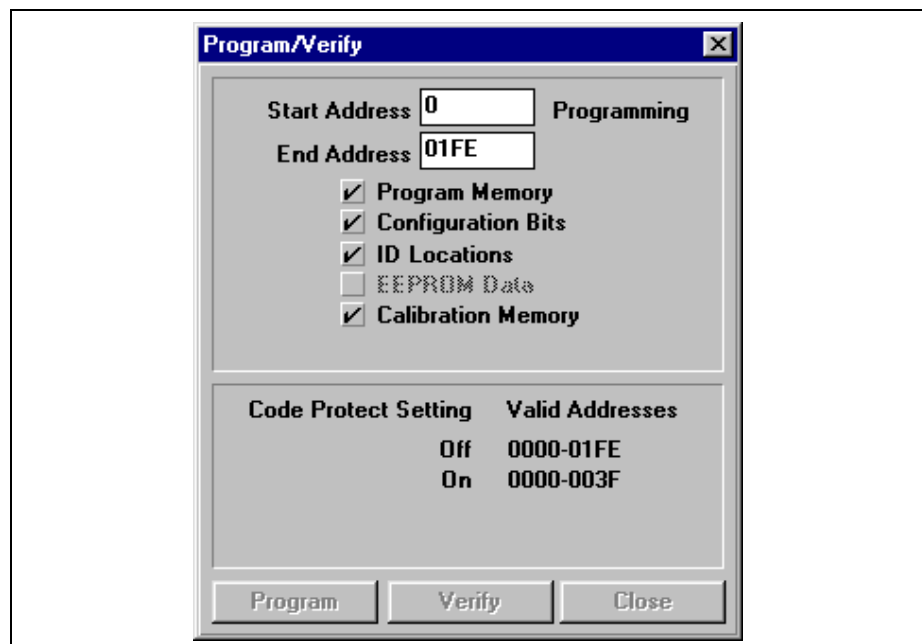Click **Program** in the PRO MATE Device Programmer dialog. This will program the entire device including program memory, calibration memory, configuration bits, etc.

The Program/Verify dialog will appear while the device is being programmed. The areas being programmed are automatically selected based on the device.

When the programming is finished, "Success" or "Failure" will appear to the right of the Start Address.

3.3.4.7 Verifying the Programming

Now that you have programmed a device, click **Verify** in the PRO MATE Device Programmer dialog to verify that the contents of the device match the values shown in the Program Memory window and PRO MATE Device Programmer dialog.

3.4 Example: Programming a Memory Device

PRO MATE II supports many of Microchip's Memory devices, specifically the 24XXX and 93XXX series Serial EEPROMs.

This example leads you through the steps involved in programming a memory device. This example does not discuss programming the device that controls the memory device. For a programming example for a mid-range PICmicro MCU device, such as the PIC16F84, refer to Section 3.2.

3.4.1   Programming Overview – Memory Devices

Programming a memory device involves the following steps:

- Configuring the Serial Port for PRO MATE II
- Setting up the MPLAB Development Mode
- Creating the MPLAB Project and Assembly Code
- Building the Hex File
- Enabling PRO MATE II
- Programming the Device
- Verifying the Programming

3.4.2   Before You Begin

Before you can begin this tutorial, you must install the PRO MATE II hardware (Section 2.4) and MPLAB IDE software (Section 2.5). Make sure you have read and completed the instructions in Section 2.8 through Section 2.11. Make sure that your PC and PRO MATE are communicating and the PRO MATE menu item appears on the MPLAB menu before you begin this tutorial.

3.4.3   Setting Up the MPLAB Development Mode

Select *Options > Development Mode* to open the Development Mode dialog (Figure 3.13) and select a processor module or device.

Under Tools, select None (Editor Only). Select the 24AA02 device and click **OK**.
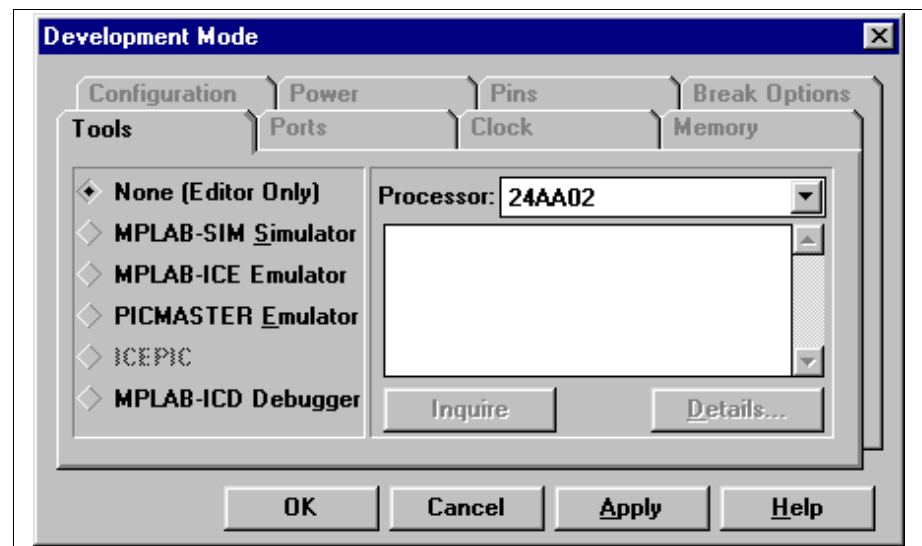
Figure 3.29: Setting the Development Mode

3.4.4   Creating the MPLAB Project and Assembly Code

In order to program the device, you need a hex file. In other examples you simply imported an existing hex file. In this example, you'll create your own source code and compile it into a hex file using MPLAB Projects.

3.4.4.1   Creating the Project

Select *Project > New Project*. Name the new project `memaa02` and select the drive and directory you want to store it in. In this example, use the MPLAB directory. Click **OK**.
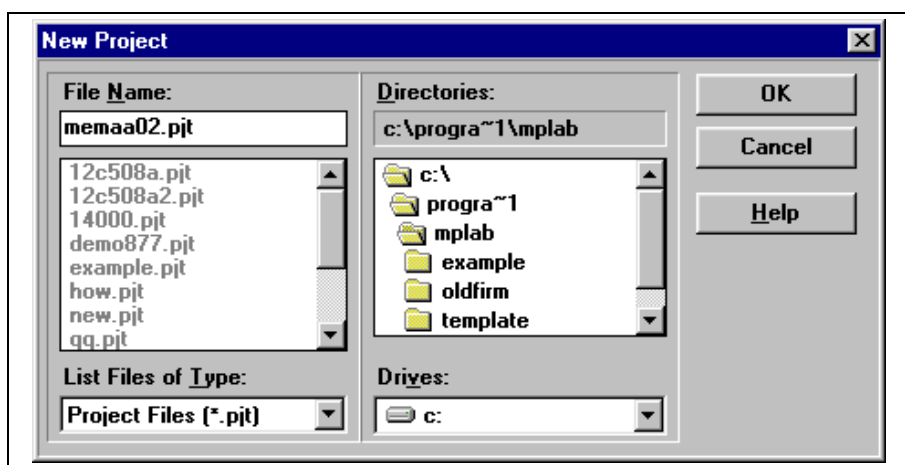
Figure 3.30: Creating a Project

The Edit Project dialog will appear. It should show the Editor Only development mode and the 24AA02 device. The name of the hex file that this project will create appears in the Project Files area.
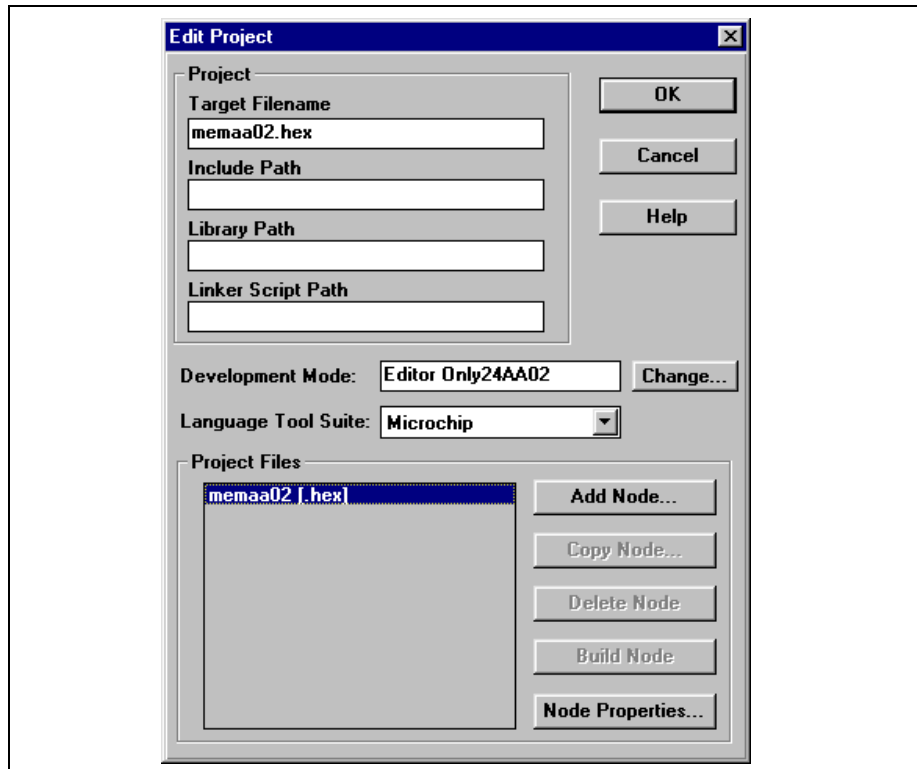
Figure 3.31: Edit Project Dialog

Select the hex file `memaa02.hex` and click **Add Node**.
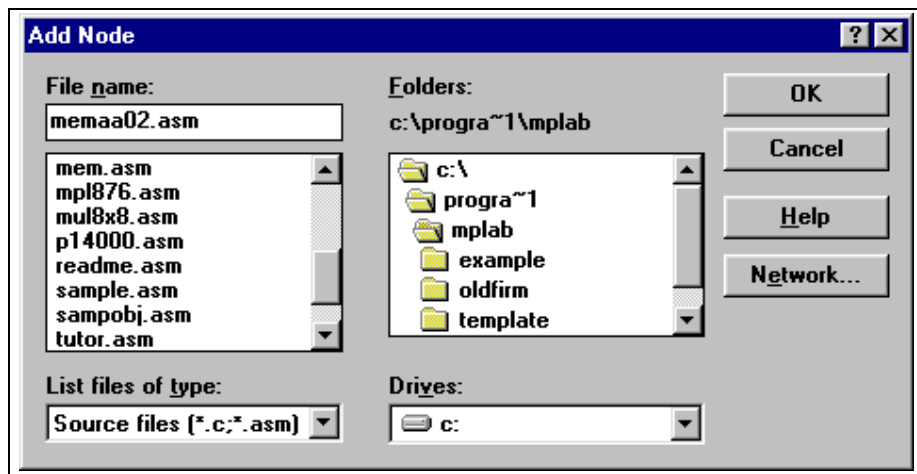
Figure 3.32: Adding the Assembly File

Type `memaa02.asm` in the File name box and click **OK**. The file `memaa02.asm` appears below `memaa02.hex` in the Project Files list in the Edit Project dialog.

Click **OK** to close the Edit Project dialog. Select *Project > Save Project* to save your project.

3.4.4.2   Creating the Assembly Code

At this point, you still need to create the source code that you will assemble to create our hex file. Select *File > New*. An Untitled source code window will open on your MPLAB desktop. Type the following lines (including indentation) in the window:

```
list p=eeprom8
#include "memory.inc"
list m=_24AA02
data 1, 2, 3, 4, 5
data "My Program v1.0", 0
end
```

When you are finished, save your file. Select *File > Save As* and specify the name memaa02.asm.
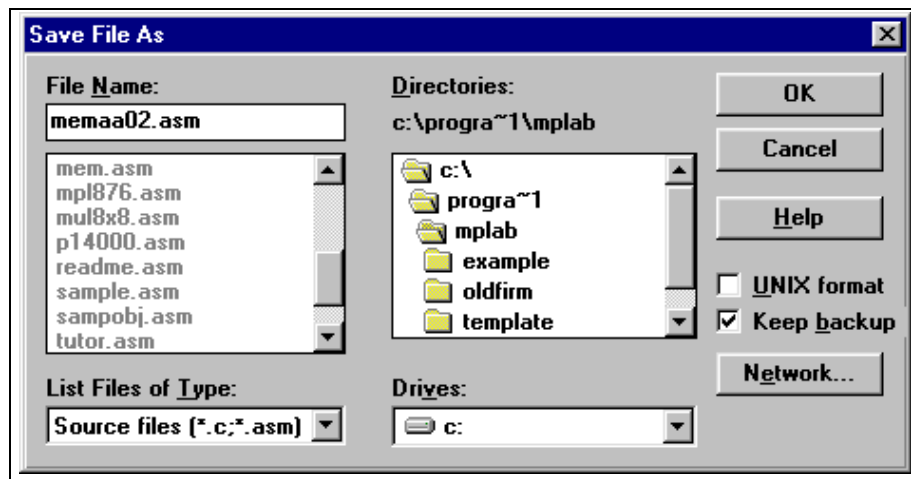
Figure 3.33: Saving Your Source File

3.4.5   Building the Project's Hex Code

Now, build the project and create your hex file. Select *Project > Build All*. If the build did not complete successfully, double-click on the line of the Build Results window that contains the error. Correct the error, save the .asm file, and try your build again.
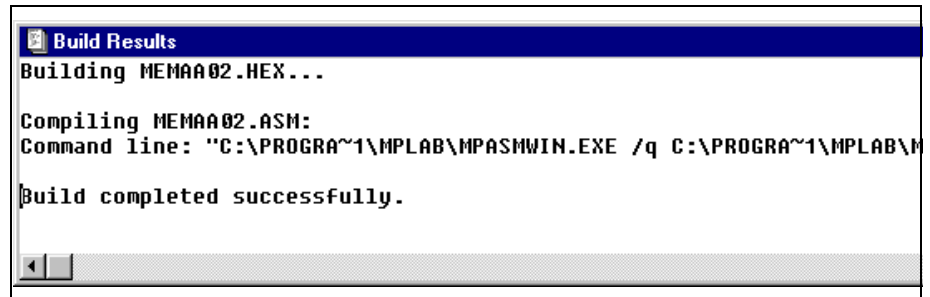
Figure 3.34: Build Results Window

3.4.6   Starting PRO MATE II

To enable PRO MATE II, select *PRO MATE > Enable Programmer* from the MPLAB menu. The PRO MATE Device Programmer dialog will appear when the programmer is enabled. The Program Memory will also open, and will display the hex file you built. If the Program Memory window did not open, select *Window > Program Memory* to open it. You can resize or move the Program Memory window on your display.
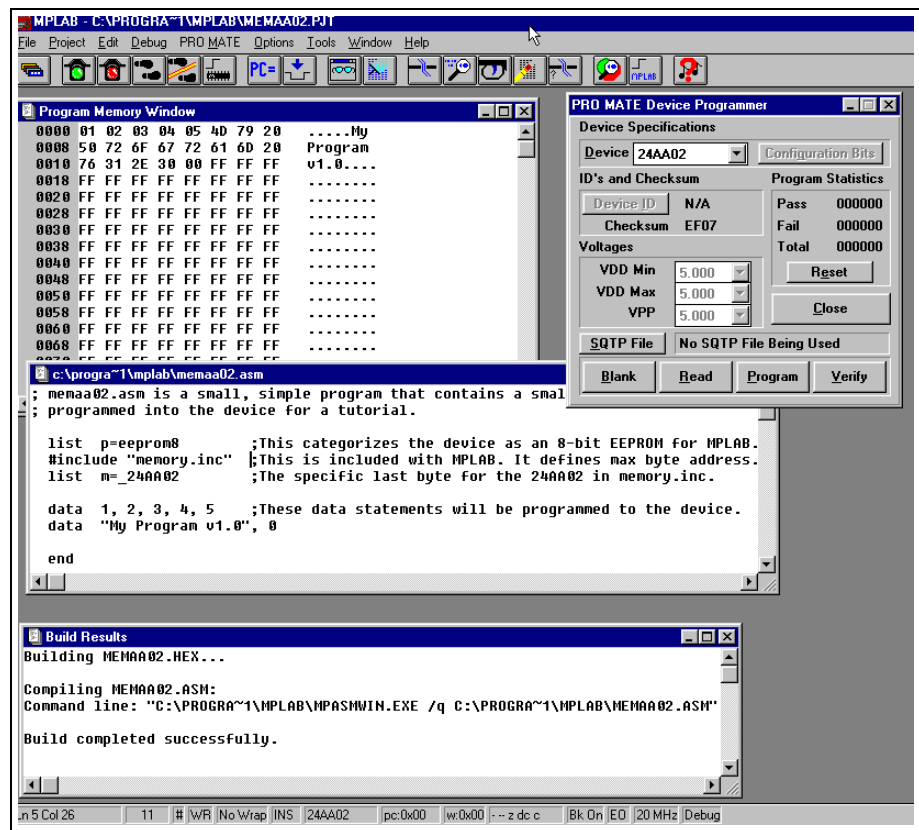
Figure 3.35: Desktop with PRO MATE II Enabled

3.4.6.1   The PRO MATE Device Programmer Dialog

The PRO MATE Device Programmer dialog must remain open during your programming session. Several of the selections in the dialog are disabled. Memory devices are programmed and verified at 5 volts VDD.

Figure 3.36:   Device Programmer Dialog - 24AA02

3.4.6.2   The Program Memory Window

The MPLAB Program Memory window displays the hex code that you will program onto your 24AA02 device. If the Program Memory window is not open, select *Window > Program Memory* to open it. If your Program Memory window does not look similar to Figure 3.37, click the system icon (to the left of the window's title) and select Hex Code Display. The window displays the hex values and the ASCII text from the source code.
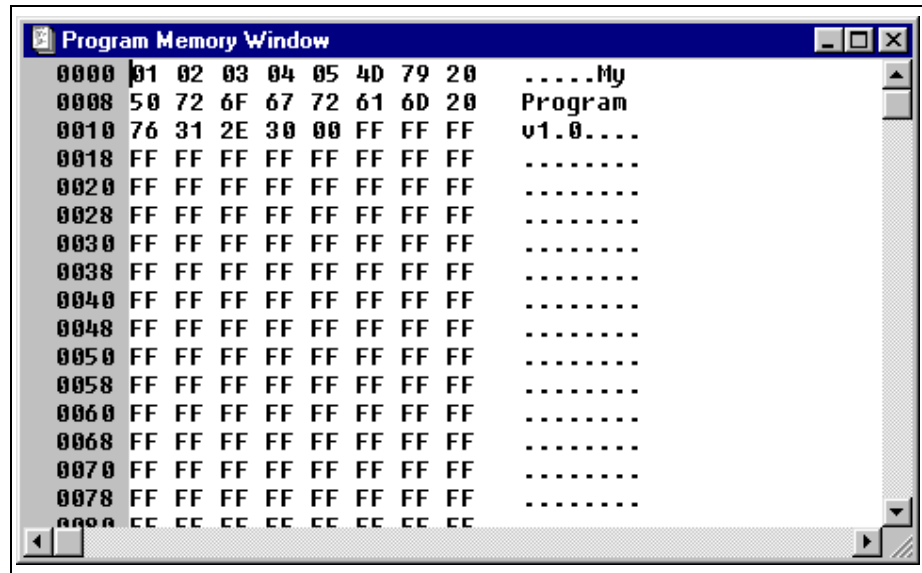
Figure 3.37: Memaa02 in Program Memory

The data displayed in the Program Memory window will be in by-16 for those parts that support a by-16 mode, and by-8 for those devices that only support by-8. The HEX file format is standard INHX8M format for either the by-8 or by-16 devices.

3.4.7   Programming the Device

Now that the data is in program memory, you can program the device. Click **Program** in the PRO MATE Device Programmer dialog. A window will indicate the progress, and will indicate success or failure when finished.

3.4.8   Verifying the Programming

Click **Verify** to double-check the programming in the device. If any address locations on the device do not match program memory, an error log will display the discrepancies.

3.5 Example: Programming a Smart Serial™ Memory Device

The PRO MATE II supports many of Microchip's Memory devices, including the Smart Serial EEPROMs.

This example leads you through the steps involved in programming a Smart Serial™ memory device. This example does not discuss programming the device that controls the memory device. For a programming example for a mid-range PICmicro device, such as the PIC16F84, refer to Section 3.2.

3.5.1 Programming Overview – Smart Serial Memory Devices

Programming a Smart Serial memory device involves the following steps:

- Configuring the Serial Port for PRO MATE II
- Setting up the MPLAB Development Mode
- Creating the MPLAB Project and Assembly Code
- Building the Hex File
- Enabling PRO MATE II
- Setting the Configuration Bits
- Programming the Device
- Verifying the Programming

3.5.2 Before You Begin

Before you can begin this tutorial, you must install the PRO MATE II hardware (Section 2.4) and the MPLAB IDE software (Section 2.5). Make sure you have read and completed the instructions in Section 2.8 through Section 2.11. Make sure that your PC and PRO MATE II are communicating and the PRO MATE menu item appears on the MPLAB menu before you begin this tutorial.

3.5.3 Setting Up the MPLAB Development Mode

Select *Options > Development Mode* to open the Development Mode dialog (Figure 3.13) and select a processor module or device.

Under Tools, select None (Editor Only). Select the 24AA65 device and click **OK**.
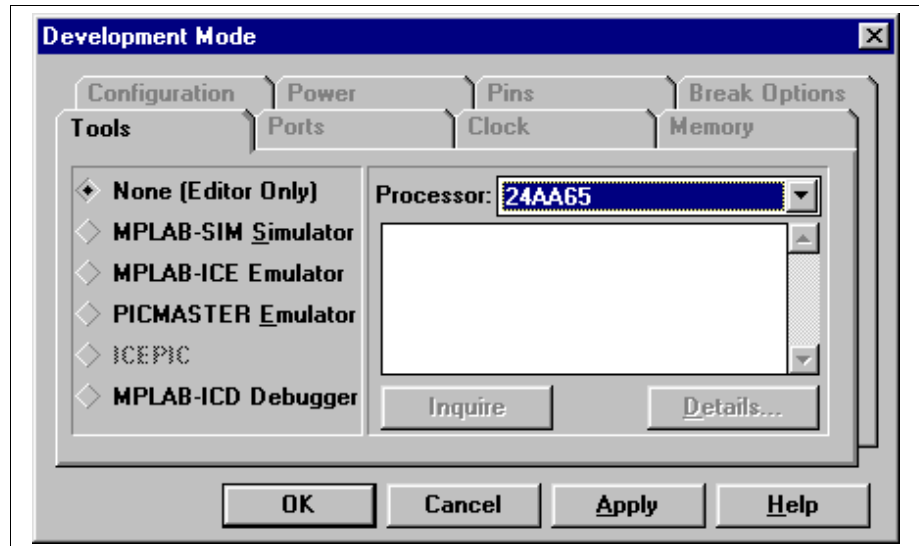
Figure 3.38: Setting the Development Mode

3.5.4 Creating the MPLAB Project and Assembly Code

In order to program the device, you'll need a hex file. In other examples you simply imported an existing hex file. In this example, you'll create our own source code and compile it into a hex file using MPLAB Projects.

3.5.4.1 Creating the Project

Select *Project > New Project*. Name the new project `memaa65` and select the drive and directory you want to store it in. In this example, use the MPLAB directory. Click **OK**.
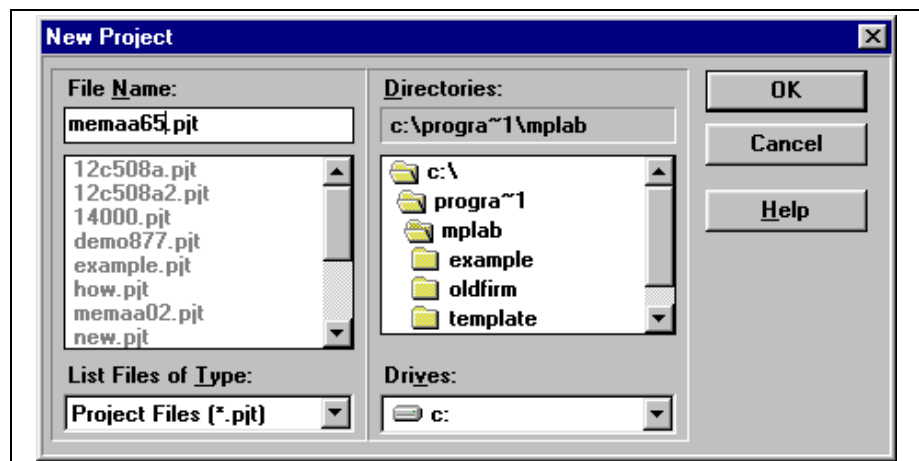
Figure 3.39: Creating a Project

PRO MATE II User's Guide

The Edit Project dialog will appear. It should show the Editor Only development mode and the 24AA65 device. The name of the hex file that this project will create appears in the Project Files area.
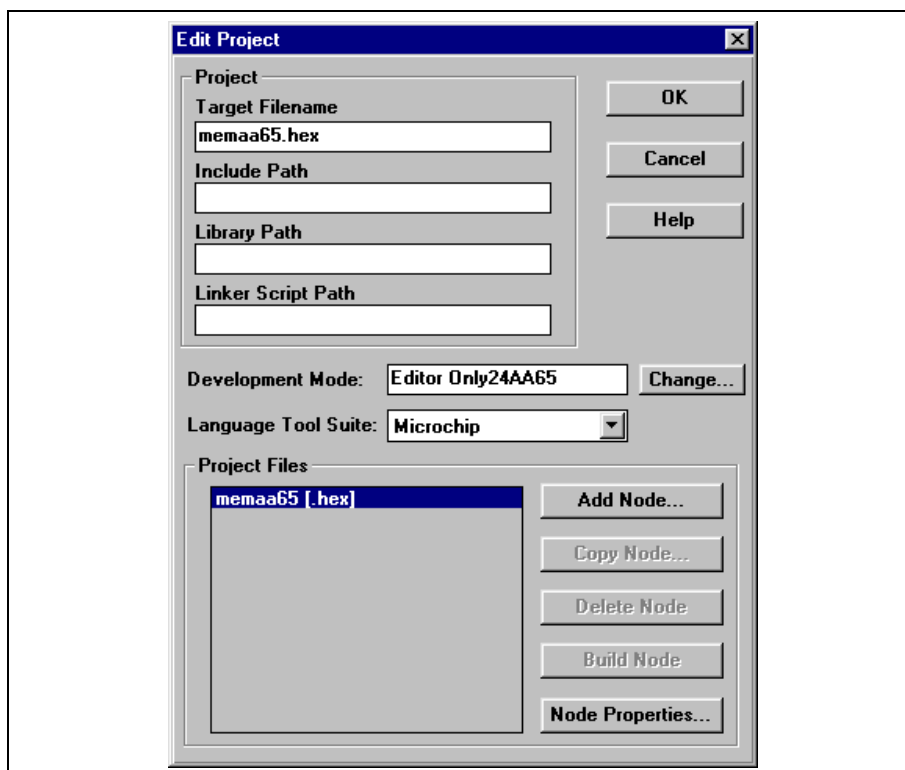
Figure 3.40: Edit Project Dialog

Select the hex file `memaa65.hex` and click **Add Node**.
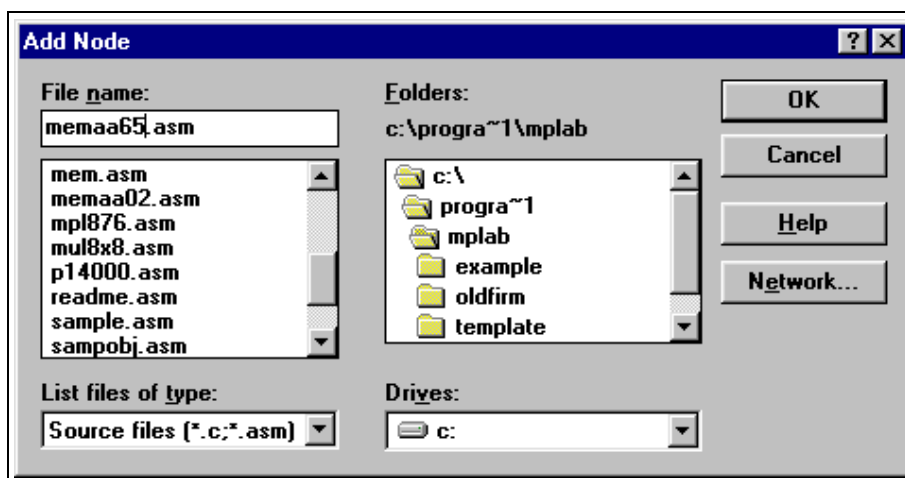
Figure 3.41: Adding the Assembly File

Type `memaa65.asm` in the File name box and click **OK**. The file `memaa65.asm` appears below `memaa65.hex` in the Project Files list in the Edit Project dialog.

Click **OK** to close the Edit Project dialog. Select *Project > Save Project* to save your project.

3.5.4.2   Creating the Assembly Code

At this point, you still need to create the source code that you will assemble to create your hex file. Select *File > New*. An Untitled source code window will open on your MPLAB desktop. Type the following lines (including indentation) in the window:

```
list p=eeprom8
#include "memory.inc"
list m=_24AA65
data 1, 2, 3, 4, 5
data "My Program v1.0", 0
end
```

When you are finished, save your file. Select *File > Save As* and specify the name `memaa65.asm`.
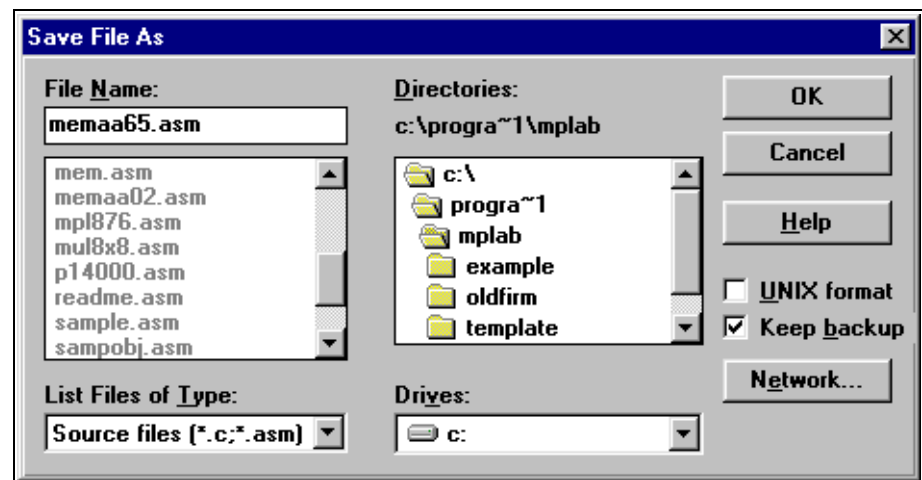
Figure 3.42: Saving Your Source File

3.5.5   Building the Project's Hex Code

Now, build the project and create the hex file. Select *Project > Build All*. If the build did not complete successfully, double-click on the line of the Build Results window that contains the error. Correct the error, save the .asm file, and try your build again.

```
■ Build Results
Building MEMAA65.HEX...

Compiling MEMAA65.ASM:
Command line: "C:\PROGRA~1\MPLAB\MPASMWIN.EXE /aINHX8M /e+ /l+ /x- /c+ /q C:\PRO|

Build completed successfully.

◄
```
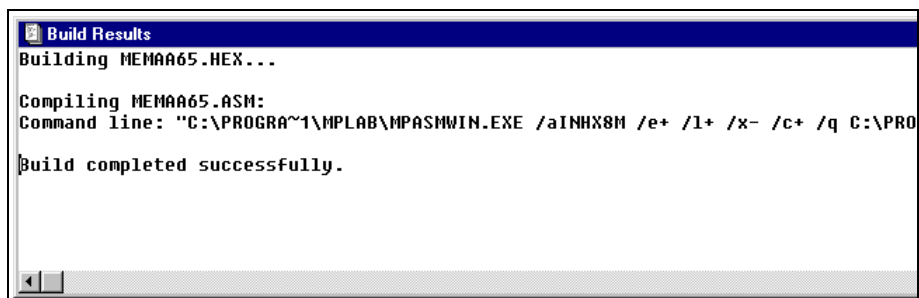
Figure 3.43: Build Results Window

3.5.6   Starting PRO MATE II

To enable PRO MATE II, select *PRO MATE > Enable Programmer* from the MPLAB menu. The PRO MATE Device Programmer dialog will appear when the programmer is enabled. The Configuration Bits dialog opens when you enable the programmer. The Program Memory will also open, and will display the hex file you built. If the Program Memory window did not open, select *Window > Program Memory* to open it. You can resize or move the Program Memory window on your display.
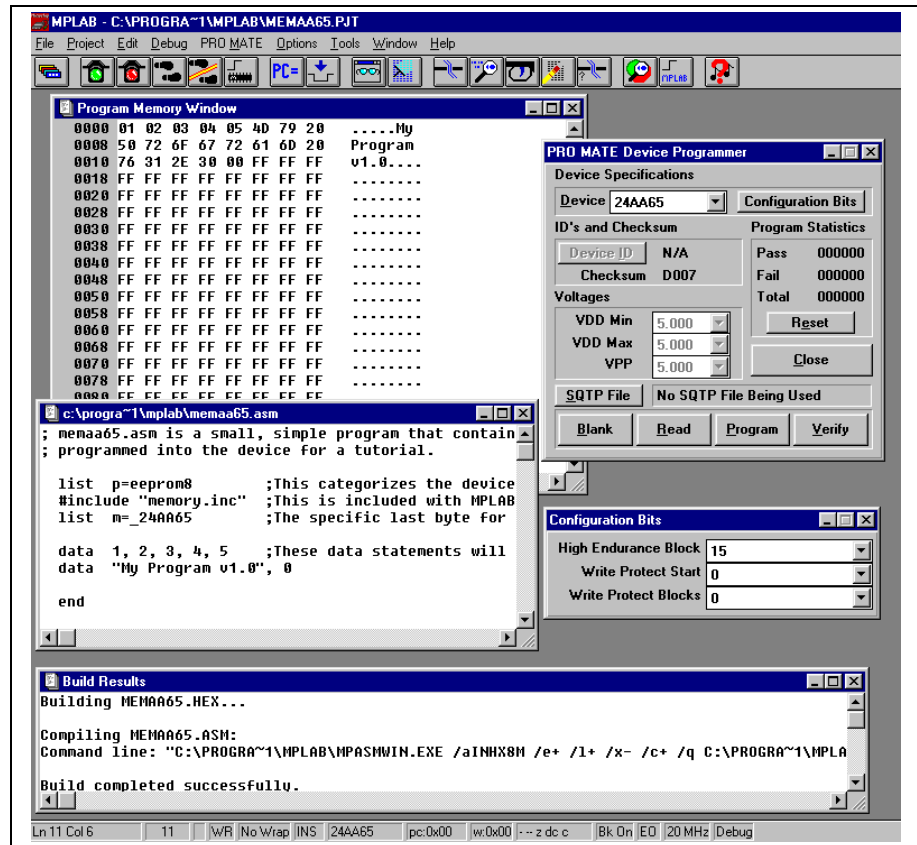
Figure 3.44: Desktop with PRO MATE II Enabled

3.5.6.1   The PRO MATE Device Programmer Dialog

The PRO MATE Device Programmer dialog must remain open during your programming session. Several of the selections in the dialog are disabled. Memory devices are programmed and verified at 5 volts VDD.
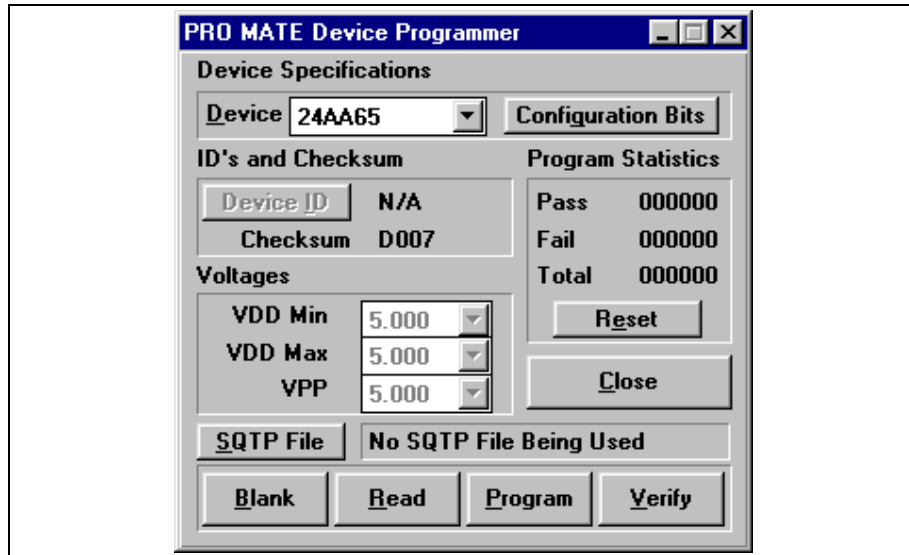
Figure 3.45: Device Programmer Dialog - 24AA65

3.5.6.2   Configuration Bits Dialog

Unlike the 24AA02, the Smart Serial devices have configuration bits. The Configuration Bits dialog opens when you enable the programmer. If you close the Configuration Bits dialog, you can click **Configuration Bits** in the PRO MATE Device Programmer dialog to reopen it.
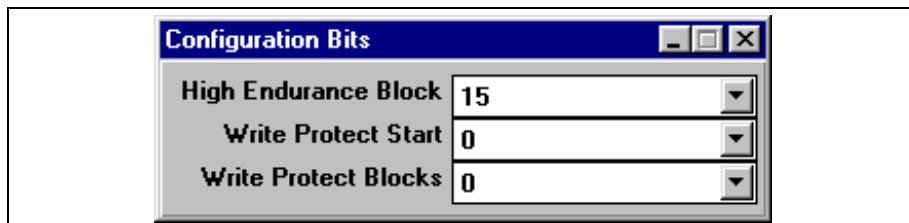
Figure 3.46: Setting Configuration Bits

Set the values of the High Endurance Block, Write Protect Start, and Write Protect Blocks in this dialog. You cannot set these bits in your source code.

3.5.6.3 The Program Memory Window

The MPLAB Program Memory window displays the hex code that you will program onto our 24AA65 device. If the Program Memory window is not open, select *Window > Program Memory* to open it. If your Program Memory window does not look similar to Figure 3.37, click the system icon (to the left of the window's title) and select Hex Code Display. The window displays the hex values and the ASCII text from the source code.

```
Program Memory Window                                    _ □ ×
0000 01 02 03 04 05 4D 79 20    .....My
0008 50 72 6F 67 72 61 6D 20    Program
0010 76 31 2E 30 00 FF FF FF    v1.0....
0018 FF FF FF FF FF FF FF FF    ........
0020 FF FF FF FF FF FF FF FF    ........
0028 FF FF FF FF FF FF FF FF    ........
0030 FF FF FF FF FF FF FF FF    ........
0038 FF FF FF FF FF FF FF FF    ........
0040 FF FF FF FF FF FF FF FF    ........
0048 FF FF FF FF FF FF FF FF    ........
0050 FF FF FF FF FF FF FF FF    ........
0058 FF FF FF FF FF FF FF FF    ........
0060 FF FF FF FF FF FF FF FF    ........
0068 FF FF FF FF FF FF FF FF    ........
0070 FF FF FF FF FF FF FF FF    ........
0078 FF FF FF FF FF FF FF FF    ........
0080 FF FF FF FF FF FF FF FF
```
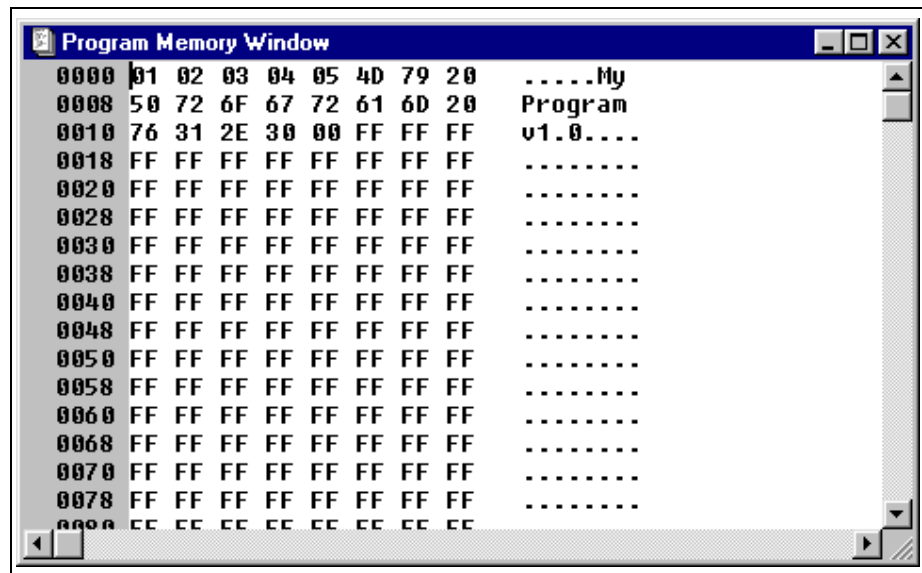
Figure 3.47: `memaa65` in Program Memory

The data displayed in the Program Memory window will be in by-16 for those parts that support a by-16 mode, and by-8 for those devices that only support by-8. The HEX file format is standard INHX8M format for either the by-8 or by-16 devices.

3.5.7 Programming the Device

Now that the data is in program memory, you can program the device. Click **Program** in the PRO MATE Device Programmer dialog. A window will indicate the progress, and will indicate success or failure when finished.

3.5.8 Verifying the Programming

Click **Verify** to double-check the programming in the device. If any address locations on the device do not match program memory, an error log will display the discrepancies.

PRO MATE II User's Guide

NOTES:

Chapter 4. Using PRO MATE II with the MPLAB IDE

4.1   Introduction

This chapter describes the main steps in programming and reading a device using the PRO MATE II development programmer.

4.2   Highlights

Topics covered in this chapter:

- Setup for Programming a Device
- Programming a PICMicro Device
- Verifying the Programming
- Reading a Device

4.3   Before You Begin

Before you can use your PRO MATE II, you must perform the following initialization steps:

1. Configure the serial port that the MPLAB IDE will use to communicate with the PRO MATE II. Select *Options > Programmer Options > Communications Port Setup* and follow the instructions in Section 2.8.
2. If the PRO MATE menu item does not appear in the MPLAB menu, select PRO MATE II as your programmer. Select *Options > Programmer Options > Select Programmer* as described in Section 2.9.
3. Select development mode other than an emulator mode. Select *Options > Development Mode* and follow the instructions in Section 2.10.
4. Enable (start) the programmer in order to display the PRO MATE II dialogs you will use to set up your programming parameters. Select *PRO MATE > Enable Programmer* as described in Section 2.11.

4.4   PRO MATE II Dialogs

The PRO MATE Device Programmer dialog (Figure 4.1) and the Configuration Bits dialog (Figure 4.2) are displayed whenever the programmer is enabled. Closing the PRO MATE Device Programmer dialog will disable the programmer. The options on the screen will show the current values. If the option is unavailable, the item appears in gray (not black) text. The bits listed in the Configuration Bits dialog depend on the device you have selected.

The next section will discuss how these dialogs are used to program devices.

4.5   Setup for Programming a Device

To program a device, you will need:

- A hex file to program into the PICmicro device (complete the tutorials in the *MPLAB IDE User's Guide* to learn how to generate application code using the MPLAB IDE

- A blank PICmicro device to program

Before you can program the device, you must have done the following:

- Installed the PRO MATE II hardware (Section 2.4) and MPLAB IDE software (Section 2.5)

- Established communications between the PRO MATE II and PC (Section 2.8)

- Selected the PRO MATE II as your MPLAB programmer (Section 2.9)

- Set up the MPLAB Development Mode (Section 2.10)

- Enabled PRO MATE II (Section 2.11)

The next step is to set up the PRO MATE Device Programmer and Configuration Bits dialogs (Section 4.5.1 and Section 4.5.2). Then you will either load a hex file into program memory or assemble/compile one in program memory from your source code (Section 4.5.4). After this, you will check that your device is blank (Section 4.5.5) before programming it (Section 4.6).

4.5.1   Setting Up the Device Programmer Dialog

The PRO MATE Device Programmer dialog (Figure 4.1) is always open when PRO MATE II is enabled. Closing this dialog will disable the programmer.

Using PRO MATE II with the MPLAB IDE
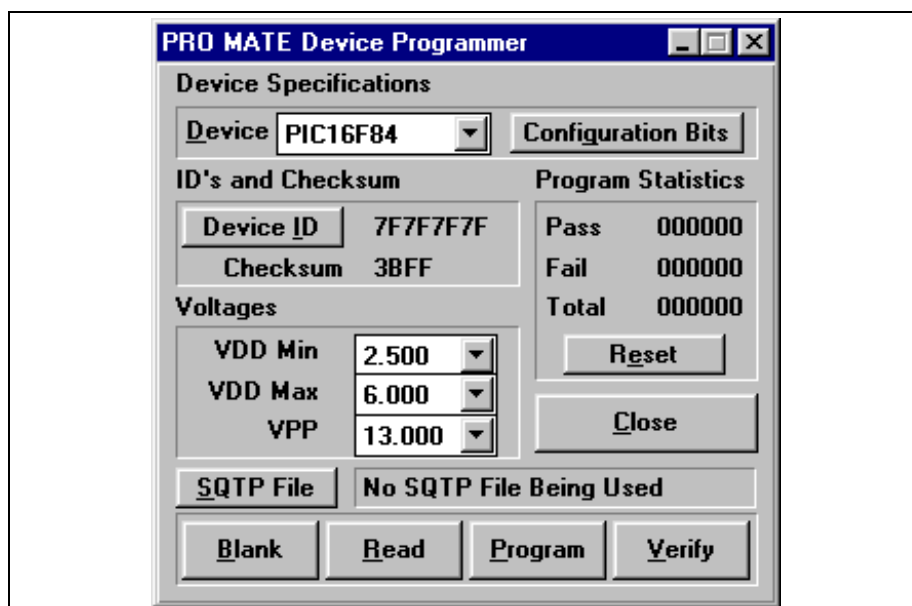
Figure 4.1: PRO MATE Device Programmer Dialog

The device list shows the device that was selected when you set up the MPLAB IDE development mode. If you wish to program a different device, you may select it here or by selecting *Options > Development Mode*. However, changing the device at this point will close any open MPLAB projects (unless you are in Editor Only mode) and clear the program memory, configuration bits, and ID locations.

If your device is not listed, you will need to upgrade your version of MPLAB software, and possibly your version of the PRO MATE II operating system (Section 7.12).

Configuration bits are set in the Configuration Bits dialog, discussed in Section 4.5.2. If you close the Configuration Bits dialog, you can reopen it by clicking **Configuration Bits** in the PRO MATE Device Programmer dialog.

You can edit the device ID value or program the unprotected checksum by clicking **Device ID**. An ID value set here overrides any value set in the program.

You can reset Program Statistics by clicking **Reset**.

The Voltages section is set to default voltages, but can be changed. See Section 4.5.3 for more details on when and how to change these settings.

SQTP File allows you to choose the SQTP[SM] file you wish to use for serial programming, discussed in Section 7.11.

For more detailed information on the PRO MATE Device Programmer dialog options, please see Section 7.3.

4.5.2   Setting Up the Configuration Bits Dialog

The Configuration Bits dialog (Figure 4.2) opens when PRO MATE II is enabled. If you close the Configuration Bits dialog, you can reopen it by clicking **Configuration Bits** in the PRO MATE Device Programmer dialog.
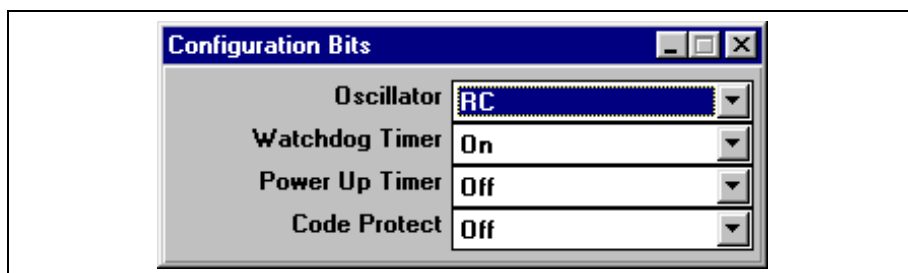
Figure 4.2: Configuration Bits Dialog – PIC16F84

The configuration bits shown are for a PIC16F84 device. The type and number of configuration bits you see will depend on the device you have selected.

If you have configuration bits set in your source code, then the configuration bits shown in this dialog will be updated with those values every time you rebuild your project. If you do not set configuration bits in your source code, the bits will not be changed in this dialog. If you change the configuration bits from their default values using the Configuration Bits dialog, they will be programmed into the PICmicro device when you program the microcontroller.

An ID value set in the Edit ID dialog overrides any value set in the program.

> **Note:**   Setting configuration bits here will not effect emulator or simulator operation. To do so, use *Options > Development Mode*. The Clock and Configuration tabs contain most of the bit settings.

4.5.3   Changing Voltage Settings

Change voltage settings only if your application runs at the extreme voltage operating range. Most users will never need to change the default voltage settings.

- V_{DD} Max and V_{DD} Min are the voltages at which programmed micro-controller devices will be verified.
- V_{PP} is the voltage at which microcontroller devices will be programmed.

The PRO MATE Device Programmer dialog displays the current voltage settings. To change a voltage, select the correct V_{DD} minimum, V_{DD} maximum, or V_{PP} voltage value from the list next to each setting.

> **Note:** The voltage range and default voltage setting may be different for each microcontroller device type. Refer to the specific device programming specification for voltage values for each device.

Voltage Settings		
Selection		**Default Voltage Value**
V_{DD} Min	(3.00 … 6.00)	3.00
V_{DD} Max	(3.00 … 6.00)	6.00
V_{PP}	(12.5 … 13.5)	13.25

4.5.4   Loading the Hex Code into Program Memory

If you have a hex file (e.g., code.hex) ready for programming into a PICmicro microcontroller, select *File > Import > Import to Memory* to load your hex code into the MPLAB Program Memory window.

> **Note:**   There is no warning for importing files with invalid hex values. That is, the hex file of a 14-bit device may be loaded into the program memory of a 12-bit device, and the hex file of a 16-bit device may be loaded into the program memory of a 14-bit device.
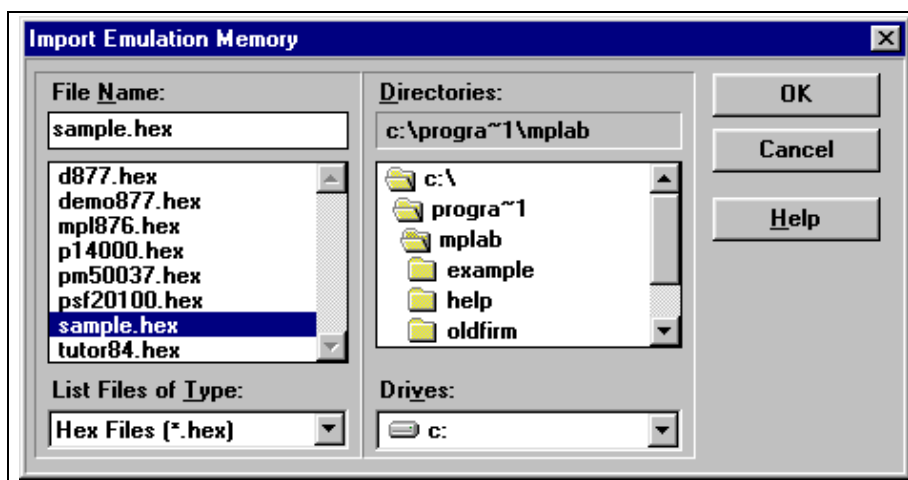
Figure 4.3: Download Emulation Memory Dialog

The Program Memory window should now contain the hex code from the hex file (Figure 4.4). If the Program Memory window is not open, select *Window > Program Memory* to open it.
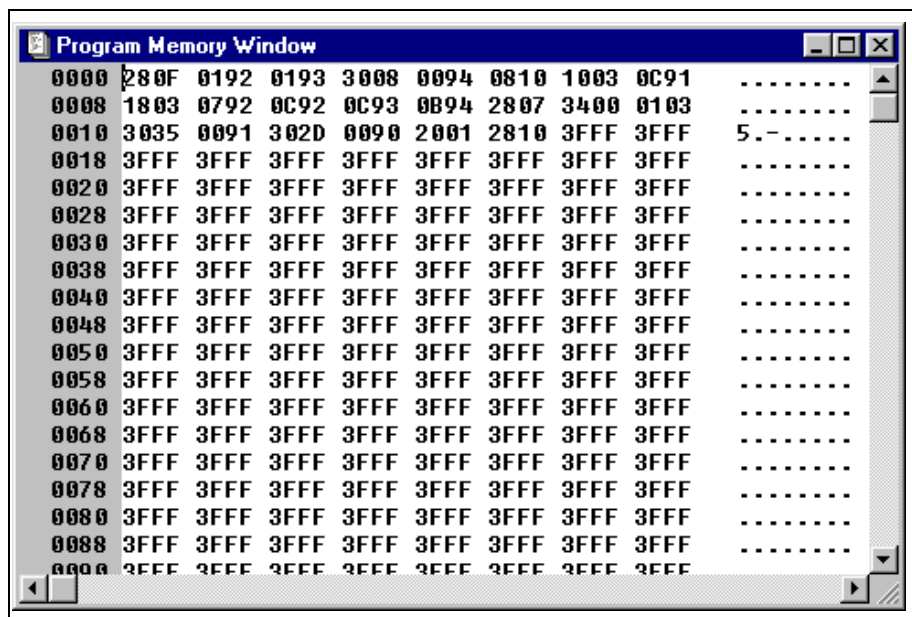
Figure 4.4: Program Memory — Hex Code Display

If you do not have a hex file with which to program your device, you can build one using MPLAB Projects. MPLAB provides a text editor for generation of source code and compatibility with various assemblers/compilers for assembling/compiling your source code into hex code in MPLAB program memory. For an example on using MPLAB Projects to create a hex file for device programming, refer to Section 3.2. Please read the *MPLAB IDE User's Guide* to see how to use MPLAB Projects to develop your own firmware. Each time you rebuild your project, the Program Memory window will be updated.

4.5.5   Checking For a Blank Device

Insert the device to be programmed into the PRO MATE II socket. Position pin one on the device to be in the pin one position as shown on the diagram next to the socket. Secure the device by pushing down the silver lever on the socket.

Click on **Blank** in the PRO MATE Device Programmer dialog to verify that the device is completely blank (all bits are set to a "1"). This will also verify that all configuration bits are set to a "1" (unprogrammed) state. You can also perform a blank check from _PRO MATE II > Blank Check All_.

If you are using a one-time programmable (OTP) device, some configuration bits (e.g., oscillator bits) might have already been programmed at the factory. Set the configuration bits to the factory settings and select _PRO MATE II > Blank Check OTP_. This will check that all program memory bits are set to 1, and that the configuration bits match the value in the dialog. An OTP device cannot be erased and reprogrammed.

If the EPROM device is not blank, you will have to erase it before programming or use another device. To erase a windowed device:

- Remove any labels covering the device window. If you do not have a windowed device (Figure 4.5), you cannot reprogram it. A windowed version of all EPROM devices may be ordered by requesting the JW package.
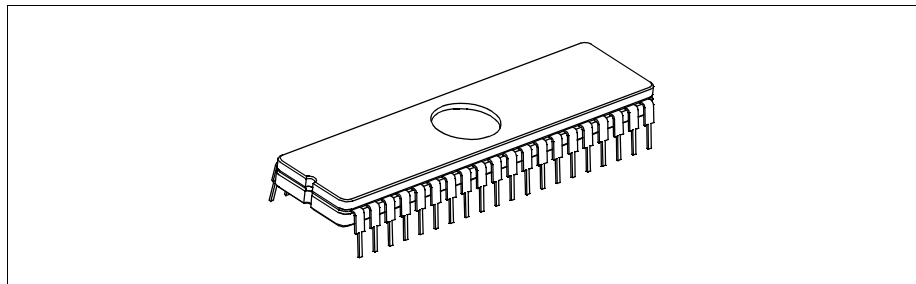
Figure 4.5: Windowed Device

- Place the device in an Ultraviolet (UV) EPROM Eraser. The amount of time required to completely erase a UV erasable device depends on: the wavelength of the light, its intensity, distance from UV source, and the process technology of the device (the size of the memory cells).

- Verify that the device is blank (i.e., perform the blank check again) before attempting to program it.

If the device is EEPROM/FLASH, you do not have to erase it before reprogramming it. These devices are electrically erased before programming.

4.6   Programming a PICmicro Device

Click **Program** in the PRO MATE Device Programmer dialog to program the entire device (all of program memory, configuration bits, etc.). If you want to program selectively (e.g., part of program memory, only configuration bits), then select *PRO MATE > Program/Verify* to open the Program/Verify dialog (Figure 4.6).
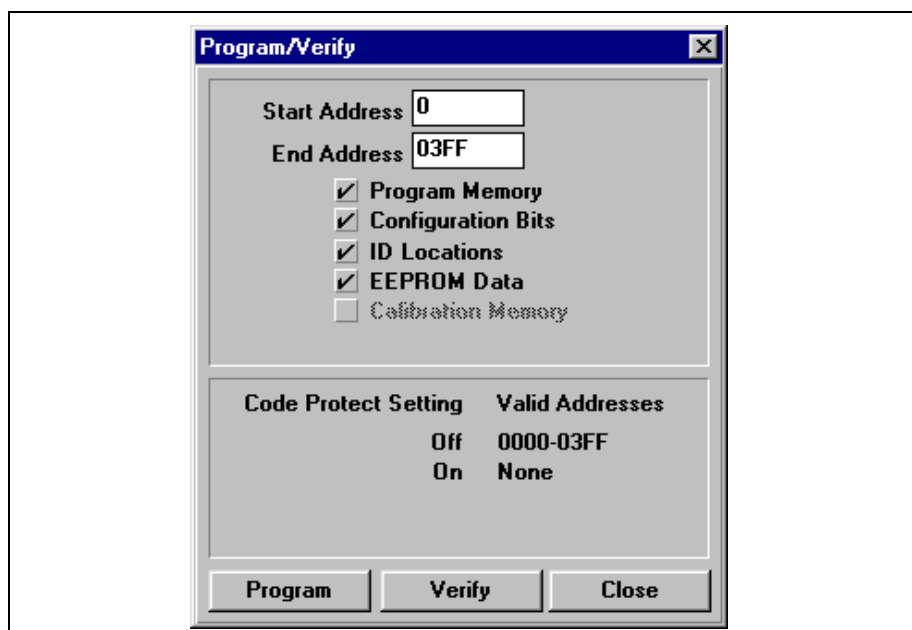
Figure 4.6: Program/Verify Dialog

Check the following settings, then click **Program** to program the device according to your settings. The memory area corresponding to the checked boxes will be programmed. Areas that are grayed out are not available on the device.

Start Address, End Address
These values default to specify the entire range of memory for the device. Specify the memory range you want to program.

Program Memory
Select this option to program or verify the program memory range specified by the Start Address and End Address.

Configuration Bits
Select this option to program or verify the configuration bits. You can set the configuration bits using the Configuration Bits dialog.

ID Locations
Select this option to program the ID locations. You can set the ID Locations by clicking **Device ID** in the PRO MATE Device Programmer dialog.

EEPROM Data	For devices with data EEPROM, program the data memory from data in the EEPROM Memory window. Refer to the *MPLAB IDE User's Guide* for more information on data EEPROM and the EEPROM Memory window.
Calibration Memory	For devices with calibration memory, program the calibration memory from data in the Calibration Memory window. For more information on calibration memory, refer to Section 3.3.
Code Protect Settings	Shows the status and availability of code protection for the selected device.

When the programming is finished, "Success" or "Failure" will appear to the right of the Start Address. If the programming failed, an error window will appear showing the good (expected) data and the bad (actual) data for each address it attempted to program. If the bad data shows 0000, try reseating the socket module. Then, do a blank check (*PRO MATE > Blank Check All*) and try to program the device again.

4.7   Verifying the Programming

Click **Verify** in the PRO MATE Device Programmer dialog or the Program/Verify dialog to verify that the contents of the device match the program memory, EEPROM data, calibration memory, ID locations, and configuration bits in MPLAB and in the PRO MATE Device Programmer dialog and Configuration Bits dialog.

4.8   Reading a Device

If you want to copy the firmware from a programmed PICmicro device into an unprogrammed device, you can read the programmed firmware (program memory, configuration bits, etc.) into the MPLAB IDE, and then program the new device based on this information.

Click **Read** in the PRO MATE Device Programmer dialog to read the entire device (i.e., all of program memory, configuration bits, etc.). If you want to read selectively (e.g, part of program memory, only configuration bits), then select *PRO MATE > Read Device* to open the Read Device dialog (Figure 4.7).
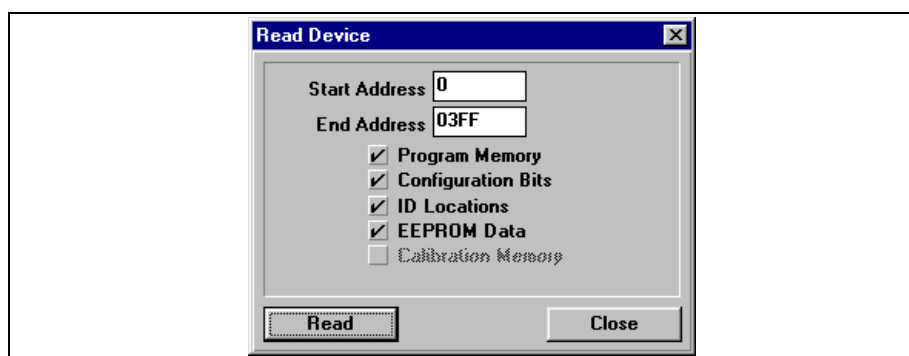
Figure 4.7: Read Device Dialog

Check the following settings, then click **Read** to read the device according to your settings. The memory area corresponding to the checked boxes will be read. Areas that are grayed out are not available on the device.

Start Address, End Address	These values default to specify the entire range of memory for the device. Specify the memory range you want to read.
Program Memory	Select this option to read the program memory range specified by the Start Address and End Address.
Configuration Bits	Select this option to read the configuration bits.
ID Locations	Select this option to read the ID locations.
EEPROM Data	For devices with data EEPROM, read the data in the data memory into the EEPROM Memory window. Refer to the *MPLAB IDE User's Guide* for more information on data EEPROM and the EEPROM Memory window.
Calibration Memory	For devices with calibration memory, read the data in calibration memory into the Calibration Memory window. For more information on calibration memory, refer to Section 3.3.

After reading a device into the MPLAB IDE, its data will appear in the Program Memory window. You can use the Modify dialog to change the data (*Window > Modify*), then save it to a hex file by selecting *File > Export > Export Hex File*.

If you have an MPLAB-ICE emulator connected to the MPLAB IDE, your code will be downloaded to the emulated program memory of the MPLAB-ICE. If you have a project open, you will be asked if you want to close it before reading memory from a device. If you keep your project open, the Absolute Listing window and the Source window might not match the data you have read into the Program Memory window. Also, symbols may not match the proper addresses in the Program Memory window.

Chapter 5. Using PRO MATE II with PROCMD

5.1   Introduction

This chapter covers the details of using the command line program PROCMD to control PRO MATE II operations. When running under DOS, the PRO MATE II device programmer must be connected to a PC that has the PROCMD software. For information on installing PROCMD software on a PC, see Chapter 2.

5.2   Highlights

Topics covered in this chapter:

- Getting Started with PROCMD
- Examples of Use
- Usage Details

5.3   Getting Started with PROCMD

Once you have PROCMD installed on your PC (Chapter 2), you can use it to send commands to the PRO MATE II.

5.3.1   Operating System Updates

If PRO MATE II did not recognize your socket module (the message "Socket not supported" appears on power-up), you may need to update the operating system. You can obtain the latest PRO MATE II operating system from the Microchip web site (www.microchip.com). Once you have the update, you will need to download this file to the PRO MATE II.

1. Turn the power off on the PRO MATE II.
2. While holding down the **F1** and **F3** keys, turn the power back on.
3. You should get the following messages on the PRO MATE II LCD: "Clearing Memory Now....." and "Ready for Download...." If you do not, repeat the previous steps.
4. Download the operating system file using PROCMD as follows:
   PROCMD /<number> /d<filename>
   where `<number>` = 1, 2, 3 or 4, depending on the COM port you are using, and `<filename>` is the name of the operating system file.
5. Once the download is done, PRO MATE II will automatically restart, as if from power-on.

5.3.2 Device Programming

You are now ready to program a device. Place the device in the socket module. Then execute the following:

PROCMD /<number> /p<partname> /f<filename> /m

where:

> `<number>` = 1, 2, 3 or 4, depending on the COM port you are using
> `<partname>` is the name of the device
> `<filename>` is the name of the hex file
> `/m` is the command to program the part.

As an example:

PROCMD /1 /p16c74a /f16c74a.hex /m

will program a PIC16C74A device with the contents of the file `16c74a.hex`, using COM1 to communicate with PRO MATE II. Only the addresses specified in `16c74a.hex` are programmed.

If you specify a hex file on the command line, it programs only the contents of the hex file. If the hex file contains the starting and ending address for the device, the command will program the entire device as if the **PGM** button was pressed. If the hex file does not contain the first and last locations, the command will use the program range command and will only program the locations contained in the hex file.

If you do not specify a hex file on the command line, it will program the entire device using the contents of the programmer memory.

It is recommended that you first download the file then use the `/m` command to program the entire part. This is especially important when trying to reprogram a FLASH part, because the full program command will first erase the device prior to programming. The program range command does not erase the device first.

For information on error messages you may receive, see Section 8.4.

5.3.3 Verifying A Device's Programming

The PRO MATE II Command Line Interface now has a Verify function (`/Y`). If you specify a hex file on the command line, it verifies only the contents of the hex file. If the hex file contains the starting and ending address for the device the command will use the full verify as if the **VFY** button was pressed. If the hex file does not contain the first and last locations, the command will use the verify range command and will only verify the locations contained in the hex file.

If you do not specify a hex file on the command line, it will verify the entire device using the contents of the programmer memory.

Examples:

To verify against a known good hex file:

procmd /1 /p16f84 /fhexcode.hex /y

Or

procmd /1 /p16f84 /fhexcode.hex
procmd /1 /p16f84 /y

To explicitly perform a verification after a program:

procmd /1 /p16f84 /fhexcode.hex /m /y

Or

procmd /1 /p16f84 /fhexcode.hex
procmd /1 /p16f84 /m /y

5.3.4   Stand-Alone Mode Setup

You set up the PRO MATE II for use in stand-alone mode by sending the hex or SQTP file that you will use to program your device(s) to the PRO MATE II:

PROCMD /<number> /p<partname> /f<filename>
PROCMD /<number> /p<partname> /s<filename>

where:

<number> = 1, 2, 3 or 4, depending on the COM port you are using
<partname> is the name of the device
<filename> is the name of the hex or SQTP file
("f" precedes a hex file and "s" precedes an SQTP file)

As an example:

PROCMD /1 /p16c74a /f16c74a.hex /m

will send the file 16c74a.hex, used to program PIC16C74A devices, via COM1 to the PRO MATE II.

Now you are ready to use PRO MATE II in stand-alone mode. See Chapter 6 for more details.

5.4   Examples of Use

Other examples of use for the command line interface are as follows:

PROCMD /? or PROCMD

Help screen displayed.

PROCMD /1 /DPF40007P.HEX

Operating system file PF40007P.HEX downloaded to PRO MATE II on COM1.

PROCMD /1 /P16C54A /R

PRO MATE II on COM1 set to a PIC16C54A, PRO MATE II set to safe mode.

`PROCMD /1 /P16C54A /N4500 /X5500 /FPROGRAM.HEX`

PRO MATE II on COM1 set to a PIC16C54A, V_{DD} Min set to 4.5V, V_{DD} Max set to 5.5V. V_{PP} set to the device's default value. File `PROGRAM.HEX` sent to the PRO MATE II.

`PROCMD /1 /P16C54A /N4500 /X5500 /FPROGRAM.HEX /M`

PRO MATE II on COM1 set to a PIC16C54A, V_{DD} Min set to 4.5V, V_{DD} Max set to 5.5V. V_{PP} set to the device's default value. File `PROGRAM.HEX` sent to the PRO MATE II. Device programmed with specified voltages. Only the addresses specified in `PROGRAM.HEX` are programmed.

`PROCMD /2 /P16C61 /FVOLTS.HEX`

PRO MATE II on COM2 set to a PIC16C61, hex file `VOLTS.HEX` sent to the PRO MATE II, voltages set to devices default values.

`PROCMD /2 /P16C61 /FVOLTS.HEX /SSQTP.NUM /M`

PRO MATE II on COM2 set to a PIC16C61, hex file `VOLTS.HEX` sent to the PRO MATE II, voltages are set to the device's default values, and one entry from the SQTP file `SQTP.NUM` is sent. All of the device's program memory is programmed, with blank values in the locations not specified by `VOLTS.HEX` and `SQTP.NUM`. Any other device areas specified by `VOLTS.HEX` are programmed. If programming is successful, the SQTP entry is marked as used.

`PROCMD /2 /P16CR83 /FDATAMEM.HEX /M`

PRO MATE II on COM2 is set to a PIC16CR83, the file `DATAMEM.HEX` is sent to PRO MATE II, voltages are set to the device's default values and the device is programmed. Only the addresses specified in `DATAMEM.HEX` that are for data memory and configuration bits are programmed.

`PROCMD /3 /P17C42A /FAPPCODE.HEX /M`

PRO MATE II on COM3 is set to a PIC17C42A, the file `APPCODE.HEX` is sent to PRO MATE II, voltages are set to the device's default values, and a device is programmed. Only the addresses specified in `APPCODE.HEX` are programmed.

`PROCMD /3 /P17C42A /M`

PRO MATE II on COM3 is set to a PIC17C42A, and a device is programmed with the current contents of the device programmer at the currently selected voltage levels.

`PROCMD /3 /P17C42A /SSERIAL.NUM /M`

PRO MATE II on COM3 is set to PIC17C42A, and one entry from the SQTP file `SERIAL.NUM` is sent. A device is programmed with the current contents of the device programmer and the serial number at the currently selected voltages. If programming is successful, the entry is marked as used.

For information on error messages you may receive, see Section 8.4.

5.5   Demo Programs

There are several demo programs that you may wish to review.

`sqtp1.bat`	An example `.bat` file that shows a SQTP file used to program several devices.
`sqtp2.bat`	An example `.bat` file that shows a SQTP file used to program several devices. Command line arguments are used.
`count2.bat`	An example program that takes a count between 1-9999 on the command line and programs that many devices. Also verbose error messages.
`count3.bat`	An example program that takes a count between 1-9999 on the command line and programs that many devices. The user is queried for options. Also verbose error messages.

5.6   Usage Details

When using PROCMD, consider the following:

- You must always specify the communication port, except when displaying the help screen.

- When downloading the operating system, you can only specify the communications port and you must use the `/d` option.

- You must specify a device except when downloading the operating system.

- It is not necessary to send the hex file every time a device is programmed. In a production environment, it would be beneficial to use one command line execution to transfer the hex file and set up any voltages, and then use subsequent command line executions to program or program with SQTP numbers.

- If you specify a hex file, only the locations specified in the hex file are programmed. If you also specify an SQTP file, then the entire device is programmed.

- If you do not specify a hex file, the entire contents of the PRO MATE II device programmer are programmed into the device.

- All values not specified in the hex file are set to blank (erased) values.

- SQTP information is only sent to the device programmer if the device is programmed.

- SQTP lines are marked as used only if programming is successful.

- When displaying the help screen (`/?`), any other parameters you specify are ignored.

- PRO MATE II now allows programming of 16F87X parts using the low-voltage programming mode with the In-Circuit Serial Programming™ (ICSP™) socket module. Upon selection of part, you must press either the YES or NO button to indicate whether you want to use Low Voltage Programming. If you answer YES then when the device is displayed on the LCD, you will see "LV" after ICSP (following the part name). You must ensure that the device has the Low Voltage Program config bit set to "enabled" or you will not be able to access the part. This is only available when the ICSP socket module is used. If you answer NO to the question, then the part will use the normal high voltage on $\overline{MCLR}$ to program the device. If you do not answer the question prior to enabling PRO MATE II when using the MPLAB IDE, you will get a communication error, since the only place to enter Low Voltage Programming mode is from the buttons on the PRO MATE II. In addition to the other required pins (RB6, RB7, V_{DD}, V_{SS}, V_{PP}), you must connect the low voltage programming pin (RB3) to Pin 2 of the 15-Pin connector on the ICSP socket module.

Chapter 6. Using PRO MATE II in Stand-Alone Mode

6.1   Introduction

This chapter tells you how to use the PRO MATE II device programmer in stand-alone mode. The device programmer provides an LCD interface that gives you complete control over a programming session.

6.2   Highlights

Topics covered in this chapter:

- Getting Started in Stand-Alone Mode
- Programming a Device
- Operating in Safe Mode

6.3   Getting Started in Stand-Alone Mode

PRO MATE II operating in stand-alone mode allows you to read, program, and verify a device without using a PC. Stand-Alone mode is useful in situations where a PC may not be available or even required, such as in the field or in a lab production environment.

6.3.1   Installing a Socket Module

To use the device programmer, you must have a socket module installed. To install the socket module, align the tip of the arrow on the socket module with the tip of the arrow on the PRO MATE II. Tighten the two socket module thumbscrews evenly and, if possible, simultaneously. Avoid over tightening them; they should be finger-tight only.

When you power up the PRO MATE II device programmer, the unit automatically detects the type of socket module installed and initializes the PRO MATE II function buttons F1 through F4. The device programmer then displays the device options that you can choose from for the currently installed socket module. You must install a new socket module prior to selecting a device not supported by the current socket module. If you power-on the device programmer without a valid socket module installed, the unit displays the message "Socket Not Supported."

> **Caution:** Ensure the device programmer is powered OFF before changing a socket module.

> **Caution:** Do not power-up the PRO MATE II with a device loaded in the socket. Damage to the device or PRO MATE II may result.

6.3.2 Downloading a Hex File into PRO MATE II Memory

To set up the PRO MATE II for stand-alone mode, you will need a PC to download the hex file into PRO MATE II memory. Also, you will need a PC to update the PRO MATE II operating system as necessary.

To download a hex file using the MPLAB IDE, select *File > Import > Import to Memory* and select the file you wish to program into the device.

To download the hex or SQTP file into PRO MATE II memory from DOS using PROCMD, use the following command:

```
PROCMD /<number> /p<partname> /f<filename>    HEX
PROCMD /<number> /p<partname> /s<filename>    SQTP
```

where:

> `<number>` = `1`, `2`, `3` or `4`, depending on the COM port you are using
> `<partname>` is the name of the device
> `<filename>` is the name of the hex or SQTP file

As an example:

```
PROCMD /1 /p16c74a /f16c74a.hex /m
```

will send the file `16c74a.hex`, used to program PIC16C74A devices, via COM1 to the PRO MATE II.

Now you are ready to use PRO MATE II in stand-alone mode.

6.4 Programming a Device

After applying power to PRO MATE II, the unit displays the product title and version number. The PRO MATE II operating system then attempts to identify the currently installed socket and displays the socket module name if the socket module is recognized.

PRO MATE II next displays a device selection menu. Select the desired device from the selection menu. After selecting a device, PRO MATE II displays the Command menu.

After you select a device, the device programmer displays the Command menu. You are now ready to program a device.
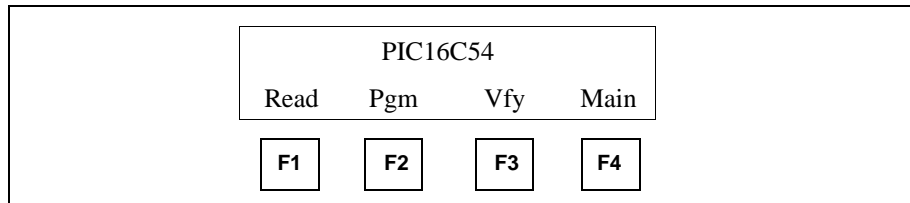
Figure 6.1: Command Menu

Press <F2> to program the device in the socket module with the hex file you previously loaded into the internal memory of the device programmer.

The device programmer checks to see if the installed microcontroller device is blank. If the device is not blank, the device programmer asks if you want to continue. Answer "Yes" to continue. Answer "No" to return to the Command menu.

The device programmer programs the contents of its memory into the microcontroller device loaded in the socket module.

After programming a device without errors, the device programmer performs a check to verify the data programmed into the device, and returns the results of the verification. For the installed device, the device programmer performs the verification at the VDD Minimum and VDD Maximum voltages.

The device programmer reports programming errors and verify errors according to where the error occurred. Errors are reported for program, configuration bits, and ID locations.

After programming, the device programmer momentarily displays the checksum and ID.

6.4.1   Verify   (Vfy, F3)

The device programmer compares the contents of its internal memory to the contents of the programmed microcontroller device loaded in the socket module. If the data and configuration bit settings are correct, "VERIFIED" will display on the LCD. The device programmer performs the verification at the VDD Minimum and VDD Maximum voltages. The device programmer reports errors according to which part of the device failed.

The Verify function also confirms that erased parts are blank. If all programmable locations are blank for a device loaded in the socket module, the device programmer displays "Device erased!" Reported results include the following:

- Same Contents
- Blank Device
- Errors

6.4.2   Read   (Read, F1)

If you want to copy firmware from a programmed device into an unprogrammed device, you can read the programmed firmware into PRO MATE II memory, then program the new device based on this information. Press <F1> to copy the contents of the device in the socket module into the internal memory of the device programmer.

For PIC16CXX devices, the device programmer will ask the question: "Code Protect Parts?" Answer "Yes" to code protect the parts that you will be programming. Code protection will remain effective until a new device is read. Answer "No" to program devices exactly as read.

After reading a device, the device programmer displays a checksum. If the device is code protected, a code protection message will be displayed prior to the read. Answer "Yes" to continue. Answer "No" to return to the Command menu.

6.5   Operating in Safe Mode

Press <F2> (main) and then select Safe Mode to enter safe mode. When in safe mode, the only functions that are allowed are Program and Verify. If any other buttons are pressed, a message is displayed stating that the system is in safe mode and the operation is not allowed.

Safe mode was designed to prevent operational errors during production.

Requesting other functions in the MPLAB IDE when the PRO MATE II is connected to the PC will deactivate safe mode.

Chapter 7. PRO MATE II – MPLAB IDE Reference

7.1 Introduction

This chapter describes the dialogs, windows, and menu items of the
PRO MATE II development system.

7.2 Highlights

Topics covered in this chapter:

- PRO MATE II Device Programmer Dialog
- Configuration Bits Dialog
- Program Memory Window
- Program/Verify Dialog
- Read Device Dialog
- PRO MATE II Menu Items
- Files Used by PRO MATE II
- Using Serial Programming
- Saving and Restoring Calibration Data
- Upgrading the PRO MATE II Operating System

7.3 PRO MATE Device Programmer Dialog

The PRO MATE Device Programmer dialog (Figure 7.1) is displayed
whenever the programmer is enabled. Closing this dialog will disable the
programmer.

The options on the screen will show the current values if active, or will be
disabled (grayed out).

PRO MATE II User's Guide
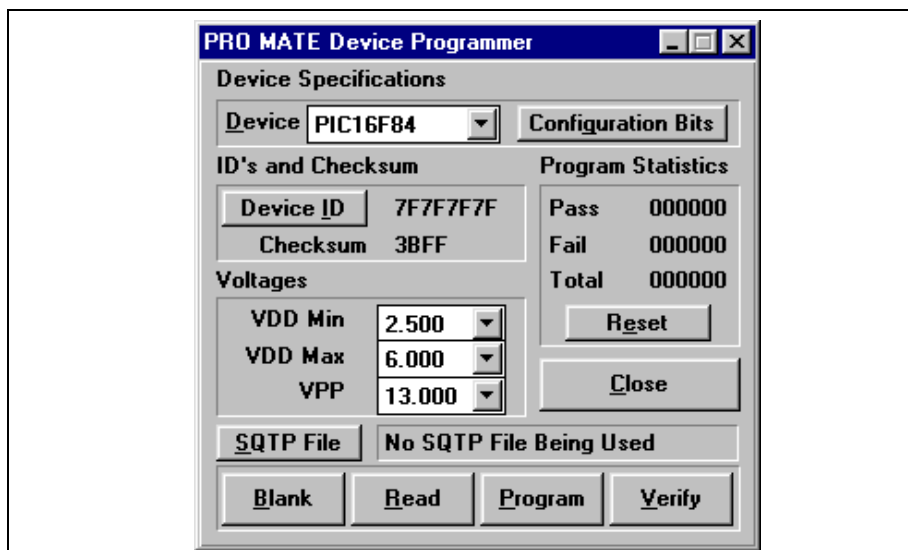
Figure 7.1: PRO MATE Device Programmer Dialog

From this dialog, you can set the following items:

- **Device** – Sets the PICmicro device type to use with PRO MATE II. The default is the device that was selected when you set up the development mode

> **Note:** Setting the device type here will close any open MPLAB projects (except in Editor Only mode) and clear the program memory, configuration bits and ID locations.

- **ID's and Checksum** – Click **Device ID** to open the Edit ID dialog to edit the ID value or program the unprotected checksum to the device. This feature allows you to identify the version of firmware on the device after it is programmed and code-protected. You can also use the _ _IDLOCS directive to set the ID bytes from MPASM. Each time you rebuild your project or reload your hex file, the ID locations will be set according to the values from the _ _IDLOCS directive. An ID value set in the Edit ID dialog overrides any value set in the program.
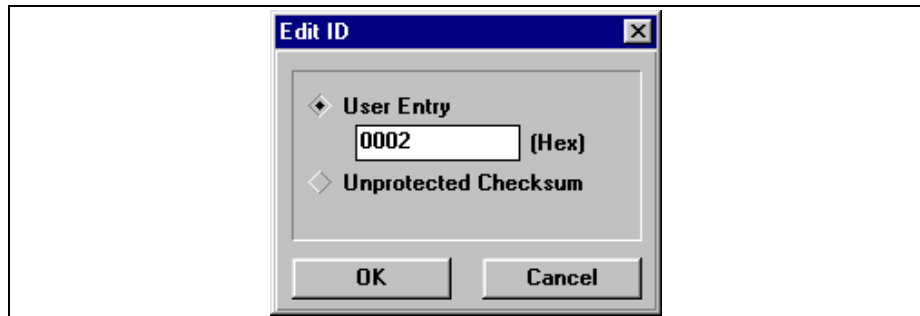
Figure 7.2: Edit ID Dialog

To program an ID value, select User Entry and enter the 4- or 8-digit hex value for the ID. Select Unprotected Checksum to have the unprotected checksum automatically programmed into the ID value for the device. **OK** sets the option, **Cancel** cancels the screen and returns to the PRO MATE Device Programmer dialog.

- **Program Statistics** – Shows the number of successful (Pass) and unsuccessful (Fail) programming instances, as well as the Total. Click on **Reset** to reset the counters.

- **Voltages** – Displays the current voltage settings. To change a voltage, select the correct VDD minimum, VDD maximum, or VPP voltage value from the list next to each setting.

- **Close** – Closes the Device Programmer dialog and disables the programmer.

- **SQTP File** – Programs the device from an SQTP (Serialized Quick Turn Production) file. For instructions on serial programming, see Section 7.11.

On the bottom of the dialog are buttons with the following functions:

- **Blank** – Verifies that the device is completely blank (all bits are set to a "1"). This will also verify that all configuration bits are set to a "1" (unprogrammed) state.

- **Read** – Reads all program memory, configuration bits, ID locations, EEPROM data, and calibration memory on the device.

- **Program** – Programs all program memory, configuration bits, ID locations, EEPROM data, and calibration memory on the device.

- **Verify** – Verifies that the programming on the device matches the program memory, configuration bits, ID locations, EEPROM, and calibration memory values in the MPLAB IDE and in the PRO MATE Device Programmer dialog.

7.4   Configuration Bits Dialog

The Configuration Bits dialog (Figure 7.3) opens when PRO MATE II is enabled. If you close the Configuration Bits dialog, you can reopen it by clicking on **Configuration Bits** in the PRO MATE Device Programmer dialog.
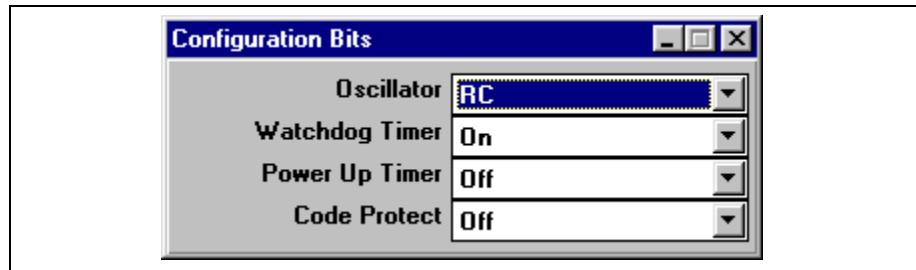
Figure 7.3: Configuration Bits Dialog

The type and number of configuration bits you see will depend on the device you have selected. For more information on the functions of configuration bits for your device, refer to the device's data sheet.

> **Note:**   Setting configuration bits here will not effect emulator or simulator operation. To do so, you will need to set options in the Development Mode dialog (*Options > Development Mode*, **Pins**, **Configuration** and **Clock** tabs.)

You can also specify the configuration bit values in your source code. Use the `_ _CONFIG` directive in MPASM to set the configuration bits for the device to be programmed. Each time you rebuild your project or reload your hex file, the configuration bits will be set according to the values from the `_ _CONFIG` directive.

If you do not set configuration bits in your source code, then these bits will not be changed. You can manually change them from their default values using the Configuration Bits dialog and they will be programmed into the PICmicro device when you program the microcontroller.

7.5   Program Memory Window

When PRO MATE II is enabled, the Program Memory window will be opened in Hex Code format, if it is not already open.

You can view the Program Memory as hex code, machine code, or disassembled with symbols (if available). Change display mode by clicking on the system button in the upper left corner of the window and selecting the display option from the system menu.

When in the MPLAB-ICE emulator mode, Program Memory shows the data that is in the emulation memory of the MPLAB-ICE pod. This memory is read by the MPLAB-ICE probe when you run, single step, or trace using the emulator.

When in the simulator mode or editor only mode, the Program Memory window reflects the contents of a memory buffer on the PC. This memory is read by MPLAB-SIM when you run, single step, or trace.

If you read program memory with PRO MATE II while in emulator or simulator mode, it will overwrite the program memory being emulated or simulated. This can cause "mismatches" between the program memory and debug information if you have an MPLAB project open. You will be given an option to either close the current MPLAB project or to continue and read the device's memory into the Program Memory window. Subsequent debug operations might not work properly.
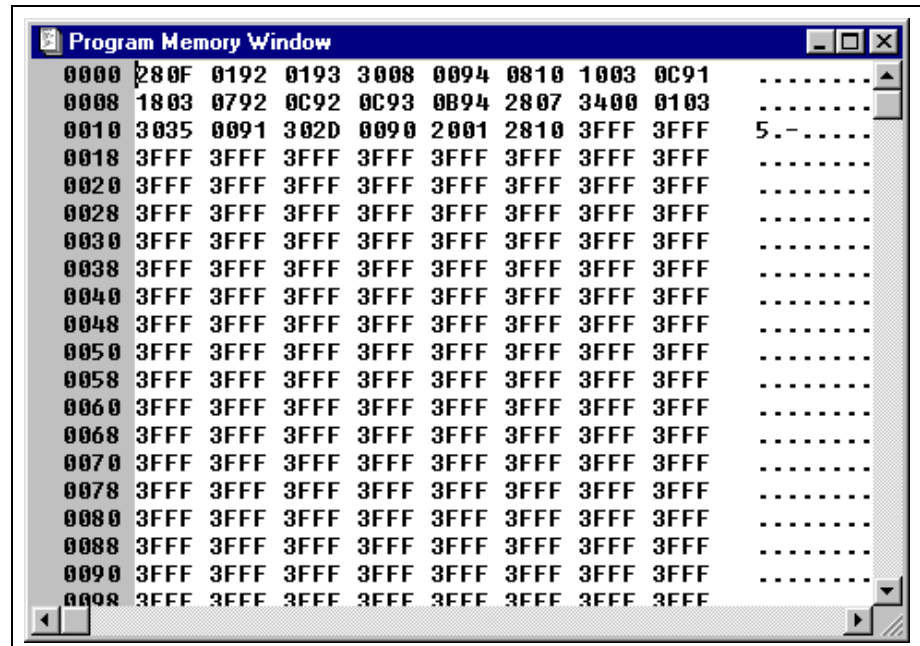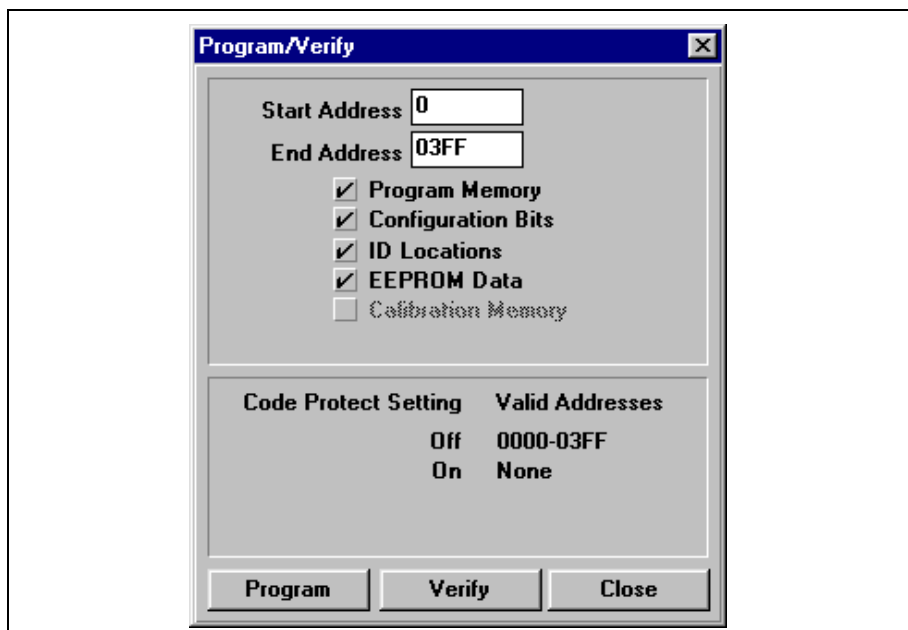

Figure 7.4: Program Memory Window

7.6   Program/Verify Dialog

To program or verify selectively (e.g., part of program memory, only configuration bits), select *PRO MATE > Program/Verify* to open the Program/Verify dialog (Figure 7.5).

PRO MATE II User's Guide

Figure 7.5: Program/Verify Dialog

Check the following settings. Then click **Program** to program the device or click **Verify** to verify that the contents of the device match the program memory, EEPROM data, calibration memory, ID locations, and configuration bits in MPLAB and in the PRO MATE Device Programmer dialog and Configuration Bits dialog.

Start Address, End Address	These values default to specify the entire range of memory for the device. Specify the memory range you want to program.
Program Memory	Select this option to program or verify the program memory range specified by the Start Address and End Address.
Configuration Bits	Select this option to program or verify the configuration bits. You can set the configuration bits using the Configuration Bits dialog.
ID Locations	Select this option to program the ID locations. You can set the ID Locations by clicking **Device ID** in the PRO MATE Device Programmer dialog.
EEPROM Data	For devices with data EEPROM, program the data memory from data in the EEPROM Memory window. Refer to the *MPLAB IDE User's Guide* for more information on data EEPROM and the EEPROM Memory window.

Calibration Memory	For devices with calibration memory, program the calibration memory from data in the Calibration Memory window. For more information on calibration memory, refer to Section 3.3.
Code Protect Settings	Shows the status and availability of code protection for the selected device.

When the programming is finished, "Success" or "Failure" will appear to the right of the Start Address. If the programming failed, an error window will appear showing the good (expected) data and the bad (actual) data for each address it attempted to program.

7.7 Read Device Dialog

To read selectively (e.g., part of program memory, only configuration bits), select *PRO MATE > Read Device* to open the Read Device dialog (Figure 7.6).
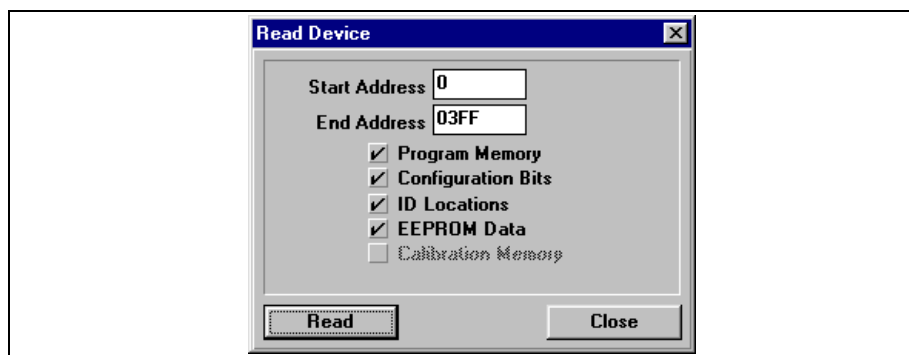
Figure 7.6: Read Device Dialog

Check the following settings, then click **Read** to read the device according to your settings. The memory area corresponding to the checked boxes will be read. Areas that are grayed out are not available on the device.

Start Address, End Address	These values default to specify the entire range of memory for the device. Specify the memory range you want to read.
Program Memory	Select this option to read the program memory range specified by the Start Address and End Address.
Configuration Bits	Select this option to read the configuration bits.
ID Locations	Select this option to read the ID locations.
EEPROM Data	For devices with data EEPROM, read the data in the data memory into the EEPROM Memory window. Refer to the *MPLAB IDE User's Guide* for more information on data EEPROM and the EEPROM Memory window.

Calibration Memory   For devices with calibration memory, read the data in calibration memory into the Calibration Memory window. For more information on calibration memory, refer to Section 3.3.

After reading a device into the MPLAB IDE, its data will appear in the Program Memory window. You can use the Modify dialog to change the data (*Window > Modify*), then save it to a hex file by selecting *File > Export > Export Hex File*.

7.8   PRO MATE II Menu Items

The PRO MATE menu items perform the following functions:

- **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. You can 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 specify the range and type of memory (e.g., program memory) areas on the target device to be programmed or verified. If the Verify process indicates several locations of bad data and those locations show as 0000, reseat the socket module and do a blank check.

- **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 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** – 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 the MPLAB IDE.

- **Erase Program Memory** – Sets all bits in the Program Memory window and EEPROM Data Memory window (if applicable) to '1'.

- **Erase Configuration Bits** – Sets all available configuration bits and ID location bits to '1'. If you reload your hex file or your project after using this menu item to erase configuration bits, the values in the Configuration Bits dialog will be updated to reflect the newly loaded configuration bit data in your hex file or code. You can select this menu item after you have loaded your hex file or rebuilt your project if you want to override the configuration bit values in your hex file or code.

- **Reset Voltages** – Sets V_{DD} Min, V_{DD} Max, and V_{PP} to their default values for the selected device.

- **Transfer to PRO MATE** – Transfers device information from the MPLAB IDE to PRO MATE II.

- **Transfer from PRO MATE** – Transfers device information from PRO MATE II to the MPLAB IDE.

- **Generate SQTP File** – Generate an SQTP file for device serialization.

- **Load SQTP File** – Chose the SQTP file you wish to use for serial programming.

- **Download PRO MATE Operating System** – Download the latest version of the PRO MATE II 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 PRO MATE Device Programmer Dialog, configuration bits, or IDs.

7.9   Files Used by PRO MATE II

PRO MATE II can use information directly from MPLAB projects without any intermediate steps. MPASM can be used separately from MPLAB to produce hex files for PRO MATE II. Alternatively, devices can be programmed with hex files from any PICmicro MCU-compatible cross-assembler or cross-compiler.

If you are using MPASM separate from the MPLAB IDE, or are generating hex files from within the MPLAB IDE for use later with PRO MATE II, you should use either INHX8M or INHX32 hex formats. MPASM's default output format for hex files is INHX8M.

If you are programming PIC17CXX devices, you should use INHX32 format. See the *MPASM User's Guide with MPLINK and MPLIB* for details on file formats.

7.10   Saving and Restoring Calibration Data

Special caution should be exercised while working with devices that contain calibration memory. These devices include:

- PIC12C508
- PIC12C508A
- PIC12C509
- PIC12C509A
- PIC14000
- PIC12C671
- PIC12C672
- PIC12CE519
- PIC12CE518
- PIC12CE673
- PIC12CE674

These parts are calibrated at the factory, with the calibration parameters stored in a special section of program memory. When a windowed (/JW) device is erased, this calibration data is erased along with the rest of program memory.

The calibration data is unique for each device. To ensure that each new windowed device operates correctly when the device is reprogrammed, the calibration data should be stored for each device before the device is initially programmed.

Note:	When working with calibration memory, it is important that the MPLAB IDE *not* be in Emulator mode. This ensures that the calibration data is correctly loaded from the device or file rather than from the emulator probe.

Before you begin, uniquely number each part with a small label.

7.10.1 Storing Calibration Data

Calibration data is encoded into each device when it is manufactured, and each device's calibration data is unique. When you erase a windowed device in order to re-program it, you also erase its calibration data. In order for the device to operate properly once it is reprogrammed, you must restore that individual device's calibration data.

Before you program a windowed device, save its calibration data to a file.

Follow these steps for each part.

1. Before you place your device in the PRO MATE II socket, select *PRO MATE > Erase Program Memory*. This sets all bits in the Program Memory, Data Memory, and Calibration Memory windows to '1'.
2. Insert the labeled part in the PRO MATE II.
3. Click **Read** in the PRO MATE Device Programmer dialog to read the device. If the socket module is not tightly fastened to the PRO MATE II, a message will indicate that the device is code-protected. Tighten the socket module and attempt to read the device again.

Figure 7.7: Code Protected Device Message

When the code protection warning no longer occurs and PRO MATE II reads the device, continue.
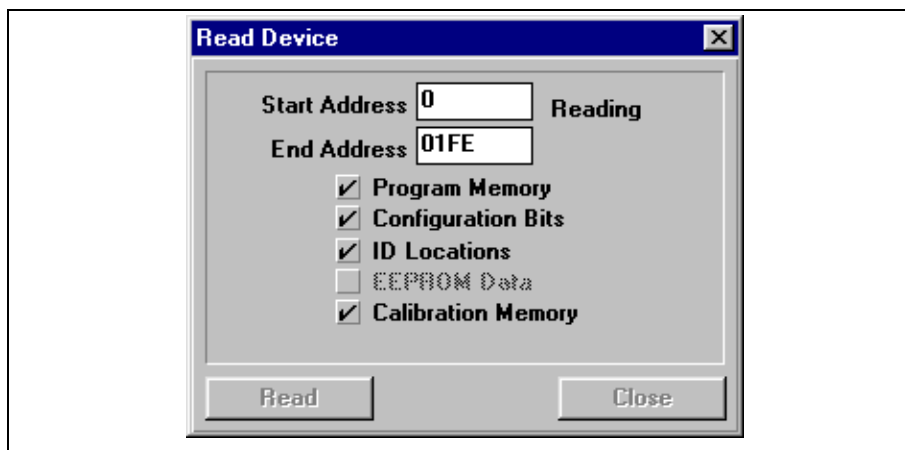
Figure 7.8: Reading the Device

4. Select _Window > Calibration Data_ to see the calibration information that was uploaded from the device to MPLAB. Click **OK**.
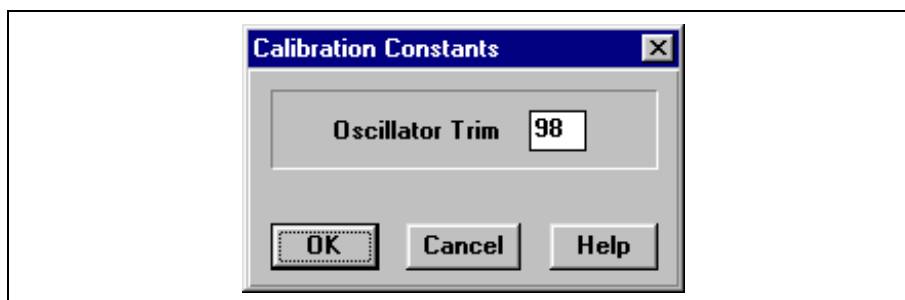
Figure 7.9: Calibration Data

5. Now that the device's calibration data is in memory, make a note of the setting and save it to a file for later use. Select _File > Export > Export Memory_.

    Check only the checkbox labeled Calibration memory. Clear the Program memory, Configuration bits, and IDs check boxes. Name the file so that you will be able to match the file to the part that is in the socket module now. Click **OK**.

PRO MATE II – MPLAB IDE Reference

Figure 7.10: Saving Calibration Data

If you have several new windowed parts that you want to program, repeat this procedure for each one, being careful to name each calibration hex file after the correct part.

7.10.2   Restoring Data After Erasure

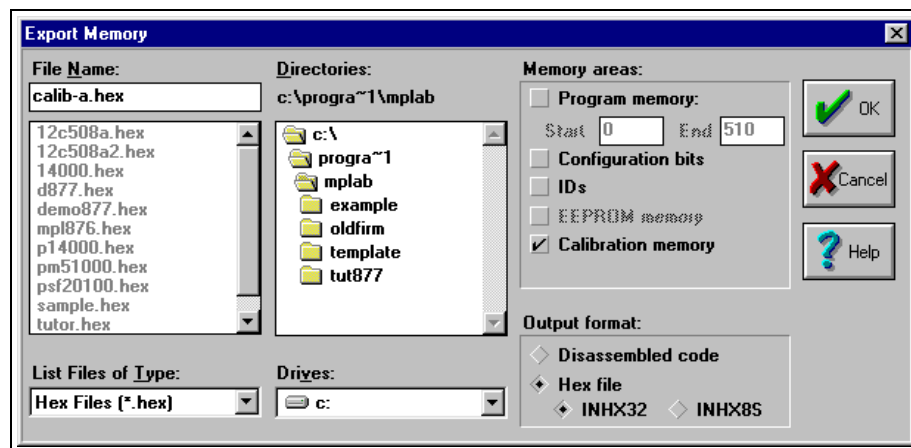In order to reprogram a windowed device, you must erase it in a UV eraser. This erases the entire device, including its calibration memory. In order to work properly, the calibration memory for that individual device must be restored by importing the calibration file you created for the chip earlier in this tutorial. You should label each device and save the calibration data for each device to a file before you program it.

To erase and reprogram a windowed part:

1. Remove the covering on the device's window. Place the part in a UV eraser. The amount of time required to completely erase a UV erasable device depends on: the wavelength of the light, its intensity, distance from UV source, and the process technology of the device (the size of the memory cells).

2. Place the erased part into the socket module.

3. Load the program file by opening your MPLAB project or selecting *File > Import > Import to Memory*. Be sure that your MPLAB project source file does not generate any program words in the calibration memory area.

4. Select *File > Import > Import to Memory* and select the saved calibration parameters file for the device.

© 2001 Microchip Technology Inc.   DS30082J-page 107
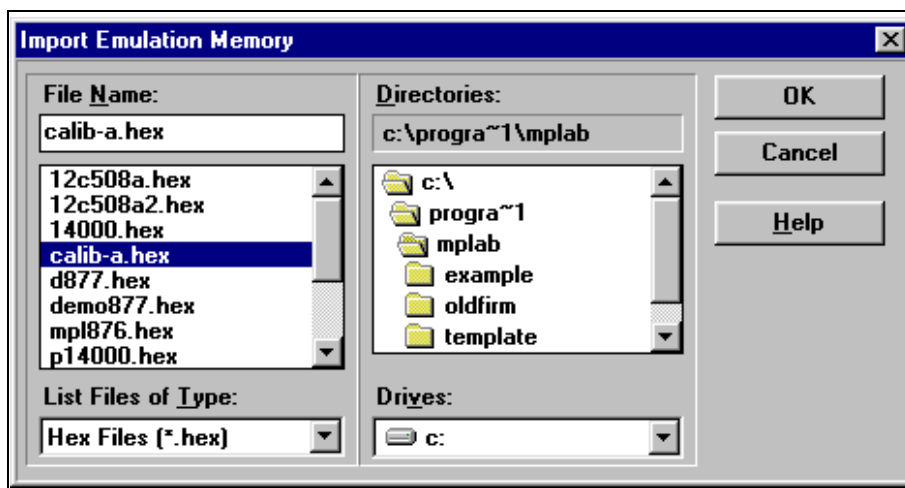
Figure 7.11: Restoring Calibration Data to an Erased Device

5. A default setting in the MPLAB IDE clears all memory on download. In order to preserve the device's memory (including its calibration memory), you must clear that option. Select *Options > Environment Setup*. Locate the Global Switches area in the **General** tab of the Development Mode dialog. Remove the check mark in the Clear Memory on Download check box.

6. Set the configuration bits as necessary. For details, see Section 7.4.

7. In the PRO MATE Device Programmer dialog, set the IDs as necessary. For details, see Section 7.3.

8. In the PRO MATE Device Programmer dialog, click **Program** to program the device.

> **Note:** If your calibration data was saved by a version of PRO MATE II earlier than v4.00, be sure to load the calibration data file before the program memory file.

7.11   Using Serial Programming

Serialization allows you to program a serial number into each microcontroller device that the Device Programmer programs. This number can be used as an entry code, password, or ID number.

Serialization is done by using a series of RETLW (Return Literal W) instructions, with the serial number bytes as the literal data. To serialize, you must first generate a serialization file, and then use that file to serialize locations in the device microcontroller:

1. Generate an SQTP file by selecting *PRO MATE > Generate SQTP File* and filling in the dialog.

2. Activate SQTP by clicking **SQTP File** in the PRO MATE Device Programmer dialog.

3. Program the device by clicking **Program** in the PRO MATE Device Programmer dialog.

7.11.1   Generate SQTP File

Select *PRO MATE > Generate SQTP File* to display a dialog box for generating an SQTP file. Fill in the values for the type of SQTP file you are generating, then click **Generate**. For example, an SQTP file for a PIC16C5X device might be generated as shown:
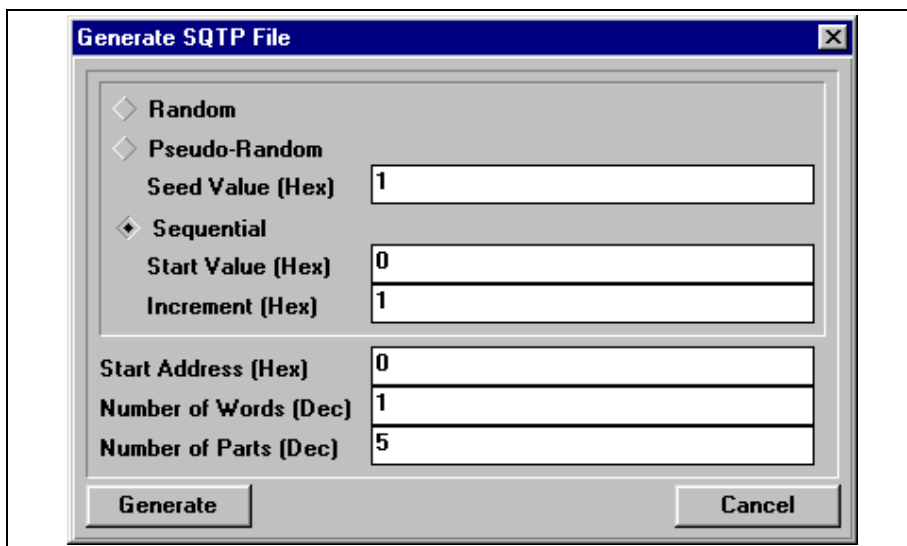

Figure 7.12: Sample Serial SQTP File

|	
Random	Generates unique, random numbers for each part.
Pseudo-Random	Generates a pseudo-random set of non-repeating numbers based on the seed value you specify.
Sequential	Generates sequential numbers based on the starting value you specify and incrementing each number by the amount you specify.
Start Address (Hex)	Specify the starting address to which the serial number is to be programmed.
Number of Words (Dec)	Specify the size of each serial number. Be sure to specify a large enough serial number for the number of parts you plan to program using this file.
Number of Parts (Dec)	Specify the number of parts to be programmed (serialized) using this file. The file will contain a line for each part.

For information on the format of SQTP files, refer to Section 7.11.4.

7.11.2   Enabling Serialization

Click **SQTP File** from the PRO MATE Device Programmer dialog and select a file to enable serialization for the current programming session.
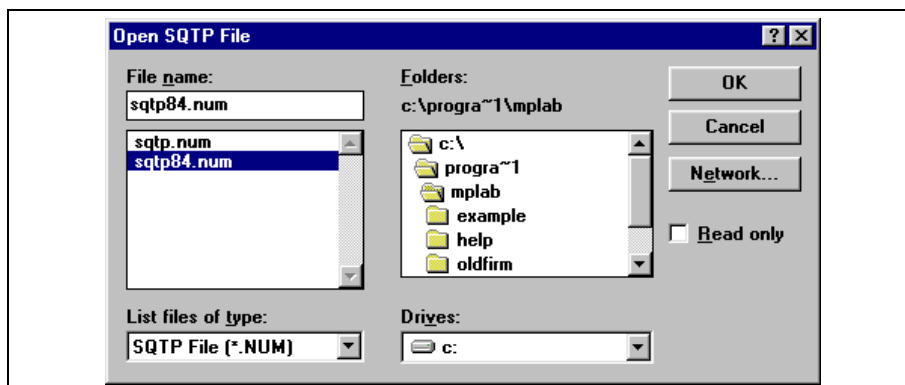
Figure 7.13: Open SQTP File Dialog

After you enable serialization, the serial number that will be programmed into the next device can be seen at the program memory address you specified when you generated the SQTP file. Look for the RETLW instruction in the Program Memory window (RETLW's opcode is 34 hex).

7.11.3 Programming SQTP Devices

To program a device with the SQTP information, simply enable SQTP and program the device normally. After the device is programmed, the Program Memory window will display the next serial number. If the last serial number in the file was used, a message will appear and serialization will be disabled.

When a serial number is used, the SQTP file is marked by replacing the colon for that entry by a semicolon so that PRO MATE II will skip that line during later programming sessions. To use the same SQTP file over multiple programming sessions without repeating any numbers, generate an SQTP file with many more parts than you'll program in a single session. For example, if you generate an SQTP file for 10,000 parts and program 1,000 parts in each session, you can use the file ten times.

7.11.4   Using Hexadecimal Record Formats

The following hexadecimal record format discussion provides the proper file format for the PICmicro device families. Make sure your assembler or compiler is configured to generate hex files in the proper format.

PRO MATE II uses the formats described in the following paragraphs as follows:

Device	Hex Format
PIC12CXXX	INHX8M
PIC14000	INHX8M
PIC16CXXX	INHX8M
PIC17CXXX	INHX32
PIC18CXXX	INHX32
Firmware Downloads	INHX32

Each hexadecimal data record has the following format:

:BBAAAATTHHHH....HHCC

:	Start Character (prefix)
BB	Two-Digit Byte Count specifying number of data blocks in record
AAAA	Four-Digit Starting Address of data record
TT	Two-Digit Record Type
	00 = Data Record
	01 = End-of-File Record
	02 = Segment Address Record
	04 = Extended Linear Address Record (INHX32)
HHHH....HH	Two-Digit Data Blocks
CC	Two-Digit Checksum—Two's complement of sum of all preceding bytes in data record except the colon.

INHX8M

The data record is output as described above.

INHX32

The extended linear address record is output to establish upper 16 bits of data address.

7.12   Upgrading the PRO MATE II Operating System

You might need to upgrade the PRO MATE II operating system if:

- PRO MATE II does not recognize your socket module or device type
- The PRO MATE II display reads "Ready for download..."
- The MPLAB IDE tells you that there is a newer version of the PRO MATE operating system when you attempt to enable PRO MATE II

The updated version of the operating system is in the MPLAB directory. You can also obtain the latest version of the PRO MATE operating system from our web site at www.microchip.com. Click Development Tools and find PRO MATE II listed under Programmers. Near the bottom of the PRO MATE II page you will find the latest operating system update.

To upgrade the PRO MATE II operating system, follow the steps below:

1. Make sure PRO MATE II is disabled. If Disable Programmer is visible on the top of the PRO MATE menu, select it now.
2. Turn the power switch on the back of the PRO MATE II to the OFF position.
3. While holding down the <F1> and <F3> keys on the PRO MATE II, turn the power switch on the back of the PRO MATE II to the ON position.

> **Note:**   This step erases the existing PRO MATE II operating system. In order for your programmer to function, you must complete this procedure.

4. Watch the PRO MATE II LCD. You should see the following messages on the PRO MATE II LCD: "Clearing Memory Now" and "Ready for Download." If you do not, repeat steps 1 through 3.
5. Once the PRO MATE II LCD indicates that it is ready for the download, select one of the following from the MPLAB menu bar:

    - *PRO MATE > Download PRO MATE PIC Operating System*
    - *PRO MATE > Download PRO MATE HCS Operating System*
    - *PRO MATE > Download PRO MATE SEE Operating System*

6. A standard file selection dialog box will appear in the MPLAB IDE. Choose an operating system to download and click **OK**. The file selection dialog uses an appropriate file name mask so only files specific to the chosen operating system will be displayed. Examples of the file you may download are `pic51015.hex`, `hcs51015.hex` and `see51015.hex`. Multiple versions of the selected operating system will be listed if they exist in the MPLAB installation directory.
7. A dialog will now appear to prompt you to select the target programmer and to check that the target programmer is ready for download.

8. "Downloading operating system to PRO MATE: nn% Completed" is displayed and continually updated in an MPLAB dialog box, where "nn" is a number between 0 and 100. "Downloading Now" is displayed on the PRO MATE II LCD, along with an activity indicator in the top right corner of the LCD. Downloading may take a couple of minutes.

9. PRO MATE II performs a calibration when download is complete. This will take approximately 30 seconds.

Once the download is done, PRO MATE II will automatically restart, as if from power-on.

> **Note:** For PRO MATE, "Complete! Press a key to begin" appears on the LCD. Press one of the four function buttons to continue. The PRO MATE II does not require a button to be pressed.

PRO MATE II User's Guide

NOTES:

Chapter 8. PROCMD Reference

8.1 Introduction

In order to run under DOS, the PRO MATE II device programmer must be connected to a PC that has the PROCMD software installed. For installation information, see Chapter 2. This chapter covers the details of using the command line program PROCMD to control PRO MATE II operations.

8.2 Highlights

Topics covered in this chapter:

- Command Line Interface Description
- Error Descriptions

8.3 Command Line Interface Description

The following commands are available in the command line interface.

Table: 8.1 Command Line Interface Description

Command	Syntax	Valid Values	Required?
Baud Rate	/b<baud>	9600,19200(default)	No
Download Operating System	/d<filename>	any valid filename	No
Target Hex File	/f<filename>	any valid filename	No
Program	/m		No
V$_{DD}$ Min	/n<voltage>	specified in mV; must be within device spec of 125mV resolution	No
Device	/p<partname>	e.g., 16C54A, 24AA01	Yes, unless download-ing operating system
Quiet Mode	/q		No
Enable Safe Mode	/r		No
SQTP File	/s<filename>	any valid filename	No
V$_{PP}$	/v<voltage>	specified in mV; must be within device spec of 125 mV resolution	No
V$_{DD}$ Max	/x<voltage>	specified in mV; must be within device spec of 125 mV resolution	No
Verify	/y		No
COM Port	/<number>	1, 2, 3, or 4	Yes
Help	/?		No

Command line parameters can appear in any order and are not case sensitive. If no parameters are specified, the help screen will be displayed.

Checks will be made to ensure that valid values are specified for command line options. Voltages must be specified in millivolts, and must be given in 125 mV steps.

> **Note:** PRO MATE only supports a 250 mV step.

You can set up the device programmer, program a device with the currently loaded contents of the device programmer, or completely specify the contents of a device to be programmed.

8.4   Error Descriptions

PROCMD returns the following error level values:

Table 8.1: PROCMD Error Descriptions

Error Level	Description	Possible Cause
0	Operation Successful	Function proceeded with no errors.
1	PRO MATE not found	Could not establish communication with PRO MATE. Possible causes are that the: 1. PRO MATE II serial cable is not attached. 2. PRO MATE II is not powered up. 3. PRO MATE II is not connected to the selected communications port.
2	Wrong socket module	The selected device is not supported by the socket module currently installed in PRO MATE.
3	File not found	A specific operating system, program memory, or SQTP file was not found.
4	Illegal device specified	The specified device is either not supported by PRO MATE II or is not supported by the command line interface to PRO MATE II. HCS devices and Smart Serial EEPROMs are not supported by the command line interface.
5	Illegal voltage specified	A specified voltage is outside of the valid range for the selected device, V_{DD} Max is less than V_{DD} Min, or a specified voltage is not an even multiple of 125 mV (for 125 mV steps).

Error Level	Description	Possible Cause
6	Program/verification error	An unspecified error was found while programming or verifying the device. Usually, a specific error informing the user what area of memory failed will be returned.
7	Download operating system must be done independently	Downloading operating system cannot be combined with any other operation other than selecting the communications port, which is required.
8	Illegal parameter	An invalid command line parameter was encountered.
9	Illegal hex file	An invalid hex record was found, or the file is not in valid INHX8M or INHX32 format.
10	Illegal SQTP file	An invalid record was found in the SQTP file, or the file is not a file in the INHX8M format.
11	No command on command line	No command line parameters were found.
12	No communications port specified	The desired communications port must be specified with the /# option.
13	Invalid communications port specified	The selected communications port could not be found or initialized.
14	No device specified	Unless downloading operating system, the target device must be specified with the /p option.
15	Device file not found	The file MPLAB.DVS must be located in the same directory as the PRO MATE II command line interface executable. This file is installed with the PRO MATE II installation.
16	Out of Memory	There was not enough memory available on the PC to allocate buffers for the device's memory.
17	Invalid device files	The device file that was located is of an incompatible version. Reinstall the application.

Error Level	Description	Possible Cause
18	Incompatible operating system version	The version of operating system currently in the PRO MATE Device Programmer is not compatible with the software. Download the version of operating system that was installed with the software, or obtain the latest version from the Microchip internet website.
19	Device not supported by operating system	The version of operating system currently in the PRO MATE Device Programmer does not support the selected device. Download the version of operating system that was installed with the software, or obtain the latest version from the Microchip internet website.
20	Communications error	An error occurred while communicating with the PRO MATE Device Programmer. Verify that the Device Programmer is powered on and all serial connections are firm.
21	No command line support	The selected device is not supported by the command line version. These devices must be programmed by the full interface version. Examples of these devices are the security devices and smart serials.
22	No SQTP number available	There are no available SQTP numbers in the specified file.
23	Cannot mark SQTP line as used	The SQTP line just read cannot be marked as used. Be sure the file is not read only.
24	Programming error - program memory	A programming or verification error occurred when programming the device's program memory.
25	Programming error - data memory	A programming or verification error occurred when programming the device's data memory.
26	Programming error - configuration bits	A programming or verification error occurred when programming the device's configuration bits.
27	Programming error - IDs	A programming or verification error occurred when programming the device's IDs.

Error Level	Description	Possible Cause
28	Parity error	A parity error occurred during programming or verifying. This often occurs when over programming a parity device. If using a windowed device, erase the device and reprogram.
29	Help shown	If the help screen is shown, no other commands are processed.
30	Missing file specification	A command line parameter requiring a file specification did not specify a file.
31	Internal error	This error should never be seen. It can occur as a result of erroneous communication between the device programmer and the PC.

PRO MATE II User's Guide

NOTES:

Chapter 9. PRO MATE II Stand-Alone Mode Reference

9.1 Introduction

This chapter tells you how to use the PRO MATE II device programmer in stand-alone mode. The device programmer provides an LCD interface that gives you complete control over a programming session.
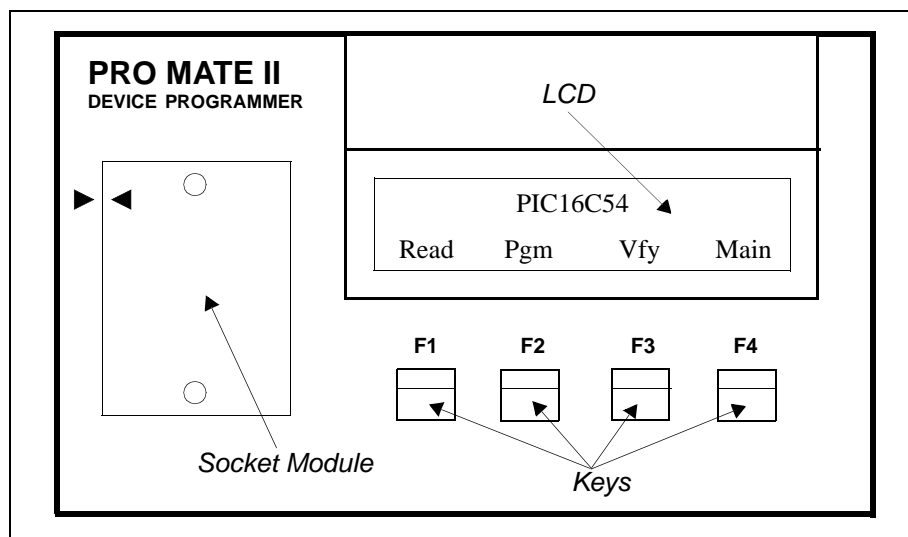
9.2 Highlights

Topics covered in this chapter:

- PRO MATE II LCD and Keys
- Operational Overview
- Start-Up Sequence and Select Menu
- Power-On with <F1> and <F3> Pressed
- Main Menu
- Command Menu
- Utilities Menu

9.3 PRO MATE II LCD and Keys

The front panel of the programmer should look like Figure 9.1.

Figure 9.1: PRO MATE II Front Panel

9.4   Operational Overview

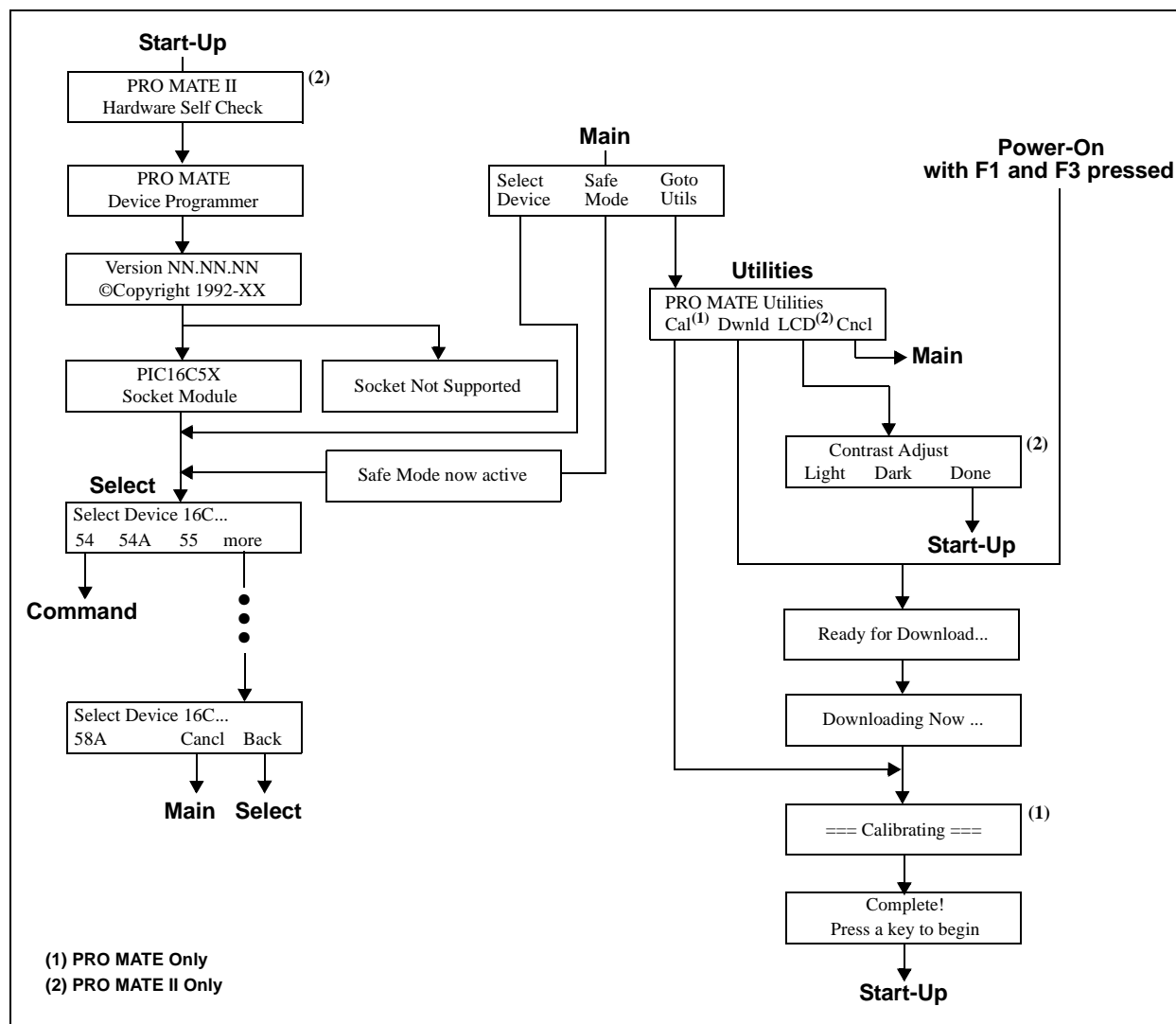**Figure 9.2: PRO MATE Menu Tree – PIC16C5X**

9.5   Start-Up Sequence and Select Menu

When power is applied to the PRO MATE II programmer, it should begin a start-up sequence of LCD displays similar to those shown in Figure 9.2. If you do not get a similar start-up sequence, see Appendix B: Troubleshooting.

> **Note:**   If you need to update the operating system, you will have to connect to a PC with MPLAB running to accomplish this.

Once the start-up sequence has completed, you will see the Select menu. Choose a device from the list by pressing a key. If you do not see your device listed, press the key corresponding to "more". You may keep pressing this key for more device options until you reach the last display, which will have "Cancl" and "Back" on the menu. Pressing the key for "Back" will take you to the first display of the Select menu. Pressing "Cancl" will take you to the Main menu.

When you have selected your desired device, you will exit the Select menu and go to the Command menu.

9.6   Power-On with <F1> and <F3> Pressed

When power is applied to the PRO MATE II programmer at the same time that <F1> and <F3> are pressed, the programmer will immediately go to the "Ready for Download" display from the Utilities menu (Figure 9.2). Use this feature when you wish to update the PRO MATE II operating system.

> **Note:**   If you need to update the operating system, you will have to connect to a PC with MPLAB running to accomplish this.

If you see the "Ready for Download" display when applying power, but do not have <F1> and <F3> pressed, refer to Appendix B, Section B.4.3.

9.7   Main Menu

The Main menu gives you three options: select/reselect the device to program, operate in safe mode or execute the utilities.

You can select or reselect the device to be programmed by pressing the key under "Select Device" to go to the Select menu.

You can choose to operate in safe mode by pressing the key under "Safe Mode". In safe mode, only Program and Verify options on the Command menu are available (see Section 6.5 for more information). After safe mode is implemented, you will be sent to the Select menu.
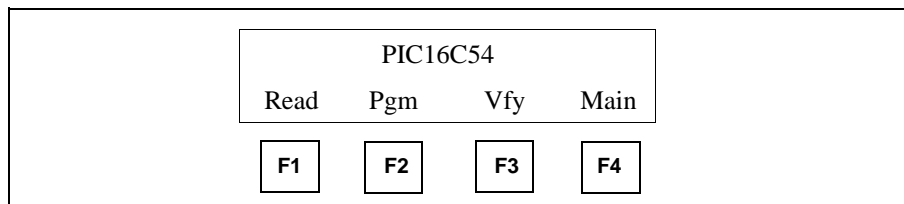
You may run available utilities by selecting "Goto Util" to go to the Utilities menu.

9.8 Command Menu

After you select a device, the device programmer displays the Command menu (Figure 9.3). The functions on the Command Menu allow you to perform the basic user tasks for programming a microcontroller device: Program, Verify, and Read. Main will return you to the main menu.

```
┌─────────────────────────────────────────────┐
│              PIC16C54                         │
│                                               │
│   Read       Pgm       Vfy       Main         │
│                                               │
│  ┌────┐    ┌────┐    ┌────┐    ┌────┐          │
│  │ F1 │    │ F2 │    │ F3 │    │ F4 │          │
│  └────┘    └────┘    └────┘    └────┘          │
└─────────────────────────────────────────────┘
```

Figure 9.3: Command Menu - PIC16C5X

9.8.1 Program (Pgm, F2)

This command programs the device in the socket module with the contents of the internal memory of the device programmer.

The device programmer checks to see if the installed microcontroller device is blank. If the device is not blank, the device programmer asks if you want to continue. Answer "Yes" to continue. Answer "No" to return to the Command menu.

The device programmer programs the contents of its memory into the microcontroller device loaded in the socket module.

After programming a device without errors, the device programmer performs a check to verify the data programmed into the device, and returns the results of the verification. For the installed device, the device programmer performs the verification at the V_{DD} Minimum and V_{DD} Maximum voltages.

The device programmer reports programming errors and verify errors according to where the error occurred. Errors are reported for program, configuration bits, and ID locations.

After programming, the device programmer displays the checksum.

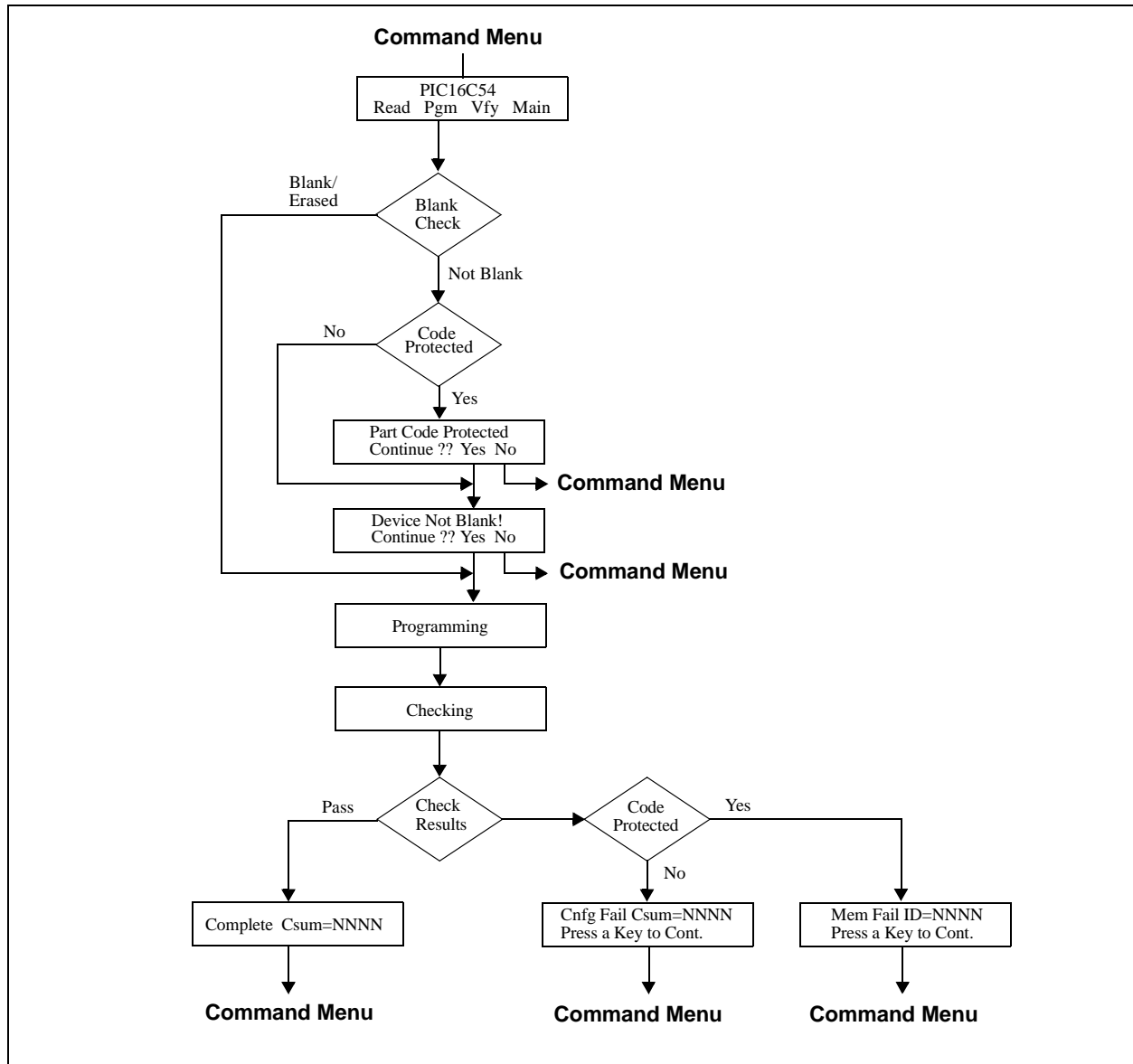**Figure 9.4: Program Menu Tree – PIC16C5X**

9.8.2 Verify (Vfy, F3)

The device programmer compares the contents of its internal memory to the contents of the programmed microcontroller device loaded in the socket module. If the data and configuration bit settings are correct, "VERIFIED" will display on the LCD. The device programmer performs the verification at the V$_{DD}$ Minimum and V$_{DD}$ Maximum voltages. The device programmer reports errors according to which part of the device failed.

The Verify function also confirms that erased parts are blank. If all programmable locations are blank for a device loaded in the socket module, the device programmer displays ERASED. Reported results include the following:

- Same Contents
- Blank Device
- Errors
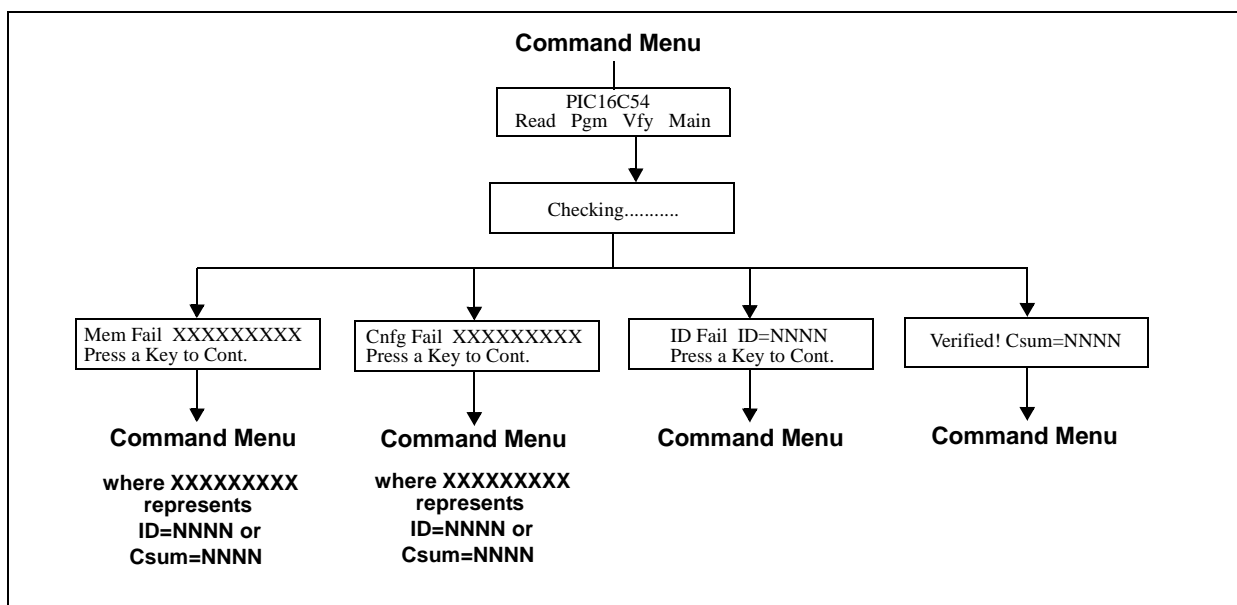
Figure 9.5: Verify Menu Tree – PIC16C5X

9.8.3 Read (Read, F1)

Press **Read** to copy the contents of the device in the socket module into the internal memory of the device programmer.

For PIC16CXX devices, the device programmer will ask the question: "Code Protect Parts?" Answer "Yes" to code protect the parts that you will be programming. Code protection will remain effective until a new device is read. Answer "No" to program devices exactly as read.

After reading a device, the device programmer displays a checksum. If the device is code protected, a code protection message will be displayed prior to the read. Answer "Yes" to continue. Answer "No" to return to the Command menu.
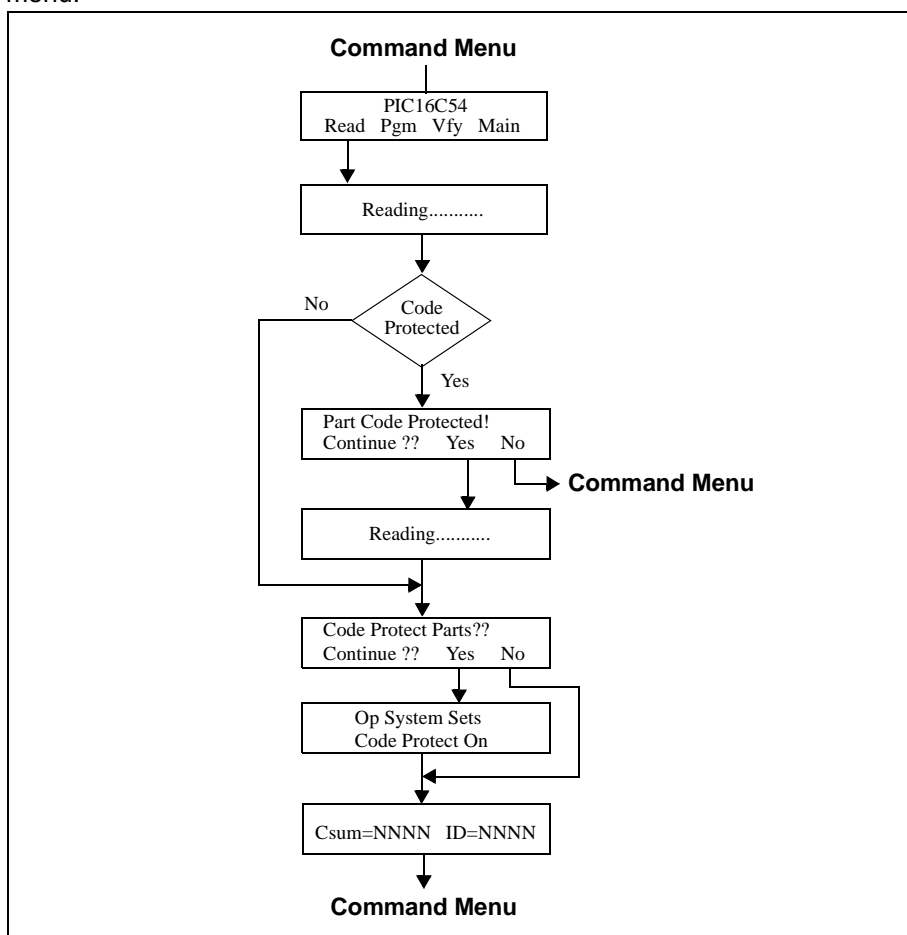

Figure 9.6: Read Menu Tree – PIC16C5X

9.8.4 Main (Main, F4)

Returns to the Main menu.

9.9   Utilities Menu

PRO MATE II includes various utilities, accessible through the Utilities menu (Figure 9.7). Not all utilities are available for all models. Calibration is only available for PRO MATE (1), and LCD contrast adjust is only available for PRO MATE II (2).
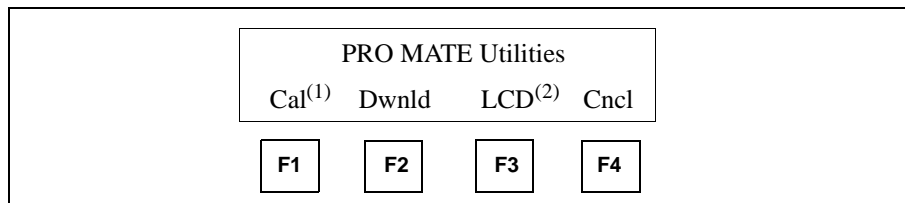
```
                    PRO MATE Utilities

           Cal⁽¹⁾   Dwnld      LCD⁽²⁾  Cncl

           ┌────┐  ┌────┐    ┌────┐  ┌────┐
           │ F1 │  │ F2 │    │ F3 │  │ F4 │
           └────┘  └────┘    └────┘  └────┘
```

Figure 9.7: Utilities Menu

9.9.1   Calibrate (Cal, F1)

Performs a calibration on the PRO MATE device programmer.

Calibrates the internal voltage generators (V$_{DD}$ and V$_{PP}$). After calibration is complete, the device programmer displays the message, "Complete! Press a key to begin." Pressing a key will take you to the Start-up sequence.

> **Note:**   In a production environment, calibrate the device programmer each week or after changing the power supply.

> **Caution:** Do not calibrate the device programmer while a device is installed in the socket module. Applied voltages will exceed the maximum ratings of all parts and will damage the device.

9.9.2   Download (Dwnld, F2)

Sets up the device programmer for downloading a new operating system.

> **Note:**   If you need to update the operating system, you will have to connect to a PC with MPLAB running to accomplish this.

Select Dwnld to begin downloading a new version of the PRO MATE II operating system into the on-board memory. "Ready for Download" will display.

After this message displays, execute the download command from MPLAB menu, _PRO MATE > Download Operating System_. When downloading starts, the LCD will show "Downloading from PC...," and an activity indicator will display on the last location of the second display line.

After download is complete, the device programmer displays the message, "Complete! Press a key to begin." Pressing a key will take you to the Start-up sequence.

> **Note:** The device programmer performs a calibration after each download for the PRO MATE device programmer.

9.9.3   LCD Contrast Adjust (LCD, F3)

The Contrast Adjust control (available with PRO MATE II hardware only) allows you to make the LCD display lighter or darker.

Press the key under Light to make the LCD display lighter. Press the key under Dark to make the LCD display darker. Press the key under Done when finished. Done will take you to the Start-up sequence.

9.9.4   Cancel (Cncl, F4)

Select Cncl to return to the main menu.

PRO MATE II User's Guide

NOTES:

Appendix A. Hardware Specifications

A.1   Introduction

This appendix describes PRO MATE II to PC connections, programmer hardware, and socket module hardware.

A.2   Highlights

Contents of this appendix:

- Connecting to a PC via the Serial Port
- Programmer Specifications
- Socket Module Specifications

A.3   Connecting to a PC via the Serial Port

The PRO MATE II interfaces to the PC via a standard RS-232 port with a DB9 connector. The unit defaults to a 19.2K baud rate. The user can alternately choose a 9.6K baud rate by depressing <F2> during power-up.

Serial port communications is as follows:

- 19.2K (default) or 9.6K baud
- 8 data bits, 1 stop bit, no parity
- Hardware flow control

The following table gives the data for connecting the PRO MATE II programmer (9-pin straight-through) to a 25-pin serial port. Connect the corresponding terminals indicated on each line of the table.

Table A.1: PC Host to PRO MATE II Signals

25-pin (PC-Host)			9-pin (PRO MATE II)	
2	TX	←⎯→	2	RX
3	RX	←⎯→	3	TX
20	DTR	←⎯→	4	DTR
7	Common	←⎯→	5	Common
6	DSR	←⎯→	6	+5 volts (pulled up)
4	RTS	←⎯→	7	CTS
5	CTS	←⎯→	8	RTS

A.4   Programmer Specifications

This section discusses the following topics:

- Physical Dimensions and Layout
- Power Input
- Device Supply Voltage (V$_{DD}$)
- Programming Voltage (V$_{PP}$)
- Device Drivers

> **Note:** Values are specified under normal operation and represent recommended operating conditions. Absolute maximum ratings have not been published at this time.

A.4.1   Physical Dimensions and Layout

The PRO MATE II device programmer has the physical dimensions of 9.40" x 6.90" x 1.36". Top and side views of the programmer are shown in Figure A.1 and Figure A.2, respectively.
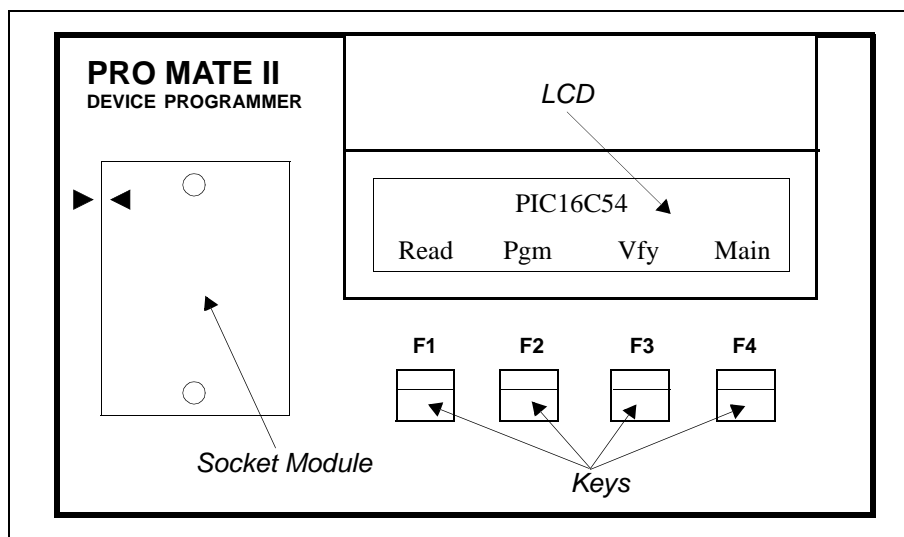
Figure A.1: PRO MATE II Top View
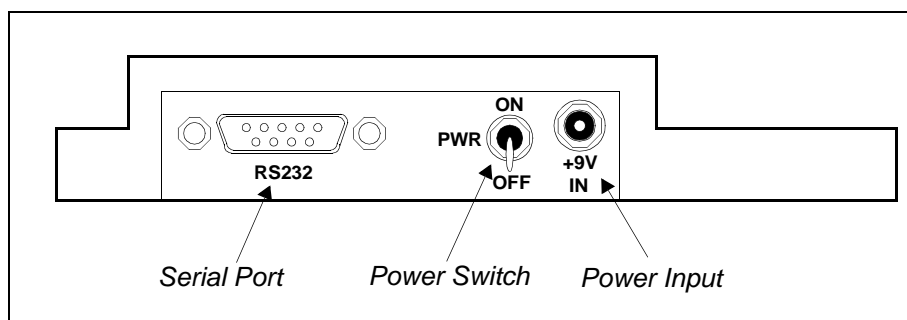
Figure A.2: PRO MATE II Side View

A.4.2   Power Input

The PRO MATE II requires a +9V DC input with 750mA capacity. The power connector is a 2.5mm DC power jack (Switchcraft, P/N RAPC-712).

A 0.75A fuse (Littlefuse Nano^2 SMF, P/N 451.750) is located near the +9V input on the PRO MATE II motherboard for circuit protection.

WARNING	
	The outer housing of the DC power jack is chassis ground and the inner pin is +9V. If a DC power jack is used with reverse polarity, the fuse will blow.

Specification summary:

- +9V DC, 750mA center positive
- Maximum total power: 6.8W

A.4.3   Device Supply Voltage (V$_{DD}$)

The device supply voltage (V$_{DD}$) has a range of 2.0 to 6.5V +/- 0.08V, providing voltage tolerances as shown in Table A.2.

Table A.2: V$_{DD}$ Voltage Tolerances

V$_{DD}$ Voltage	+/- 0.08V
2.0V	4.0%
3.0V	2.7%
5.0V	1.6%
6.5V	1.3%

Specification summary:

- DC output voltage: 2 to 6.5V
- Maximum output current: 60 mA
- Maximum overshoot/undershoot: 3.5%
- Adjustable voltage increment: 40mV
- Voltage accuracy: +/-0.08V

A.4.4   Device Programming Voltage (V$_{PP}$)

The device programming voltage (V$_{PP}$) has a range of 10.0 to 15.0V +/- 0.16V, providing voltage tolerances as shown in Table A.3.

Table A.3: V$_{PP}$ Voltage Tolerances

V$_{PP}$ Voltage	+/- 0.16V
10.0V	1.6%
12.0V	1.4%
13.5V	1.2%
15.0V	1.1%

Specification summary:

- DC output voltage: 10 to 15V
- Maximum output current: 60 mA
- Maximum overshoot/undershoot: 3.5%
- Adjustable voltage increment: 80mV
- Voltage accuracy: +/-0.16V

A.4.5   Device Drivers

VDD level signals are provided to the data and clock pins of the device to be programmed. Also, the system is able to read the data pins of the device at VDD levels. There are 32 I/O VDD pin drivers.

The pin drivers are active devices, capable of driving the device pin low or high. Each driver can drive the device to be programmed at the required voltage (VDD), one CMOS load per driver. The device drivers provide a DC output voltage of 0 to 6.5V, with a maximum rise time of 200 nsec. Maximum overshoot/undershoot is less than 5%. DC characteristics are as follows:

Table A.4: DC Characteristics

Symbol	Parameter	VDD	Limit	Unit
VOH	Minimum High-level Output Voltage	2.0	1.9	V
		4.5	4.4	
		6.0	5.9	
VOL	Maximum Low-level Output Voltage	2.0	0.1	V
		4.5	0.1	
		6.0	0.1	

Rise time of the pin drivers is on the order of 100 nsec, with a propagation delay of less than 200 nsec from the end of the write pulse (measurements taken at the socket module connector, over the full voltage range of 2.0 to 6.5 V).

Specification Summary:

- DC output voltage: 0 to 6.5V

- Maximum resistive load on output: 100 kOhms

- Maximum capacitive load on output: 1000 pF

- Minimum high-level output voltage: 1.9V

Note: Minimum high-level output voltage (VOH) will be no less than 0.1V below VDD supply voltage.

- Maximum low-level output voltage: 0.1V

- Maximum rise time: 200nsec

Note: Rise time is the time the transient spends between 10 and 90% of its final value (VOH).

- Maximum fall time: 200nsec

- Maximum overshoot/undershoot: 3.5%

A.5   Socket Module Specifications

This section discusses the following topics:

- PRO MATE II Socket Modules
- Compatibility with PRO MATE Socket Modules
- Socket Life Expectancy and Cleaning Procedures
- ICSP Device Support Switch Settings

A.5.1   PRO MATE II Socket Modules

The PRO MATE II utilizes socket modules to accommodate various Microchip devices. PRO MATE II socket modules have alignment holes in each corner which fit over the four brass alignment pins located on the two surface mount (SMT) interface connectors of the system board (Figure A.3).
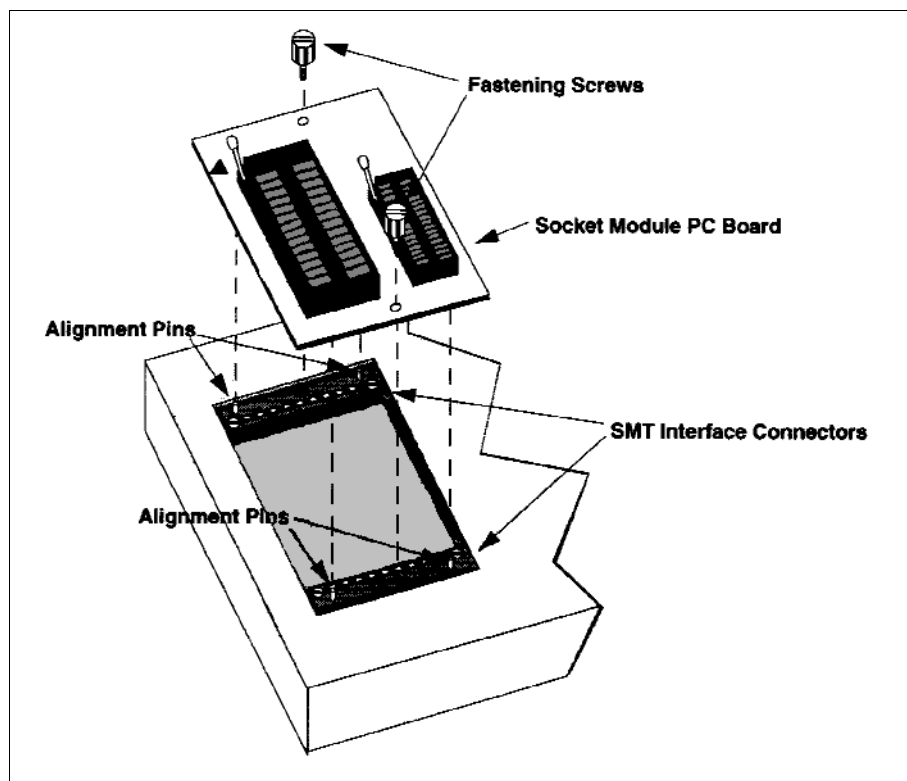
Figure A.3: Socket Module Alignment

The socket module interface consists of a generic dual-row PCB pad array having the dimensions shown in Figure A.4 and signal connections shown in Figure A.5.

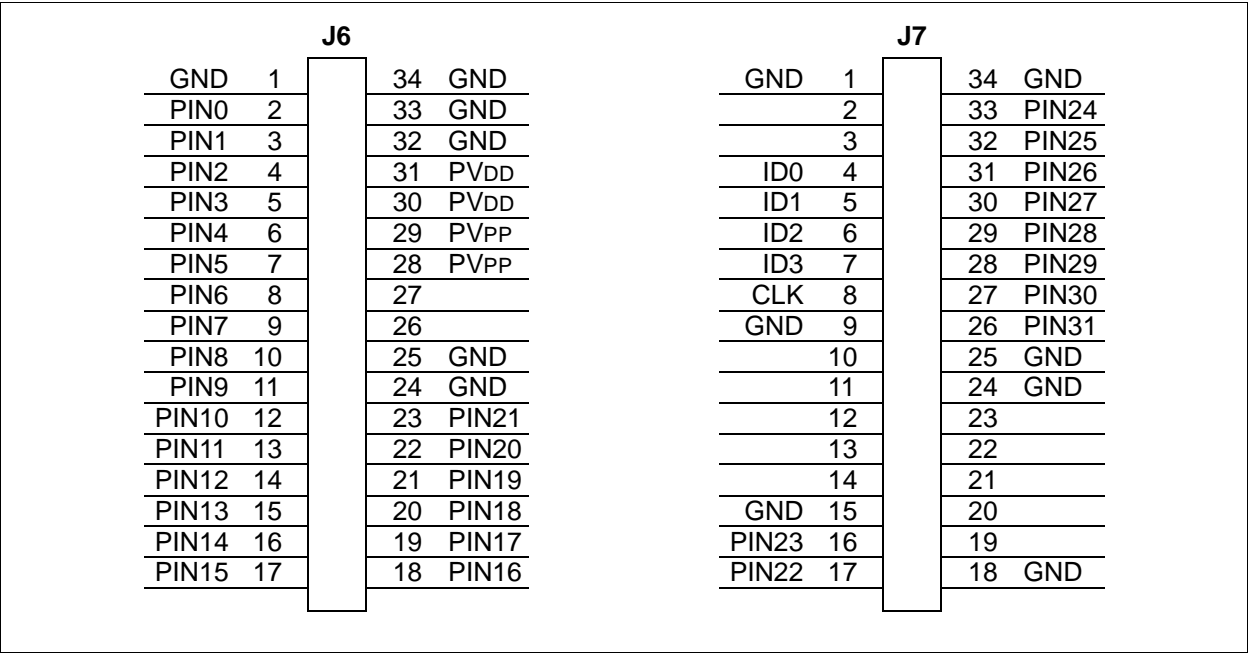| WARNING	Socket modules should be connected or removed with the power turned off. Attempts to do so with the power turned on may result in damage to the PRO MATE II. Make sure that the socket module is properly aligned before applying power to PRO MATE II.
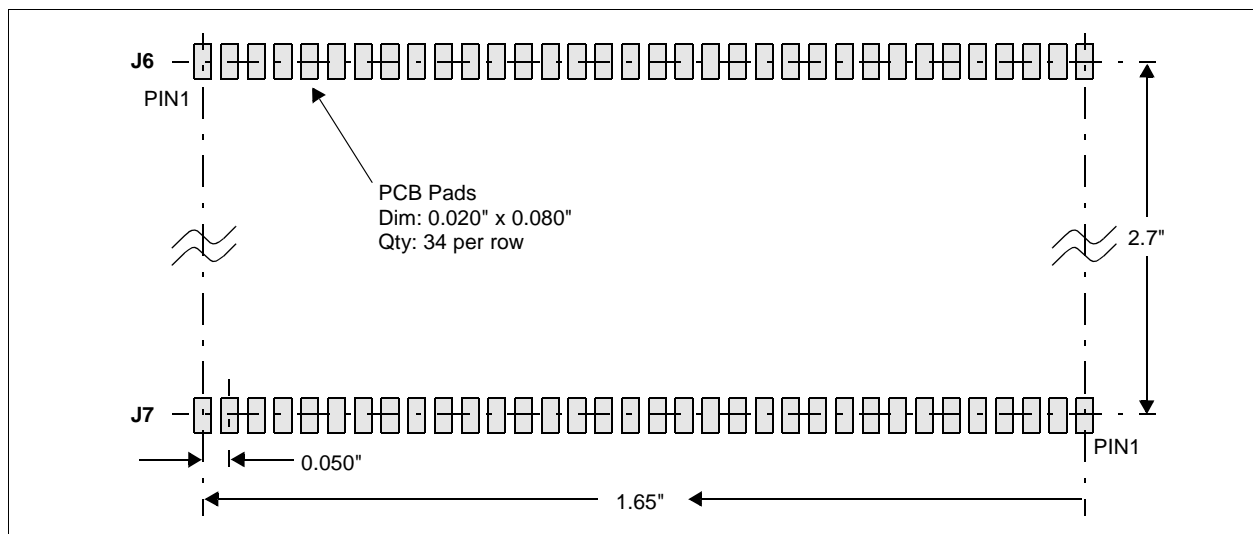
Figure A.4: Module PCB Pad Interface

PRO MATE II User's Guide

```
                    J6                                          J7
    GND     1  ┌────────┐  34   GND          GND     1  ┌────────┐  34   GND
    PIN0    2  │        │  33   GND                  2  │        │  33   PIN24
    PIN1    3  │        │  32   GND                  3  │        │  32   PIN25
    PIN2    4  │        │  31   PVDD         ID0     4  │        │  31   PIN26
    PIN3    5  │        │  30   PVDD         ID1     5  │        │  30   PIN27
    PIN4    6  │        │  29   PVPP         ID2     6  │        │  29   PIN28
    PIN5    7  │        │  28   PVPP         ID3     7  │        │  28   PIN29
    PIN6    8  │        │  27                CLK     8  │        │  27   PIN30
    PIN7    9  │        │  26                GND     9  │        │  26   PIN31
    PIN8   10  │        │  25   GND                 10  │        │  25   GND
    PIN9   11  │        │  24   GND                 11  │        │  24   GND
    PIN10  12  │        │  23   PIN21              12  │        │  23
    PIN11  13  │        │  22   PIN20              13  │        │  22
    PIN12  14  │        │  21   PIN19              14  │        │  21
    PIN13  15  │        │  20   PIN18       GND    15  │        │  20
    PIN14  16  │        │  19   PIN17       PIN23  16  │        │  19
    PIN15  17  └────────┘  18   PIN16       PIN22  17  └────────┘  18   GND
```

Figure A.5: Module Interface Pin-Out

A.5.2   Compatibility with PRO MATE Socket Modules

PRO MATE socket modules are pin-compatible with PRO MATE II socket
modules. However, they do not have the four alignment holes discussed in
the previous section. Therefore, you must use a different set of surface mount
(SMT) interface connectors that do not have the brass align pins to use PRO
MATE socket modules on the PRO MATE II.

> **WARNING**
>
> Do not attempt to connect a PRO MATE socket module
> to the PRO MATE II without changing the SMT interface
> connectors, or damage to the connectors may result.

To change the PRO MATE II SMT interface connectors to PRO MATE SMT
interface connectors:

- Remove the four screws holding the PRO MATE II SMT connectors to
  the programmer

- Lift the PRO MATE II SMT connectors off of the programmer

- Place the PRO MATE SMT connectors in the same location on the pro-
  grammer

- Secure the PRO MATE SMT connectors to the programmer using the
  four screws

A.5.3   Socket Life Expectancy and Cleaning Procedures

Microchip uses socket modules from several manufacturers. Table A.5 gives the expected life (in number of automatic insertions) and cleaning method for each socket module as reported by the manufacturer.

Table A.5: Socket Life Expectancy and Cleaning Method

Manufacturer	Insertions	Cleaning Method
Aries (28 pin)	10,000	#2
AMP (18 pin)	25,000	#2
3M Textool	10,000	#1
AMP (Dip)	25,000	#2
3M Textool (SOIC)	10,000	#1
Aries	10,000	#2
Yamaichi	25,000	#1

A.5.3.1   Manufacturer

All sockets (except Yamaichi) are labeled with the manufacturer's name. Identify a Yamaichi socket by looking for the letters IC51– (as the prefix to a part number) on the socket.

A.5.3.2   Insertions

The expected life for manual insertions has been found to be less than the manufacturer's reported number. The number of manual insertions depends on the socket condition and how often the socket is cleaned.

Careless insertions or dirty socket conditions can bring the number of insertions down to less than 5,000. Cleanliness and care in inserting devices into a socket are most important with surface mount devices because the socket contactors must remain planar to function properly.

Any bent or nonplanar contacts will result in a failure. Nonplanar socket module contacts occur earlier in the life of a socket module when devices are inserted manually into a socket module. Early contact failure from manual insertions is due to the nonrepeatability of the manual insertion method. Therefore, the listed number of insertions might not be reached for sockets if devices are inserted manually. No good method exists to ensure that the contacts are planar.

A.5.3.3   Cleaning Method

Method #1 - Methyl Alcohol

Clean with methyl alcohol, and then blow off the contacts with dry compressed air.

DANGER	*Methyl alcohol is highly flammable. Use methyl alcohol in a well ventilated area away from sparks, flames, or any other source of ignition. Methyl alcohol is poisonous and may cause blindness if taken internally. Avoid inhaling methyl alcohol vapor.*

Method #2 – Not Recommended

There is no cleaning procedure for this socket type. If contacts get extremely contaminated, replace the socket.

A.5.4   ICSP Device Support Switch Settings

The following list shows the dip switch settings for the ICSP socket module for each device type. For more information on using In-Circuit Serial Programming (ICSP) with PRO MATE II, consult the *ICSP Socket Module User's Guide* (DS51113).

Device	Switch			
	1	**2**	**3**	**4**
18-pin mid-range part	Off	Off	On	On
28-pin mid-range part	Off	On	Off	Off
40-pin mid-range part	Off	On	Off	Off
PIC12CXXX	On	Off	Off	On
PIC14000	Off	On	Off	On
PIC16C505	On	Off	Off	On
PIC16C92X	Off	On	On	Off
PIC17C75X	Off	Off	On	Off
PIC18CXXX	Off	On	Off	Off

Appendix B. Troubleshooting

B.1   Introduction

The troubleshooting information in this chapter can help you resolve typical problems or obstacles in programming microcontroller devices.

B.2   Highlights

Topics covered in this appendix:

- Troubleshooting Hardware

    - Calibration
    - Communication Failure
    - Ensuring Proper Socket Module Contact
    - Socket Module Alignment
    - Socket Module Failure

- Troubleshooting Operation Type Problems

    - Device Selection in Stand-Alone Mode
    - Reading a Device Master in Stand-Alone Mode
    - Unstable EEPROM in Device Programmer
    - Device Pin Damage

- Troubleshooting Software

    - Establishing Communication with PRO MATE II
    - Default Serial Port

B.3   Troubleshooting Hardware

B.3.1   Calibration

An internal hardware problem could prevent proper calibration. If you receive the message "Calibration Error!" on the Device Programmer, contact your Microchip Sales Office for further instructions.

B.3.2   Communication Failure

Appendix A: Hardware Specifications, gives the data for connecting the PRO MATE II Device Programmer to a 25-pin serial port. Connect the corresponding terminals on each line of Table A.1. If communication fails, check your PC serial port.

B.3.3   Ensuring Proper Socket Module Contact

After changing a socket, insert a blank device and do a blank check (Vfy/F2 on the Device Programmer) to insure the socket is making proper contact. A blank device will show erased.

B.3.4   Socket Not Supported

Power-off the Device Programmer and realign the socket module properly. Tighten the screws evenly and securely (do not overtighten).

B.3.5   Socket Module Failure

If you can program a master chip, and if you can read and try to program code protected chips, but the chips fail the programming attempts, then potential socket pin damage may be the cause of the problem.

Contact your FAE if your socket module is not operating properly.

B.4   Troubleshooting Operational Problems

B.4.1   Device Selection in Stand-Alone Mode

When you power-up the Device Programmer, the unit will automatically detect the type of socket module installed and determine the device(s) for the installed socket module. A new socket module must be installed prior to selecting a new device type.

B.4.2   Reading a Device Master in Stand-Alone Mode

When reading a device master in stand-alone mode, the Device Programmer asks the question: "Code Protect Parts" being programmed. Answer **Yes** to code protect the parts that you will be programming.

B.4.3   Unstable EEPROM in Device Programmer

Programmer memory (EEPROM) can become damaged or "unstable." If the EEPROM contents are damaged, the message, "Ready for Download" appears on the LCD Display at normal power-up. Update your operating system when you receive this message. If this does not correct the problem, contact customer support.

B.4.4   Device Pin Damage

On the smaller device packages (SSOP, PQFP, and SOIC) the pins can bend easily and cause problems in programming the devices.

B.5 Troubleshooting Software

B.5.1 Establishing Communication with PRO MATE II

MPLAB attempts to establish communication with the PRO MATE II when you enable the programmer. If communication cannot be established, no programming can occur. A dialog box appears if the attempt to establish communication fails. If you encounter communications problems, try the following:

- Make sure that the RS-232 cable is connected, the power supply is connected, and the power switch on the PRO MATE II is on.

- Try connecting the PRO MATE II to a different serial port. If your PC has a 25-pin serial port, you will need a 25 to 9 serial port adapter.

- Make sure that a COM port is properly set up exclusively for use by PRO MATE II. Check the resources to ensure they are operating properly and that there are no conflicts with other devices. This commonly happens when you have a modem or other serial device that is improperly configured. Consult your Windows manual or other reference literature. You can try removing, reconfiguring, or disabling the conflicting device, but do so only if you are familiar with those procedures. See the steps below for Windows 3.1 or Windows 95.

- Some system errors are caused by driver and hardware incompatibility. See the steps below for Windows 95.

- If you have a COM port, but MPLAB will not let you select it (the option is grayed out), you may be able to assign the port manually by editing the `MPLAB.ini file`. Typically, this occurs if you have a gap in your COM port list (i.e., you have a COM1, COM2, and a COM4, but no COM3). In this case you may be able to fix it by opening `MPLAB.ini` (use FIND to locate this file) and editing the section called [programmers] so that the setting CommPort=1 is set to the port you want selected. This is just a work-around to a deeper problem in which Windows is incorrectly reporting port availability through the 16-bit driver.

- You must use the Microsoft Windows communications driver that is native to the version of Windows that you use. If you use Windows 3.10, look for the file `COMM.DRV` in your \WINDOWS\SYSTEM directory. That file MUST have a time of 3:10a. The time denotes the version. If you are using Windows for Workgroups, look for the same file as above, but the time stamp on that file should be 3:11a. If these files differ, you may need to re-install windows or install that file from another source. This problem is not likely to occur in Windows 95.

- Make sure you are not using a third party communications driver. Open your `SYSTEM.INI` file and look for the line in the [OPTIONS] section that reads

    COMM.DRV=COMM.DRV

If this line reads differently you are using a different communications driver.

Windows 3.1:

A serial mouse will use a COM port, as will an external modem. An internal modem has its own COM port, so if you have a second COM port on your PC, set it so it won't conflict with either the mouse or the modem.

Windows 95/98, Windows NT, Windows 2000:

Windows 95/98, Windows NT, and Windows 2000 require special attention to setting up COM ports. If you suspect a driver - hardware incompatibility, try changing Flow Control to Hardware and/or turning off the FIFO for the serial port. This is done in the Control Panel.

In Windows 95/98, click the System Icon. Click the **Device Manager** tab, and click **Ports**. If necessary, expand the Ports selection by clicking the "+" sign next to it. Double-click the I/O port that PICSTART Plus is connected to. This is where you can set flow control to Hardware. To turn off FIFO, click the **Advanced** button, deselect the Use FIFO box, and click **OK**.

In Windows NT, click the Ports Icon. Select the COM port, click **Settings**, and then click the **Advanced** button. Deselect the FIFI box, and click **OK**.

In Windows 2000, click the System Icon. Click the **Hardware** tab, and click the **Device Manger** button. If necessary, expand the Ports selection by clicking the "+" sign next to it. Double-click the I/O port that PICSTART Plus is connected to. This is where you can set flow control to Hardware. To turn off FIFO, click the **Port Settings** tab, and click the **Advanced** button. Deselect the Use FIFO box, and click **OK**.

B.5.2   No PRO MATE Menu Visible

If you have been using another programmer (e.g., PICSTART Plus), the PRO MATE menu might not be available. Select *Options > Programmer Options > Select Programmer* to open the Select Programmer dialog, and choose PRO MATE II from the list. After you select the programmer, MPLAB will issue you a warning prompt and shut itself down. You must restart MPLAB before the programmer options are available. The PRO MATE menu will appear on the menu bar when you restart MPLAB.

B.5.3   Default Serial Port

PRO MATE II uses COM1 as the default serial port the first time you run PRO MATE. If you change your serial port selection with the *Options > Programmer Options > Communications Port Setup* command, then the next time you run PRO MATE II, the host software will use the same serial port used in the previous session.

B.5.4   Operating System Update Needed

If the device you selected when setting up the development mode in MPLAB is not supported by the PRO MATE II Operating System, a message box will appear when you try to enable the programmer.

Make sure to get the latest versions for MPLAB software and PRO MATE II operating system (Section 7.12).

PRO MATE II User's Guide

NOTES:

Glossary

Introduction

To provide a common frame of reference, this glossary defines the terms for several Microchip tools.

Highlights

This glossary contains terms and definitions for the following tools:

- MPLAB IDE, MPLAB-SIM, 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.

PRO MATE II User's Guide

IDE

Integrated <u>D</u>evelopment <u>E</u>nvironment. 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

R̲andom A̲ccess M̲emory (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

R̲ead O̲nly M̲emory (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

S̲pecial F̲unction R̲egisters of a PICmicro MCU.

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.
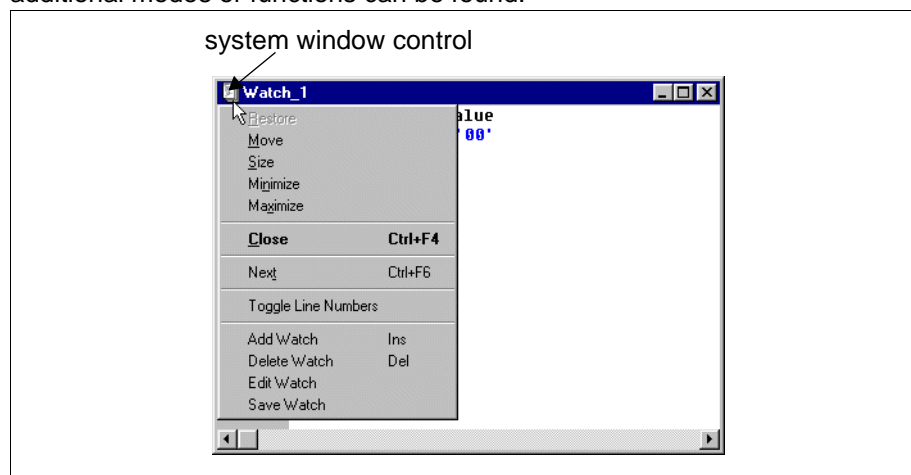
Figure G1: System Window Control Menu - Watch Window

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.

PRO MATE II USER'S GUIDE

Index

PRO MATE II User's Guide

Index

PRO MATE II User's Guide

NOTES:

PRO MATE II User's Guide

NOTES:

NOTES:

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

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

Austin - Analog
8303 MoPac Expressway North
Suite A-201
Austin, TX 78759
Tel: 512-345-2030 Fax: 512-345-6085

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Boston - Analog
Unit A-8-1 Millbrook Tarry Condominium
97 Lowell Road
Concord, MA 01742
Tel: 978-371-6400 Fax: 978-371-0050

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

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Dayton
Two Prestige Place, Suite 130
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

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

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

New York
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

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai)
Co., Ltd., Beijing Liaison Office
Unit 915
New China Hong Kong Manhattan Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai)
Co., Ltd., Chengdu Liaison Office
Rm. 2401, Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou
Microchip Technology Consulting (Shanghai)
Co., Ltd., Fuzhou Liaison Office
Rm. 531, North Building
Fujian Foreign Trade Center Hotel
73 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7557563 Fax: 86-591-7557572

China - Shanghai
Microchip Technology Consulting (Shanghai)
Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen
Microchip Technology Consulting (Shanghai)
Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong
Microchip Technology Hongkong Ltd.
Unit 901, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

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

Taiwan
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
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 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Germany - Analog
Lochhamer Strasse 13
D-82152 Martinsried, Germany
Tel: 49-89-895650-0 Fax: 49-89-895650-22

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 5869 Fax: 44-118 921-5820

06/01/01

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

[Microchip](#):
  [DV007003](#)