PICkit™ 3 Programmer/Debugger User's Guide

© 2010 Microchip Technology Inc.

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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC^{32} logo, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL 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, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Octopus, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, REAL ICE, rfLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, 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.

© 2010, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.

ISBN: 978-1-60932-373-8

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

© 2010 Microchip Technology Inc.

Table of Contents

© 2010 Microchip Technology Inc.

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 online help. Select the Help menu, and then Topics to open a list of available online help files.

INTRODUCTION

This chapter contains general information that will be useful to know before using the PICkit™ 3 programmer/debugger. Items discussed include:

- Document Layout
- Conventions Used in this Guide
- Warranty Registration
- Recommended Reading
- The Microchip Web Site
- Development Systems Customer Change Notification Service
- Customer Support
- Revision History

DOCUMENT LAYOUT

This document describes how to use the PICkit 3 as a development tool to emulate and debug firmware on a target board. The manual layout is as follows:

- **Chapter 1. Overview** – An overview of the PICkit 3 programmer/debugger.
- **Chapter 2. Theory of Operation** – A simplified description of how the PICkit 3 programmer/debugger works.
- **Chapter 3. Installation** – How to install the PICkit 3 programmer/debugger.
- **Chapter 4. General Setup** – Provides Instructions on how to get started using the PICkit 3 programmer/debugger to program supported devices.
- **Chapter 5. PICkit 3 Debug Express** – Provides basic information on using the PICkit™ 3 Debug Express.
- **Chapter 6. PICkit 3 Programmer-To-Go** – Provides instruction on using the PICkit 3 unit to program devices without being connected to a PC.
- **Chapter 7. Troubleshooting First Steps** – Begins the troubleshooting process by identifying first steps and common reasons for problems with debugging.
- **Chapter 8. Frequently Asked Questions (FAQs)** – Provides information on solving common problems.
- **Chapter 9. Error Messages** – Provides specific error messages and general corrective actions.
- **Chapter 10. Debugger Function Summary** – Summarizes the available debugging functions.
- **Appendix A. Hardware Specification** – Details hardware and electrical specifications for the PICkit 3.
- **Appendix B. PICkit 3 Schematics** – Provides hardware schematic diagrams for the PICkit 3 programmer/debugger.

© 2010 Microchip Technology Inc.

CONVENTIONS USED IN THIS GUIDE

The following conventions may appear in this documentation:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic	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 with right angle bracket	A menu path	*File>Save*
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain	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	A variable argument	*`file`*`.o`, where *`file`* can be any valid filename
Square brackets [ ]	Optional arguments	`mpasmwin [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.

RECOMMENDED READING

This user's guide describes how to use PICkit 3 . Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

In-Circuit Debugger Design Advisory (DS51764)

Please read this first! This document contains important information about operational issues that should be considered when using the PICkit 3 with your target design.

44-Pin Demo Board User's Guide (DS41296)

Consult this document for instructions on how to use the 44-Pin demo board as a development tool to emulate and debug firmware on a target board.

Low Pin Count Demo Board User's Guide (DS51556)

Consult this document for instructions on how to use Microchip Technology's low pin count device (8-pin, 14-pin and 20-pin). This document includes a series of tutorials.

MPLAB® IDE User's Guide/Help (DS51519)

Consult this document for more information pertaining to the installation and features of the MPLAB Integrated Development Environment (IDE) software. An online Help version is also available.

In-Circuit Serial Programmer™ (ICSP™) Guide (DS30277)

This document contains helpful design guidelines for successful ICSP programming. It includes application notes on hardware designs and the ICSP programming specifications.

MPASM™ Assembler, MPLINK™ Object Linker, MPLIB™ Object Librarian User's Guide (DS33014)

Describes how to use the Microchip PIC® MCU assembler (MPASM assembler), linker (MPLINK linker), and librarian (MPLIB librarian).

README for PICkit™ 3 Debug Express

For the latest information on using the PICkit 3 Debug Express, read the "`Readme for PICkit 3.htm`" file (an HTML file) in the Readmes subdirectory of the MPLAB IDE installation directory. The Readme file contains updated information and known issues that may not be included in this user's guide.

PICkit™ 3 Debug Express C18 Lessons

These tutorials guide you through using the PICkit 3 Debug Express with the MPLAB C Compiler for PIC18 MCU's. They are available on the MPLAB IDE CDROM and on the Microchip web site.

Readme Files

For the latest information on using other tools, read the tool-specific Readme files in the Readmes subdirectory of the MPLAB IDE installation directory. The Readme files contain updated information and known issues that may not be included in this user's guide.

© 2010 Microchip Technology Inc.

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, assemblers, linkers and other language tools. These include all MPLAB C compilers; all MPLAB assemblers (including MPASM™ assembler); all MPLAB linkers (including MPLINK™ object linker); and all MPLAB librarians (including MPLIB™ object librarian).
- **Emulators** – The latest information on Microchip in-circuit emulators.This includes the MPLAB REAL ICE™ and MPLAB ICE 2000 in-circuit emulators.
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debuggers. This includes MPLAB ICD 3 in-circuit debuggers and PICkit™ 2 and 3 debug express.
- **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 IDE Project Manager, MPLAB Editor and MPLAB SIM simulator, as well as general editing and debugging features.
- **Programmers** – The latest information on Microchip programmers. These include production programmers such as MPLAB REAL ICE in-circuit emulator, MPLAB ICD 3 in-circuit debugger, and MPLAB PM3 device programmers. Also included are nonproduction development programmers such as PICSTART® Plus and PICkit 1, 2 and 3.

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. See our web site for a complete, up-to-date listing of sales offices.

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

Documentation errors or comments may be e-mailed to docerrors@microchip.com.

REVISION HISTORY

Revision A (January 2009)

This is the initial release of this document.

Revision B (July 2010)

Section 1.5 Device and Feature Support has been removed. It is available through the PICkit 3 online help in MPLAB IDE.

Section 2.4.4 Debugger Powered – third paragraph was replaced.

Section 2.7 Programming – modified to include the three ways to program a device.

Added a new Chapter 6. PICkit 3 Programmer-To-Go.

Table 10.6 Program Memory Options – added new option for "Use high voltage on MCLR" to table.

Section 10.5.8 Settings Dialog, Power Tab – added explanatory text regarding the "Power target circuit from PICkit 3" option.

Section 10.5.10 Setting Dialog, Programmer-to-go Tab has been added.

Renamed Chapter 11 Hardware Specification to Appendix A Hardware Specification.

© 2010 Microchip Technology Inc.

Part 1 – Getting Started

NOTES:

© 2010 Microchip Technology Inc.

Chapter 1. Overview

1.1 INTRODUCTION

An overview of the PICkit 3 programmer/debugger system is given.

- PICkit 3 Programmer/Debugger Defined
- How the PICkit 3 Programmer/Debugger Helps You
- PICkit 3 Programmer/Debugger Components

1.2 PICkit 3 PROGRAMMER/DEBUGGER DEFINED

The PICkit 3 programmer/debugger (see Figure 1-1) is a simple, low-cost in-circuit debugger that is controlled by a PC running MPLAB IDE (v8.20 or greater) software on a Windows® platform. The PICkit 3 programmer/debugger is an integral part of the development engineer's toolsuite. The application usage can vary from software development to hardware integration.

The PICkit 3 programmer/debugger is a debugger system used for hardware and software development of Microchip PIC® microcontrollers (MCUs) and dsPIC® Digital Signal Controllers (DSCs) that are based on In-Circuit Serial Programming™ (ICSP™) and Enhanced In-Circuit Serial Programming 2-wire serial interfaces. In addition to debugger functions, the PICkit 3 programmer/debugger system also may be used as a development programmer. The PICkit 3 programmer/debugger is not intended to be used as a production programmer.

The debugger system executes code like an actual device because it uses a device with built-in emulation circuitry, instead of a special debugger chip, for emulation. All available features of a given device are accessible interactively, and can be set and modified by the MPLAB IDE interface.

The PICkit 3 debugger was developed for emulating embedded processors with debug facilities. The PICkit 3 features include:

- Full-speed USB support using Windows standard drivers
- Real-time execution
- Processors run at maximum speeds
- Built-in over-voltage/short circuit monitor
- Low voltage to 5V (1.8-5V range)
- Diagnostic LEDs (power, active, status)
- Read/write program and data memory of microcontroller
- Erase of all memory types (EEPROM, ID, configuration and program) with verification
- Peripheral freeze at breakpoint

FIGURE 1-1: PICkit™ 3 MCU PROGRAMMER/DEBUGGER

Legend:

1 – Lanyard Connection
2 – USB Port Connection
3 – Pin 1 Marker
4 – Programming Connector
5 – Status LEDs
6 – Push Button

1.2.1 Lanyard Connection

A convenient lanyard connection is available on the programmer.

1.2.2 USB Port Connection

The USB port connection is a USB mini-B connector. Connect the PICkit 3 to the PC using the supplied USB cable.

1.2.3 Pin 1 Marker

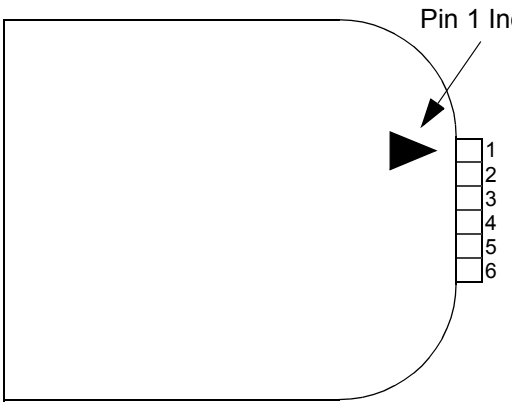
This marker designates the location of pin 1 for proper connector alignment.

1.2.4 Programming Connector

The programming connector is a 6-pin header (0.100" spacing) that connects to the target device. See the pinout specification in Figure 1-2.

© 2010 Microchip Technology Inc.

FIGURE 1-2: PICkit™ 3 PROGRAMMER CONNECTOR PINOUT



Pin Description*
1 = $\overline{\text{MCLR}}$/VPP
2 = VDD Target
3 = VSS (ground)
4 = PGD (ICSPDAT)
5 = PGC (ICSPCLK)
6 = PGM (LVP)

* The 6-pin header (0.100" spacing) accepts 0.025" square pins.

Note: The programming connector pin functions are different for programming Serial EEPROM devices. See the ReadMe file for the PICkit 3 (*Help>Readme)* included with the MPLAB IDE software for these pinouts.

1.2.5 Status LEDs

The Status LEDs indicate the status of the PICkit 3.

1. **Power** (green) – Power is supplied to the PICkit 3 via the USB port.
2. **Active** (blue) – The PICkit 3 has connection to the PC USB port and the communication link is active.
3. **Status:**

   Busy (yellow) – The PICkit 3 is busy with a function in progress, such as programming.

   Error (red) – The PICkit 3 has encountered an error.

1.2.6 Push Button

The push button is used for Programmer-To-Go. See **Chapter 6. "PICkit 3 Programmer-To-Go"**.

1.3 HOW THE PICkit 3 PROGRAMMER/DEBUGGER HELPS YOU

The PICkit 3 programmer/debugger allows you to:

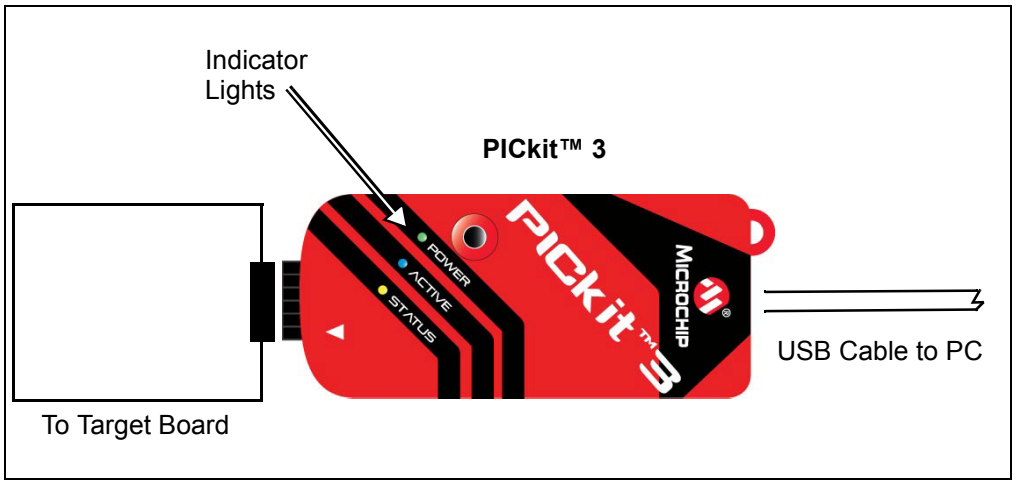
- Debug your application on your own hardware in real time
- Debug with hardware breakpoints
- Set breakpoints based on internal events
- Monitor internal file registers
- Emulate at full speed
- Program your device

1.4 PICkit 3 PROGRAMMER/DEBUGGER COMPONENTS

The components of the PICkit 3 programmer/debugger system are:

1. PICkit 3 with indicator lights for power, activity and status.
2. USB cable to provide communications between the debugger and a PC and to provide power to the debugger.
3. CD-ROM with MPLAB IDE software and online documentation.

FIGURE 1-3: BASIC DEBUGGER SYSTEM



Additional hardware that may be ordered separately:

- PICkit 3 Debug Express Kit which includes:
  - a 44-pin demo board with a PIC18F45K20 MCU
  - free version of MPLAB C Compiler for PIC18 MCUs
  - easy-to-understand lessons and tutorials
  - other software utilities, examples with source code and full documentation
- Transition socket
- ICD headers
- MPLAB IDE processor extension kits

© 2010 Microchip Technology Inc.

Chapter 2. Theory of Operation

2.1 INTRODUCTION

A simplified description of how the PICkit 3 programmer/debugger system works is provided here. It is intended to provide enough information so a target board can be designed that is compatible with the debugger for both emulation and programming operations. The basic theory of in-circuit emulation and programming is described so that problems, if encountered, are quickly resolved.

- PICkit 3 vs. PICkit 2
- Debugger to Target Communication
- Communication Connections
- Debugging
- Requirements for Debugging
- Programming
- Resources Used by the Debugger

2.2 PICkit 3 VS. PICkit 2

The PICkit 3 programmer/debugger system is similar in function to the PICkit 2 in-circuit debugger system. Similarities of the two debuggers include:

- Powered via USB cable to PC
- Provides a programmable voltage power supply

The PICkit 3 differs from the PICkit 2 by providing:

- Extended EE program image space (512 Kbytes)
- True voltage reference
- Increased voltage range (1.8-5V V_{DD}; 1.8-14V V_{PP})

2.3 DEBUGGER TO TARGET COMMUNICATION

The debugger system configurations are discussed in the following sections.

CAUTION
Do not change hardware connections while the PICkit 3 or target is powered.

Standard ICSP Device Communication

The debugger system can be configured to use standard ICSP communication for both programming and debugging functions. This 6-pin connection is the same one used by the PICkit 2 programmer/debugger.

The modular cable can be either (1) inserted into a matching socket at the target, where the target device is on the target board (Figure 2-1), or (2) inserted into a standard adapter/header board combo (available as a Processor Pak), which is then plugged into the target board (Figure 2-2).

Note:	Older header boards used a 6-pin (RJ-11) connector instead of an 8-pin connector, so these headers may be connected to the debugger with the AC164110 ICSP adapter.

For more on standard communication, see **Appendix A. "Hardware Specification"**.

FIGURE 2-1: STANDARD DEBUGGER SYSTEM – DEVICE WITH ON-BOARD ICE CIRCUITRY

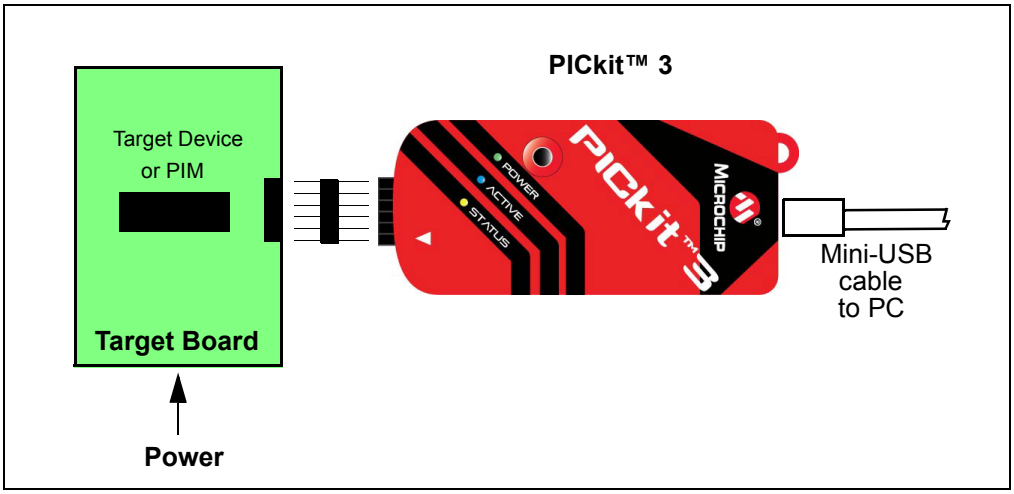
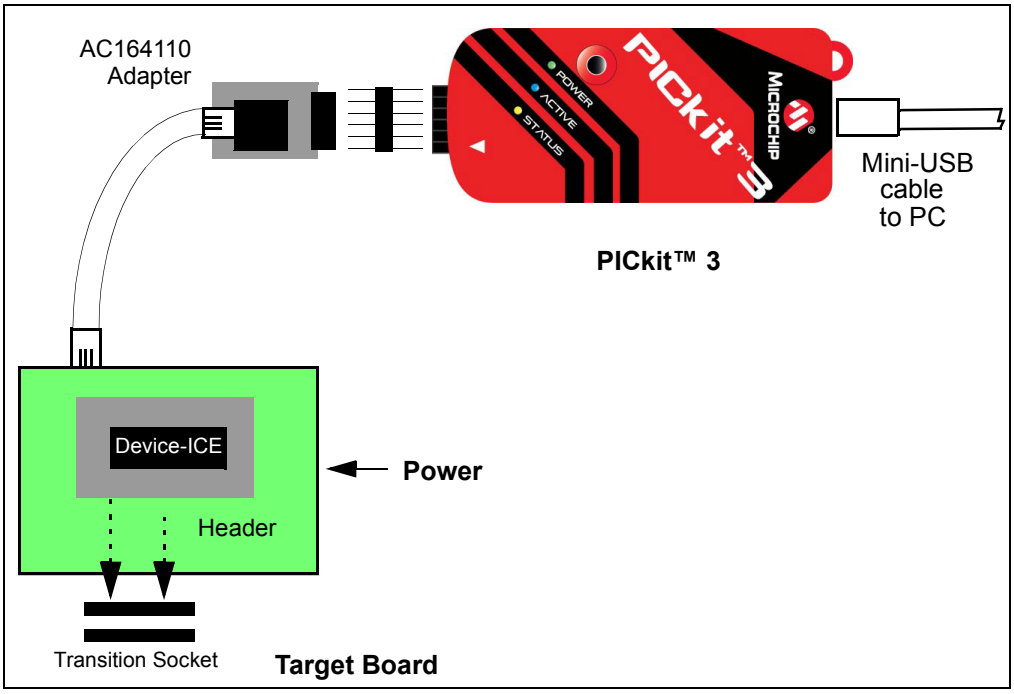


FIGURE 2-2: STANDARD DEBUGGER SYSTEM – ICE DEVICE



© 2010 Microchip Technology Inc.

2.4 COMMUNICATION CONNECTIONS

2.4.1 Communication Target Connections

2.4.1.1 USING SINGLE IN-LINE CONNECTOR

Use the 6-pin in-line connector between the PICkit 3 programmer/debugger and the target board connector. See Figure 2-1. Also see Table 2-1 and **Section A.6 “Standard Communication Hardware”**.

TABLE 2-1: TARGET CONNECTOR PINOUT

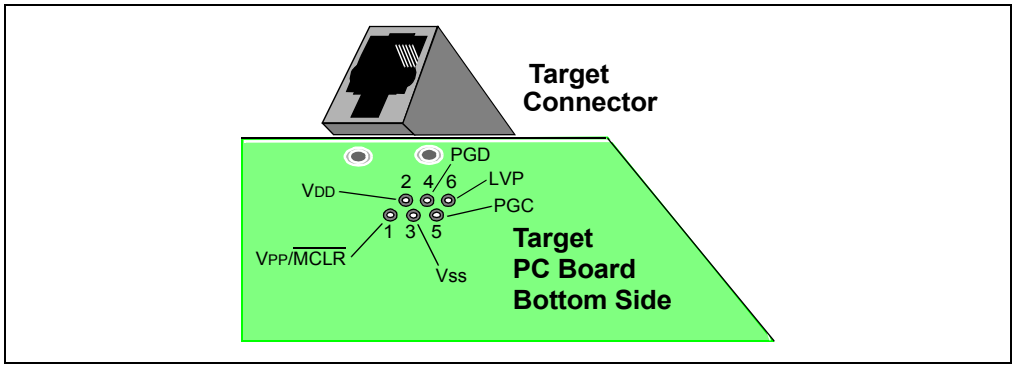
Connector Pin	Microcontroller Pin
1	$\overline{\text{MCLR}}$/VPP
2	VDD
3	Ground
4	PGD (ICSPDAT)
5	PGC (ICSPCLK)
6	PGM (LVP)

2.4.1.2 USING AN ADAPTER

Use the AC164110 adapter between the PICkit 3 programmer/debugger and the target device with the modular interface (six conductor) cable. The pin numbering for the connector is shown from the bottom of the target PC board in Figure 2-3.

Note: Cable connections at the debugger and target are mirror images of each other, i.e., pin 1 on one end of the cable is connected to pin 6 on the other end of the cable. See **Section A.6.2.3 “Modular Cable Specification”**.

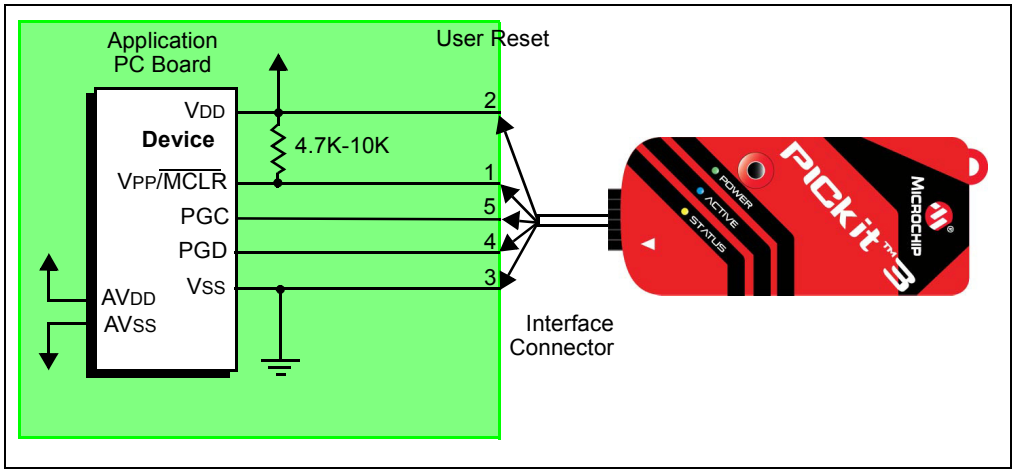
FIGURE 2-3: STANDARD RJ-11 CONNECTION AT TARGET



2.4.2 Target Connection Circuitry

Figure 2-4 shows the interconnections of the PICkit 3 programmer/debugger to the connector on the target board. The diagram also shows the wiring from the connector to a device on the target PC board. A pull-up resistor (usually around 10 kΩ) is recommended to be connected from the VPP/MCLR line to VDD so that the line may be strobed low to reset the device.

FIGURE 2-4: STANDARD CONNECTION TARGET CIRCUITRY



2.4.3 Target Powered

In the following descriptions, only three lines are active and relevant to core debugger operation: pins 1 (V_{PP}/MCLR), 5 (PGC) and 4 (PGD). Pins 2 (V_{DD}) and 3 (V_{SS}) are shown on Figure 2-4 for completeness. PICkit 3 has two configurations for powering the target device: internal debugger and external target power.

The recommended source of power is external and derived from the target application. In this configuration, target V_{DD} is sensed by the debugger to allow level translation for the target low voltage operation. If the debugger does not sense voltage on its V_{DD} line (pin 2 of the interface connector), it will not operate.

2.4.4 Debugger Powered

The internal debugger power is limited to 30 mA. This may be of benefit for very small applications that have the device V_{DD} separated from the rest of the application circuit for independent programming, but is not recommended for general usage as it imposes more current demands from the USB power system derived from the PC.

Not all devices have the AV_{DD} and AV_{SS} lines, but if they are present on the target device, all must be connected to the appropriate levels in order for the debugger to operate. They cannot be left floating.

Also, devices with a V_{CAP} line (PIC18FXXJ for example) should be connected to the appropriate capacitor or level.

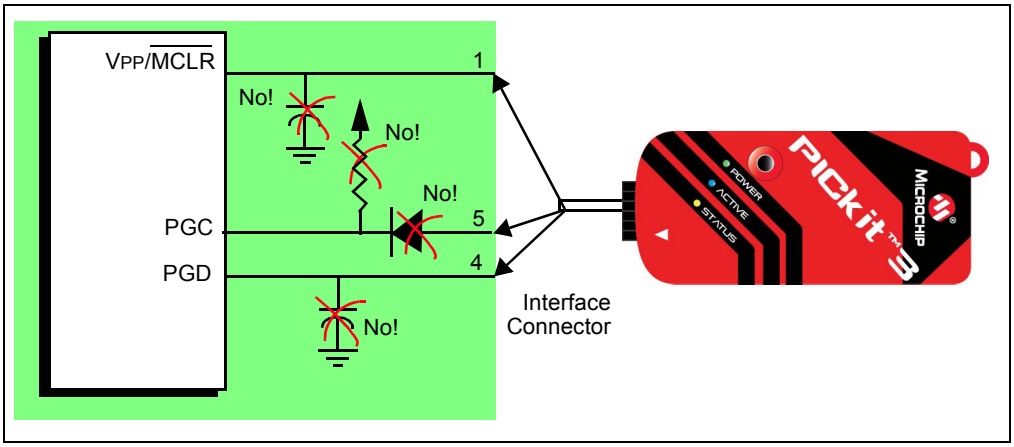
Note:	The interconnection is very simple. Any problems experienced are often caused by other connections or components on these critical lines that interfere with the operation of the PICkit 3 programmer/debugger, as discussed in the following section.

© 2010 Microchip Technology Inc.

2.4.5 Circuits That Will Prevent the Debugger From Functioning

Figure 2-5 shows the active debugger lines with some components that will prevent the PICkit 3 debugger system from functioning.

FIGURE 2-5: IMPROPER CIRCUIT COMPONENTS



Specifically, these guidelines must be followed:

- Do not use pull-ups on PGC/PGD – they will disrupt the voltage levels, since these lines have 4.7 kΩ pull-down resistors in the debugger.
- Do not use capacitors on PGC/PGD – they will prevent fast transitions on data and clock lines during programming and debug communications.
- Do not use capacitors on $\overline{\text{MCLR}}$ – they will prevent fast transitions of VPP. A simple pull-up resistor is generally sufficient.
- Do not use diodes on PGC/PGD – they will prevent bidirectional communication between the debugger and the target device.

2.5 DEBUGGING

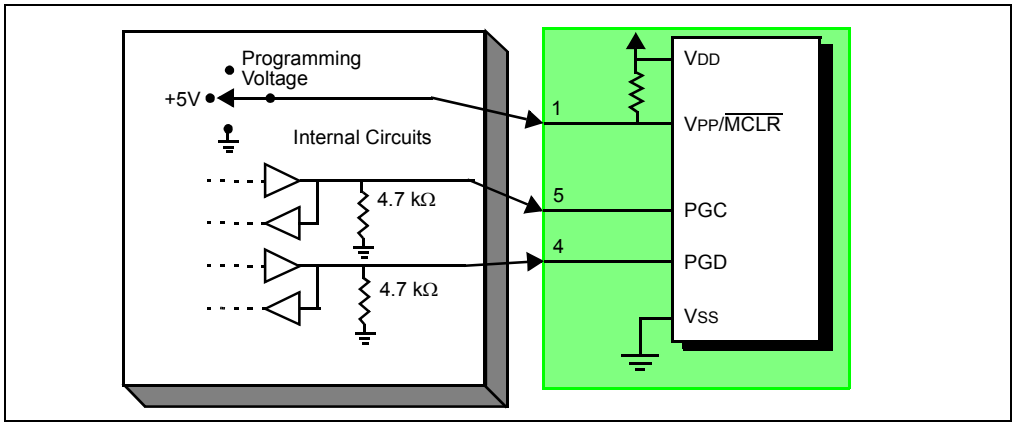
There are two steps to using the PICkit 3 programmer/debugger system as a debugger. The first requires that an application be programmed into the target device (usually with the PICkit 3 itself). The second uses the internal in-circuit debug hardware of the target Flash device to run and test the application program. These two steps are directly related to the MPLAB IDE operations:

1. Program the code into the target and activate special debug functions (see the next section for details).
2. Use the debugger to set breakpoints and run.

If the target device cannot be programmed correctly, the PICkit 3 programmer/debugger will not be able to debug.

Figure 2-6 shows the basic interconnections required for programming. Note that this is the same as Figure 2-4, but for the sake of clarity, the VDD and VSS lines from the debugger are not shown.

FIGURE 2-6: PROPER CONNECTIONS FOR PROGRAMMING



A simplified diagram of some of the internal interface circuitry of the PICkit 3 programmer/debugger is shown. For programming, no clock is needed on the target device, but power must be supplied. When programming, the debugger puts programming levels on VPP/MCLR, 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 device under development.

2.6 REQUIREMENTS FOR DEBUGGING

To debug (set breakpoints, see registers, etc.) with the PICkit 3 programmer/debugger system, there are critical elements that must be working correctly:

- The debugger must be connected to a PC. It must be powered by the PC via the USB cable, and it must be communicating with the MPLAB IDE software via the USB cable. See **Chapter 3. "Installation"** for details.
- The debugger must be connected as shown to the VPP, PGC and PGD pins of the target device with the modular interface cable (or equivalent). VSS and VDD are also required to be connected between the debugger and target device.
- The target device must have power and a functional, running oscillator. If the target device does not run, for any reason, the PICkit 3 programmer/debugger cannot debug.
- The target device must have its Configuration Words programmed correctly:
  - The oscillator Configuration bits should correspond to RC, XT, etc., depending upon the target design.
  - For some devices, the Watchdog Timer is enabled by default and needs to be disabled.
  - The target device must not have code protection enabled.
  - The target device must not have table read protection enabled.
- PGM (LVP) should be disabled.

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

- Sequence of Operations Leading to Debugging
- Debugging Details

© 2010 Microchip Technology Inc.

2.6.1 Sequence of Operations Leading to Debugging

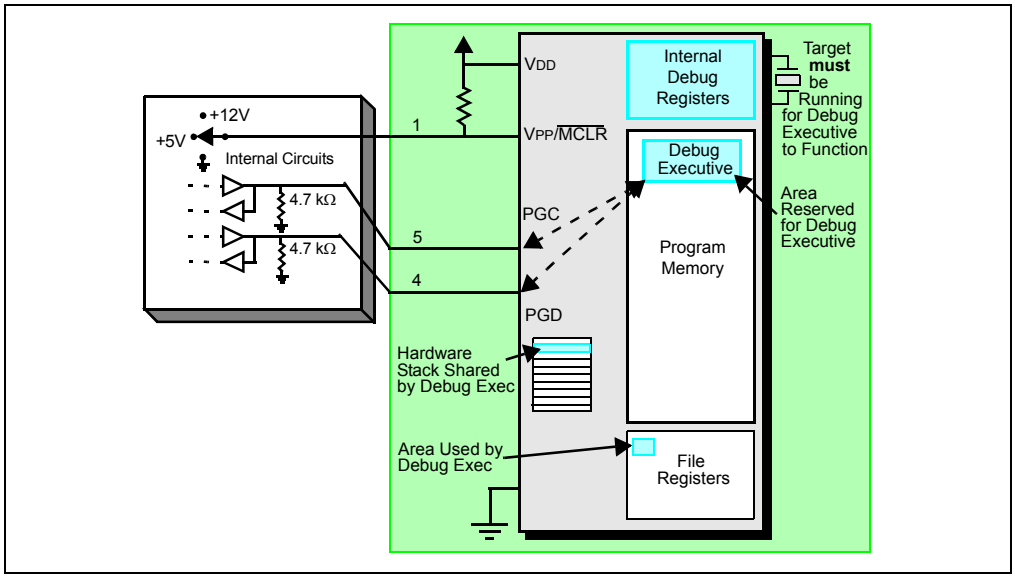
Given that the requirements for debugging (see previous section) are met, these actions can be performed when the PICkit 3 programmer/debugger is set as the current debugger from the MPLAB IDE menu (*Debugger>Select Tool>PICkit 3*):

- The application code is compiled/assembled by selecting *Project>Build Configuration>Debug*.
- When *Debugger>Program* is selected, the application code is programmed into the device'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 device automatically by MPLAB IDE. Since the debug executive must reside in program memory, the application program must not use this reserved space. Some devices have special memory areas dedicated to the debug executive. Check your device data sheet for details.
- Special "in-circuit debug" registers in the target device are enabled. These allow the debug executive to be activated by the debugger.
- The target device is held in Reset by keeping the VPP/$\overline{\text{MCLR}}$ line low.

2.6.2 Debugging Details

Figure 2-7 illustrates the PICkit 3 programmer/debugger system when it is ready for debugging.

FIGURE 2-7: PICkit™ 3 DEBUGGER 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 MPLAB IDE, the address of the breakpoint is stored in the special internal debug registers of the target device. 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. The debugger will then tell the debug executive to run. The target will start from the Reset vector 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 device "fires" and transfers the device's Program Counter to the debug executive (much like an interrupt) and the user's application is effectively

halted. The debugger communicates with the debug executive via PGC and PGD, gets the breakpoint status information and sends it back to MPLAB IDE. MPLAB IDE then sends a series of queries to the debugger to get information about the target device, 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 stack for its temporary variables. If the device 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 the PICkit 3 programmer/debugger 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 so that the in-circuit debug mechanism of the target device 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 the debugger communications with the debug executive to interrogate the state of the target device.

2.7 PROGRAMMING

There are three ways to program a device with the PICkit 3 unit:

- Through MPLAB IDE with the PICkit 3 connected to the PC.
- Through PICkit 3 Programmer-To-Go, after setting it up through MPLAB IDE. (See **Chapter 6. "PICkit 3 Programmer-To-Go"** for more information.)
- Through PICkit 3 Programmer Application, a software program that allows you to program devices with PICkit 3 without using MPLAB IDE. (See "*PICkit™ 3 Programmer Application User's Guide"* for instructions.)

Use the PICkit 3 programmer/debugger as a programmer to program an actual (non -ICE/-ICD) device, i.e., a device not on a header board. From the MPLAB IDE menu, select "PICkit 3" from *Programmer>Select Programmer* and compile/assemble your application code with the "Build Configuration" list box on the MPLAB IDE toolbar set to "Release". Also, it may be set by selecting *Project>Build Configuration>Release*.

All debug features are turned off or removed when the debugger is used as a programmer. When using the *Programmer>Program* selection to program a device, MPLAB IDE will disable the in-circuit debug registers so the PICkit 3 programmer/debugger will program only the target application code and the Configuration bits (and EEPROM data, if available and selected) into the target device. The debug executive will not be loaded. As a programmer, the debugger can only toggle the $\overline{\text{MCLR}}$ line to reset and start the target. A breakpoint cannot be set, and register contents cannot be seen or altered.

The PICkit 3 programmer/debugger system programs the target using ICSP. V_{PP}, PGC and PGD lines should be connected as described previously. No clock is required while programming, and all modes of the processor can be programmed, including code protection, Watchdog Timer and table read protection.

2.8 RESOURCES USED BY THE DEBUGGER

For a complete list of resources used by the debugger for your device, please see the online help file in MPLAB IDE for the PICkit 3 programmer/debugger.

© 2010 Microchip Technology Inc.

Chapter 3. Installation

3.1 INTRODUCTION

How to install the PICkit 3 programmer/debugger system is discussed.

- Installing the Software
- Connecting the Target
- Setting Up the Target Board
- Setting Up MPLAB® IDE

3.2 INSTALLING THE SOFTWARE

To install the MPLAB IDE software, first acquire the latest MPLAB IDE installation executable (`MP`*xxxxx*`.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). Then run the executable and follow the screens to install MPLAB IDE.

Note:	MPLAB IDE v8.20 or greater is required to use the PICkit 3 programmer/debugger.

3.3 CONNECTING THE TARGET

A connection is built-in to select the type of communication with the target. See **Section 2.3 "Debugger to Target Communication"** for more details and a diagram.

1. Plug in the USB/power cable if not already connected.
2. Attach the communication cable(s) between debugger and target if using RJ11 plug or connect directly to a 6-pin inline header.

FIGURE 3-1: INSERT COMMUNICATIONS AND USB/POWER CABLES

3.4 SETTING UP THE TARGET BOARD

3.4.1 Using Production Devices

For production devices, the debugger may be connected directly to the target board. The device on the target board must have built-in debug circuitry in order for the PICkit 3 programmer/debugger to perform emulation with it. Consult the device data sheet to see if the device has the needed debug circuitry, i.e., it should have a "Background Debugger Enable" Configuration bit.

Note:	In the future, devices with circuitry that support ICD may be used.

The target board must have a connector to accommodate the communications chosen for the debugger. For connection information, see **Section 2.3 "Debugger to Target Communication"**, "Standard ICSP Device Communication".

3.4.2 Using ICE Devices

For ICE devices, an ICE header board is required. The header board contains the hardware necessary to emulate a specific device or family of devices. For more information on ICE headers, see the "*Header Board Specification"* (DS51292).

Note:	In the future, ICD header boards with ICD devices (*Device*-ICD) may be used.

A transition socket is used with the ICE header to connect the header to the target board. Transition sockets are available in various styles to allow a common header to be connected to one of the supported surface mount package styles. For more information on transition sockets, see the *"Transition Socket Specification"* (DS51194).

Header board layout will be different for headers or processor extension packs. For connection information, see **Section 2.3 "Debugger to Target Communication"**, "Standard ICSP Device Communication".

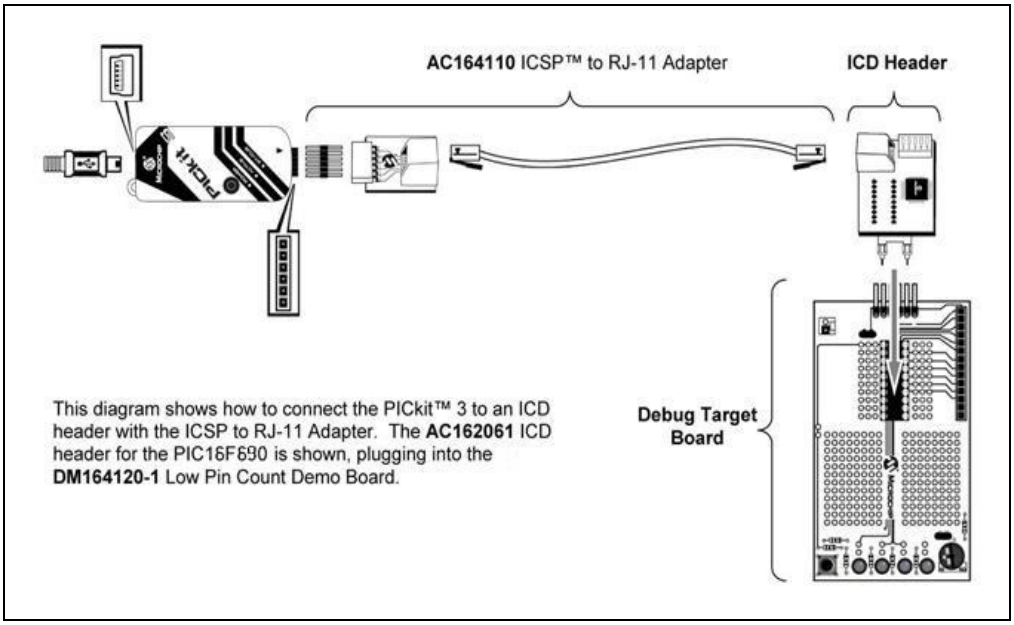
3.4.3 Using an ICD Header

All Baseline and some Mid-Range PIC microcontrollers require a special –ICD device mounted on a debug header circuit board to enable the debugging feature. For a list of these devices and the required ICD header board part number, please see the "*Header Board Specification*" (DS51292). The Header Board Specification is included on the PICkit 3 CD-ROM, and is available online at www.microchip.com.

Each ICD header board comes with the necessary –ICD device, and is used on the target board instead of the production microcontroller. However, most header boards have an RJ-11 debug connector which requires the AC164110 RJ-11 to ICSP adapter kit to connect it to PICkit 3. Figure 3-2 illustrates using the AC162061 ICD Header for the PIC18F45K20 with the AC164110 adapter kit and Low Pin Count Demo Board.

© 2010 Microchip Technology Inc.

FIGURE 3-2: USING AN ICD HEADER BOARD



Many Mid-Range PIC microcontrollers and all PIC18 and 16-bit PIC microcontroller devices do not require an ICD header and can be debugged directly through the ICSP programming connections.

3.4.4 Powering the Target

These are configuration essentials:

- When using the USB connection, PICkit 3 can be powered from the PC but it can only provide a limited amount of current, up to 30 mA, at V_{DD} from 1.8-5V to a small target board.
- The desired method is for the target to provide V_{DD} since it can provide a higher current. The additional benefit is that plug-and-play target detection facility is inherited, i.e., MPLAB IDE will let you know in the Output window when it has detected the target and has detected the device.

Note: The target voltage is only used for powering up the drivers for the ICSP interface; the target voltage does not power up the PICkit 3. The PICkit 3 power is derived strictly from the USB port.

If you have not already done so, connect the PICkit 3 to the target using the appropriate cables (see **Section 3.3 "Connecting the Target"**). Then power the target. If you are powering the target through the PICkit 3, see **Section 10.5.8 "Settings Dialog, Power Tab"** for instructions.

3.5 SETTING UP MPLAB® IDE

Once the hardware is connected and powered, MPLAB IDE may be set up for use with the PICkit 3 programmer/debugger.

On some devices, you must select the communications channel in the Configuration bits, e.g., PGC1/EMUC1 and PGD1/EMUD1. Make sure the pins selected here are the same ones physically connected to the device.

For more on setting up a project and getting started with PICkit 3, see **Chapter 4. "General Setup"**.

NOTES:

© 2010 Microchip Technology Inc.

Chapter 4. General Setup

4.1 INTRODUCTION

How to get started using the PICkit 3 programmer/debugger is discussed.

- Starting the MPLAB IDE Software
- Creating a Project
- Viewing the Project
- Building the Project
- Setting Configuration Bits
- Setting the Debugger or Programmer
- Debugger/Programmer Limitations

4.2 STARTING THE MPLAB IDE SOFTWARE

After installing the MPLAB IDE software (**Section 3.2 "Installing the 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 `mplab ide\core` subdirectory of the MPLAB IDE installation directory.

For more information on using the software, see:

- *"MPLAB® IDE User's Guide"* (DS51519) – Comprehensive guide for using MPLAB IDE.
- The online help files – The most up-to-date information on MPLAB IDE and PICkit 3 programmer/debugger.
- Readme files – Last minute information on each release is included in `Readme for MPLAB IDE.txt` and `Readme for PICkit 3 Debugger.txt`. Both files are found in the `Readmes` subdirectory of the MPLAB IDE installation directory.

4.3 CREATING 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, etc., to the various "nodes" on the Project window. See MPLAB IDE documentation for more detail on using this wizard. The basic steps are provided here:

- Select your device (e.g., PIC18F45K20)
- Select a language toolsuite (e.g., Microchip C Compiler Toolsuite)
- Name the project
- Add application files (e.g., `program.c`, `support.s`, `counter.asm`)

Note: If you do not have a custom linker script in your project, the Project Manager will select the appropriate linker script for you.

4.4 VIEWING THE PROJECT

After the Project Wizard has created a project, the project and its associated files are visible in 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.

See MPLAB IDE documentation for more detail on using the Project window.

4.5 BUILDING THE PROJECT

After the project is created, the application needs to be built. This will create object (hex) code for the application that can be programmed into the target by the PICkit 3 programmer/debugger.

To set build options, select *Project>Build Options>Project*.

> **Note:** On the Project Manager toolbar (*View>Toolbars>Project Manager*), select "Debug" from the drop-down list when using the PICkit 3 as a debugger, or select "Release" when using it as a programmer.

When done, choose *Project>Build All* to build the project.

4.6 SETTING CONFIGURATION BITS

Although device Configuration bits may be set in code, they also may be set in the MPLAB IDE Configuration window. Select *Configure>Configuration Bits*. By clicking on the text in the "Settings" column, these can be changed.

Some Configuration bits of interest are:

- **Watchdog Timer Enable** – On most devices, the Watchdog Timer is enabled initially. It is usually a good idea to disable this bit.
- **Comm Channel Select** – For some devices, you will need to select the communications channel for the device, e.g., PGC1/EMUC1 and PGD1/EMUD1. Make sure the pins selected here are the same ones physically connected to the device.
- **Oscillator** – Select the configuration setting that matches the target oscillator.

4.7 SETTING THE DEBUGGER OR PROGRAMMER

Select *Debugger>Select Tool>PICkit 3* to choose the PICkit 3 programmer/debugger 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 PICkit 3 status and communications will be displayed on the **PICkit 3** tab. For more information, see **Section 10.2 "Debugging Functions"** and **Section 10.3 "Debugging Dialogs/Windows"**.

Select *Programmer>Select Programmer>PICkit 3* to choose the PICkit 3 programmer/debugger as the programmer tool. 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 ICE status and communications will be displayed on the **PICkit 3** tab. For more information, see **Section 10.4 "Programming Functions"**.

Select *Debugger>Settings* or *Programmer>Settings* to open the Settings dialog (**Section 10.5 "Settings Dialog"**) and set up options as needed.

If errors occurs, see:

- **Chapter 9. "Error Messages"**
- **Chapter 8. "Frequently Asked Questions (FAQs)"**

© 2010 Microchip Technology Inc.

4.8 DEBUGGER/PROGRAMMER LIMITATIONS

For a complete list of debugger limitations for your device, please see the PICkit 3 online help file in MPLAB IDE by selecting *Help>Topics>PICkit 3* and click **OK**.

NOTES:

© 2010 Microchip Technology Inc.

Chapter 5. PICkit 3 Debug Express

5.1 INTRODUCTION

The PICkit 3 Debug Express kit works in conjunction with the MPLAB IDE application to run, stop and single-step through programs. One or more breakpoints can be set and the processor can be reset. Once the processor is stopped, the register's contents can be examined and modified.

For more information on how to use MPLAB IDE, reference the following documentation:

- MPLAB® IDE User's Guide (DS51519)
- MPLAB® IDE Online Help

5.2 PICkit 3 DEBUG EXPRESS KIT CONTENTS

The PICkit 3 Debug Express kit (DV164131) contains the following items:

1. The PICkit 3 Development Programmer/Debugger
2. USB cable
3. 44-Pin Demo Board with device*
4. MPLAB IDE CD-ROM
5. PICkit 3 Debug Express C18 Lessons (tutorials) on CD-ROM

* The Explorer 16 board may also be used to debug.

5.3 INSTALLING THE HARDWARE AND SOFTWARE

Install the PICkit 3 hardware and software, if not already done, as specified in **Chapter 3. "Installation"**.

Note:	PICkit 3 Debug Express requires MPLAB IDE version 8.20 or later.

5.3.1 Reserved Resources

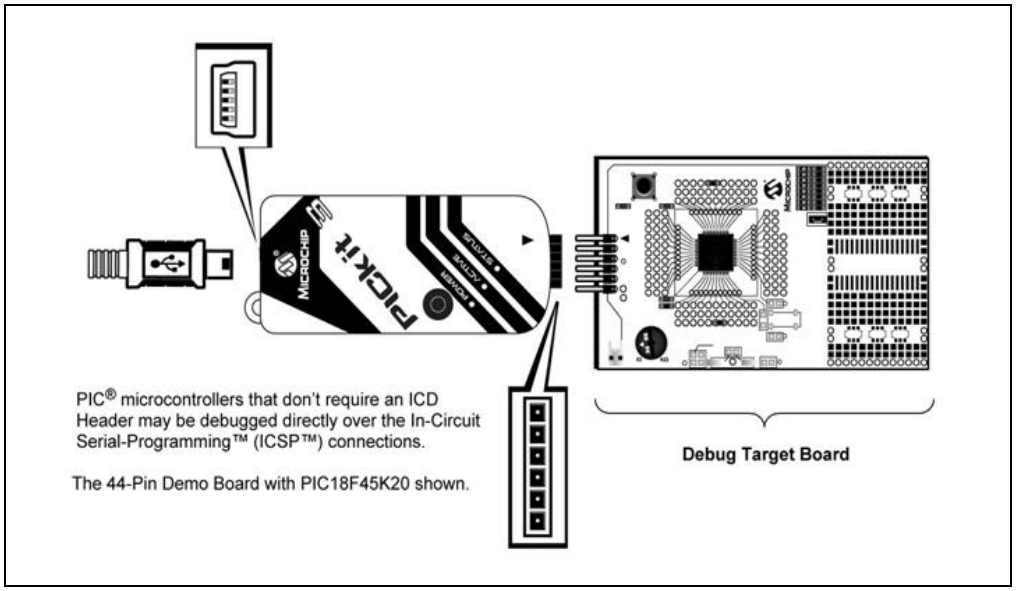
Due to the built-in in-circuit debugging capability of ICD devices and the ICSP function offered by the debugger, the PICkit 3 Debug Express uses some on-chip resources when debugging.

For information on device resources that are needed for in-circuit debugging, please refer to the MPLAB PICkit 3 Help, found in the MPLAB IDE under *Help>Topics*. The device reserved resource information found under "Resources Used By MPLAB PICkit 3" is the same for the PICkit 3 Debug Express.

5.3.2 Connecting the Demo Board

The PIC18F45K20 included on the 44-Pin Demo Board can be debugged by simply connecting the demo board to the PICkit 3 as shown in Figure 5-1.

FIGURE 5-1: CONNECTING THE DEMO BOARD TO THE PICkit™ 3



5.3.3 Configuration Bits and Debug Express

PIC microcontroller devices that do not require an ICD Header and may be debugged directly contain a $\overline{\text{DEBUG}}$ bit in the Configuration Word(s) that enables and disables the Debug mode on the PIC microcontroller.

This bit is automatically set appropriately by the MPLAB IDE when using PICkit 3 Debug Express and should not be specified in source code configuration settings.

CAUTION

The $\overline{\text{DEBUG}}$ configuration bit value should not be specified in source code Configuration settings under normal conditions. Doing so may cause the bit to be asserted when programming a device outside the debugger. This will cause the device to function improperly or not all at in the application circuit.

Many 16-bit PIC microcontroller devices such as PIC24 and dsPIC33 families have multiple ICSP programming and debugging port pins labeled PGC1/EMUC1 and PGD1/EMUD1, PGC2/EMUC2 and PGD2/EMUD2, etc. While any ICSP port may be used for programming, only one port is active at a time for debugging. The active EMU port is set in the device Configuration bits. If the active port setting does not match the EMU port to which the PICkit 3 is connected, the device will be unable to enter debug mode. In the MPLAB IDE Configuration Bits dialog, these bits are usually referred to as the "Comm Channel Select" bits.

© 2010 Microchip Technology Inc.

Chapter 6. PICkit 3 Programmer-To-Go

6.1 INTRODUCTION

The PICkit 3 Programmer-To-Go functionality allows a PIC MCU memory image to be downloaded into the PICkit 3 for later programming into a specific PIC MCU. No software or PC is required to program devices once the PICkit 3 programmer is set up for Programming-To-Go. A USB power source for the PICkit 3 is all that is needed.

Note: Although the PICkit 3 unit is capable of programming and debugging, when using the Programming-To-Go feature, you can only program. No debugging capabilities are available with Programming-To-Go.

Topics discussed in this section are:

- USB Power for PICkit 3 Programmer-To-Go
- PICkit 3 Programmer-To-Go Supported Devices
- Setting up PICkit 3 for Programmer-To-Go Operation
- Using PICkit 3 Programmer-To-Go
- Exiting Programmer-To-Go Mode

6.2 USB POWER FOR PICkit 3 PROGRAMMER-TO-GO

The PICkit 3 programmer hardware does not have the capability to be powered in its entirety by the target through the ICSP connector V_{DD} pin. Therefore, it must be powered by a 5V power supply through the USB mini-B port at the top of the PICkit 3 unit. There are several options for providing power, such as, using:

- Any available PC USB port or USB hub port. (No USB communication is necessary; it is only used to provide power.)
- A USB host port on a portable device.
- A USB power adapter or charger with a USB mini-B connector, either from an automotive power jack or an AC wall plug.
- A portable battery charge or power source for cell phones or other portable devices with USB mini-B connector.
- A custom battery pack that supplies regulated 5V into the PICkit 3 USB port.

6.2.1 Power Requirements

The USB power source used should meet the following minimum criteria:

- Is able to supply at least 100mA of current to the PICkit 3 unit.
- Provides a steady, regulated 4.5V to 5.5V output.

Note 1: Most portable chargers/power devices with their own batteries will not give an indication when their internal battery voltage gets low and the output drops below 4.5V. Therefore, you must be sure the device's battery has sufficient remaining capacity to power the PICkit 3 above 4.5V.

2: Any battery-based power sources should be disconnected from the PICkit 3 unit when it is not in use. Otherwise, the PICkit 3 unit will drain the power source battery.

6.3 PICkit 3 PROGRAMMER-TO-GO SUPPORTED DEVICES

All devices in the following families supported by the PICkit 3 with MPLAB IDE are supported for the Programmer-To-Go operation. Table 6-1 lists supported device families and program memory limitations.

TABLE 6-1: PROGRAMMER-TO-GO SUPPORTED DEVICES

Supported Families	Supported Parts
Baseline	All[1]
Mid-range	All[1]
PIC18F	All[1]
PIC18 J-Series	All[1]
PIC18 K-Series	All[1]
PIC24	All[1,2]
dsPIC33	All[1,2]
dsPIC30	All[1]
dsPIC30 SMPS	All[1]
PIC32MX	All[3]

Note 1: Supports all family parts that are supported by MPLAB IDE. See menu *Help > ReadMe* for a list of parts supported by the application. Large memory parts are supported.

2: PICkit 3 Programmer-To-Go does not support using the Programming Executive (Enhanced ICSP) for these devices. When using PICkit 3 Programmer-To-Go with these parts they will be programmed using low level ICSP methods.

3: Supported by MPLAB IDE v8.53 or greater.

© 2010 Microchip Technology Inc.

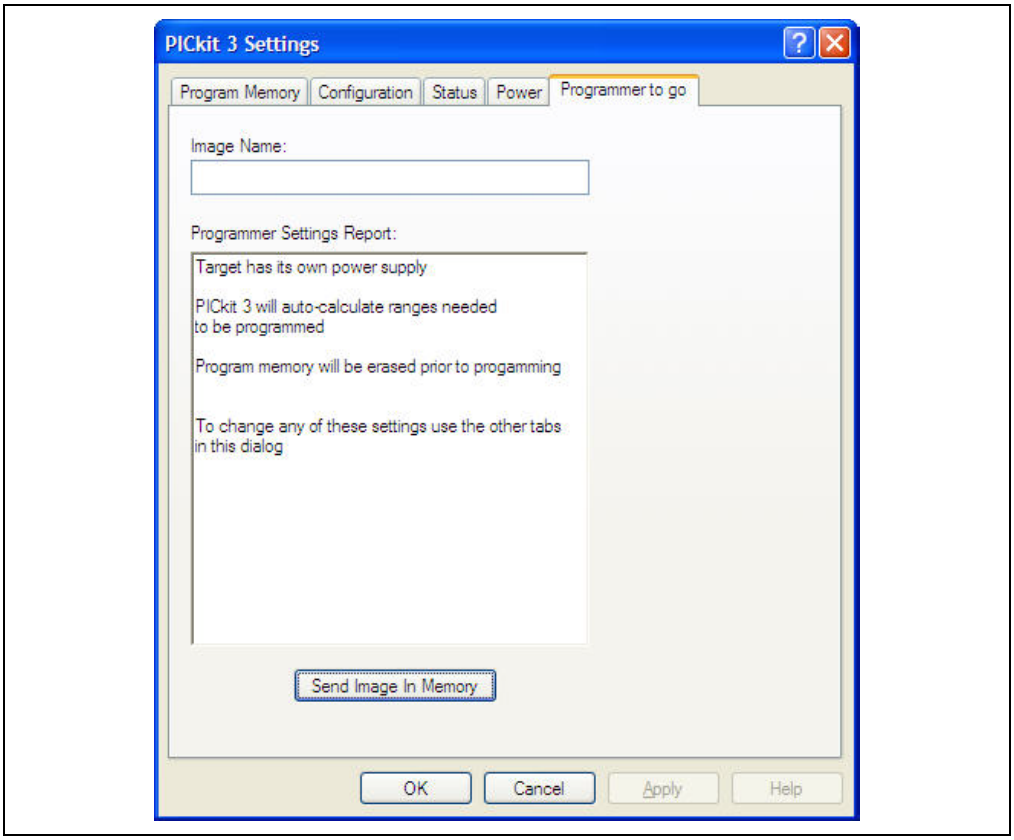
6.4 SETTING UP PICkit 3 FOR PROGRAMMER-TO-GO OPERATION

Before downloading a memory image to PICkit 3 for Programmer-To-Go operation, the PICkit 3 programmer software options and buffers should be set up as desired during Programmer-To-Go operation. In fact, it is highly recommended to test programming a target device from the software first, with all desired options, to ensure the device programs as expected before downloading an image to Programmer-To-Go. Refer to **Chapter 4. "General Setup"** for information on using MPLAB IDE and PICkit 3 to program a device.

6.4.1 PICkit 3 Settings, Programmer to go Tab

From MPLAB IDE, select *Programmer>Programmer to go Tab* (Figure 6-1).

FIGURE 6-1: PICkit 3 PROGRAMMER TO GO TAB

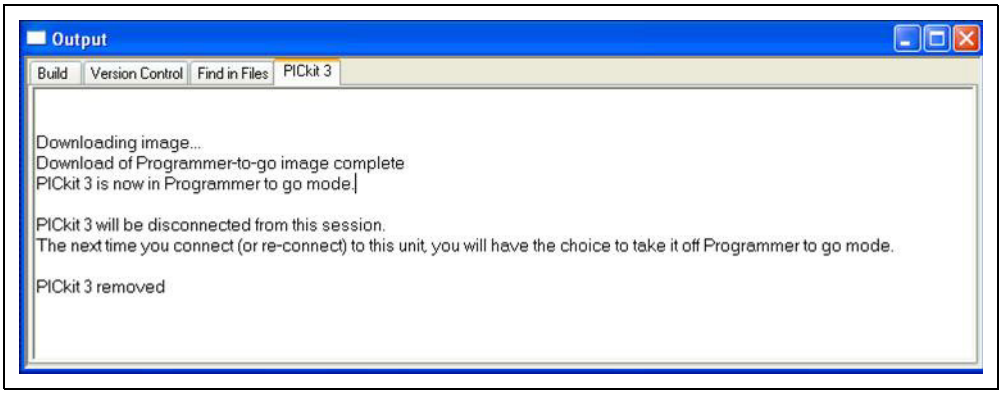


- In the "Image Name" field, type in the name you want to use for your programming image.
- The "Programmer Setting Report" panel displays the setting you've selected for programming the device. If you want to change any of these settings, use the appropriate tab in the PICkit 3 Settings dialog.
- Click **Send Image In Memory** to execute an image transfer to the PICkit 3 unit. Once the image is stored in the PICkit 3, you no longer need MPLAB IDE or a PC to program a device. You can take the PICkit 3 to other locations that may not have a computer and program a device with the push of a button.

6.4.2 Download to PICkit 3 Complete

After you click **Send Image In Memory**, the Output window displays a message (Figure 6-2) when the download is complete. The PICkit 3 unit's "Active" LED should be blinking to indicate it is in Programmer-To-Go mode and ready to program.

FIGURE 6-2: DOWNLOAD COMPLETE



Disconnect the PICkit 3 from the PC USB port. When any USB power source is applied, the PICkit 3 unit will power-up in Programmer-To-Go mode, indicated by the blinking "Active" LED.

6.5 USING PICkit 3 PROGRAMMER-TO-GO

To use PICkit 3 Programmer-To-Go to program a target device once it has been set up, follow the steps below.

1. Connect a USB power source as discussed in **Section 6.2 "USB Power for PICkit 3 Programmer-To-Go"** to the PICkit 3 unit.
2. Ensure the PICkit 3 "Power" LED is lit. The "Active" LED blinks once to indicate the unit is in Programmer-To-Go mode and ready to program.
3. Connect the PICkit 3 unit ICSP connector to the target. Ensure the target is powered properly if not powering from PICkit 3.
4. Press the PICkit 3 push button to begin programming.

During the programming operation the PICkit 3 "Status" LED will turn orange and remain lit continuously while the operation takes place.

When the programming operation is complete, the PICkit 3 unit will provide feedback on the operation via the unit LEDs. A green "Status" LED indicates a successful operation. Red indicates a programming failure. See Table 6-2 for the feedback codes.

© 2010 Microchip Technology Inc.

TABLE 6-2: PROGRAMMER-TO-GO OPERATION FEEDBACK CODES

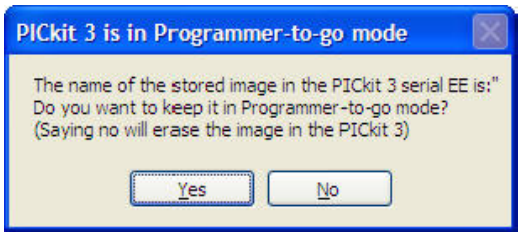
LED Status		Interpretation	
Active LED	**Status LED**	**Code**	**Description**
Single Blink (Blue)	Green	Success/Ready	No errors were encountered during the programming operation. PICkit 3 Programmer-To-Go is ready to program again.
Off	Red – Continuous Rapid Blinking: •••••••.....	VDD/VPP Error	PICkit 3 was unable to set the VDD or VPP voltage to the expected value. If PICkit 3 is not providing VDD, then the error must be a VPP error. See (**Section 2.4 "Communication Connections"** and **Section 3.4.4 "Powering the Target"**) for VDD and VPP information.
Off	Red – 2 blinks in succession: •• •• •• ••.....	Device ID Error	PICkit 3 received an unexpected Device ID from the target. Ensure the target part matches that selected when the PICkit 3 Programmer-To-Go was set up. May indicate a bad ICSP connection preventing the PICkit 3 from communicating with the target. Not applicable to Baseline devices.
Off	Red – 3 blinks in succession: ••• ••• •••.....	Verify Error	The target did not Verify successfully after programming. Ensure the target VDD meets the minimum required. With Baseline devices, this error may indicate ICSP communication problems.
Off	Red – 4 blinks in succession: •••• ••••.....	Internal Error	An unexpected internal Programmer-To-Go error occurred. If it happens a second time, try downloading to the PICkit 3 again.

Note: Press the PICkit 3 push button to clear the error code and initiate a new programming operation.

6.6 EXITING PROGRAMMER-TO-GO MODE

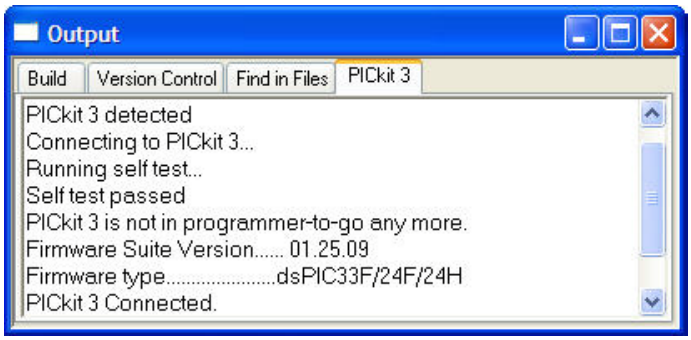
To exit from Programmer-To-Go mode, plug the PICkit 3 unit into a PC USB port and connect to MPLAB IDE. The following message displays these choices:

FIGURE 6-3: EXITING PROGRAMMER-TO-GO MODE



Click **Yes** to preserve the image or **No** to erase the image. If you select **No**, the Output window displays a message similar to Figure 6-4 and exits Programmer-To-Go mode.

FIGURE 6-4: EXITING MESSAGE IN OUTPUT WINDOW



© 2010 Microchip Technology Inc.

Part 2 – Troubleshooting

NOTES:

© 2010 Microchip Technology Inc.

Chapter 7. Troubleshooting First Steps

7.1 INTRODUCTION

If you are having problems with PICkit 3 programmer/debugger operation, start here.

- The 5 Questions to Answer First
- Top 10 Reasons Why You Can't Debug
- Other Things to Consider

7.2 THE 5 QUESTIONS TO ANSWER FIRST

1. What device are you working with? Often an upgrade to a newer version of MPLAB IDE is required to support newer devices. That is, yellow light = untested support.
2. Are you using a Microchip demo board or one of your own design? Have you followed the guidelines for resistors/capacitors for communications connections? See **Chapter 2. "Theory of Operation"**.
3. Have you powered the target? The debugger cannot power the target if greater than 30 mA.
4. Are you using a USB hub in your set up? Is it powered? If you continue to have problems, try using the debugger without the hub (plugged directly into the PC.)
5. Are you using the standard communication cable (RJ-11) shipped with the debugger? If you have made a longer cable, it can cause communications errors.

7.3 TOP 10 REASONS WHY YOU CAN'T DEBUG

1. The oscillator is not working. Check your Configuration bits setting for the oscillator.
2. The target board is not powered. Check the power cable connection.
3. The debugger has become physically disconnected from the PC and/or the target board. Check the communications cables' connections.
4. The device is code-protected. Check your Configuration bit's setting for code protection.
5. You are trying to rebuild the project while in Release mode. Select **Debug** in the Build Configuration drop-down list on the project toolbar, then rebuild the project.
6. The debugger is selected as a programmer, and not as a debugger, in MPLAB IDE.
7. The debugger to PC communications has been interrupted. Reconnect to the debugger in MPLAB IDE.
8. The target application has become corrupted or contains errors.Try rebuilding and reprogramming the target application. Then initiate a Power-on Reset of the target.
9. Other configuration settings are interfering with debugging. Any configuration setting that would prevent the target from executing code will also prevent the debugger from putting the code into Debug mode.
10. The debugger cannot always perform the action requested. For example, the debugger cannot set a breakpoint if the target application is currently running.

7.4 OTHER THINGS TO CONSIDER

1. It is possible the error was a one-time glitch. Try the operation again.
2. There may be a problem programming in general. As a test, switch to programmer mode and program the target with the simplest application possible (e.g., a program to blink an LED). If the program will not run, then you know that something is wrong with the target setup.
3. It is possible that the target device has been damaged in some way (e.g., over current.) Development environments are notoriously hostile to components. Consider trying another target device.
4. Microchip Technology Inc. offers demonstration boards to support most of its microcontrollers. Consider using one of these applications, which are known to work, to verify correct PICkit 3 programmer/debugger functionality.
5. Review debugger debug operation to ensure proper application setup (**Chapter 2. "Theory of Operation"**.)
6. If the problem persists contact Microchip.

© 2010 Microchip Technology Inc.

Chapter 8. Frequently Asked Questions (FAQs)

8.1 INTRODUCTION

Look here for answers to frequently asked questions about the PICkit 3 programmer/debugger system.

- How Does It Work
- What's Wrong

8.2 HOW DOES IT WORK

- **What's in the silicon that allows it to communicate with the PICkit 3 programmer/debugger?**
  The PICkit 3 programmer/debugger can communicate with Flash silicon via the ICSP interface. It uses the debug executive downloaded into program or test memory.

- **How is the throughput of the processor affected by having to run the debug executive?**
  The debug executive doesn't run while in Run mode, so there is no throughput reduction when running your code, i.e., the debugger doesn't 'steal' any cycles from the target device.

- **How does the PICkit 3 programmer/debugger compare with other in-circuit emulators/debuggers?**
  Please refer to **Section 2.2 "PICkit 3 vs. PICkit 2"**.

- **On the MPLAB ICE 2000/4000 debuggers, the data must come out on the bus in order to perform a complex trigger on that data. Is this also required on the PICkit 3 programmer/debugger? For example, could I halt based on a flag going high?**
  The MPLAB ICE 2000/4000 debuggers use a special debugger chip (-ME) for monitoring. There is no -ME with the PICkit 3 programmer/debugger so there are no busses to monitor externally. With the PICkit 3 programmer/debugger, rather than using external breakpoints, the built-in breakpoint circuitry of the debug engine is used – the busses and breakpoint logic are monitored inside the part.

- **Does the PICkit 3 programmer/debugger have complex breakpoints like MPLAB ICE 2000/4000?**
  No. But you can break based on a value in a data memory location or program address. See **Section 10.3.1 "Breakpoints Dialog"** for more information.

- **Is the PICkit 3 optoisolated or electrically isolated?**
  No. You cannot apply a floating or high voltage (120V) to the current system.

- **What limitations are there with the standard cable?**
  The standard ICSP RJ-11 cable does not allow for clock speeds greater than about 15 Mbps.

- **Will the PICkit 3 slow down the running of the program?**
  No, the device will run at any device speed as specified in the data sheet.

- **Is it possible to debug a dsPIC DSC running at any speed?**
  The PICkit 3 is capable of debugging at any device speed as specified in the device's data sheet.

- **What is the function of pin 6, the LVP pin?**
  Pin 6 is reserved for the LVP (Low-Voltage Programming) connection.

8.3 WHAT'S WRONG

- **My PC went into power-down/hibernate mode, and now my debugger won't work. What happened?**
  When using the debugger for prolonged periods of time, and especially as a debugger, be sure to disable the Hibernate mode in the Power Options Dialog window of your PC's operating system. Go to the Hibernate tab and clear or uncheck the "Enable hibernation" check box. This will ensure that all communication is maintained across all the USB subsystem components.

- **I set my peripheral to NOT freeze on halt, but it is suddenly freezing. What's going on?**
  For dsPIC30F/33F and PIC24F/H devices, a reserved bit in the peripheral control register (usually either bit 14 or 5) is used as a Freeze bit by the debugger. If you have performed a write to the entire register, you may have overwritten this bit. (The bit is user-accessible in Debug mode.)
  To avoid this problem, write only to the bits you wish to change for your application (`BTS`, `BTC`) instead of to the entire register (`MOV`).

- **When using a 16-bit device, an unexpected Reset occurred. How do I determine what caused it?**
  Some things to consider:
  - To determine a Reset source, check the RCON register.
  - Handle traps/interrupts in an Interrupt Service Routine (ISR). You should include `trap.c` style code, i.e.,

```
void __attribute__((__interrupt__)) _OscillatorFail(void);
       :
void __attribute__((__interrupt__)) _AltOscillatorFail(void);
       :
void __attribute__((__interrupt__)) _OscillatorFail(void)
{
     INTCON1bits.OSCFAIL = 0;        //Clear the trap flag
     while (1);
}
       :
void __attribute__((__interrupt__)) _AltOscillatorFail(void)
{
     INTCON1bits.OSCFAIL = 0;
     while (1);
}
       :
```

  - Use ASSERTs.

© 2010 Microchip Technology Inc.

- **I have finished debugging my code. Now I've programmed my part, but it won't run. What's wrong?**

  Some things to consider are:

  - Have you selected the debugger as a programmer and then tried to program a header board? A header board contains an -ICE/-ICD version of the device and may not function like the actual device. Only program regular devices with the debugger as a programmer. Regular devices include devices that have on-board ICE/ICD circuitry, but are not the special -ICE/-ICD devices found on header boards.
  - Have you selected the debugger as a debugger and then tried to program a production device? Programming a device when the debugger is a debugger will program a debug executive into program memory and set up other device features for debug (see **Section 2.6.1 "Sequence of Operations Leading to Debugging"**). To program final (release) code, select the debugger as a programmer.
  - Have you selected "Release" from the Build Configuration drop-down list or Project menu? You must do this for final (release) code. Rebuild your project, reprogram the device, and try to run your code again.

- **I don't see my problem here. Now what?**

  Try the following resources:

  - **Section 2.8 "Resources Used by the Debugger"**
  - **Section 9.2 "Specific Error Messages"**
  - **Section 9.3 "General Corrective Actions"**

NOTES:

© 2010 Microchip Technology Inc.

Chapter 9. Error Messages

9.1 INTRODUCTION

The PICkit 3 programmer/debugger produces many different error messages; some are specific and others can be resolved with general corrective actions.

- Specific Error Messages
- General Corrective Actions

9.2 SPECIFIC ERROR MESSAGES

PICkit 3 programmer/debugger error messages are listed below in numeric order.

Note: Numbers may not yet appear in displayed messages. Use the Search tab on the Help viewer to find your message and highlight it below.

Text in error messages listed below of the form %x (a variable) will display as text relevant to your particular situation in the actual error message.

PK3Err0001: Failed while writing to program memory.

PK3Err0002: Failed while writing to EEPROM.

PK3Err0003: Failed while writing to configuration memory.

See **Section 9.3.1 "Read/Write Error Actions"**.

PK3Err0005: PICkit 3 is currently busy and cannot be unloaded at this time.

If you receive this error when attempting to deselect the debugger as a debugger or programmer:

1. Wait – give the debugger time to finish any application tasks. Then try to deselect the debugger again.
2. Select Halt to stop any running applications. Then try to deselect the debugger again.
3. Unplug the debugger from the PC. Then try to deselect the debugger again.
4. Shut down MPLAB IDE.

PK3Err0006: Failed while writing to user ID memory.

PK3Err0007: Failed while reading program memory.

PK3Err0008: Failed while reading EEPROM.

PK3Err0009: Failed while reading configuration memory.

PK3Err0010: Failed while reading user ID memory.

See **Section 9.3.1 "Read/Write Error Actions"**.

PK3Err0011: Bulk erase failed.

See **Section 9.3.1 "Read/Write Error Actions"**.

If these do not work, try another device.

PK3Err0012: Download debug exec failed

If you receive this error while attempting to program from the Debugger menu:

1. Deselect the debugger as the debug tool.
2. Close your project and then close MPLAB IDE.
3. Restart MPLAB IDE and re-open your project.
4. Reselect the debugger as your debug tool and attempt to program your target device again.

If this does not work, see **Section 9.3.4 "Corrupted Installation Actions"**.

PK3Err0013: NMMR register write failed.

PK3Err0014: File register write failed.

See **Section 9.3.2 "Debugger-to-Target Communication Error Actions"**.

PK3Err0015: Data transfer was unsuccessful. %d byte(s) expected, %d byte(s) transferred.

See **Section 9.3.3 "Debugger-to-PC Communication Error Actions"**.

PK3Err0016: Cannot transmit. PICkit 3 not found.

The debugger is not connected to the PC.

PK3Err0017: File register read failed.

PK3Err0018: NMMR register read failed.

PK3Err0019: Failed while reading emulation registers.

PK3Err0020: Failed while writing emulation registers.

See **Section 9.3.2 "Debugger-to-Target Communication Error Actions"**.

PK3Err0021: Command not echoed properly. Sent %x, received %x.

PK3Err0022: Failed to get PICkit 3 version information.

PK3Err0024: Download RS failed.

PK3Err0025: Download AP failed.

See **Section 9.3.3 "Debugger-to-PC Communication Error Actions"**.

PK3Err0026: Download program exec failed.

If you receive this error while attempting to program from the Debugger menu:

1. Deselect the debugger as the debug tool.
2. Close your project and then close MPLAB IDE.
3. Restart MPLAB IDE and re-open your project.
4. Reselect the debugger as your debug tool and attempt to program your target device again.

If this does not work, see **Section 9.3.4 "Corrupted Installation Actions"**.

PK3Err0027: Bulk transfer failed due to invalid checksum

See **Section 9.3.3 "Debugger-to-PC Communication Error Actions"**.

Also, ensure that the cables used are the correct length.

PK3Err0028: Download device database failed

If you receive this error:

1. Try downloading again. It may be a one-time error.
2. Try manually downloading. Select *Debugger>Settings*, **Configuration** tab, and click **Manual Download**. Select the highest number `.jam` file and click **Open**.

© 2010 Microchip Technology Inc.

PK3Err0029: Communication failure. Unexpected command echo response %x received from PICkit 3.

See **Section 9.3.3 "Debugger-to-PC Communication Error Actions"**.

PK3Err0030: Unable to read/find firmware File %s.

If the Hex file exists:

- Reconnect and try again.
- If this does not work, the file may be corrupted. Reinstall MPLAB IDE.

If the Hex file does not exist:

- Reinstall MPLAB IDE.

PK3Err0031: Failed to get PC.

PK3Err0032: Failed to set PC.

See **Section 9.3.2 "Debugger-to-Target Communication Error Actions"**.

PK3Err0033: %d bytes expected, %d bytes received.

See **Section 9.3.3 "Debugger-to-PC Communication Error Actions"**.

PK3Err0034: This version of MPLAB IDE does not support hardware revision %06x. Please upgrade to the latest version of MPLAB IDE before continuing.

Find the latest MPLAB IDE at www.microchip.com.

PK3Err0035: Failed to get Device ID.

See **Section 9.3.1 "Read/Write Error Actions"**.

PK3Err0036: MPLAB IDE has lost communication with PICkit 3.

See **Section 9.3.3 "Debugger-to-PC Communication Error Actions"**.

PK3Err0037: Timed out waiting for response from PICkit 3.

PK3Err0038: Failed to initialize PICkit 3.

PK3Err0039: PICkit 3 self-test failed.

For this error, the debugger is not responding:

1. Unplug and plug in the debugger.
2. Reconnect to the debugger in MPLAB IDE.
3. If the problem persists contact Microchip.

PK3Err0040: The target device is not ready for debugging. Please check your Configuration bit settings and program the device before proceeding.

You will receive this message when you have not programmed your device for the first time and try to Run. If you receive this message after this, or immediately after programming your device, please refer to **Section 9.3.6 "Debug Failure Actions"**.

PK3Err0043: RS download failed.

PK3Err0044: AP download failed.

PK3Err0045: You must connect to a target device to use PICkit 3.

No power has been found.

1. Ensure V_{DD} and GND are connected between the debugger and target.
2. Ensure that the target is powered.
3. Ensure that the target power is sufficient to be detected by the debugger (see **Appendix A. "Hardware Specification"**.)

PK3Err0046: An error occurred while trying to read the stopwatch count. The stopwatch count may not be accurate.

See **Section 9.3.2 "Debugger-to-Target Communication Error Actions"**.

PK3Err0047: Bootloader download failed.

See **Section 9.3.3 "Debugger-to-PC Communication Error Actions"**.

PK3Err0052: The current PICkit 3 hardware version %x, is out of date. This version of MPLAB IDE will support only version %x or higher.

Did you click **Cancel** when asked to download the latest firmware? If so, you will need to download it now. Select *Debugger>Settings*, **Configuration** tab, and click **Manual Download**. Select the highest number `.jam` file and click **Open**.

If you cannot find any files to download or if this does not work (corrupted file), you will need to get the latest version of MPLAB IDE and install it. Find the latest MPLAB IDE at www.microchip.com.

PK3Err0053: Unable to get PICkit 3 protocol versions.

See **Section 9.3.3 "Debugger-to-PC Communication Error Actions"**.

PK3Err0054: MPLAB IDE's PICkit 3 protocol definitions are out of date. You must upgrade MPLAB IDE to continue.

Find the latest MPLAB IDE at www.microchip.com.

PK3Err0055: Unable to set firmware suite version.

PK3Err0056: Unable to get voltages from PICkit 3.

See **Section 9.3.3 "Debugger-to-PC Communication Error Actions"**.

PK3Err0068: Failed while writing to boot Flash memory.

PK3Err0069: Failed while reading boot Flash memory.

PK3Err0070: Failed while writing peripheral memory.

PK3Err0071: Failed while reading peripheral memory.

See **Section 9.3.1 "Read/Write Error Actions"**.

PK3Err0072: Unable to send freeze peripheral information.

PK3Err0073: Device is code-protected.

The device on which you are attempting to operate (read, program, blank check or verify) is code-protected, i.e., the code cannot be read or modified. Check your Configuration bits setting for code protection.

To disable code protection, set or clear the appropriate Configuration bits in code or in the Configuration Bits window (*Configure>Configuration Bits*), according to the device data sheet. Then erase and reprogram the *entire* device.

PK3Err0075: Unable to set power.

PICkit 3 cannot supply power to the target.

PK3Err0080: Failed setting shadow bit(s).

9.3 GENERAL CORRECTIVE ACTIONS

These general corrective actions may solve your problem:

- Read/Write Error Actions
- Debugger-to-Target Communication Error Actions
- Debugger-to-PC Communication Error Actions
- Corrupted Installation Actions
- USB Port Communication Error Actions
- Debug Failure Actions
- Internal Error Actions

© 2010 Microchip Technology Inc.

9.3.1 Read/Write Error Actions

If you receive a read or write error:

1. Did you hit Abort? This may produce read/write errors.
2. Try the action again. It may be a one-time error.
3. Ensure that the target is powered and at the correct voltage levels for the device. See the device data sheet for required device voltage levels.
4. Ensure that the debugger-to-target connection is correct (PGC and PGD are connected.)
5. For write failures, ensure that “Erase all before Program” is checked on the **Program Memory** tab of the Settings dialog.
6. Ensure that the cables used are of the correct length.

9.3.2 Debugger-to-Target Communication Error Actions

The PICkit 3 programmer/debugger and the target device are out-of-sync with each other.

1. Select **Reset** and then try the action again.
2. Ensure that the cable(s) used are the correct length.

9.3.3 Debugger-to-PC Communication Error Actions

The PICkit 3 programmer/debugger and MPLAB IDE are out-of-sync with each other.

1. Unplug and then plug in the debugger.
1. Reconnect to the debugger.
2. Try the operation again. It is possible the error was a one time glitch.
3. The version of MPLAB IDE installed may be incorrect for the version of firmware loaded on the PICkit 3 programmer/debugger. Follow the steps outlined in **Section 9.3.4 “Corrupted Installation Actions”**.

9.3.4 Corrupted Installation Actions

The problem is most likely caused by a incomplete or corrupted installation of MPLAB IDE.

1. Uninstall all versions of MPLAB IDE from the PC.
2. Reinstall the desired MPLAB IDE version.
3. If the problem persists contact Microchip.

9.3.5 USB Port Communication Error Actions

The problem is most likely caused by a faulty or non-existent communications port.

1. Reconnect to the PICkit 3 programmer/debugger
2. Make sure the debugger is physically connected to the PC on the appropriate USB port.
3. Make sure the appropriate USB port has been selected in the debugger Settings.
4. Make sure the USB port is not in use by another device.
5. If using a USB hub, make sure it is powered.
6. Make sure the USB drivers are loaded.

9.3.6 Debug Failure Actions

The PICkit 3 programmer/debugger was unable to perform a debugging operation. There are numerous reasons why this might occur. See **Section 7.3 “Top 10 Reasons Why You Can’t Debug”** and **Section 7.4 “Other Things to Consider”**.

9.3.7 Internal Error Actions

Internal errors are unexpected and should not happen. They are primarily used for internal Microchip development.

The most likely cause is a corrupted installation (**Section 9.3.4 "Corrupted Installation Actions"**).

Another likely cause is exhausted system resources.

1. Try rebooting your system to free up memory.
2. Make sure you have a reasonable amount of free space on your hard drive (and that it is not overly fragmented.)

If the problem persists contact Microchip.

© 2010 Microchip Technology Inc.

Part 3 – Reference

NOTES:

© 2010 Microchip Technology Inc.

Chapter 10. Debugger Function Summary

10.1 INTRODUCTION

A summary of the PICkit 3 programmer/debugger functions on MPLAB IDE menus, in windows and on dialogs is listed here.

- Debugging Functions
- Debugging Dialogs/Windows
- Programming Functions
- Settings Dialog

10.2 DEBUGGING FUNCTIONS

When you select the PICkit 3 from the Debugger menu, the following items will be added to the MPLAB IDE functions:

- Debugger Menu – additional options are added to the drop-down menu
- Right Mouse Button Debugger Menu – additional options are added to this menu
- Toolbars/Status Bar – a toolbar appears below the menu bar; additional information appears in the status bar

10.2.1 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*, 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

Execute the instruction at the current program counter location. At a `CALL` instruction, Step Over executes the called subroutine and halts at the address following the `CALL`. If the Step Over is too long or appears to "hang", click Halt.

Step Out

Not available.

Reset > Processor Reset F6

Issue a Reset sequence to the target processor. This issues a $\overline{\text{MCLR}}$ to reset the Program Counter to the Reset vector.

Breakpoints

Open the Breakpoint dialog (see **Section 10.3.1 "Breakpoints Dialog"**). Use hardware breakpoints to halt code execution at specified lines in your code. Set multiple breakpoints in this dialog.

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

Program

Download your code to the target device.

Read

Read target memory. Information uploaded to MPLAB IDE.

Erase Flash Device

Erase all Flash memory.

Abort Operation

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

Reconnect

Attempt to re-establish communications between the PC and the PICkit 3 programmer/debugger. The progress of this connection is shown on the PICkit 3 tab of the Output dialog.

Settings

Open the PICkit 3 Settings dialog (see **Section 10.5 "Settings Dialog"**). Set up program and firmware options.

© 2010 Microchip Technology Inc.

10.2.2 Right Mouse Button Debugger Menu

These debugger menu options will appear on the right mouse menus in code displays, such as program memory and source code files. Descriptions of other menu options not listed here can be found in the MPLAB IDE Help or the MPLAB Editor Help.

Set Breakpoint

Set or remove 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.

Center Debug Location

Center the current PC line in the window.

10.2.3 Toolbars/Status Bar

When the PICkit 3 programmer/debugger is selected as a debugger, these toolbars are displayed in MPLAB IDE:

- Basic debug toolbar (Run, Halt, Animate, Step Into, Step Over, Step Out, Reset, Breakpoints).
- Simple program toolbar (Program, Read, Erase Flash Device).

The selected debug tool (PICkit 3), as well as other development information, is displayed in the status bar on the bottom of the MPLAB IDE desktop. Refer to the MPLAB IDE online help for information on the contents of the status bar.

10.3 DEBUGGING DIALOGS/WINDOWS

Open the following debug dialogs and windows using the menu items mentioned in **Section 10.2 “Debugging Functions”**.

- Breakpoints Dialog
  - Set Breakpoint Dialog
  - Stopwatch Dialog
  - Event Breakpoints Dialog

10.3.1 Breakpoints Dialog

To set up breakpoints, select *Debugger>Breakpoints*.

Set up different types of breakpoints in this dialog. Click on **Add Breakpoint** to add breakpoints to the dialog window. Depending on your selected device, there may be other buttons for more advanced breakpoint options.

10.3.1.1 BREAKPOINT DIALOG WINDOW

Information about each breakpoint is visible in this window.

TABLE 10-1: BREAKPOINT DIALOG WINDOW

Control	Function
Breakpoint Type	Type of breakpoint – program or data
Address	Hex address of breakpoint location
File Line #/Symbol Name	File name and line number of breakpoint location
Enabled	Check to enable a breakpoint

Once a breakpoint has been added to the window, you may right click on it to open a menu of options:

- Delete – delete selected breakpoint
- Edit/View – open the Set Breakpoint Dialog
- Delete All – delete all listed breakpoints
- Disable All – disable all listed breakpoints

10.3.1.2 BREAKPOINT DIALOG BUTTONS

Use the buttons to add a breakpoint and set up additional break conditions. Also, a stopwatch is available for use with breakpoints and triggers.

Note: Buttons displayed will depend on the selected device.

TABLE 10-2: BREAKPOINT DIALOG BUTTONS

Control	Function	Related Dialog
Add Breakpoint	Add a breakpoint	**Section 10.3.2 "Set Breakpoint Dialog"**
Stopwatch	Set up the stopwatch	**Section 10.3.3 "Stopwatch Dialog"**
Event Breakpoints	Set up break on an event	**Section 10.3.4 "Event Breakpoints Dialog"**

© 2010 Microchip Technology Inc.

10.3.2 Set Breakpoint Dialog

Click **Add Breakpoint** in the Breakpoints Dialog to display this dialog.

Select a breakpoint for the Breakpoints dialog here.

10.3.2.1 PROGRAM MEMORY TAB

Set up a program memory breakpoint here.

TABLE 10-3: PROGRAM MEMORY BREAKPOINT

Control	Function
Address	Location of breakpoint in hex
Breakpoint Type	The type of program memory breakpoint. See the device data sheet for more information on table reads/writes. *Program Memory Execution* – break on execution of above address *TBLRD Program Memory* – break on table read of above address *TBLWT Program Memory* – break on table write to above address
Pass Count	Break on pass count condition. *Always break* – always break as specified in “Breakpoint type” *Break occurs Count instructions after Event* – wait Count (0-255) instructions before breaking after event specified in “Breakpoint type” *Event must occur Count times* – break only after event specified in “Breakpoint type” occurs Count (0-255) times

10.3.2.2 DATA MEMORY TAB

Set up a data memory breakpoint here.

TABLE 10-4: DATA MEMORY BREAKPOINT

Control	Function
Address	Location of breakpoint in hex
Breakpoint Type	The type of data memory breakpoint. See the device data sheet for more information on X Bus reads/writes. *X Bus Read* – break on an X bus read of above address *X Bus Read Specific Byte* – break on an X bus read of above address for the specific byte value in “Specific Value” *X Bus Read Specific Word* – break on an X bus read of above address for the specific word value in “Specific Value” *X Bus Write* – break on an X bus write of above address *X Bus Write Specific Byte* – break on an X bus write of above address for the specific byte value in “Specific Value” *X Bus Write Specific Word* – break on an X bus write of above address for the specific word value in “Specific Value”
Pass Count	Break on pass count condition. *Always break* – always break as specified in “Breakpoint type” *Break occurs Count instructions after Event* – wait Count (0-255) instructions before breaking after event specified in “Breakpoint type” *Event must occur Count times* – break only after event specified in “Breakpoint type” occurs Count (0-255) times

10.3.3 Stopwatch Dialog

Click **Stopwatch** in the Breakpoints Dialog to display the Stopwatch Setup dialog.

Use the stopwatch with breakpoints to time code execution. The stopwatch allows timing from one breakpoint/trigger condition to the next. The stopwatch value is in decimal.

TABLE 10-5: STOPWATCH SETUP

Control	Function
Start Condition	Select an available breakpoint or trigger condition to start the stopwatch. Available breakpoints/triggers are those previously added to the breakpoint dialog. Select **None** to clear the start condition. To halt the program run on this condition, check the check box next to "Start condition will cause the target device to halt".
Stop Condition	Select an available breakpoint or trigger condition to stop the stopwatch. Available breakpoints/triggers are those previously added to the breakpoint dialog. Select **None** to clear the stop condition. To halt the program run on this condition, check the check box next to "Stop condition will cause the target device to halt".
Reset stopwatch on run	Reset the stopwatch values to zero every time the program is run.

To set a breakpoint in code, do one of the following:

- Double click or right click on a line of code to set up an individual breakpoint.
- Select *Debugger>Breakpoints* to open the Breakpoints dialog and set up multiple breakpoints and breakpoint conditions. See **Section 10.3.1 "Breakpoints Dialog"** for more information.

To determine the time between the breakpoints, use the stopwatch:

1. Open the Breakpoints dialog (*Debugger>Breakpoints*).
1. Click **Stopwatch** on the Breakpoints dialog to open the Stopwatch dialog.
2. Under "Start Condition", select a breakpoint from the drop-down list. Also decide if "Start condition will cause the target device to halt".
3. Under "Stop Condition", select another breakpoint from the drop-down list. Also decide if "Stop condition will cause the target device to halt".
4. Decide if there will be a "Reset stopwatch on run".
5. Click **OK**.

10.3.4 Event Breakpoints Dialog

Click **Event Breakpoints** in the Breakpoints Dialog to display this dialog.

Select a condition where the program will always break:

- Break on Watchdog Timer – Break every time the Watchdog Timer times out. Make sure the Watchdog Timer is enabled in the Configuration bits.
- Break on `SLEEP` instruction – Break when a `SLEEP` instruction is encountered in the program.

© 2010 Microchip Technology Inc.

10.4 PROGRAMMING FUNCTIONS

When you select the PICkit 3 programmer/debugger from the Programmer menu, program items will be added to the following MPLAB IDE functions:

- Programmer Menu
- Toolbars/Status Bar

10.4.1 Programmer Menu

Program

Program specified memory areas: program memory, Configuration bits, ID locations and/or EEPROM data. See the Settings dialog for programming options.

Verify

Verify programming of 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. See the Settings dialog for read options.

Blank Check All

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

Erase Flash Device

Erase all Flash memory.

Abort Operation

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

Reconnect

Attempt to re-establish communications between the PC and the PICkit 3 programmer/debugger. The progress of this connection is shown on the PICkit 3 tab of the Output dialog.

Settings

Open the Programmer dialog (see **Section 10.5 “Settings Dialog”**). Set up program and firmware options.

10.4.2 Toolbars/Status Bar

When the PICkit 3 programmer/debugger is selected as a programmer, these toolbars are displayed in MPLAB IDE:

- Basic program toolbar (Program, Read, Verify, Erase Flash Device, Blank Check All).

The selected programmer (PICkit 3), as well as other programming information, is displayed in the status bar on the bottom of the MPLAB IDE desktop. Refer to the MPLAB IDE online help for information on the contents of the status bar.

10.5 SETTINGS DIALOG

Select either *Debugger>Settings* or *Programmer>Settings* to open the Settings dialog and set up the PICkit 3 programmer/debugger.

Note: The tabs displayed will depend on the selected device and whether you have the PICkit 3 selected as a debugger or a programmer.

- Settings Dialog, Program Memory Tab
- Settings Dialog, Configuration Tab
- Settings Dialog, Freeze on Halt Tab
- Settings Dialog, Status Tab
- Settings Dialog, Clock Tab
- Settings Dialog, Secure Segment Tab
- Settings Dialog, Warnings Tab
- Settings Dialog, Power Tab
- Settings Dialog, Limitations Tab
- Settings Dialog, Programmer-to-go Tab

10.5.1 Settings Dialog, Program Memory Tab

This tab allows you to set up debug/programming options.

TABLE 10-6: PROGRAM MEMORY OPTIONS

Allow PICkit 3 to select memories and ranges	If selected, the PICkit™ 3 uses your selected device and default settings to determine what to program.
Manually select memories and ranges	If selected, you must determine the Memories, Program Memory Range, and Program Options for the selected device.
Memories:	
Program	Check to program Program Memory into target.
Configuration	Check to program Configuration bits into target. **Note:** This memory is always programmed when debugger selected as a debugger.
EEPROM	Check to erase and then program EEPROM memory on target, if available. Uncheck to erase EEPROM memory on target.
ID	Check to program ID Memory into target.
Boot Flash	If supported, check to program boot memory on target
Program Memory Range:	
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, you need to 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.
Program Options:	
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.

© 2010 Microchip Technology Inc.

TABLE 10-6: PROGRAM MEMORY OPTIONS (CONTINUED)

Preserve Program Memory Range	Check to preserve memory range. Start, End – The starting and ending hex address range in program memory for preservation.
Use high voltage on MCLR	*For PIC24FXXKAXXX devices:* Check this option to use high voltage to configure pin as either a normal input pin or as a $\overline{\text{MCLR}}$ Reset. Leave unchecked for a low-voltage $\overline{\text{MCLR}}$ Reset. **Notes:** 1. As long as the configuration setting is “MCLR pin enabled; RA5 input pin disabled”, then you can use Low Voltage Entry (uncheck the “Use high voltage on MCLR” option). 2. If you have the configuration setting to “RA5 input pin enabled; MCLR disabled”, then you must check the “Use high voltage on MCLR” option. 3. If you want to program the “MCLR pin enable bit” configuration setting, you must check the “Use high voltage on MCLR” option. 4. If you are using a -ICE header, the setting of this checkbox does not matter.
Automatically	
Program after successful build	After the application code successfully builds, program this code into the device.
Run after successful program	*This option is available only in Debug mode.* After the application code is successfully programmed into the target device, run the code.

10.5.2 Settings Dialog, Configuration Tab

Configure debugger operation on this tab.

TABLE 10-7: CONFIGURATION ITEMS

Download Firmware	Set up firmware download options.
Auto Download Latest Firmware	Check to allow automatic download of the latest version of firmware for the target device (**recommended**).
Manual Download	Manually select a firmware file to download to the target device.

10.5.3 Settings Dialog, Freeze on Halt Tab

Select peripherals to freeze on halt on this tab.

PIC12/16 MCU Devices

To freeze/unfreeze all device peripherals on halt, check/uncheck the “Freeze on Halt” checkbox. If this does not halt your desired peripheral, be aware that some peripherals have no freeze on halt capability and cannot be controlled by the emulator.

There may be device-specific limitations for peripherals that do support freeze. Click the **Limitations** button to find these.

PIC18 MCU Devices

To freeze/unfreeze all device peripherals on halt, check/uncheck the “Freeze on Halt” checkbox. If this does not halt your desired peripheral, be aware that some peripherals have no freeze on halt capability and cannot be controlled by the debugger.

dsPIC30F/33F, PIC24F/H and PIC32MX Devices

For peripherals in the list "Peripherals to Freeze on Halt", check to freeze that peripheral on a halt. Uncheck the peripheral to let it run while the program is halted. If you do not see a peripheral on the list, check "All Other Peripherals". If this does not halt your desired peripheral, be aware that some peripherals have no freeze on halt capability and cannot be controlled by the debugger.

To select all peripherals, including "All Other Peripherals", click **Check All**. To deselect all peripherals, including "All Other Peripherals", click **Uncheck All**.

10.5.4 Settings Dialog, Status Tab

View the status of your PICkit 3 system on this tab.

TABLE 10-8: STATUS ITEMS

Versions	
Firmware Suite Version	Debugger firmware suite version. The firmware suite consists of the three items specified below.
Algorithm Plug-in Version	Debugger algorithm plug-in version. For your selected device, an algorithm is used to support the device plugged into the target.
OS Version	Debugger operating system version.
Voltages	
PICkit™ 3 V_{PP}	Debugger V_{PP}.
PICkit 3 V_{DD}	Debugger V_{DD}.
Refresh Voltages	Sensing of Status tab items occurs when the tab is activated. To see updates, click this button.

© 2010 Microchip Technology Inc.

10.5.5 Settings Dialog, Clock Tab

The Debug Clock mode is selected on this tab. Check the box for “FRC in Debug mode” to use the fast RC clock while debugging.

Device emulation occurs at the speed of the processor. If it is running at low frequency (for example 32 kHz), all operations such as stepping, refreshing the Watch window, etc. are slow. However, some processors have a built-in Fast RC oscillator. If the “FRC in Debug mode” option is checked, after the application running at normal speed halts, it switches to the faster oscillator enabling emulation and peripherals (if not frozen) to run faster.

10.5.6 Settings Dialog, Secure Segment Tab

For CodeGuard™ Security devices, set up secure segment properties on this tab.

For more details on CodeGuard Security functionality, please refer to the CodeGuard Security reference manual for 16-bit devices (DS70180) and dsPIC33F/PIC24H and dsPIC30F device programming specifications found on our web site.

TABLE 10-9: SECURE SEGMENT OPTIONS

Full Chip Programming	Click to select to program all program memory segments.
Segment Programming	Click to select segment programming. Select from: - Boot, secure and general segments - Secure and general segments - General segment only

10.5.7 Settings Dialog, Warnings Tab

A list of all PICkit 3 programmer/debugger warnings are displayed on this tab.

- Check a warning to enable it. The warning will be displayed in the Output window.
- Uncheck a warning to disable it.

Warnings are not errors and will not stop your project from building. If you receive error messages, please see **Chapter 9. “Error Messages”**.

10.5.8 Settings Dialog, Power Tab

Set up the power options on this tab.

- Click in the checkbox to enable/disable “Power target circuit from PICkit 3”.

  If you enable (check) this option, the Power on/off button will be enabled on the toolbar. It will initially be in the Power On state. Every time it is clicked it will toggle to the opposite state. If it is on, it will toggle to off, and if it is off it will toggle to on. If the Power target circuit setting is disabled (unchecked) the Power on/off button will go back to the disabled state.

  Whatever state it is in when the workspace was last saved will be the state that it is in when the workspace is reopened.
- Adjust the slide bar to select the voltage. The value in the field changes according to your adjustments.

10.5.9 Settings Dialog, Limitations Tab

This tab displays general limitations for the selected device. For specific limitation information, click **Details**.

10.5.10 Settings Dialog, Programmer-to-go Tab

This tab enables you to name the image of the programming and send to memory in the PICkit 3.

TABLE 10-10: PROGRAMMER-TO-GO OPTIONS

Image Name	Type the name of the programming image.
Programmer Setting Report	Displays the settings selected for programming the device. If you want to change any of these settings, use the appropriate tab in the PICkit 3 Settings dialog.
Send Image In Memory	Executes an image transfer from MPLAB IDE to the PICkit 3 unit.

© 2010 Microchip Technology Inc.

Appendix A. Hardware Specification

A.1 INTRODUCTION

The hardware and electrical specifications of the PICkit 3 programmer/debugger system are detailed.

A.2 HIGHLIGHTS

This chapter discusses:

- Declaration of Conformity
- USB Port/Power
- PICkit 3 Programmer/Debugger
- Standard Communication Hardware
- Target Board Considerations

A.3 DECLARATION OF CONFORMITY

We

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

hereby declare that the product:

PICkit 3 Programmer/Debugger

complies with the following standards, provided that the restrictions stated in the operating manual are observed:

Standards: EN61010-1 Laboratory Equipment
Microchip Technology, Inc.
Date: January 2009

Important Information Concerning the Use of the PICkit 3 programmer/Debugger

Due to the special nature of the PICkit 3 programmer/debugger, the user is advised that it can generate higher than normal levels of electromagnetic radiation which can interfere with the operation of all kinds of radio and other equipment.

To comply with the European Approval Regulations therefore, the following restrictions must be observed:

1. The development system must be used only in an industrial (or comparable) area.
2. The system must not be operated within 20 meters of any equipment which may be affected by such emissions (radio receivers, TVs etc.).

A.4 USB PORT/POWER

The PICkit 3 programmer/debugger is connected to the host PC via a mini Universal Serial Bus (USB) port, version 2.0 compliant. The USB connector is located on the top of the pod.

The system is capable of reloading the firmware via the USB interface.

System power is derived from the USB interface. The debugger is classified as a high power system per the USB specification, and requires slightly more than 100 mA of power from the USB to function in all operational modes (debugger/programmer).

> **Note:** The PICkit 3 programmer/debugger is powered through its USB connection. The target board is powered from its own supply. Alternatively, the PICkit 3 can power it only if the target consumes less than 30 mA.

Cable Length – The PC-to-debugger cable length for proper operation is shipped in the debugger kit.

Powered Hubs – If you are going to use a USB hub, make sure it is self-powered. Also, USB ports on PC keyboards do not have enough power for the debugger to operate.

PC Hibernate/Power-Down Modes – Disable the hibernate or other power saver modes on your PC to ensure proper USB communications with the debugger.

A.5 PICKIT 3 PROGRAMMER/DEBUGGER

The debugger consists of a main board enclosed in the casing with a USB connector and a single in-line connector. On the debugger enclosure are indicator lights (LEDs).

A.5.1 Main Board

This component has the interface processor with integrated USB 2.0 interface capable an SPI, serial EE for programming into the emulation device on-board Flash and LED indicators.

A.5.2 Indicator Lights (LEDs)

The indicator lights have the following significance.

LED	Color	Description
Power	Green	Lit when power is first applied or when target is connected.
Active	Blue	Lit when the PICkit™ 3 has established communication with the PC or sending/receiving commands.
Status	Green	Lit when the debugger is operating normally – standby.
	Yellow	Lit when an operation is busy.
	Red	Lit when the debugger has failed.

© 2010 Microchip Technology Inc.

A.6 STANDARD COMMUNICATION HARDWARE

For standard debugger communication with a target (**Section 2.3 "Debugger to Target Communication"**, "Standard ICSP Device Communication"), use an adapter with an RJ-11 connector.

To use this type of communication with a header board, you may need a device-specific Processor Pak, which includes an 8-pin connector header board containing the desired ICE/ICD device and a standard adapter board.

> **Note:** Older header boards used a 6-pin (RJ-11) connector instead of an 8-pin connector, so these headers may be connected directly to the debugger.

For more on available header boards, see the "*Header Board Specification*" (DS51292).

A.6.1 Standard Communication

The standard communication is the main interface to the target processor. It contains the connections to the high voltage (V_{PP}), V_{DD} sense lines, and clock and data connections required for programming and connecting with the target devices.

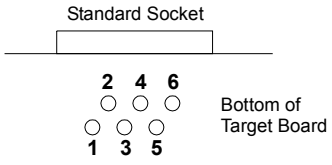
The V_{PP} high-voltage lines can produce a variable voltage that can swing from 1.8 to 14 volts to satisfy the voltage requirements for the specific emulation processor.

The V_{DD} sense connection draws current from the target processor.

The clock and data connections are interfaces with the following characteristics:

- Clock and data signals are in High-Impedance mode (even when no power is applied to the PICkit 3 programmer/debugger system)
- Clock and data signals are protected from high voltages caused by faulty targets systems, or improper connections
- Clock and data signals are protected from high current caused from electrical shorts in prototype or target systems

FIGURE A-1: 6-PIN STANDARD PINOUT



Pin	Name	Function
1	$\overline{MCLR}/V_{PP}$	Power
2	V_{DD}_TGT	Power on target
3	GND	Ground
4	PGD (ICSPDAT)	Standard Com Data
5	PGC (ICSPCLK)	Standard Com Clock
6	PGM (LVP)	Low Voltage Programming

A.6.2 Modular Cable and Connector

For standard communications, a modular cable connects the debugger and the target application. The specifications for this cable and its connectors are listed below.

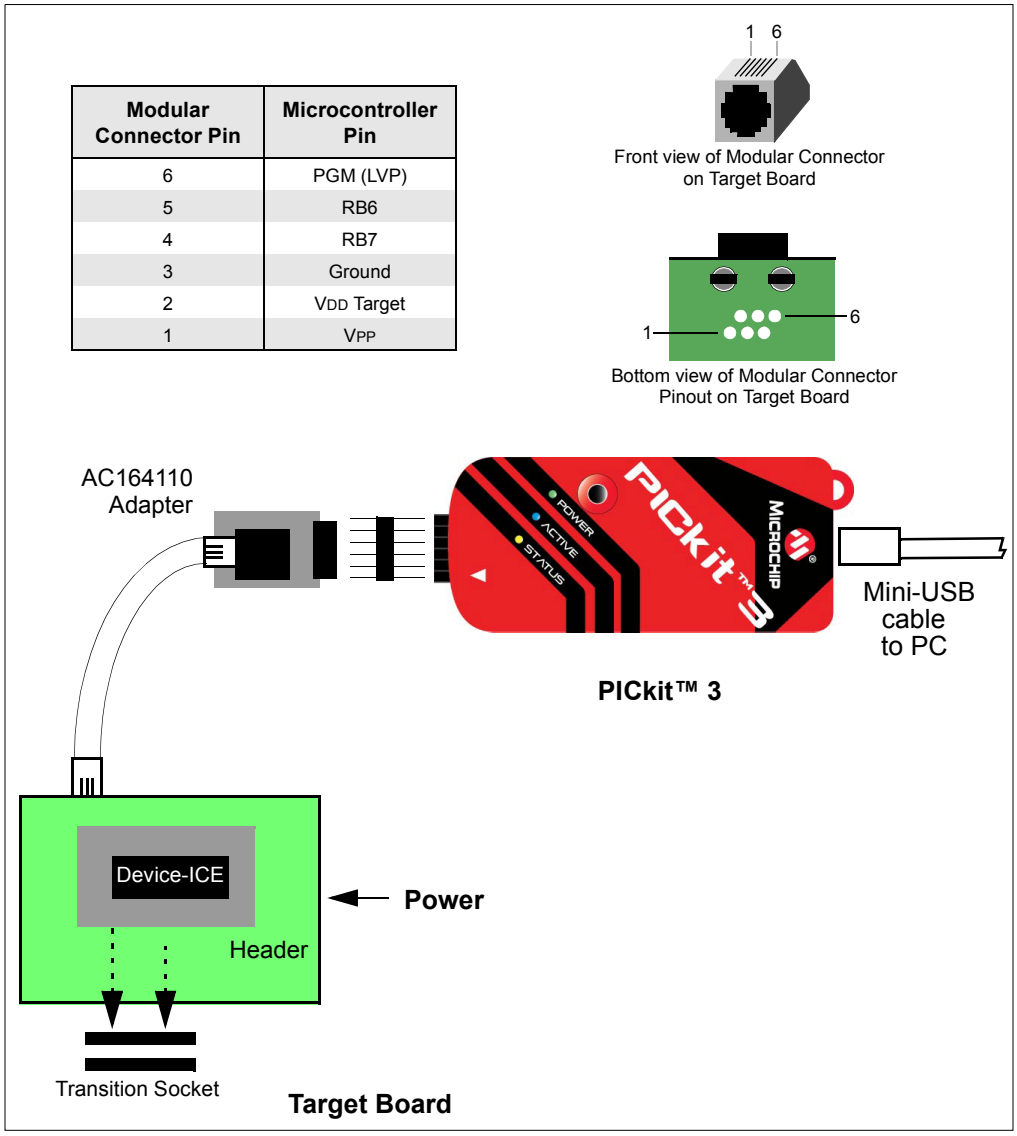
A.6.2.1 MODULAR CONNECTOR SPECIFICATION

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

The table within Figure A-2 shows how the modular connector pins on an application correspond to the microcontroller pins. This configuration provides full ICD functionality.

FIGURE A-2: MODULAR CONNECTOR PINOUT OF TARGET BOARD

Modular Connector Pin	Microcontroller Pin
6	PGM (LVP)
5	RB6
4	RB7
3	Ground
2	VDD Target
1	VPP



A.6.2.2 MODULAR PLUG SPECIFICATION

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

© 2010 Microchip Technology Inc.

A.6.2.3 MODULAR CABLE SPECIFICATION

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

A.7 TARGET BOARD CONSIDERATIONS

The target board should be powered according to the requirements of the selected device (1.8V-5.0V) and the application.

Depending on the type of debugger-to-target communications used, there will be some considerations for target board circuitry:

- **Section 2.4.2 “Target Connection Circuitry”**
- **Section 2.4.5 “Circuits That Will Prevent the Debugger From Functioning”**

NOTES:

© 2010 Microchip Technology Inc.

Appendix B. PICkit 3 Schematics

PICkit 3 Development Programmer/Debugger schematic diagrams are shown here. Demo board schematics are found in their respective user's guides.

FIGURE B-1: PICkit™ 3 SCHEMATIC DIAGRAM (PAGE 1 OF 2)

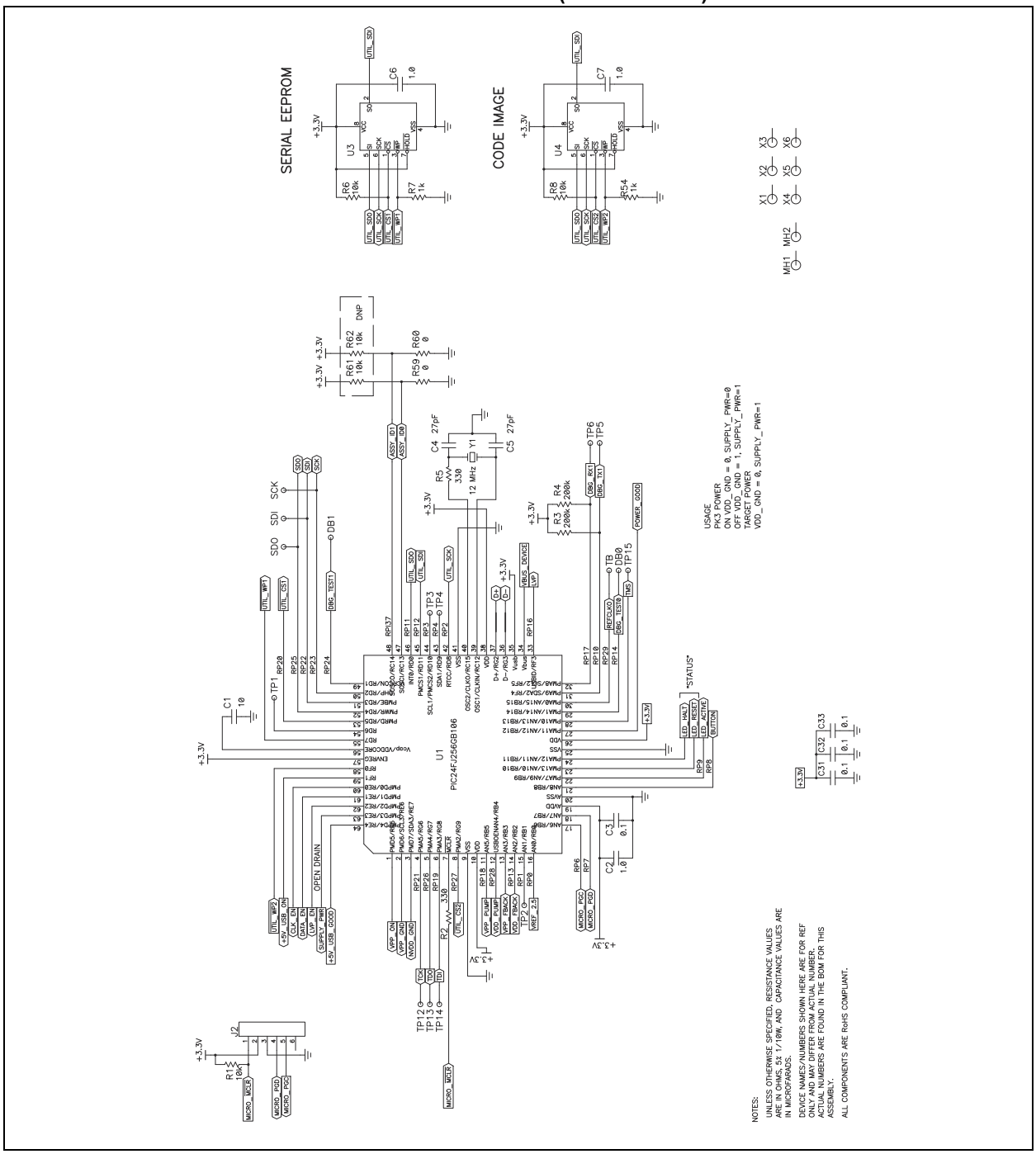
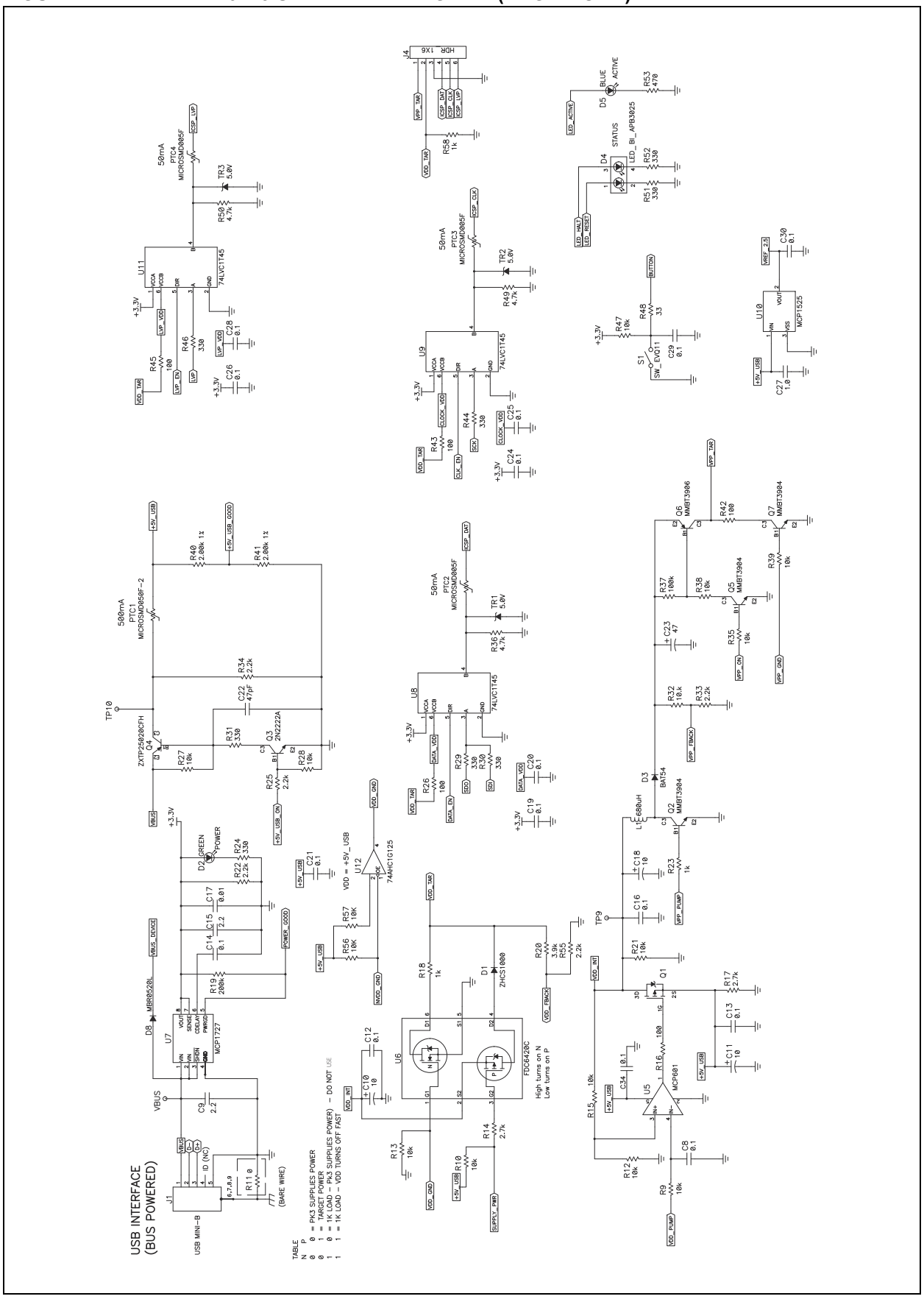


FIGURE B-2: PICkit™ 3 SCHEMATIC DIAGRAM (PAGE 2 OF 2)



© 2010 Microchip Technology Inc.

Glossary

Absolute Section

A section with a fixed (absolute) address that cannot be changed by the linker.

Access Memory

PIC18 Only – Special registers on PIC18 devices that allow access regardless of the setting of the Bank Select Register (BSR).

Access Entry Points

Access entry points provide a way to transfer control across segments to a function which may not be defined at link time. They support the separate linking of boot and secure application segments.

Address

Value that identifies a location in memory.

Alphabetic Character

Alphabetic characters are those characters that are letters of the arabic alphabet (a, b, …, z, A, B, …, Z).

Alphanumeric

Alphanumeric characters are comprised of alphabetic characters and decimal digits (0,1, …, 9).

ANDed Breakpoints

Set up an ANDed condition for breaking, i.e., breakpoint 1 AND breakpoint 2 must occur at the same time before a program halt. This can only be accomplished if a data breakpoint and a program memory breakpoint occur at the same time.

Anonymous Structure

C30 **–** An unnamed structure.

C18 **–** An unnamed structure that is a member of a C union. The members of an anonymous structure may be accessed as if they were members of the enclosing union. For example, in the following code, `hi` and `lo` are members of an anonymous structure inside the union `caster`.

```
union castaway
 int intval;
 struct {
  char lo; //accessible as caster.lo
  char hi; //accessible as caster.hi
 };
} caster;
```

ANSI

American National Standards Institute is an organization responsible for formulating and approving standards in the United States.

Application

A set of software and hardware that may be controlled by a PIC® microcontroller.

Archive

A collection of relocatable object modules. It is created by assembling multiple source files to object files, and then using the archiver to combine the object files into one library file. A library can be linked with object modules and other libraries to create executable code.

Archiver

A tool that creates and manipulates libraries.

ASCII

American Standard Code for Information Interchange is a character set encoding that uses 7 binary digits to represent each character. It includes upper and lowercase letters, digits, symbols and control characters.

Assembler

A language tool that translates assembly language source code into machine code.

Assembly Language

A programming language that describes binary machine code in a symbolic form.

Assigned Section

A section which has been assigned to a target memory block in the linker command file.

Asynchronously

Multiple events that do not occur at the same time. This is generally used to refer to interrupts that may occur at any time during processor execution.

Asynchronous Stimulus

Data generated to simulate external inputs to a simulator device.

Attribute

Characteristics of variables or functions in a C program which are used to describe machine-specific properties.

Attribute, Section

Characteristics of sections, such as "executable", "readonly", or "data" that can be specified as flags in the assembler `.section` directive.

Binary

The base two numbering system that uses the digits 0-1. The rightmost digit counts ones, the next counts multiples of 2, then $2^2 = 4$, etc.

Bookmarks

Use bookmarks to easily locate specific lines in a file.

Under the Edit menu, select Bookmarks to manage bookmarks. Toggle (enable/disable) a bookmark, move to the next or previous bookmark, or clear all bookmarks.

Breakpoint

Hardware Breakpoint: An event whose execution will cause a Halt.

Software Breakpoint: An address where execution of the firmware will halt. Usually achieved by a special break instruction.

Build

Compile and link all the source files for an application.

C

A general-purpose programming language which features economy of expression, modern control flow and data structures, and a rich set of operators.

© 2010 Microchip Technology Inc.

Calibration Memory

A Special Function Register or registers used to hold values for calibration of a PIC microcontroller on-board RC oscillator or other device peripherals.

Central Processing Unit

The part of a device that is responsible for fetching the correct instruction for execution, decoding that instruction, and then executing that instruction. When necessary, it works in conjunction with the arithmetic logic unit (ALU) to complete the execution of the instruction. It controls the program memory address bus, the data memory address bus, and accesses to the stack.

Clean

Under the MPLAB IDE Project menu, Clean removes all intermediary project files, such as object, hex and debug files, for the active project. These files are recreated from other files when a project is built.

COFF

Common Object File Format. An object file of this format contains machine code, debugging and other information.

Command Line Interface

A means of communication between a program and its user based solely on textual input and output.

Compiler

A program that translates a source file written in a high-level language into machine code.

Conditional Assembly

Assembly language code that is included or omitted based on the assembly-time value of a specified expression.

Conditional Compilation

The act of compiling a program fragment only if a certain constant expression, specified by a preprocessor directive, is true.

Configuration Bits

Special-purpose bits programmed to set PIC microcontroller modes of operation. A Configuration bit may or may not be preprogrammed.

Control Directives

Directives in assembly language code that cause code to be included or omitted based on the assembly-time value of a specified expression.

CPU

See Central Processing Unit.

Cross Reference File

A file that references a table of symbols and a list of files that references the symbol. If the symbol is defined, the first file listed is the location of the definition. The remaining files contain references to the symbol.

Data Directives

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

Data Memory

On Microchip MCU and DSC devices, data memory (RAM) is comprised of General Purpose Registers (GPRs) and Special Function Registers (SFRs). Some devices also have EEPROM data memory.

Debugger

Hardware that performs debugging.

Debugger System

The debugger systems include the pod, processor module, device adapter, target board, cables, and MPLAB IDE software.

Debugging Information

Compiler and assembler options that, when selected, provide varying degrees of information used to debug application code. See compiler or assembler documentation for details on selecting debug options.

Deprecated Features

Features that are still supported for legacy reasons, but will eventually be phased out and no longer used.

Device Programmer

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

Digital Signal Controller

A microcontroller device with digital signal processing capability, i.e., Microchip dsPIC DSC devices.

Digital Signal Processing

The computer manipulation of digital signals, commonly analog signals (sound or image) which have been converted to digital form (sampled).

Digital Signal Processor

A microprocessor that is designed for use in digital signal processing.

Directives

Statements in source code that provide control of the language tool's operation.

Download

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

DSC

See Digital Signal Controller.

DSP

See Digital Signal Processor.

dsPIC DSCs

dsPIC Digital Signal Controllers (DSCs) refers to all Microchip DSC families.

DWARF

Debug With Arbitrary Record Format. DWARF is a debug information format for ELF files.

EEPROM

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

© 2010 Microchip Technology Inc.

ELF

Executable and Linking Format. An object file of this format contains machine code. Debugging and other information is specified in with DWARF. ELF/DWARF provide better debugging of optimized code than COFF.

Emulation

The process of executing software loaded into emulation memory as if it were firmware residing on a microcontroller device.

Emulation Memory

Program memory contained within the emulator.

Emulator

Hardware that performs emulation.

Emulator System

The MPLAB ICE 2000 and MPLAB ICE 4000 emulator systems include the pod, processor module, device adapter, target board, cables, and MPLAB IDE software. The MPLAB REAL ICE system consists of a pod, a driver (and potentially a receiver) card, target board, cables, and MPLAB IDE software.

Endianness

The ordering of bytes in a multi-byte object.

Environment

IDE **–** The particular layout of the desktop for application development.

MPLAB PM3 **–** A folder containing files on how to program a device. This folder can be transferred to a SD/MMC card.

Epilogue

A portion of compiler-generated code that is responsible for deallocating stack space, restoring registers and performing any other machine-specific requirement specified in the runtime model. This code executes after any user code for a given function, immediately prior to the function return.

EPROM

Erasable Programmable Read Only Memory. A programmable read-only memory that can be erased usually by exposure to ultraviolet radiation.

Error File

A file containing error messages and diagnostics generated by a language tool.

Errors

Errors report problems that make it impossible to continue processing your program. When possible, errors identify the source file name and line number where the problem is apparent.

Event

A description of a bus cycle which may include address, data, pass count, external input, cycle type (fetch, R/W), and time stamp. Events are used to describe triggers, breakpoints and interrupts.

Executable Code

Software that is ready to be loaded for execution.

Export

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

Expressions

Combinations of constants and/or symbols separated by arithmetic or logical operators.

Extended Microcontroller Mode

In extended microcontroller mode, on-chip program memory as well as external memory is available. Execution automatically switches to external if the program memory address is greater than the internal memory space of the PIC18 device.

Extended Mode

In Extended mode, the compiler will utilize the extended instructions (i.e., `ADDFSR`, `ADDULNK`, `CALLW`, `MOVSF`, `MOVSS`, `PUSHL`, `SUBFSR` and `SUBULNK`) and the indexed with literal offset addressing.

External Label

A label that has external linkage.

External Linkage

A function or variable has external linkage if it can be referenced from outside the module in which it is defined.

External Symbol

A symbol for an identifier which has external linkage. This may be a reference or a definition.

External Symbol Resolution

A process performed by the linker in which external symbol definitions from all input modules are collected in an attempt to resolve all external symbol references. Any external symbol references which do not have a corresponding definition cause a linker error to be reported.

External Input Line

An external input signal logic probe line (TRIGIN) for setting an event based upon external signals.

External RAM

Off-chip Read/Write memory.

Fatal Error

An error that will halt compilation immediately. No further messages will be produced.

File Registers

On-chip data memory, including General Purpose Registers (GPRs) and Special Function Registers (SFRs).

Filter

Determine by selection what data is included/excluded in a trace display or data file.

Flash

A type of EEPROM where data is written or erased in blocks instead of bytes.

FNOP

Forced No Operation. A forced `NOP` cycle is the second cycle of a two-cycle instruction. Since the PIC microcontroller architecture is pipelined, it prefetches the next instruction in the physical address space while it is executing the current instruction. However, if the current instruction changes the program counter, this prefetched instruction is explicitly ignored, causing a forced `NOP` cycle.

© 2010 Microchip Technology Inc.

Frame Pointer

A pointer that references the location on the stack that separates the stack-based arguments from the stack-based local variables. Provides a convenient base from which to access local variables and other values for the current function.

Free-Standing

An implementation that accepts any strictly conforming program that does not use complex types and in which the use of the features specified in the library clause (ANSI '89 standard clause 7) is confined to the contents of the standard headers `<float.h>`, `<iso646.h>`, `<limits.h>`, `<stdarg.h>`, `<stdbool.h>`, `<stddef.h>` and `<stdint.h>`.

GPR

General Purpose Register. The portion of device data memory (RAM) available for general use.

Halt

A stop of program execution. Executing Halt is the same as stopping at a breakpoint.

Heap

An area of memory used for dynamic memory allocation where blocks of memory are allocated and freed in an arbitrary order determined at runtime.

Hex Code

Executable instructions stored in a hexadecimal format code. Hex code is contained in a hex file.

Hex File

An ASCII file containing hexadecimal addresses and values (hex code) suitable for programming a device.

Hexadecimal

The base 16 numbering system that uses the digits 0-9 plus the letters A-F (or a-f). The digits A-F represent hexadecimal digits with values of (decimal) 10 to 15. The rightmost digit counts ones, the next counts multiples of 16, then 16^2 = 256, etc.

High Level Language

A language for writing programs that is further removed from the processor than assembly.

ICD

In-Circuit Debugger. MPLAB ICD and PICkit (with Debug Express), are Microchip's in-circuit debuggers.

ICE

In-Circuit Emulator. MPLAB ICE 2000 and MPLAB ICE 4000 system are Microchip's classic in-circuit emulators. MPLAB REAL ICE system is Microchip's next-generation in-circuit emulator.

ICSP™

In-Circuit Serial Programming. A method of programming Microchip embedded devices using serial communication and a minimum number of device pins.

IDE

Integrated Development Environment. MPLAB IDE is Microchip's integrated development environment.

Identifier

A function or variable name.

IEEE

Institute of Electrical and Electronics Engineers.

Import

Bring data into the MPLAB IDE from an outside source, such as from a hex file.

Initialized Data

Data which is defined with an initial value. In C,

```
int myVar=5;
```

defines a variable which will reside in an initialized data section.

Instruction Set

The collection of machine language instructions that a particular processor understands.

Instructions

A sequence of bits that tells a central processing unit to perform a particular operation and can contain data to be used in the operation.

Internal Linkage

A function or variable has internal linkage if it can not be accessed from outside the module in which it is defined.

International Organization for Standardization

An organization that sets standards in many businesses and technologies, including computing and communications.

Interrupt

A signal to the CPU that suspends the execution of a running application and transfers control to an Interrupt Service Routine (ISR) so that the event may be processed. Upon completion of the ISR, normal execution of the application resumes.

Interrupt Handler

A routine that processes special code when an interrupt occurs.

Interrupt Request

An event which causes the processor to temporarily suspend normal instruction execution and to start executing an interrupt handler routine. Some processors have several interrupt request events allowing different priority interrupts.

Interrupt Service Routine

ALU30, C18, C30 **–** A function that handles an interrupt.

IDE **–** User-generated code that is entered when an interrupt occurs. The location of the code in program memory will usually depend on the type of interrupt that has occurred.

Interrupt Vector

Address of an interrupt service routine or interrupt handler.

IRQ

See Interrupt Request.

ISO

See International Organization for Standardization.

ISR

See Interrupt Service Routine.

© 2010 Microchip Technology Inc.

L-value

An expression that refers to an object that can be examined and/or modified. An l-value expression is used on the left-hand side of an assignment.

Latency

The time between an event and its response.

Librarian

See Archiver.

Library

See Archive.

Linker

A language tool that combines object files and libraries to create executable code, resolving references from one module to another.

Linker Script Files

Linker script files are the command files of a linker. They define linker options and describe available memory on the target platform.

Listing Directives

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

Listing File

A listing file is an ASCII text file that shows the machine code generated for each C source statement, assembly instruction, assembler directive, or macro encountered in a source file.

Little Endian

A data ordering scheme for multibyte data whereby the Least Significant Byte (LSB) is stored at the lower addresses.

Local Label

A local label is one that is defined inside a macro with the LOCAL directive. These labels are particular to a given instance of a macro's instantiation. In other words, the symbols and labels that are declared as local are no longer accessible after the ENDM macro is encountered.

Logic Probes

Up to 14 logic probes can be connected to some Microchip emulators. The logic probes provide external trace inputs, trigger output signal, +5V, and a common ground.

Loop-Back Test Board

Used to test the functionality of the MPLAB REAL ICE in-circuit emulator.

LVDS

Low Voltage Differential Signaling. A low noise, low-power, low amplitude method for high-speed (gigabits per second) data transmission over copper wire.

LVDS differs from normal input/output (I/O) in a few ways:

Normal digital I/O works with 5 volts as a high (binary '1') and 0 volts as a low (binary '0'). When you use a differential, you add a third option (-5 volts), which provides an extra level with which to encode, and results in a higher maximum data transfer rate.

A higher data transfer rate means fewer wires are required, as in UW (Ultra Wide) and UW-2/3 SCSI hard disks, which use only 68 wires. These devices require a high transfer rate over short distances. Using standard I/O transfer, SCSI hard drives would require a lot more than 68 wires.

Low voltage means that the standard 5 volts is replaced by either 3.3 volts or 1.5 volts.

LVDS uses a dual wire system, running 180 degrees of each other. This enables noise to travel at the same level, which in turn can get filtered more easily and effectively.

With standard I/0 signaling, data storage is contingent upon the actual voltage level. Voltage level can be affected by wire length (longer wires increase resistance, which lowers voltage). But with LVDS, data storage is distinguished only by positive and negative voltage values, not the voltage level. Therefore, data can travel over greater lengths of wire while maintaining a clear and consistent data stream.

Source: http://www.webopedia.com/TERM/L/LVDS.html.

Machine Code

The representation of a computer program that is actually read and interpreted by the processor. A program in binary machine code consists of a sequence of machine instructions (possibly interspersed with data). The collection of all possible instructions for a particular processor is known as its "instruction set".

Machine Language

A set of instructions for a specific central processing unit, designed to be usable by a processor without being translated.

Macro

Macro instruction. An instruction that represents a sequence of instructions in abbreviated form.

Macro Directives

Directives that control the execution and data allocation within macro body definitions.

Makefile

Export to a file the instructions to Make the project. Use this file to Make your project outside of MPLAB IDE, i.e., with a `make`.

Under *Project>Build Options>Project*, **Directories** tab, you must have selected "Assemble/Compile/Link in the project directory" under "Build Directory Policy" for this feature to work.

Make Project

A command that rebuilds an application, recompiling only those source files that have changed since the last complete compilation.

MCU

Microcontroller Unit. An abbreviation for microcontroller. Also uC.

Memory Model

C30 **–** A representation of the memory available to the application.

C18 **–** A description that specifies the size of pointers that point to program memory.

Message

Text displayed to alert you to potential problems in language tool operation. A message will not stop operation.

Microcontroller

A highly integrated chip that contains a CPU, RAM, program memory, I/O ports and timers.

Microcontroller Mode

One of the possible program memory configurations of PIC18 microcontrollers. In microcontroller mode, only internal execution is allowed. Thus, only the on-chip program memory is available in microcontroller mode.

© 2010 Microchip Technology Inc.

Microprocessor Mode

One of the possible program memory configurations of PIC18 microcontrollers. In microprocessor mode, the on-chip program memory is not used. The entire program memory is mapped externally.

Mnemonics

Text instructions that can be translated directly into machine code. Also referred to as opcodes.

MPASM™ Assembler

Microchip Technology's relocatable macro assembler for PIC microcontroller devices, KEELOQ® devices and Microchip memory devices.

MPLAB *Language Tool* for *Device*

Microchip's C compilers, assemblers and linkers for specified devices. Select the type of language tool based on the device you will be using for your application, e.g., if you will be creating C code on a PIC18 MCU, select the MPLAB C Compiler for PIC18 MCUs.

MPLAB ICD

Microchip's in-circuit debuggers that work with MPLAB IDE. The ICDs supports Flash devices with built-in debug circuitry. The main component of each ICD is the pod. A complete system consists of a pod, header board (with a *device*-ICD), target board, cables, and MPLAB IDE software.

MPLAB ICE 2000/4000

Not recommended for new designs. See the MPLAB REAL ICE in-circuit emulator.

Microchip's classic in-circuit emulators that work with MPLAB IDE. MPLAB ICE 2000 supports 8-bit PIC MCUs. MPLAB ICE 4000 supports PIC18F and PIC24 MCUs and dsPIC DSCs. The main component of each ICE is the pod. A complete system consists of a pod, processor module, cables, and MPLAB IDE software.

MPLAB IDE

Microchip's Integrated Development Environment. MPLAB IDE comes with an editor, project manager and simulator.

MPLAB PM3

A device programmer from Microchip. Programs PIC18 microcontrollers and dsPIC digital signal controllers. Can be used with MPLAB IDE or stand-alone. Replaces PRO MATE II.

MPLAB REAL ICE™ In-Circuit Emulator

Microchip's next-generation in-circuit emulators that works with MPLAB IDE. The MPLAB REAL ICE emulator supports PIC MCUs and dsPIC DSCs. The main component of each ICE is the pod. A complete system consists of a pod, a driver (and potentially a receiver) card, cables, and MPLAB IDE software.

MPLAB SIM

Microchip's simulator that works with MPLAB IDE in support of PIC MCU and dsPIC DSC devices.

MPLIB™ Object Librarian

Microchip's librarian that can work with MPLAB IDE. MPLIB librarian is an object librarian for use with COFF object modules created using either MPASM assembler (mpasm or mpasmwin v2.0) or MPLAB C18 C compiler.

MPLINK™ Object Linker

MPLINK linker is an object linker for the Microchip MPASM assembler and the Microchip C18 C compiler. MPLINK linker also may be used with the Microchip MPLIB librarian. MPLINK linker is designed to be used with MPLAB IDE, though it does not have to be.

MRU

Most Recently Used. Refers to files and windows available to be selected from MPLAB IDE main pull down menus.

Native Data Size

For Native trace, the size of the variable used in a Watch window must be of the same size as the selected device's data memory: bytes for PIC18 devices and words for 16-bit devices.

Nesting Depth

The maximum level to which macros can include other macros.

Node

MPLAB IDE project component.

Non-Extended Mode

In Non-Extended mode, the compiler will not utilize the extended instructions nor the indexed with literal offset addressing.

Non Real Time

Refers to the processor at a breakpoint or executing single-step instructions or MPLAB IDE being run in simulator mode.

Non-Volatile Storage

A storage device whose contents are preserved when its power is off.

NOP

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

Object Code

The machine code generated by an assembler or compiler.

Object File

A file containing machine code and possibly debug information. It may be immediately executable or it may be relocatable, requiring linking with other object files, e.g., libraries, to produce a complete executable program.

Object File Directives

Directives that are used only when creating an object file.

Octal

The base 8 number system that only uses the digits 0-7. The rightmost digit counts ones, the next digit counts multiples of 8, then 8^2 = 64, etc.

Off-Chip Memory

Off-chip memory refers to the memory selection option for the PIC18 device where memory may reside on the target board, or where all program memory may be supplied by the emulator. The **Memory** tab accessed from *Options>Development Mode* provides the Off-Chip Memory selection dialog box.

© 2010 Microchip Technology Inc.

One-to-One Project-Workspace Model

The most common configuration for application development in MPLAB IDE to is have one project in one workspace. Select *Configure>Settings*, **Projects** tab and check “Use one-to-one project-workspace model”.

Opcodes

Operational Codes. *See* Mnemonics.

Operators

Symbols, like the plus sign ‘+’ and the minus sign ‘-’, that are used when forming well-defined expressions. Each operator has an assigned precedence that is used to determine order of evaluation.

OTP

One Time Programmable. EPROM devices that are not in windowed packages. Since EPROM needs ultraviolet light to erase its memory, only windowed devices are erasable.

Pass Counter

A counter that decrements each time an event (such as the execution of an instruction at a particular address) occurs. When the pass count value reaches zero, the event is satisfied. You can assign the Pass Counter to break and trace logic, and to any sequential event in the complex trigger dialog.

PC

Personal Computer or Program Counter.

PC Host

Any PC running a supported Windows operating system.

Persistent Data

Data that is never cleared or initialized. Its intended use is so that an application can preserve data across a device reset.

Phantom Byte

An unimplemented byte in the dsPIC architecture that is used when treating the 24-bit instruction word as if it were a 32-bit instruction word. Phantom bytes appear in dsPIC hex files.

PIC MCUs

PIC microcontrollers (MCUs) refers to all Microchip microcontroller families.

PICkit 1, 2, and 3

Microchip’s developmental device programmers with debug capability through Debug Express. See the Readme files for each tool to see which devices are supported.

PICSTART Plus

A developmental device programmer from Microchip. Programs 8-, 14-, 28-, and 40-pin PIC microcontrollers. Must be used with MPLAB IDE software.

Plug-ins

The MPLAB IDE has both built-in components and plug-in modules to configure the system for a variety of software and hardware tools. Several plug-in tools may be found under the Tools menu.

Pod

MPLAB REAL ICE system: The box that contains the emulation control circuitry for the ICE device on the header or target board. An ICE device can be a production device with built-in ICE circuitry or a special ICE version of a production device (i.e., *device*-ICE).

MPLAB ICD: The box that contains the debug control circuitry for the ICD device on the header or target board. An ICD device can be a production device with built-in ICD circuitry or a special ICD version of a production device (i.e., *device*-ICD).

MPLAB ICE 2000/4000: The external emulator box that contains emulation memory, trace memory, event and cycle timers, and trace/breakpoint logic.

Power-on-Reset Emulation

A software randomization process that writes random values in data RAM areas to simulate uninitialized values in RAM upon initial power application.

Pragma

A directive that has meaning to a specific compiler. Often a pragma is used to convey implementation-defined information to the compiler. MPLAB C30 uses attributes to convey this information.

Precedence

Rules that define the order of evaluation in expressions.

PRO MATE II

No longer in Production. See the MPLAB PM3 device programmer.

A device programmer from Microchip. Programs most PIC microcontrollers as well as most memory and KEELOQ devices. Can be used with MPLAB IDE or stand-alone.

Production Programmer

A production programmer is a programming tool that has resources designed in to program devices rapidly. It has the capability to program at various voltage levels and completely adheres to the programming specification. Programming a device as fast as possible is of prime importance in a production environment where time is of the essence as the application circuit moves through the assembly line.

Microchip production programmers, such as MPLAB PM3, MPLAB REAL ICE in-circuit emulator, and MPLAB ICD 3, have been designed with robustness in mind to tolerate these demanding environments.

Some top-end tools have additional accessories. The MPLAB REAL ICE Performance Pak has accelerators to speed up the communication and In-Circuit Serial Programming (ICSP) process. The MPLAB PM3 programmer has interchangeable socket modules to support various devices out-of-circuit.

Profile

For MPLAB SIM simulator, a summary listing of executed stimulus by register.

Program Counter

The location that contains the address of the instruction that is currently executing.

Program Counter Unit

ALU30 – A conceptual representation of the layout of program memory. The program counter increments by 2 for each instruction word. In an executable section, 2 program counter units are equivalent to 3 bytes. In a read-only section, 2 program counter units are equivalent to 2 bytes.

© 2010 Microchip Technology Inc.

Program Memory

IDE – The memory area in a device where instructions are stored. Also, the memory in the emulator or simulator containing the downloaded target application firmware.

ALU30, C30 – The memory area in a device where instructions are stored.

Project

A project contains the files needed to build an application (source code, linker script files, etc.) along with their associations to various build tools and build options.

Prologue

A portion of compiler-generated code that is responsible for allocating stack space, preserving registers and performing any other machine-specific requirement specified in the runtime model. This code executes before any user code for a given function.

Prototype System

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

PWM Signals

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

Qualifier

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

Radix

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

RAM

Random Access Memory (Data Memory). Memory in which information can be accessed in any order.

Raw Data

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

Read Only Memory

Memory hardware that allows fast access to permanently stored data but prevents addition to or modification of the data.

Real Time

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

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

Real-Time Watch

A Watch window where the variables change in real-time as the application is run. See individual tool documentation to determine how to set up a real-time watch. Not all tools support real-time watches.

Recursive Calls

A function that calls itself, either directly or indirectly.

Recursion

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

Reentrant

A function that may have multiple, simultaneously active instances. This may happen due to either direct or indirect recursion or through execution during interrupt processing.

Relaxation

The process of converting an instruction to an identical, but smaller instruction. This is useful for saving on code size. MPLAB ASM30 currently knows how to RELAX a `CALL` instruction into an `RCALL` instruction. This is done when the symbol that is being called is within +/- 32k instruction words from the current instruction.

Relocatable

An object whose address has not been assigned to a fixed location in memory.

Relocatable Section

ALU30 – A section whose address is not fixed (absolute). The linker assigns addresses to relocatable sections through a process called relocation.

Relocation

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

ROM

Read Only Memory (Program Memory). Memory that cannot be modified.

Run

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

Run-time Model

Describes the use of target architecture resources.

Scenario

For MPLAB SIM simulator, a particular setup for stimulus control.

Section

A portion of an application located at a specific address of memory.

Section Attribute

A characteristic ascribed to a section (e.g., an `access` section).

Sequenced Breakpoints

Breakpoints that occur in a sequence. Sequence execution of breakpoints is bottom-up; the last breakpoint in the sequence occurs first.

Serialized Quick Turn Programming

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

SFR

See Special Function Registers.

© 2010 Microchip Technology Inc.

Shell

The MPASM assembler shell is a prompted input interface to the macro assembler. There are two MPASM assembler shells: one for the DOS version and one for the Windows version.

Simulator

A software program that models the operation of devices.

Single Step

This command steps though code, one instruction at a time. After each instruction, MPLAB IDE updates register windows, watch variables, and status displays so you can analyze and debug instruction execution. You can also single step C compiler source code, but instead of executing single instructions, MPLAB IDE will execute all assembly level instructions generated by the line of the high level C statement.

Skew

The information associated with the execution of an instruction appears on the processor bus at different times. For example, the executed opcodes appears on the bus as a fetch during the execution of the previous instruction, the source data address and value and the destination data address appear when the opcodes is actually executed, and the destination data value appears when the next instruction is executed. The trace buffer captures the information that is on the bus at one instance. Therefore, one trace buffer entry will contain execution information for three instructions. The number of captured cycles from one piece of information to another for a single instruction execution is referred to as the skew.

Skid

When a hardware breakpoint is used to halt the processor, one or more additional instructions may be executed before the processor halts. The number of extra instructions executed after the intended breakpoint is referred to as the skid.

Source Code

The form in which a computer program is written by the programmer. Source code is written in a formal programming language which can be translated into machine code or executed by an interpreter.

Source File

An ASCII text file containing source code.

Special Function Registers

The portion of data memory (RAM) dedicated to registers that control I/O processor functions, I/O status, timers or other modes or peripherals.

SQTP

See Serialized Quick Turn Programming.

Stack, Hardware

Locations in PIC microcontroller where the return address is stored when a function call is made.

Stack, Software

Memory used by an application for storing return addresses, function parameters, and local variables. This memory is typically managed by the compiler when developing code in a high-level language.

MPLAB Starter Kit for *Device*

Microchip's starter kits contains everything needed to begin exploring the specified device. View a working application and then debug and program you own changes.

Static RAM or SRAM

Static Random Access Memory. Program memory you can read/write on the target board that does not need refreshing frequently.

Status Bar

The Status Bar is located on the bottom of the MPLAB IDE window and indicates such current information as cursor position, development mode and device, and active tool bar.

Step Into

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

Step Over

Step Over allows you to debug code without stepping into subroutines. When stepping over a `CALL` instruction, the next breakpoint will be set at the instruction after the `CALL`. If for some reason the subroutine gets into an endless loop or does not return properly, the next breakpoint will never be reached. The Step Over command is the same as Single Step except for its handling of `CALL` instructions.

Step Out

Step Out allows you to step out of a subroutine which you are currently stepping through. This command executes the rest of the code in the subroutine and then stops execution at the return address to the subroutine.

Stimulus

Input to the simulator, i.e., data generated to exercise the response of simulation to external signals. Often the data is put into the form of a list of actions in a text file. Stimulus may be asynchronous, synchronous (pin), clocked and register.

Stopwatch

A counter for measuring execution cycles.

Storage Class

Determines the lifetime of the memory associated with the identified object.

Storage Qualifier

Indicates special properties of the objects being declared (e.g., `const`).

Symbol

A symbol is a general purpose mechanism for describing the various pieces which comprise a program. These pieces include function names, variable names, section names, file names, struct/enum/union tag names, etc. Symbols in MPLAB IDE refer mainly to variable names, function names and assembly labels. The value of a symbol after linking is its value in memory.

Symbol, Absolute

Represents an immediate value such as a definition through the assembly `.equ` directive.

System Window Control

The system window control is located in the upper left corner of windows and some dialogs. Clicking on this control usually pops up a menu that has the items "Minimize," "Maximize," and "Close."

Target

Refers to user hardware.

© 2010 Microchip Technology Inc.

Target Application

Software residing on the target board.

Target Board

The circuitry and programmable device that makes up the target application.

Target Processor

The microcontroller device on the target application board.

Template

Lines of text that you build for inserting into your files at a later time. The MPLAB Editor stores templates in template files.

Tool Bar

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

Trace

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

Trace Memory

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

Trace Macro

A macro that will provide trace information from emulator data. Since this is a software trace, the macro must be added to code, the code must be recompiled or reassembled, and the target device must be programmed with this code before trace will work.

Trigger Output

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

Trigraphs

Three-character sequences, all starting with ??, that are defined by ISO C as replacements for single characters.

Unassigned Section

A section which has not been assigned to a specific target memory block in the linker command file. The linker must find a target memory block in which to allocate an unassigned section.

Uninitialized Data

Data which is defined without an initial value. In C,

```
int myVar;
```

defines a variable which will reside in an uninitialized data section.

Upload

The Upload function transfers data from a tool, such as an emulator or programmer, to the host PC or from the target board to the emulator.

USB

Universal Serial Bus. An external peripheral interface standard for communication between a computer and external peripherals over a cable using bi-serial transmission. USB 1.0/1.1 supports data transfer rates of 12 Mbps. Also referred to as high-speed USB, USB 2.0 supports data rates up to 480 Mbps.

Vector

The memory locations that an application will jump to when either a Reset or interrupt occurs.

Warning

IDE – An alert that is provided to warn you of a situation that would cause physical damage to a device, software file, or equipment.

ALU30, C30 – Warnings report conditions that may indicate a problem, but do not halt processing. In MPLAB C30, warning messages report the source file name and line number, but include the text '`warning`:' to distinguish them from error messages.

Watch Variable

A variable that you may monitor during a debugging session in a Watch window.

Watch Window

Watch windows contain a list of watch variables that are updated at each breakpoint.

Watchdog Timer

A timer on a PIC microcontroller that resets the processor after a selectable length of time. The WDT is enabled or disabled and set up using Configuration bits.

WDT

See Watchdog Timer.

Workbook

For MPLAB SIM stimulator, a setup for generation of SCL stimulus.

WorkSpace

A workspace contains MPLAB IDE information on the selected device, selected debug tool and/or programmer, open windows and their location, and other IDE configuration settings.

© 2010 Microchip Technology Inc.

Index

© 2010 Microchip Technology Inc.

NOTES:

WORLDWIDE SALES AND SERVICE

AMERICAS

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

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-6578-300
Fax: 886-3-6578-370

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

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

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/05/10

© 2010 Microchip Technology Inc.

Mouser Electronics

Authorized Distributor

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

Digilent:
240-035P

OCEAN CHIPS
Океан Электроники
Поставка электронных компонентов

Компания «Океан Электроники» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Поставка оригинальных импортных электронных компонентов напрямую с производств Америки, Европы и Азии, а так же с крупнейших складов мира;
- Широкая линейка поставок активных и пассивных импортных электронных компонентов (более 30 млн. наименований);
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Помощь Конструкторского Отдела и консультации квалифицированных инженеров;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Поставка электронных компонентов под контролем ВП;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- При необходимости вся продукция военного и аэрокосмического назначения проходит испытания и сертификацию в лаборатории (по согласованию с заказчиком);
- Поставка специализированных компонентов военного и аэрокосмического уровня качества (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Actel, Aeroflex, Peregrine, VPT, Syfer, Eurofarad, Texas Instruments, MS Kennedy, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits, General Dynamics и др.);

Компания «Океан Электроники» является официальным дистрибьютором и эксклюзивным представителем в России одного из крупнейших производителей разъемов военного и аэрокосмического назначения «**JONHON**», а так же официальным дистрибьютором и эксклюзивным представителем в России производителя высокотехнологичных и надежных решений для передачи СВЧ сигналов «**FORSTAR**».

AVIC JONHON

«JONHON» (основан в 1970 г.)

Разъемы специального, военного и аэрокосмического назначения:

(Применяются в военной, авиационной, аэрокосмической, морской, железнодорожной, горно- и нефтедобывающей отраслях промышленности)

«FORSTAR» (основан в 1998 г.)

ВЧ соединители, коаксиальные кабели, кабельные сборки и микроволновые компоненты:

(Применяются в телекоммуникациях гражданского и специального назначения, в средствах связи, РЛС, а так же военной, авиационной и аэрокосмической отраслях промышленности).

FORSTAR®

Телефон: 8 (812) 309-75-97 (многоканальный)
Факс: 8 (812) 320-03-32
Электронная почта: ocean@oceanchips.ru
Web: http://oceanchips.ru/
Адрес: 198099, г. Санкт-Петербург, ул. Калинина, д. 2, корп. 4, лит. А