MPLAB® ICD USER'S GUIDE

Table of Contents

Preface

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

Chapter 1. Overview and Installation

1.1 Introduction 9
1.2 Highlights 9
1.3 What MPLAB ICD Is 9
1.4 Resources Used By MPLAB ICD 10
1.5 MPLAB ICD System Components 11
1.6 How MPLAB ICD Helps You 14
1.7 Installing MPLAB ICD Hardware 15
1.8 Applying Power to the System Components 17
1.9 Installing MPLAB IDE Software 17

Chapter 2. Tutorial – PIC16F877

2.1 Introduction 19
2.2 Highlights 19
2.3 Running MPLAB IDE 20
2.4 Creating a Hex File for Debugging 21
2.5 Setting Up MPLAB ICD 29
2.6 Setting Programming and Debugging Options 30
2.7 Programming the PIC16F877 32

2.8 Setting Up the Demo Board 33
2.9 Running Tut877 33
2.10 Debugging Tut877 34
2.11 Tut877 Main Routine 39
2.12 Tut877 Corrected Source Code 40

Chapter 3. General Setup

3.1 Introduction 43
3.2 Highlights 43
3.3 Running MPLAB IDE 43
3.4 Setting Up the Development Mode 44
3.5 Setting Up the MPLAB ICD Dialog 45
3.6 Setting Up the ICD Options Dialog 47
3.7 Using MPLAB Projects 51

Chapter 4. Basic Functions

4.1 Introduction 53
4.2 Highlights 53
4.3 MPLAB ICD Operations 53
4.4 Program Execution 54
4.5 Breakpoints 55
4.6 Modifying Functions 55
4.7 Loading a Hex File for Debugging 56
4.8 Upgrading the MPLAB ICD Operating System (Firmware) 56

Chapter 5. Troubleshooting

5.1 Introduction 57
5.2 Highlights 57
5.3 Common Problems 57

Appendix A. Hardware Specifications

A.1 Introduction 61
A.2 MPLAB ICD Module and Header 61
A.3 MPLAB ICD Demo Board 68

Table of Contents

Glossary

Introduction 75

Highlights 75

Terms 75

Index 93

Worldwide Sales and Service 96

Preface

Introduction

This first chapter contains general information that will be useful to know before using MPLAB ICD.

Highlights

The information you will garner from 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 MPLAB ICD as a development tool to debug firmware on a target board. The manual layout is as follows:

- **General Information** – This section.
- **Chapter 1: Overview and Installation** – Describes what MPLAB ICD is, how it works and how to install MPLAB ICD hardware and MPLAB IDE software and establish communications.
- **Chapter 2: Tutorial – PIC16F877** – Shows you how to develop and debug an application using MPLAB IDE Projects and MPLAB ICD.
- **Chapter 3: General Set Up** – Tells you how to get MPLAB ICD up and running.
- **Chapter 4: Basic Functions** – Describes the basic functions of MPLAB ICD.
- **Chapter 5: Troubleshooting** – Provides information on solving common problems.
- **Appendix A: Hardware Specifications** – Provides a technical description of the MPLAB ICD Header, MPLAB ICD Module, and MPLAB ICD Demo Board hardware.

- **Glossary** – A glossary of terms used in this guide.
- **Index** – Cross-reference listing of terms, features, and sections of this document.
- **Worldwide Sales and Service** – A listing of Microchip Technology 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`
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>

Table: Documentation Conventions (Continued)

Description	Represents	Examples
Documents (Helvetica font):		
Italic characters	Referenced books	*MPLAB IDE User's Guide*

Documentation Updates

All documentation becomes dated, and this user's guide is no exception. Since MPLAB IDE and other Microchip tools are constantly evolving to meet customer needs, some MPLAB IDE dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site at www.microchip.com 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 MPLAB ICD. The user may also find the data sheets for specific microcontroller devices informative in developing firmware.

README.ICD

For the latest information on using MPLAB ICD, read the README.ICD file (ASCII text file) included with the MPLAB ICD software. The README.ICD file contains update information that may not be included in this document.

README.XXX

For the latest information on other Microchip tools (MPLAB, MPASM™, etc.), read the associated README files (ASCII text file) included with the MPLAB IDE software.

MPLAB IDE User's Guide (DS51025)

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

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

This user's guide describes how to use the Microchip PICmicro® assembler (MPASM), the linker (MPLINK), and the librarian (MPLIB).

Microchip Technology Library CD-ROM (DS00161)
This CD-ROM contains comprehensive application notes, data sheets, and technical briefs for all of Microchip products. To obtain this CD-ROM, contact the nearest Microchip Sales and Service location (see back page).

Embedded Control Handbook Vol.1 & 2 and the Embedded Control Update 2000 (DS00092, DS00167, and DS00711)
These handbooks contain a wealth of information about microcontroller applications. To obtain these documents, contact the nearest Microchip Sales and Service location (see back page).

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

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

Troubleshooting

See Chapter 5 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 web site is available by using your favorite Internet browser to connect to:

http://www.microchip.com

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

ftp://ftp.microchip.com

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

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

Development Systems Customer Notification Service

Microchip provides a customer notification service to help our customers keep current on Microchip products with the least amount of effort. Once you subscribe to one of our list servers, you will receive email notification whenever we change, update, revise or have errata related to that product family or development tool. See the Microchip web site at www.microchip.com.

The Development Systems list names are:

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

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

```
listserv@mail.microchip.com
```

with the following as the body:

```
subscribe <listname> yourname
```

Here is an example:

```
subscribe mplab John Doe
```

To UNSUBSCRIBE from these lists, send a message to:

```
listserv@mail.microchip.com
```

with the following as the body:

```
unsubscribe <listname> yourname
```

Here is an example:

```
unsubscribe mplab John Doe
```

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

Compilers

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

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® emulators.

To SUBSCRIBE to this list, send a message to:

```
listserv@mail.microchip.com
```

with the following as the body:

```
subscribe emulators yourname
```

Programmers

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

To SUBSCRIBE to this list, send a message to:

```
listserv@mail.microchip.com
```

with the following as the body:

```
subscribe programmers yourname
```

MPLAB

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

To SUBSCRIBE to this list, send a message to:

```
listserv@mail.microchip.com
```

with the following as the body:

```
subscribe mplab yourname
```

Otools (Other Tools)

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

To SUBSCRIBE to this list, send a message to:

```
listserv@mail.microchip.com
```

with the following as the body:

subscribe otools yourname

Customer Support

Users of Microchip products can receive assistance through several channels:

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

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

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

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

The Hotline Numbers are:

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

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

Chapter 1. Overview and Installation

1.1 Introduction

This section gives you a preview of the MPLAB ICD (In-Circuit Debugger) features and functions, as well as its hardware and software elements.

1.2 Highlights

Topics covered in this chapter:

- What MPLAB ICD Is
- Resources Used By MPLAB ICD
- MPLAB ICD System Components
- How MPLAB ICD Helps You
- MPLAB ICD Kit Components
- Installing MPLAB ICD Hardware
- Applying Power to the System Components
- Installing MPLAB IDE Software

1.3 What MPLAB ICD Is

MPLAB ICD utilizes the In-Circuit Debugging capability of the PIC16F87X and Microchip's In-Circuit Serial Programming™ (ICSP™) protocol to both program and act as an in-circuit debugger for PIC16F87X devices. It operates under MPLAB IDE, connects to an application, and runs like the PIC16F87X microcontroller in the design.

MPLAB ICD is intended to be used as an evaluation and debugging aid in a laboratory environment.

The MPLAB ICD offers these features:

- Real-time and single-step code execution
- Breakpoint
- In-circuit debugging
- Built-in programming
- 3.0V to 5.5V operating range
- Operation from voltage (V_{DD}) supplied by the target application
- Operating frequencies from 32 kHz to 20 MHz
- Source level and symbolic debugging
- MPLAB IDE user interface
- Compatibility with Microsoft Windows® 3.X, Windows 95/98, Windows NT®, and Windows 2000®
- RS-232 Interface

1.4 Resources Used By MPLAB ICD

Due to the built-in in-circuit debugging capability of the PIC16F87X and ICSP function offered by the debugger, the MPLAB ICD uses the following on-chip resources:

- $\overline{\text{MCLR}}$/VPP is shared for programming
- Low-voltage ICSP programming is disabled

  Note: Low Voltage Programming is not supported by MPLAB ICD. MPLAB ICD requires you to disable low voltage programming while in Debug mode.
- RB6 and RB7 reserved for programming and in-circuit debugging
- Six or seven general purpose file registers are reserved for debug monitor (see below)
- First program memory location (address 0x0000) must be a NOP instruction
- The last 256 or 288 words of program memory area are reserved for Debug Code (depending on device, see below)
- One stack level is not available

Data and Program Memory Used by MPLAB ICD

The MPLAB ICD uses the following file register and program memory locations in the PIC16F87X target device:

Processor	File Registers Used	Program Memory Used
PIC16F870/871/872	0x70, 0x0BB-0x0BF	0x06E0-0x07FF
PIC16F873/874	0x6D,0x16D, 0x0EB-0x0F0, 0x1EB-0x1F0	0x0EE0-0x0FFF
PIC16F876/877	0x70, 0x1EB-0x1EF	0x1F00-0x1FFF

Note: If you have an MPLAB ICD firmware version older than 2.00, you must contact your sales office to obtain an MPLAB ICD firmware upgrade in order to use the MPLAB ICD with a PIC16F87X other than the PIC16F876/877.

1.5 MPLAB ICD System Components

There are two versions of MPLAB ICD. The MPLAB ICD Evaluation Kit (DV164001) and the MPLAB ICD Module (DV164002). The MPLAB ICD Evaluation Kit (DV164001) includes the items listed below (Figure 1.1). The MPLAB ICD Module (DV164002) contains the same items except for items 1, 3 and 5, which are not included.

1. The MPLAB ICD Header
2. The MPLAB ICD Module
3. The MPLAB ICD Demo Board
4. RS-232 cable
5. 40-pin DIP and 28-pin SDIP connection sockets
6. 9-inch 6-conductor modular cable
7. CD with complete MPLAB IDE software and documentation
8. Manuals:
    - *MPLAB ICD User's Guide* (this document)
    - *MPLAB IDE User's Guide* (not shown)
    - *MPASM with MPLINK and MPLIB User's Guide* (not shown)
9. A Warranty Registration card (not shown)

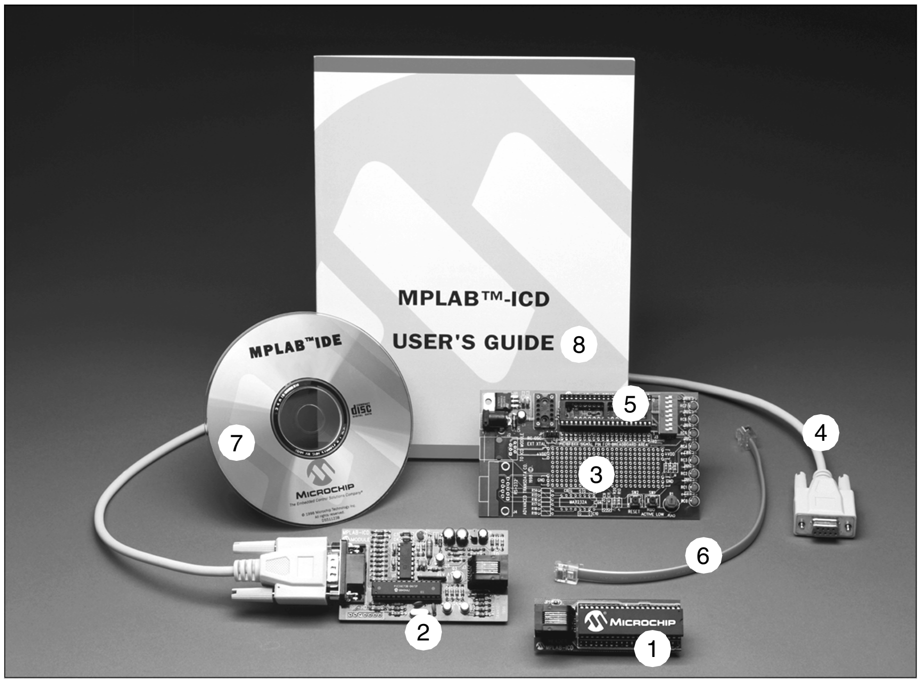


Figure 1.1: MPLAB ICD Evaluation Kit Components

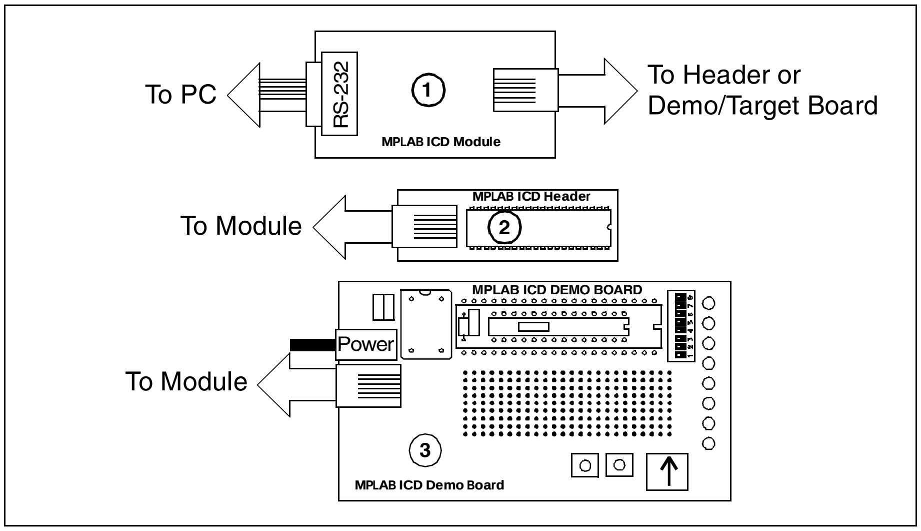


Figure 1.2: MPLAB ICD System Components

In addition to the included components, you will need a power supply to power the MPLAB ICD Demo Board or your target application. Power can be supplied by a PICSTART Plus power supply (part AC162039) connected to the MPLAB ICD Demo Board or by your target application. For more information, refer to Section A.3.

System Configurations

There are two possible MPLAB ICD system configurations:

- Connect the PC and the MPLAB ICD Module. Connect the MPLAB ICD Module and the MPLAB ICD Header. Plug the MPLAB ICD Header into the socket on the MPLAB ICD Demo Board or your target application. This configuration only applies if you have the MPLAB ICD Evaluation Kit.
- Connect the PC and the MPLAB ICD Module. Connect the MPLAB ICD Module and the MPLAB ICD Demo Board (if you have the MPLAB ICD Evaluation Kit) or your target application. Plug a PIC16F87X device into the socket on the MPLAB ICD Demo Board or your target application. (The PIC16C87X may be soldered directly onto the target board.)

1.5.1 MPLAB ICD Module

The MPLAB ICD Module contains all debugging, programming, and control logic. It is connected to the PC's serial port via a 9-pin serial cable and to the MPLAB ICD Demo Board or target application using a 6-wire modular cable. The application board must have a 6-wire modular connection (shown as item 12 in Section 1.5.3) unless you have the MPLAB ICD Evaluation Kit, in which case the MPLAB ICD Module can be connected to the MPLAB ICD Header and then the MPLAB ICD Header plugged into a socket on the MPLAB ICD Demo Board or target application.

The MPLAB ICD Module contains the firmware to provide serial communications to the PC, to drive the MPLAB ICD communications to the Demo Board/target board (or to the MPLAB ICD Header), and to program a target PIC16F87X using ICSP, all from the MPLAB IDE. The MPLAB ICD Module is powered from the MPLAB ICD Demo Board or the target application and requires 70 mA (max) in addition to what the target consumes.

The modular cable connects to the MPLAB ICD Header or the target application with the appropriate connections to the PIC16F87X to allow in-circuit emulation of QFP and DIE parts as described in Section 1.7.2.

1.5.2 MPLAB ICD Header

The MPLAB ICD Header, included only in the MPLAB ICD Evaluation Kit, provides a means of connecting the MPLAB ICD Module to a target board that does not incorporate a 6-wire modular connection into its design. The MPLAB ICD Module connects to the MPLAB ICD Header via a 9-inch modular cable. For in-circuit emulation, a 40-pin PIC16F87X needs to be plugged into the MPLAB ICD Header, which then plugs into a 28-pin or 40-pin PIC16F87X DIP socket on the MPLAB ICD Demo Board or target application as described in Section 1.7.1.

The MPLAB ICD Header is powered by the Demo Board/target board, from a 3.0V to 5.5V source.

1.5.3 MPLAB ICD Demo Board

The MPLAB ICD Demo Board, included only in the MPLAB ICD Evaluation Kit, is provided for demonstration and evaluation of the PIC16F87X without a target application board. It can be connected to the MPLAB-ICD Module directly or via the MPLAB ICD Header. The PIC16F877 can be unplugged from the MPLAB ICD Header and plugged directly into the MPLAB ICD Demo Board for stand-alone operation.

The MPLAB ICD Demo Board (Figure 1.3) has the following hardware features:

1. 40- and 28-pin sockets. For information on using stand-offs to select the desired pin count socket, see Section A.3.

2. Eight DIP switches to connect and disconnect each of the eight LEDs to and from its respective PORTC pin.
3. Eight red LEDs connected to PORTC for displaying 8-bit binary values.
4. Two push-button switches, one for RESET and one for external stimulus on RB0.
5. A potentiometer for analog input on RA0.
6. A small prototyping area.
7. A connector area to access the I/O pins of the PIC16F87X for expansion prototyping.
8. A jumper to select the RC oscillator (see Section A.3.8) or an external crystal.
9. Socket for external crystal.
10. A connector for a 9V, 0.75A power supply, similar to the PICSTART Plus programmer power supply. Refer to refer to Section A.3.
11. Provisions for a MAX232 and associated hardware that may be populated to add RS-232 capability.
12. A modular cable connection that can be used to connect the MPLAB ICD Demo Board directly to the MPLAB ICD Module.

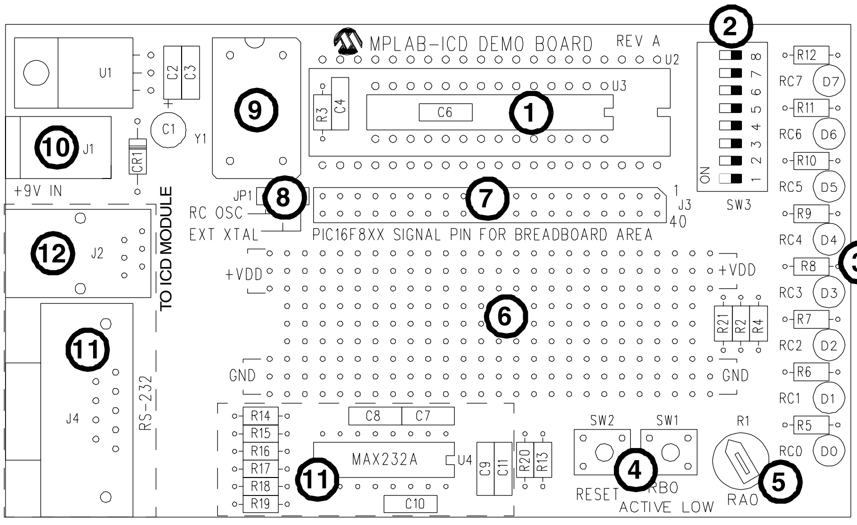


Figure 1.3: MPLAB ICD Demo Board

1.6 How MPLAB ICD Helps You

The MPLAB ICD allows you to:

- Debug your source code in your own application
- Debug your hardware in real-time
- Program a target PIC16F87X using Microchip's ICSP™ protocol

1.7 Installing MPLAB ICD Hardware

There are two ways to use the MPLAB ICD and, hence, two ways to install the hardware.

1.7.1 Connecting the MPLAB ICD Module, Header, and Demo/Target Board

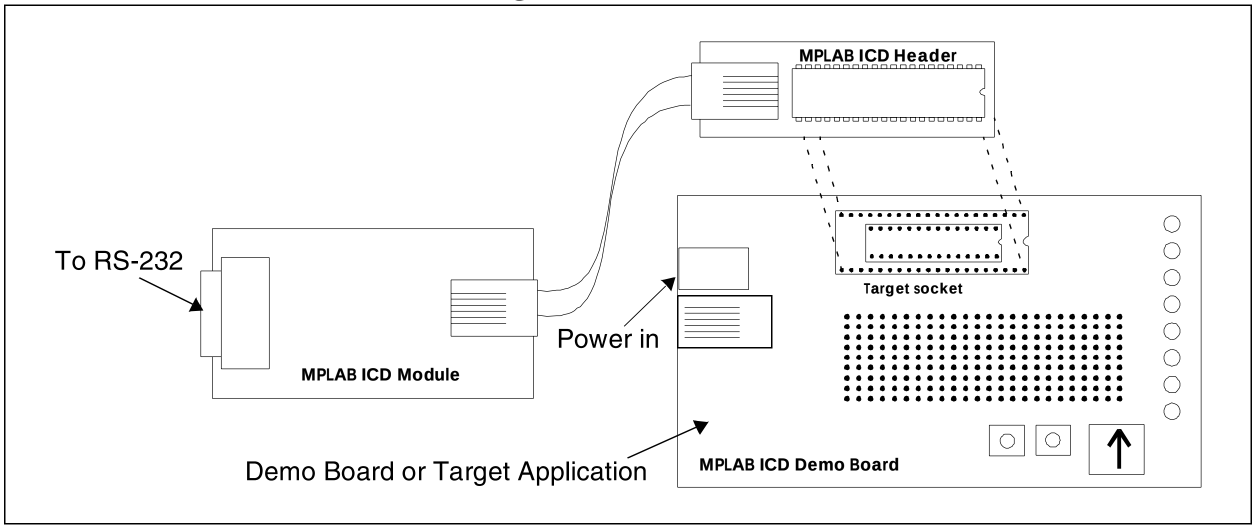


Figure 1.4: MPLAB ICD Connection with Header

This type of installation is only applicable if you have the MPLAB ICD Evaluation Kit. Install the MPLAB ICD system hardware by following these steps:

1. Plug the 40-pin PIC16F87X device into the 40-pin DIP socket in the MPLAB ICD Header board.

To debug this part on the target or Demo Board	Use this part on the Header
PIC16F870	PIC16F871
PIC16F871	PIC16F871
PIC16F873	PIC16F874
PIC16F874	PIC16F874
PIC16F876	PIC16F877
PIC16F877	PIC16F877

Note 1: Devices without a 40-pin equivalent cannot be used with the MPLAB ICD Header. To debug such devices, you must connect the MPLAB ICD Module directly to the demo/target board as described in Section 1.7.2.

2: If you have an MPLAB ICD firmware version older than 2.00, you must contact your sales office to obtain an MPLAB ICD firmware upgrade in order to use the MPLAB ICD with a PIC16F87X other than the PIC16F876/877.

2. Connect the 9-inch modular cable between the MPLAB ICD Module and the MPLAB ICD Header.
3. If you are debugging a 40-pin part using the MPLAB ICD Demo Board or a target board, insert the two 20-pin male to male headers (stand-offs) onto the MPLAB ICD Demo Board or target board sockets. Then, plug the MPLAB ICD Header board into the stand-offs (see Section A.3).

   If you are debugging a 28-pin part using the MPLAB ICD Demo Board or a target board, insert the two 14-pin male to male headers (stand-offs) onto the MPLAB ICD Demo Board or target board sockets. Then, plug the MPLAB ICD Header board into the stand-offs (see Section A.3).
4. Connect the RS-232 cable between the serial port of the host computer and the MPLAB ICD Module.

1.7.2 Connecting the MPLAB ICD Module and Demo/ Target Board

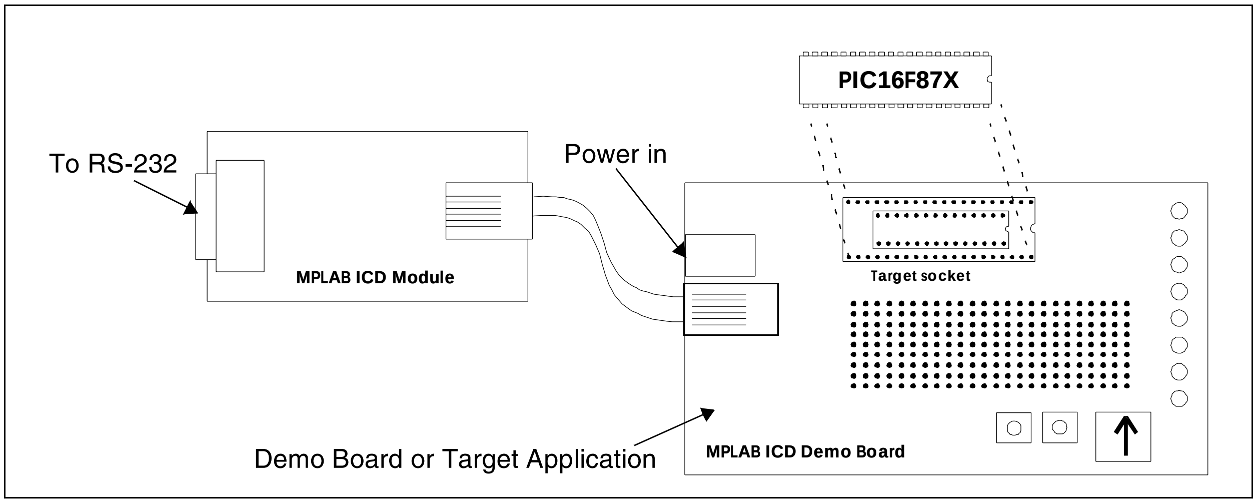


Figure 1.5: MPLAB ICD Connection Without Header

You can connect the MPLAB ICD Module directly to the MPLAB ICD Demo Board (if you have the MPLAB ICD Evaluation Kit), or directly to your target board if your target board incorporates a modular cable connection described in the following table.

Pin	Signal
1	RB3
2	RB6
3	RB7
4	Ground
5	$+V_{DD}$
6	V_{PP}

This configuration also provides full MPLAB ICD functionality.

1. Plug a PIC16F87X into the 28-pin or 40-pin DIP socket in the MPLAB ICD Demo Board or target board.
2. Connect the 9-inch modular cable between the MPLAB ICD Module and the MPLAB ICD Demo Board or your target board.
3. Connect the RS-232 cable between the serial port of the host computer and the MPLAB ICD Module.

1.8 Applying Power to the System Components

MPLAB ICD must be run using external (target board) power.

To prevent damage to any part of the system, power up the system components in the following sequence:

1. Turn on the power to the host computer.
2. Turn on the power to the MPLAB ICD Demo Board/target application which also powers the MPLAB ICD Module. (For the MPLAB ICD Demo Board, you can use the 9 VDC center-positive power adapter supplied with the PICSTART Plus Programmer.)

Turn the system components off in reverse order.

1.9 Installing MPLAB IDE Software

To install the MPLAB IDE software, refer to the installation instructions found in the *MPLAB IDE User's Guide*.

The MPLAB ICD functions with the MPLAB IDE software under the following operating systems:

- Microsoft Windows 3.1x
- Windows 95/98
- Windows NT
- Windows 2000

NOTES:

Chapter 2. Tutorial – PIC16F877

2.1 Introduction

The MPLAB ICD is a programmer for the PIC16F87X family, as well as an in-circuit debugger. This tutorial will help you start using the MPLAB ICD hardware and software to program your part and debug source code.

This tutorial uses a sample project, Tut877. The `tut877.asm` program is a simple implementation of the PIC16F877's analog-to-digital (A/D) converter using the MPLAB ICD Demo Board. This program configures the A/D module to convert an A/D channel 0 (connected to the potentiometer on the MPLAB ICD Demo Board) and display the results on the LEDs on PORTC.

This tutorial requires the use of the MPLAB ICD Demo Board. Therefore, if you have the standalone MPLAB ICD Module (DV164002) rather than the MPLAB ICD Evaluation Kit (DV164001), this chapter is not applicable.

2.2 Highlights

Topics covered in this chapter:

- Running MPLAB IDE
- Creating a Hex File for Debugging
- Setting up MPLAB ICD
- Setting Programming and Debugging Options
- Programming the PIC16F877
- Setting up the Demo Board
- Running Tut877
- Debugging Tut877
- Tut877 Main Routine
- Tut877 Corrected Source Code

2.3 Running MPLAB IDE

After installing the MPLAB IDE software, invoke it by executing the file `mplab.exe`.

For more information on installing and using the MPLAB IDE, refer to the *MPLAB IDE User’s Guide* (DS51025) and the `README.LAB` file in the MPLAB install directory.

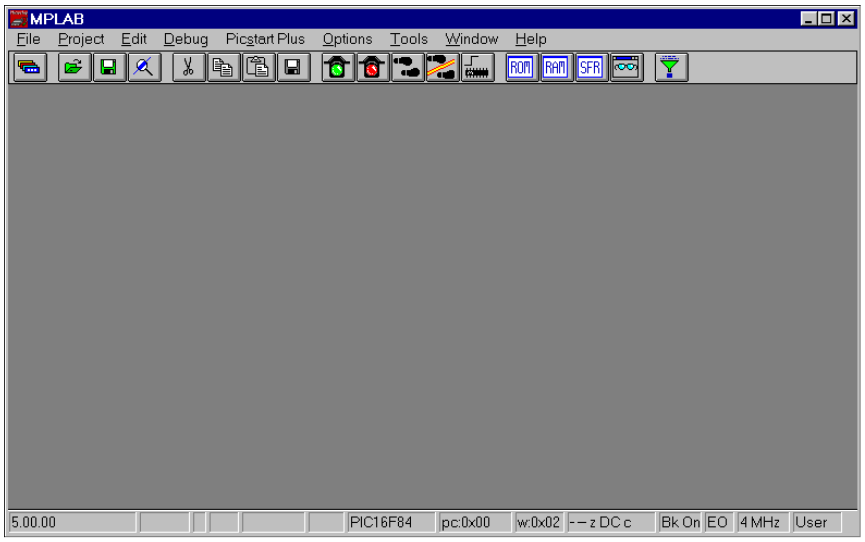

MPLAB
File Project Edit Debug Picstart Plus Options Tools Window Help
ROM RAM SFR
5.00.00 PIC16F84 pc:0x00 w:0x02 --z DC c Bk On EO 4 MHz User

Figure 2.1: MPLAB IDE

2.4 Creating a Hex File for Debugging

You will need to create a new project to include the source file `tut877.asm` and build the hex file `tut877.hex`. See the next sections for instructions on how to create a new project, `tut877.pjt`.

2.4.1 New Project Directory

In File Manager (Windows 3.1) or Windows Explorer, create a subdirectory for the new project, `\MPLAB\tut877`. Move the `tut877.asm` file from `\MPLAB` to this subdirectory.

2.4.2 New Project

Select *Project > New Project*, select the directory for the new project, and then enter `tut877.pjt` in the File Name box. Click **OK**.

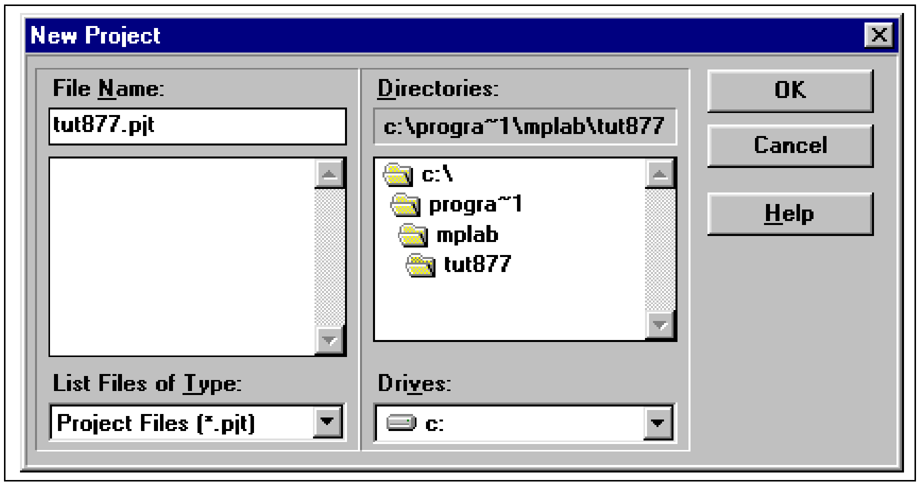
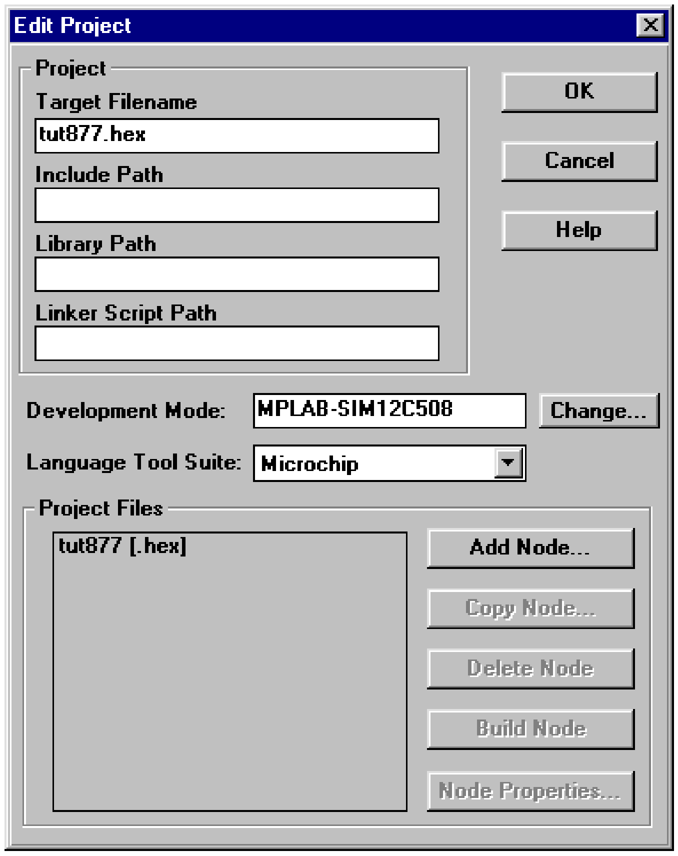


Figure 2.2: New Project – `tut877.pjt`

2.4.3 Project Dialog

The Edit Project dialog should open.



Target Filename: tut877.hex

Include Path:

Library Path:

Linker Script Path:

Development Mode: MPLAB-SIM12C508

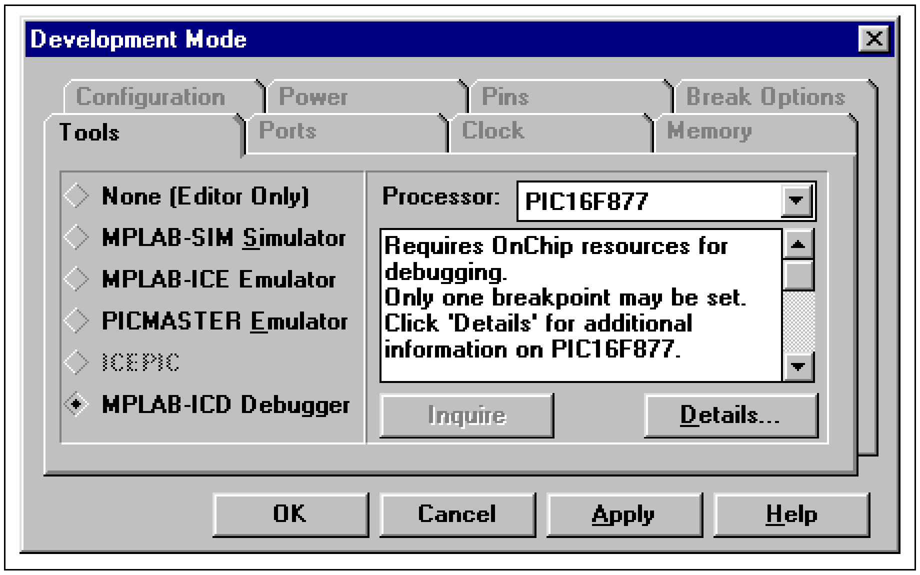
Language Tool Suite: Microchip

tut877 [.hex]

Figure 2.3: Edit Project Dialog Before Setting Development Mode

Notice the development mode. It indicates we are working with the MPLAB SIM simulator and a PIC12C508 processor. Your display will indicate whatever development mode and processor you were working with previously in the MPLAB IDE. We will need to change these settings now.

Click **Change**.



Processor: PIC16F877

Requires OnChip resources for debugging.
Only one breakpoint may be set.
Click 'Details' for additional information on PIC16F877.

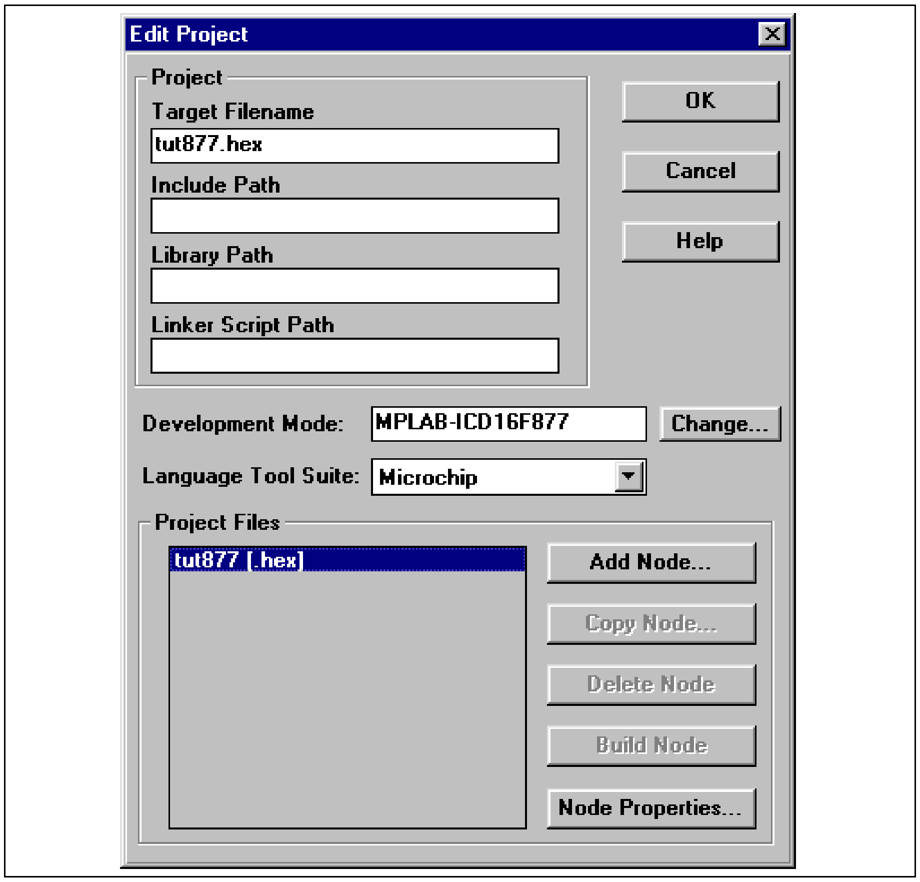
Figure 2.4: Setting the Development Mode

Select MPLAB ICD Debugger under Tools. Make sure you select the PIC16F877 processor. Click **OK**.

A window will inform you that no hex file has been built for this project. We will build the hex file later. Click **OK**.

MPLAB IDE will now establish communications with the MPLAB ICD. The MPLAB ICD dialog will momentarily appear during this process. If you receive an error message, double-check the connections for power supply, socket seating, and cable seating. For more detailed troubleshooting information, refer to Chapter 5.

The MPLAB ICD dialog must remain open throughout the tutorial. Do not close the MPLAB ICD dialog when it appears. If you want, you can move the dialog.


Edit Project

Project

Target Filename: tut877.hex

Include Path:

Library Path:

Linker Script Path:

Development Mode: MPLAB-ICD16F877

Language Tool Suite: Microchip

Project Files: tut877 [.hex]

Figure 2.5: Edit Project Dialog

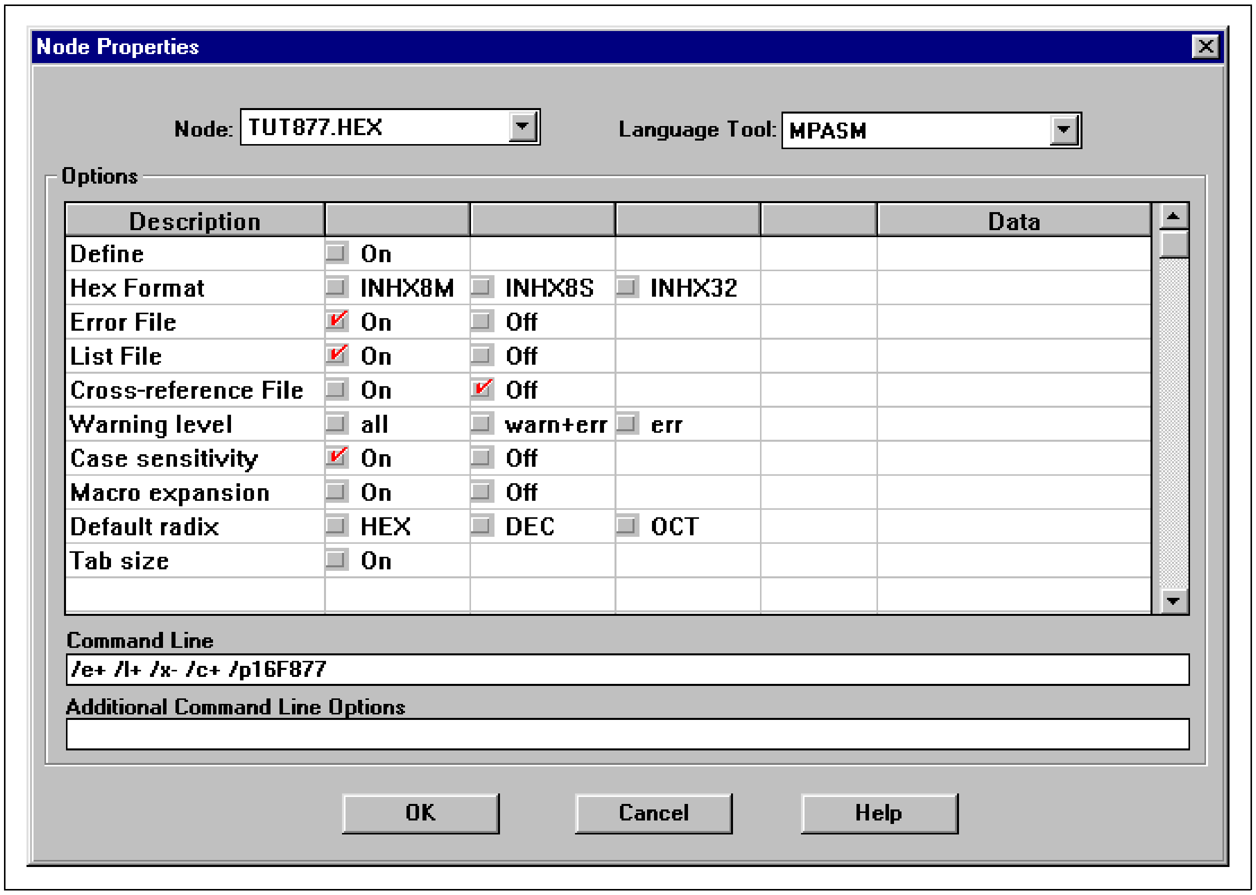
Notice that the correct development mode and processor are shown in the Edit Project dialog.

Click on the line `tut877 [.hex]` in the Project Files area of the Edit Project dialog, and then click the **Node Properties** button.

2.4.4 Set Node Properties

The Node Properties dialog shows the command line switches for the tool, in this case the MPASM assembler. When you first open this dialog, the checked boxes represent the default values for the tool. For this tutorial, these settings do not need to be changed from their defaults.

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



Node: TUT877.HEX

Language Tool: MPASM

Options

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

Command Line

```
/e+ /l+ /x- /c+ /p16F877
```

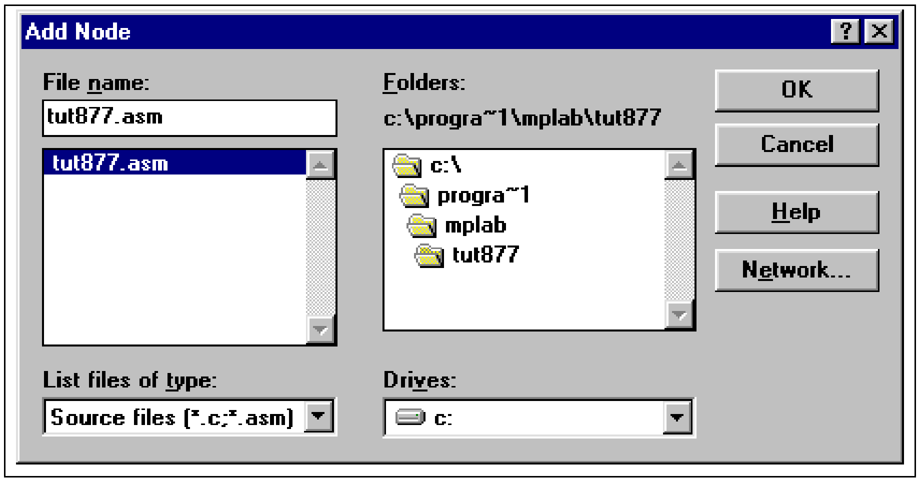
Additional Command Line Options

Figure 2.6: Node Properties Dialog

2.4.5 Add Node

Click **Add Node** from the Edit Project dialog to open the Add Node dialog.

Select `tut877.asm` for this tutorial, and then click **OK**.



File name: tut877.asm

Folders: c:\progra~1\mplab\tut877

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

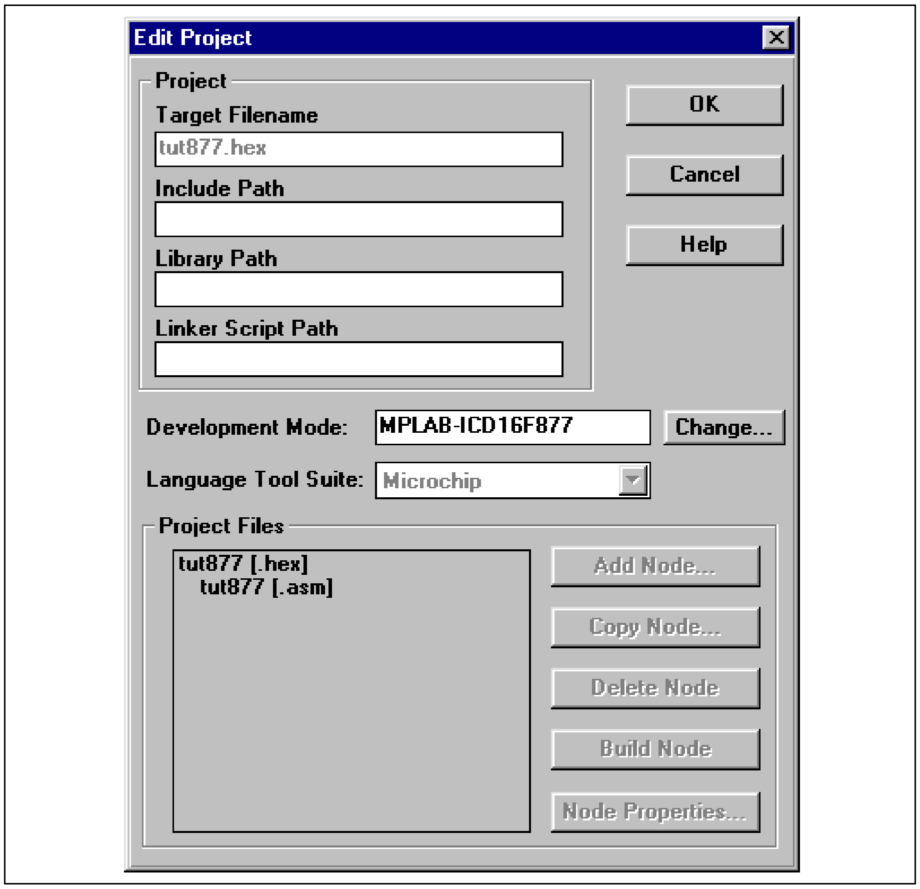
Drives: c:

Figure 2.7: Add Node

2.4.6 Complete Project Setup

In this simple example, no entries were made in the Path boxes. As your application becomes more complex, you may need to enter the directories of your Include Files in the appropriate boxes.

The MPASM assembler always makes a `.hex` file with the same name as the source `.asm` file. The Project Manager will create a `tut877.hex` file when the project is built.



Target Filename: tut877.hex
Include Path:
Library Path:
Linker Script Path:
Development Mode: MPLAB-ICD16F877
Language Tool Suite: Microchip

tut877 [.hex]
  tut877 [.asm]

Figure 2.8: Edit Project Dialog with Node

Click **OK** to close the Edit Project Dialog.

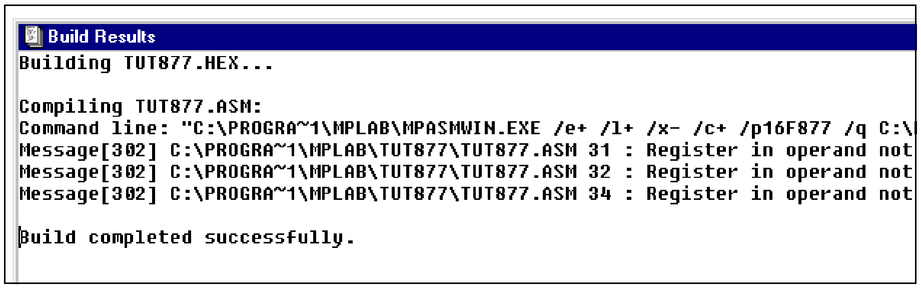
Now select *Project > Save Project* from the MPLAB IDE menu to save your project.

2.4.7 Make Project

Select *Project > Make Project* from the menu to compile the application using the MPASM assembler.

> **Note:** When you successfully build the project, a window will inform you that the project has been rebuilt and that you must reprogram the ICD in order to effect the change. If you do not want to see this warning each time you rebuild, check the checkbox to ignore this warning. We will do this later in the tutorial. Click **Close**.

A Build Results window shows the command line sent to the assembler. It should look like this:



```
Building TUT877.HEX...

Compiling TUT877.ASM:
Command line: "C:\PROGRA~1\MPLAB\MPASMWIN.EXE /e+ /l+ /x- /c+ /p16F877 /q C:\
Message[302] C:\PROGRA~1\MPLAB\TUT877\TUT877.ASM 31 : Register in operand not
Message[302] C:\PROGRA~1\MPLAB\TUT877\TUT877.ASM 32 : Register in operand not
Message[302] C:\PROGRA~1\MPLAB\TUT877\TUT877.ASM 34 : Register in operand not

Build completed successfully.
```

Figure 2.9: Build Results Window

> **Note:** Message [302] is simply letting you know that, at the designated line number (i.e., 31, 32, and 34) you are specifying a file register that is not in bank zero. This is not an error and your code will compile correctly.

Click the X in the upper right corner of the Build Results window to close it. Or, click the Restore icon next to the X to resize the window. Subsequent builds will use this smaller Build Results window.

2.5 Setting Up MPLAB ICD

At this point, the MPLAB ICD dialog should be on your desktop. Make the selections described in this section to set up the MPLAB IDE for use with the MPLAB ICD hardware.

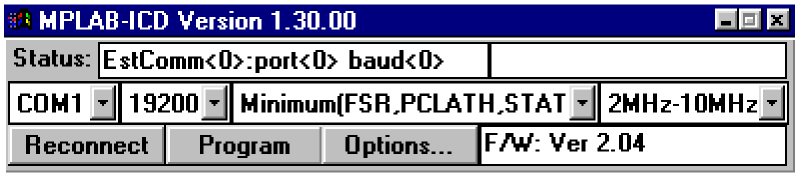


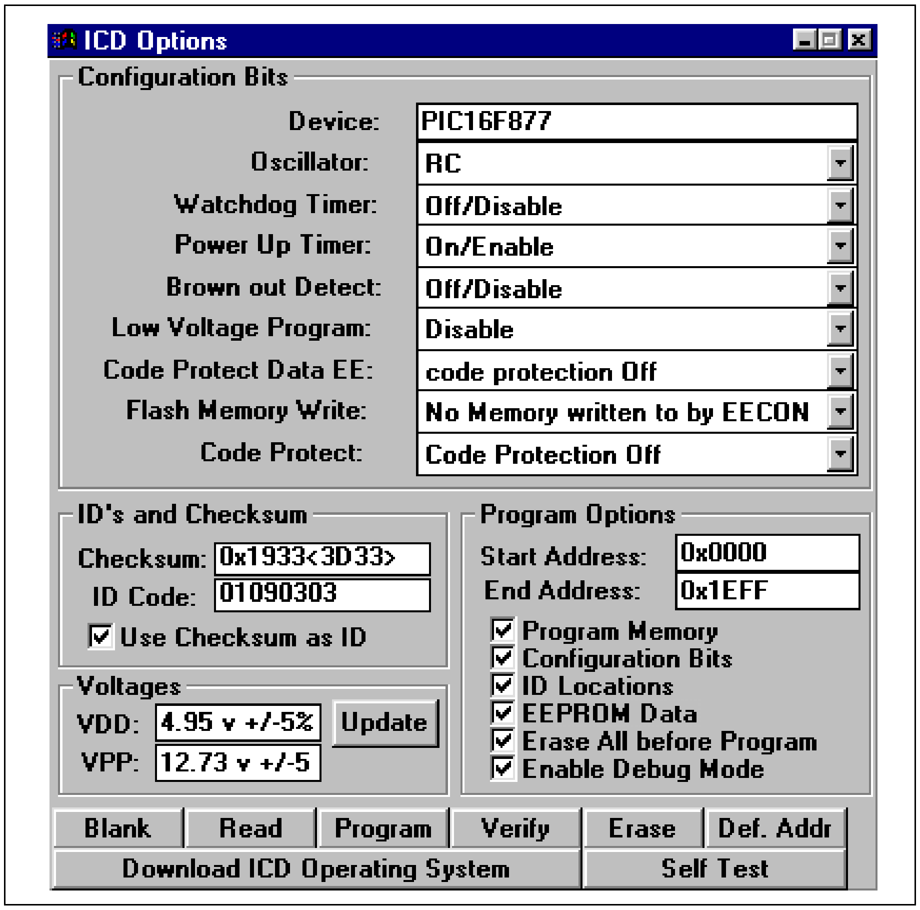
Figure 2.10: MPLAB ICD Dialog

Table 2.1: MPLAB ICD Dialog

Item	Options
Status	The Status bar displays the executed MPLAB ICD function and the status. When you program a device, you can watch the progress in this area. When the operation is complete, the Status box displays the message "Waiting for user command."
COM Port and Baud Rate Pull-Down Menus	Do not change these values for this tutorial.
Upload Options Pull-Down Menu	Select Minimum for now. Later in the tutorial we will change this for debugging.
Operating Frequency Range Pull-Down Menu	Select the operation frequency range of 2 MHz – 10 MHz.
Options	Click this button to bring up the ICD Options dialog.

2.6 Setting Programming and Debugging Options

To program the target PIC16F877 device, the ICD Options dialog must first be set up for programming. The small MPLAB ICD dialog was opened when you entered the MPLAB ICD development mode. Click **Options** in the MPLAB ICD dialog to open the ICD Options dialog.



ICD Options

Configuration Bits

Device:	PIC16F877
Oscillator:	RC
Watchdog Timer:	Off/Disable
Power Up Timer:	On/Enable
Brown out Detect:	Off/Disable
Low Voltage Program:	Disable
Code Protect Data EE:	code protection Off
Flash Memory Write:	No Memory written to by EECON
Code Protect:	Code Protection Off

ID's and Checksum

Checksum: 0x1933<3D33>
ID Code: 01090303
✓ Use Checksum as ID

Voltages

VDD: 4.95 v +/-5%
VPP: 12.73 v +/-5

Program Options

Start Address: 0x0000
End Address: 0x1EFF
✓ Program Memory
✓ Configuration Bits
✓ ID Locations
✓ EEPROM Data
✓ Erase All before Program
✓ Enable Debug Mode

Figure 2.11: MPLAB ICD Options Dialog

2.6.1 Configuration Bits and Device Selection

This section of the ICD Options dialog allows you to set the various configuration bits on the PIC16F87X processors. Click the arrow and select from the list.

Table 2.2: Configuration Bits and Device Selection

Item	Options
Device	The PIC16F877 device should be shown, as selected in the Development Mode dialog. If not, select *Options > Development Mode*, select the correct device, and click **OK**. Then use the Project menu item *Project > Save* to save your project.
Oscillator	RC is used in this tutorial. Check the MPLAB ICD Demo Board to make sure the oscillator jumper JP1 is correctly placed for the RC OSC option (see item 8 in Figure 1.3). For more information on the oscillator, see Section A.3.8.
Watchdog Timer	For this tutorial, the Watchdog Timer (WDT) should be off/disabled.
Power Up Timer	For this tutorial, the Power-Up Timer (PWRT) should be on/enabled.
Brown Out Detect	For this tutorial, the Brown-Out Detect (BOD) should be off/disabled.
Low Voltage Program	Low voltage ICSP programming should be disabled when using the MPLAB ICD. This means that you may use RB3 as digital I/O and you must use HV on $\overline{\text{MCLR}}$ for programming.
Code Protect Data EE	Turn off code protection for this tutorial.
Flash Memory Write	No memory will be written to EECON for this tutorial.
Code Protect	Turn off code protection for this tutorial.

2.6.2 IDs and Checksum

The checksum for the data and the ID code are displayed here. For this tutorial, Use Checksum as ID by selecting the checkbox.

2.6.3 Voltages

Allows you to check the V_{DD} and V_{PP} voltages on the target application by clicking the **Update** button.

MPLAB ICD develops its needed $V_{PP} \approx 13V$ from the target board's V_{DD} through use of a switching boost converter.

2.6.4 Program Options

The program address range (Start Address and End Address) is the range of program or data memory that will be read, programmed or verified.

Make sure that all checkboxes under Program Options are checked. This means that all memory, ID, and configuration bits will be programmed. Also, all memory will be erased before programming and Debug Mode will be enabled.

Click **Def. Addr** to set the address range to the maximum program memory available based on the device you selected.

2.6.5 Function Buttons

During the tutorial you will click these buttons to perform the assigned function on the PIC16F87X in the MPLAB ICD Header or Demo/Target board.

2.7 Programming the PIC16F877

Click the **Program** button to program `tut877.hex` and debug code into the PIC16F87X in the MPLAB ICD Header or MPLAB ICD Demo Board. Programming may take a couple of minutes. During programming, the Status box shows the current phase of the operation. When programming is complete, the Status box displays the message "Waiting for user command."

Note: The debug code is special code at 1F00h-1FFFh in the PIC16F877 that must be present to use the in-circuit debugging capabilities of the MPLAB ICD.

You can minimize or move the MPLAB ICD dialog, but do not close it. Closing the MPLAB ICD dialog will disable the ICD. To reenable the ICD, you will have to select *Options > Development Mode*. Select Editor Only and click **Apply**. Then select MPLAB ICD and click **OK** to reenable the ICD.

2.8 Setting Up the Demo Board

Before you begin debugging, make sure the MPLAB ICD Demo Board is set up as follows:

- the RC OSC option has been selected using jumper JP1.
- the DIP switch (SW3) has all switches in the ON position to connect all LEDs to their respective PORTC pin.

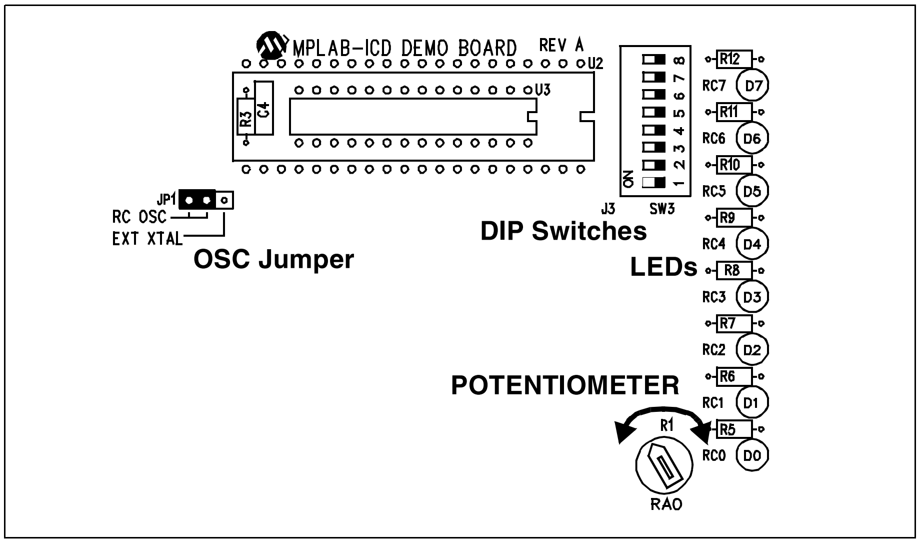


Figure 2.12: Setting Up the Demo Board

2.9 Running Tut877

The MPLAB ICD executes in real-time mode or in step mode.

- Real-time execution occurs when the PIC16F87X in the MPLAB ICD Header is put in MPLAB IDE's Run mode.
- Step mode execution can be accessed after the processor is halted.

Begin in real-time mode:

- Open the `tut877.asm` file for viewing (*File > Open*)
- Click the Run toolbar button or issue the *Debug > Run > Run* command

The Status bar at the bottom of the MPLAB IDE desktop should turn yellow. To change the colors that signify a program run, select *Options > Environment Setup* and click the **Colors** tab.

On the MPLAB ICD Demo Board, turn the arrow on the potentiometer (RA0) till it points to the DIP switch (SW3). Observe the LEDs. If the program were working correctly, you would see a binary representation of the voltage value across the potentiometer. However, an insidious bug has been placed in the Tut877 program! Of course, this will give us the opportunity to debug the code.

Click the Halt toolbar button or issue the *Debug > Run > Halt* command to stop the program execution. Reset the program by selecting *Debug > Run > Reset*.

2.10 Debugging Tut877

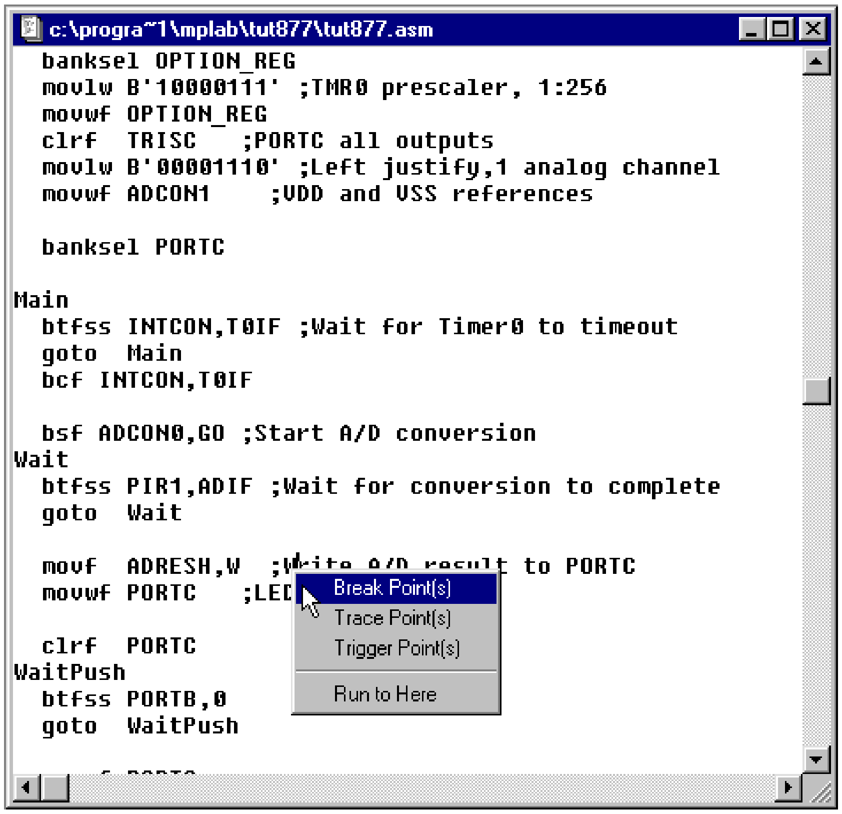
Any of the following could be preventing the Tut877 program from working:

- The A/D converter value is not being properly written to PORTC (LEDs)
- The A/D converter is not on or has not been set to convert
- A typo in the source code is causing the program to function improperly

To explore the first possibility, you will set a breakpoint at the line of the file that writes the value of A/D result to PORTC. Highlight or place the cursor on the following line of code from `tut877.asm`:

```
movf ADRESH,W ;Write A/D result to PORTC
```

Click on the right mouse button to access a shortcut menu. Select *Break Point(s)* from the shortcut menu as shown in Figure 2.13. This line will now be marked as a breakpoint. To change the colors that mark a breakpoint, select *Options > Environment Setup* and click the **Colors** tab.



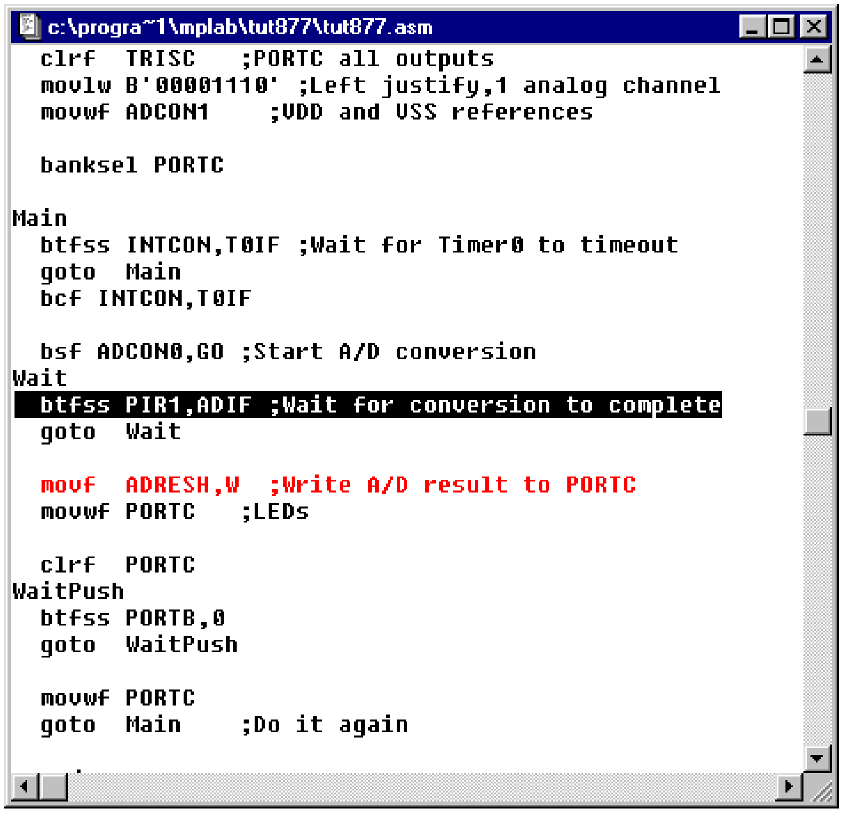
```
  banksel OPTION_REG
  movlw B'10000111' ;TMR0 prescaler, 1:256
  movwf OPTION_REG
  clrf  TRISC   ;PORTC all outputs
  movlw B'00001110' ;Left justify,1 analog channel
  movwf ADCON1    ;VDD and VSS references

  banksel PORTC

Main
  btfss INTCON,T0IF ;Wait for Timer0 to timeout
  goto  Main
  bcf INTCON,T0IF

  bsf ADCON0,GO ;Start A/D conversion
Wait
  btfss PIR1,ADIF ;Wait for conversion to complete
  goto  Wait

  movf  ADRESH,W  ;Write A/D result to PORTC
  movwf PORTC   ;LED

  clrf  PORTC
WaitPush
  btfss PORTB,0
  goto  WaitPush
```

Figure 2.13: Set Breakpoint

Click the Run toolbar button or issue the *Debug > Run > Run* command to run the program once again in real-time mode.

A breakpoint will stop a program's execution when the program executes the line marked as a breakpoint. However, our sample program is not halting. Therefore, halt it yourself now by clicking the Halt toolbar button or issuing the *Debug > Run > Halt* command.

Look at the source code (`tut877.asm`) window and notice where the program halted. Our sample program halted on one of the two lines in the Wait routine as shown in Figure 2.14. Based on the halt location and the fact that the program never reaches the breakpoint, we conclude that the problem is in the A/D conversion—the A/D flag for conversion complete is not being set.



```
  clrf  TRISC   ;PORTC all outputs
  movlw B'00001110' ;Left justify,1 analog channel
  movwf ADCON1    ;VDD and VSS references

  banksel PORTC

Main
  btfss INTCON,T0IF ;Wait for Timer0 to timeout
  goto  Main
  bcf INTCON,T0IF

  bsf ADCON0,GO ;Start A/D conversion
Wait
  btfss PIR1,ADIF  ;Wait for conversion to complete
  goto  Wait

  movf  ADRESH,W  ;Write A/D result to PORTC
  movwf PORTC   ;LEDs

  clrf  PORTC
WaitPush
  btfss PORTB,0
  goto  WaitPush

  movwf PORTC
  goto  Main    ;Do it again
```

Figure 2.14: Program Halted

A/D conversion initialization and setup occurs at the beginning of the program. To check out this code, first reset the program by selecting *Debug > Run > Reset*. The first instruction after Start should be highlighted.

Open a new watch window so you can watch the A/D register values change as the program executes. Select *Window > Watch Windows > New Watch Window*. The Add Watch Symbol dialog will open, with the Watch_1 new watch window behind it. Add the symbols ADCON0 and ADCON1 as shown in Figure 2.15.

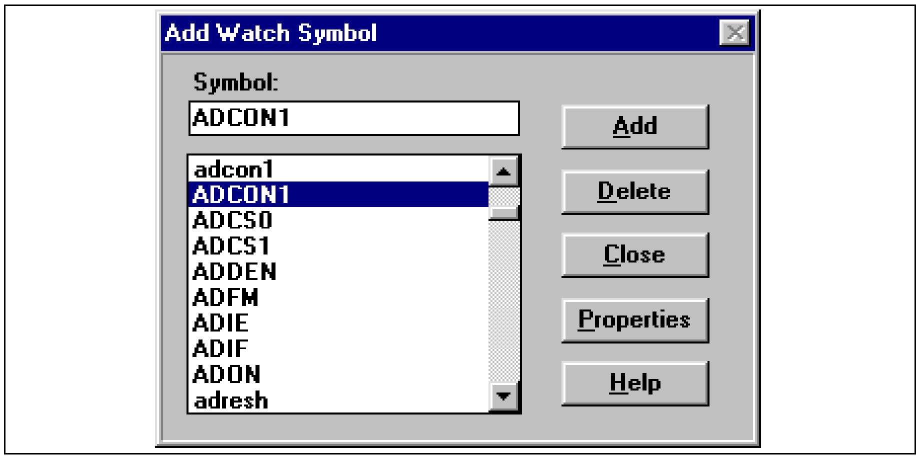
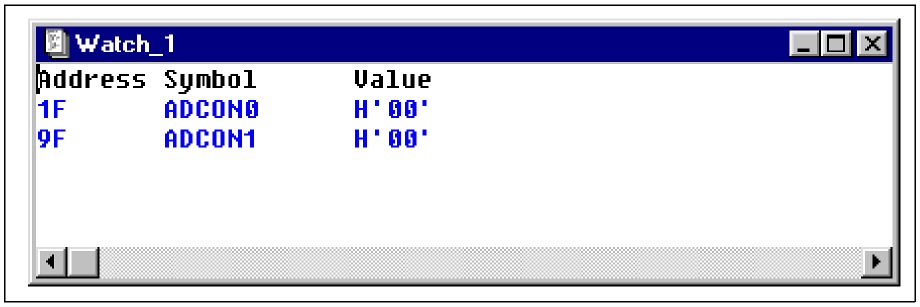


Figure 2.15: Adding Watch Symbols

Click **Close** when finished. The selected symbols should now be visible in the watch window as shown in Figure 2.16.



```
Address Symbol      Value
1F      ADCON0      H'00'
9F      ADCON1      H'00'
```

Figure 2.16: Watch Window

In the MPLAB ICD dialog, set the upload options to Minimum & Watch Windows. Single stepping and halting may be slower now, but this setting allows us to see the Watch windows for debugging. Close or minimize the MPLAB ICD Options dialog.

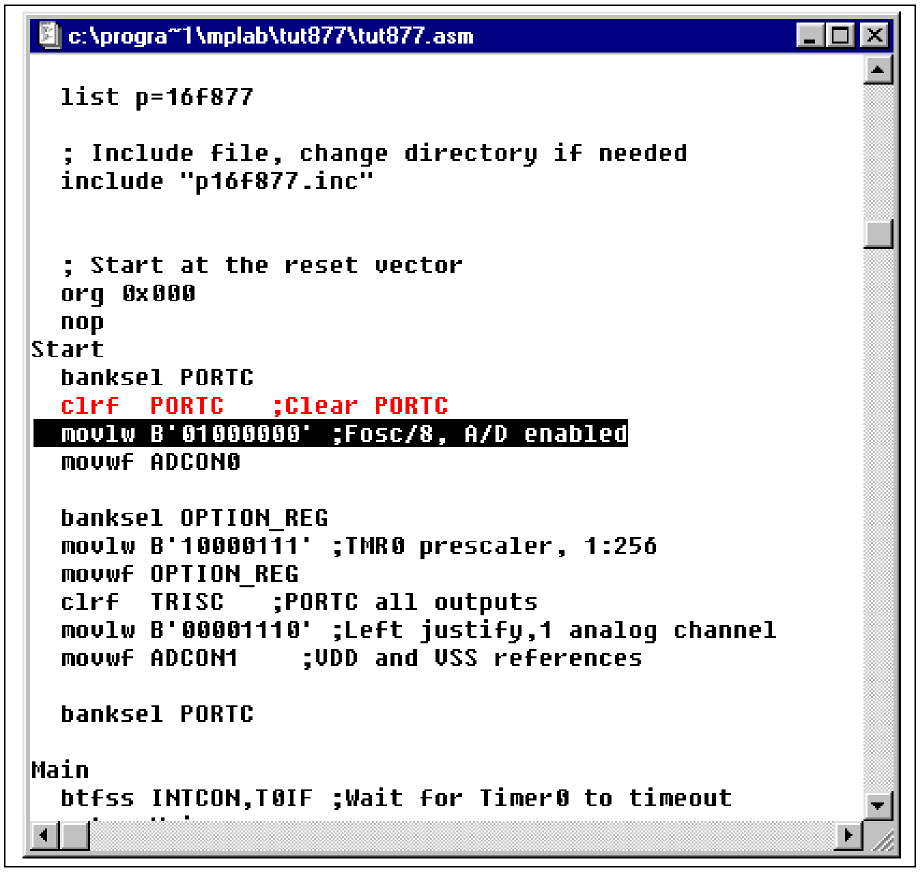
In the `tut877.asm` source code, set a breakpoint at the second instruction after Start. Again, highlight or place the cursor on the following line of code from `tut877.asm`:

```
clrf PORTC ;Clear PORTC
```

Click on the right mouse button to access a shortcut menu. Select *Break Point(s)* from the shortcut menu. This line will now be marked as a breakpoint.

Finally, click the Run toolbar button or issue the *Debug > Run > Run* command to run the program in real-time mode.

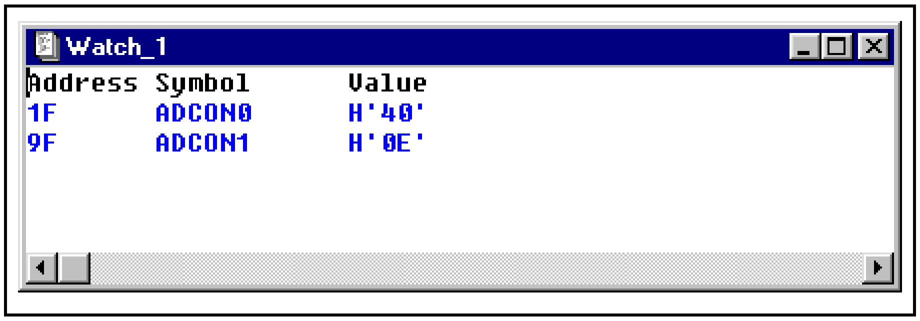
This time the program will stop after it executes the breakpoint line of code, and the instruction after the breakpoint will be highlighted as shown in Figure 2.17.



```
  list p=16f877

  ; Include file, change directory if needed
  include "p16f877.inc"


  ; Start at the reset vector
  org 0x000
  nop
Start
  banksel PORTC
  clrf  PORTC    ;Clear PORTC
  movlw B'01000000' ;Fosc/8, A/D enabled
  movwf ADCON0

  banksel OPTION_REG
  movlw B'10000111' ;TMR0 prescaler, 1:256
  movwf OPTION_REG
  clrf  TRISC    ;PORTC all outputs
  movlw B'00001110' ;Left justify,1 analog channel
  movwf ADCON1     ;VDD and VSS references

  banksel PORTC

Main
  btfss INTCON,T0IF ;Wait for Timer0 to timeout
```

Figure 2.17: Program Halted After Break

Now single step by clicking on the Step toolbar button or issuing the *Debug > Run > Step* command. Single step twice and then examine the values of the registers ADCON0 and ADCON1 in the Watch window. You should notice that ADCON0 has a value of 40 hex as shown in Figure 2.18.



```
Address Symbol      Value
1F      ADCON0      H'40'
9F      ADCON1      H'0E'
```

Figure 2.18: Updated Watch Window

This corresponds to the binary value designated in the program, but is this value correct? On examining the *PIC16F87X Data Sheet* (DS30292) section on A/D, you will see that the last bit should be a one, not a zero, to turn the A/D module on.

To fix this bug, change:

```
movlw B'01000000' ;Fosc/8, A/D enabled
```

to

```
movlw B'01000001' ;Fosc/8, A/D enabled
```

Select *File > Save* to save your changes and select *Project > Make Project* to rebuild the project.

A message will tell you the program has been rebuilt and you must reprogram the ICD in order for your changes to take effect. Click the **Program** button in the MPLAB ICD dialog to reprogram the ICD to reflect your change.

When the MPLAB ICD dialog's Status box indicates that it is waiting for your next command, you are ready to run your program again.

Click the **Run** toolbar button or issue the *Debug > Run > Run* command to run the program in real-time mode. Some of the LEDs should now be lit. Turn the potentiometer (RA0) to change the value displayed on the LEDs.

The source code in this tutorial contained only one bug. However, real code may have more. Using the MPLAB ICD and MPLAB IDE debugging functions, you can successfully find and fix the bugs in your code.

2.11 Tut877 Main Routine

The main routine of `tut877.asm` (Figure 2.19) begins by configuring PORTC, the A/D module and Timer0. It then waits for a Timer0 overflow to start the A/D conversion of the value from the potentiometer. When the conversion is complete, the value is displayed on the LEDs, and the program loops back to wait for another Timer0 overflow to start another A/D conversion.

For more information on A/D module operation and a list of related application notes, refer to the *PICmicro Mid-Range MCU Family Reference Manual* (DS33023).

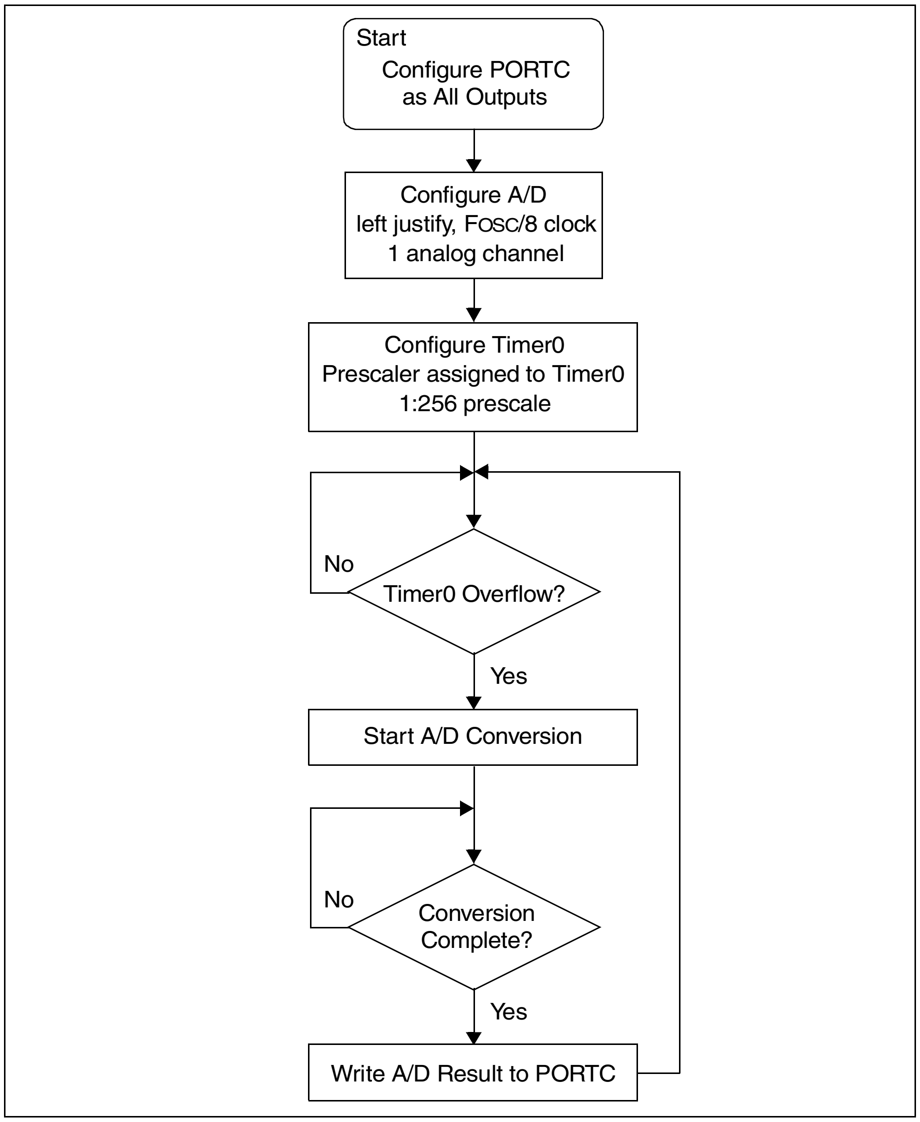

Start
Configure PORTC as All Outputs
Configure A/D left justify, FOSC/8 clock 1 analog channel
Configure Timer0 Prescaler assigned to Timer0 1:256 prescale
No
Timer0 Overflow?
Yes
Start A/D Conversion
No
Conversion Complete?
Yes
Write A/D Result to PORTC

Figure 2.19: Program Flow Chart

2.12 Tut877 Corrected Source Code

This is a functional version of `tut877.asm`.

```
;*****************************************************************
;*  TUT877.ASM
;*****************************************************************
;*  Microchip Technology Incorporated
;*  16 December 1998
;*  Assembled with MPASM V2.20
;*****************************************************************
;*  This program configures the A/D Module to convert on
;*  A/D channel 0 (the potentiometer) and display the
;*  results on the LEDS on PORTC.  Make sure that the DIP
;*  switch SW3 has all switches in the ON position.
;*****************************************************************

     list p=16f877

     ; Include file, change directory if needed
     include <p16f877.inc>

     ; Start at the reset vector
     org     0x000
     nop
Start
     banksel   PORTC
     clrf      PORTC          ;Clear PORTC
     movlw     B'01000001'    ;Fosc/8, A/D enabled
     movwf     ADCON0

     banksel   OPTION_REG
     movlw     B'10000111'    ;TMR0 prescaler, 1:256
     movwf     OPTION_REG
     clrf      TRISC          ;PORTC all outputs
     movlw     B'00001110'    ;Left justify,1 analog channel
     movwf     ADCON1         ;VDD and VSS references

     banksel   PORTC

Main
     btfss     INTCON,T0IF    ;Wait for Timer0 to timeout
     goto      Main
     bcf       INTCON,T0IF

     bsf       ADCON0,GO      ;Start A/D conversion
Wait
     btfss     PIR1,ADIF      ;Wait for conversion to complete
     goto      Wait
     movf       ADRESH,W       ;Write A/D result to PORTC
     movwf      PORTC          ;LEDs

     clrf       PORTC
WaitPush
     btfss      PORTB,0
     goto       WaitPush

     movwf      PORTC
     goto       Main           ;Do it again

     end
```

NOTES:

Chapter 3. General Setup

3.1 Introduction

This chapter describes how to get the hardware and software for the MPLAB ICD up and working.

3.2 Highlights

Topics covered in this chapter:

- Running MPLAB ICD
- Setting Up the Development Mode
- Setting Up the MPLAB ICD Dialog
- Setting Up the ICD Options Dialog
- Using MPLAB Projects

3.3 Running MPLAB IDE

After installing MPLAB IDE, invoke it by executing the file `mplab.exe`.

For more information on installing and using MPLAB IDE, refer to the *MPLAB IDE User's Guide* (DS51025) and the included file `readme.lab` in the MPLAB IDE install directory.

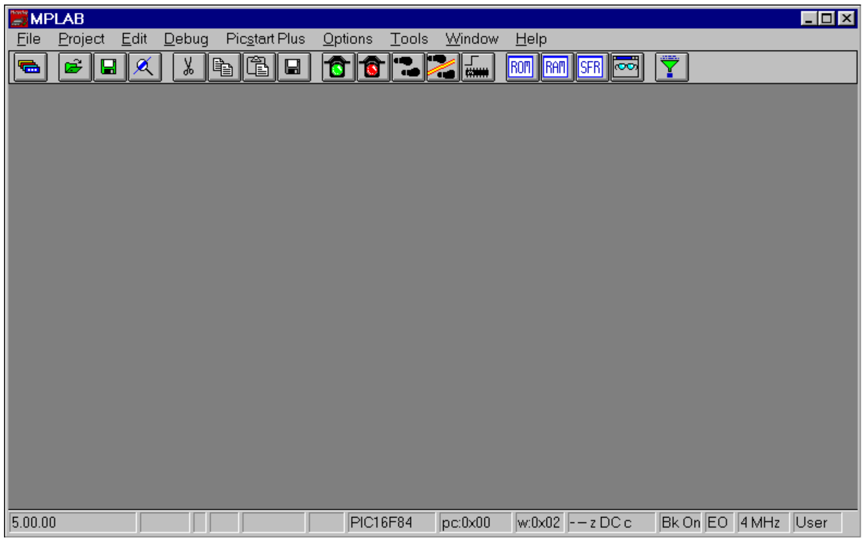


Figure 3.1: MPLAB IDE

3.4 Setting Up the Development Mode

Use the Development Mode dialog to select MPLAB ICD for use with the MPLAB IDE software.

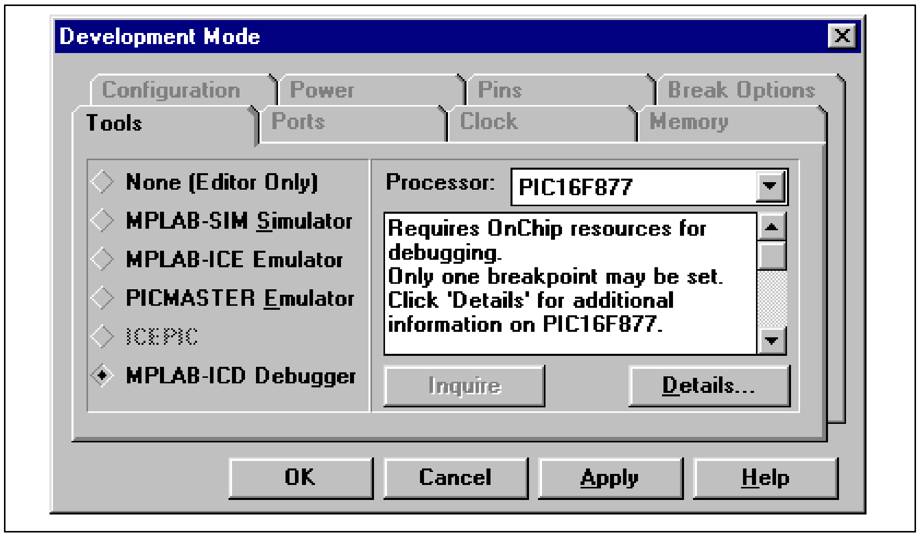
1. Select _Options > Development Mode_. Select the MPLAB ICD Debugger development mode, and select the processor you are going to use. Click **OK**.

   Below the Processor is a brief description of any limitations that exist for this processor on the MPLAB ICD. A more detailed description of limitations may be viewed by clicking on **Details**.

2. The MPLAB ICD dialog will appear. You will use this dialog to set up the debugger.

Note: You can move the MPLAB ICD dialog, but do not close it.

3. If MPLAB IDE cannot find the MPLAB ICD, check your hardware connections (Section 1.7) and then go to step 1. If you still cannot connect, see the next section or troubleshooting chapter (Chapter 5).



Processor: PIC16F877

Requires OnChip resources for debugging.
Only one breakpoint may be set.
Click 'Details' for additional information on PIC16F877.

Figure 3.2: Development Mode Dialog Box

You may also access the Development Mode dialog from the Edit Project dialog by clicking **Change** next to the Development Mode item. See Chapter 2 for more information on setting up projects.

3.5 Setting Up the MPLAB ICD Dialog

The MPLAB ICD dialog is always open when the ICD is enabled. You can minimize or move the MPLAB ICD dialog, but do not close it. Closing this dialog disables the debugger. You must then select *Options > Development Mode*, select None (Editor Only), and click **Apply**. Then, select MPLAB ICD, and click **OK** to reenable the debugger.

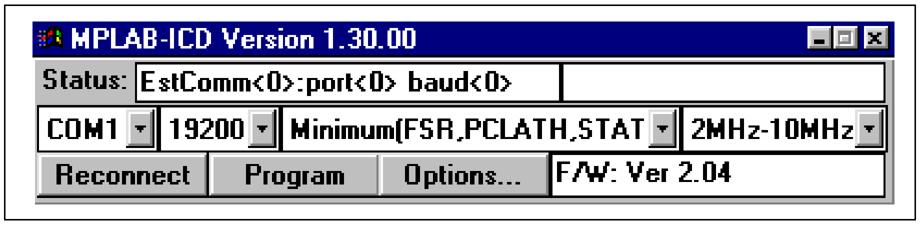

MPLAB-ICD Version 1.30.00
Status: EstComm<0>:port<0> baud<0>
COM1 | 19200 | Minimum(FSR,PCLATH,STAT | 2MHz-10MHz
F/W: Ver 2.04

Figure 3.3: MPLAB ICD Dialog

The MPLAB ICD dialog has the following settings and buttons.

Table 3.1: MPLAB ICD Dialog

Item	Options
Status	The Status bar displays the executed MPLAB ICD function and the status.
COM Port	Select the COM port (COM1, COM2, COM3 or COM4) for MPLAB ICD communications.
Baud Rate	Select the COM port baud rate for MPLAB ICD communications.
Upload Options	The options for the amount of data uploaded are: • Minimum (FSR, W, Status, PCLATH) (Very Fast) • SFRs only (Fast) • Minimum and Watch windows (Pretty Fast) • All Registers (Slow) If you are using breakpoints and single step, you can upload selected data to improve speed. The first selection has less than a second delay, where as the last selection will have a delay of approximately 2 seconds. Data will be uploaded on a single step, a breakpoint or a halt.

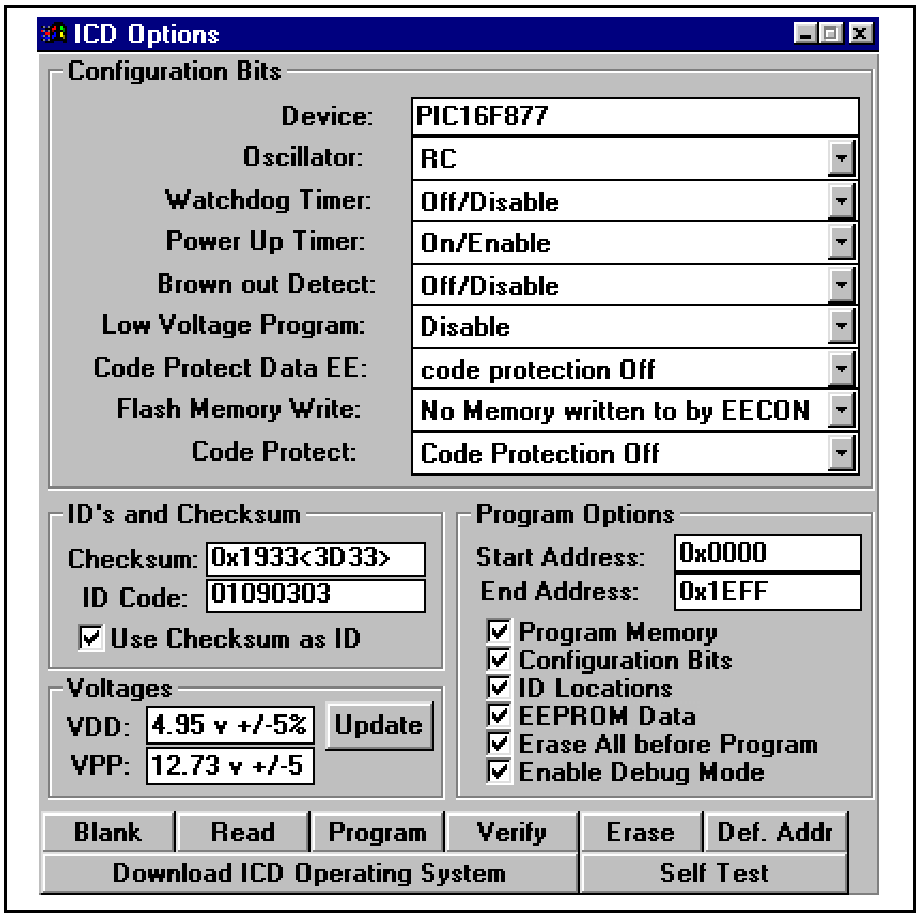
Table 3.1: MPLAB ICD Dialog (Continued)

Item	Options
Operating Frequency Range	Select the operation frequency range of the MPLAB ICD. Options are: • 32 kHz – 500 kHz • 500 kHz – 2 MHz • 2 MHz – 10 MHz • 10 MHz – 20 MHz
Note: If you are using a version of MPLAB IDE previous to 5.20, you need to change development modes to save the above settings. Select *Options > Development Mode*, select None (Editor Only), and click **Apply**. Then select MPLAB ICD Debugger, and click **OK**.	
Reconnect	If you had to change the COM port or baud rate, click Reconnect to reestablish communications using the new values.
Program	Programs the device.
Options	Opens the ICD Options dialog (see next section).

3.6 Setting Up the ICD Options Dialog

The ICD Options dialog is accessed by clicking **Options** in the MPLAB ICD dialog. The ICD Options dialog can be closed or minimized without disabling the ICD.

The ICD Options dialog (Figure 3.4) has all the programming and setup functions for the MPLAB ICD. These functions include basic programmer functions like blank check, read, program, verify, erase, and the debugger functionality.


ICD Options

Configuration Bits

Device:	PIC16F877
Oscillator:	RC
Watchdog Timer:	Off/Disable
Power Up Timer:	On/Enable
Brown out Detect:	Off/Disable
Low Voltage Program:	Disable
Code Protect Data EE:	code protection Off
Flash Memory Write:	No Memory written to by EECON
Code Protect:	Code Protection Off

ID's and Checksum

Checksum: 0x1933<3D33>
ID Code: 01090303
✓ Use Checksum as ID

Voltages

VDD: 4.95 v +/-5%
VPP: 12.73 v +/-5

Program Options

Start Address: 0x0000
End Address: 0x1EFF
✓ Program Memory
✓ Configuration Bits
✓ ID Locations
✓ EEPROM Data
✓ Erase All before Program
✓ Enable Debug Mode

Figure 3.4: MPLAB ICD Options Dialog

3.6.1 Configuration Bits and Device Selection

This section of the dialog allows you to set the various configuration bits on the PIC16F87X processors. To select, click on the arrow to see the list of choices, and then select the correct item.

Table 3.2: Configuration Bits and Device Selection Options

Item	Options
Device	This box shows the processor you selected in the Development Mode dialog. If you change the device here, any open MPLAB IDE projects will close and program memory, configuration bits, and IDs will be cleared.
Oscillator	RC, LP, XT or HS
Watchdog Timer	Select On/Enable or Off/Disable. Usually should be disabled for debugging.
Power Up Timer	Select On/Enable or Off/Disable. Usually should be disabled for debugging.
Brown Out Detect	Select On/Enable or Off/Disable. Must be disabled for debugging.
Low Voltage Program	Select low-voltage programming function or use RB3 as digital I/O and use HV on $\overline{\text{MCLR}}$ for programming. Low-voltage programming must be disabled for debugging.
Code Protect Data EE	Select On/Enable or Off/Disable. Must be disabled for debugging.
Flash Memory Write	Select whether to allow unprotected program memory to be written by EECON control. Usually should be disabled for debugging.
Code Protect	Select the desired range. Must be disabled for debugging.

3.6.2 IDs and Checksum

Table 3.3: IDs and Checksum Selection Options

Item	Options
Checksum	Displays the checksum for the data.
ID Code	Displays the ID code.
Use Checksum as ID	To use the checksum as the ID, select the checkbox.

3.6.3 Voltages

Table 3.4: Voltage Selection Options

Item	Options
VDD	Displays the current value of the VDD voltage. MPLAB ICD develops its needed VPP ≈ 13V from the target board's VDD through use of a charge pump.
VPP	Displays the current value of the VPP voltage.
Update	Allows you to check the current value of the VDD and VPP voltages on the target application.

3.6.4 Program Options

Select or clear the checkbox next to the item to select or deselect the memory areas for the programmer functions. For example, if you want to read program memory only, select Program Memory and clear the checkboxes for the other memory types.

Table 3.5: Program Selection Options

Item	Options
Start Address, End Address	The starting and ending address range in program memory for programming, reading or verification. **Note:** The address range does not apply to the Erase function. The Erase function will erase all data on the PIC16F87X. The default program address range is set to the maximum program memory available based on the device you selected. To identify file register and program memory locations that the ICD uses in the target device, see Section 1.4.
Configuration Bits	Program the configuration bits. You can set the configuration bits in the top portion of the ICD Options dialog.
ID Locations	Program the ID locations. You can set the ID locations in the IDs and Checksum area of the ICD Options dialog.
EEPROM Data	For devices with data EEPROM, program the data memory from the data in the EEPROM Memory window.
Erase All before Program	If this option is checked and you click **Program**, all program memory will be erased before programming begins. This results in faster programming times. **Do not use this option with VDD voltages below 4.5.**

Table 3.5: Program Selection Options (Continued)

Item	Options
Enable Debug Mode	Select this option to program debug code into a device. The debugging operations will be enabled each time you click **Program**. If you are only using MPLAB ICD to program a part, clear the Enable Debug Mode check box. The debug code will not be downloaded to program memory and MPLAB ICD will not be enabled for debug operations.

3.6.5 Function Buttons

Click the **Function** buttons to perform the assigned function on the PIC16F87X in the MPLAB ICD Header. If you have specified an address range in the Program Options section, the assigned function will be performed on the specified memory range and type only (except for the erase function).

Table 3.6: Function Button Selection Options

Item	Options
Blank	Checks to see if device is blank.
Read	Reads memory areas specified under Program Options: program memory, configuration bits, ID locations, and/or EEPROM data.
Program	Programs memory areas specified under Program Options: program memory, configuration bits, ID locations, and/or EEPROM data. Also downloads debug code in program memory if Enabled Debug Mode is checked.
Verify	Verifies programming of memory areas specified under Program Options: program memory, configuration bits, ID locations, and/or EEPROM data.
Erase	Erases all data on the PIC16F87X including memory, ID, and configuration bits. **Do not use this option with VDD voltages below 4.5.**
Download Operating System	Downloads an updated version of the MPLAB ICD operating system (firmware) to the MPLAB ICD's FLASH processor. See also Section 4.8.
Self Test	Performs a self-test on the MPLAB ICD hardware. **Note:** If there is no PIC16F87X on the MPLAB ICD Header or target/Demo Board or if the device is not programmed with debug code, the self test will indicate that the debug module doesn't exist.

Table 3.6: Function Button Selection Options (Continued)

Item	Options
Def.Addr	Restores the Start and End addresses to the defaults for the device. End addresses with Debug Mode enabled: • PIC16F870/871/872:0x6DF • PIC16F873/874:0xEDF • PIC16F876/877:0x1EFF End address with Debug Mode disabled: • PIC16F870/871/872:0x7FF • PIC16F873:0xFFF • PIC16F876/877:0x1FFF

3.7 Using MPLAB Projects

MPLAB IDE is the host software for the MPLAB ICD and the MPLAB SIM simulator. It functions as a sophisticated debugging tool, providing access to RAM, ROM, EEPROM, and a variety of other debug functions.

Note: If you do not put your source files into a project, MPLAB IDE cannot debug properly.

3.7.1 MPLAB IDE Project Features

Developing and debugging code in the MPLAB IDE environment is based on projects. Although emulation can be performed without having a project open, projects have the following advantages:

- Single or multiple source files can be easily built and maintained.
- Symbolic debugging is available.
- The debugging environment can be saved for later use.

Some of the information that is retained with a project is:

- Development mode and processor
- Source files associated with the project
- Name of the final PICmicro MCU executable file
- Open windows and their sizes and positions
- Named break settings
- Configuration bit settings

3.7.2 Creating a Project

Select *Project > New Project* to create a project and open the Edit Project dialog. The project will contain information about your source, object, and other files, as well as a variety of important project settings.

3.7.3 Saving a Project

To save your current project to retain the values for later use or to use as a backup or default as you continue with your debugging, click **OK** in the Edit Project dialog. Then select *Project > Save Project* to save your project.

Development Mode

The selected development mode is retained with the project information. To change the development mode for a project, follow these steps:

- Open the project. The previously used development mode will be selected.
- Change the development mode by selecting *Options > Development Mode* to access the Development Mode dialog. Select the development mode and click **OK**.
- Select *Project > Save Project* to save the project.

For more information on creating and using projects, refer to the *MPLAB IDE User's Guide* (DS51025).

Chapter 4. Basic Functions

4.1 Introduction

This chapter discusses the basic operations of MPLAB ICD and MPLAB IDE debugging functions of the In-Circuit Debugger. For more information on general debugging features, refer to the *MPLAB IDE User's Guide (DS51025)*.

4.2 Highlights

Topics covered in this chapter:

- MPLAB ICD Operations
- Program Execution
- Breakpoints
- Modifying Functions
- Loading a Hex File for Debugging
- Upgrading the MPLAB ICD Operating System (Firmware)

4.3 MPLAB ICD Operations

The MPLAB ICD is a programmer for the PIC16F87X family, as well as an in-circuit debugger. It programs hex files into the PIC16F87X and offers basic debugging features like real-time code execution, stepping, and breakpoints. Its debug feature is built inside the PIC16F87X and activated by programming the Debug Code into the target processor. It has limited functions when compared to a full-featured in-circuit emulator (Section 1.4), but provides cost-effective functions to debug and program applications for a PIC16F87X.

To enable in-circuit debugging, the Debug Code (residing in the microcontroller in the MPLAB ICD Module) is programmed into the target PIC16F87X. The code is an MPASM module that will be programmed into the PIC16F87X automatically by MPLAB IDE. This code will reside at the end of program memory. For example, it will reside in 0x1F00 to 0x1FFF of the PIC16F877 processor (see Section 1.4).

4.4 Program Execution

The MPLAB ICD executes in real-time mode or in step mode.

- Real-time execution occurs when the PIC16F87X in the MPLAB ICD Header or Demo Board/target board is put in the MPLAB IDE's Run mode.
- Step mode execution can be accessed after the processor is halted.

4.4.1 Real-Time Execution

When the MPLAB ICD is run in real time, instructions execute just as the processor would without the debugger. The PIC16F87X executes in real time until a breakpoint halts the debugger or until the HALT function is manually executed.

To execute in real time, click the Run toolbar button or issue the *Debug > Run > Run* command. The Debug toolbar provides Run, Halt, and Step buttons for controlling the debugger. While in the run mode, register displays on the screen will not update.

4.4.2 Step-Mode Execution

Step-Mode Execution occurs when you single step the processor with the *Debug > Run > Step* command. Step-mode execution allows you to step through the code one instruction at a time, watch the program flow, and see the register contents at each instruction (as set in the dialog box).

Note: With the design of the MPLAB ICD, it is IMPOSSIBLE to single step through an interrupt. Due to hardware restrictions, it is IMPOSSIBLE to jump to the interrupt vector memory location while doing a single step function.

Debug > Run > Animate automatically single steps the processor until you halt it. To view the changing registers in the Special Function Register window or the Watch windows, use Animate. Animate runs slower than the Run function.

4.5 Breakpoints

A breakpoint is a state where the processor halts after a certain condition is met. The MPLAB ICD provides the followings ways to set a breakpoint:

- Break on Address Match
- Break on User Halt

4.5.1 Break on Address Match

The Debug function of the PIC16F87X allows one breakpoint to be set. This breakpoint can be at any program memory address location. The processor breaks after the instruction is executed. For example, if a breakpoint is set at address 005Ah, the processor breaks after executing the instruction at address 005Ah.

To set a breakpoint, select *Debug > Break Settings*. You can also set a breakpoint by selecting the source code address in Program Memory or selecting the line of code in the source code window, clicking the right mouse button to access a shortcut menu, and selecting *Break Point(s)* from the shortcut menu.

Note: MPLAB ICD only supports one breakpoint. If you set multiple breakpoints, they will be shown as enabled in the Breakpoint Settings Window, but only the last breakpoint set will actually be enabled, whether it was set through the Breakpoint Settings or by using the right mouse button.

4.5.2 Break on User Halt

The PIC16F87X executes until you click the **Halt** button or select *Debug > Run > Halt.*

4.6 Modifying Functions

If you need to modify the program being debugged, simply assemble a new hex file and download it. You can also use the Modify dialog to edit program memory before downloading it into the PIC16F87X with the **Program** button. If you do this, the object code generated by your source code will not match the code in the PIC16F87X.

You can use *Window > Modify* to change a register, words of program memory or data in the EEPROM area. Select *Window > File Registers* and *Window > Special Function Registers* to view internal data registers.

4.7 Loading a Hex File for Debugging

If you have a hex code file that you would like to program into your device for debugging (i.e., you have used a tool other than the MPLAB IDE to generate your hex code), you may load the hex code to the MPLAB IDE using the *File > Import > Import to Memory* command. To identify file register and program memory locations that ICD uses in the target device, see Section 1.4. Your code should be limited accordingly.

Note: If you need to specify values for the EEPROM data memory in your source code, use a starting address of 0x2100.

Once the hex file is imported, you can use the Windows menu to open the following windows:

- Program Memory
- Special File Registers
- File Registers
- EEPROM Memory

4.8 Upgrading the MPLAB ICD Operating System (Firmware)

MPLAB ICD now supports downloadable firmware starting with Firmware v2.02 and MPLAB IDE v5.00.

After upgrading to the PIC16LF876, future downloads can be performed through the MPLAB ICD software. If operation below 4.5V is not required in the target application, a PIC16F876 can be substituted for the PIC16LF876.

See engineering tech. note ETN-21D on the our web site (http:\\www.microchip.com) for more information on upgrading.

Operating System Downloads After System Upgrade

To perform a download of a new MPLAB ICD operating system after system upgrade, do the following:

- From the MPLAB ICD dialog, click **Options** to open the ICD Options dialog.
- In the ICD Options dialog, click **Download ICD Operating System**. The Open Existing File dialog will appear.
- Select the version of the operating system (firmware) you wish to download from the list. The file name format is icd*.obj, where * is the operating system version number. Click **OK**.
- The selected operating system file will download to the MPLAB ICD.

Chapter 5. Troubleshooting

5.1 Introduction

This section describes some common problems associated with running the MPLAB ICD and the steps to follow to resolve those problems.

5.2 Highlights

Topic covered in this chapter:

- Common Problems

5.3 Common Problems

Communications cannot be established with the MPLAB ICD.

If you cannot establish communications with the MPLAB ICD, follow these steps:

1. Make sure there is power to the MPLAB ICD Demo Board/target application. MPLAB ICD is powered by the MPLAB ICD Demo Board/target application. Also, if you are using the MPLAB ICD Demo Board, make sure the power supply has the correct rating (9.0V, 0.75A).
2. Check that the MPLAB ICD Header is plugged into the MPLAB ICD Demo Board/target application correctly; e.g., all MPLAB ICD Header pins are plugged into the socket and the MPLAB ICD Header is correctly oriented. Also, make sure you are using the correct stand-off if you are using the MPLAB ICD Demo Board.
3. Check that the PIC16F87X is plugged into the MPLAB ICD Header correctly; e.g., all pins are plugged into the socket and the PIC16F87X is correctly oriented.
4. Check that the connection between the MPLAB ICD and the host computer, via the RS-232 cable, is secure.
5. Check that the connection between the MPLAB ICD Header and MPLAB ICD Module, via the 9-inch modular cable, is secure.
6. Check the settings in the MPLAB ICD dialog. Make sure you selected the correct PICmicro MCU, COM port, and baud rate for your application.

The MPLAB IDE attempts to establish communication with the MPLAB ICD upon enabling the debugger. If communication cannot be established, no programming or debugging can occur. An error message appears if the attempt to establish communication fails. If a communication attempt fails, try again after correcting the problem or cancel.

SOLUTIONS:

- Make sure the power supply is connected and the LED on the MPLAB ICD Module is on and not blinking. If the LED on the MPLAB ICD Module is blinking, reset the MPLAB ICD Module by cycling the power to the application on and off. Then reselect the COM port for the MPLAB ICD in the MPLAB ICD dialog box. The LED should stop blinking.
- Try connecting the MPLAB ICD Module to a different serial port. If your PC has a 25-pin serial port, you will need a 25-pin to 9-pin serial port adapter.
- Make sure that a COM port is properly set up exclusively for use by the debugger. 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 the MPLAB IDE 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 [MPLAB ICD] 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:

Windows 95 requires 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. In Windows 95 this is done in the Control Panel. Click the **System** Icon. Next, click the **Device Manager** tab, and click **Port Settings**. If necessary, expand the Ports selection by clicking the "+" sign next to it. Double-click the I/O port that the MPLAB ICD is connected to. This is where you can set flow control to Hardware. To turn off the FIFO, click the **Advanced** button, deselect the Use FIFO box, and click **OK**.

If communications still cannot be established, contact Microchip Customer Support as described in *General Information*.

The MPLAB ICD Module returns a communications error on Reset, Single Step, Halt or Run.

When using the MPLAB ICD Demo Board, make sure that the oscillator select jumper is correctly set to RC or oscillator and that the part is correctly programmed for this selection. If using an oscillator, make sure that there is one in the socket. This problem usually results in the part being programmed OK and debug mode set OK, but when a reset, single step, halt or run command is executed, the MPLAB ICD Module returns a communications error.

Debug mode doesn't seem to be working.

1. Closing the MPLAB ICD dialog will disable the ICD. If you have closed this dialog, you will need to re-enable MPLAB ICD. Select *Options > Development Mode*, select None (Editor Only) and click **Apply**. Select MPLAB ICD and click **OK**.
2. Debug code must be programmed into the device before debug mode will work. Select **Enable Debug Mode** from the ICD Options dialog to program the device with debug code.

When single stepping, the program runs too slowly.

Check the upload options in the MPLAB ICD dialog. If you are uploading a large amount of data; e.g., all memory, you can expect about a 5 second delay between steps. To reduce the delay, select an option that uploads less data.

When single stepping, the program runs too quickly OR some registers are not updated.

Check the upload options in the MPLAB ICD dialog. If you are uploading a small amount of data, the time between steps will be less than if you were uploading all memory. If you are not getting all of the data uploaded that you expect, check that you have selected the correct type(s) of registers for upload.

When halting, single stepping or stopping on a breakpoint or SLEEP command, MPLAB IDE seems to lock up.

Check the upload options in the MPLAB ICD dialog. If you are uploading a large amount of data; e.g., all memory, you can expect about a 5 second delay between steps. To reduce the delay, select an option that uploads less data.

The following I/O pins are not functioning correctly: RB3, RB6 or RB7.

These pins are reserved for programming/debugging. Refer to Section 1.4 for more information.

One or more of my memory addresses (Program or GPR) is not correct.

Several GPR's and Program Memory Locations are reserved for debugging. Refer to Section 1.4 for more information.

Appendix A. Hardware Specifications

A.1 Introduction

The hardware included with the MPLAB ICD Evaluation Kit (DV164001) consists of the MPLAB ICD Module, MPLAB ICD Header, and MPLAB ICD Demo Board. The MPLAB ICD Module (DV164002) does not contain either the MPLAB ICD Header or the MPLAB ICD Demo Board.

A.2 MPLAB ICD Module and Header

The MPLAB ICD Module and MPLAB ICD Header are used to perform the ICD functions on the MPLAB ICD Demo Board or in the target application.

A.2.1 Voltage and Current Specifications

The MPLAB ICD Header is powered by the target application, from a 3.0V to 5.5V source. The current ratings for the ICD in different operating modes and voltages are as follows:

Operating Mode	5V	3V
Debug	35 mA (max)	20 mA (max)
Run	40 mA	25 mA
Verify	60 mA	50 mA
Program	60 mA	50 mA

A.2.2 Silkscreens and Schematics

This section contains the silkscreen and schematic diagrams for the MPLAB ICD Module and MPLAB ICD Header.

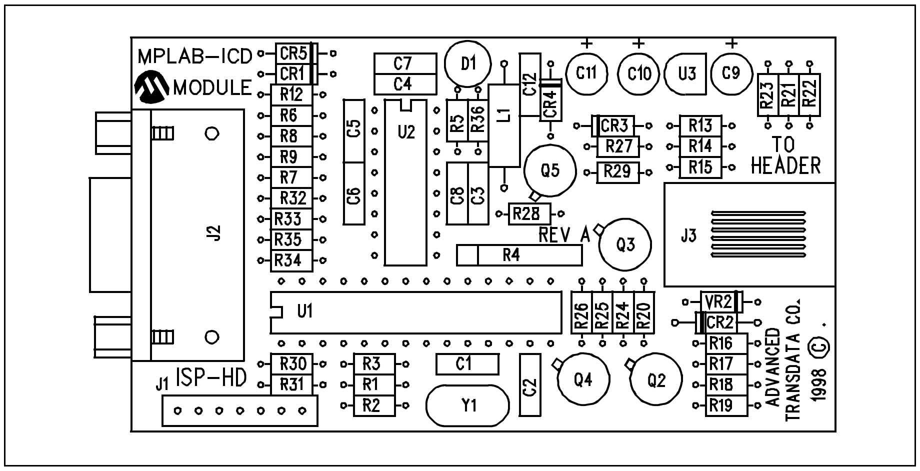


Figure A.1: MPLAB ICD Module Silkscreen

Figure A.2: MPLAB ICD Module Schematic, Part 1

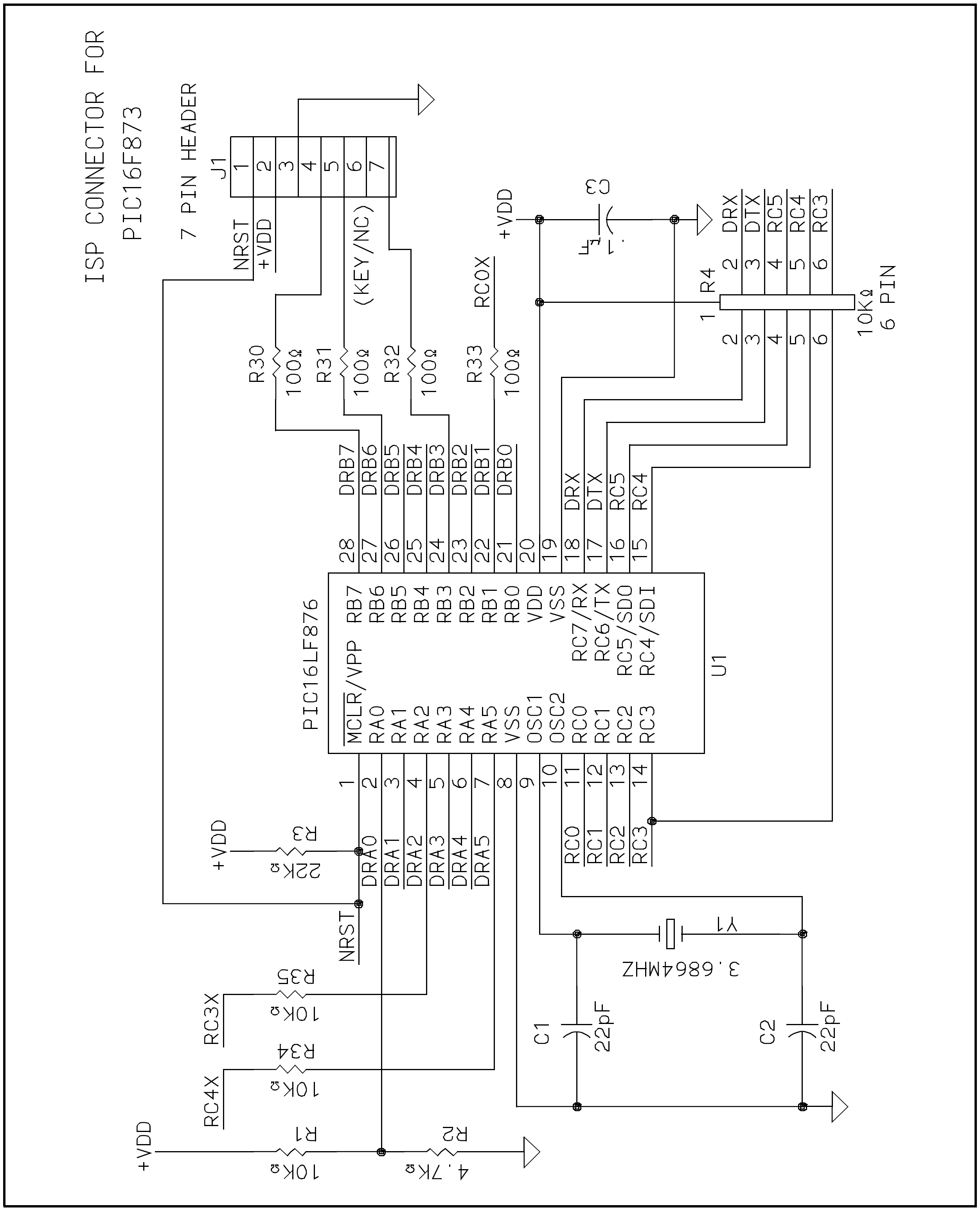


Figure A.3: MPLAB ICD Module Schematic, Part 2

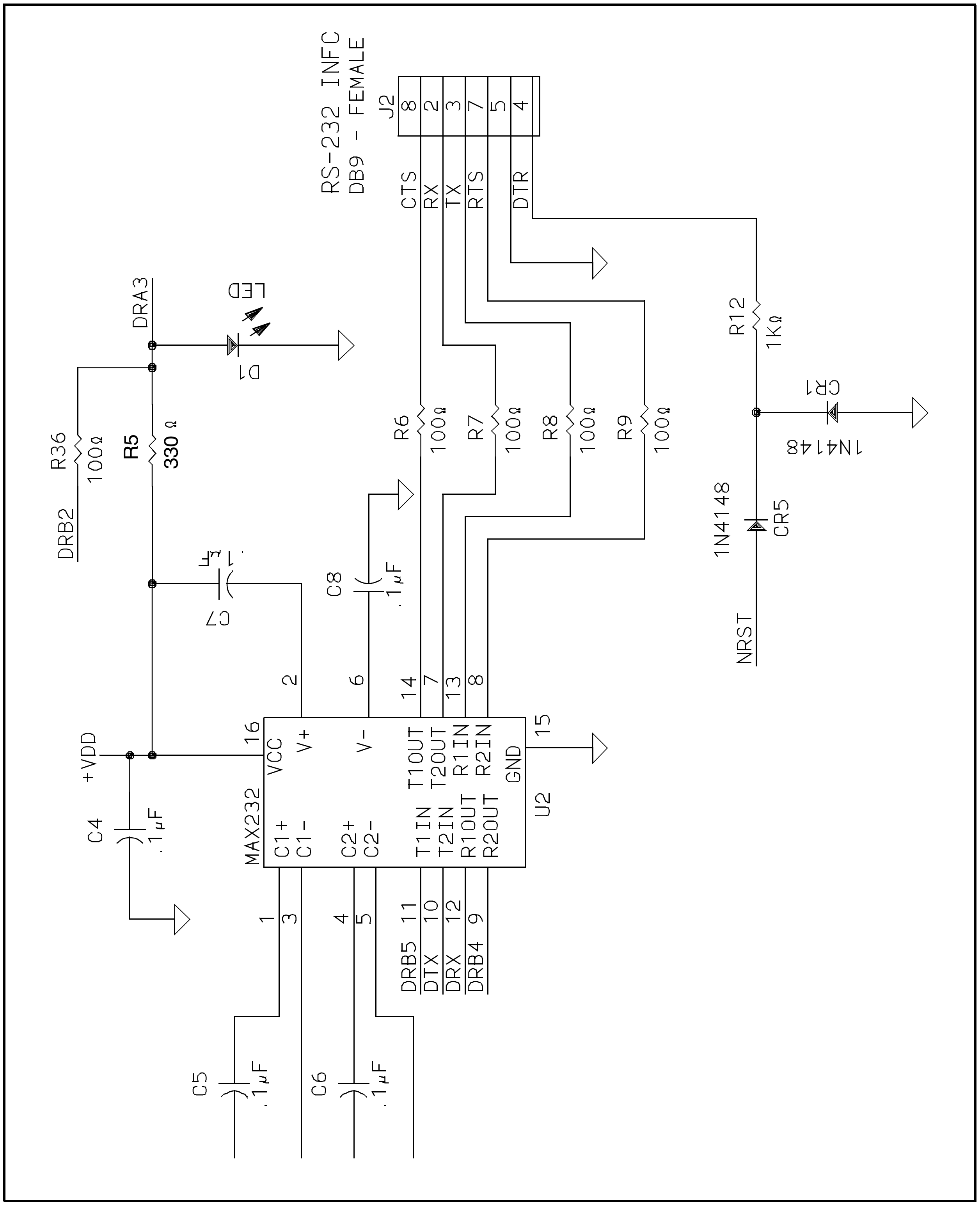


Figure A.4: MPLAB ICD Module Schematic, Part 3

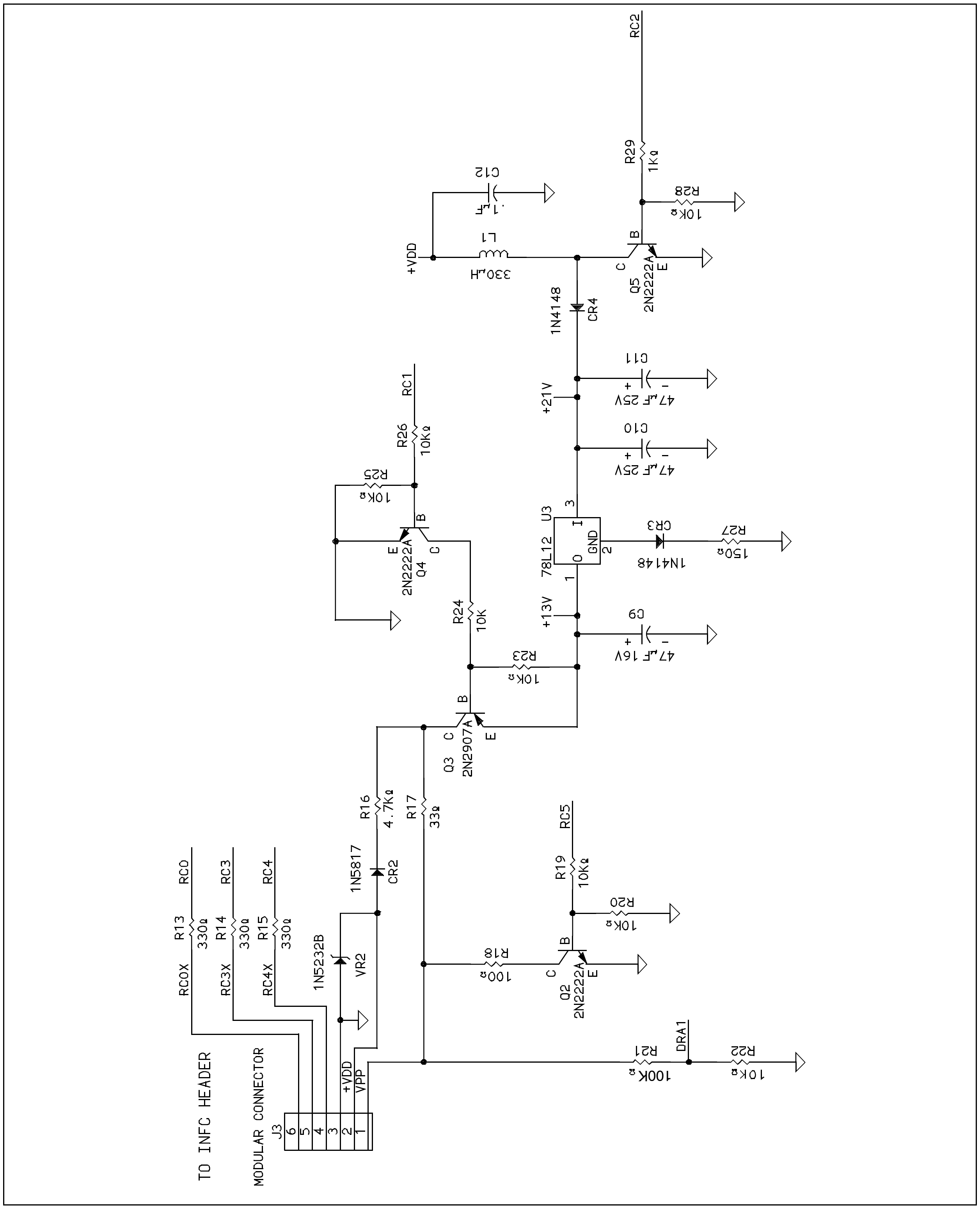
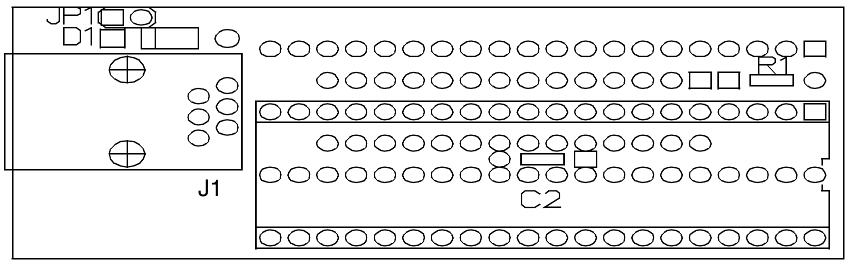
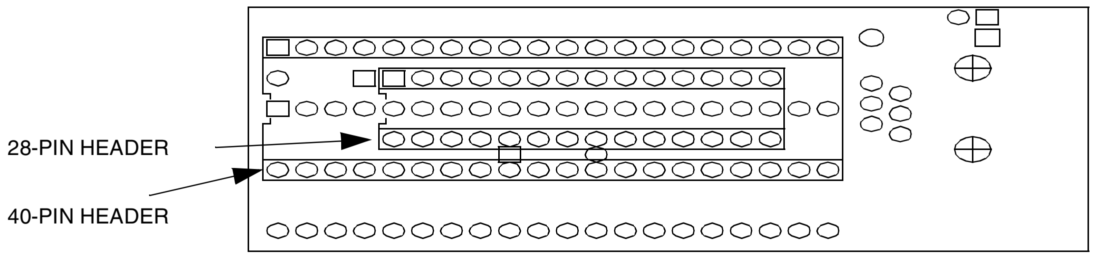


Figure A.5: MPLAB ICD Header Silkscreen

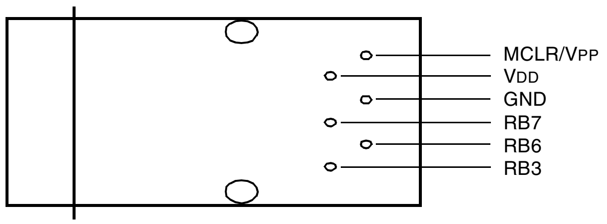


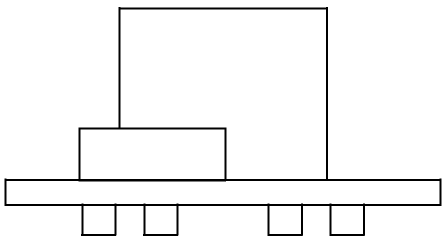
TOP SOLDER MASK



BOTTOM SOLDER MASK

CONNECTOR AS VIEWED FROM THE TOP OF PCB

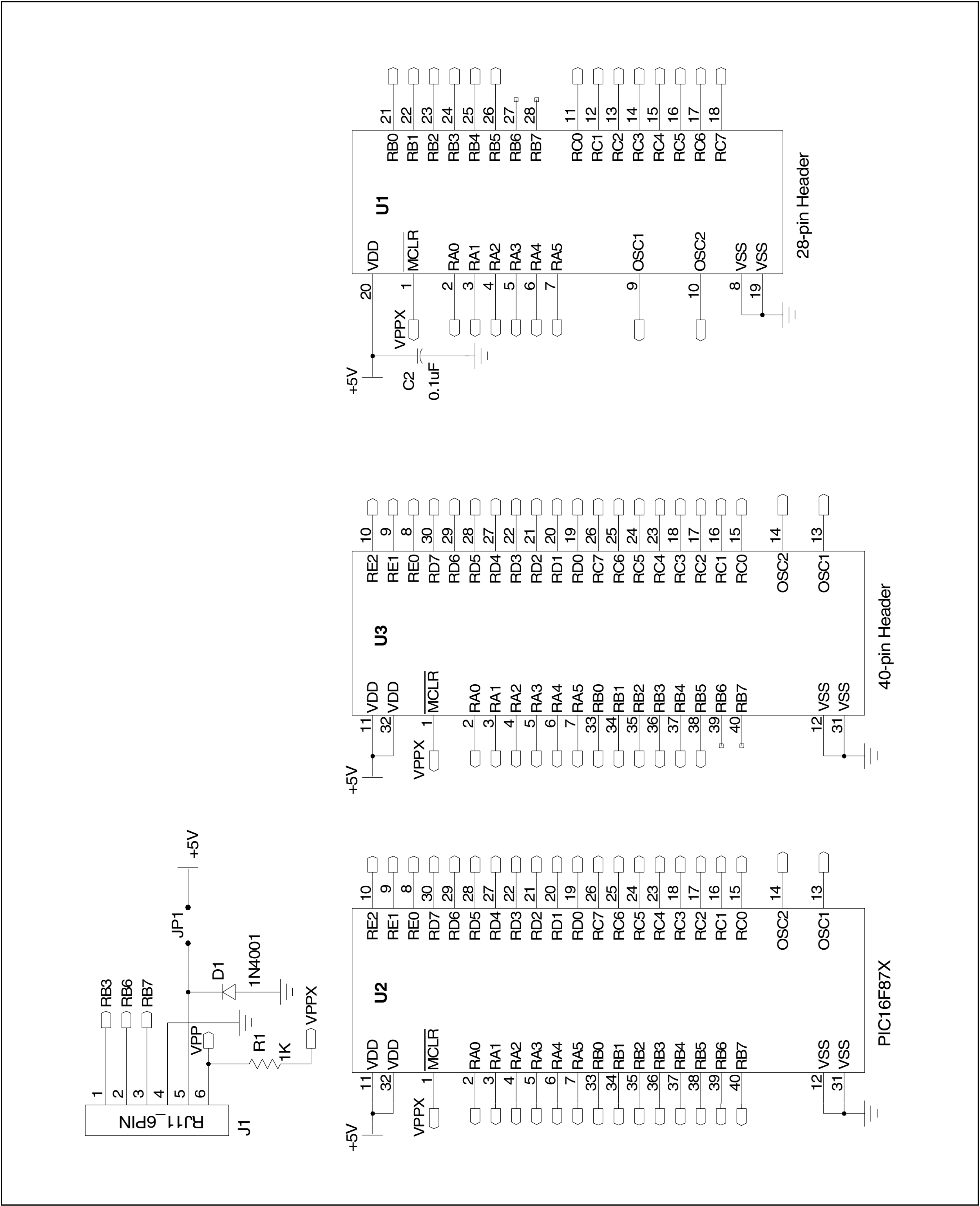




J1 - ICD Header
J2 - ICD Demo Board

This layout may change in future revisions of the product. However, newer layouts will be functionally identical.

Figure A.6: MPLAB ICD Header Schematic



A.3 MPLAB ICD Demo Board

The MPLAB ICD Demo Board is a simple demonstration board that supports PIC16F87X microcontrollers. The board can be used stand-alone with a pre-programmed part or with the MPLAB ICD Module and MPLAB ICD Header. A sample program is provided to demonstrate the unique features of the devices. Although a 40-pin and a 28-pin socket are provided to accommodate the different packages, only one processor can be run at a time.

A.3.1 Processor Sockets

Available sockets are:

- 40-pin socket for PIC16F871/874/877
- 28-pin socket for PIC16F870/872/873/876

Additional 20- and 14-pin male to male headers have been provided as stand-offs. Simply insert the desired headers onto the MPLAB ICD Demo Board and then plug the MPLAB ICD Header into the stand-offs. Without the stand-offs, the MPLAB ICD Header would not plug into the sockets. Or, simply insert the device directly onto the MPLAB ICD Demo Board.

A.3.2 Display

Eight red LEDs are connected to PORTC of each processor type. The PORTC pins are set high to light the LEDs.

A.3.3 DIP Switches

Eight DIP Switches are provided in a package as SW3. When all switches are in the ON position, each of eight red LEDs is connected to a pin of PORTC.

A.3.4 Power Supply

The MPLAB ICD Demo Board uses a 9.0V power supply to create a 5.0V V_{DD} voltage to power the MPLAB ICD Module. This 9V power supply is **NOT** provided with the MPLAB ICD system. If you own a PICSTART Plus Programmer, you can use its power supply to power the MPLAB ICD Demo Board. The part number is AC162039. The minimum power supply specifications for the MPLAB ICD Demo Board as shipped are:

- DC power supply: 9VDC @ 0.3A
- With barrel connector: ID = 2.5 mm, OD = 5.5 mm, Barrel length: 10.0 mm, Inside positive

A.3.5 ICD Connection

A modular cable connection next to the power supply can be used to connect the MPLAB ICD Demo Board directly to the MPLAB ICD Module.

A.3.6 RS-232 Serial Port

A RS-232 level shifting circuit has been provided to support connection to a RS-232 host through the DB9 connector. The port is configured as DCE, and can be connected to a PC using a straight through cable, as opposed to a null modem cable. The circuitry must be populated by the user:

- U4 – Analog Devices MAX233A or equivalent
- R14-R19 – 330Ω, 1/8W resistor
- C7-C11 – 0.1 µF capacitors
- J4 – DB9 female right angle connector

A.3.7 Push-Button Switches

Two push buttons provide the following functions:

- $\overline{\text{MCLR}}$ to reset the processor
- Active low switch connected to RB0

A.3.8 Oscillator Options

You can use the on-board RC oscillator circuit or plug an oscillator in the 4-pin socket. Make sure to set the jumper (JP1) to the proper selection.

- Socket provided for clock oscillator – use an oscillator from 32 kHz to 20 Mhz
- RC circuit – the frequency generated by the R3 resistor and C4 capacitor ranges from about 3.5 Mhz to 6 Mhz, depending on the operating voltage and ambient temperature.

A.3.9 Analog Input

A 1Kohm potentiometer is connected through a series 470 ohm resistor (to protect the part should the pin be configured as an output) to RA0/AN0. The port can be adjusted from V_{DD} to GND to provide an analog input to the PIC16F87X parts.

A.3.10 Silkscreens and Schematics

This section contains the silkscreen and schematic diagrams for the MPLAB ICD Demo Board.

Figure A.7: MPLAB ICD Demo Board Silkscreen

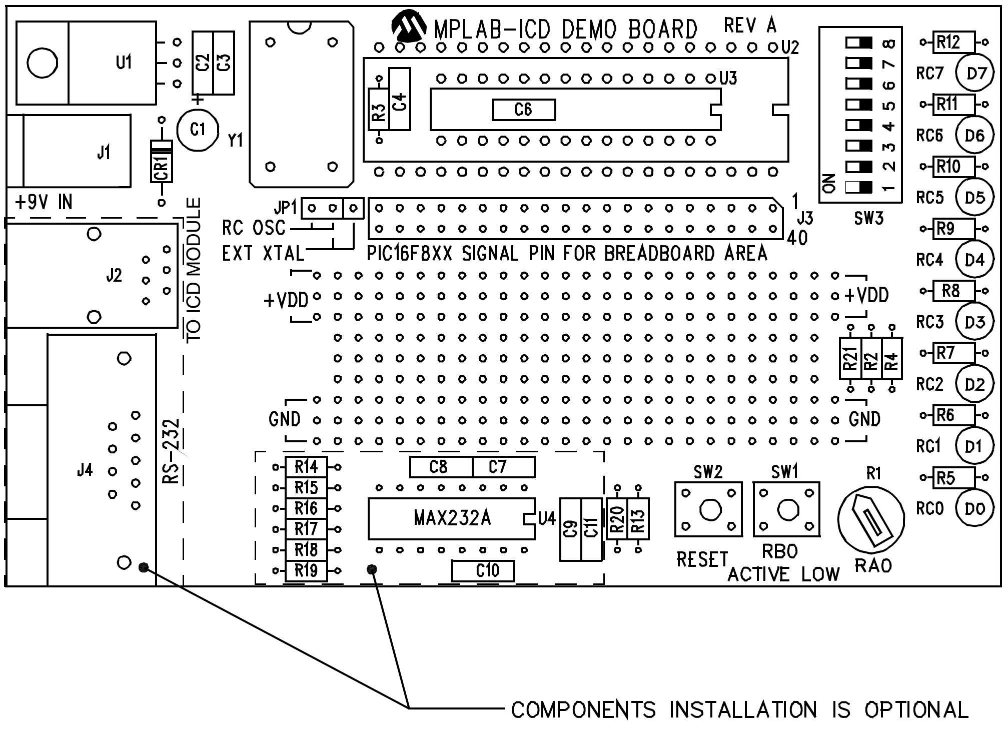


Figure A.8: MPLAB ICD Demo Board Schematic, Part 1

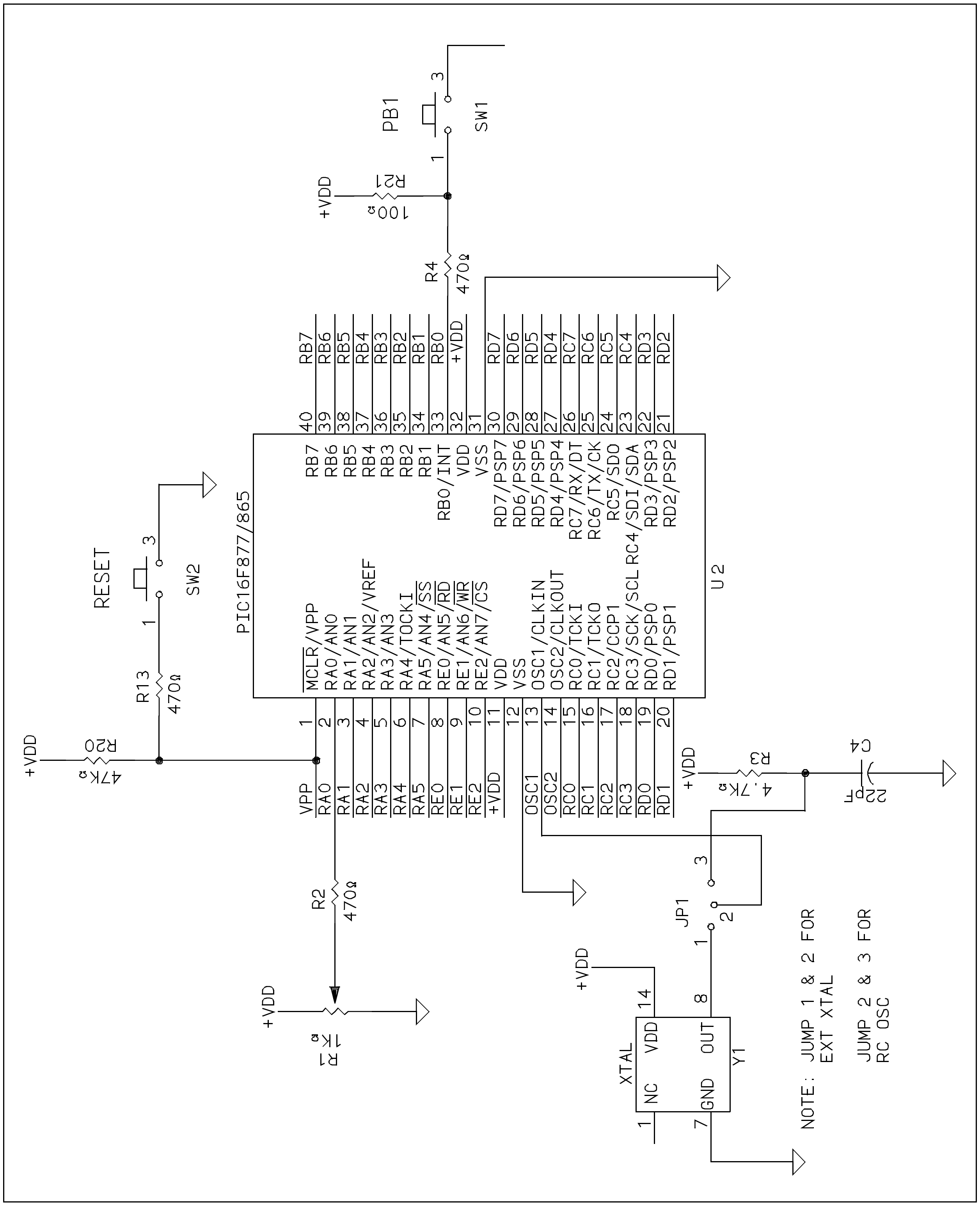


Figure A.9: MPLAB ICD Demo Board Schematic, Part 2

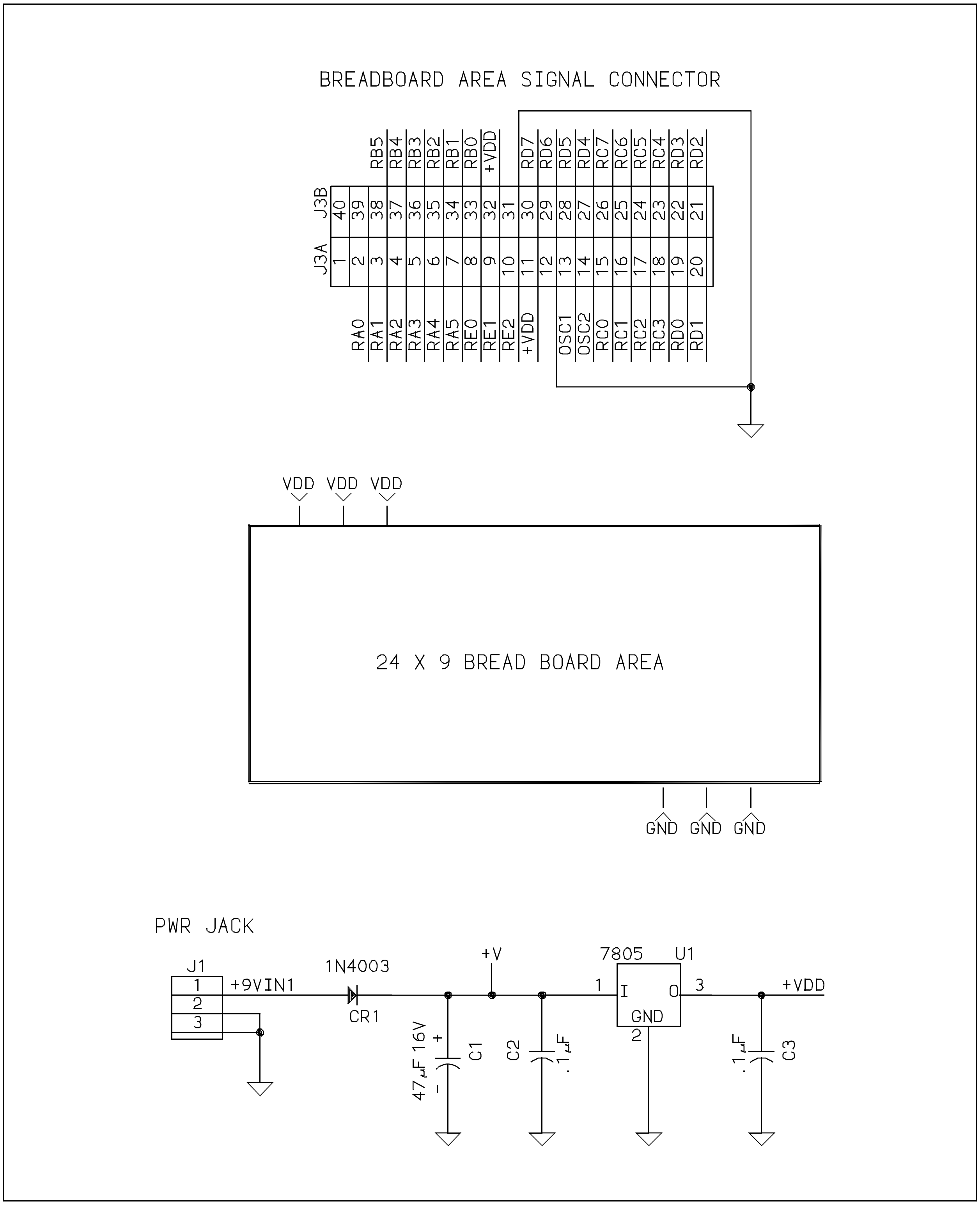


Figure A.10: MPLAB ICD Demo Board Schematic, Part 3

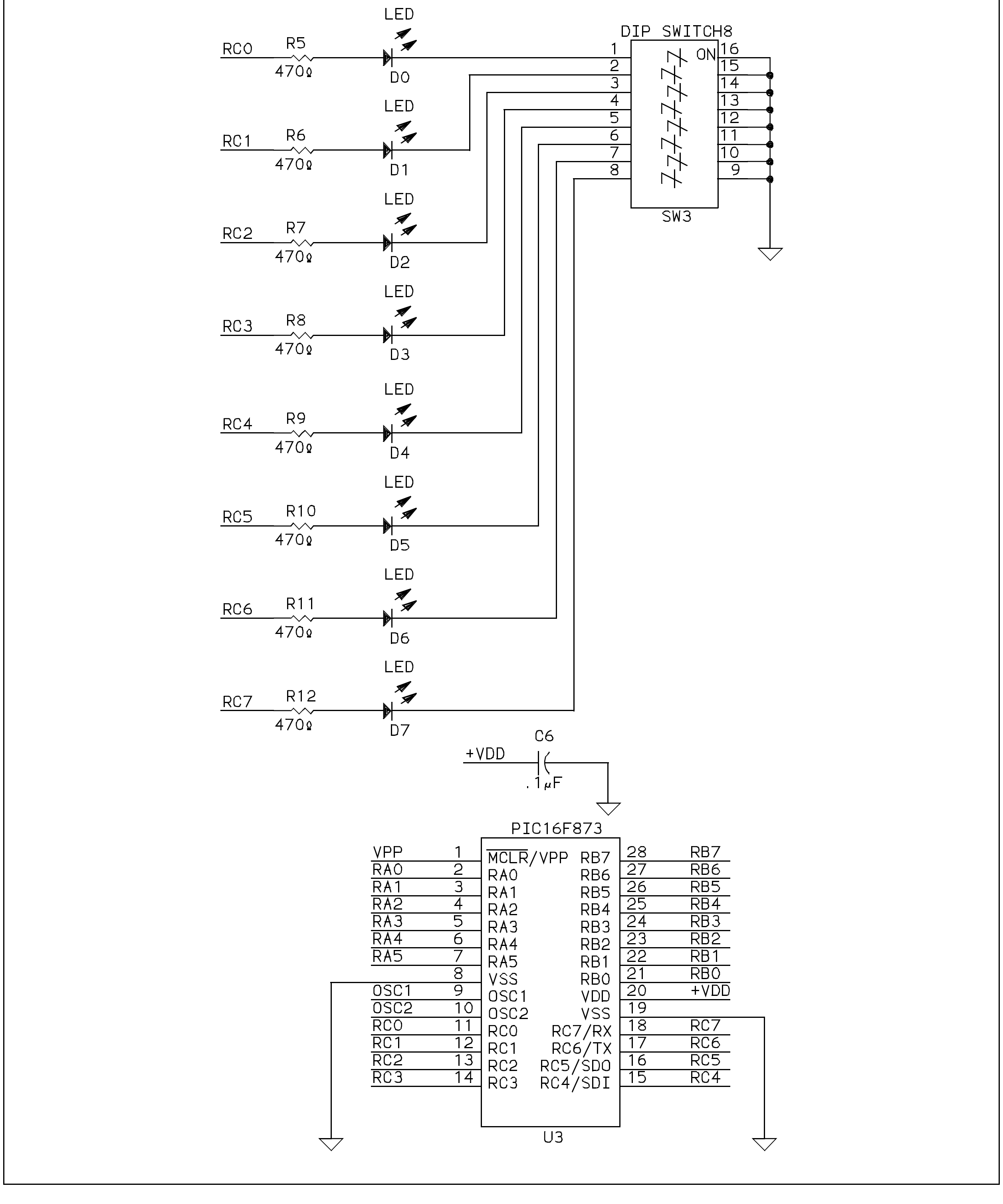
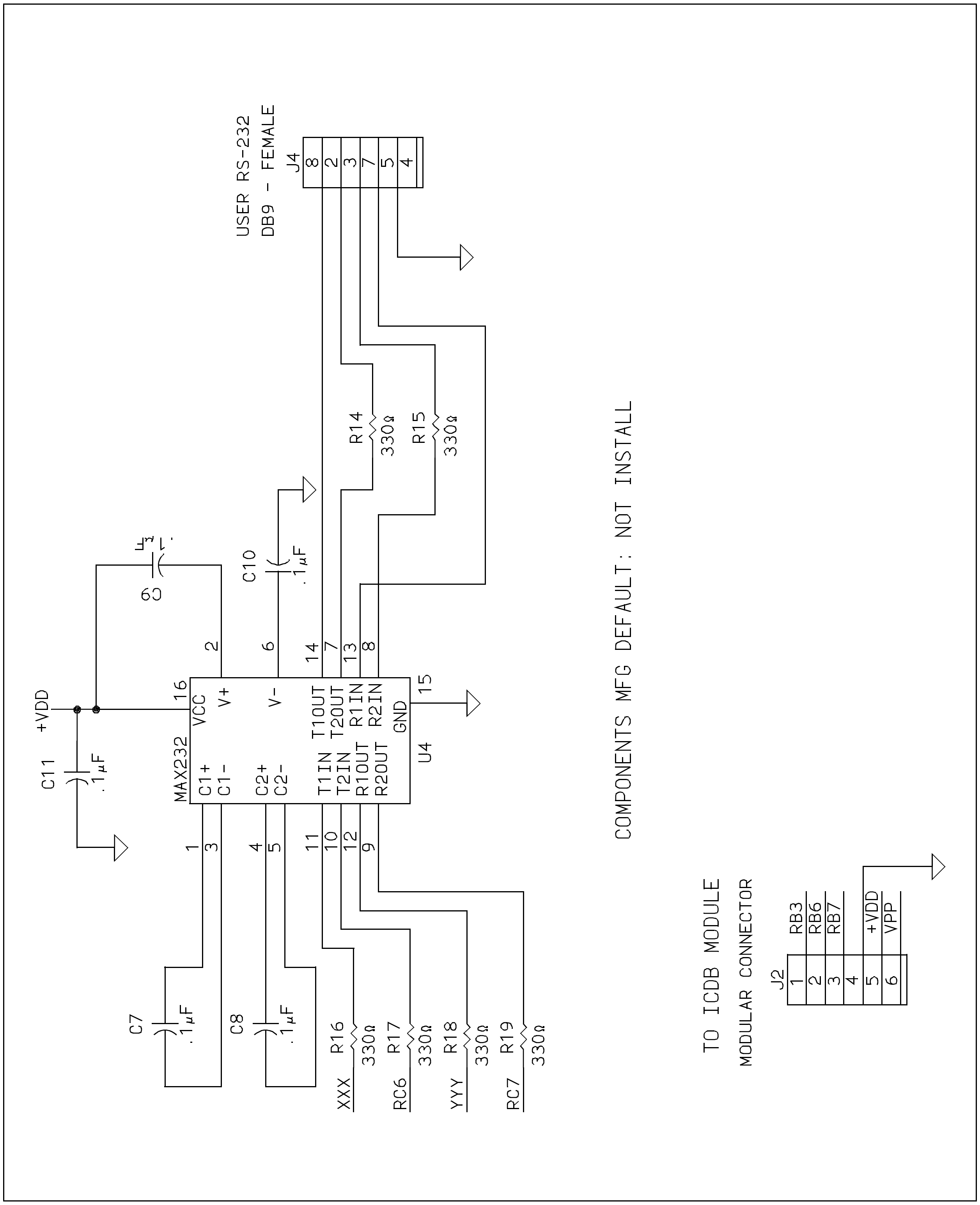


Figure A.11: MPLAB ICD Demo Board Schematic, Part 4



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 Assembler, MPLINK Object Linker, MPLIB Object Librarian
- 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 MCU.

Assemble

What an assembler does. See assembler.

Assembler

A language tool that translates a user's assembly source code (`.asm`) into machine code. The MPASM assembler 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.

Breakpoint – Hardware

An event whose execution will cause a halt.

Breakpoint – Software

An address where execution of the firmware will halt. Usually achieved by a special break 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 MCU on-board RC oscillator.

COFF

Common Object File Format. An intermediate file format generated by the MPLINK object linker that contains machine code and debugging information.

Command Line Interface

Command Line Interface refers to executing a program on the DOS command line with options. Executing the MPASM assembler with any command line options or just the file name will invoke the assembler. In the absence of any command line options, a prompted input interface (shell) will be executed.

Compile

What a compiler does. See compiler.

Compiler

A language tool that translates a user's C source code into machine code. MPLAB C17 and MPLAB C18 are Microchip's C compilers for PIC17CXXX and PIC18CXXX devices, respectively.

Configuration Bits

Unique bits programmed to set PICmicro MCU modes of operation. A configuration bit may or may not be preprogrammed. These bits are set in the *Options > Development Mode* dialog for simulators or emulators and in the _ _ `CONFIG` MPASM directive for programmers.

Control Directives

Control directives in the MPASM assembler permit sections of conditionally assembled code.

Data Directives

Data directives are those that control the MPASM assembler's allocation of memory and provide a way to refer to data items symbolically; that is, by meaningful names.

Data Memory

General purpose file registers (GPRs) from RAM on the PICmicro MCU device being emulated. The File Register window displays data memory.

Directives

Directives provide control of the assembler's operation by telling the MPASM assembler how to treat mnemonics, define data, and format the listing file. Directives make coding easier and provide custom output according to specific needs.

Download

Download is the process of sending data from the PC host to another device, such as an emulator, programmer or target board.

EEPROM

Electrically Erasable Programmable Read Only Memory. A special type of PROM that can be erased electrically. Data is written or erased one byte at a time. EEPROM retains its contents even when power is turned off.

Emulation

The process of executing software loaded into emulation memory as if the firmware resided on the microcontroller device under development.

Emulation Memory

Program memory contained within the emulator.

Emulator

Hardware that performs emulation.

Emulator System

The MPLAB ICE emulator system includes the pod, processor module, device adapter, cables, and MPLAB IDE software. The PICMASTER emulator system includes the pod, device-specific probe, cables, and MPLAB IDE software.

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 breakpoints.

Executable Code

See Hex Code.

Export

Sends data out of the MPLAB IDE in a standardized format.

Expressions

Expressions are used in the operand field of the MPASM assembler's source line and may contain constants, symbols or any combination of constants and symbols separated by arithmetic operators. Each constant or symbol may be preceded by a plus or minus to indicate a positive or negative expression.

Note: The MPASM assembler 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 2-cycle instruction. Since the PICmicro MCU architecture is pipelined, it prefetches the next instruction in the physical address space while it is executing the current instruction. However, if the current instruction changes the program counter, this prefetched instruction is explicitly ignored, causing a forced NOP cycle.

GPR

See Data Memory.

Halt

A function that stops the emulator. Executing Halt is the same as stopping at a breakpoint. 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 the MPLAB IDE.

ICE

In-Circuit Emulator. MPLAB ICE is Microchip's in-circuit emulator that works with the MPLAB IDE.

IDE

Integrated Development Environment. An application that has multiple functions for firmware development. The MPLAB IDE integrates a compiler, an assembler, a project manager, an editor, a debugger, a simulator, and an assortment of other tools within one Windows application. A user developing an application can write code, compile, debug, and test an application without leaving the MPLAB IDE desktop.

Identifier

A function or variable name.

Import

Bring data into the MPLAB 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. The MPLIB object librarian 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 MPLINK object linker.

Linker Script Files

Linker script files are the command files of the MPLINK object linker (`.lkr`). They define linker options and describe available memory on the target platform.

Listing Directives

Listing directives are those directives that control the MPASM assembler listing file format. They allow the specification of titles, pagination, and other listing control.

Listing File

A listing file is an ASCII text file that shows the machine code generated for each C source statement, assembly instruction, MPASM 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 connect 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, recompiling 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. The MPASM assembler is a DOS or Windows-based PC application that provides a platform for developing assembly language code for Microchip's PICmicro MCU families. Generically, the MPASM assembler will refer to the entire development platform including the macro assembler and utility functions.

The MPASM assembler will translate source code into either object or executable code. The object code created by the MPASM assembler may be turned into executable code through the use of the MPLINK object linker.

MPLAB CXX

Refers to the MPLAB C17 and MPLAB C18 C compilers.

MPLAB ICD

Microchip's in-circuit debugger for PIC16F87X devices. MPLAB ICD works with the MPLAB IDE. The MPLAB ICD system consists of a MPLAB ICD Module, MPLAB ICD Header (optional), MPLAB ICD Demo Board (optional), cables, and MPLAB IDE software.

MPLAB ICE

Microchip's in-circuit emulator that works with the MPLAB IDE.

MPLAB IDE

The name of the main executable program that supports the IDE with an Editor, Project Manager, and Emulator/Simulator Debugger. The MPLAB IDE software resides on the PC host. The executable file name is `mplab.exe`. `mplab.exe` calls many other files.

MPLAB SIM

Microchip's simulator that works with the MPLAB IDE.

MPLIB Object Librarian

The MPLIB object librarian is used with COFF object modules (`filename.o`) created using either MPASM v2.0, MPASMWIN v2.0 or MPLAB C v2.0 or later.

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

MPLINK Object Linker

The MPLINK object linker is a linker for the Microchip relocatable assembler, MPASM assembler, and the Microchip C compilers, MPLAB C17 or MPLAB C18. The MPLINK object linker also may be used with the Microchip librarian, MPLIB. It is designed to be used with the 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 breakpoint or executing single step instructions or the MPLAB IDE being run in simulator mode.

Node

MPLAB IDE project component.

NOP

No Operation. An instruction that has no effect when executed except to advance the program counter.

Object Code

The intermediate code that is produced from the source code after it is processed by an assembler or compiler. Relocatable code is code produced by the MPASM assembler or MPLAB CXX that can be run through the MPLINK object linker 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.

Opcode

Operational Code. 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, breakpoint 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 MCUs. Must be used with the MPLAB IDE software.

Pod

The external emulator box that contains emulation memory, trace memory, event and cycle timers, and trace/breakpoint 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 the MPASM assembler, operators of the same precedence are evaluated from left to right. Use parentheses to alter the order of evaluation.

Program Counter

A register that specifies the current execution address.

Program Memory

The memory area in a PICmicro MCU where instructions are stored. Memory in the emulator or simulator containing the downloaded target application firmware.

Programmer

A device used to program electrically programmable semiconductor devices such as microcontrollers.

Project

A set of source files and instructions to build the object and executable code for an application.

PRO MATE Programmer

A device programmer from Microchip. Programs all PICmicro MCUs and most memory and KEELOQ® devices. Can be used with the MPLAB IDE or stand-alone.

Prototype System

A term referring to a user's target application or target board.

PWM Signals

Pulse Width Modulation Signals. Certain PICmicro MCU devices have a PWM peripheral.

Qualifier

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

Radix

The number base, hex or decimal, used in specifying an address and for entering data in the *Window > Modify* command.

RAM

Random Access Memory (Data Memory).

Raw Data

The binary representation of code or data associated with a section.

Real-Time

When released from the halt state in the emulator or MPLAB ICD mode, the processor runs in real-time mode and behaves exactly as the normal chip would behave. In real-time mode, the real-time trace buffer of the MPLAB ICE is enabled and constantly captures all selected cycles, and all break logic is enabled. In the emulator or MPLAB ICD, the processor executes in real-time until a valid breakpoint causes a halt or until the user halts the emulator.

In the simulator real-time simply means execution of the microcontroller instructions as fast as they can be simulated by the host CPU.

Recursion

The concept that a function or macro, having been defined, can call itself. Great care should be taken when writing recursive macros; it is easy to get caught in an infinite loop where there will be no exit from the recursion.

Relocatable Section

A section whose address is not fixed (absolute). The linker assigns addresses to relocatable sections through a process called relocation.

Relocation

A process performed by the linker in which absolute addresses are assigned to relocatable sections and all identifier symbol definitions within the relocatable sections are updated to their new addresses.

ROM

Read Only Memory (Program Memory).

Run

The command that releases the emulator from halt, allowing it to run the application code and change or respond to I/O in real time.

Section

An portion of code or data which has a name, size, and address.

SFR

Special Function Registers of a PICmicro 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 MCU.

Single Step

This command steps though code, one instruction at a time. After each instruction, the 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, the 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 breakpoint is used to halt the processor, one or more additional instructions may be executed before the processor halts. The number of extra instructions executed after the intended breakpoint is referred to as the skid.

Source Code – Assembly

Source code consists of PICmicro MCU 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 MCU 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 MCU 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 breakpoint will be set at the instruction after the CALL. If for some reason the subroutine gets into an endless loop or does not return properly, the next breakpoint will never be reached.

The Step Over command is the same as Single Step except for its handling of CALL instructions.

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 the 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.

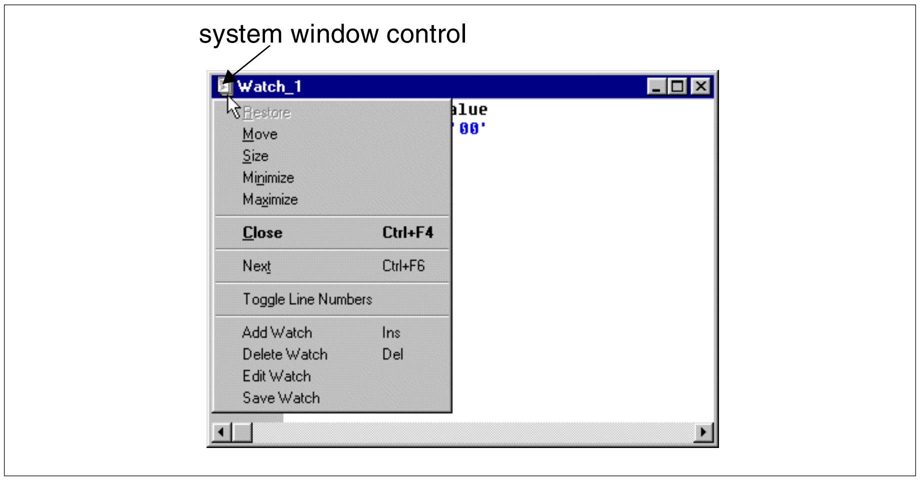


Figure G1: System Window Control Menu – Watch Window

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 the 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 the MPLAB IDE's trace window.

Trace Memory

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

Trigger Output

Trigger output refers to an emulator output signal that can be generated at any address or address range, and is independent of the trace and breakpoint settings. Any number of trigger output points can be set.

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 MCU 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 breakpoint.

NOTES:

Index

A

A/D Converter 34
Absolute Section 75
Access RAM 75
Address Ranges 49
Analog Input 69
Animate 54
Assembler 75
Assigned Section 76

B

Basic Functions 53
Baud Rate 45
Blank Check a Device 50
Breakpoints 34, 36, 55
    Break on Address Match 55
    Break on User Halt 55
    Hardware 76
    Software 76
Brown Out Detection 48

C

Calibration Memory 76
CD-ROM 4
Checksum 48
Code Protection 48
COM Port 45
Command Line Interface 76
Common Problems 57
Communications 57, 59
Compiler 76
Components, System 11
Configuration Bits 31, 48, 49, 77
Connection 68
Creating a Hex File 21
Customer Notification Service 6
Customer Support 8

D

Data Memory 77
Debug Code 10, 53
Debug Mode 50, 59
Debugging 34
Debugging Procedure 56
Demo Board 11, 13, 16, 33, 68
Development Mode 52
Device Selection 31, 48
DIP Switches 68
Directives 77
    Control 77
    Data 77
    Listing 81
    Macro 81
    Object File 84
Displaying Voltages 49
Documentation
    Conventions 2
    Layout 1
    Updates 3

E

EEPROM 77
EEPROM Data 49
Emulator 77
Enabling Debug Mode 50
Erase 50
Erasing Before Programming 49
Executable Code 78
Export 78
Expressions 78
Extended Microcontroller Mode 78
External
    Inputs 78
    Linkage 78
    Memory 78
    Symbol 78

F

Features 9
File, Listing 81
Flash Memory Write Options 48
Frequency 46
Function Buttons 32, 50

G

Getting Started 43
Glossary 75

H

Hardware 61
    Installation 15
Header 11, 13, 15, 61

Hex Code 79
Hex File
    Creating 21
    Loading 56

I

ICD 9, 79
ICD Connection 68
ICD Options Dialog 30, 47
ICD Status 45
ICE 80
ID Codes 48
ID Locations 49
ID's and Checksum 31
IDE 80
Import 80
Initialized Data 80
Installation 15
Internal
    Linkage 80
Internet Address 5

L

LED 68
Librarian 80
Library 80
Linker 80
Linker Script Files 80
Listing File 81
Loading the Hex File 56
Local Label 81
Logic Probes 81
Low-Voltage Program 48
Low-Voltage Programming 10

M

MCU 81
Memory
    Calibration 76
    Data 77
    Program 85
    Trace 90
Memory Models 81
Microchip Internet Web Site 5
Microcontroller Mode 82
Microprocessor Mode 82
Mnemonics 82
Modifying the Program 55
Module 11, 13, 16, 17, 61
MPASM Assembler 75, 82
MPLAB CXX 82
MPLAB ICD 9, 82
    Resources Used 10
MPLAB ICD Operating System
    Downloading 50
MPLAB ICE 82
MPLAB IDE 83
MPLAB SIM 83
MPLAB, Running 20, 43
MPLAB-ICD Tutorial 19
MPLIB Object Librarian 80, 83
MPLINK Object Linker 80, 83

O

Object Code 84
Off-Chip Memory 84
Opcode 84
Operating Frequency 46
Operators 84
Oscillator Options 48, 69

P

Pass Counter 84
PICMASTER Emulator 85
PICmicro 87, 88
PICSTART Plus 85
Power Supply 68
Power Up Timer 48
Precedence 85
PRO MATE Programmer 85
Processor Sockets 68
Program 50
Program Counter 85
Program Execution 54
Program Memory 10, 85
Program Options 32, 49
Programmer 85
Programming 32
    Configuration Bits 49
    EEPROM Data 49
    Erase All Before Programming 49
    ID Locations 49
Project 85
Pushbutton Switches 69

Q

Qualifier 86

R

Radix 86
Read 50
Reading, Recommended 4
Real-Time 54, 86
Reconnecting 46
Relocatable Section 86
Reserved Resources 10
Resources Used by MPLAB ICD 10
RS-232 69
Run Mode 54

S

Schematics 61, 69
Section 87
    Absolute 75
    Assigned 76
    Relocatable 86
    Shared 87
    Unassigned 90
Selecting Oscillator Options 48
Selecting the Device 48
Selecting Watchdog Timer Options 48
Self Test 50
Serial Port 69
Shared Section 87
Silkscreens 61, 69
Simulator 87
Single Step 54, 59, 60, 87
Skew 87
Skid 87
Source Code, Assembly 87
Source Code, C 88
Specifying an Address Range 49
Stack 10
Stack, Hardware 88
Stack, Software 88
Start and End Addresses 49
Status 45
Stimulus 89
Stopwatch 89
Switches, DIP 68
Switches, Pushbutton 69
Symbol 89
System Button, Windows 89
System Window Control 89

T

Target 90
Trace Memory 90
TRGIN 78
Trigger
    In 78
Troubleshooting 57
Tut877 19
    Main Routine 39
    Source Code 40
Tutorial 19

U

Unassigned Section 90
Uninitialized Data 90
Upload Options 45
Use Checksum as ID 48
Using Breakpoints 55

V

Verify 50
Voltages 31, 49

W

Warranty Registration 3
Watch Window 36, 38, 91
Watchdog Timer 48
Watchdog Timer (WDT) 91
Write to EECON 48
WWW Address 5

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

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3838 Fax: 978-692-3821

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

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

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

China - Beijing
Microchip Technology Beijing 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 - Shanghai
Microchip Technology Shanghai Office
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

Hong Kong
Microchip Asia Pacific
RM 2101, 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 Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

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

ASIA/PACIFIC (continued)

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

Taiwan
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

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

10/01/00

DNV Certification, Inc.
USA

ANSI★RAB QMS ACCREDITED

DNV MSC
The Netherlands
Accredited by the RvA

DNV

ISO 9001 / QS-9000
REGISTERED FIRM

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