MPLAB® ICD 2 In-Circuit Debugger User's Guide

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.

Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE**.** Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

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

AmpLab, FilterLab, Linear Active Thermistor, Migratable Memory, MXDEV, MXLAB, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

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

Printed on recycled paper.

MPLAB® ICD 2 USER'S GUIDE

Table of Contents

NOTES:

MPLAB® ICD 2 USER'S GUIDE

Preface

NOTICE TO CUSTOMERS

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

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

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

MPLAB® ICD 2 User's Guide

INTRODUCTION

The support information discussed here can help you when using the MPLAB ICD 2 in-circuit debugger.

Topics covered in this chapter:

- Document Layout
- Conventions Used in this Guide
- Warranty Registration
- Reference Documents
- The Microchip Web Site
- Development Systems Customer Change Notification Service
- Customer Support

DOCUMENT LAYOUT

This document describes how to use the MPLAB ICD 2 as a development tool to emulate and debug firmware on a target board. The manual layout is as follows:

- **Part 1 – Operation**
  - **Chapter 1. MPLAB ICD 2 Overview** – explains the MPLAB ICD 2, how it helps you and how it works.
  - **Chapter 2. Getting Started** – describes the system components, provides installation and configuration instructions, explains how to set up the environment and discusses the development process.
  - **Chapter 3. MPLAB ICD 2 Tutorial** – provides a tutorial using the Project Wizard.
  - **Chapter 4. Additional Topics** – discusses upgrading firmware, using hex files and linker scripts, and provides information on ROMless devices and ICD/ICE devices.
  - **Chapter 5. ICD Function Summary** – describes all the ICD menus, toolbars and dialogs.
- **Part 2 – Troubleshooting**
  - **Chapter 6. Troubleshooting Tips** – describes the chain of functionality.
  - **Chapter 7. Self-Test** – discusses the various self-test features.
  - **Chapter 8. General Troubleshooting** – provides answers to frequently asked questions, discusses common problems, and provides error and warning messages.
- **Appendix A. Hardware Specifications** – contains information on the MPLAB ICD 2 module, the cable and connector, and power supply.

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	*MPLAB® IDE User's Guide*
	Emphasized text	...is the *only* compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	*File>Save*
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
N'Rnnnn	A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>

DOCUMENTATION CONVENTIONS (CONTINUED)

Description	Represents	Examples
Courier New font:		
Plain Courier New	Sample source code	`#define START`
	Filenames	`autoexec.bat`
	File paths	`c:\mcc18\h`
	Keywords	`_asm, _endasm, static`
	Command-line options	`-Opa+, -Opa-`
	Bit values	`0, 1`
	Constants	`0xFF, 'A'`
Italic Courier New	A variable argument	*`file`*`.o`, where *`file`* can be any valid file-name
Square brackets [ ]	Optional arguments	`mcc18 [options]` *`file`* `[options]`
Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments; an OR selection	`errorlevel {0|1}`
Ellipses...	Replaces repeated text	var_name [, var_name...]
	Represents code supplied by user	void main (void) { ... }

WARRANTY REGISTRATION

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

REFERENCE DOCUMENTS

This document describes how to use MPLAB ICD 2. Other useful documents are listed below.

MPLAB® ICD 2 Design Advisory (DS51566)

Please read this first! Contains important information about using the MPLAB ICD 2 with your target design.

Readme for MPLAB ICD 2

For the latest information on using MPLAB ICD 2, read the Readme for MPLAB ICD 2.txt file (an ASCII text file) in the MPLAB IDE directory. The README file contains update information and known issues that may not be included in this on-line help file.

Using MPLAB® ICD 2 (DS51265)

A poster showing the various ways you can set up MPLAB ICD 2 hardware.

Header Board Specification (DS51292)

A small booklet describing how to install and use MPLAB ICD 2 headers. Headers are used to better debug selected devices using special -ICD/ICE device versions, without the loss of pins or resources.

Universal Programming Module Instruction Sheet (DS51280)

A sheet describing how to use the UPM.

THE MICROCHIP WEB SITE

Microchip provides online support via our web site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

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

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

The Development Systems product group categories are:

- **Compilers** – The latest information on Microchip C compilers and other language tools. These include the MPLAB C18 and MPLAB C30 C compilers; MPASM™ and MPLAB ASM30 assemblers; MPLINK™ and MPLAB LINK30 object linkers; and MPLIB™ and MPLAB LIB30 object librarians.
- **Emulators** – The latest information on Microchip in-circuit emulators.This includes the MPLAB ICE 2000 and MPLAB ICE 4000.
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debugger, MPLAB ICD 2.
- **MPLAB IDE** – The latest information on Microchip MPLAB IDE, the Windows® Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB SIM simulator, MPLAB IDE Project Manager and general editing and debugging features.
- **Programmers** – The latest information on Microchip programmers. These include the MPLAB PM3 and PRO MATE® II device programmers and the PICSTART® Plus and PICkit™ 1 development programmers.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

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

NOTES:

MPLAB® ICD 2 USER'S GUIDE

Part 1 – Operation

NOTES:

Chapter 1. MPLAB ICD 2 Overview

1.1 INTRODUCTION

An overview of the MPLAB ICD 2 development tool is given here. MPLAB ICD 2 is defined and its operation explained. Additionally, device resources used during debugging are listed.

Topics covered in this chapter are:

- What is MPLAB ICD 2?
- How MPLAB ICD 2 Helps You
- How MPLAB ICD 2 Works
- Resources Used By MPLAB ICD 2

1.2 WHAT IS MPLAB ICD 2?

The MPLAB ICD 2 is a low-cost in-circuit debugger (ICD) and in-circuit serial programmer.

MPLAB ICD 2 is intended to be used as an evaluation, debugging and programming aid in a laboratory environment.

The MPLAB ICD 2 offers these features:

- Real-time and single-step code execution
- Breakpoints, Register and Variable Watch/Modify
- In-circuit debugging
- Target V_{DD} monitor
- Diagnostic LEDs
- MPLAB IDE user interface
- RS-232 serial or USB interface to a host PC

1.3 HOW MPLAB ICD 2 HELPS YOU

The MPLAB ICD 2 allows you to:

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

1.4 HOW MPLAB ICD 2 WORKS

A simplified description of how MPLAB ICD 2 works is provided here. It is intended to provide enough information so a target board can be designed that is compatible with MPLAB ICD 2 for both programming and debugging operations. The basic theory of programming and in-circuit debugging is described so that problems, if encountered, are quickly resolved.

This section provides an overview of MPLAB ICD 2, followed by a description of MPLAB ICD 2 programming and debugging modes. The following topics are covered:

- ICD vs. ICE
- Modular Interface Connections
- Circuits That Will Prevent MPLAB ICD 2 From Functioning
- Debug Mode
- Requirements For Debug Mode
- Debug Reserved Resources
- Programmer Mode

1.4.1 ICD vs. ICE

The in-circuit debugger (ICD) is a cost-efficient alternative to an in-circuit emulator (ICE). It can do many things that were previously done only with more expensive hardware, but the cost benefits come with a trade-off of some of the conveniences of an in-circuit emulator. If users are willing to design their application to be ICD compatible, they can enjoy the benefits of a low-cost hardware debugger.

As opposed to an ICE, some of the requirements of the in-circuit debugger are:

- The in-circuit debugger requires exclusive use of some hardware and software resources of the target.
- The target PIC® MCU must have a functioning clock and be running.
- The ICD can debug only when all the links in the system are fully functional.

An emulator provides memory and a clock, and can run code – even without being connected to the target application board. During the development and debugging cycle, an ICE provides the most power to get the system fully functional, whereas an ICD may not be able to debug at all if the application does not run. On the other hand, an in-circuit debug connector can be placed on the application board and connected to an ICD even after the system has been produced, allowing easy testing, debugging and reprogramming of the application. Even though an ICD has some drawbacks in comparison to an ICE, in this situation it has some distinct advantages:

- A connection to the application after it has been produced does not require extraction of the microcontroller in order to insert an ICE probe.
- The ICD can reprogram the firmware in the target application without any other connections or equipment.

Note:	An ICE uses custom hardware to emulate the target microcontroller. An ICD uses hardware on the target microcontroller to do some of the functions of an ICE. An ICD also employs software running on the target to do ICE-like functions and, as a result, relies upon the target microcontroller for some memory space, CPU control, stack storage and I/O pins for communication.

1.4.2 Modular Interface Connections

MPLAB ICD 2 is connected to the target PIC MCU with the modular interface cable, which is a six conductor cable. The pin numbering for the MPLAB ICD 2 connector is shown from the bottom of the target PC board in Figure 1-1.

Note: The ICD cable has mirror imaged connections on each end, and connections on the MPLAB ICD 2 module are the opposite of connections shown here.

FIGURE 1-1: PIN NUMBERING FOR MODULAR CONNECTOR

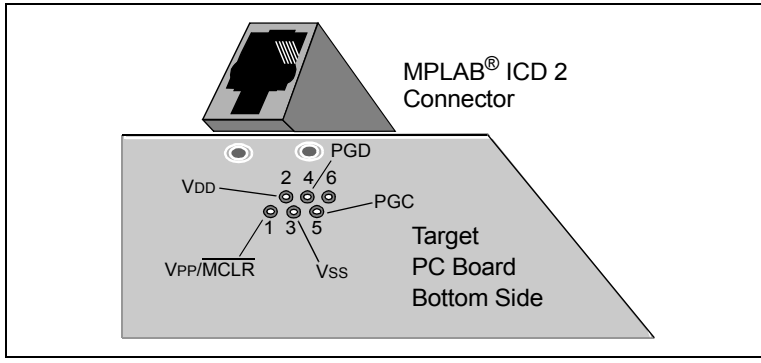
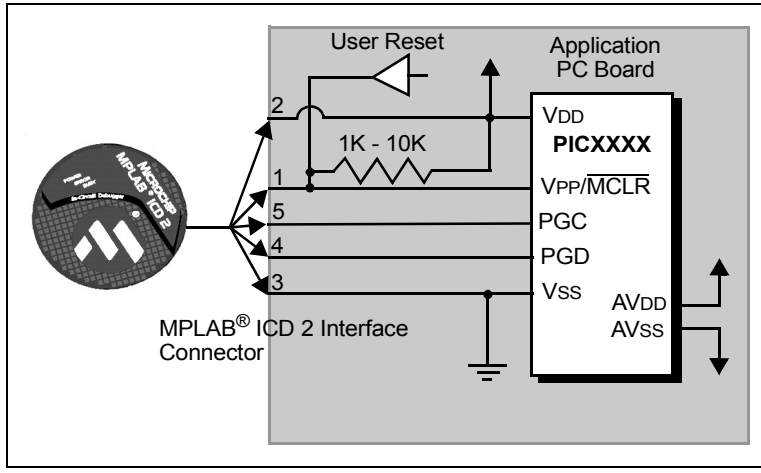


Figure 1-2 shows the interconnections of the MPLAB ICD 2 to the modular connector on the target board. There are six pins on the ICD connector, but only five are used. The diagram also shows the wiring from the connector to the PIC MCU device on the target PC board. A pull-up resistor (usually around 10k Ohm) is recommended to be connected from the VPP/MCLR line to VDD so that the line may be strobed low to reset the PIC MCU.

FIGURE 1-2: MPLAB® ICD 2 CONNECTIONS TO TARGET BOARD



Although pin 2 (VDD) can supply a limited amount of power to the target application under certain conditions, for the purposes of these descriptions, pins 2 and 3 (VSS) are omitted. They are shown on the diagram for completeness, but in the following descriptions only three lines are active and relevant to core MPLAB ICD 2 operation: VPP/MCLR, PGC and PGD.

Note: In the following discussions, VDD is ignored. But be aware that the target VDD is also used to power the output drivers in MPLAB ICD 2. This allows level translation for target low-voltage operation. If MPLAB ICD 2 does not have voltage on its VDD line (pin 2 of the ICD connector), either from power being supplied to the target by MPLAB ICD 2 or from a separate target power supply, it will not operate.

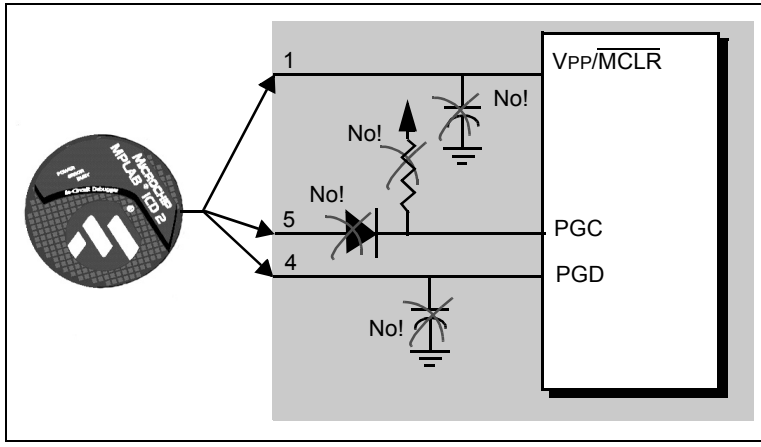
Not all PIC MCUs have the AVDD and AVSS lines, but if they are present on the target PIC MCU, all must be connected in order for MPLAB ICD 2 to operate.

The interconnection is very simple, any problems experienced are often caused by other connections or components on these critical lines that interfere with the operation of MPLAB ICD 2, as discussed in **Section 1.4.3 "Circuits That Will Prevent MPLAB ICD 2 From Functioning"**.

1.4.3 Circuits That Will Prevent MPLAB ICD 2 From Functioning

Figure 1-3 shows the active MPLAB ICD 2 lines with some components that will prevent MPLAB ICD 2 from functioning:

FIGURE 1-3: IMPROPER CIRCUIT COMPONENTS



Specifically, these guidelines must be followed:

- No pull-ups on PGC/PGD – they will divide the voltage levels, since these lines have 4.7KΩ pull-down resistors in MPLAB ICD 2.
- No capacitors on PGC/PGD – they will prevent fast transitions on data and clock lines during programming and debug communications.
- No capacitors on $\overline{\text{MCLR}}$ – they will prevent fast transitions of VPP. A simple pull-up resistor is generally sufficient.

- No diodes on PGC/PGD – they will prevent bidirectional communication between MPLAB ICD 2 and the target PIC MCU.

1.4.4 Debug Mode

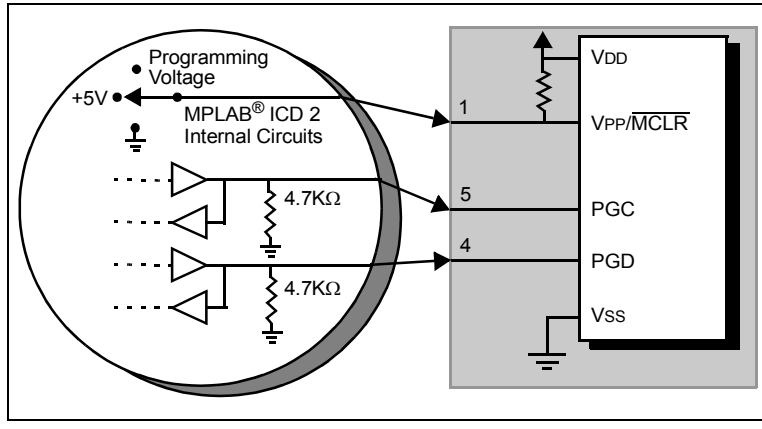
There are two steps to using MPLAB ICD 2 as a debugger. The first requires that an application be programmed into the target PIC MCU. The second uses the internal in-circuit debug hardware of the target Flash PIC MCU to run and test the application program. These two steps are directly related to the MPLAB IDE operations:

1. Programming the code into the target.
2. Using the debugger to set breakpoints and run.

If the target PIC MCU cannot be programmed correctly, MPLAB ICD 2 will not be able to debug.

Figure 1-4 shows the basic interconnections required for programming. Note that this is the same as Figure 1-2, but for the sake of clarity, the VDD and VSS lines from MPLAB ICD 2 are not shown.

FIGURE 1-4: PROPER CONNECTIONS FOR PROGRAMMING



A simplified diagram of some of the internal interface circuitry of the MPLAB ICD 2 is shown. For programming, no clock is needed on the target PIC MCU, but power must be supplied. When programming, MPLAB ICD 2 puts programming levels on VPP, sends clock pulses on PGC and serial data via PGD. To verify that the part has been programmed correctly, clocks are sent to PGC and data is read back from PGD. This conforms to the ICSP protocol of the PIC MCU under development.

1.4.5 Requirements For Debug Mode

To debug (set breakpoints, see registers, etc.) with the MPLAB ICD 2 there are critical elements that must be working correctly:

- MPLAB ICD 2 must be connected to a PC. It must be powered by an external power supply or the PC via the USB cable, and it must be communicating with MPLAB IDE software via the RS-232 or USB cable. See the on-line help for details.
- The MPLAB ICD 2 must be connected as shown to the VPP, PGC and PGD pins of the target PIC MCU with the modular interface cable (or equivalent). VSS and VDD are also required to be connected between the MPLAB ICD 2 and target PIC MCU.
- The target PIC MCU must have power and a functional, running oscillator. If the target PIC MCU does not run – for whatever reason – MPLAB ICD 2 cannot debug.
- The target PIC MCU must have its configuration words programmed correctly:
  - The oscillator Configuration bits should correspond to RC, XT, etc., depending upon the target design.
  - The target PIC MCU must not have the Watchdog Timer enabled.
  - The target must not have code protection enabled.
  - The target must not have table read protection enabled.

Once the above conditions are met, you may proceed to the following:

- Sequence of Operations Leading to Debug Mode
- Debug Mode Details

1.4.5.1 SEQUENCE OF OPERATIONS LEADING TO DEBUG MODE

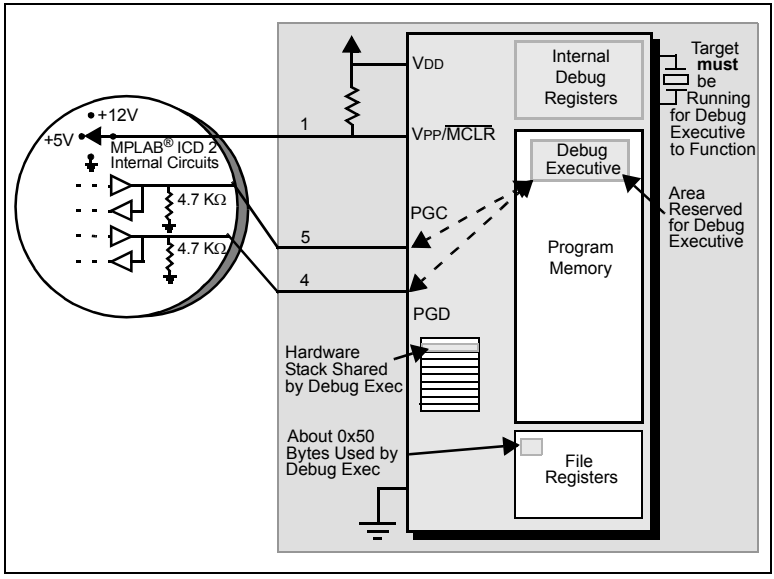
Given that the Requirements For Debug Mode are met, these actions can be performed when MPLAB ICD 2 is set as the current debugger (*Debugger>Select Tool*):

- When *Debugger>Program* is selected, the application code is programmed into the PIC MCU's memory via the ICSP protocol as described above.
- A small "debug executive" program is loaded into the high area of program memory of the target PIC MCU. Since the debug executive must reside in program memory, the application program must not use this reserved space. The debug executive typically needs about 0x120 words of program memory.
- Special "in-circuit debug" registers in the target PIC MCU are enabled. These allow the debug executive to be activated by the MPLAB ICD 2.
- The target PIC MCU is held in reset by keeping the V_{PP}/MCLR line low.

1.4.5.2 DEBUG MODE DETAILS

Figure 1-5 illustrates the MPLAB ICD 2 ready for debugging.

FIGURE 1-5: MPLAB® ICD 2 READY FOR DEBUGGING



Typically, in order to find out if an application program will run correctly, a breakpoint is set early in the program code. When a breakpoint is set from the user interface of the MPLAB IDE, the address of the breakpoint is stored into the special internal debug registers of the target PIC MCU. Commands on PGC and PGD communicate directly to these registers to set the breakpoint address.

Next, the *Debugger>Run* function or the Run icon (forward arrow) is usually pressed from MPLAB IDE. MPLAB ICD 2 will raise the VPP/MCLR line to allow the target to run, the target will start from address zero and execute until the program counter reaches the breakpoint address previously stored in the internal debug registers.

After the instruction at the breakpoint address is executed, the in-circuit debug mechanism of the target PIC MCU “fires” and transfers the PIC MCU’s program counter to the debug executive (much like an interrupt) and the user’s application is effectively halted. MPLAB ICD 2 communicates with the debug executive via PGC and PGD, gets the breakpoint status information and sends it back to the MPLAB IDE. The MPLAB IDE then sends a series of queries to MPLAB ICD 2 to get information about the target PIC MCU, such as file register contents and the state of the CPU. These queries are ultimately performed by the debug executive.

The debug executive runs just like an application in program memory. It uses some locations on the hardware stack (usually just one or two) and, typically, about fourteen file registers for its temporary variables. If the PIC MCU does not run, for whatever reason, such as no oscillator, a faulty power supply connection, shorts on the target board, etc., then the debug executive cannot communicate to MPLAB ICD 2 and MPLAB IDE will issue an error message.

Another way to get a breakpoint is to press the MPLAB IDE’s **Halt** button (the “pause” symbol to the right of the Run arrow). This toggles the PGC and PGD lines in such a way that the in-circuit debug mechanism of the target PIC MCU switches the program counter from the user’s code in program memory to the debug executive. Again, the target application program is effectively halted, and MPLAB IDE uses MPLAB ICD 2 communications with the debug executive to interrogate the state of the target PIC MCU.

1.4.6 Debug Reserved Resources

When developing with MPLAB ICD 2, you must be aware of the device resources reserved for debugging. See **Section 1.5 “Resources Used By MPLAB ICD 2”** for a list of reserved resources by device type.

1.4.7 Programmer Mode

When using the *Programmer>Program* selection to program a device, the in-circuit debug registers should be disabled in the MPLAB IDE so the MPLAB ICD 2 will program only the target application code and the Configuration bits (and EEPROM data, if available and selected) into the target PIC MCU. The debug executive will not be loaded. In this mode the MPLAB ICD 2 can only toggle the MCLR line to reset and start the target. A breakpoint cannot be set, and register contents cannot be seen or altered.

The MPLAB ICD 2 programs the target using ICSP. No clock is required while programming, and all modes of the processor can be programmed, including code-protect, Watchdog Timer enabled and table read protect.

Note: A header board is required to debug smaller pin count parts with the MPLAB ICD 2. These parts can be programmed without the header by connecting the VPP, PGC and PGD lines as described previously.

1.5 RESOURCES USED BY MPLAB ICD 2

Refer the MPLAB ICD 2 on-line help for up-to-date information on reserved resources.

Chapter 2. Getting Started

2.1 INTRODUCTION

The use of MPLAB ICD 2 with MPLAB IDE software and an explanation of power options and recommended power-up sequences are discussed here. Following this is a step-by-step tutorial using MPLAB ICD 2. Before continuing on to the tutorial, be sure that the software and hardware is configured according to the instructions given here.

In addition, please refer to the *"MPLAB ICD 2 Design Advisory"* (DS51566) for hardware configuration issues.

Topics covered in this chapter:

- MPLAB ICD 2 System Components
- Installing and Configuring MPLAB IDE for MPLAB ICD 2
- Applying Power
- Connection Options
- Setting Up the Environment
- MPLAB ICD 2 Development Process

2.2 MPLAB ICD 2 SYSTEM COMPONENTS

In addition to the MPLAB ICD 2 module, the following components are required:

- MPLAB IDE software (version 6.20 or later) – Installed on the PC to control MPLAB ICD 2.
- RS-232 or USB cable – To connect the MPLAB ICD 2 module to a COM or USB port on the PC.

> **Note:** Do not connect the USB cable until installation of the MPLAB IDE software and instructions for configuring the USB driver have been completed.

- Modular interface cable – To connect the MPLAB ICD 2 module to a header board, demo board or the target application.

> **Note:** Although the serial or USB communications from MPLAB IDE to the MPLAB ICD 2 can be set up without a target connection, MPLAB ICD 2 will not function as a debugger without being connected to a target.

- Header board (optional) – If an ICD device is used, a header board will be required to connect the device with on-board debug capabilities to the modular interface (and the MPLAB ICD 2). The header may then be plugged into a demo board or the target application. See **Section 4.6 "ICD/ICE Devices"**.
- Demo board or target application – To connect the device with on-board debug capabilities to the modular interface (and the MPLAB ICD 2).
- Power adapter(s) – To power MPLAB ICD 2 and the target application.

2.3 INSTALLING AND CONFIGURING MPLAB IDE FOR MPLAB ICD 2

Note: When using the USB cable, do not connect it before installing MPLAB IDE software.

To install the MPLAB IDE software, first acquire the latest MPLAB IDE installation executable (`MPxxxxx.exe`, where `xxxxx` represents the version of MPLAB IDE) from either the Microchip web site (www.microchip.com) or the MPLAB IDE CD-ROM (DS51123).

- Setting Up Communications
- Starting MPLAB IDE
- Select the Device
- Set MPLAB ICD 2 as the Debug Tool
- Configure the Interface

2.3.1 Setting Up Communications

MPLAB ICD 2 can communicate with the PC serially or using USB.

Note: Do not connect both RS-232 and USB cables to MPLAB ICD 2.

2.3.1.1 SERIAL COMMUNICATIONS

If using the RS-232 cable, connect it to the MPLAB ICD 2 and the PC. The Flow Control and FIFO settings must be set properly for a serial communications port on a PC with a Windows operating system. See **Section 8.3.1.3 “Changing Serial Port Settings”** for instructions.

2.3.1.2 USB COMMUNICATIONS

Instructions are supplied with MPLAB IDE to install the USB drivers. These instructions will pop up during MPLAB IDE installation. If you accidentally close these instructions, they may be found at: MPLAB IDE installation directory\ICD2\Drivers\Ddicd2*nn*.htm

where *nn* represents the version of Windows OS. If the instructions support more than one version, no OS version (*nn*) will be specified.

Note:	If you change USB ports/hubs, you will need to reinstall the drivers.

2.3.2 Starting MPLAB IDE

After installing the MPLAB IDE software, invoke it by using any of these methods:

- Select *Start>Programs>Microchip>MPLAB IDE vx.xx>MPLAB IDE*, where vx.xx is the version number.
- Double click the MPLAB IDE desktop icon.
- Execute the file `mplab.exe` in the `\core` subdirectory of the MPLAB IDE installation directory. For more information on using the MPLAB IDE software, see:
  - *"MPLAB® IDE User's Guide"* (DS51519) – Comprehensive guide for using MPLAB IDE.
  - *"MPLAB® IDE Quick Start Guide"* (DS51281) – Chapters 1 and 2 of the user's guide.
  - The on-line help for MPLAB IDE – The most up-to-date information on MPLAB IDE.
  - The file `Readme for MPLAB IDE.txt` – Last-minute information on each release. Included in the MPLAB IDE installation directory.
  - Also see the file `Readme for MPLAB ICD 2.txt` for the most current information about MPLAB ICD 2.

After starting MPLAB IDE, it will need to be configured for MPLAB ICD 2 use:

1. Select a device supported by MPLAB ICD 2.
2. Set MPLAB ICD 2 as the current debugger.
3. Configure the MPLAB ICD 2 RS-232 or USB interface.

2.3.3 Select the Device

Use the Device Selection dialog (*Configure>Select Device*) to select the device to be debugged with the MPLAB ICD 2. Devices fully supported by MPLAB ICD 2 will have a “green light” icon next to MPLAB ICD 2 under Microchip Tool Support. Devices with preliminary support will have a yellow light” icon. Devices not currently supported will have a “red light” icon.

2.3.4 Set MPLAB ICD 2 as the Debug Tool

Select *Debugger>Select Tool>MPLAB ICD 2* to choose MPLAB ICD 2 as the debug tool. The Debugger menu and MPLAB IDE toolbar will change to display debug options once the tool is selected. Also, the Output window will open and messages concerning ICD status and communications will be displayed on the **MPLAB ICD 2** tab.

Note:	MPLAB ICD 2 may be selected as either a debug tool (Debugger Menu) or a programmer (Programmer menu). Do not select both.

2.3.5 Configure the Interface

Go to the *Debugger>Settings*, **Communications** tab to select the USB interface or the correct COM port for RS-232 communications.

2.4 APPLYING POWER

There are a number of configurations for powering MPLAB ICD 2 and the target. The following are configuration essentials:

- When using the USB connection, MPLAB ICD 2 can be powered from the PC but a power supply must be connected to the target.
- When using the RS-232 connection to the PC, MPLAB ICD 2 must have a power supply connected to it.
- When MPLAB ICD 2 has its own power supply, it can provide a limited amount of current, up to 200 mA, at five volts to a small target board.
- MPLAB ICD 2 cannot be powered from the target board.
- Power should be applied to MPLAB ICD 2 before applying power to the target.

There are two power-up sequences for the ICD:

- Power Sequence When MPLAB ICD 2 is Providing Power to the Target
- Power Sequence When Target Has a Separate Power Supply

2.4.1 Power Sequence When MPLAB ICD 2 is Providing Power to the Target

Use this sequence to power the target board from MPLAB ICD 2. This configuration allows up to 200 mA of current at 5 volts only.

1. Apply power to MPLAB ICD 2. DO NOT power the target.
2. Start MPLAB IDE.
3. Under the Debugger menu of MPLAB IDE, select **Connect**.
4. After establishing communications with the MPLAB ICD 2, select *Debugger>Settings*.

5. In the Settings dialog, click the **Power** tab and ensure that the check box for “Power target circuit from MPLAB ICD 2” is checked. Click **OK**.

> **Note:** If any of the self-tests on the **Status** tab of the Settings dialog do not pass, it may not be possible to erase and program chips. See **Section 7.7 “Failed Self-Test Error – Vpp/Vdd High/Low”** for more information.

2.4.2 Power Sequence When Target Has a Separate Power Supply

Use this sequence to power the target board from its own power supply. For this configuration, the target power can be 2-5 volts at higher currents than can be provided by the MPLAB ICD 2 (>200 mA). Check the PIC MCU data sheet to verify the operational voltage range for the device being used.

1. Apply power to MPLAB ICD 2. DO NOT power the target.
2. Start MPLAB IDE.
3. Under the Debugger menu of MPLAB IDE, select **Connect**.
4. After establishing communications with the MPLAB ICD 2, select *Debugger>Settings*.
5. In the Settings dialog, click the **Power** tab and ensure that the check box for “Power target circuit from MPLAB ICD 2” is NOT checked. Click **OK**.
6. Power the target system and then select *Debugger>Connect*.

> **Note:** If any of the self tests on the **Status** tab of the Settings dialog do not pass, it may not be possible to erase and program chips. See **Section 7.7 “Failed Self-Test Error – Vpp/Vdd High/Low”** for more information.

2.5 CONNECTION OPTIONS

Once you have initially connected with the MPLAB ICD 2, you may continue to manually connect each time you select the ICD as a debugger (by using *Debugger>Connect*) or you may set the ICD to connect automatically (by checking "Automatically connect at start-up" on the **Status** tab of *Debugger>Settings*.)

On connection, for dsPIC® DSC devices, silicon version information will be displayed in the Output window.

2.6 SETTING UP THE ENVIRONMENT

This section provides a quick summary on MPLAB ICD 2 operation.

- Setting Debugging and Programming Options
- Creating and Building a Project

2.6.1 Setting Debugging and Programming Options

The easiest way to set up the MPLAB ICD 2 for operation is to use the MPLAB ICD 2 Setup Wizard (*Debugger>MPLAB ICD 2 Setup Wizard*.) Additionally, these dialogs allow you to set or reset debugging and programming options:

- Configuration Bits dialog (*Configure>Configuration Bits*) – Select the Configuration bits on the PIC microcontroller. For complete details about these options, see the Special Features – Configuration Bits section of the data sheet for the device being programmed.

Note: Configuration bits can be specified in the source code instead of setting them in this dialog. When doing this, every time the project is rebuilt, the Configuration bits will get reset to the values specified in the source code.

- ICD Settings dialog (*Debugger>Settings* or *Programmer>Settings*) – Set up communications, power, programming and warning message output, as well as view status, limitations, and version information.

  You have previously used the **Communications** and **Power** tabs when setting up the MPLAB ICD 2 hardware. The following tabs are useful for completing the ICD setup. For detailed information on these and the other tabs on the Settings dialog, see **Section 5.4 "Settings Dialog"**.

  - **Program** tab – Set up programming options (Select Memories, Program and External Memory Ranges, ID, and Program Options, Erase All). If programming the ID bits is desired, set the value to be programmed in the *Configure>ID Memory* dialog.
  - **Warning** tab – Determine which warning messages will be displayed. Can be useful to select messages if they are to be outputted to a file (see **Status** tab.)
  - **Status** tab – In addition to auto-connect on startup and self-test control, output messages may be selected to be logged to a file. These messages may assist users or, if necessary, Microchip technical support in finding errors.

2.6.2 Creating and Building a Project

The easiest way to create a new project is to select *Project>Project Wizard*. With the help of the Project Wizard, a new project and the language tools for building that project can be created. The wizard will guide you through the process of adding source files, libraries, linker scripts, etc. to the various "nodes" on the project window. See MPLAB IDE documentation for detailed steps on using this wizard.

After the project is created, choose *Project>Build All* to build the application. This will create object code for the application that can be programmed into the target with MPLAB ICD 2.

2.7 MPLAB ICD 2 DEVELOPMENT PROCESS

The development process with MPLAB ICD 2 consists of these steps:

1. Programming the Target Processor for Debugging.
2. Debugging The Application.
3. Modifying Target Application Code and Rebuilding the Hex File.
4. Finishing the Application: Using MPLAB ICD 2 as a Programmer.
5. Additional Programming Options.

2.7.1 Programming the Target Processor for Debugging

To program the application project's code into the MPLAB ICD 2 for debugging, follow these steps:

1. Select *Debugger>Settings* and then click the **Program** tab to set the programming options for your application.
2. Select *Configure>Configuration Bits* and set the oscillator and other Configuration bit settings appropriate to the target.
3. Select *Debugger>Program* to download your code and debug executive to the device in the application or demo board that is connected to the MPLAB ICD 2. Download time will depend on the memory used (i.e., program memory, EEPROM), size of program, and voltage.

2.7.2 Debugging The Application

To debug your code, you must execute (run) it.

- Real-Time Execution
- Breakpoints
- Step Mode Execution
- Writing Data EEPROM

2.7.2.1 REAL-TIME EXECUTION

Real-time execution occurs when the device in the demo/application board is put in MPLAB IDE's Run mode.

When the MPLAB ICD 2 is run in real time, instructions execute just as the processor would without the debugger. While in the Run mode, register displays on the screen will not update.

To execute the code in real-time:

1. Open the source file (double click on the file name in the Project Window or use *File>Open*) or program memory window (*View>Program Memory*) for viewing.
2. Select *Debugger>Run* (or click the toolbar **Run** button).

The processor will run until a breakpoint is reached or until the processor is halted by selecting *Debugger>Halt* (or clicking the toolbar **Halt** button).

2.7.2.2 BREAKPOINTS

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

You may set a breakpoint using either:

- Right mouse button menu – click on the line in code at which you wish to set a breakpoint and select Set Breakpoint.
- Breakpoint dialog – open the dialog and enter a breakpoint at a specific address

> **Note:** For most devices, only one breakpoint will be set (active) at a time, although you may specify more than one breakpoint in the breakpoint dialog (i.e., one breakpoint will be active and the others will be inactive).

Breakpoints are a function of the MPLAB IDE and discussed in more detail in documentation for that tool.

Advanced breakpoints are a function of the MPLAB ICD 2. For more information, see **Section 5.6 "Advanced Breakpoints Dialog"**.

2.7.2.3 STEP MODE EXECUTION

Step mode execution can be accessed after the processor is halted.

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

Note:	While single stepping, the MPLAB ICD 2 may not respond to interrupts. See limitations for your device to determine if it supports stepping into interrupts.

2.7.2.4 WRITING DATA EEPROM

If data EEPROM is written during program execution, the EEPROM window of MPLAB IDE will not reflect the changes. You will need to perform a READ of EEPROM memory in order to update the values in the window.

If some register or memory values do not seem correct in other windows, remember that MPLAB ICD 2 has reserved resources.

2.7.3 Modifying Target Application Code and Rebuilding the Hex File

To modify the code and rebuild the hex file:

1. Open the source file (double click on the file name in the Project Window or use *File>Open*).
2. Make the necessary changes to debug the code.
3. Rebuild the hex file using *Project>Build All*.
4. Select *Debugger>Program* to program the device with the updated hex file.

2.7.4 Finishing the Application: Using MPLAB ICD 2 as a Programmer

Once the code has been debugged and the application is running as designed, the device can be programmed without the debugger enabled. The device resources reserved for ICD operation then will be free for other use.

> **Note:** MPLAB ICD 2 may be selected as either a debug tool or a programmer. Do not select both.

To change MPLAB ICD 2 mode from debug to program:

1. Set the Debugger to "None" from the *Debugger>Select Tool* menu.
2. Select *Programmer>Select Programmer>MPLAB ICD 2* to choose MPLAB ICD 2 as a programmer. The Programmer menu and MPLAB IDE toolbar will change to display programmer options once the tool is selected. Also, the Output window will open and messages concerning ICD status and communications will be displayed on the **MPLAB ICD 2** tab.

The project should rebuild now so that all debugging modes will be disabled and the Configuration bits as defined in the source code will be programmed into the target.

To program the application project's code into the device, follow these steps:

> **Note:** If you are using the Universal Programming Module (UPM), MPLAB ICD 2 must be powered with a power supply to be used as a programmer. Also, the option "Power target circuit from MPLAB ICD 2" on the **Power** tab of *Programmer>Settings* must be checked.

1. Select *Programmer>Settings* and then click the **Program** tab to set the programming options for your application.
2. Configuration bits will be set as defined in the source files. To set manually, select *Configure>Configuration Bits* and

set the oscillator and other Configuration bit settings appropriate to the target.

3. If desired, set the ID bits by selecting *Configure>ID Memory.*
4. Select *Programmer>Blank Check* to check that the device is blank/has been erased and is ready to program.
5. Select *Programmer>Program* to download your code to the device in the application or demo board that is connected to the MPLAB ICD 2. Download time will depend on the memory used (i.e., program memory, EEPROM), size of program and voltage.

2.7.5 Additional Programming Options

As a programmer, MPLAB ICD 2 has several other programmer functions besides simply programming the target device.

2.7.5.1 VERIFYING THE PROGRAMMING

Select Verify from the Programmer menu or toolbar to verify that the device was programmed correctly based on the hex code in MPLAB IDE memory and the settings on the **Program** tab of the Settings dialog.

2.7.5.2 READING A DEVICE

Select Read from the Programmer menu or toolbar to read the device memory, based on the settings on the **Program** tab of the Settings dialog, into MPLAB IDE memory.

Memory read from a device into MPLAB IDE memory may then be saved to a file (*File>Export*). The type of file saved will depend on the type of memory read.

If you attempt to read a code-protected device, you will get a warning indicating that the device is code-protected and that the program memory may be invalid. If this happens, obtain the original hex code from a file or a non-protected device.

2.7.5.3 ERASE/BLANK CHECK A DEVICE

From the Programmer menu or toolbar, select the following items to erase and then blank check a device:

- Erase Device – Select Erase to erase the device memory.

> **Note:** Internal oscillator and bandgap calibration bits are always preserved by MPLAB ICD 2 for the erase cycle. Only during programming may their values be changed.

- Blank Check – Select Blank Check to verify that the device is erased/blank.

NOTES:

Chapter 3. MPLAB ICD 2 Tutorial

3.1 INTRODUCTION

This tutorial walks you through the process of developing a simple project using the sample program, `TUT452.asm`, found in the directory `C:\Program Files\Microchip\MPASM Suite\Example`. This program is an implementation of the PIC18F452 analog-to-digital (A/D) converter using the PICDEM™ 2 Plus Demo Board (DM163022). The program configures the A/D module to convert input from A/D channel 0 (connected to the potentiometer on the demo board) and display the results on the four PORTB LEDs (RB3:RB0).

Topics covered in this chapter:

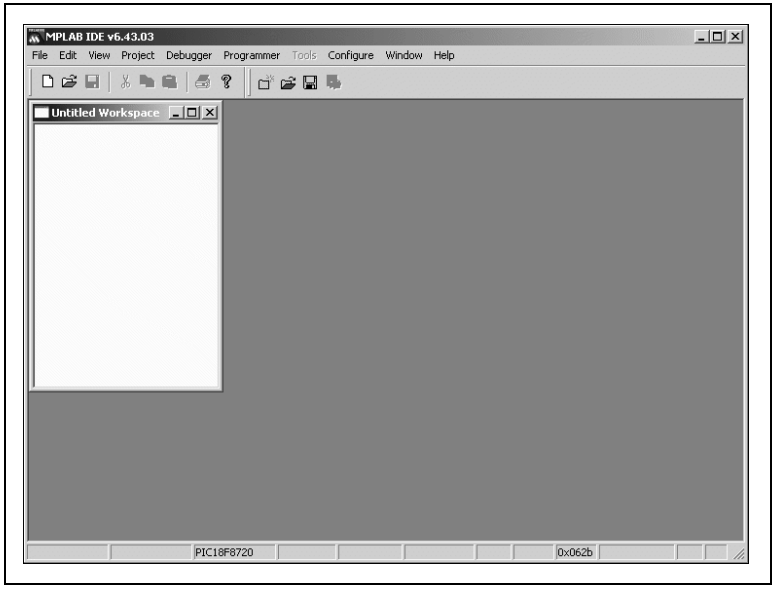
- Setting Up The Environment
- Running the Project Wizard
- Viewing the Project
- Creating a Hex File
- Setting Debug Options
- Setting Up the Demo Board
- Loading Program Code For Debugging
- Running TUT452
- Debugging TUT452
- Programming the Application
- TUT452 Main Routine and Source Code

3.2 SETTING UP THE ENVIRONMENT

Before beginning this tutorial, follow the steps in **Chapter 2. "Getting Started"** to set up the hardware and MPLAB IDE software. Some of the initial settings covered in this tutorial may already be in place as a result of the previous set up.

Once launched, the MPLAB IDE desktop should appear.

FIGURE 3-1: MPLAB® IDE



3.2.1 Selecting the Device and Development Mode

To select the device for this tutorial:

1. Select *Configure>Select Device*.
2. In the Device Selection dialog, choose PIC18F452 from the Device list box. The "light" icon next to MPLAB ICD 2 in the Microchip Tool Support section should be green.
3. Click **OK**.

To select MPLAB ICD 2 as a debugger:

1. Select *Debugger>Select Tool>MPLAB ICD 2*. The Debugger menu will then show the other debug options available. Also, the Output window will open to display connection information.

Note:	If MPLAB IDE attempts to connect on start up to the MPLAB ICD 2 but fails because a USB port or a COM port other than COM1 is being used, allow the connection to fail. The communications port will be setup in the next step.

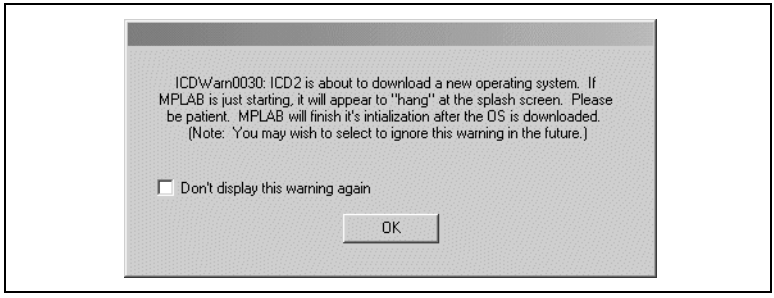
2. Select *Debugger>MPLAB ICD 2 Setup Wizard* to set up ICD operation:
   a) Select a port – COM or USB
   b) Select target power supply
   c) Enable auto-connection feature
   d) Enable automatic OS download feature
   e) For ROMless devices, set up external memory
3. Select *Debugger>Connect* to connect to the MPLAB ICD 2.

3.2.2 Updating MPLAB ICD 2 Firmware (Operating System)

Depending on the version of MPLAB IDE or the device selected, a message may appear indicating that the firmware needs to be updated. MPLAB IDE will automatically install new firmware.

The update dialog may look like this:

FIGURE 3-2: UPDATING MPLAB® ICD 2 FIRMWARE DIALOG



Also, since different MPLAB ICD 2 firmware is used for different families of devices, this dialog may appear when switching to a different device.

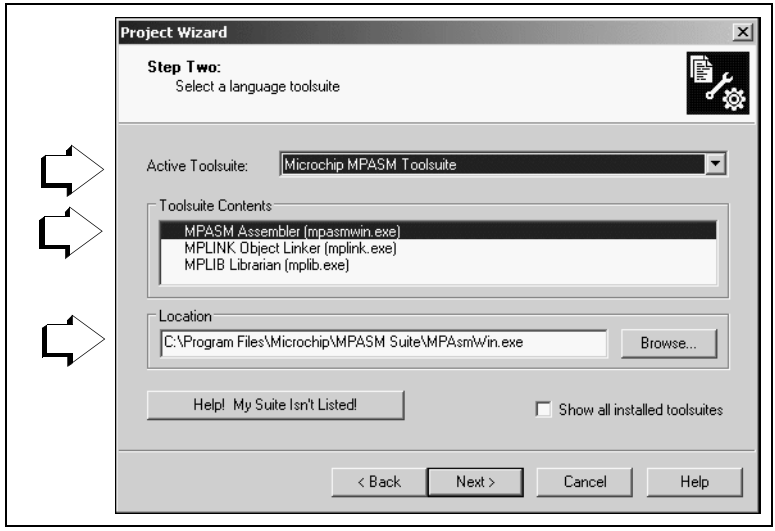
The firmware can be manually changed by following the steps in **Section 4.2 "Upgrading the MPLAB ICD 2 Firmware (Operating System)"**.

3.3 RUNNING THE PROJECT WIZARD

The MPASM assembler will be used in this project.

1. To set up the first project, select *Project>Project Wizard*.
2. Proceed to the second dialog of the wizard. The PIC18F452 should be selected.
3. Proceed to the next dialog of the wizard to set up MPASM assembler as the language tool. In the "Active Toolsuite" pull-down, select "Microchip MPASM Toolsuite." Make sure that the tools are set to the proper executables, by default located in the directory `C:\Program Files\Microchip\MPASM Suite`. MPASM assembler should be pointing to `mpasmwin.exe`, MPLINK linker should be pointing to `mplink.exe`, and MPLIB librarian should be pointing to `mplib.exe`.

FIGURE 3-3: WIZARD TOOLSUITE DIALOG

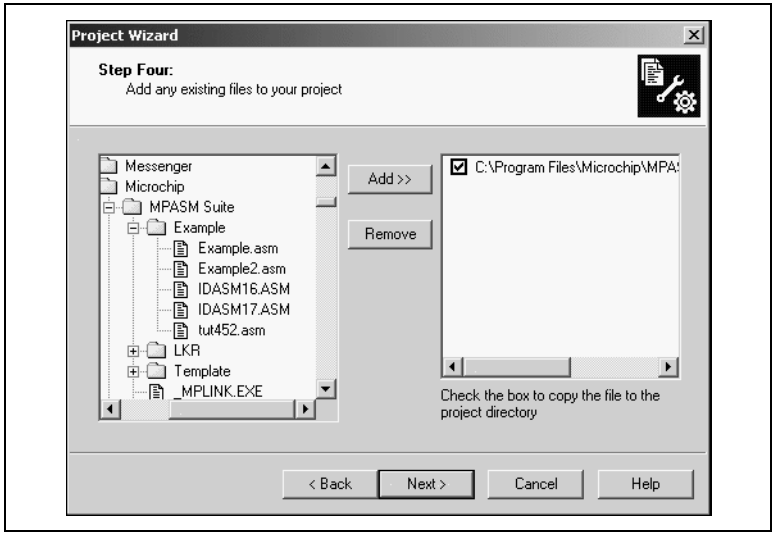


4. Proceed to the next dialog of the wizard to give a name and location to your project. You may **Browse** to find a location.
5. Proceed to the next dialog of the wizard where project files can be added. Files can also be added later if something is missed.

   For this example, go to `C:\Program Files\Microchip\MPASM Suite\Example\TUT452.ASM`.

   Click on `TUT452.ASM` to highlight it, then Click on **ADD>>** to add it to the right pane. Then click in the checkbox next to the file in this pane to copy it into the project directory.

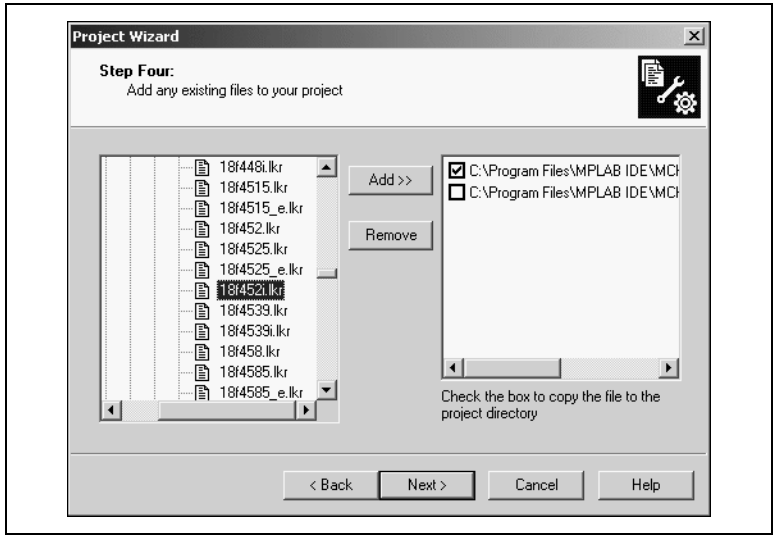
FIGURE 3-4: WIZARD ADD FILES



6. The second file needed for this project is the linker script. Click on the `C:\Program Files\Microchip\MPASM Suite\LKR` folder to expand it, and then scroll down to select the file `18F452i.lkr`. Make sure that the file name has an "`i`."

   Press **ADD>>** to move the linker script to the list on the right. You will not be copying this file into the project directory.

FIGURE 3-5: WIZARD ADD FILES – LINKER SCRIPT

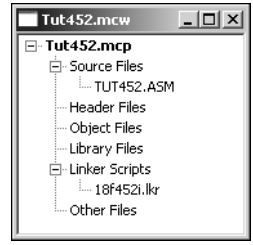


7. Proceed to the Summary screen. If you have made any errors, click **<Back** to return to a previous wizard dialog. If everything is correct, click **Finish**.

3.4 VIEWING THE PROJECT

After exiting the wizard, the MPLAB IDE desktop will again be visible. Close all other windows on the desktop to see the Project window.

FIGURE 3-6: PROJECT WINDOW

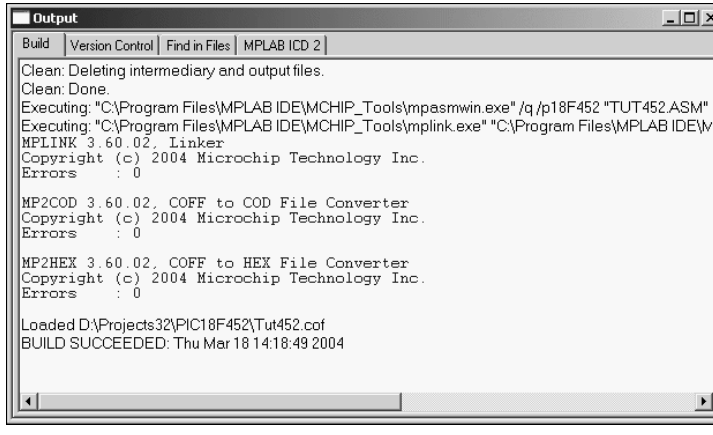


Additional files can be added to the project using the project window. Right click on any line in the project window tree to pop up a menu with additional options for adding and removing files.

3.5 CREATING A HEX FILE

To create a hex file for debugging, select *Project>Build All* or right click on the project name in the project window and select "Build All" from the popup menu. MPASM assembler always makes a `.hex` file with the same name as the source `.asm` file.

FIGURE 3-7: OUTPUT WINDOW



3.6 SETTING DEBUG OPTIONS

Before you begin debugging your code, you will need to set up or check the default settings of several items.

3.6.1 Configuration Bits

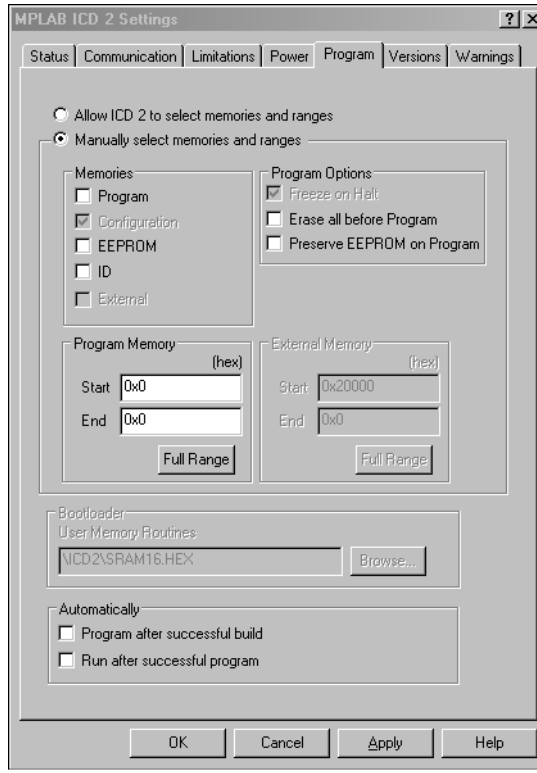
To set Configuration bits to be programmed into the device, select *Configure>Configuration Bits*. By clicking on the text in the "Settings" column, these can be changed. In this dialog, the following Configuration bits should be set for this tutorial:

- Oscillator – EC-OS2 as RA6 (this is for PICDEM 2 Plus, if using another target, change to match)
- OSC Switch Enable – Disabled
- Power-Up Timer – Enabled
- Brown-Out Detect – Disabled
- Watchdog Timer – Disabled
- CCP2 MUX – RC1
- Stack Overflow Reset – Disabled
- Low Voltage Programming – Disabled
- All other Configuration bits should be disabled.

3.6.2 Programming Options

To set program options for this tutorial, select *Debugger>Settings* and click on the **Program** tab.

FIGURE 3-8: MPLAB® ICD 2 SETTINGS DIALOG, PROGRAM TAB



- The "Memories" section should have "Program" checked, and "EEPROM" and "ID" unchecked. When using MPLAB ICD 2 as a debugger, Configuration bits will always be programmed and the "Configuration" box will be checked and grayed out.
- For the PIC18FXX2 devices, all memory will be erased each time the chip is programmed. Therefore, in the "Program Options" section, "Erase all before Program" will have no effect.

- The “Program Memory” addresses (“Start” and “End” address) set the range of program memory that will be read, programmed or verified. Click **Full Range** to set the address range to the maximum program memory available based on the device selected. The end of program memory is adjusted to leave room for the MPLAB ICD 2 debug executive. Programs cannot extend beyond this upper limit while using MPLAB ICD 2 as a debugger.

For additional information on all the options on the **Program** tab, see **Section 5.4.5 “Program Tab”**.

3.7 SETTING UP THE DEMO BOARD

Before beginning to debug, make sure the PICDEM 2 Plus Demo Board is set up as follows:

- EC OSC option has been selected by removing jumper J7
- LEDs have been enabled using jumper J6
- Power is supplied

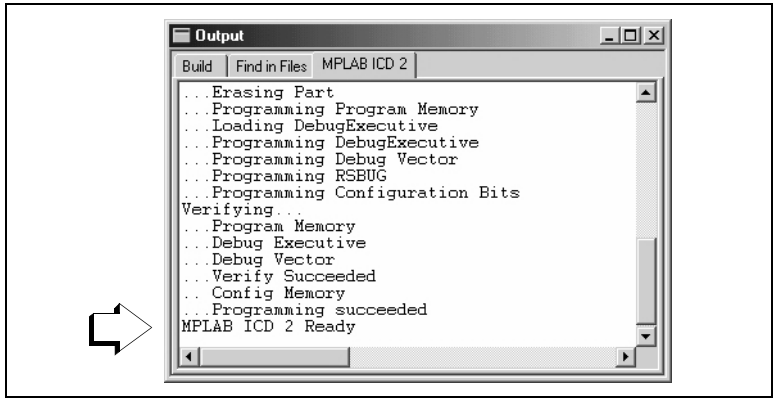
3.8 LOADING PROGRAM CODE FOR DEBUGGING

Select *Debugger>Program* to program `TUT452.hex` into the PIC18F452 in the PICDEM 2 Plus demo board.

> **Note:** The debug executive code is automatically programmed in upper program memory for MPLAB ICD 2 debug functions. Debug code must be programmed into the target PIC MCU to use the in-circuit debugging capabilities of the MPLAB ICD 2.

Programming may take a couple of minutes. During programming, the **MPLAB ICD 2** tab of the Output dialog shows the current phase of operation. When programming is complete, the dialog should look similar to Figure 3-9.

FIGURE 3-9: OUTPUT WINDOW – MPLAB® ICD 2 TAB



3.9 RUNNING TUT452

MPLAB ICD 2 executes in Real-Time or in Step mode.

- Real time execution occurs when the PIC18F452 in the PICDEM 2 Plus demo board is put in the MPLAB IDE's Run mode.
- Step mode execution can be accessed after the processor is halted.

These toolbar buttons can be used for quick access to commonly used debug operations.

Debugger Menu	Toolbar Buttons
Run	
Halt	
Animate	
Step Into	
Step Over	
Step Out	
Reset	

Begin in Real-Time mode:

1. Open the `TUT452.asm` file (double click on the file name in the Project Window or use *File>Open*).
2. Select *Debugger>Run* (or click the **Run** toolbar button).
3. On the demo board, turn the arrow on the potentiometer (RA0). Observe the LEDs. If the program were working correctly, the user would see a binary representation of the voltage value across the potentiometer. However, there is a bug in the TUT452 program. **Section 3.10 "Debugging TUT452"** will detail debugging the code and correcting it.
4. Select *Debugger>Halt* (or click the **Halt** toolbar button) to stop the program execution.
5. Select *Debugger>Reset* to reset the program.

3.10 DEBUGGING TUT452

Any of the following can prevent the TUT452 program from working.

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

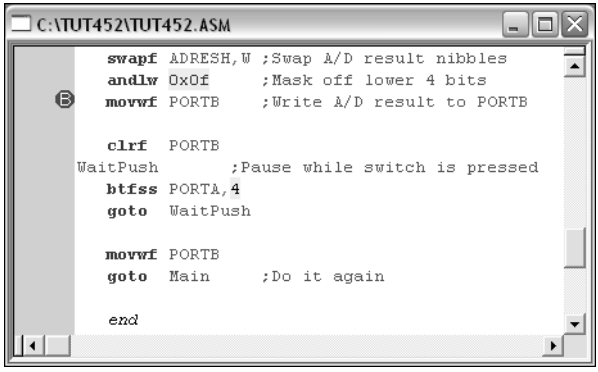
To explore the first possibility, set a breakpoint at the line of the file that writes the value of A/D result to PORTB.

1. Highlight or place the cursor on the following line of code from `TUT452.asm`:

   ```
   movwf PORTB ;Write A/D result to PORTB
   ```

2. Right click to display a shortcut menu.
3. Select *Set Breakpoint* from the shortcut menu. This line is now marked as a breakpoint (B in red stop sign) as shown in Figure 3-10.

FIGURE 3-10: SET BREAKPOINT



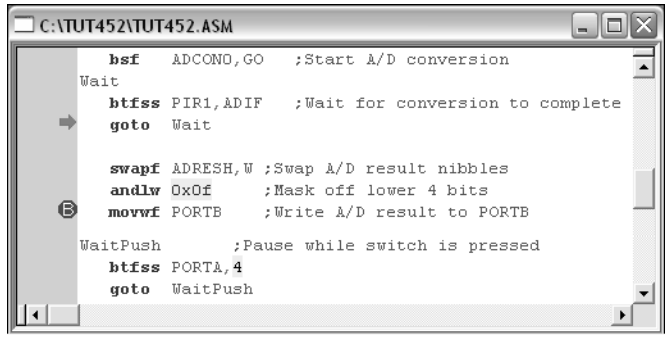
4. Select *Debugger>Run* (or click the **Run** toolbar button) to run the program once again in Real-Time mode.

   A breakpoint stops a program's execution when the program executes the line marked as a breakpoint. However, the sample program is not halting.

5. Select *Debugger>Halt* (or click the **Halt** button on the toolbar) to halt the program now.

   In the source code window, the sample program will have halted on one of the two lines in the Wait routine as shown in Figure 3-11 (at the arrow).

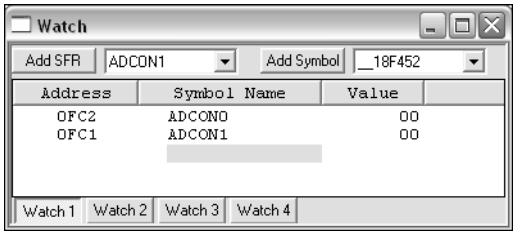
FIGURE 3-11: PROGRAM HALTED



Based on the halt location and the fact that the program never reaches the breakpoint, it can be concluded that the problem is in the A/D conversion. The A/D flag for conversion complete is not being set. A/D conversion initialization and setup occurs at the beginning of the program.

6. To reset the program, select *Debugger>Reset*. The first instruction should be indicated with a green arrow.
7. Open a new Watch window to watch the A/D register value change as the program executes. Select *View>Watch*. The Watch dialog opens with the **Watch_1** tab selected. Select “ADCON0” from the list next to Add SFR, and then click the button. ADCON0 is added to the Watch window. Repeat for ADCON1. The selected symbols should now be visible in the Watch window as shown in Figure 3-12.

FIGURE 3-12: WATCH WINDOW



8. In the `TUT452.asm` source code, set a breakpoint at the first instruction after Start. Highlight or place the cursor in the following line of code from `TUT452.asm`:

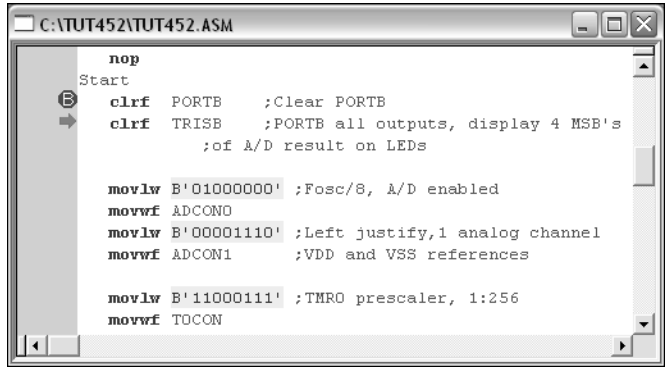
   ```
   clrf PORTB ;Clear PORTB
   ```

   Right click to display a shortcut menu. Select *Set Breakpoint* from the shortcut menu. This line will now be marked as a breakpoint.

9. Select *Debugger>Run* (or click the **Run** toolbar button) 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 indicated as shown in Figure 3-13.

FIGURE 3-13: PROGRAM HALTED AFTER BREAK



10. Single step five times, select *Debugger>Step* (or click the **Step** toolbar button) to single step through the program. The following line of code should be indicated when finished:

    ```
    movlw B'11000111'  ;TMR0 prescaler, 1:256
    ```

11. Examine the values of the registers ADCON0 and ADCON1 in the Watch window. Notice that ADCON0 has a value of 40 HEX. This corresponds to the binary value designated in the program, but is this value correct?

    A review of the PIC18F452 Data Sheet (DS39564) section on A/D, indicates 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
    ```

12. Select *File>Save* to save the changes.
13. Select *Project>Build All* to rebuild the project. A message will indicate that the program has been rebuilt. The MPLAB ICD 2 must be reprogrammed for the changes to take effect.

14. Select *Debugger>Program* to reprogram the MPLAB ICD 2 with the changes. When the MPLAB ICD 2 dialog indicates '...Programming succeeded', the program is ready to run again.
15. Right click on the line of code that previously had the breakpoint (now indicated by a red stop sign outline). Select *Remove Breakpoint*.
16. Select *Debugger>Run* (or click the **Run** toolbar button) to run the program in Real-Time mode. 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 2 and MPLAB IDE debugging functions, users can successfully find and fix the problems in their code.

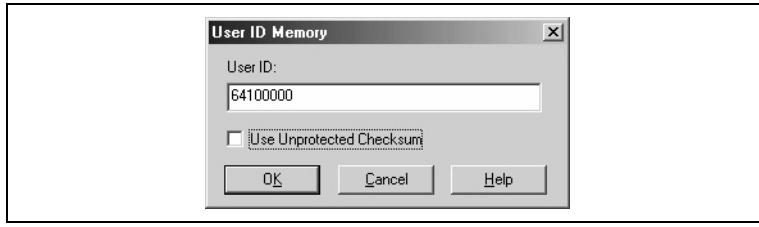
3.11 PROGRAMMING THE APPLICATION

When the program is successfully debugged and running, usually the next step is to program the PIC MCU for stand alone operation in the finished design. When doing this, the resources reserved by the ICD are released for use by the application.

To program the application follow these steps:

1. Disable MPLAB ICD 2 as a debug tool by selecting *Debugger>Select Tool>None*.
2. Select MPLAB ICD 2 as the programmer in *Programmer>Select Tool* menu.
3. *Optional:* Set up the ID in *Configure>ID Memory (see Figure 3-14).*

FIGURE 3-14: USER ID MEMORY DIALOG



4. Set up the parameters for programming on the *Programmer>Settings* **Program** tab.
5. Select *Programmer>Program*.

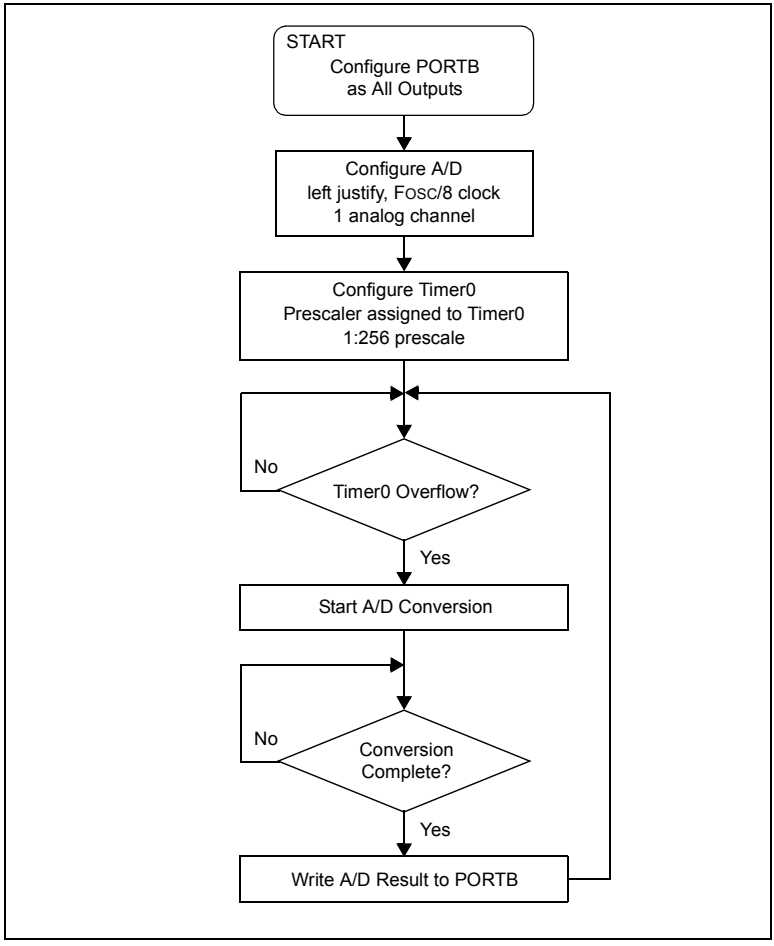
Now MPLAB ICD 2 can reset and run the target (MPLAB ICD 2 can also be disconnected from the target; after selecting the **Reset** button, the application will run).

3.12 TUT452 MAIN ROUTINE AND SOURCE CODE

The main routine of `TUT452.asm` (Figure 3-14) begins by configuring PORTB, 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 "*PIC® 18C MCU Family Reference Manual*" (DS39500).

FIGURE 3-15: TUT452 MAIN ROUTINE FLOW



```
;*************************************
;*              TUT452.ASM           *
;*************************************

          list p=18f452
          include "p18f452.inc"

Reset_Vector                        ;Put a GOTO at
          code    0x0               ;reset address
          goto    Start

Start     code    0x0002A           ;Start app beyond
                                    ;vector area
          clrf    PORTB             ;Clear PORTB
          clrf    TRISB             ;PORTB all outputs,
                                    ;display 4 MSB's of
                                    A/D
                                    ;result on LEDs
          movlw   B'01000000'       ;Fosc/8,
                                    ;A/D enabled
          movwf   ADCON0
          movlw   B'00001110'       ;Left justify,
                                    ;1 analog channel
          movwf   ADCON1            ;VDD and VSS ref's
          movlw   B'11000111'       ;TMR0 prescaler,1:256
          movwf   T0CON

Main
          btfss   INTCON,TMR0IF     ;Wait for Timer0
                                    ;to timeout
          goto    Main
          bcf     INTCON,TMR0IF
```

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

          swapf   ADRESH,W        ;Swap A/D result nib-
                                  bles
          andlw   0x0f            ;Mask off lower 4 bits
          movwf   PORTB           ;Write A/D result
                                  ;to PORTB
          clrf    PORTB

WaitPush                          ;Pause while switch
          btfss   PORTA,4         ;is pressed
          goto    WaitPush

          movwf   PORTB
          goto    Main            ;Do it again

          end
```

Chapter 4. Additional Topics

4.1 INTRODUCTION

As you work with MPLAB ICD 2 to develop your application, these are some additional topics that you may find useful.

Topics covered in this chapter are:

- Upgrading the MPLAB ICD 2 Firmware (Operating System)
- Creating/Loading a Hex File
- Special Linker Script Files
- ROMless Device Considerations
- ICD/ICE Devices

4.2 UPGRADING THE MPLAB ICD 2 FIRMWARE (OPERATING SYSTEM)

By default, MPLAB IDE will automatically download the correct and most up-to-date operating system for your selected device to the MPLAB ICD 2. This feature is selected from the Settings dialog, **Status** tab, by checking the checkbox "Automatically download firmware if needed". If you have unchecked this checkbox, you may manually download the operating system (not recommended) as follows:

1. Select *Debugger>Download ICD2 Operating System* or *Programmer>Download ICD2 Operating System*. The Select ICD Firmware File dialog will open.
2. Select the firmware file to download from the list or browse for it. The file will be of the form `icdxxxxxx.hex`, where `xxxxxx` is the version number. See the readme file for information on which file supports which devices.

3. Click **Open**. MPLAB IDE will download the new operating system to the MPLAB ICD 2.

Note:	If the wrong firmware is selected, MPLAB ICD 2 will experience unknown errors.

4.3 CREATING/LOADING A HEX FILE

Once the hex file is created/imported, you can use MPLAB ICD 2 to debug and/or program your code.

4.3.1 Creating a Hex File

If you need to create your hex code, MPLAB IDE offers a complete development environment.

To create a hex file in MPLAB IDE:

- Set up a project and related workspace to develop your application. For information on using projects and workspaces, please see MPLAB IDE Help.
- Use MPLAB Editor to create or import source code.
- Use language tools specified under the Project menu to assemble/compile your source code into hex code.

4.3.2 Loading a Hex File

If you already have a hex code file that you would like to program into your device, (i.e., you have used a tool other than the MPLAB IDE to generate your hex code), you may load the hex code to MPLAB IDE.

To load a hex file into MPLAB IDE:

- Use the *File>Import* command.

4.3.3 Hex File Requirements

Whatever tool you use to develop your hex code, please keep in mind the following:

- Your code should be limited depending on the device because a portion of program memory on the device is reserved.
- The start of EEPROM data memory needs to be specified. For most PIC MCU's, the start should be at 0x2100 (org H'2100'). For PIC18FXXXX devices, the start should be at 0xF00000 (org H'F00000'). Please check the programming specification for your selected device to determine the correct address.

4.4 SPECIAL LINKER SCRIPT FILES

MPLAB IDE provides separate linker script files that reserve the resources used by the MPLAB ICD 2. There is a separate linker file for each supported device. The filenames contain the name of the device, followed by an "`i`".

Examples:

- `18F452i.lkr` – for the PIC18F452 device
- `18F4580i_e.lkr` – for the PIC18F4580 device with extended memory

Customers that are using the MPLAB ICD 2 should use the linker file supplied for the device they are programming, instead of the standard linker files.

4.5 ROMLESS DEVICE CONSIDERATIONS

Working with ROMless (no on-chip program memory) devices is different from working with regular devices. You must create the routines that read/write memory. For more information, see:

- Implementing the External Memory Interface on PIC18C601/801 Application Note (AN778)

4.5.1 PICDEM 18R

There is a Microchip demonstration board, PICDEM 18R, available for development using ROMless devices. Even if you do not purchase the demo board kit (DM163006), you can download the software and documentation for free from the Microchip website and use it to help you develop your own code:

- PICDEM 18R Demonstration Board User's Guide (DS39565)
- PICDEM 18R Schematics
- PICDEM 18R Software

MPLAB IDE contains files that may be used with MPLAB ICD 2 and PICDEM 18R in the `\ICD2` subdirectory when mapping External Memory for ROMless devices.

4.5.2 Additional Reference Documents

- PIC18C601/801 Data Sheet (DS39541)
- PIC18CXXX OTP In-Circuit Serial Programming (DS39028)
- PIC18XXXX MCU Family Product Overview (DS30327)

4.5.3 External Memory

This section only applies to ROMless (PIC18C601/801) devices.

File	Click the **Browse** button to select the location of the WriteProgramWord and EraseProgramMemory files. See PICDEM 18R Files. You must use a different memory routine file for the 8-bit bus width and the 16-bit bus width devices. The memory routine files must match the following naming conventions: *<Memory Type><Bus Width>.<extension>* where *<Memory Type><Bus Width>* does not exceed 8 characters. Valid 16-bit filename examples: `29F16016.hex` `SRAM16.hex` Valid 8-bit filename examples: `29F16008.hex` `SRAM08.hex` See the `\ICD801\Source` folder for information on writing your custom routines. These code routines will be used to program memory. They must be relocatable and comply with the format used in the included source files.
Use Default Program Routines	Select this option to use the default memory routines provided with MPLAB IDE instead of providing your own memory routines. See PICDEM 18R Files.

4.5.4 PICDEM 18R Files

There is a folder called `\ICD2` that was copied into the MPLAB IDE installation directory. This folder has two files which can be used with the PICDEM 18R demo board (DM163006):

- SRAM16.HEX – allows program download to the static RAM
- 29F16016.HEX – allows program download to the Flash memory

When using PICDEM 18R, you must use one of the above files in the "Location of WriteProgramWord and EraseProgramMemory" dialog on the MPLAB ICD 2 Advanced Dialog. Also, you must remember to do an erase before programming as this is not done automatically.

For your own design, which probably has different programming algorithms for the memory on your target, you must substitute your own memory read/write routine in order for the MPLAB ICD 2 to download code. See the PICDEM 18R documentation for information on writing your custom routines. These code routines will be used to program your memory and care must be taken to ensure they are relocatable and comply with the format used in the included source files.

4.6 ICD/ICE DEVICES

Special ICD versions of some devices (*Device*-ICD) are available to provide the clock, data and MCLR functions required for in-circuit debugging on dedicated pins. Therefore, the ICD version necessarily has a larger pin count than the original device. However, no pins are lost to in-circuit debug operation.

Also, some devices have ICE versions (*Device*-ICE) for use with an emulator. However, these device may be used with MPLAB ICD 2 to provide the functionality of an ICD device.

Currently, the following debug configurations are available:

- *Device* has no built-in debug circuitry. Therefore, *Device*-ICD must be used for development.
- *Device* has built-in debug circuitry for development. No *Device*-ICD/ICE available.
- *Device* has built-in debug circuitry for development. *Device*-ICD/ICE available, with built-in debug circuitry plus additional pins/memory for development.

Chapter 5. ICD Function Summary

5.1 INTRODUCTION

When you select MPLAB ICD 2 from the Debugger menu, debugging functions will be added to MPLAB IDE.

When you select MPLAB ICD 2 from the Programmer menu, programming functions will be added to MPLAB IDE.

The functions made available are summarized here.

MPLAB ICD 2 functions are added to MPLAB IDE as follows:

- Debugging Functions
- Programming Functions
- Settings Dialog
- Setup Wizard
- Advanced Breakpoints Dialog

5.2 DEBUGGING FUNCTIONS

When you select MPLAB ICD 2 from the Debugger menu, debug items will be added to the following MPLAB IDE functions:

- Debugger Menu
- Toolbars
- Right Mouse Button Menus

5.2.1 Debugger Menu

Run F9

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

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

Animate

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

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

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

Halt F5

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

Step Into F7

Single step through program code.

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

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

Note: Do not step into a `SLEEP` instruction.

Step Over F8

Not available on MPLAB ICD 2. Performs the same as Step Into.

Step Out

Not available on MPLAB ICD 2.

Reset F6

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

Breakpoints F2

Open the Breakpoint dialog. Set multiple breakpoints in this dialog; however, only one breakpoint is enabled at a time.

Note:	You may also right click on a line of code to set a breakpoint.

Advanced Breakpoints

Open the Advanced Breakpoint dialog. Setup advanced breakpoint features for devices that support this ICD feature.

MPLAB ICD 2 Setup Wizard

Launch the wizard to help you set up the MPLAB ICD 2.

Program

Download the debug executive module (if selected in **Program** tab) and your code to the target device.

Read

Read target memory. Information uploaded to MPLAB IDE.

Read EEPROM

Read device EEPROM data and then reset.

Abort Operation

Abort any programming operation (e.g., program, read, etc.) Terminating an operation will leave the device in an unknown state.

Connect

Establish communications between the MPLAB ICD 2 and PC.

Download ICD2 Operating System

Download MPLAB ICD 2 operating system.

Settings

Opens the MPLAB ICD 2 Settings dialog. Set up communication, output file, power and program options. Also, find out information about the current system configuration and device limitations.

5.2.2 Toolbars

When the MPLAB ICD 2 is selected as a debugger, these toolbars are displayed in MPLAB IDE:

- Standard debug toolbar (Run, Halt, Step Into, Step Over, Reset). See MPLAB IDE Help for more information.
- MPLAB ICD 2 debug toolbar (Program Target Device, Read Target Device, Read device EEPROM, Reset and Connect to ICD).

5.2.3 Right Mouse Button Menus

The following will appear on the right mouse menus in code displays, such as program memory and source code files:

Set/Remove Breakpoint

Set or remove a breakpoint at the currently selected line.

Enable/Disable Breakpoint

Enable or disable a breakpoint at the currently selected line.

Breakpoints

Remove, enable or disable all breakpoints.

Run To Cursor

Run the program to the current cursor location. Formerly Run to Here.

Set PC at Cursor

Set the Program Counter (PC) to the cursor location.

5.3 PROGRAMMING FUNCTIONS

When you select MPLAB ICD 2 from the Programmer menu, program items will be added to the following MPLAB IDE functions:

- Programmer Menu
- Toolbar

5.3.1 Programmer Menu

MPLAB ICD 2 Setup Wizard

Launch the ICD setup wizard.

Program

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

Read

Read specified memory areas: program memory, Configuration bits, ID locations and/or EEPROM data.

Verify

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

Erase Part

Erase all data on the PIC MCU device including memory, ID and Configuration bits.

Blank Check

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

Release from Reset

Set $\overline{\text{MCLR}}$ to V_{DD}.

Hold in Reset

Set $\overline{\text{MCLR}}$ to ground (zero.)

Abort Operation

Abort any programming operation (e.g., program, read, etc.) Terminating an operation will leave the device in an unknown state.

Connect

Establish communications between the MPLAB ICD 2 and PC.

Download ICD2 Operating System

Download MPLAB ICD 2 operating system.

Settings

Opens the MPLAB ICD 2 Programmer dialog. Set up communication, output file, power and program options. Also, find out information about the current system configuration and device limitations.

5.3.2 Toolbar

Program target device

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

Read target device

Read specified memory areas: program memory, Configuration bits, ID locations and/or EEPROM data.

Verify target device memory

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

Erase target device

Erase all data on the PIC MCU device including memory, ID, and Configuration bits.

Verify target device is erased

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

Reset and Connect to ICD

Establish communications between the MPLAB ICD 2 and PC.

5.4 SETTINGS DIALOG

Select either *Debugger>Settings* or *Programmer>Settings* to open the Settings dialog and set up the MPLAB ICD 2.

- Status Tab
- Communication Tab
- Limitations Tab
- Power Tab
- Program Tab
- Versions Tab
- Warnings Tab

5.4.1 Status Tab

This tab of the MPLAB ICD 2 Programmer dialog allows you to set connection and message options.

Connect Status	
Connect Status	Shows the current connection status.
Automatically Connect at Startup	Enable/disable auto-connect on startup between MPLAB® IDE and MPLAB ICD 2.
Automatically download firmware if needed	If checked, the correct firmware for the selected device will automatically be downloaded to the MPLAB ICD 2. If unchecked, you will be prompted before firmware is downloaded.
Messages	
Output to Debug File	Enable/disable the outputting of messages to a file.
Self-Test	
Run Self-Test	Execute a self-test on the MPLAB ICD 2.
Self-Test Results	The results of the self-test are displayed in Target V_{DD}, Module V_{PP}, $\overline{MCLR}$ Gnd, $\overline{MCLR}$ V_{DD}, and $\overline{MCLR}$ V_{PP}.

5.4.2 Communication Tab

This tab of the MPLAB ICD 2 Programmer dialog allows you to set how the MPLAB ICD 2 and the PC communicate.

COM Port	Select the COM port (COM1, COM2, COM3 or COM4) for serial communications or USB for universal serial bus communications. **Default: COM1**
Baud Rate	The initial COM port baud rate for MPLAB® ICD 2 communications is 19200. After communications have been established, you may wish to select the 57600 option for improved performance. If you have increased communication errors at this speed, change back to the default. **Default: 19200**

5.4.3 Limitations Tab

This tab of the MPLAB ICD 2 Programmer dialog allows you to view MPLAB ICD 2 limitations for your selected device. Brief limitations are shown in the text box under the device. For more detailed limitations, click **Details**.

5.4.4 Power Tab

This tab of the MPLAB ICD 2 Programmer dialog allows you to view MPLAB ICD 2 and target power parameters and set the MPLAB ICD 2 to power the target.

View values for Target V_{DD}, Target V_{PP} and MPLAB ICD 2 V_{PP}. If you know the values have changed since you opened the dialog, you may click **Update** to see these values update immediately or wait for one of the conditions specified on the dialog.

Click in the checkbox to enable/disable “Power target circuit from MPLAB ICD 2 (5V V_{DD})”.

5.4.5 Program Tab

This tab of the MPLAB ICD 2 Programmer dialog allows you to set up debug/programming options.

- Allow ICD 2 to select memories and ranges – MPLAB ICD 2 uses your selected device and default settings to determine what to program.
- Manually select memories and ranges – you select the type and range of memory to program.

TABLE 5-1: MANUAL SELECTION OPTIONS

Memories	
Program	Check to program Program Memory into target.
Configuration	Check to program Configuration Bits into target. **Note:** This memory is always programmed when in Debug mode.
EEPROM	Check to erase and then program EEPROM memory on target. Uncheck to erase EEPROM memory on target.
ID	Check to program ID Memory into target.
External	Check to program External Memory into target. **Note:** To select External, your device must support external memory and you must have enabled "Use External Memory" in the External Memory Settings dialog (*Configure>External Memory*).
Program Options	
Freeze on Halt	Set/clear all peripherals to freeze on halt.
Erase all before Program	Check to erase all memory before programming begins. Unless programming new or already erased devices, it is important to have this box checked. If not checked, the device is not erased and program code will be merged with the code already in the device.

TABLE 5-1: MANUAL SELECTION OPTIONS (CONT.)

Preserve EEPROM on Program	Check to keep EEPROM memory on target from being overwritten on programming. Target EEPROM memory values are read into MPLAB® IDE, erased from the target and then written back to the target. Uncheck to use EEPROM checkbox functionality under Memories.
Program Memory	
Start, End	The starting and ending hex address range in program memory for programming, reading, or verification. If you receive a programming error due to an incorrect End address, perform a reconnect, correct the End address and program again. **Note:** The address range does not apply to the Erase function. The Erase function will erase all data on the device.
Full Range	Enter the full range of device program memory in hex.
External Memory	
Start, End	The starting and ending hex address range in program memory for programming, reading, or verification. **Note:** To select External, your device must support external memory and you must have enabled “Use External Memory” in the External Memory Settings dialog (*Configure>External Memory*). If you receive a programming error due to an incorrect End address, perform a reconnect, correct the End address and program again. **Note:** The address range does not apply to the Erase function. The Erase function will erase all data on the device.
Full Range	Enter the full range of device program memory in hex.

TABLE 5-2: BOOTLOADER OPTIONS

Enter your own bootloader or Browse for the file in User Memory Routines.

TABLE 5-3: AUTOMATIC OPTIONS

Program after successful build	If the project builds successfully (no errors), automatically program the device.
Run after successful program	If the device programs successfully (no errors), run the program.

Note: If you are using the MPLAB ICD 2, you must check the "Automatically Program after successful build" in the MPLAB ICD 2 settings dialog on the **Program** tab. Do not check "Automatically Run after successful program."

5.4.6 Versions Tab

This tab of the MPLAB ICD 2 Programmer dialog allows you to view the version numbers for MPLAB ICD 2 elements.

MPLAB® ICD 2 Version	PC software DLL version
Firmware	MPLAB ICD 2 operating system
Firmware Type	Device type for firmware
Firmware Version	MPLAB ICD 2 operating system version
Bootloader Version	MPLAB ICD 2 bootloader version
Debug Exec Version	Debug executive module version

5.4.7 Warnings Tab

This tab of the MPLAB ICD 2 Programmer dialog allows you to select which MPLAB ICD 2 warning messages you wish to display in the Output window. This is useful for disabling warnings that you know you will ignore and do not want to view.

Click in the checkbox next to a warning in the list to enable/disable the warning. Checked is enabled; unchecked is disabled.

5.5 SETUP WIZARD

A setup wizard is available from *Debugger>MPLAB ICD 2 Setup Wizard* to help walk you through setting up your MPLAB ICD 2.

- Setup Wizard – Welcome
- Setup Wizard – Select a Port
- Setup Wizard – Select Target Power
- Setup Wizard – Enable Auto Connect
- Setup Wizard – Enable Automatic OS Download
- Setup Wizard – External Memory
- Setup Wizard – Summary

5.5.1 Setup Wizard – Welcome

When you first select MPLAB ICD 2 as a debugger or programmer, the MPLAB ICD 2 Setup Wizard will open.

Follow the dialogs in the MPLAB ICD 2 Wizard to set up MPLAB ICD 2 for use with MPLAB IDE.

Click **Next** to continue.

5.5.2 Setup Wizard – Select a Port

Step 1: Select a communications method – Select a communications port to which you will attach the MPLAB ICD 2.

- COM Port – Select either a serial port (COM1-COM4) or a USB port (USB).
- Baud Rate – For a serial port, select 19200 or 57600.

> **Note:** You may wish to try the higher baud rate to see if you can communicate at this speed. If you experience errors, return to the lower baud rate.

Click **Next** to continue.

5.5.3 Setup Wizard – Select Target Power

Step 2: Select target power source – Choose from where the target board will get its power.

- Target has own power supply – The target board will use its own power supply.
- Power target from the MPLAB ICD 2 – You may use the MPLAB ICD 2 to power the target board as long as:
  - MPLAB ICD 2 has a power supply (the USB cannot provide enough power for both the MPLAB ICD 2 and a target board.)
  - Target board power requirements are within what the MPLAB ICD 2 provide.

Click **Next** to continue.

5.5.4 Setup Wizard – Enable Auto Connect

Step 3: Enable auto-connection – Set up MPLAB IDE to automatically connect to MPLAB ICD 2 on project start-up.

- MPLAB IDE automatically connects to MPLAB ICD 2 – Check to connect on project start-up. Uncheck to connect manually, i.e., using *Debugger>Connect*.

CAUTION

If there is a device other than the MPLAB ICD 2 on the port when you auto-connect, damage to that device may result.

Click **Next** to continue.

5.5.5 Setup Wizard – Enable Automatic OS Download

Step 4: Enable autodownload of operating systems – Automatically download the correct OS for the selected device.

> **Note:** There are different operating systems for different devices.

- Required operating system automatically downloaded to MPLAB ICD 2 – Check to automatically download the correct OS for your selected device. The OS will be downloaded when MPLAB IDE detects that the current MPLAB ICD 2 operating system is not correct for the selected device or there is a newer version of this operating system.

  Uncheck to be prompted to download the OS. If you choose not to download the OS, it may be downloaded manually later using *Debugger>Download ICD Operating System*.

> **Note:** Downloading an incorrect operating system will cause unknown errors. It is recommended that you allow MPLAB IDE to automatically download the OS.

Click **Next** to continue.

5.5.6 Setup Wizard – External Memory

> **Note:** This page will only appear if you have selected a ROMless device (PIC18C601/801.)

Step 5: Specify amount of external memory – Specify the amount of external memory you will use for your ROMless device.

The Start Address is always zero (0). Enter an End Address value in decimal or hex (use 0x before number.)

Click **Next** to continue.

5.5.7 Setup Wizard – Summary

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

Click **Finish** when you are satisfied with the MPLAB ICD 2 setup.

5.6 ADVANCED BREAKPOINTS DIALOG

In addition to the basic breakpoints that you may set (right-mouse menu and standard Breakpoint dialog), MPLAB ICD 2 has an Advanced Breakpoints dialog (*Debugger>Advanced Breakpoints*.)

The options on the Advanced Breakpoint Dialog are dependent on your selected device.

- PIC18F Devices – Regular PIC18F MCU devices.
- Extended PIC18F Devices V1 – Version 1 extended PIC18F devices have an extended instruction set (available for MPLAB C18 C compiler use) that may be enabled through a Configuration bit (XINST).
- Extended PIC18F Devices V2 – Version 2 extended PIC18F devices have the version 1 extended instruction set, as well as additional Breakpoint capabilities.
- dsPIC30F Devices – dsPIC30F DSC devices.
- dsPIC33F/PIC24 Devices – dsPIC33F DSC and PIC24 MCU devices.

5.6.1 PIC18F Devices

The Advanced Breakpoints Dialog for PIC18F devices contains the following options.

Breakpoint is in Program Memory

Check this checkbox if the breakpoint will be located in program memory. Then, enter the "Program Memory Address" as the location of the breakpoint.

Breakpoint is in File Registers

Check this checkbox if the breakpoint will be located in data memory (file registers). Then:

- Read Access/Write Access – Specify file register access.
- File Register Address – Enter location of breakpoint.
- File Register Value – If “File Register Must be Equal to Following Value”, check the checkbox and then enter “File Register Value”.

Additional Breakpoint Setup

In addition to breakpoint memory setup, you may select the following:

- Break on Stack Over/Underflow – Break on stack overflow or underflow.
- Pass Count – For the breakpoint set above, break for the entered pass count value (0-255).

5.6.2 Extended PIC18F Devices V1

The Advanced Breakpoints Dialog for Extended PIC18F devices (version 1) contains the following options.

Breakpoint Number

These devices support up to three breakpoints. Enter a number and then set up that breakpoint below. Change the number to set up another breakpoint.

Breakpoint is in Program Memory

Check this checkbox if the breakpoint will be located in program memory. Then, enter the “Program Memory Address” as the location of the breakpoint.

Breakpoint is in File Registers

Check this checkbox if the breakpoint will be located in data memory (file registers). Then:

- Read Access/Write Access – Specify file register access.
- File Register Address – Enter location of breakpoint.
- File Register Value – If "File Register Must be Equal to Following Value", check the checkbox and then enter "File Register Value".

Additional Breakpoint Setup

In addition to breakpoint memory setup, you may select the following:

- Break on Stack Over/Underflow – Break on stack overflow or underflow.
- Pass Count – For the breakpoint set above, break for the entered pass count value (0-255).

5.6.3 Extended PIC18F Devices V2

The Advanced Breakpoints Dialog for Extended PIC18F devices (version 2) contains the following options.

5.6.3.1 BREAKPOINT PARAMETERS

These options apply to the breakpoint specified in "Breakpoint #".

Breakpoint Number

These devices support up to three breakpoints. Enter a number and then set up that breakpoint below. Change the number to set up another breakpoint.

Breakpoint is in Program Memory

Check this checkbox if the breakpoint will be located in program memory. Then, enter the "Program Memory Address" as the location of the breakpoint.

Breakpoint is in File Registers

Check this checkbox if the breakpoint will be located in data memory (file registers). Then:

- Read Access/Write Access – Specify file register access.
- File Register Address – Enter location of breakpoint.
- File Register Value – If “File Register Must be Equal to Following Value”, check the checkbox and then enter “File Register Value”.

Data Sample

Check this checkbox if the breakpoint will signify the start of data sampling. Then:

- File Register Address – Enter location of breakpoint.

Data samples will appear in the Output window under the **ICD2 Data Sample** tab. To have this information scroll, select “Scroll Data Sample Output” under **Section 5.6.3.2 “Emulator Features”**.

Additional Breakpoint Setup

In addition to the above breakpoint setup, you may select the following at the bottom of the dialog:

- Pass Count – For the breakpoint set above, break for the entered pass count value (0-255).

5.6.3.2 EMULATOR FEATURES

The following features apply independent of the breakpoint setup above.

Event Breakpoints

For all breakpoints:

- Break on Stack Over/Underflow – Break on stack overflow or underflow. (Future feature)
- Break on Watchdog Timer – Break on a WDT time-out. (Future feature)
- Break on `SLEEP` – Break on a `SLEEP` instruction. The `SLEEP` instruction will NOT be executed. Running or stepping from this point will execute the instruction after `SLEEP`.

Enable Stopwatch

Check to coordinate breakpoints with Stopwatch operation.

- Halt on Start Condition (Breakpoint 2)
  - If checked, halt program (breakpoint 2 enabled) and program/Stopwatch go on Run.
  - If unchecked, program continues to run (breakpoint 2 disabled) and Stopwatch is started.
- Halt on Stop Condition (Breakpoint 3)
  - If checked, halt program (breakpoint 3 enabled) and Stopwatch.
  - If unchecked, program continues to run (breakpoint 3 disabled), but Stopwatch halted.
- Reset Stopwatch on Run – Any time you run the program, reset the stopwatch to zero.

Scroll Data Sample Output

Check to scroll data sample information in the Output window, if selected under **Section 5.6.3.1 "Breakpoint Parameters"**.

5.6.4 dsPIC30F Devices

The Advanced Breakpoints Dialog for dsPIC30F DSC devices contains the following options.

Breakpoint Combinations

This section is grayed out unless there are two breakpoints. The options are:

- Both breakpoints must occur at the same time to cause a break.
- Breakpoint 0/1 does not break until after breakpoint 1/0 occurs – Click **Swap BPs** to change the order.

Breakpoints

Breakpoint #	Select a breakpoint to set up, either 0 or 1.
Breakpoint Type	Select the type of breakpoint. Options below will change depending on the selection made here. **Disabled** – Breakpoint disabled (default). Other options grayed out. **Other Selections** – See tables below.

TABLE 5-4: PROGRAM MEMORY

Breakpoint Type	Program Memory Execution TBLWT Program Memory TBLRD Program Memory PSV Read
Program Memory Address	Enter the Program Memory Address of the breakpoint.
Pass Count Type	Enter the pass count type, if desired. Pass counting disabled – default. Event must occur *pass count* times. Break occurs *pass count* instructions after event.
Pass Count	Enter a pass count value, a number between 0 and 255.

TABLE 5-5: DATA MEMORY – BYTE OR WORD

Breakpoint Type	X Bus Write Specific Byte X Bus Read Specific Byte Y Bus Read Specific Word X Bus Write Specific Word X Bus Read Specific Word
Data Memory Address	Enter the Data Memory Address of the breakpoint.
Data Value	Specify a value that the Byte or Word must be equal to before breaking.
Pass Count Type	Enter the pass count type, if desired. Pass counting disabled – default. Event must occur *pass count* times. Break occurs *pass count* instructions after event.
Pass Count	Enter a pass count value, a number between 0 and 255.

TABLE 5-6: DATA MEMORY

Breakpoint Type	Y Bus Read X Bus Write X Bus Read
Data Memory Address	Enter the Data Memory Address of the breakpoint.
Pass Count Type	Enter the pass count type, if desired. Pass counting disabled – default. Event must occur *pass count* times. Break occurs *pass count* instructions after event.
Pass Count	Enter a pass count value, a number between 0 and 255.

5.6.5 dsPIC33F/PIC24 Devices

The Advanced Breakpoints Dialog for dsPIC33F DSC and PIC24 MCU devices contains the following options.

- Breakpoints Tab
- Breakpoint Combinations Tab
- Emulator Features Tab

5.6.5.1 BREAKPOINTS TAB

Breakpoints

Breakpoint #	Select a breakpoint to set up, either 0 or 1.
Breakpoint Type	Select the type of breakpoint. Options below will change depending on the selection made here. **Disabled** – Breakpoint disabled (default). Other options grayed out. **Other Selections** – See tables below.

TABLE 5-7: PROGRAM MEMORY

Breakpoint Type	Program Memory Execution TBLWT Program Memory TBLRD Program Memory PSV Read
Program Memory Address	Enter the Program Memory Address of the breakpoint.
Pass Count Type	Enter the pass count type, if desired. Pass counting disabled – default. Event must occur *pass count* times. Break occurs *pass count* instructions after event.
Pass Count	Enter a pass count value, a number between 0 and 255.

TABLE 5-8: DATA MEMORY – BYTE OR WORD

Breakpoint Type	X Bus Write Specific Byte X Bus Read Specific Byte Y Bus Read Specific Word X Bus Write Specific Word X Bus Read Specific Word
Data Memory Address	Enter the Data Memory Address of the breakpoint.
Data Value	Specify a value that the Byte or Word must be equal to before breaking.
Pass Count Type	Enter the pass count type, if desired. Pass counting disabled – default. Event must occur *pass count* times. Break occurs *pass count* instructions after event.
Pass Count	Enter a pass count value, a number between 0 and 255.

TABLE 5-9: DATA MEMORY

Breakpoint Type	Y Bus Read X Bus Write X Bus Read
Data Memory Address	Enter the Data Memory Address of the breakpoint.
Pass Count Type	Enter the pass count type, if desired. Pass counting disabled – default. Event must occur *pass count* times. Break occurs *pass count* instructions after event.
Pass Count	Enter a pass count value, a number between 0 and 255.
Enable Data Sample	Check to enable data sampling. View data in Output window, **ICD2 Data Sample** tab.

5.6.5.2 BREAKPOINT COMBINATIONS TAB

Breakpoint(s) must previously have been set up in **Section 5.6.5.1 "Breakpoints Tab"** for options to be available here.

ANDED Breakpoints

Check the checkbox next to each breakpoint to AND together.

Breakpoint Sequencing

Select a breakpoint sequence.

Swap Breakpoints

Choose one breakpoint to swap with another.

5.6.5.3 EMULATOR FEATURES TAB

The following features apply independent of the breakpoint setup above.

Event Breakpoints

For all breakpoints:

- Break on Watchdog Timer – Break on a WDT time-out.
- Break on `SLEEP` – Break on a `SLEEP` instruction. The `SLEEP` instruction will NOT be executed. Running or stepping from this point will execute the instruction after `SLEEP`.

Enable Stopwatch

Check to coordinate breakpoints with Stopwatch operation.

- Halt on Start Condition (Breakpoint 1)
  - If checked, halt program (breakpoint 1 enabled) and program/Stopwatch go on Run.
  - If unchecked, program continues to run (breakpoint 1 disabled) and Stopwatch is started.
- Halt on Stop Condition (Breakpoint 0)

  - If checked, halt program (breakpoint 0 enabled) and Stopwatch.
  - If unchecked, program continues to run (breakpoint 0 disabled), but Stopwatch halted.
- Reset Stopwatch on Run – Any time you run the program, reset the stopwatch to zero.

Data Sample Scroll Output

Check to scroll data sample information in the Output window, if selected under **Section 5.6.5.1 "Breakpoints Tab"**.

MPLAB ICD 2 USER'S GUIDE

Part 2 – Troubleshooting

NOTES:

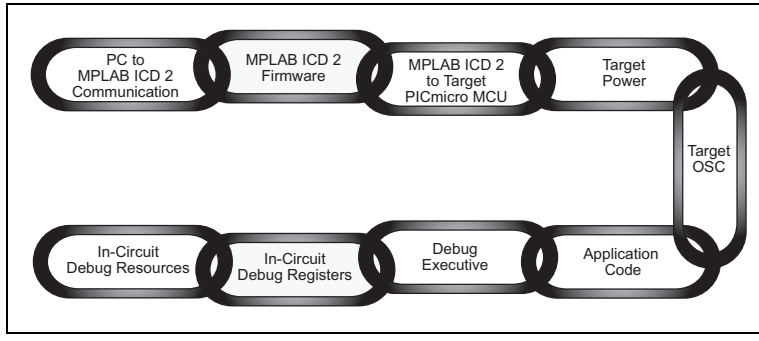
Chapter 6. Troubleshooting Tips

6.1 INTRODUCTION

Since many things in MPLAB IDE, MPLAB ICD 2 and the PIC MCU target application circuit can affect the operation of MPLAB ICD 2, it is important to understand the operation of each link in the chain of ICD functionality.

Figure 6-1 is a diagram showing the various links. Each link must be functional for MPLAB ICD 2 to program and debug. This section discusses these links to help find and fix problems if they occur.

FIGURE 6-1: CHAIN OF MPLAB® ICD 2 FUNCTIONALITY



Functional links are covered as follows:

- Link: PC to MPLAB ICD 2 Communications
- Link: MPLAB ICD 2 Firmware
- Link: MPLAB ICD 2 to Target PIC MCU Device
- Link: Target Power
- Link: Target Oscillator
- Link: Application Code
- Link: Debug Executive
- Link: In-Circuit Debug Registers
- Link: In-Circuit Debug Resources

6.2 LINK: PC TO MPLAB ICD 2 COMMUNICATIONS

This link is the foundation for all subsequent links. If it doesn't work, MPLAB IDE will issue communications errors. Make sure that the MPLAB USB drivers are installed according to the instructions. Look at the Windows hardware manager dialog under the USB section to see that the MPLAB ICD 2 driver is properly installed on the PC. If using RS-232, make sure that the FIFO buffers are turned off and flow control is set for hardware in the COMM driver dialog. If the MPLAB ICD 2 USB driver is not visible in the USB section of the Hardware Manager, go to the ICD2/Drivers subdirectory of the MPLAB IDE installation directory. Open the `.htm` files with a web browser to view details on driver installation for the appropriate operating system. See **Section 2.3.1 "Setting Up Communications"** for more information.

6.3 LINK: MPLAB ICD 2 FIRMWARE

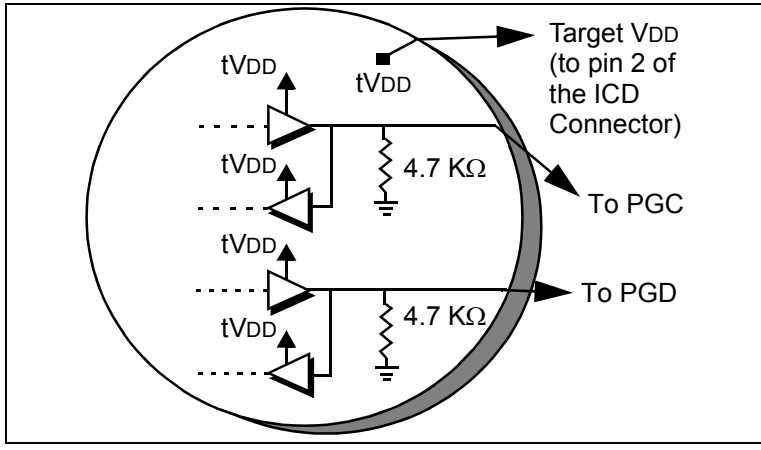
Be sure that the latest version of firmware is loaded in the MPLAB ICD 2. Updated firmware can be downloaded with the MPLAB IDE software. (See **Section 4.2 "Upgrading the MPLAB ICD 2 Firmware (Operating System)"** for automatically downloading firmware.) The latest production software should always be downloaded from the Microchip web site, www.microchip.com.

The version of firmware will determine which target PIC MCUs are supported. See the README file for the latest information on the firmware and on MPLAB ICD 2.

6.4 LINK: MPLAB ICD 2 TO TARGET PIC MCU DEVICE

Verify that all lines are connected and that no other signals or components are interfering with ICD signals on VPP, PGC and PGD. If MPLAB ICD 2's VDD and VSS lines are not connected to the target power and ground, it will not work. MPLAB ICD 2 can provide VDD to the target (<200mA, 5 volts only) and to these output buffers when providing power to the target circuit (selected by the *Debugger>Settings* dialog **Power** tab, "Power target circuit from MPLAB ICD 2"). Otherwise, the target's VDD will be used. VDD is sensed by MPLAB ICD 2 to check that voltage levels are correct. Powering from the target VDD allows level translation for target low-voltage operation. A simplified circuit for the internal buffer circuits of MPLAB ICD 2 is shown in Figure 6-2.

FIGURE 6-2: MPLAB® ICD 2 INTERNAL BUFFER CIRCUITS



If MPLAB ICD 2 does not have voltage on its VDD line (pin 2 of the ICD connector) it will not operate. Using a scope, communications on PGC and PGD with a full target VDD to VSS peak-to-peak waveform should be seen. VPP should show +12, +5, and 0V levels depending upon the operation being performed. Signals should be seen on those lines only after performing an MPLAB ICD 2 function from the MPLAB user interface. Otherwise, all these lines should be idle and noise free.

6.5 LINK: TARGET POWER

MPLAB ICD 2 can work within a range of about 2V to 5.5V on the target PIC MCU's VDD. The application must be powered by its own power supply. Remember, that the PGC and PGD I/O drivers in MPLAB ICD 2 are powered from the target VDD (see Figure 6-2).

6.6 LINK: TARGET OSCILLATOR

When programming a device, it will program without a running oscillator, but the debug function will not work unless the target oscillator is functioning. Unlike an in-circuit emulator, while debugging, MPLAB ICD 2 requires that the target PIC MCU is running.

Testing may be necessary to make sure that the target system is executing instructions properly. For instance, a small program could be written to flash an LED and then programmed into the target PIC MCU with MPLAB ICD 2 debug disabled. Disconnect the ICD and reset the target. If the LED does not flash, check the circuitry to find out why it's not operating. If MPLAB ICD 2 is connected after programming the target with debug disabled, MPLAB can still toggle the Reset. If it runs stand-alone but not with ICD connected (after selecting *Debugger>Run* in the MPLAB IDE), then the interconnections may be suspicious. Use a scope to look at PGC, PGD and V_{PP}. Make sure that the oscillator starts up quickly. If it takes too long to get the target oscillator started, then MPLAB ICD 2 can time out and give errors.

Make sure that the correct Oscillator mode is selected (look at *Configure>Configuration Bits*). The Oscillator mode will depend upon the oscillator selected. Try HS mode for a crystal, and RC mode for an external resistor/capacitor oscillator. These settings may need to be changed for the final design, but they are some of the easiest modes to start with. Oscillator operation can be checked on OSC2 after Configuration bits are set at programming time and power is applied in debug or normal mode.

6.7 LINK: APPLICATION CODE

Verify that MPLAB ICD 2 is programming properly by either running an LED blink test similar to that described in Link: Target Oscillator or by programming code into the target device. MPLAB ICD 2 will execute a verify function after programming to confirm that the data in the target PIC MCU matches MPLAB ICD 2's program memory. *Programmer>Verify* can also be selected manually to compare the target PIC MCU's memory with the MPLAB IDE image. If the target PIC MCU has AVDD and AVSS, ensure that they are connected properly. These lines should be connected to power (VDD) and ground (VSS) respectively. Consult the device data sheet for details. MPLAB ICD 2 will not operate if any of these are available and not connected. Make sure that Low Voltage Programming is disabled (*Configure>Configuration Bits*).

6.8 LINK: DEBUG EXECUTIVE

When using the *Debugger>Program* function from MPLAB IDE, the debug executive will be downloaded and the in-circuit debug registers in the target PIC MCU will be enabled. Programming should always be done from the Debug menu, not the Programmer menu when debugging. When programming from the Programmer menu (after turning MPLAB ICD 2 off as a debugger), the debug executive will not be downloaded and the in-circuit debug registers will be disabled.

By looking at the *Configure>Configuration Bits* menu selection, it can be determined whether MPLAB is going to download the debug executive. If the bits labeled "Background Debug" are enabled, then the debug executive will be downloaded when the device is programmed.

6.9 LINK: IN-CIRCUIT DEBUG REGISTERS

Make sure that debug is enabled on the MPLAB ICD 2 dialog in MPLAB IDE before programming a device. By looking at the *Configure> Configuration Bits* dialog, it can be verified that the

in-circuit debug registers are enabled. The line labeled “Background Debug” should say “Enabled.” Programming should be done from the Debug menu, not the Programmer Menu.

6.10 LINK: IN-CIRCUIT DEBUG RESOURCES

Look at the specific registers used for a device in the on-line help. The file registers or program memory areas reserved by MPLAB ICD 2 can not be used. If using the `xxxxxi.lkr` linker script (name ends in “`i`”), then, unless the script has been modified, these resources will be marked reserved and will not be available to the application. If the linker is not being used, use `CBLOCK` or `EQU`s for variable storage in the code, make sure that registers required by MPLAB ICD 2 are not used. The in-circuit debugger will not work if “code protect” or “table read protect” is enabled, if the Watchdog Timer is running or if the oscillator is not set to the correct mode by the Configuration bits. If using fast interrupts or the `CALL FAST` instruction, MPLAB ICD 2 uses the shadow stack and users will not be able to exit properly from the fast interrupt routine or `CALL FAST` function.

NOTES:

Chapter 7. Self-Test

7.1 INTRODUCTION

The **Run Self Test** button (Settings dialog, **Status** tab) is helpful in determining problems with the MPLAB ICD 2 module or target connection.

Self-Test features covered are:

- Target V_{DD}
- Module V_{PP}
- $\overline{MCLR}$ = GND
- $\overline{MCLR}$ = V_{DD}
- $\overline{MCLR}$ = V_{PP}
- Failed Self-Test Error – V_{PP}/V_{DD} High/Low

7.2 TARGET V_{DD}

Tests the V_{DD} provided from the MPLAB ICD 2 (5 volts only) if "Power target circuit from MPLAB ICD 2" is selected.

Pass/Fail codes:

0/0 = Pass	V_{DD} is within specified limits
01 = Min error	V_{DD} is below specified limits
80 = Max error	V_{DD} is above specified limits

An error can indicate:

- Target voltage/current mismatch from provided voltage/current
- MPLAB ICD 2 pod hardware problems

7.3 MODULE VPP

Tests the programming voltage (VPP) provided from the MPLAB ICD 2 to the target VPP/MCLR pin during a programming cycle

Pass/Fail codes:

00 = Pass	VPP is within specified limits
01 = Min error	VPP is below specified limits
80 = Max error	VPP is above specified limits

An error can indicate:

- The target VPP/MCLR pin is not correctly wired

7.4 MCLR = GND

Test the ability of the MPLAB ICD 2 to provide a ground level to the target VPP/MCLR pin for target reset

Pass/Fail codes:

00 = Pass	Ground can be provided to target VPP/MCLR pin
80 = Max error	Ground level too high for target VPP/MCLR pin

An error can indicate:

- The target VPP/MCLR pin is not correctly wired

7.5 $\overline{\text{MCLR}}$ = VDD

Test the ability of the MPLAB ICD 2 to provide VDD to the target VPP/$\overline{\text{MCLR}}$ pin during normal operation (such as Run).

Pass/Fail codes:

00 = Pass	VDD can be provided to target VPP/$\overline{\text{MCLR}}$ pin
01 = Min error	VDD too low for target VPP/$\overline{\text{MCLR}}$ pin
80 = Max error	VDD too high for target VPP/$\overline{\text{MCLR}}$ pin

An error can indicate:

- Target voltage/current mismatch from provided voltage/current
- MPLAB ICD 2 pod hardware problems

7.6 $\overline{\text{MCLR}}$ = VPP

Test the ability of the MPLAB ICD 2 to provide VPP to the target VPP/$\overline{\text{MCLR}}$ pin during programming operations.

Pass/Fail codes:

00 = Pass	VPP can be provided to target VPP/$\overline{\text{MCLR}}$ pin
01 = Min error	VPP too low for target VPP/$\overline{\text{MCLR}}$ pin
80 = Max error	VPP too high for target VPP/$\overline{\text{MCLR}}$ pin

An error can indicate:

- The target VPP/$\overline{\text{MCLR}}$ pin is not correctly wired

7.7 FAILED SELF-TEST ERROR – VPP/VDD HIGH/LOW

Symptom:

When connecting to the MPLAB ICD2 in MPLAB IDE or running a self-test (*Debugger/Programmer>Settings>Status>Self Test*), MPLAB IDE may report a self-test error in the Output window.

Cause:

This could be due to the firmware routines used to determine minimum and maximum voltages (a less than compare is used in one place while a less-than or equal-to is used in another).

Solution:

Check the actual voltages measured by the ICD. This can be found at *Programmer/Debugger>Settings>Power*. The **Update** button will cause these voltages to be remeasured and displayed. As long as the voltages displayed are within tolerance for the target device, the MPLAB ICD 2 unit can still be used for programming and debugging operations.

The firmware will be changed for better support of self-test functions.

Chapter 8. General Troubleshooting

8.1 INTRODUCTION

Frequently Asked Questions (FAQs) and common problems are documented here to help you as you use the MPLAB ICD 2. Also, messages generated by the ICD and its tool limitations are referenced.

Topics covered in this chapter are:

- Frequently Asked Questions
- Common Communication Problems
- Common Problems
- Error and Warning Messages
- Limitations

8.2 FREQUENTLY ASKED QUESTIONS

Questions are listed in the order of most-to-least frequently asked.

8.2.1 *Why does my system fail to program or verify?*

Check the PGC, PGD and VPP connections and voltages as described in **Chapter 2.'Getting Started"**. Make sure the target PIC MCU has power applied. If the target PIC MCU has AVSS and AVDD pins, verify that these are all connected properly.

8.2.2 *Why do I need the ICD header adapters*?

The low pin count parts that are supported by MPLAB ICD 2 could not be used very effectively if in-circuit debug pins are reserved – imagine losing three out of six I/O lines on an eight-pin part! For this reason, special bond-out PIC MCUs have been manufactured

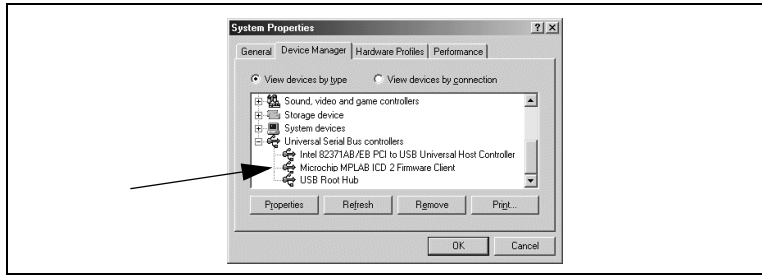
that can emulate these low pin count parts with an adapter, and which will allow the use of all pins in the target application. The bond-out PIC MCU has the in-circuit communications pins to interface to MPLAB ICD 2.

The advantage of this is that the low pin parts can be used in development with MPLAB ICD 2. The disadvantage is that these parts cannot simply have an MPLAB ICD 2 connector on the target application for in-circuit debugging. These bond-out PIC MCUs are similar to emulator chips. They can support more than one device. Jumpers are available on the header boards to configure the chip to match the device under development. These low pin count devices can be programmed with MPLAB ICD 2 using the Universal Programmer Adapter or by putting an MPLAB ICD 2 connector on the target application to connect VPP, PGC and PGD on these parts.

8.2.3 *I can't connect to MPLAB ICD 2. What do I do now?*

Is the MPLAB ICD 2 power light on? The LED should shine brightly. If it is dim, only USB may be connected and the user may need to connect the power supply. The RS-232 connection from the PC cannot supply power to MPLAB ICD 2, but a USB connection can. Note that some USB hubs cannot supply power. Check the on-line help for current troubleshooting hints. Are the USB drivers installed properly? The MPLAB ICD 2 USB driver should be visible in the Windows Device Manager dialog (see Figure 8-1). Some USB hubs cannot supply power to the attached USB devices. A power supply will need to be connected to MPLAB ICD 2 with these hubs.

FIGURE 8-1: SYSTEM PROPERTIES DIALOG



8.2.4 *MPLAB ICD 2 responds with "Target not in debug mode error." What does this mean?*

Usually this means that MPLAB ICD 2 cannot communicate with the debug executive. The debug executive can only be downloaded by programming the target PIC MCU with the user's application from the *Debugger>Program* menu selection. There are other reasons that the debug executive might not be able to communicate, such as target clock or power supply problems. Check the Configuration bits to see that the "Background Debug" is enabled. Look at *Configure>Configuration Bits* to make sure that the Watchdog TImer is disabled, code protection is turned off and that the oscillator setting is correct.

8.2.5 *Can MPLAB ICD 2 operate with a target device running at low voltage?*

Yes. As long as the target PIC MCU supports low-voltage operation, it can run down to about 2 volts V_{DD}. There are level converters in the input/output buffers of the MPLAB ICD 2. These are powered from the V_{DD} of the target device. Also, MPLAB ICD 2 will sense the operating voltage of the target and correctly adjust its functionality to deal with this operation, i.e., using the correct Flash erase algorithms. V_{DD} needs to be selected "From Target" in the MPLAB ICD 2 "Settings" dialog, and have a power supply on the target for low-voltage operation.

8.2.6 *Does MPLAB ICD 2 support Low Voltage Programming (LVP)?*

No. But this does not mean that it will not work correctly when running at low-voltage V_{DD} on the target. It just means that the programming voltages applied to V_{PP} will always be +12V.

8.2.7 *Why do I have problems when configuring the PLL Oscillator? MPLAB ICD 2 gets "hung up."*

This is a requirement of the actual PIC MCU. After programming the Configuration bits for the PLL Oscillator, power needs to be removed then reapplied to the target. If this is not done, the target PIC MCU will not have a clock. With no clock, Debug mode will not function. Also, if power is not removed and reconnected when switching to PLL mode, the device may run, but without the PLL multiplier.

8.2.8 *When I try to install, why are the drivers not found, even though I can see them in the driver folders, and I point the driver wizard to the correct folder?*

This problem can be solved by exiting from the driver install wizard, and going to the Control Panel "Add New Hardware" selection. After the system searches for new hardware, choose, "No, the device isn't in the list." Then choose "No, I want to select the hardware from a list." Then choose "Universal Serial Bus Controller," and when the **Have Disk...** button appears, go to the driver folder (MPLAB IDE\ICD2\Drivers) and select the proper driver.

8.2.9 *Can I use code protection with MPLAB ICD 2?*

No. Code protection, especially table read protection on any area in program memory will prevent MPLAB ICD 2 from functioning. Do not use any code protection or table read protection configura-

tion settings when debugging with MPLAB ICD 2. Code protection can be enabled when programming a part for testing without MPLAB ICD 2.

8.2.10 *How does MPLAB ICD 2 deal with Calibration Data?*

It is automatically handled. Any values that are in program memory that are used by the PIC MCU for calibration data will be read and preserved by MPLAB ICD 2 when erasing, programming and debugging. No action is required to protect this data.

8.2.11 *Why am I getting erratic values from my EEData area?*

MPLAB ICD 2 can read the EEData areas directly, without going through the EECON register's required TABLRD sequence of instructions. The buffers that MPLAB uses can sometimes interfere with the user's code. Avoid going back and forth from program reads of data to MPLAB ICD 2 reads of the EEData area when single stepping code.

8.2.12 *Why does my program keep resetting?*

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

8.2.13 *Why is "Erase All Before Programming" grayed out?*

In some of the newer Flash parts, the programming algorithm requires that non-contiguous areas of program memory are programmed in banks. For these parts, all memory must be erased before programming.

8.2.14 *Can my program read and write from Port B or GPIO without interfering with MPLAB ICD 2?*

Yes. When the in-circuit debug facilities are enabled, PGC and PGD are always used by MPLAB ICD 2, and user code that reads or writes from PORT B will not interfere. Note that the values read from PGC and PGD are not necessarily valid, and writes to these two pins will be ignored. In addition, if interrupts are enabled for changes on Port B, signals on PGC and PGD will not cause an interrupt.

8.2.15 *Why are the timers behaving erratically when single-stepping?*

This is one of the drawbacks of using the in-circuit debugger. Since code is actually running in the debug executive, timers can continue to run during the operation of the debug executive, even when the user's application program is "halted."

8.2.16 *Why am I getting warnings and errors when using the PIC12F629/675 or PIC16F630/676?*

The GP1/RA1 pins on these parts cannot be pulled high while using MPLAB ICD 2. Refer to the header board specification for more information on using MPLAB ICD 2 for these parts (DS51292).

8.2.17 *What would make the power and busy LED blink on and off?*

This may indicate that the target MPLAB ICD 2 connector is wired backwards (opposite from the wiring diagram). The blinking indicates that the MPLAB ICD 2 is shutting down due to high current. The following tests indicate that the target is connected backwards:

- Look for blinking Power and Busy LEDs (power may go out altogether).
- Execute a "Self-Test" and look for a min. error on

"$\overline{\text{MCLR}}$=VPP" (all other tests will probably pass).

- Use the default address range for the target controller and execute a "Program" cycle. A voltage level on pin 1 (VPP) of 7-8 volts is too low.

If the target was wired backwards, protection circuitry in the MPLAB ICD 2 will prevent damage to the module. Normal operation should be seen when the target is wired correctly.

The Error LED flashing indicates an over-voltage condition on either the clock or data line on the ICSP interface. This circuitry is in place to protect the buffers on the MPLAB ICD 2 from being damaged. When a high voltage is detected greater than 6.5V a clamping circuit brings the clock and data line low. Once this error occurs, MPLAB will need to be reinitialized and the fault removed for operation to resume normally.

8.2.18 *What does the MPLAB ICD 2 "Self-Test" do?*

The MPLAB ICD 2 Self-Test is helpful in determining problems with the MPLAB ICD 2 module or target connection. For more details, see **Chapter 7.'Self-Test"**.

8.2.19 *Why are the W, STATUS and BSR registers getting changed when using high priority interrupts with the* `RETFIE` *instruction?*

The shadow registers, which are used for high priority interrupts and `CALL FAST` are used by MPLAB ICD 2. These are reserved resources for MPLAB ICD 2 operation. A problem will be encountered if a breakpoint is set inside a `CALL FAST` subroutine or inside the service routine for a high priority interrupt that uses the shadow registers with the `RETURN FAST` or `RETFIE` instruction.

8.2.20 *When setting a breakpoint at the first location in my program, why does it stop at address 0001 rather than 0000?*

MPLAB ICD 2 stops at the instruction *after* the breakpoint. This means that a breakpoint at address 0000 will be executed, then the program counter will be pointing at address 0001 when it gets a breakpoint. If a user needs to stop at the first instruction in their code, they must put a `NOP` at address 0000.

8.2.21 *Why is my calibration memory displayed as the erase value?*

MPLAB IDE is displaying the default memory value. A device read must be performed with MPLAB ICD 2 to display the actual value on the device.

8.2.22 *When single stepping through the code, my timer times out, but why does my timer interrupt routine not execute?*

When single stepping, the in-circuit debugger will not allow the PIC MCU to respond to interrupts. If it did, and users had external interrupts, then single stepping would almost always end up in interrupt routines. To debug an interrupt, set a breakpoint *inside* the Interrupt Service Routine and Run to get a breakpoint after the interrupt.

8.2.23 *How do I program a ROMless (PIC18C601/801) device?*

For these devices, you must create the routines that read/write memory. See **Section 4.5 "ROMless Device Considerations"** for details.

8.2.24 *Why can't I step through a return for my dsPIC30F device?*

Make sure that you have:

- checked the option to build for ICD, i.e., go to *Project>Build Options>Project*, **MPLAB LINK30** tab, and check "Link for ICD2".
- initialized the software stack, i.e., use the startup module `libpic30.a` (for MPLAB C30) or `crt0.o` (for MPLAB ASM30) or consult a template to write your own SP and SPLIM initialization code.

Then you should able to step through a return (from a function, etc.)

8.3 COMMON COMMUNICATION PROBLEMS

The following are a list of common problems with MPLAB ICD 2 communications, both USB and serial.

8.3.1 Communications cannot be established with MPLAB ICD 2

First, click *Debugger>Connect*. If you cannot establish communications with MPLAB ICD 2, follow the steps below. If communications still cannot be established, contact Microchip Engineering Support.

8.3.1.1 IN GENERAL

1. Check that the RS-232 or USB cable is connected securely to the MPLAB ICD 2 and the host computer.
2. Check that the modular cable is connected securely to the demo board/target application and the MPLAB ICD 2 module.
3. Make sure there is power to the demo board/target application. Check V_{DD} Source in the MPLAB ICD 2 Settings dialog, **Power** tab.

4. Check that the device is plugged into the demo board/target application correctly; e.g., all pins are plugged into the socket and the device is correctly oriented.

8.3.1.2 SERIAL COMMUNICATIONS

1. Make sure the power supply is connected and the Power LED on the MPLAB ICD 2 module is on.
2. There may be a driver-hardware incompatibility; try changing **Flow Control** to **Hardware** and/or turning off the **FIFO** for the serial port.

   For specific instructions, see **Section 8.3.1.3 "Changing Serial Port Settings"**.
3. Check the settings on the MPLAB ICD 2 Setting dialog, **Communication** tab. Make sure you selected the correct COM port and baud rate for your application.

> **Note:** If you are having problems at a COM port baud rate speed of **57600**, switch back to the initial speed of **19200**.

4. Try connecting the MPLAB ICD 2 module to a different port. When first connecting to the MPLAB ICD 2, the default COM port is COM1. If you are using the MPLAB ICD 2 on another COM port, select *Debugger>Settings* and then click the **Communications** tab. Set your COM port and baud rate here.
5. 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 only do so if you are familiar with those procedures.

6. If you have a COM port but 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 2]` 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.

8.3.1.3 CHANGING SERIAL PORT SETTINGS

Complete the following steps to change the Flow Control and FIFO settings for a serial communications port on a PC running the Windows operating system.

Windows 98/ME

1. On your PC, select *Start>Settings>Control Panel*.
2. In the Control Panel, double click the System Icon.
3. In the Systems Properties dialog, click the **Device Manager** tab.
4. If necessary, expand the Ports selection by clicking the “+” sign next to it.
5. Double click the I/O port to which the MPLAB ICD 2 is connected.
6. In the Flow Control field, select Hardware.
7. Click the **Advanced** button, deselect the “Use FIFO” box, and click OK.
8. Reboot the PC to implement the change.

Windows NT

> **Note:** You may need administrator privileges on your computer to change these settings.

1. On your PC, select *Start>Settings>Control Panel*.
2. In the Control Panel, double click the Ports Icon.
3. Double click the I/O port to which the MPLAB ICD 2 is connected.
4. Select the **Port Settings** tab.
5. In the Flow Control field, select Hardware.
6. Click the **Advanced** button, deselect the "Use FIFO" box, and click OK.
7. Reboot the PC to implement the change.

Windows 2000/XP

> **Note:** You may need administrator privileges on your computer to change these settings.

1. On your PC, select *Start>Settings>Control Panel*.
2. In the Control Panel, double click the System Icon.
3. In the Systems Properties dialog, click the **Hardware** tab and click the **Device Manager** button.
4. Double click "Ports (COM & LPT)" to expand the Ports selection.
5. Double click the I/O port to which the MPLAB ICD 2 is connected.
6. Select the **Port Settings** tab.
7. In the Flow Control field, select Hardware.
8. Click the **Advanced** button, deselect the "Use FIFO" box, and click OK.
9. Reboot the PC to implement the change.

8.3.1.4 USB COMMUNICATIONS

CAUTION

If the Windows OS picked a USB driver, MPLAB ICD 2 will not work. Follow the instructions in ICD2\Drivers\clnicd2.htm of the MPLAB IDE installation directory to remove the Windows drivers and replace them with the correct USB drivers.

1. If MPLAB ICD 2 is powering a target application, make sure a power supply is connected and the Power LED on the MPLAB ICD 2 module is on.
2. Make sure you have used the MPLAB IDE supplied USB driver for MPLAB ICD 2.

8.3.2 Connection failure occurs when using a COM port 2, 3 or 4

When first connecting to the MPLAB ICD 2, the default COM port is COM1. If you are using the MPLAB ICD 2 on another COM port, select *Debugger>Settings* and then click the **Communications** tab. Set your COM port and baud rate here.

8.3.3 MPLAB ICD 2 Connects But Target Operations Fail

This can indicate that the VDD setting is incorrect.

The MPLAB ICD 2 can provide VDD to the target (5 volts only). This is set in the MPLAB ICD 2 Debugger Settings dialog, **Power** tab. If “Power target circuit from MPLAB ICD 2” is selected, the MPLAB ICD 2 will provide VDD to the target (the target does not need to be powered). However, if the target needs a voltage other than 5V or more than 200mA of current, then the target cannot be powered by MPLAB ICD 2 and will require its own power source.

When you click the **Reset** button in Debug mode to reset the program, it will go to zero and halt. The Reset command will not rerun the program when the MPLAB ICD 2 is in Debug mode.

8.3.4 Command to program/erase target memory failed

Make sure that you have selected the appropriate Bus Width and Oscillator settings for the device you are programming.

8.3.5 Programming error occurs when programming a range of memory on PIC18C601/801

For the PIC18C601/801, you need to align the End Address + 1 so it can be divided by 8. For example, 0x7FF or 0x1FFF7. If you receive a programming error due to an incorrect End Address, you need to perform a reconnect, correct the End Address and select Program again.

8.4 COMMON PROBLEMS

The following are a list of common problems you may encounter.

8.4.1 Debug Mode Not Working

1. Make sure you have selected all of the areas of the device that you want to program. Select *Debugger>Settings* and click on the **Program** tab. Select Memories as desired.
2. If you are using the Start/End Address fields in the Memory Addresses section of the MPLAB ICD 2 Settings dialog (*Debugger>Settings*, **Program** tab), ensure that the Start Address is set to the beginning of an 8-byte block and the End Address is set to the end of an 8-byte block. For example, a Start Address of 0x10 and an End Address of 0x1F.

8.4.2 Can't uncheck programming of Configuration bits in Debug mode

For MPLAB ICD 2 to function in Debug mode, the ICD **must** ensure that the Background Debug Configuration bit is programmed on the target. Therefore, configuration memory is always programmed when in Debug mode.

8.4.3 When single stepping, the program runs too slowly

Upload of Data Issue

Data is uploaded from the MPLAB ICD 2 depending on the amount of data displayed in the IDE, i.e., the size and amount of open debug windows.

- If your program is running slow, consider closing some windows or using Watch windows to view specific registers.
- If not all register data is being updated, you probably do not have all the registers visible when you run/step. That is, if you step, then scroll to find a register, it will not have the updated data from the step execution.

8.4.4 Single Stepping Slow when Programmer Enabled

If you have a programmer enabled when using MPLAB ICD 2 to debug a part with EEData, your stepping through code will be very slow. Disable the programmer to speed up stepping time.

The slowdown occurs because the programmer must attach to the entire range of EEData so it can be programmed. The debugger, seeing that something is attached to the EEData (whether a programmer or EEData Display) will read back the entire range after each step in case the data was modified while running.

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

This is an upload of data issue. See **Section 8.4.3 "When single stepping, the program runs too slowly"**.

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

Your program may not have locked up, but may be running very slowly. This is an upload of data issue. See **Section 8.4.3 "When single stepping, the program runs too slowly"**.

Also, if you are using PIC16F87X device, the first program memory location (address 0x0000) must be a `NOP` instruction or you will not be able to single step past location zero.

8.4.7 The following I/O pins are not functioning correctly: RB6 or RB7

These pins are reserved for debugging.

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

Several GPR's and Program Memory locations are reserved for debugging.

8.4.9 EEPROM window does not reflect changes

In order to see the changes in the window, you must do a read of the memory.

8.4.10 Stack window does not reflect changes/is blank

This is a tool limitation. See **Chapter 11.'Limitations"**.

8.4.11 Program keeps resetting

See **Section 8.2.12 "Why does my program keep resetting?"**.

8.4.12 Execute Reset or $\overline{\text{MCLR}}$ command, program does not rerun

When you select *Debugger>Reset* to reset the program, it will go to zero and halt. The Reset command will not rerun the program when the MPLAB ICD 2 is in Debug mode.

8.4.13 File Register window does not display Top-of-Stack Registers

The three Top-of-Stack registers (TOSU, TOSH, and TOSL) are not displayed in the File Registers window when using the MPLAB ICD 2.

8.4.14 Power and busy LED blink on and off

See **Section 8.2.17 "What would make the power and busy LED blink on and off?"**.

8.5 ERROR AND WARNING MESSAGES

MPLAB ICD 2 generates several error and warning messages. Details on the meanings of most messages may be found in on-line help for MPLAB ICD 2.

Warning messages may be selected to display or not display using the **Warnings** tab of the Settings dialog.

All messages may be output to a file for debugging by selecting the "Output to Debug File" checkbox on the **Status** tab of the Settings dialog.

8.6 LIMITATIONS

MPLAB ICD 2 has limitations for both its debugger and programmer modes. These, and additional device-specific limitations, may be found in on-line help for MPLAB ICD 2.

NOTES:

Appendix A. Hardware Specifications

A.1 INTRODUCTION

Hardware specifications for MPLAB ICD 2 system components are listed here.

In addition, please refer to the *"MPLAB ICD 2 Design Advisory"* (DS51566) for hardware configuration issues.

Topics covered in this chapter are:

- MPLAB ICD 2 Module
- Modular Cable and Connector
- Power Supply

A.2 MPLAB ICD 2 MODULE

The MPLAB ICD 2 module contains all debugging, programming and control logic. It is connected to either a PC's serial port via a 9-pin serial cable or a USB port via a USB cable and to the PICDEM 2 Plus demo board or target application using a 6-wire modular cable.

The module contains the firmware to provide serial communications to the PC, to drive the MPLAB ICD 2 communications to the target application or demo board and to program a supported PIC MCU device using ICSP, all from the MPLAB IDE. The module can provide power to the demo board/target application. VDD source is selected in the MPLAB ICD 2 Settings dialog, **Power** tab. If the target application draws over 200 mA, an additional power adapter must be connected to the demo board or target application. The target application also provides power to the module only for the purpose of logic level conversion.

The MPLAB ICD 2 interface cable must be plugged into a modular connector on the application circuit with the appropriate connections to the PIC MCU device. The interface cable carries the signals necessary to allow in-circuit debugging of the target application

A.3 MODULAR CABLE AND CONNECTOR

A modular cable connects the MPLAB ICD 2 and the target application. The specifications for this cable and its connectors are listed below.

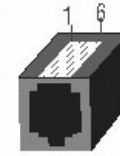
A.3.1 Modular Connector Specification

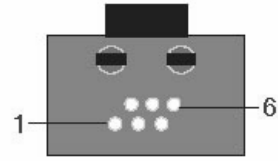
- Manufacturer, Part Number – AMP Incorporated, 555165-1
- Distributor, Part Number – Digikey, A9031ND

The following table shows how the modular connector pins on an application correspond to the microcontroller pins. This configuration provides the full MPLAB ICD 2 functionality.

FIGURE A-1: MODULAR CONNECTOR PINOUT OF DESIGNER'S BOARD

Modular Connector Pin	Microcontroller Pin
6	Not Used
5	RB6
4	RB7
3	Ground
2	Vdd
1	Vpp





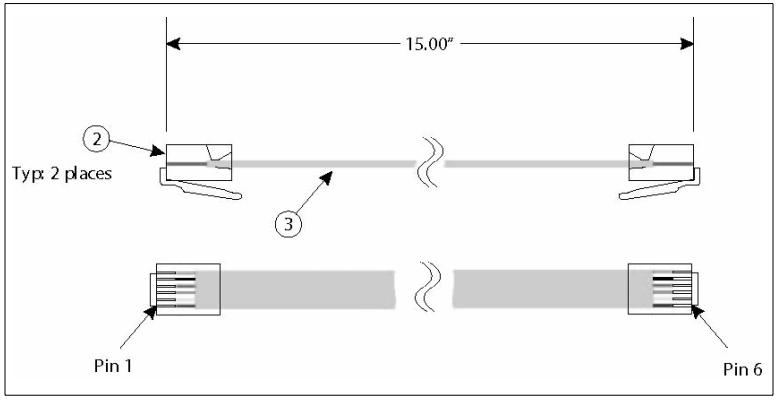
Bottom View of Modular Connector Pinout on Designer's Board

A.3.2 Modular Plug Specification

- Manufacturer, Part Number – AMP Incorporated, 5-554710-3
- Distributor, Part Number – Digikey, A9117ND

A.3.3 Modular Cable Specification

- Manufacturer, Part Number – Microchip Technology, 07-00024



A.4 POWER SUPPLY

The MPLAB ICD 2 module requires a 9.0 V, 750 mA power supply. The demo board or your target application may also require an additional power supply if it is drawing too much power from the MPLAB ICD 2 module. The MPLAB ICD 2 module can power a demo board/target application of up to 200 mA for 5 VDC.

A power adapter is included in the following product kits:

Part Number	Description	Includes these Components
DV164006	MPLAB® ICD 2 Evaluation Kit	MPLAB ICD 2 module, USB, MPLAB ICD 2 interface and RS-232 cables, PICDEM 2 Plus demo board, power adapter, MPLAB IDE CD-ROM
DV164007	MPLAB ICD 2 Module SW	MPLAB ICD 2 module, USB, MPLAB ICD 2 interface and RS-232 cables, power adapter, MPLAB IDE CD-ROM
AC162048	RS-232 Desk Top Kit	Power adapter and RS-232 cable

The power supply specifications for the MPLAB ICD 2 module and the PICDEM 2 Plus demo board are:

- **DC power supply:** 9 VDC @ 0.75A
- **With barrel connector:** ID = 2.5 mm, OD = 5.5 mm, barrel length = 10.0 mm, inside positive

NOTES:

Index

R

S

NOTES:

NOTES:

NOTES:

Worldwide Sales and Service

AMERICAS

Corporate Office
Tel: 480-792-7200

Atlanta
Tel: 678-957-9614

Boston
Tel: 774-760-0087

Chicago
Tel: 630-285-0071

Dallas
Tel: 972-818-7423

Detroit
Tel: 248-538-2250

Kokomo
Tel: 765-864-8360

Los Angeles
Tel: 949-462-9523

Santa Clara
Tel: 408-961-6444

Toronto
Tel: 905-673-0699

ASIA/PACIFIC

Hong Kong
Tel: 852-2401-1200

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8528-2100

China - Chengdu
Tel: 86-28-8665-5511

China - Fuzhou
Tel: 86-591-8750-3506

China - Hong Kong SAR
Tel: 852-2401-1200

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-5407-5533

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8203-2660

China - Shunde
Tel: 86-757-2839-5507

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7250

India - Bangalore
Tel: 91-80-4182-8400

India - New Delhi
Tel:91-11-4160-8631

India - Pune
Tel:91-20-2566-1512

Japan - Yokohama
Tel: 81-45-471- 6166

Korea - Gumi
Tel: 82-54-473-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Penang
Tel: 60-4-646-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-572-9526

Taiwan - Kaohsiung
Tel: 886-7-536-4818

Taiwan - Taipei
Tel: 886-2-2500-6610

Thailand - Bangkok
Tel: 66-2-694-1351

EUROPE

Austria - Wels
Tel: 43-7242-2244-39

Denmark - Copenhagen
Tel: 45-4450-2828

France - Paris
Tel: 33-1-69-53-63-20

Germany - Munich
Tel: 49-89-627-144-0

Italy - Milan
Tel: 39-0331-742611

Netherlands - Drunen
Tel: 31-416-690399

Spain - Madrid
Tel: 34-91-708-08-90

UK - Wokingham
Tel: 44-118-921-5869