PICkit™ 3
In-Circuit Debugger/Programmer
User's Guide

For MPLAB® X IDE

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.

Trademarks

QUALITY MANAGEMENT SYSTEM

CERTIFIED BY DNV

═ ISO/TS 16949 ═

Object of Declaration: PICkit™3 In-Circuit Debugger/Programmer

EU Declaration of Conformity

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

This declaration of conformity is issued by the manufacturer.

The development/evaluation tool is designed to be used for research and development in a laboratory environment. This development/evaluation tool is not intended to be a finished appliance, nor is it intended for incorporation into finished appliances that are made commercially available as single functional units to end users. This development/evaluation tool complies with EU EMC Directive 2004/108/EC and as supported by the European Commission's Guide for the EMC Directive 2004/108/EC (8th February 2010).
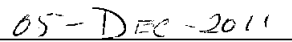
This development/evaluation tool complies with EU RoHS2 Directive 2011/65/EU.

For information regarding the exclusive, limited warranties applicable to Microchip products, please see Microchip's standard terms and conditions of sale, which are printed on our sales documentation and available at www.microchip.com.

Signed for and on behalf of Microchip Technology Inc. at Chandler, Arizona, USA

Derek Carlson   05 – DEC – 2011
VP Development Tools   Date

NOTES:

MPLAB® X PICkit™ 3 USER'S GUIDE

Table of Contents

MPLAB® X PICkit™ 3 User's Guide

Troubleshooting

Reference

MPLAB® X PICkit™ 3 USER'S GUIDE

Preface

NOTICE TO CUSTOMERS

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

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

For the most up-to-date information on development tools, see the MPLAB® X 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™ starter kit. Items discussed in this chapter include:

• Document Layout
• Conventions Used in this Guide
• Recommended Reading

MPLAB® X PICkit™ 3 User's Guide

DOCUMENT LAYOUT

This document describes how to use the PICkit 3 as a development tool to emulate and debug firmware on a target board, as well as how to program devices. The document is organized as follows:

Part 1 – Getting Started

- **Chapter 1. About the In-Circuit Debugger/Programmer**
  Describes the PICkit 3, and how it can help you develop your application.
- **Chapter 2. Operation**
  Presents the theory of PICkit 3 operation. Explains configuration options.
- **Chapter 3. Debugger Usage**
  Discusses installation and set up, common debug features, using targets, setting up MPLAB® Integrated Development Environment (IDE), and related debugger topics.
- **Chapter 4. PICkit 3 Debug Express**
  Provides basic information about using the PICkit 3 Debug Express.
- **Chapter 5. PICkit 3 Programmer-To-Go**
  Provides instructions for using the PICkit 3 unit to program a device even though it is not connected to a personal computer (PC).

Part 2 – Troubleshooting

- **Chapter 6. Troubleshooting First Steps** – The first things you should try if you are having issues with debugger operation.
- **Chapter 7. Frequently Asked Questions (FAQs)** – A list of frequently asked questions, useful for troubleshooting.
- **Chapter 8. Error Messages** – A list of error messages and suggested resolutions.
- **Chapter 9. Engineering Technical Notes (ETNs)**

Part 3 – Reference

- **Appendix A. Hardware Specification** – The hardware and electrical specifications of the debugger system.
- **Appendix B. PICkit 3 Schematics**
- **Appendix C. Revision History**

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	*MPLAB® IDE User's Guide*
	Emphasized text	...is the *only* compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	*File>Save*
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
N'Rnnnn	A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain Courier New	Sample source code	`#define START`
	Filenames	`autoexec.bat`
	File paths	`c:\mcc18\h`
	Keywords	`_asm, _endasm, static`
	Command-line options	`-Opa+, -Opa-`
	Bit values	`0, 1`
	Constants	`0xFF, 'A'`
Italic Courier New	A variable argument	`file`.o, where `file` can be any valid filename
Square brackets [ ]	Optional arguments	`mcc18 [options] file [options]`
Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments; an OR selection	`errorlevel {0|1}`
Ellipses...	Replaces repeated text	`var_name [, var_name...]`
	Represents code supplied by user	`void main (void) { ... }`

MPLAB® X PICkit™ 3 User's Guide

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.

Multi-Tool Design Advisory (DS51764)

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

Release Notes for PICkit 3

For the latest information on using PICkit 3, read the release notes under "Release Notes and Support Documentation" on the Start Page. The release notes contain update information and known issues that may not be included in this user's guide.

MPLAB X - Using PICkit 3 In-Circuit Debugger Poster (DS52010)

This poster shows you how to hook up the hardware and install the software for the PICkit 3 programmer Debugger using standard communications and a target board.

PICkit 3 In-Circuit Debugger/Programmer Online Help File

A comprehensive help file for the debugger is included with MPLAB X IDE. Usage, troubleshooting and hardware specifications are covered. This may be more up-to-date than the printed documentation. Also, limitations are listed for various devices.

Processor Extension Pak and Header Specification (DS51292)

This booklet describes how to install and use PICkit in-circuit debug headers to better debug selected devices without the loss of pins or resources. See also the PEP and Header online help file.

Transition Socket Specification (DS51194)

Consult this document for information on transition sockets available for use with headers.

MPLAB® X PICkit™ 3 USER'S GUIDE

Part 1 – Getting Started

NOTES:

Chapter 1. About the In-Circuit Debugger/Programmer

1.1 INTRODUCTION

An overview of the PICkit 3™ In-Circuit Debugger/Programmer system is provided.

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

1.2 PICKIT 3 IN-CIRCUIT DEBUGGER/PROGRAMMER DEFINED

The PICkit 3 In-Circuit Debugger/Programmer (see Figure 1-1) is a simple, low-cost in-circuit debugger that is controlled by a PC running MPLAB X IDE software on a Windows® platform. The PICkit 3 In-Circuit Debugger/Programmer is an integral part of the development engineer's tool suite. The application usage can vary from software development to hardware integration.

The PICkit 3 In-Circuit Debugger/Programmer is a debugger system used for hardware and software development with 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 In-Circuit Debugger/Programmer system also may be used as a development programmer.

> **Note:** The PICkit 3 In-Circuit Debugger/Programmer is NOT a production programmer. It should be used for development purposes only.

The PICkit 3 debugger was developed for programming and debugging embedded processors with debug functions. The PICkit 3 features include:

- Full-speed USB support using Windows standard drivers
- Real-time execution
- Processors running 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
- Erasing of all memory types (EEPROM, ID, configuration and program) with verification
- Peripheral freeze at breakpoint

MPLAB® X PICkit™ 3 User's Guide

FIGURE 1-1: **PICkit™ 3 MCU IN-CIRCUIT DEBUGGER/PROGRAMMER**

Legend:

1 – Lanyard Loop

2 – USB Port Connection

3 – Pin 1 Marker

4 – Programming Connector

5 – Indicator LEDs

6 – Push Button

1.2.1   Lanyard Loop

The lanyard loop provides a point of attachment so that the PICkit 3 can be suspended or worn.

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.

FIGURE 1-2: **PICkit™ 3 PROGRAMMER CONNECTOR PINOUT**

Pin 1 Indicator

Pin Description*

1 = $\overline{MCLR}$/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.
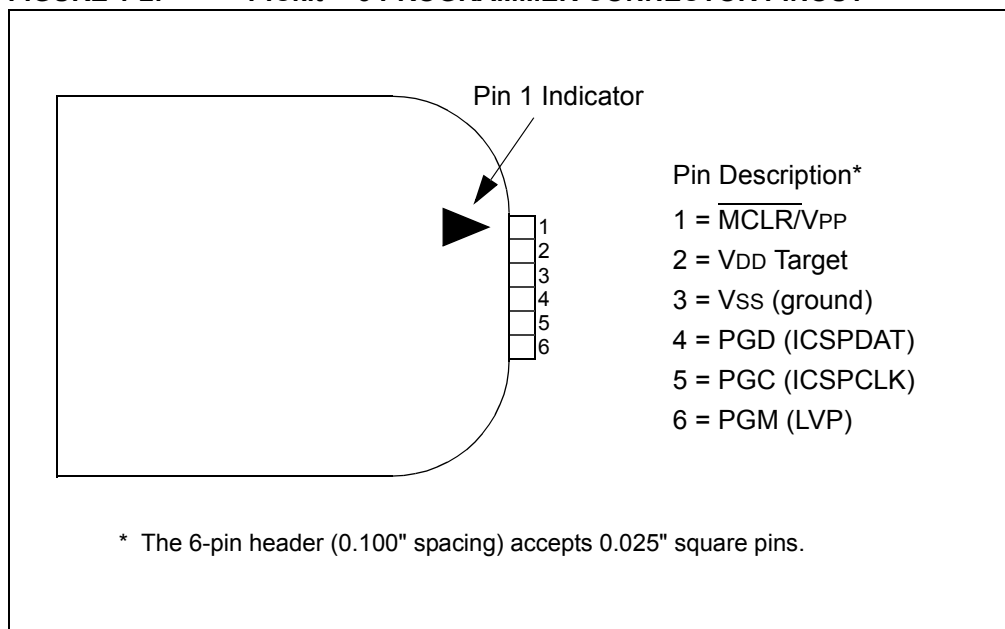
Note: Programming Serial EEPROMS devices requires a different programming connector pinout. Pinouts for those types of devices are available in the ReadMe file for the PICkit 3 included with the MPLAB X IDE software (*MPLAB X IDE Start Page, click on Release Notes and Support Documentation*).

1.2.5 Indicator LEDs

The indicator LEDs indicate the status of operations on the PICkit 3.

1. **Power** (green) – power is supplied to the PICkit 3 via the USB port
2. **Active** (blue) – connected to the PC USB port and the communication link is active
3. **Status** (one of three colors)

    Success (green) – ready to start, or successful completion

    Busy (orange) – busy with a function in progress, e.g., programming

    Error (red) – an error has occurred

Note: Blinking LEDs indicate additional information. For details, see Table 5-2.

1.2.6 Push Button

The push button is used for Programmer-To-Go operations. See **Chapter 5. "PICkit 3 Programmer-To-Go"**.

MPLAB® X PICkit™ 3 User's Guide

1.3   HOW THE PICKIT 3 IN-CIRCUIT DEBUGGER/PROGRAMMER HELPS YOU

The PICkit 3 In-Circuit Debugger/Programmer enables 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 IN-CIRCUIT DEBUGGER/PROGRAMMER COMPONENTS

The PICkit 3 In-Circuit Debugger/Programmer system contains the following components:

1. the PICkit 3 with indicator lights for power, activity and status.
2. a USB cable to provide communication between the debugger and a PC, and to provide power to the debugger.

FIGURE 1-3:   BASIC DEBUGGER SYSTEM
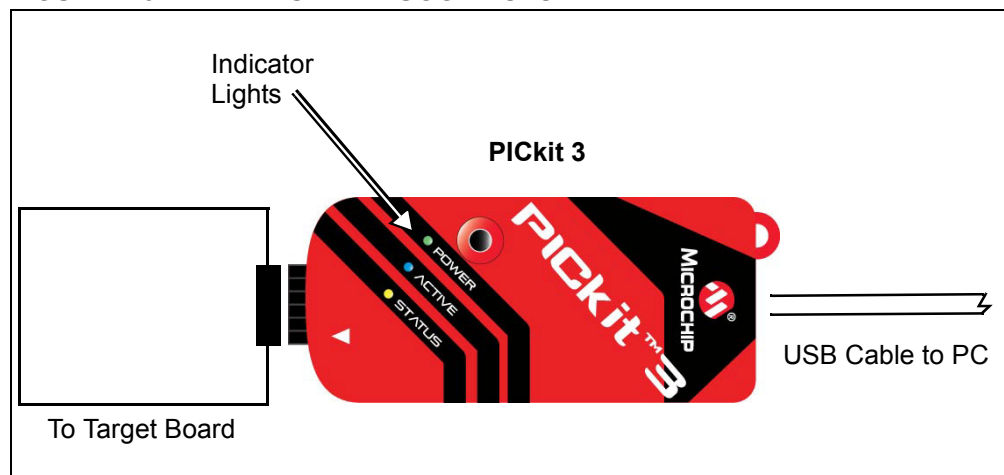
Additional items can be ordered separately:

• PICkit 3 Debug Express Kit, which includes:
   - 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 X IDE processor extension kits

Chapter 2. Operation

2.1 INTRODUCTION

A simplified description of how the PICkit 3 In-Circuit Debugger/Programmer 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 debugging and programming operations. The basic theory of in-circuit debugging and programming is described so that problems, if encountered, are quickly resolved.

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

2.2 TOOLS COMPARISON

The PICkit 3 In-Circuit Debugger/Programmer system differs physically and operation-ally from other Microchip debug tools as shown below. Specific features may vary by device (see the online help file, **"Device and Feature Support"**.)

TABLE 2-1:   DEBUG TOOLS COMPARISON

Features	PICkit 3 Programmer/ Debugger	MPLAB ICD 3 In-Circuit Debugger	MPLAB REAL ICE In-Circuit Emulator
USB Speed	Full Only	High and Full	High and Full
USB Driver	HID	Microchip	Microchip
USB Powered	Yes	Yes	Yes
Power to Target	Yes	Yes	No
Programmable V_{PP} and V_{DD}	Yes	Yes	Yes
Vdd Drain from Target	20 ma	<1 ma	<1 ma
Overvoltage/Overcurrent Protection	Yes (SW)	Yes (HW)	Yes (HW)
Device emulation	Full speed	Full speed	Full speed
HW Breakpoints	Simple	Complex	Complex
Stopwatch	Yes	Yes	Yes
SW Breakpoints	No	Yes	Yes
Program Image	512K bytes	No	No
Serialized USB	Yes	Yes	Yes
Trace	No	No	Yes
Data Capture	No	No	Yes
Logic Probe Triggers	No	No	Yes
High Speed/LVDS Connection	No	No	Yes
Production Programmer	No	Yes	Yes

2.3 PICkit 3 vs. PICkit 2

The PICkit 3 In-Circuit Debugger/Programmer 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 these additional features:

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

2.4 DEBUGGER TO TARGET COMMUNICATION

The debugger system configurations are discussed in the following sections.

CAUTION

Install the software before making any hardware connections, i.e., do NOT connect the hardware before installing the software and USB drivers.

Do NOT change hardware connections when the PICkit 3 and/or the target are 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 older PICkit 2 Development Programmer/Debugger.

The modular cable can be inserted into either:

- a matching socket at the target, where the target device is on the target board (Figure 2-1), or
- 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 modular connector instead of an 6-pin single in-line connector, so these headers can be connected to the debugger using an 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

MPLAB® X PICkit™ 3 User's Guide
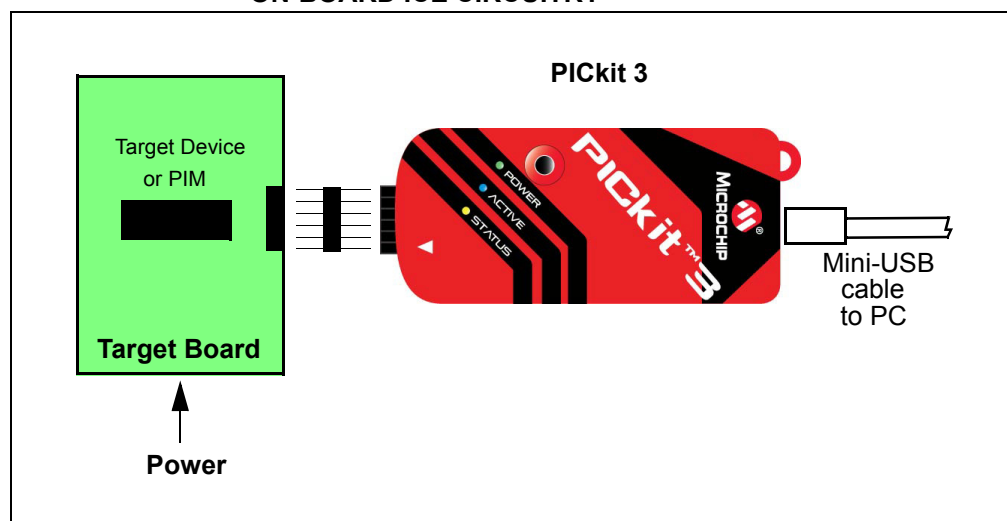
FIGURE 2-2:   **STANDARD DEBUGGER SYSTEM – ICE DEVICE**
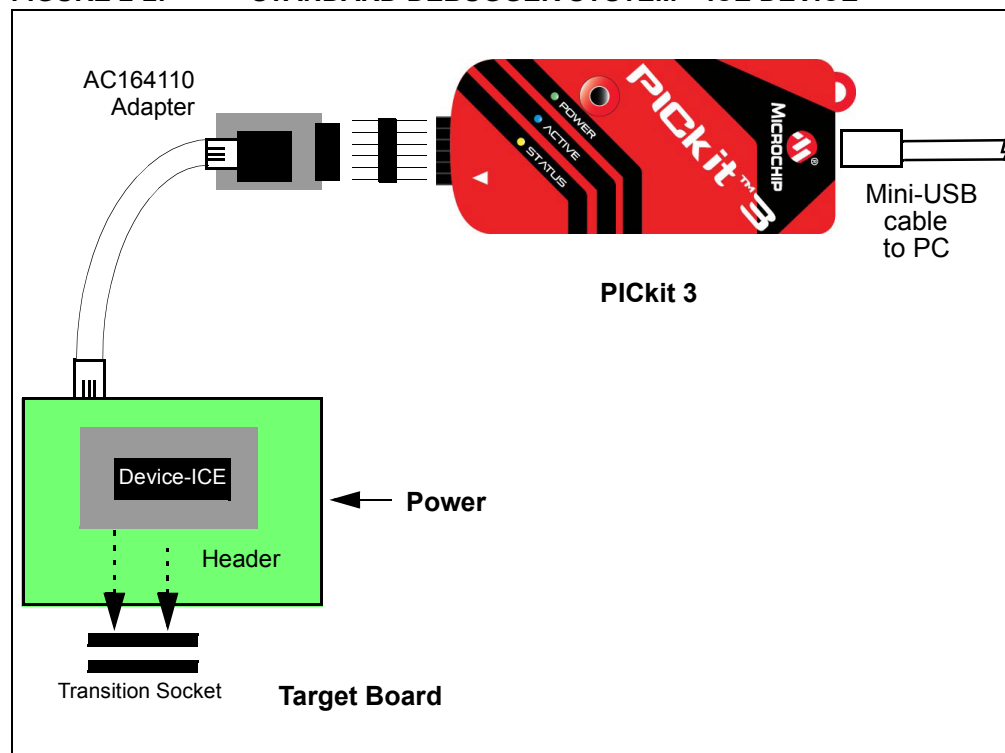
2.5   COMMUNICATION CONNECTIONS

2.5.1   Standard Communication Target Connections

2.5.1.1   USING SINGLE IN-LINE CONNECTOR

Use the 6-pin in-line connector between the PICkit 3 In-Circuit Debugger/Programmer and the target board connector. See Figure 2-1. Also see Table 2-2 and **Section A.6 "Standard Communication Hardware"**.

TABLE 2-2:   TARGET CONNECTOR PINOUT

Connector Pin	Microcontroller Pin
1	$\overline{MCLR}$/V$_{PP}$
2	V$_{DD}$
3	Ground
4	PDG (ICSPDAT)
5	PGC (ICSPCLK)
6	PGM (LVP)

2.5.1.2   USING AN ADAPTER

Use the AC164110 adapter between the PICkit 3 In-Circuit Debugger/Programmer 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
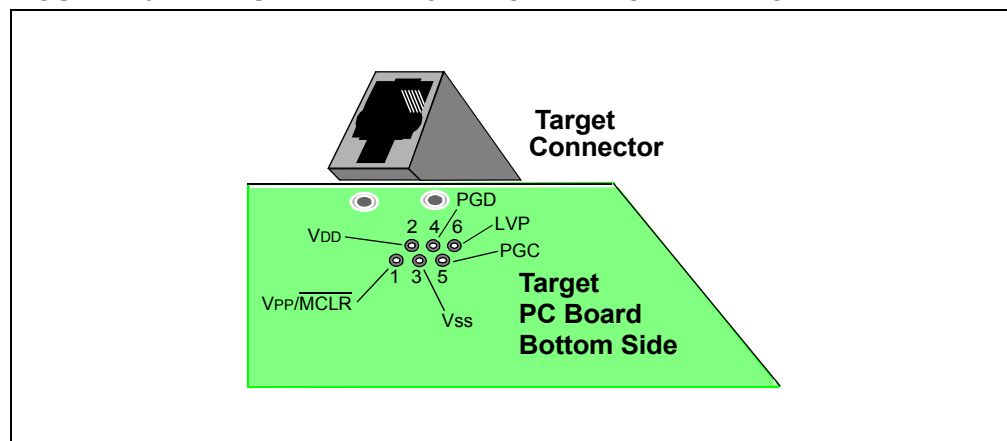
2.5.2 Target Connection Circuitry

Figure 2-4 shows the interconnections of the PICkit 3 In-Circuit Debugger/Programmer 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 (typically 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
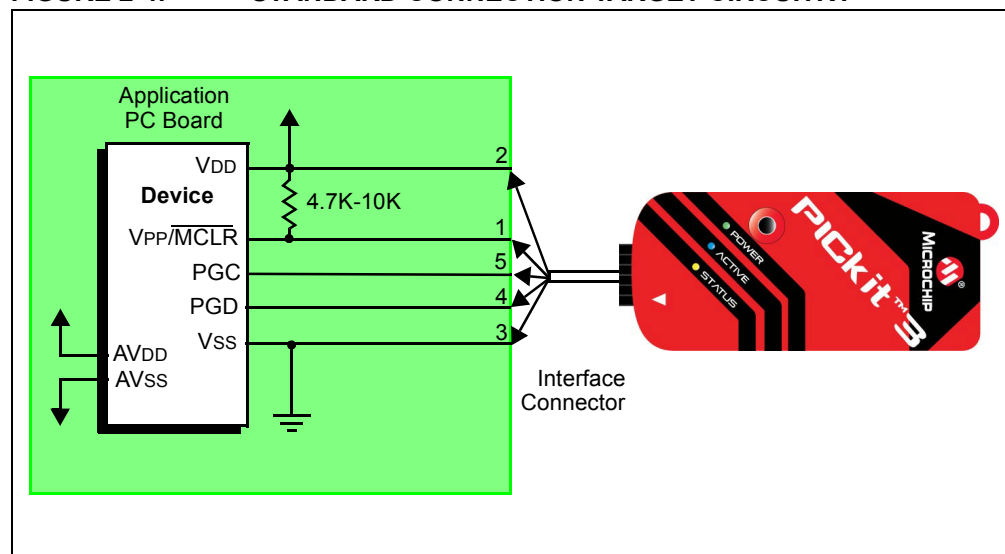
2.5.3 Target Powered

In the following descriptions, only three lines are active and relevant to core debugger operation: pins 1 (VPP/MCLR), 5 (PGC) and 4 (PGD). Pins 2 (VDD) and 3 (VSS) 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 VDD is sensed by the debugger to allow level translation for the target low-voltage operation. If the debugger does not sense voltage on its VDD line (pin 2 of the interface connector), it will not operate.

2.5.4 Debugger Powered

The internal debugger power is limited to 30 mA. This can be helpful with very small applications that have the device VDD separated from the rest of the application circuit for independent programming. However, is not recommended for general usage because imposes more current demands from the USB power system derived from the PC.

Not all devices have the AVDD and AVSS 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 VCAP line (PIC18FXXJ MCUs, for example) should be connected to the appropriate capacitor or level.

Note:	The interconnection is very simple. Any problems that occur are often caused by other connections or components on these critical lines that interfere with the operation of the PICkit 3 In-Circuit Debugger/Programmer.

2.5.5 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**
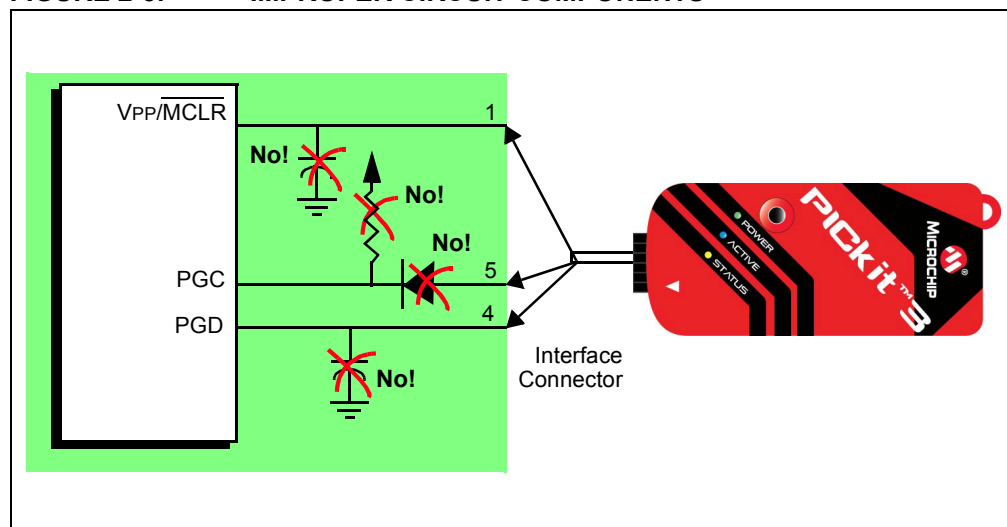
In particular, these guidelines must be followed:

• 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 V$_{PP}$. 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.

2.6   DEBUGGING

There are two steps to using the PICkit 3 In-Circuit Debugger/Programmer system as a debugger. The first step requires that an application be programmed into the target device (usually with the PICkit 3 itself). The second step 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 X IDE operations:

1. Programming the code into the target and activating special debug functions (see the next section for details).
2. Using the debugger to set breakpoints and run.

If the target device cannot be programmed correctly, the PICkit 3 In-Circuit Debugger/Programmer 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 V$_{DD}$ and V$_{SS}$ lines from the debugger are not shown.

FIGURE 2-6:   PROPER CONNECTIONS FOR PROGRAMMING
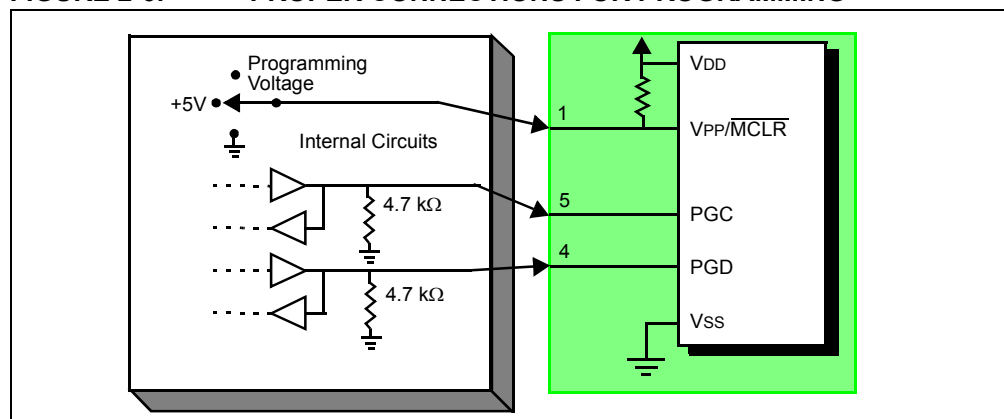
A simplified diagram of some of the internal interface circuitry of the PICkit 3 In-Circuit Debugger/Programmer is shown. For programming, no clock is needed on the target device, but power must be supplied. When programming, the debugger puts programming levels on V$_{PP}$/MCLR, sends clock pulses on 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.

2.7   REQUIREMENTS FOR DEBUGGING

To debug (set breakpoints, see registers, etc.) with the PICkit 3 In-Circuit
Debugger/Programmer 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 X IDE software via the
  USB cable. See **Chapter 3. "Debugger Usage"** for details.
- The debugger must be connected as shown in Figure 2-6 to the V$_{PP}$, PGC and
  PGD pins of the target device with the modular interface cable (or equivalent).
  V$_{SS}$ and V$_{DD}$ 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 In-Circuit
  Debugger/Programmer 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.

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

- Sequence of Operations Leading to Debugging
- Debugging Details

2.7.1   Sequence of Operations Leading to Debugging

Given that the Requirements for Debugging are met, these actions can be performed
when the PICkit 3 In-Circuit Debugger/Programmer is set as the current tool from the
MPLAB X IDE menu (*Edit>Project Properties, Advanced, MPLAB Environment*):

- When *Debug>Debug Project* is selected, the application code is programmed into
  the device's memory via the ICSP protocol as described at the beginning of this
  section.
- A small "debug executive" program is loaded into the high area of program
  memory on the target device. 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 by MPLAB X
  IDE. These allow the debug executive to be activated by the debugger. For more
  information on the device's reserved resources, see **Section 2.9 "Resources
  Used by the Debugger"**.
- The target device is run in Debug mode.

2.7.2   Debugging Details

Figure 2-7 illustrates the PICkit 3 In-Circuit Debugger/Programmer system when it is ready to perform debugging.

FIGURE 2-7:   PICkit™ 3 DEBUGGER READY FOR DEBUGGING
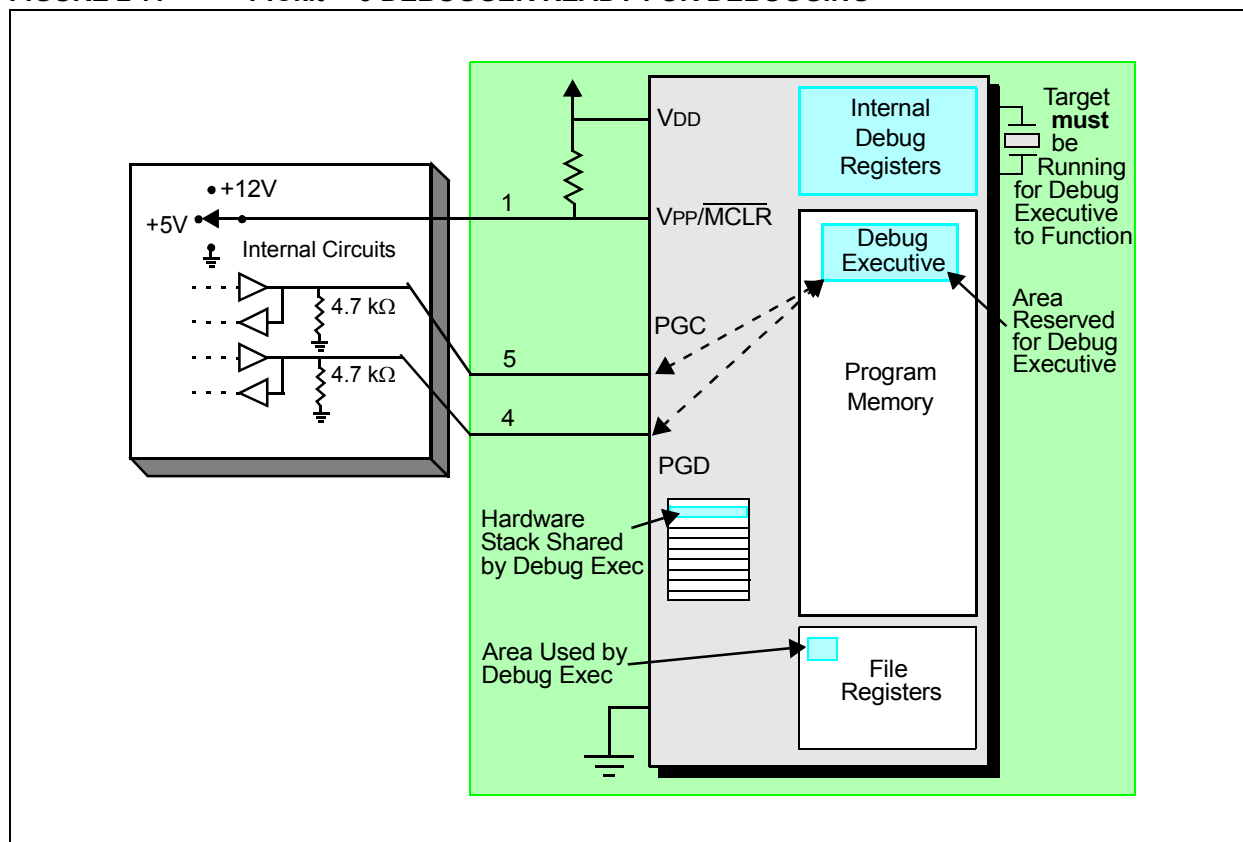
Typically, to find out whether an application program will run correctly, a breakpoint is set early in the program code. When a breakpoint is set from the user interface of MPLAB X IDE, the address of the breakpoint is stored in the special internal debug registers of the target device. Commands on PGC and PGD communicate directly to these registers to set the breakpoint address.

Next, the _Debug>Debug Project_ function is selected in MPLAB X IDE. The debugger tells the debug executive to run. The target starts from the Reset vector and executes until the program counter reaches the breakpoint address that was stored previously 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 X IDE. MPLAB X 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, i.e., no oscillator, faulty power supply connection, shorts on the target board, etc., then the debug executive cannot communicate to the PICkit 3 In-Circuit Debugger/Programmer, and MPLAB X IDE will issue an error message.

Another way to get a breakpoint is to select *Debug>Pause*. 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 X IDE uses the debugger communications with the debug executive to interrogate the state of the target device.

2.8   PROGRAMMING

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

- Through MPLAB X IDE with the PICkit 3 connected to the PC.
- Through PICkit 3 Programmer-To-Go, after setting it up through MPLAB X IDE. (See **Chapter 5. "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 X IDE. (See "*PICkit 3 Programmer Application User's Guide"* for instructions.)

Use the PICkit 3 as a programmer to program an actual (non -ICE/-ICD) device, i.e., a device not on a header board. Set the PICkit 3 as the current tool (*Edit>Project Properties*, Advanced, MPLAB Environment) to perform these actions:

- When *Run>Run Project* is selected, the application code is programmed into the device's memory via the ICSP protocol. No clock is required while programming, and all modes of the processor can be programmed, including code protect, Watchdog Timer enabled, and table read protect.
- A small "program executive" program may be loaded into the high area of program memory for some target device. This increases programming speeds for devices with large memories.
- Special "in-circuit debug" registers in the target device are disabled by MPLAB X IDE, along with all debug features. This means that a breakpoint cannot be set, and register contents cannot be seen or altered.
- The target device is run in Release mode. As a programmer, the debugger can only toggle the $\overline{\text{MCLR}}$ line to Reset and start the target.

The PICkit 3 In-Circuit Debugger/Programmer 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.

2.9   RESOURCES USED BY THE DEBUGGER

For a complete list of resources used by the debugger for your device, please see MPLAB X IDE Start Page, click on *Release Notes and Support Documentation*, then click on link for the Reserved Resources for PICkit 3.

NOTES:

Chapter 3. Debugger Usage

3.1 INTRODUCTION

The following topics regarding how to install and use the PICkit 3 In-Circuit Debugger/Programmer system are discussed here.

- Installation and Setup
- Common Debug Features
- Connecting the Target
- Setting Up the Target Board
- Setting Up MPLAB X IDE
- Starting and Stopping Debugging
- Viewing Processor Memory and Files
- For more on the Editor, see NetBeans Help, IDE Basics>Basic File Features.

3.2 INSTALLATION AND SETUP

Refer to the Help file "Getting Started with MPLAB X IDE" for details on installing the IDE and setting up the debugger to work with it.

In summary:

1. Install MPLAB X IDE.
2. Connect the PICkit 3 to the PC and allow the default drivers to install. For more information on target connections, see **Chapter 2. "Operation"**.

   > **Note:** The debugger can power a target board only up to 100 mA.

3. Install the language toolsuite/compiler you want to use for development.
4. Launch MPLAB X IDE.
5. Use the New Project wizard (*File>New Project*) to add your "PICkit 3" to your project.
6. Use the project Properties dialog (*File>Project Properties*) to set up options.
7. Use the project Properties dialog (*File>Project Properties<Hardware tool>*) to set up tool options for programming.
8. Run the project (build and run) from *Run>Run Project*.

Items of note are:

1. Each debugger contains a unique identifier which, when first installed, will be recognized by the OS, regardless of which computer USB port is used.
2. MPLAB X IDE operation connects to the hardware tool at runtime (Run or Debug Run). To always be connected to the hardware tool (like MPLAB IDE v8), see *Tools>Options*, **Embedded** button, **Generic Settings** tab, "Keep hardware tool connected" checkbox.
3. Configuration bits can only be viewed in the Configuration Bits window. To set them in code, select *Window>PIC Memory Views*. Then, select "Configuration Bits" from the Memory drop list, and select "Read/Write" from the Format drop list to enable access to the settings.

MPLAB® X PICkit™ 3 User's Guide

3.3 COMMON DEBUG FEATURES

Refer to the Help file "Getting Started with MPLAB X IDE", Debugging Code section, for details on debug features. This section includes:

1. Debug Running the project (build, program and run) from *Debug>Debug Project*.
2. Using breakpoints
3. Stepping through code
4. Using the Watch window
5. Viewing Memory, Variables and the Call Stack
6. Using the Call Graph

3.4 CONNECTING THE TARGET

A connection is built in to select the type of communication with the target. See **Section 2.4 "Debugger to Target Communication"** for more details and a diagram.

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

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

© 2013 Microchip Technology Inc.
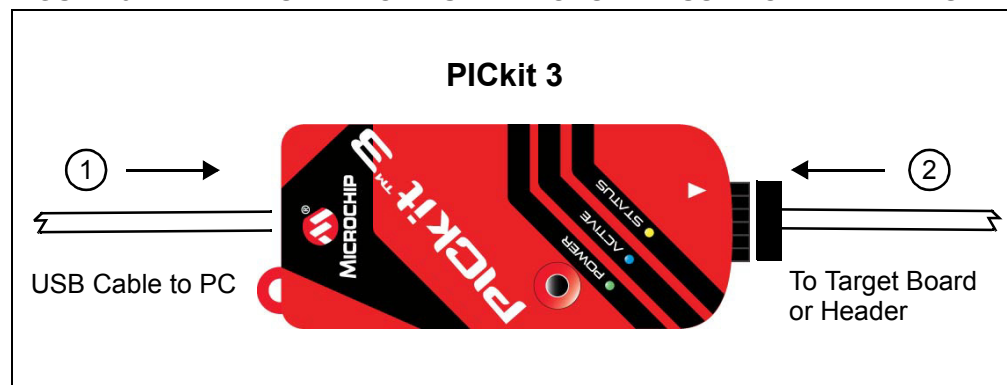
3.5   SETTING UP THE TARGET BOARD

3.5.1   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 to debug with the PICkit 3 In-Circuit Debugger/Programmer. Consult the device data sheet to see whether the device has the necessary 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 2.4 "Debugger to Target Communication"**, "Standard ICSP Device Communication".

3.5.2   Using ICE Devices

For ICE devices, an ICE header board is required. The header board contains the hardware that is required to emulate a specific device or family of devices. For more information on ICE headers, see the "*Processor Extension Pak and Header 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 2.4 "Debugger to Target Communication"**, "Standard ICSP Device Communication".

3.5.3   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 "*Processor Extension Pak and Header Specification*" (DS51292). The Processor Extension Pak and Header Specification is available online at www.microchip.com.

Each ICD header board comes with the necessary –ICD device, and it 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.

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.

FIGURE 3-2: **USING AN ICD HEADER BOARD**

This diagram shows how to connect the PICkit™ 3 to an ICD header with the ICSP to RJ-11 Adapter. The **AC162061** ICD header for the PIC16F690 is shown, plugging into the **DM164120-1** Low Pin Count Demo Board.
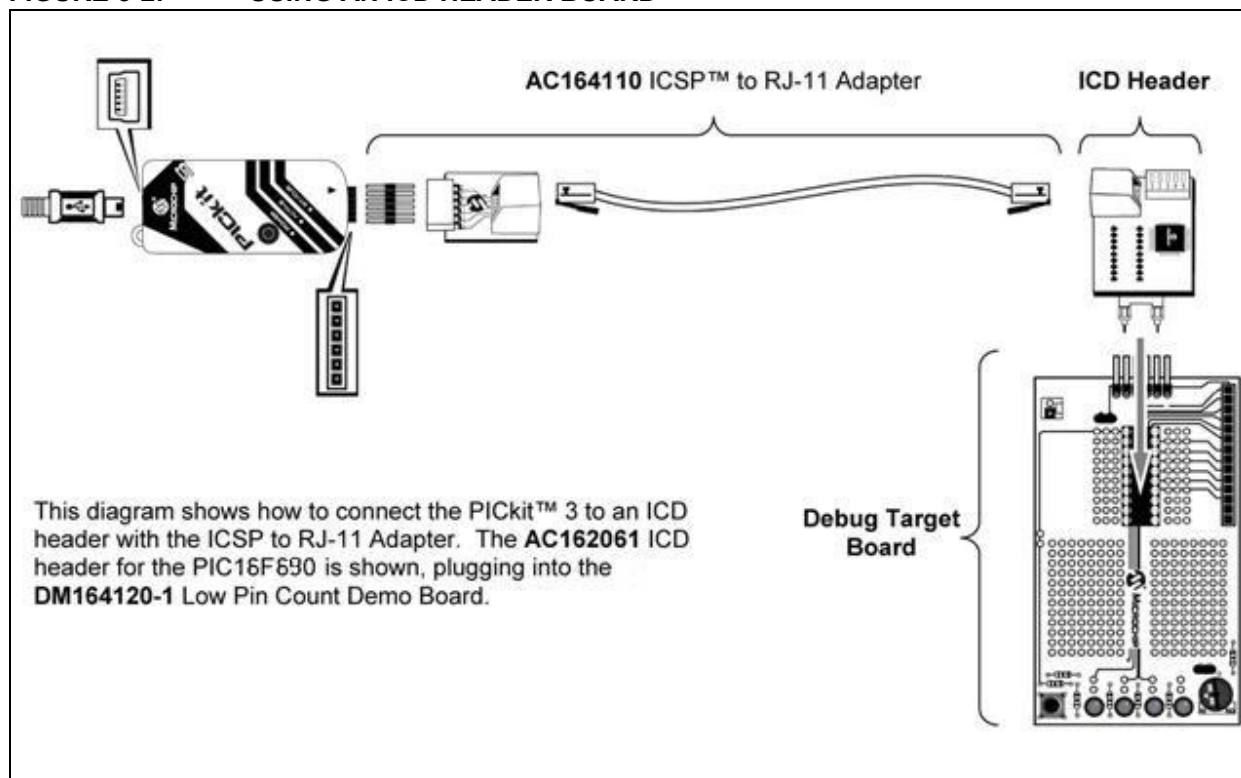
3.5.4 Powering the Target

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 X 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 3.4 "Connecting the Target"**). Then power the target. You can also power the target from PICkit 3.

3.6   SETTING UP MPLAB X IDE

Once the hardware is connected and powered, MPLAB X IDE may be set up for use with the PICkit 3 in-circuit 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.

Refer to the MPLAB X IDE Help for details on installing the software and setting up the debugger to work with it.

3.7   STARTING AND STOPPING DEBUGGING

To debug an application in the MPLAB X IDE, you must create a project containing your source code so that the code may be built, programmed into your device, and executed as specified below:

- To run your code, select either *Debug>Debug Project* or **Debug Project** from the Run toolbar.
- To halt your code, select either *Debug>Pause* or **Pause** from the Debug toolbar.
- To run your code again, select either *Debug>Continue* or **Continue** from the Debug toolbar.
- To step through your code, select either *Debug>Step Into* or **Step Into** from the Debug toolbar. Be careful not to step into a Sleep instruction or you will have to perform a processor Reset to resume emulation.
- To step over a line of code, select either *Debug>Step Over* or **Step Over** from the Debug toolbar.
- To end code execution, select either *Debug>Finish Debugger Session* or **Finish Debugger Session** from the Debug toolbar.
- To perform a processor Reset on your code, select either *Debug>Reset* or **Reset** from the Debug toolbar. Additional Resets, such as POR/BOR, MCLR and System, may be available, depending on the device.

3.8   VIEWING PROCESSOR MEMORY AND FILES

MPLAB X IDE provides several windows for viewing debug and various processor memory information that are selectable from the Window menu. See MPLAB X IDE online help for more information on using these windows.

- *Window>PIC Memory Views* - View data (RAM) and code (ROM) device memory. Select from RAM, Flash, special function registers (SFRs), CPU, and Configuration bits.
- *Window>Debugging* - View debug information. Select from variables, watches, call stack, breakpoints, and stopwatch.

To view your source code, find the source code file you wish to view in the Project window and double-click to open it in a File window. Code in this window is color-coded according to the processor and build tool that you have selected. To change the style of color-coding, select *Tools>Options*, **Fonts & Colors**, **Syntax** tab.

For more on the Editor, see NetBeans Help, *IDE Basics>Basic File Features*.

NOTES:

Chapter 4. PICkit 3 Debug Express

4.1 INTRODUCTION

The PICkit 3 Debug Express kit works in conjunction with the MPLAB X 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 contents of the register can be examined and modified.

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

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

4.2 PICkit 3 DEBUG EXPRESS KIT CONTENTS

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

1. The PICkit 3 Development In-Circuit Debugger/Programmer
2. USB cable
3. 44-Pin Demo Board with device

4.3 INSTALLING THE HARDWARE AND SOFTWARE

Install the PICkit 3 hardware and software.

> **Note:** PICkit 3 Debug Express requires MPLAB X IDE version 1.20 or later.

4.3.1 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 a complete list of resources used by the debugger for your device, please see MPLAB X IDE Start Page, click on *Release Notes and Support Documentation*, then click on link for the Reserved Resources for PICkit 3.

4.3.2 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 4-1. (The Explorer 16 board may also be used for debugging.)

FIGURE 4-1: **CONNECTING THE DEMO BOARD TO THE PICkit™ 3**

PIC® microcontrollers that don't require an ICD Header may be debugged directly over the In-Circuit Serial-Programming™ (ICSP™) connections.

The 44-Pin Demo Board with PIC18F45K20 shown.

Debug Target Board
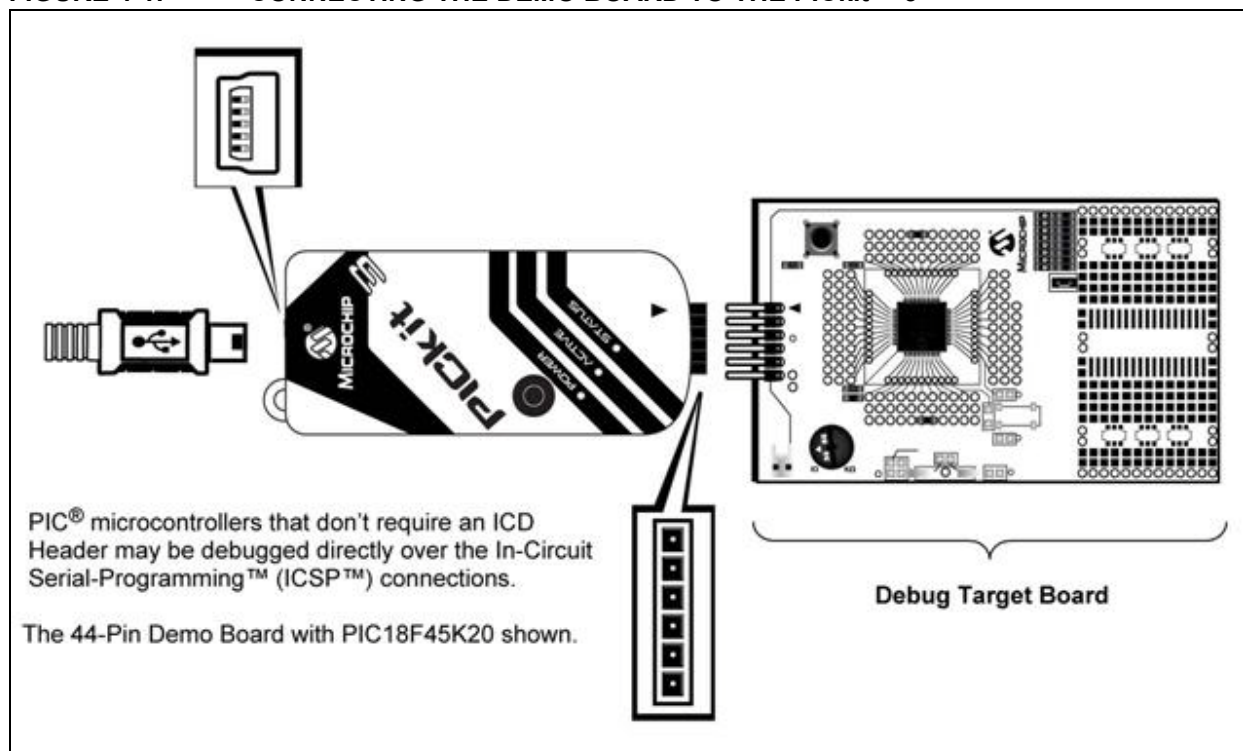
4.3.3 Configuration Bits and Debug Express

PIC microcontroller devices that do not require an ICD Header and may be debugged directly contain a 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 X IDE when using PICkit 3 Debug Express and should not be specified in source code configuration settings.

CAUTION
The 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 at all, 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 X IDE Configuration Bits dialog, these bits are usually referred to as the "Comm Channel Select" bits.

Chapter 5. PICkit 3 Programmer-To-Go

5.1 INTRODUCTION

The PICkit 3 Programmer-To-Go functionality allows a PIC MCU memory image to be downloaded into the PICkit 3 for programming later 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

5.2 USB POWER FOR PICKIT 3 PROGRAMMER-TO-GO

The PICkit 3 programmer hardware does not have the capability to be powered entirely 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 (strictly for power, not for communication)
- 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

5.2.1 Power Requirements

The USB power source should meet the following minimum criteria:

- supplies at least 100 mA of current to the PICkit 3 unit
- provides a steady, regulated 4.5-5.5V 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.

5.3 PICKIT 3 PROGRAMMER-TO-GO SUPPORTED DEVICES

All devices in the following families that are supported by the PICkit 3 with MPLAB X IDE are also supported for Programmer-To-Go operation. Table 5-1 lists supported device families and program memory limitations.

TABLE 5-1: **PROGRAMMER-TO-GO SUPPORTED DEVICES**

Supported Families	Supported Parts
Baseline	All[1]
Midrange	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]
PIC32	All

Note 1: Supports all family parts that are supported by MPLAB X IDE. See the MPLAB X IDE Start page, click the link for *Release Notes and Support Documentation*, then click on Device Support.htm 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 method

5.4 SETTING UP PICKIT 3 FOR PROGRAMMER-TO-GO OPERATION

Before downloading a memory image to PICkit 3 for Programmer-To-Go functionality, 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 by programming a target device from the software first, with all desired options, to ensure that the device programs as expected before downloading an image to Programmer-To-Go. Refer to the MPLAB X IDE online help for information on using MPLAB X IDE to program a device.

> **Note:** The PICkit 3 is NOT a production programmer. It should be used for development purposes only. The Programmer-To-Go operation offers programming portability for field environments, not for production purposes.

5.4.1 Set Up Programmer-to-Go

From MPLAB X IDE, follow these steps to set up the Programmer-To-Go option:

1. Click on your project and select _File>Project Properties_. In the Hardware Tool area, click on the serial number (SN) for the PICkit 3 you want to use for your project. Click **Apply**.

FIGURE 5-1: **PICKIT 3 PROJECT PROPERTIES**
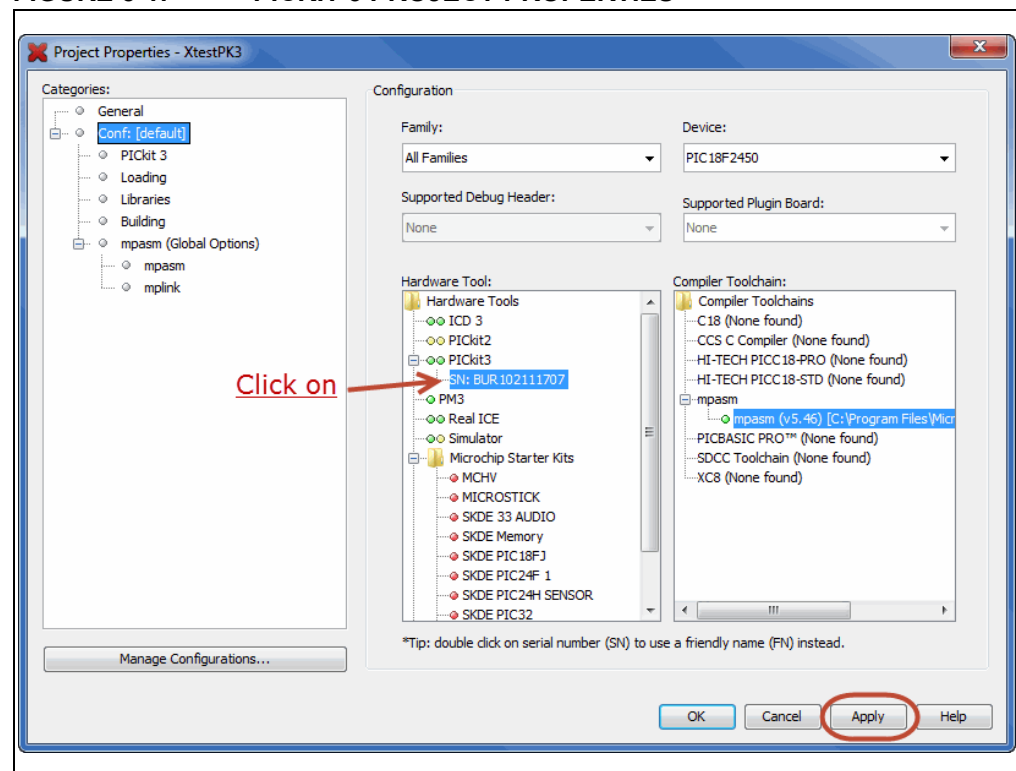
2. On the left side of the dialog under Categories, click on "PICkit 3". Now you can select "Programmer To Go" from the Options categories drop-down menu. See Figure 5-2.

> **Note:** The Preserve Memories option is not supported in Programmer-To-Go mode.

3. In the "Image Name" field, type in the name you want to use for your program-
    ming image.

FIGURE 5-2: NAME THE IMAGE
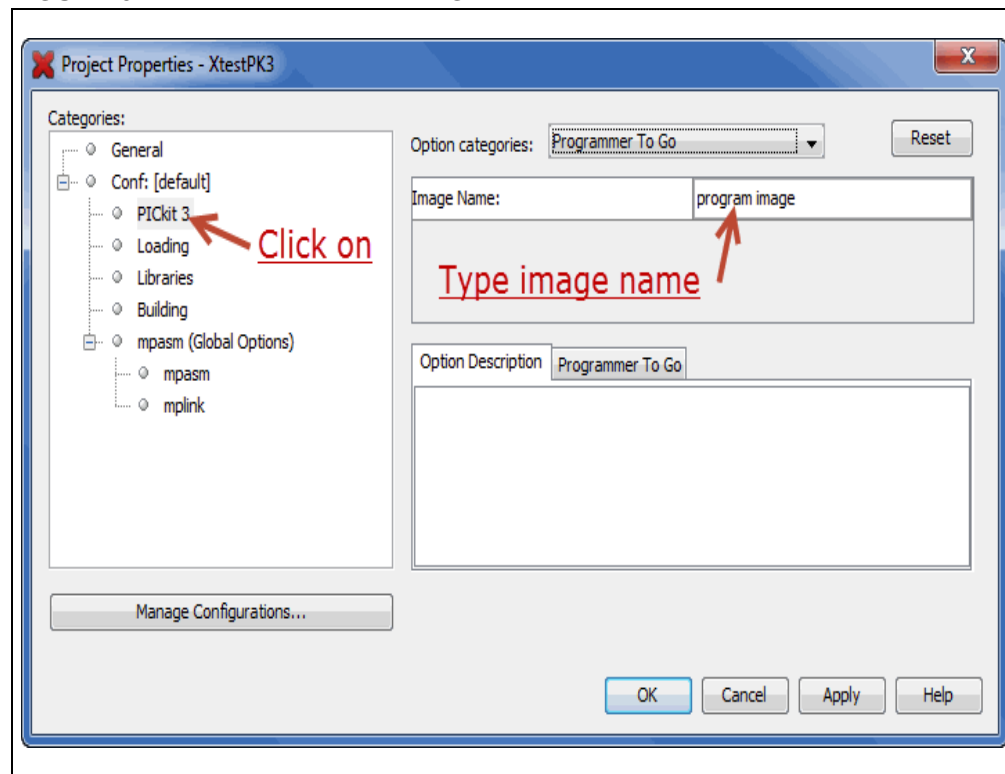
4. Click on the **Programmer To Go** tab shown in Figure 5-3 to display the setting
    you've selected for programming the device. If you want to change any of these
    settings, use the Project Properties dialog.

FIGURE 5-3: Programmer To Go Tab
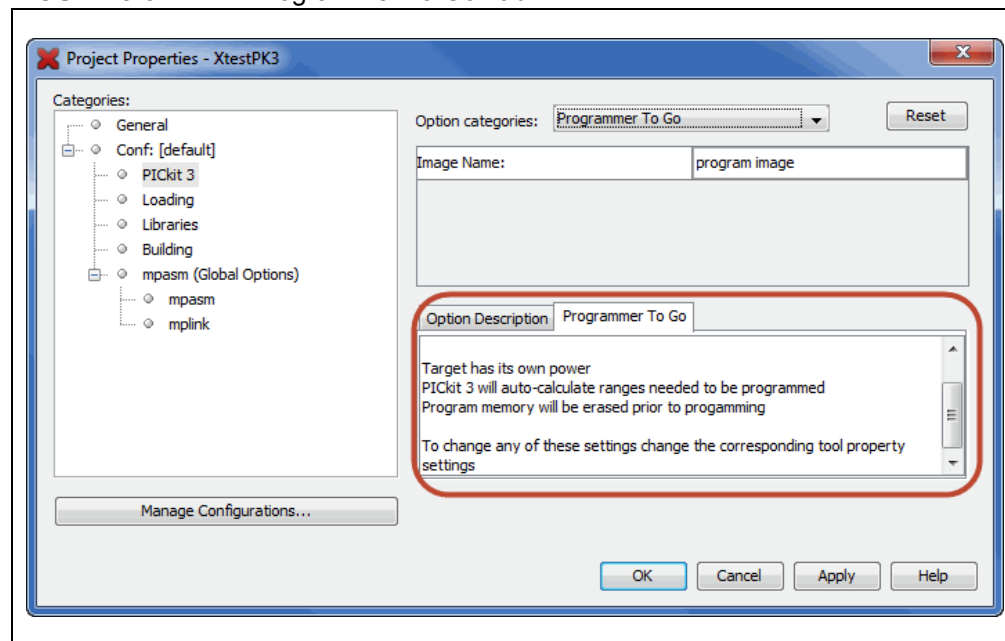
5. Click **OK**.

5.4.2 Download to PICkit 3 Complete

Use the down arrow next to the Make and Program Device icon and select "Programmer To Go PICkit3" (see Figure 5-4) to execute an image transfer to the PICkit 3 unit. Once the image is stored in the PICkit 3, you no longer need MPLAB X 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.

FIGURE 5-4:   DOWNLOAD TO PICKIT 3
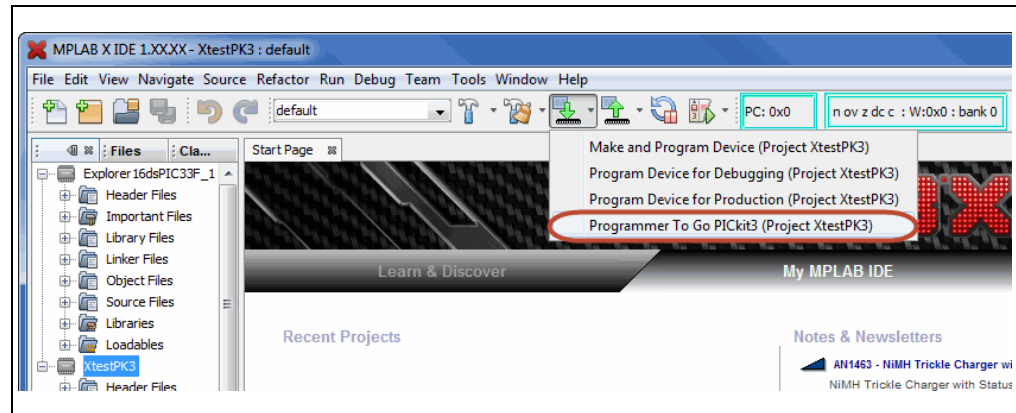
After you download the image, the Output window displays a message when the download is complete:

> "PICkit 3 is now in Programmer to go mode. The next time you connect to this unit, you will have the choice to take it out of Programmer to go mode."

The PICkit 3 unit's "Active" LED should be blinking to indicate it is in Programmer-To-Go mode and ready to program.

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.

MPLAB® X PICkit™ 3 User's Guide

5.5   USING PICKIT 3 PROGRAMMER-TO-GO

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

1. Connect a USB power source as discussed in **Section 5.2 "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 it is not powering from the PICkit 3.
4. Press the PICkit 3 push button to begin programming.

During the programming operation, the PICkit 3 "Status" LED shows orange and remains lit while the operation is performed.

When the programming operation has completed, the PICkit 3 unit provides feedback on the operation via the unit LEDs. A green "Status" LED indicates a successful operation. Red indicates a programming failure. See Table 5-2 for the feedback codes.

TABLE 5-2:   PROGRAMMER-TO-GO OPERATION FEEDBACK CODES

LED Status		Interpretation	
"Active" LED	"Status" LED	Code	Description
Blue Single blink	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 2.5 "Communication Connections"**) for VDD and VPP information.
Off	Red 2 blinks in succession: •• •• •• •• ••	Device ID Error	PICkit 3 received an unexpected Device ID from the target. Check that the target part matches the one that was selected when the PICkit 3 Programmer-To-Go was set up. This error may indicate that a bad ICSP connection is 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 pushbutton to clear the error code and initiate a new programming operation.

5.6   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 X IDE. Select *Run>Run Project* and the following message displays:

"PICkit 3 is in Programmer-to-go mode. The name of the stored image is: _____. Do you want to stay in Programmer-to-go mode? (Saying no will erase the image in the PICkit 3.)"

Select **No** to erase the image and exit Programmer-To-Go mode.

NOTES:

Part 2 – Troubleshooting

NOTES:

Chapter 6. Troubleshooting First Steps

6.1   INTRODUCTION

If you are having problems with PICkit 3 In-Circuit Debugger/Programmer operation, start here.

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

6.2   THE 5 QUESTIONS TO ANSWER FIRST

1. Which device are you working with?
   Often an upgrade to a newer version of MPLAB X 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 regarding resistors/capacitors for communications connections? See **Chapter 2. "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 debugger?
   If you have used a longer cable, it could cause communications errors.

6.3   TOP 10 REASONS WHY YOU CAN'T DEBUG

1. The oscillator is not working.
   Check your Configuration bits settings for the oscillator. If you are using an external oscillator, try using an internal oscillator. If you are using an internal PLL, make sure your PLL settings are correct.
2. The target board is not powered.
   Check the power cable connection.
3. The V_{DD} voltage is outside the specifications for this device.
   See the device programming specification for details.
4. The debugger has become physically disconnected from the PC and/or the target board.
   Check the connections of the communications cables.
5. The device is code-protected.
   Check your Configuration bit's setting for code protection.
6. Debugger to PC communication has been interrupted.
   Reconnect to the debugger in MPLAB X IDE.

7. The production device that you are trying to debug does not have debugging capabilities.
   Use a debug header instead. (See the "*Processor Extension Pak and Header Specification*" that is mentioned in **"Recommended Reading"**).

8. The target application has somehow become corrupted or contains errors.
   For example, the regular linker script was used in the project instead of the debugger version of the linker script (e.g., 18F8722.lkr was used instead of 18F8722i.lkr). Try rebuilding and reprogramming the target application. Then, initiate a Power-On-Reset of the target.

9. You do not have the correct PGC/PGD pin pairs programmed in your Configuration bits (for devices with multiple PGC/PGD pin pairs).

10. 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.

11. Brown-Out Detect voltage is greater than the operating voltage V_{DD}.
    This means the device is in Reset and cannot be debugged.

12. The communication connection guidelines in **Chapter 2. "Operation"** were not followed.

13. The debugger cannot always perform the action requested.
    For example, the debugger cannot set a breakpoint if the target application is currently running.

6.4 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 boards, which are known to work, to verify correct PICkit 3 In-Circuit Debugger/Programmer functionality.

5. Review debugger operation to ensure proper application setup.
   For more information, see **Chapter 2. "Operation"**.

6. If the problem persists, contact Microchip Support.

Chapter 7. Frequently Asked Questions (FAQs)

7.1 INTRODUCTION

Look here for answers to frequently asked questions about the PICkit 3 In-Circuit Debugger/Programmer system.

- How Does It Work
- What's Wrong

7.2 HOW DOES IT WORK

- **What's in the silicon that allows it to communicate with the PICkit 3 In-Circuit Debugger/Programmer?**

  PICkit 3 In-Circuit Debugger/Programmer 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 In-Circuit Debugger/Programmer compare with other in-circuit emulators/debuggers?**

  Please refer to **Section 2.2 "Tools Comparison"**.

- **Does the PICkit 3 In-Circuit Debugger/Programmer have complex breakpoints like other in-circuit emulators/debuggers?**

  No. But you can break based on a value in a data memory location or program address.

- **Is the PICkit 3 opto isolated 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 speed that is specified in the device 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 data sheet.

- **What is the function of pin 6, the LVP pin?**

  Pin 6 is reserved for the LVP (Low-Voltage Programming) connection.

7.3   WHAT'S WRONG

- **Performing a Verify fails after programming the device. Is this a programming issue?**

  If 'Run' (*Run>Run Program*) is selected, the device will automatically run immediately after programming. Therefore, if your code changes the flash memory, verification could fail. To prevent the code from running after programming, please select 'Hold in Reset'.

- **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.

- **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. Program only those "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 that are 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 2.7.1 "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.

- **How can I manually download the firmware?**

  You can download it manually. Select F*ile>Project Properties*. Under Categories, click "PICkit 3", and select "Firmware" from the Option Categories drop-down. Uncheck "Use Latest Firmware" and click on "Press to browse for a specific firmware version". Browse for the Firmware File, located in the Directories pane under MPLABX\mplab_ide\mplablibs\modules\ext\PICKIT3.jar. In the Firmware Files pane, select the `.jam` file you want and click **OK**. Click **Reset** on the Project Properties dialog.

- **I accidentally disconnected my PICkit 3 while firmware was downloading. What do I do now?**

  Reconnect the PICkit 3. It will begin to erase what had been written so it can restart. This erasing will take about 7 seconds. Please be patient. The LEDs are all on during this process. When it is done, MPLAB X IDE will recognize the device and start the recovery process, i.e., begin firmware download.

- **My memory window does not reflect changes**

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

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

  Try the following resources:

  - **Section 2.9 "Resources Used by the Debugger"**
  - **Section 8.3 "General Corrective Actions"**
  - **Section 8.2 "Specific Error Messages"**
  - **Chapter 9. "Engineering Technical Notes (ETNs)"**

NOTES:

Chapter 8. Error Messages

8.1 INTRODUCTION

The PICkit 3 In-Circuit Debugger/Programmer produces various error messages; some are specific and others can be resolved with general corrective actions. In general, read any instructions under your error message. If these fail to fix the problem or if there are no instructions, refer to the following sections.

- Specific Error Messages
- General Corrective Actions

8.2 SPECIFIC ERROR MESSAGES

8.2.1 Debugger-to-Target Communication Errors

Failed to send database

If you receive this error:

1. Try downloading again. It may be a one-time error.
2. Try manually downloading the highest-number `.jam` file.

If these actions fail to fix the problem, see **Section 8.3.2 "Debugger-to-Target Communication Error Actions"**.

8.2.2 Corrupted/Outdated Installation Errors

Failed to download firmware

If the Hex file exists:

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

If the Hex file does not exist:

Reinstall MPLAB X IDE.

Unable to download debug executive

If you receive this error while attempting to debug:

1. Deselect the debugger as the debug tool.
2. Close your project, and then close MPLAB X IDE.
3. Restart MPLAB X IDE, and re-open your project.
4. Select the debugger as the debug tool, and try to debug the target device again.

Unable to download program executive

If you receive this error while attempting to program:

1. Deselect the debugger as the programmer.
2. Close your project, and then close MPLAB X IDE.
3. Restart MPLAB X IDE, and re-open your project.
4. Select the debugger as the programmer, and try to program the target device again.

If these actions fail to fix the problem, see **Section 8.3.4 "Corrupted Installation Actions"**.

8.2.3   Debug Failure Errors

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 if you try to Run before programming your device. If you receive this message after trying to Run, or immediately after programming your device, please refer to **Section 8.3.6 "Debug Failure Actions"**.

The 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 settings for code protection.

Disable code protection, set or clear the appropriate Configuration bits in code or in the Configuration Bits window according to the device data sheet. Then erase and reprogram the *entire* device.

8.2.4   Miscellaneous Errors

PICkit 3 is busy. Please wait for the current operation to finish.

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 X IDE.

8.3   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

8.3.1   Read/Write Error Actions

If you receive a read or write error:

1. Did you hit Abort? This can produce read/write errors.
2. Try the action again. It could 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 being used are of the correct length.

8.3.2 Debugger-to-Target Communication Error Actions

The PICkit 3 In-Circuit Debugger/Programmer 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) in use are the correct length.

8.3.3 Debugger-to-PC Communication Error Actions

The PICkit 3 In-Circuit Debugger/Programmer and MPLAB X 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 that the error was a one time glitch.
3. The version of MPLAB X IDE installed may be incorrect for the version of firmware loaded on the PICkit 3 In-Circuit Debugger/Programmer. Follow the steps outlined in **Section 8.3.4 "Corrupted Installation Actions"**.

8.3.4 Corrupted Installation Actions

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

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

8.3.5 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 In-Circuit Debugger/Programmer
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.

8.3.6 Debug Failure Actions

The PICkit 3 In-Circuit Debugger/Programmer was unable to perform a debugging operation. There are numerous reasons why this might occur. See **Section 6.3 "Top 10 Reasons Why You Can't Debug"** and **Section 6.4 "Other Things to Consider"**.

8.3.7 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 8.3.4 "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.

NOTES:

Chapter 9. Engineering Technical Notes (ETNs)

The following ETNs are related to the PICkit 3. Please go to the www.microchip.com site, PICkit 3 In-Circuit Debugger page and click on the ETN in the Downloads section for details.

- **ETN-32 PICkit 3 Operation at Low Voltage - Modification:** Applies to Assembly #10-00424-R4 or below.

NOTES:

MPLAB® X PICkit™ 3 USER'S GUIDE

Part 3 – Reference

NOTES:

Appendix A. Hardware Specification

A.1 INTRODUCTION

The hardware and electrical specifications of the PICkit 3 In-Circuit Debugger/Programmer system are detailed.

A.2 HIGHLIGHTS

This chapter discusses:

• Declaration of Conformity
• USB Port/Power
• PICkit 3 In-Circuit Debugger/Programmer
• 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 In-Circuit Debugger/Programmer

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 In-Circuit Debugger/Programmer

Due to the special nature of the PICkit 3 In-Circuit Debugger/Programmer, 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 In-Circuit Debugger/Programmer 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 In-Circuit Debugger/Programmer 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 IN-CIRCUIT DEBUGGER/PROGRAMMER

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 of SPI serial EE for programming into the on-board Flash emulation device 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.
	Orange	Lit when an operation is busy.
	Red	Lit when the debugger has failed.

A.6 STANDARD COMMUNICATION HARDWARE

For standard debugger communication with a target (**Section 2.4 "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 6-pin SIL connector, so these headers may be connected directly to the debugger.

For more on available header boards, see the "*Processor Extension Pak and Header 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 to 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 In-Circuit Debugger/Programmer system)
- Clock and data signals are protected from high voltages caused by faulty target 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

Standard Socket

Bottom of Target Board
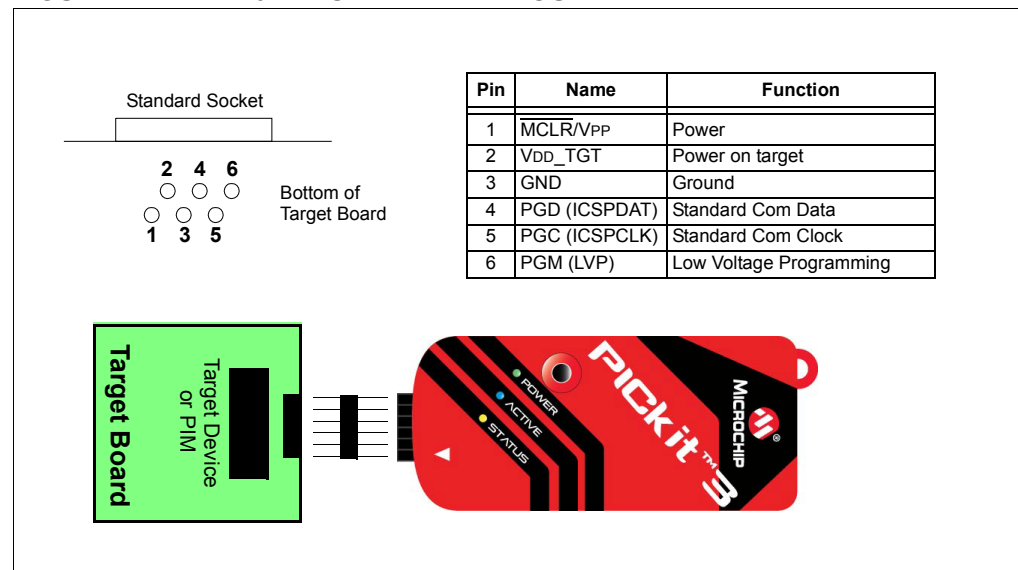
A.6.2 Modular Cable and Connector

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	V_{DD} Target
1	V_{PP}

Front view of Modular Connector on Target Board

Bottom view of Modular Connector Pinout on Target Board

AC164110 Adapter

PICkit 3

Mini-USB cable to PC

Device-ICE

Power

Header

Transition Socket
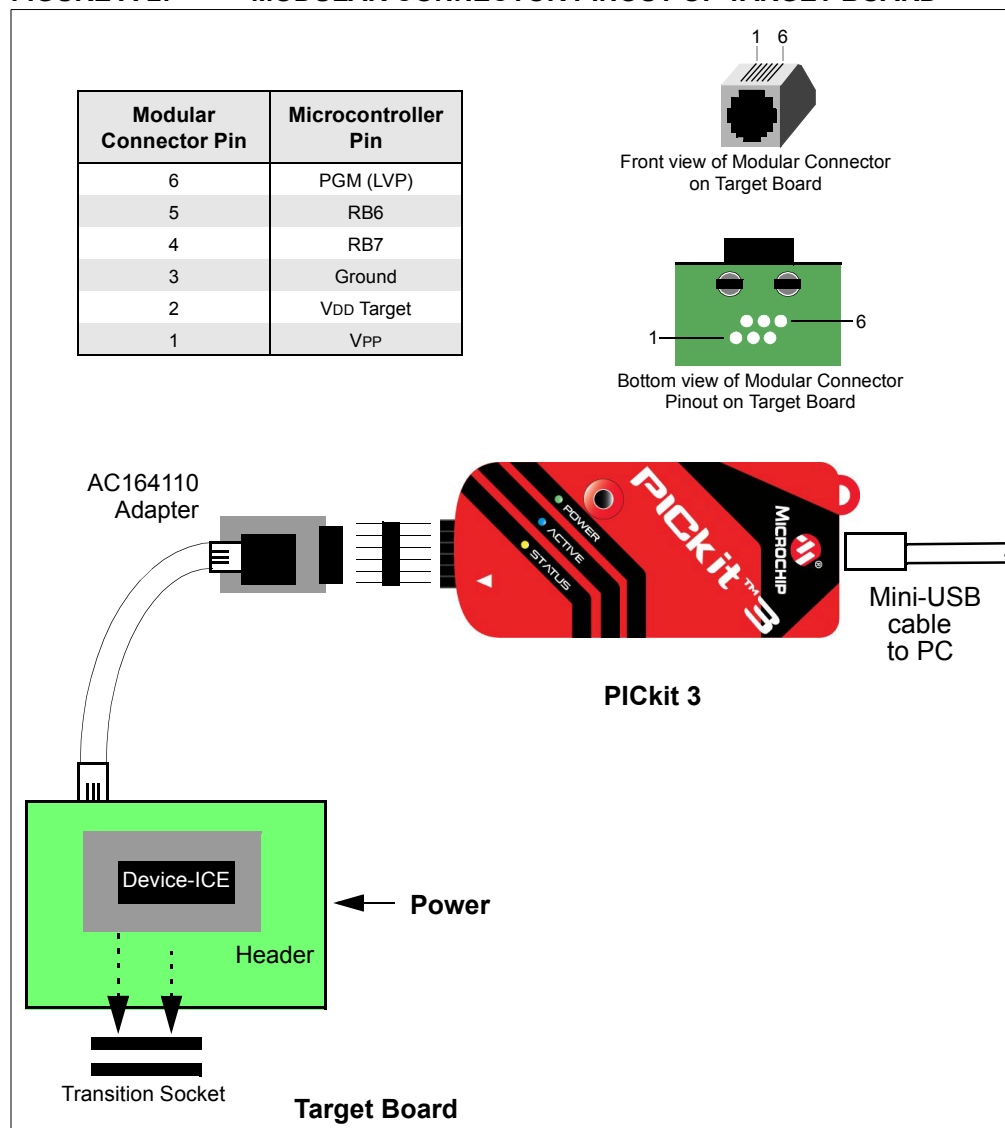
Target Board

A.6.2.2   MODULAR PLUG SPECIFICATION

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

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 2.5.2 "Target Connection Circuitry"**
- **Section 2.5.5 "Circuits That Will Prevent the Debugger From Functioning"**

NOTES:

MICROCHIP

MPLAB® X PICkit™ 3 USER'S GUIDE

Appendix B. PICkit 3 Schematics

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

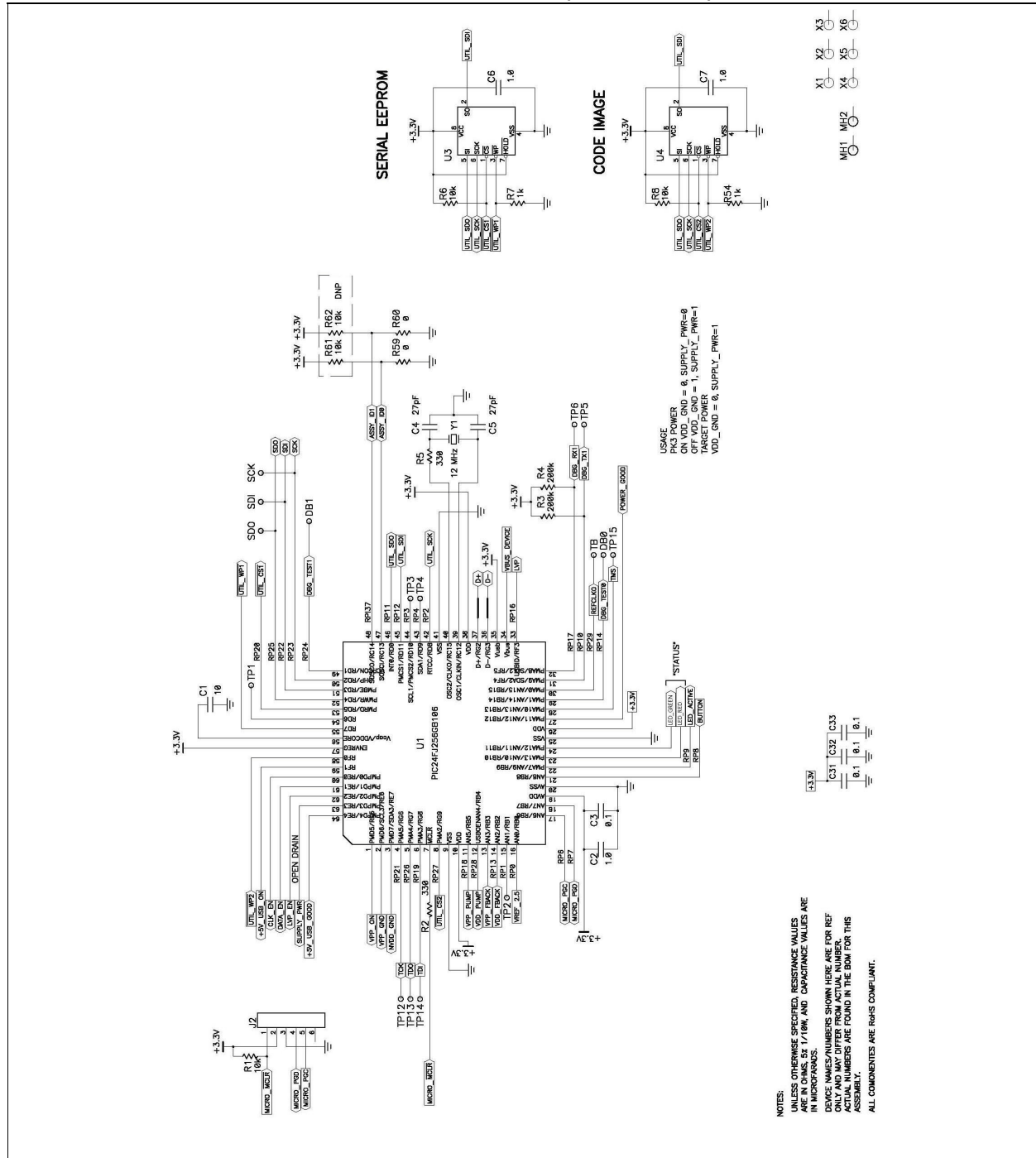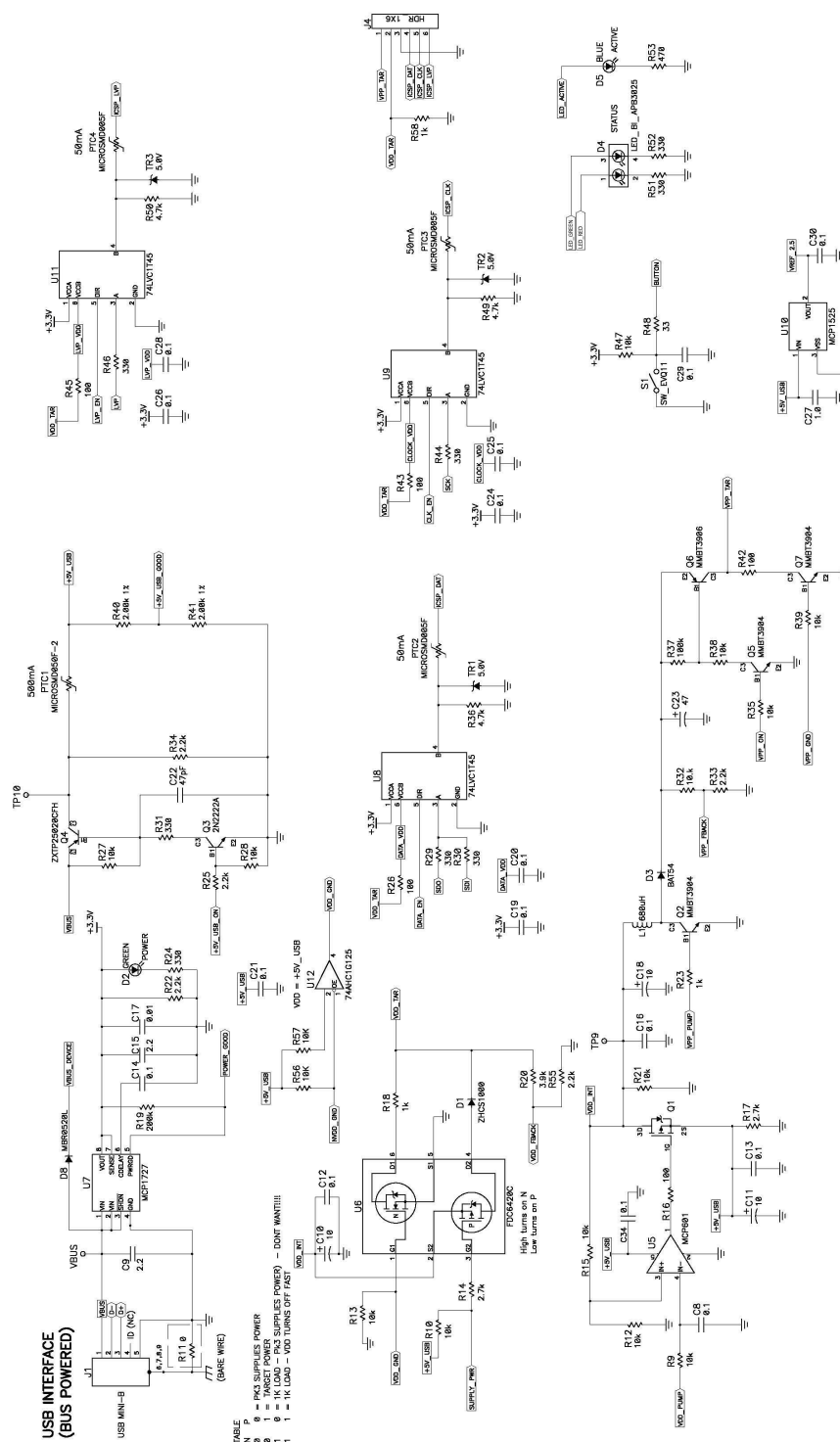**FIGURE B-1:**   PICkit™ 3 SCHEMATIC DIAGRAM (PAGE 1 OF 2)

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

Appendix C. Revision History

Revision A (April 2013)

This is the initial release of this document.

NOTES:

Glossary

A

Absolute Section

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

Absolute Variable/Function

A variable or function placed at an absolute address using the OCG compiler's @ *address* syntax.

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

16-bit C Compiler **–** An unnamed structure.

PIC18 C Compiler **–** 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/Archiver

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

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 lower case letters, digits, symbols and control characters.

Assembly/Assembler

Assembly is a programming language that describes binary machine code in a symbolic form. An assembler is a language tool that translates assembly language source code into machine code.

Assigned Section

A GCC compiler 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

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

Attribute, Section

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

B

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.

Select Toggle Bookmarks on the Editor toolbar to add/remove bookmarks. Click other icons on this toolbar to move to the next or previous bookmark.

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

C\C++

C is a general-purpose programming language which features economy of expression, modern control flow and data structures, and a rich set of operators. C++ is the object-oriented version of C.

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

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.

Compiled Stack

A region of memory managed by the compiler in which variables are statically allocated space. It replaces a software or hardware stack when such mechanisms cannot be efficiently implemented on the target device.

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.

D

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.

Data Monitor and Control Interface (DMCI)

The Data Monitor and Control Interface, or DMCI, is a tool in MPLAB X IDE. The interface provides dynamic input control of application variables in projects. Application-generated data can be viewed graphically using any of 4 dynamically-assignable graph windows.

Debug/Debugger

See ICE/ICD.

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 A digital signal controller (DSC) is a microcontroller device with digital signal processing capability, i.e., Microchip dsPIC DSC devices.

Digital Signal Processing\Digital Signal Processor

Digital signal processing (DSP) is the computer manipulation of digital signals, commonly analog signals (sound or image) which have been converted to digital form (sampled). A digital signal processor is 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.

DWARF

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

E

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.

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/Emulator

See ICE/ICD.

Endianness

The ordering of bytes in a multi-byte object.

Environment

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/Error File

An error reports a problem that makes it impossible to continue processing your program. When possible, an error identifies the source file name and line number where the problem is apparent. An error file contains error messages and diagnostics generated by a language tool.

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/MPLAB X 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 (PIC18 MCUs)

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.

F

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.

Fixup

The process of replacing object file symbolic references with absolute addresses after relocation by the linker.

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.

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>`.

G

GPR

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

H

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\Hex File

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

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.

I

ICE/ICD

In-Circuit Emulator/In-Circuit Debugger: A hardware tool that debugs and programs a target device. An emulator has more features than an debugger, such as trace.

In-Circuit Emulation/In-Circuit Debug: The act of emulating or debugging with an in-circuit emulator or debugger.

-ICE/-ICD: A device (MCU or DSC) with on-board in-circuit emulation or debug circuitry. This device is always mounted on a header board and used to debug with an in-circuit emulator or debugger.

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, as in MPLAB IDE/MPLAB X IDE.

Identifier

A function or variable name.

IEEE

Institute of Electrical and Electronics Engineers.

Import

Bring data into the MPLAB IDE/MPLAB X 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. Also known as ISO.

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 Service Request (IRQ)

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 (ISR)

Language tools – A function that handles an interrupt.

MPLAB IDE/MPLAB X 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.

L

L-value

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

Latency

The time between an event and its response.

Library/Librarian

See Archive/Archiver.

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 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.

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.

M

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/MPLAB X IDE, i.e., with a `make`.

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

For C compilers, a representation of the memory available to the application. For the PIC18 C compiler, 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.

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.

Module

The preprocessed output of a source file after preprocessor directives have been executed. Also known as a translation unit.

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 in-circuit debugger that works with MPLAB IDE/MPLAB X IDE. *See* ICE/ICD.

MPLAB IDE/MPLAB X IDE

Microchip's Integrated Development Environment. MPLAB IDE/MPLAB X 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/MPLAB X IDE or stand-alone. Replaces PRO MATE II.

MPLAB REAL ICE™ In-Circuit Emulator

Microchip's next-generation in-circuit emulator that works with MPLAB IDE/MPLAB X IDE. *See* ICE/ICD.

MPLAB SIM

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

MPLIB™ Object Librarian

Microchip's librarian that can work with MPLAB IDE/MPLAB X 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/MPLAB X IDE, though it does not have to be.

MRU

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

N

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/MPLAB X IDE project component.

Non-Extended Mode (PIC18 MCUs)

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/MPLAB X 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.

O

Object Code/Object File

Object code is the machine code generated by an assembler or compiler. An object file is 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.

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.

P

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 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.

Plug-ins

The MPLAB IDE/MPLAB X 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

The enclosure for an in-circuit emulator or debugger. Other names are "Puck", if the enclosure is round, and "Probe", not be confused with logic probes.

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.

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.

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

16-bit assembler – 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.

Program Memory

MPLAB IDE/MPLAB X 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.

16-bit assembler/compiler – 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.

Psect

The OCG equivalent of a GCC section, short for program section. A block of code or data which is treated as a whole by the linker.

PWM Signals

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

Q

Qualifier

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

R

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.

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

16-bit assembler – 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.

Runtime Watch

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

S

Scenario

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

Section

The GCC equivalent of an OCG psect. A block of code or data which is treated as a whole by the linker.

Section Attribute

A GCC 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.

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/MPLAB X 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/MPLAB X 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 (SFRs)

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 dynamically allocated at runtime by instructions in the program. It allows for reentrant function calls.

Stack, Compiled

A region of memory managed and allocated by the compiler in which variables are statically assigned space. It replaces a software stack when such mechanisms cannot be efficiently implemented on the target device. It precludes reentrancy.

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/MPLAB X 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/MPLAB X 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."

T

Target

Refers to user hardware.

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/MPLAB X 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/MPLAB X 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.

U

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.

Glossary

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.

V

Vector

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

Volatile

A variable qualifier which prevents the compiler applying optimizations that affect how the variable is accessed in memory.

W

Warning

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

16-bit assembler/compiler – 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 (WDT)

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.

Workbook

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

NOTES:

Index

MPLAB® X PICkit™ 3 User's Guide

W

Index

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://www.microchip.com/support
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

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

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: 86-10-8569-7000
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 - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2943-5100
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: 86-755-8864-2200
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 - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

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: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7828
Fax: 886-7-330-9305

Taiwan - Taipei
Tel: 886-2-2508-8600
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

11/29/12