



**PICkit™ 3**  
**Programmer Application**  
**User's Guide**

---

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

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

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

---

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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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

#### **Trademarks**

The Microchip name and logo, the Microchip logo, dsPIC, FlashFlex, KEELQ, KEELQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC<sup>32</sup> logo, rfPIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MTP, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.


Analog-for-the-Digital Age, Application Maestro, BodyCom, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Omniscent Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, REAL ICE, rLAB, Select Mode, SQI, Serial Quad I/O, Total Endurance, TSHARC, UniWinDriver, WiperLock, ZENA and Z-Scale are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

GestIC and ULPP are registered trademarks of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

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

© 2013, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

ISBN: 978-1-62077-170-9

**QUALITY MANAGEMENT SYSTEM**  
**CERTIFIED BY DNV**  
**= ISO/TS 16949 =**

*Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

## Table of Contents

<b>Preface .....</b>	<b>5</b>
<b>Chapter 1. Overview</b>	
1.1 Introduction .....	9
1.2 PICKit 3 Programmer Application Defined .....	9
1.3 PICKit 3 Programmer Tool Defined .....	10
<b>Chapter 2. Getting Started</b>	
2.1 Introduction .....	13
2.2 Installing the PICKit 3 Hardware .....	13
2.3 Installing and Launching the PICKit 3 Programmer Application .....	14
2.4 Connecting to the Device .....	15
2.5 Selecting Target Power .....	17
2.6 Importing a Hex File .....	19
2.7 Writing the Program to the Device .....	19
2.8 Verifying the Device .....	20
2.9 Reading Device Memory .....	21
2.10 Code Protecting the Device .....	21
2.11 Erasing and Blank Checking the Device .....	22
2.12 Automating Write/Read Procedures .....	22
2.13 PICKit 3 Unit ID .....	23
<b>Chapter 3. Using In-Circuit Serial Programming (ICSP)</b>	
3.1 Introduction .....	25
3.2 Isolate VPP/ $\overline{\text{MCLR}}$ /Port Pin .....	26
3.3 Isolate ICSPCLK or PGC and ICSPDAT or PGD pins .....	26
3.4 VDD .....	27
3.5 VSS .....	28
3.6 Cable Lengths .....	28
3.7 Serial EEPROM and KEELOQ HCS Devices .....	28
3.8 Logic Tool .....	28
<b>Chapter 4. PICKit 3 Programmer Application</b>	
4.1 PICKit 3 Programmer Application Dialog .....	29
4.2 PICKit 3 Programmer Application Menu Bar .....	30
4.3 Device Configuration .....	32
4.4 Status Window .....	33
4.5 Progress Bar .....	33
4.6 Device VDD .....	33
4.7 Device $\overline{\text{MCLR}}$ State .....	33
4.8 Memory Source .....	34

# PICkit™ 3 Programmer Application User's Guide

---

4.9 Program Memory .....	34
4.10 Hex File Options .....	34
4.11 EEPROM Data Memory .....	34
<b>Chapter 5. Troubleshooting</b>	
5.1 Introduction .....	35
5.2 Frequently Asked Questions .....	35
<b>Chapter 6. Updating and Reverting the PICkit 3 OS</b>	
6.1 Introduction .....	41
6.2 Updating the PICkit 3 OS to Scripting Mode .....	41
6.3 Reverting the PICkit 3 OS to MPLAB Mode .....	42
<b>Chapter 7. Logic Tool</b>	
7.1 Introduction .....	43
7.2 Logic I/O Mode .....	44
7.3 Configuring the Logic Tool Logic I/O .....	45
7.4 Logic Analyzer Mode .....	48
7.5 The Logic Analyzer Window .....	49
<b>Appendix A. Hardware Specification</b>	
A.1 Introduction .....	59
A.2 Highlights .....	59
A.3 Declaration of Conformity .....	59
A.4 Device Support .....	60
A.5 USB Port/Power .....	60
A.6 PICkit 3 Programmer .....	60
A.7 Standard Communication Hardware .....	62
A.8 Target Board Considerations .....	64
<b>Appendix B. PICkit 3 Schematics</b>	
<b>Glossary .....</b>	<b>67</b>
<b>Index .....</b>	<b>87</b>
<b>Worldwide Sales and Service .....</b>	<b>90</b>

---

---

## 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](http://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 “DSXXXXA”, where “XXXX” is the document number and “A” is the revision level of the document.

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

## INTRODUCTION

This chapter contains general information that will be useful to know before using PICKit™ 3 Programmer Application. Items discussed include:

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

## DOCUMENT LAYOUT

This document describes how to use the PICKit 3 Programmer Application with the PICKit 3 unit as a development tool to program firmware on a target board. The manual layout is as follows:

- **Chapter 1. Overview** – describes the PICKit 3 Programmer Application and how it can help you develop your application.
- **Chapter 2. Getting Started** – Provides installation instructions and setup information.
- **Chapter 3. Using In-Circuit Serial Programming (ICSP)** – Explains using the application with ICSP.
- **Chapter 4. PICKit 3 Programmer Application** – Describes how to use the application.
- **Chapter 5. Troubleshooting** – Provides basic troubleshooting information.
- **Chapter 6. Updating and Reverting the PICKit 3 OS** – Explains how to update or revert the operating system firmware.
- **Chapter 7. Logic Tool** – Provides information on using the Logic Tool.
- **Appendix A. Hardware Specification** – Provides information on conformity, device support, USB power, communication, and target board considerations.
- **Appendix B. PICKit 3 Schematics** – Presents PICKit 3 schematics.

# PICkit™ 3 Programmer Application User's Guide

## CONVENTIONS USED IN THIS GUIDE

The following conventions may appear in this documentation:

### DOCUMENTATION CONVENTIONS

Description	Represents	Examples
<b>Arial font:</b>		
Italic	Referenced books	<i>MPLAB® IDE User's Guide</i>
	Emphasized text	...is the <i>only</i> compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic with right angle bracket	A menu path	<u><i>File&gt;Save</i></u>
Bold characters	A dialog button	Click <b>OK</b>
	A tab	Click the <b>Power</b> tab
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
<b>Courier New font:</b>		
Plain	Sample source code	#define START
	Filenames	autoexec.bat
	File paths	c:\mcc18\h
	Keywords	_asm, _endasm, static
	Command-line options	-Opa+, -Opa-
	Bit values	0, 1
	Constants	0xFF, 'A'
Italic	A variable argument	<i>file.o</i> , where <i>file</i> can be any valid filename
Square brackets [ ]	Optional arguments	mpasmwin [options] <i>file</i> [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) { ... }

## RECOMMENDED READING

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

### **Development Tools Design Advisory (DS51764)**

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

### **README for PICkit™ 3**

For the latest information on using the PICkit 3, read the "Readme for PICkit 3.htm" file in the Readmes subdirectory of the programmer application installation directory. The Readme file contains updated information and known issues that may not be included in this user's guide.

### **PICkit™ 3 Standalone Programmer Application Video**

Watch this online video at [www.microchip.com](http://www.microchip.com) for a quick look at the PICkit 3 Programming Application.

### **PICkit 3 In-Circuit Programmer/Debugger User's Guide/Help (DS51795)**

Consult this document for more information pertaining to the installation and features of the PICkit 3 In-Circuit Programmer/Debugger. An online Help version is also available.

### **MPLAB® or MPLAB X IDE User's Guide/Help (DS51519, DS52027)**

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

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

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

### **MPASM™ Assembler, MPLINK™ Object Linker, MPLIB™ Object Librarian User's Guide (DS30003014)**

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

# PICkit™ 3 Programmer Application User's Guide

---

NOTES:



---

## Chapter 1. Overview

---

### 1.1 INTRODUCTION

**Note:** This document specifically discusses using the PICKit 3 programmer with the PICKit 3 Programmer Application. For information on using the PICKit 3 unit with MPLAB® X IDE software for programming and debugging capabilities, please see the “*PICKit 3 Programmer/Debugger User's Guide*” (DS51795).

This chapter introduces the PICKit 3 Programmer Application.

- PICKit 3 Programmer Application Defined
- PICKit 3 Programmer Tool Defined

### 1.2 PICKIT 3 PROGRAMMER APPLICATION DEFINED

The PICKit 3 Programmer Application is a standalone GUI that provides programming capability to the PICKit 3 (without using the MPLAB X IDE).

The PICKit 3 Programmer Application will program all Microchip 8-, 16-, and 32-bit devices. Supported programming operations include read, write, verify erase, and blank check. A progress bar displays during the operation.

Program memory and data memory may be selected or deselected for programming operations. Programming options include verify on write, clear memory buffers on erase, hold device in Reset, and write on PICKit button. Alert sounds can be set to play on programming events such as success, warning or error.

Once a device has been read, the contents of the device can be saved to a hex file.

Configuration bits can be displayed in an editable window and changes in the values are shown in red.

If the target device is not self powered, the target can be powered by the PICKit 3. Note that target power is automatically enabled during programming operations.

For application installation instructions, see **Chapter 2. “Getting Started”**.

For more information on how to use Microchip the PICKit 3 Programmer Application, see **Chapter 4. “PICKit 3 Programmer Application”**.

# PICKit™ 3 Programmer Application User's Guide

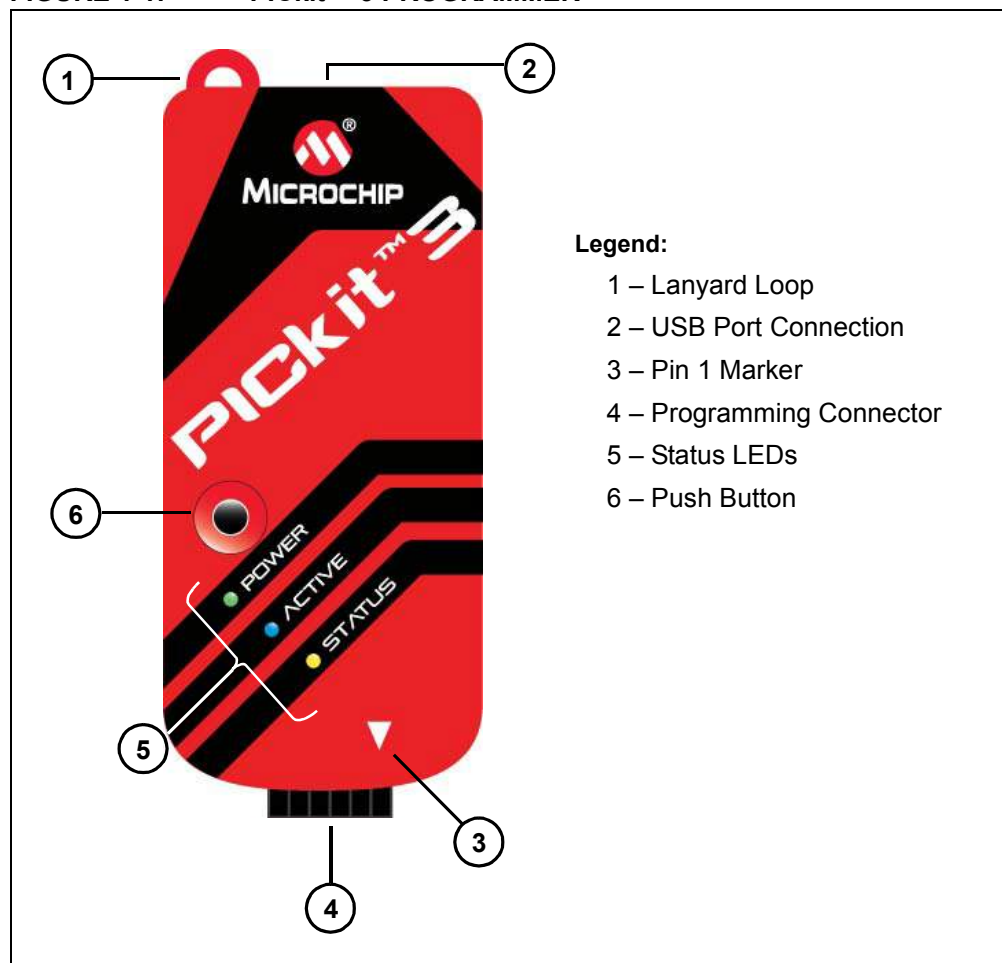
## 1.3 PICKIT 3 PROGRAMMER TOOL DEFINED

The PICKit 3 tool (see Figure 1-1) is a simple, low-cost in-circuit programmer that is capable of programming most of Microchip's Flash microcontrollers and serial EEPROM devices. For specific device support, see the [README](#) file.

**Note:** The PICKit 3 is not a production programmer. It is meant for development purposes only. For production programming, consider the MPLAB PM3 device programmer or other third-party programmers that are designed for a production environment.

Support for new devices can be added by updating the programming software. The latest software is available on Microchip's web site page for the PICKit 3: [www.microchip.com/pickit3](http://www.microchip.com/pickit3).

**FIGURE 1-1: PICKit™ 3 PROGRAMMER**



### 1.3.1 Lanyard Loop

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

### 1.3.2 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.

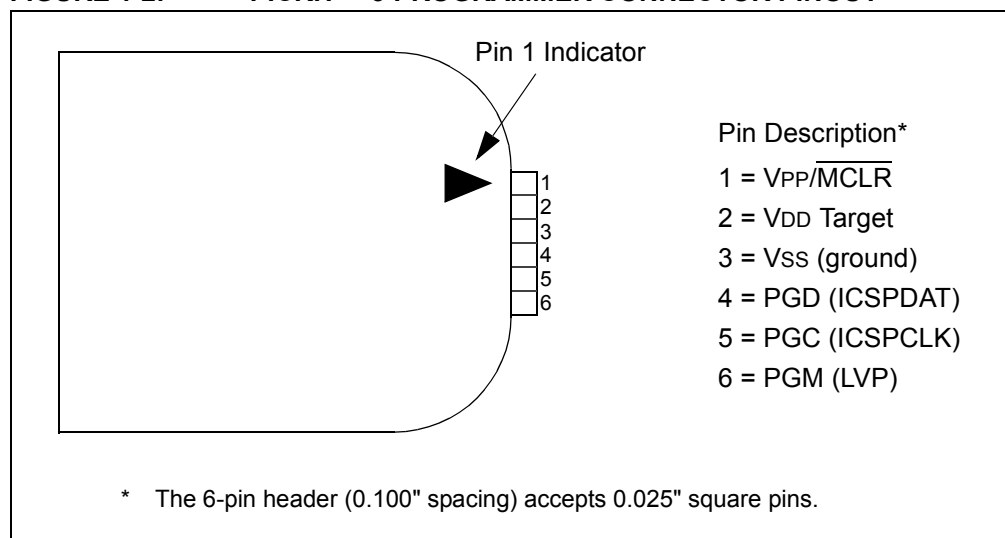
## 1.3.3 Pin 1 Marker

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

## 1.3.4 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**



**Note:** The programming connector pin functions are different for programming Serial EEPROMS and HCS devices. See the ReadMe file for the PICKit 3 ([Help>Readme](#)) included with the PICKit 3 Programmer Application software for these pinouts.

## 1.3.5 Indicator LEDs

The indicator LEDs indicate the status of the PICKit 3.

1. **Power** (green) – power is supplied to the PICKit 3 via the USB port
2. **Active** (blue) – 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

## 1.3.6 Push Button

The push button may be used to initiate the Write Device programming function when Programmer>Write on PICKit Button is checked on the PICKit 3 Programmer Application menu (see item labeled 2 in Figure 1-1.)

The push button may also be used to put the PICKit 3 unit operating system firmware into Bootloader mode. For more information on this feature, see **Chapter 6. “Updating and Reverting the PICKit 3 OS”**.

When using the Logic Tool, the push button can be used to stop the analyzer from running. See **Section 7.5.4.3 “Running the Analyzer”** for more information on using the push button with the logic tool.

# PICkit™ 3 Programmer Application User's Guide

---

NOTES:

---

## Chapter 2. Getting Started

---

### 2.1 INTRODUCTION

This chapter gives instructions on how to get started using the PICKit 3 development programmer to program Flash-based PIC® microcontroller units.

For information on how to use the PICKit 3 with In-Circuit Serial Programming™ (ICSP™), refer to **Chapter 3. “Using In-Circuit Serial Programming (ICSP)”**.

For information on how to update the PICKit 3 operating system (firmware), refer to **Chapter 6. “Updating and Reverting the PICKit 3 OS”**.

- Installing the PICKit 3 Hardware
- Installing and Launching the PICKit 3 Programmer Application
- Connecting to the Device
- Selecting Target Power
- Importing a Hex File
- Writing the Program to the Device
- Verifying the Device
- Reading Device Memory
- Code Protecting the Device
- Erasing and Blank Checking the Device
- Automating Write/Read Procedures
- PICKit 3 Unit ID

### 2.2 INSTALLING THE PICKit 3 HARDWARE

To install the PICKit 3 hardware:

- Plug one end of the USB cable into PICKit 3 USB connector. Plug the other end into a USB port on your PC.
- Connect the PICKit 3 to a target board via a 6-pin in-line connector. The target board can be the included demo board or any target equipped with the appropriate 6-pin connector.
- Do not connect the PICKit 3 to a target board that has its own power supply if it is not connected to a powered USB port.
- To connect the PICKit 3 to a target with an MPLAB ICD 2 style RJ-11 connector, the AC164110 RJ-11 to ICSP Adapter kit is required.

Before connecting the PICKit 3 to a PC via USB, disconnect any target boards that may be attached to the PICKit 3. Similarly, when starting up or rebooting the host PC, ensure that the PICKit 3 is not connected to a target board.

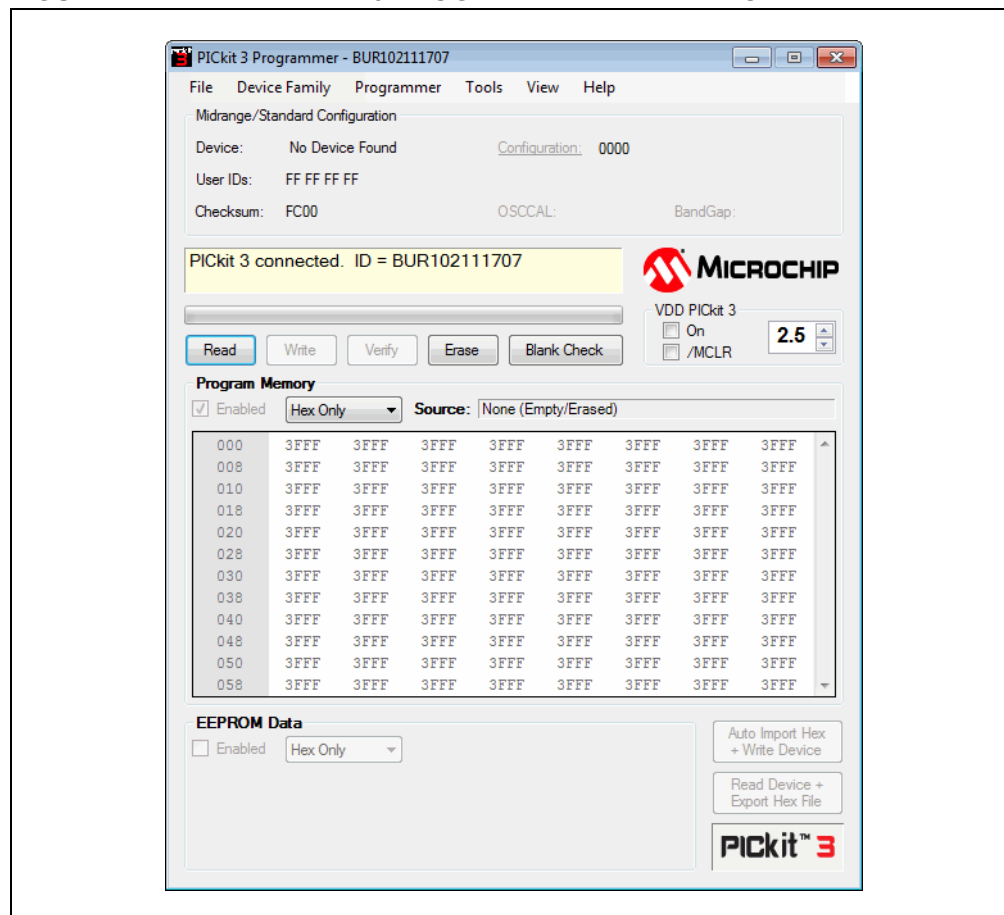
# PICKit™ 3 Programmer Application User's Guide

## 2.3 INSTALLING AND LAUNCHING THE PICKit 3 PROGRAMMER APPLICATION

Get the PICKit 3 Programmer Application from the **Downloads** section of the Microchip web site at [www.microchip.com/pickit3](http://www.microchip.com/pickit3). Follow the installation wizard to install the application. Once installed, start the PICKit 3 Programmer Application by selecting *Start>All Programs>Microchip>PICKit 3*.

A listing of the features and functions may be found in **Section 4.1 “PICKit 3 Programmer Application Dialog”**.

**FIGURE 2-1: PICKIT™ 3 PROGRAMMER APPLICATION**

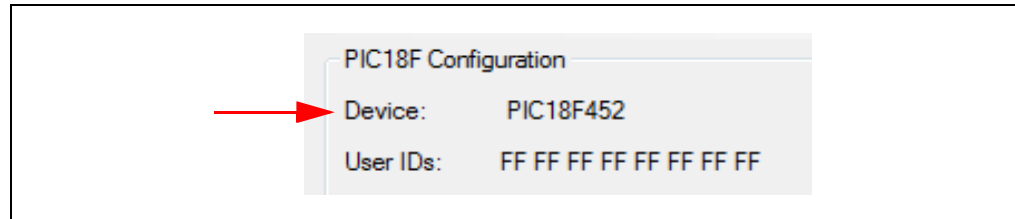


## 2.4 CONNECTING TO THE DEVICE

The PICkit 3 is capable of programming a variety of Flash-based Microchip PIC micro-controllers and serial EEPROM devices. Supported devices are listed in the PICkit 3 (Programmer Application) Readme file, which can also be viewed by selecting [\*Help>Readme\*](#).

When the PICkit 3 Programmer Application is first opened, it will attempt to identify the connected device by the device ID and display it in the Configuration window, as shown in Figure 2-2. The application will automatically download the proper firmware for the selected device.

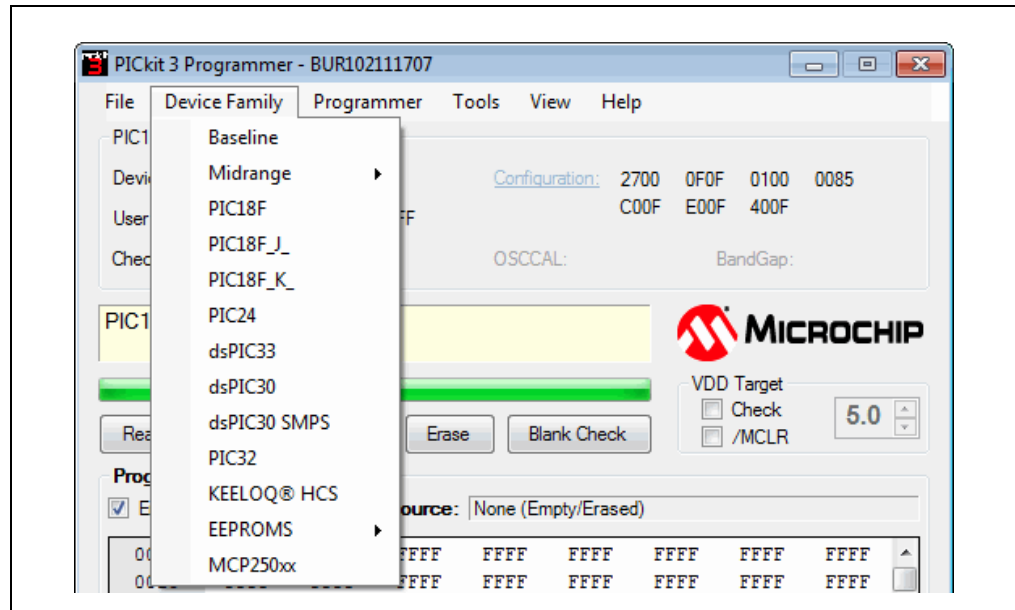
**FIGURE 2-2: IDENTIFY DEVICE**



If the device on the target is not correctly identified, programming operations will not be performed. Check the target power (**Section 2.5 “Selecting Target Power”**) and device ICSP connections before attempting to reselect or change the device.

At any time, the device family may be selected to search for connectivity to a device in that family. To connect to a device when the application is already running, select the device family by clicking on the Device Family menu, as shown in Figure 2-3.

**FIGURE 2-3: SELECT DEVICE FAMILY**



# PICkit™ 3 Programmer Application User's Guide

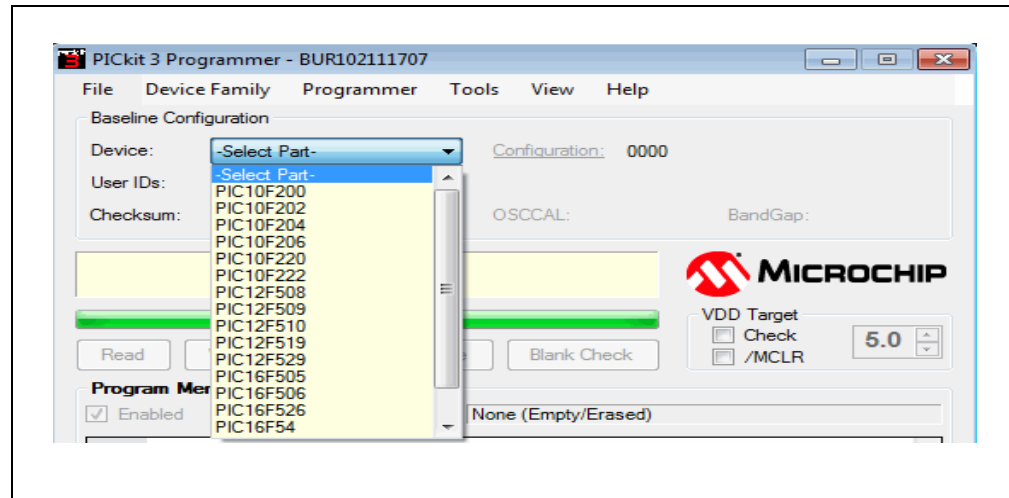
If the Baseline (12-bit core), KEELOQ® HCS or EEPROMs device family is selected, you must select the specific device from the device drop-down box, as shown in Figure 2-4. These devices do not have a device ID and do not support automatic detection.

## CAUTION

Ensure that the correct Baseline has been selected. These devices do not contain a device ID to confirm device selection.

Choosing the wrong Baseline may **cause the erasure of the OSCCAL value** that was stored in the last memory location.

**FIGURE 2-4: SELECT BASELINE FLASH DEVICE**





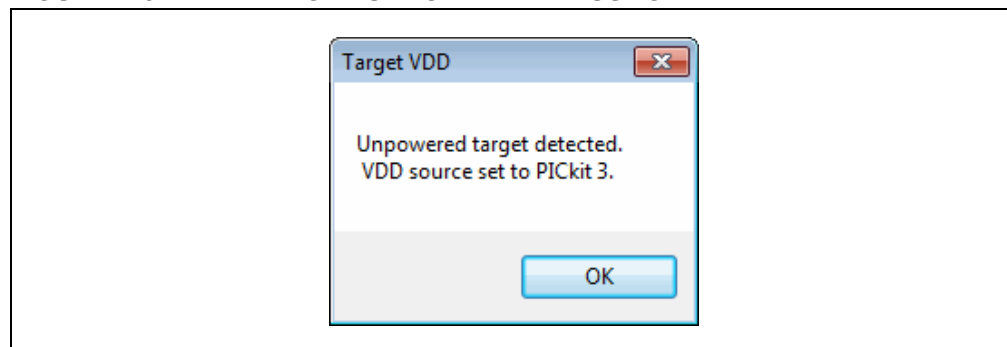
## 2.5 SELECTING TARGET POWER

The PICkit 3 can supply power to the target or the target may be powered externally.

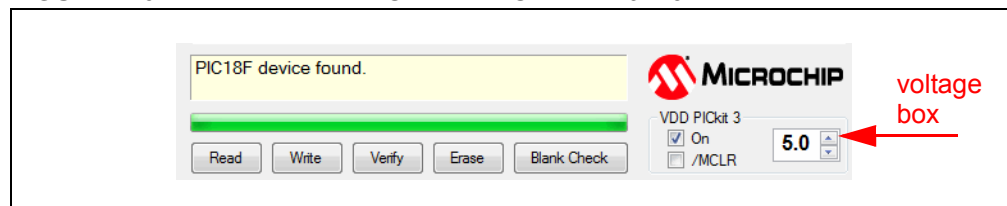
### 2.5.1 Target Powered from PICkit 3

If you are going to power the target board from the PICkit 3, do not attach a power supply to the target or the PICkit 3 will sense it and not give you the option to use PICkit 3 power. For a target board not connected to an external power supply, you will see the message shown in Figure 2-5 and the power options displayed in Figure 2-6.

**FIGURE 2-5: TARGET UNPOWERED MESSAGE**



**FIGURE 2-6: ENABLE POWER FROM PICkit™ 3**



To enable PICkit 3 to power the target device, check the VDD Target Power “On” checkbox as shown. The default setting is off, i.e., the checkbox is cleared.

**Note:** If a target power supply is not detected, the PICkit 3 will always supply power to the target during programming, regardless of the Target Power “On” checkbox state.

The voltage supplied to the target may be adjusted before or after enabling power by adjusting the Target Power voltage box (Figure 2-6).

If a short or heavy current load is detected on the programmer-supplied VDD, then you will receive an error and VDD will be automatically disabled.

### CAUTION

When using bus-powered hubs, the maximum current supplied to the PICkit 3 is 100 mA. If the total of the target and the programmer exceeds this current limit, the USB port might turn off. The target can be powered externally if more power is necessary.

To avoid heavy current load errors, target current consumption should be maintained at a level that is lower 30 mA. In addition, to avoid slowing VDD rise time, large VDD capacitances should not be permitted. Allowed VDD rise time is 500  $\mu$ s or less.

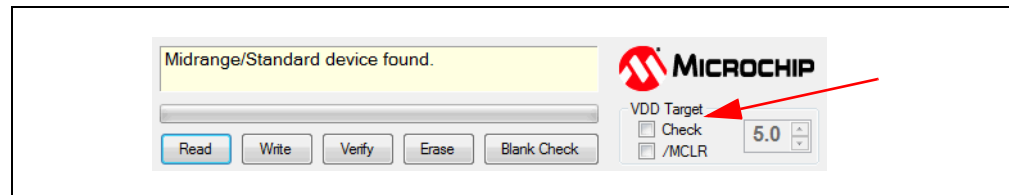
## 2.5.2 Target Powered from External Supply

The target device may also be powered externally. By default, the PICKit 3 will automatically detect an externally powered board. The heading “VDD PICKit 3” will be changed to “VDD Target”, the “On” checkbox will be replaced by a checkbox named “Check”, and the detected VDD voltage is displayed in the grayed out voltage box, as shown in Figure 2-7.

Clicking the “Check” checkbox will update the detected VDD voltage displayed in the voltage box. If no VDD voltage is detected when the checkbox is clicked, then PICKit 3 will return to supplying VDD power to the target device.

**Note:** The maximum external VDD that may be used with the PICKit 3 is 5.5 Volts.  
The minimum external VDD that may be used with the PICKit 3 is 1.8 Volts.

**FIGURE 2-7: EXTERNALLY POWERED TARGET**



## 2.6 IMPORTING A HEX FILE

To import a compiled program (hex file) for it to be programmed into the target device, select *File>Import HEX*.

Browse for the hex file and click **Open**. The code is displayed in the Program Memory and EEPROM Data windows. The name of the hex file is displayed in the Source block under Program Memory.

The PICKit 3 Programmer Application will warn you if the hex file does not contain any Configuration Words.

You will also be warned that the hex file is larger than the selected device if the hex file contains memory locations that do not exist in the current device. Any data for non-existent locations will not be imported.

## 2.7 WRITING THE PROGRAM TO THE DEVICE

After a device family has been selected and a hex file has been imported, the target device is programmed by clicking **Write**. The device is erased and then programmed with the hex code that was imported.

When erasing the device during programming, a Bulk Erase method is used. All Base-line, Mid-Range, and many dsPIC30F and PIC18F devices require a minimum VDD for the Bulk Erase. Some of these devices support a low-voltage row erase method that can be used at lower voltages. However, this method takes longer to erase the device. If a device does not support row erasing, a dialog will pop up to warn you that the device VDD is below the minimum required for a Bulk Erase.

<p><b>Note:</b> If any Code Protect, Data Protect, Write Protect, or Read Protect configuration bits are currently set in the device, the Bulk Erase method must be used prior to programming. The lower voltage row erase procedure will not succeed.</p>
--

The status of the Write operation is displayed in the status bar located under the Device Configuration window. If the write is successful, the status bar turns green and displays "Programming Successful".

If the write fails, the status bar turns red and displays "Programming Failed". This error indicates that the data was corrupted during the programming sequence. If this error is displayed, try writing the program to the device again. If the error continues, see **Chapter 5. "Troubleshooting"** for assistance.

Other write issues may be displayed as warnings and will turn the status bar yellow. In this case, the PICKit 3 and demo board have become disconnected.

## 2.7.1 Writing to Specific Memory Regions

If a device has EEPROM data memory, the “Enabled” checkbox next to Program Memory and EEPROM Data will become available.

The checkboxes select which memory regions’ programming operations will be affected. Refer to Table 2-1 for a description of how programming operations are affected by the checkboxes. Erase and Blank Check always operate on all memory regions.

**TABLE 2-1: MEMORY REGION SELECTION**

Program Memory Enabled	EEPROM Data Enabled	Write/Read/Verify	Erase/Blank Check
Checked	Checked	All Memory Regions	All Memory Regions
Checked	—	Program Memory User IDs Configuration	All Memory Regions
—	Checked	EEPROM only	All Memory Regions
—	—	<i>Not Allowed</i>	

During a Write, regions that are unchecked will remain unchanged in the device.

For example, if Program Memory is unchecked while EEPROM Data is checked, then a Write operation will only write EEPROM Data, while Program Memory, User IDs and Configuration Words in the device will remain unchanged.

If Program Memory is checked while EEPROM Data is unchecked, then a Write operation will program Program Memory, User IDs, and Configuration Words, while EEPROM Data in the device will remain unchanged.

Due to programming constraints in some devices, the PICKit 3 Programmer Application may read and re-write EEPROM data memory during a Write to preserve it.

Both memory regions cannot remain unchecked.

## 2.7.2 Automatic File Reload

Prior to each Write, the imported hex file time stamp is compared to the version on the drive. If the version on the drive is newer, it is reloaded. This comparison only occurs when a hex file has been read from the drive.

The automatic reload feature ensures that the latest version built will be written to the device. It can be used with the *Tools>Program on PICKit Button* feature to program the latest MPLAB Integrated Development Environment (IDE) build, without switching to the PICKit 3 Programmer software, simply by pressing the PICKit 3 unit push button.

## 2.8 VERIFYING THE DEVICE

The Verify function verifies that the program in device memory matches the hex file imported into the PICKit 3 Programmer Application. It compares all areas of memory including program memory, data EEPROM memory, ID, and Configuration bits.

To verify the code, import the hex file and click **Verify**.

Note that a Write operation is automatically verified if *Programmer>Verify on Write* is checked.

If the code is the same, the status bar turns green and displays “Device Verified”. If a discrepancy is found, the status bar turns red and displays where the error is located: “Error in Program Memory, Data EEPROM Memory, or Configuration Bits”.

Table 2-1 illustrates how Verify is affected by the memory region checkboxes.

## 2.9 READING DEVICE MEMORY

To view the code written to the device, click Read.

The code is displayed in the Program Memory and EEPROM Data windows for review. If the display shows all zeros, it is possible that the device is code-protected. (See **Section 2.10 “Code Protecting the Device”**.)

Table 2-1 illustrates how Read is affected by the memory region checkboxes.

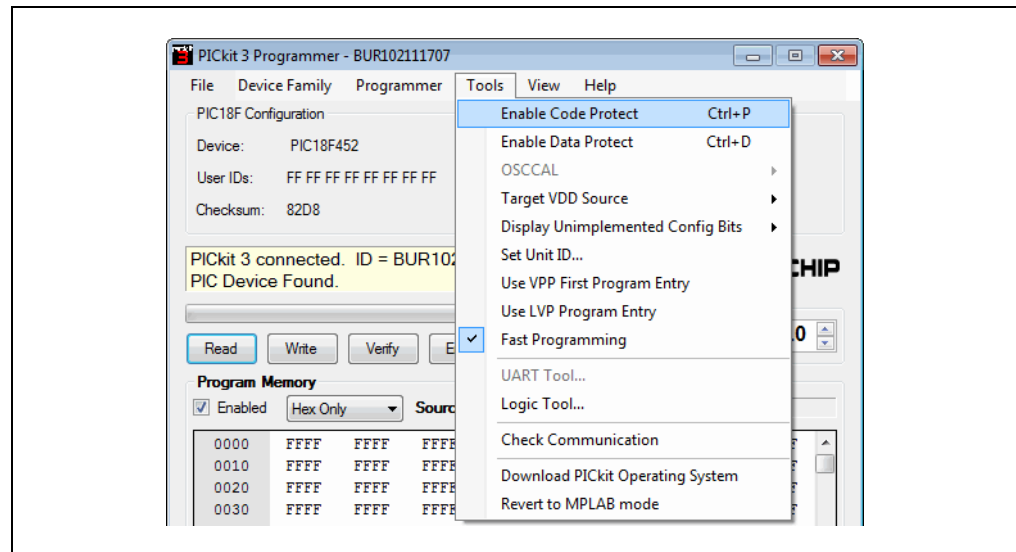
## 2.10 CODE PROTECTING THE DEVICE

The Code and Data Protect functions enable the read protection features of the device. To protect the program memory code, complete the following steps:

1. Import the hex file.
2. Select *Tools>Enable Code Protect*, as shown in Figure 2-8.
3. Click **Write**.

For devices that have EEPROM data memory, it can be protected by selecting *Tools>Enable Data Protect*.

**FIGURE 2-8: ENABLE CODE PROTECT**



**Note:** If the device is read after it has been protected, the protected memory regions will display all zeros. Unchecking “Enable Code Protect” will not allow you to read the region. You must erase and reprogram all device memory before you can read that memory region again.

# PICKit™ 3 Programmer Application User's Guide

---

## 2.11 ERASING AND BLANK CHECKING THE DEVICE

The Erase function erases the program memory, data EEPROM memory, ID, and Configuration bits, regardless of the state of the Program Memory and EEPROM Data “Enabled” checkboxes. However, this function is not normally needed because the Write function performs an erase operation prior to programming the device.

To erase the device, click **Erase**.

**Note:** The PICKit 3 Erase function always uses the Bulk Erase method that requires a minimum VDD, even on devices that support row erasing for the Write function. You will be warned if VDD is below the minimum for the connected device.

The Blank Check function will read the entire device to determine if Program Memory, EEPROM Data memory, User IDs, and Configuration bits are erased. All memory regions will be examined, regardless of the state of the Program Memory and EEPROM Data “Enabled” checkboxes.

To Blank Check the device, click **Blank Check**.

## 2.12 AUTOMATING WRITE/READ PROCEDURES

The PICKit 3 Programmer Application has two buttons for automating multiple functions.

**FIGURE 2-9: AUTOMATING BUTTONS**



### 2.12.1 Auto Import Hex + Write Device Button

This feature allows the PICKit 3 Programmer Application to automatically import a hex file and write it to a connected device when the hex file is updated; for example, on a new firmware build.

To use this feature, click **Auto Import Hex + Write Device**. This will bring up an Import Hex file dialog that defaults to the first hex file in the file history (under the File menu). After selecting a file, it is written to the device. The PICKit 3 Programmer Application then monitors the selected hex file for updates. When the file has been updated (has a newer time stamp), the application automatically re-imports the hex file and writes to the target device.

While this feature is enabled, other programming operations are disabled. The **Auto Import Hex + Write Device** button remains pressed while this feature is active. To stop using this feature, click **Auto Import Hex + Write Device** again.

If an error is encountered during hex file importing or device programming, the application will automatically exit this feature mode.

### 2.12.2 Read Device + Export Hex File Button

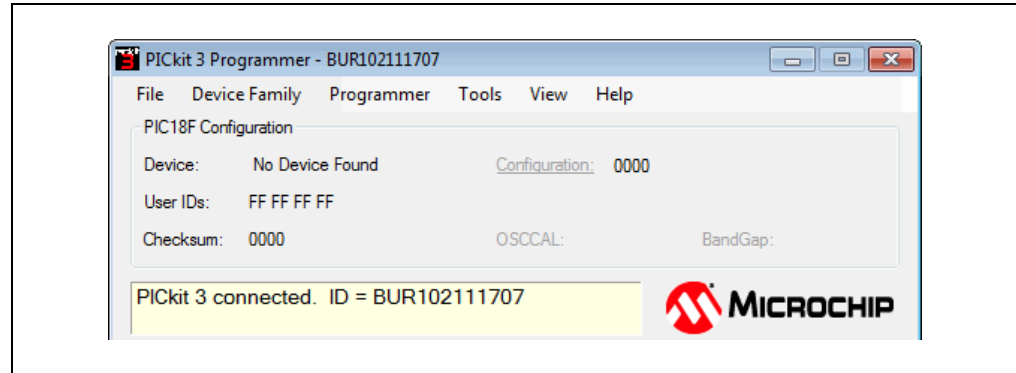
When clicked, this button will read the target device and open an **Export Hex File** dialog.

## 2.13 PICKIT 3 UNIT ID

The PICKit 3 might be assigned (optionally) a Unit ID string to identify it uniquely.

Once assigned, the PICKit 3 Unit ID displays in the PICKit 3 Programmer software title bar, and in the Status Window, when first connecting to the PICKit 3. An example is shown in Figure 2-19 where the Unit ID is “BUR102111707”.

**FIGURE 2-10: UNIT ID**



# PICkit™ 3 Programmer Application User's Guide

---

NOTES:





### 3.1 INTRODUCTION

- VPP – Programming Voltage; when applied, the device goes into Programming mode.
- ICSPCLK or PGC – Programming Clock; a unidirectional synchronous serial clock line from the programmer to the target.
- ICSPDAT or PGD – Programming Data; a bidirectional synchronous serial data line.
- VDD – Power Supply positive voltage.
- VSS – Power Supply ground reference.

**Note:** For details on how a specific device is programmed, refer to the device programming specification available from the Microchip web site at [www.microchip.com](http://www.microchip.com).

[illegible]

## 3.2 ISOLATE VPP/MCLR/PORT PIN

When VPP voltage is applied, the application circuit needs to take into consideration that the typical VPP voltage is +12V. This may be an issue in the following situations.

### If the VPP pin is used as a MCLR pin:

The application circuit is typically connected to a pull up resistor/capacitor circuit, as recommended in the device data sheet. Care must be taken so that the VPP voltage slew rate is not slowed down and exceeds the rise time in the programming specification (typically 1  $\mu$ s).

If a supervisory circuit or a push button is interfaced to the MCLR pin, it is recommended that they be isolated from the VPP voltage by using a Schottky-type diode or limiting resistor, as shown in Figure 3-1. For more information about using supervisory circuits with ICSP, see Application Note AN820 “*System Supervisors in ICSP™ Architectures*” (DS00820).

### If the VPP pin is used as an I/O port pin:

The application circuit that connects to the I/O pin may not be able to handle the +12V voltage. It is recommended to use a Schottky-type diode or limiting resistor, as shown in Figure 3-1 to isolate the circuitry.

## 3.3 ISOLATE ICSPCLK OR PGC AND ICSPDAT OR PGD PINS

The ICSPCLK or PGC and ICSPDAT or PGD pins need to be isolated from the application circuit to prevent the programming signals from being affected by the application circuitry. ICSPCLK or PGC is a unidirectional synchronous serial programming clock line from the programmer to the target. ICSPDAT or PGD is a bidirectional synchronous serial programming data line.

If the design permits, dedicate these pins for ICSP. However, if the application circuit requires that these pins be used in the application circuit, design the circuitry in a manner that does not alter the signal level and slew rates. Isolation circuitry will vary according to the application. Figure 3-1 shows one possibility by using series resistors to isolate the ICSP signals from the application circuit.

## 3.4 VDD

During ICSP programming, the device needs to be powered in accordance with the device specification. Typically, the device supply voltage is connected to the application circuit supply voltage. The application circuit can be powered by the PICkit 3 or externally. There are a few precautions that need to be observed in the situations covered in the following three sections.

### 3.4.1 The application circuit is powered by the PICkit 3

The PICkit 3 supply voltage may set between the maximum and minimum voltages allowed by the device programming specification, unless the minimum is below +2.5V. Be sure to set the voltage box to the appropriate voltage before programming the device or turning on VDD.

#### CAUTION

When using bus powered hubs, the maximum current supplied to the PICkit 3 is 100 mA. If the target plus programmer exceeds this current limit, the USB port may turn off. The target may be powered externally if more power is required.

**Note:** Current draw should be limited to 30 mA when using the programmer to power the application circuit. Ensure that the application circuit does not slow the VDD rise time to longer than 500  $\mu$ s.

### 3.4.2 The application circuit is powered externally

The PICkit 3 may be used with application circuits powered externally between +5.5V and +1.8V.

### 3.4.3 Bulk Erase is used

Some devices use a Bulk Erase function to erase program memory, data EEPROM memory, ID locations, and Configuration bits. Typically, the Bulk Erase function requires a supply voltage (VDD) of 4.5 to 5.5 Volts (refer to the device programming specification for device specific requirements).

This voltage range can be a problem if the application circuit is designed to operate at a different supply voltage range. In order to Bulk Erase the device, the application circuit needs to take into consideration the Bulk Erase voltage requirement while protecting any voltage sensitive circuitry.

If the application circuit VDD is below the minimum required for the Bulk Erase, a dialog will warn the user before attempting to erase the device.

## 3.5 Vss

The power supply ground reference, Vss, must be at the same potential as the application circuit.

## 3.6 CABLE LENGTHS

Minimize the distance the ICSP signals must travel by placing the ICSP connector as close to the application circuit device as possible. Minimize any cable length between the PICKit 3 and application circuit device. The goal is to keep the ICSP signals within the level and slew rate specifications for successful programming.

## 3.7 SERIAL EEPROM AND KEELOQ HCS DEVICES

The programming signals and connections for these devices are different than those for microcontrollers as described in **Section 3.1 “Introduction”** and Figure 3-1. See the PICKit 3 Programmer Readme file, [Help>Readme](#), for programming signal connectivity for serial EEPROM and KEELOQ HCS devices.

Additionally, these devices are not intended to be programmed in-circuit. Attempting to program serial EEPROM devices while in-circuit may fail due to conflicts with other devices on the serial bus.

## 3.8 LOGIC TOOL

The PICKit 3 Logic Tool allows the PICKit 3 ICSP connector pins to be used for stimulating and probing digital signals in a target circuit, and collectively as a simple 3 channel logic analyzer. The Logic Tool is opened by selecting [Tools>Logic Tool](#) in the main PICKit 3 Programmer Application window. See **Chapter 7. “Logic Tool”** for more information.

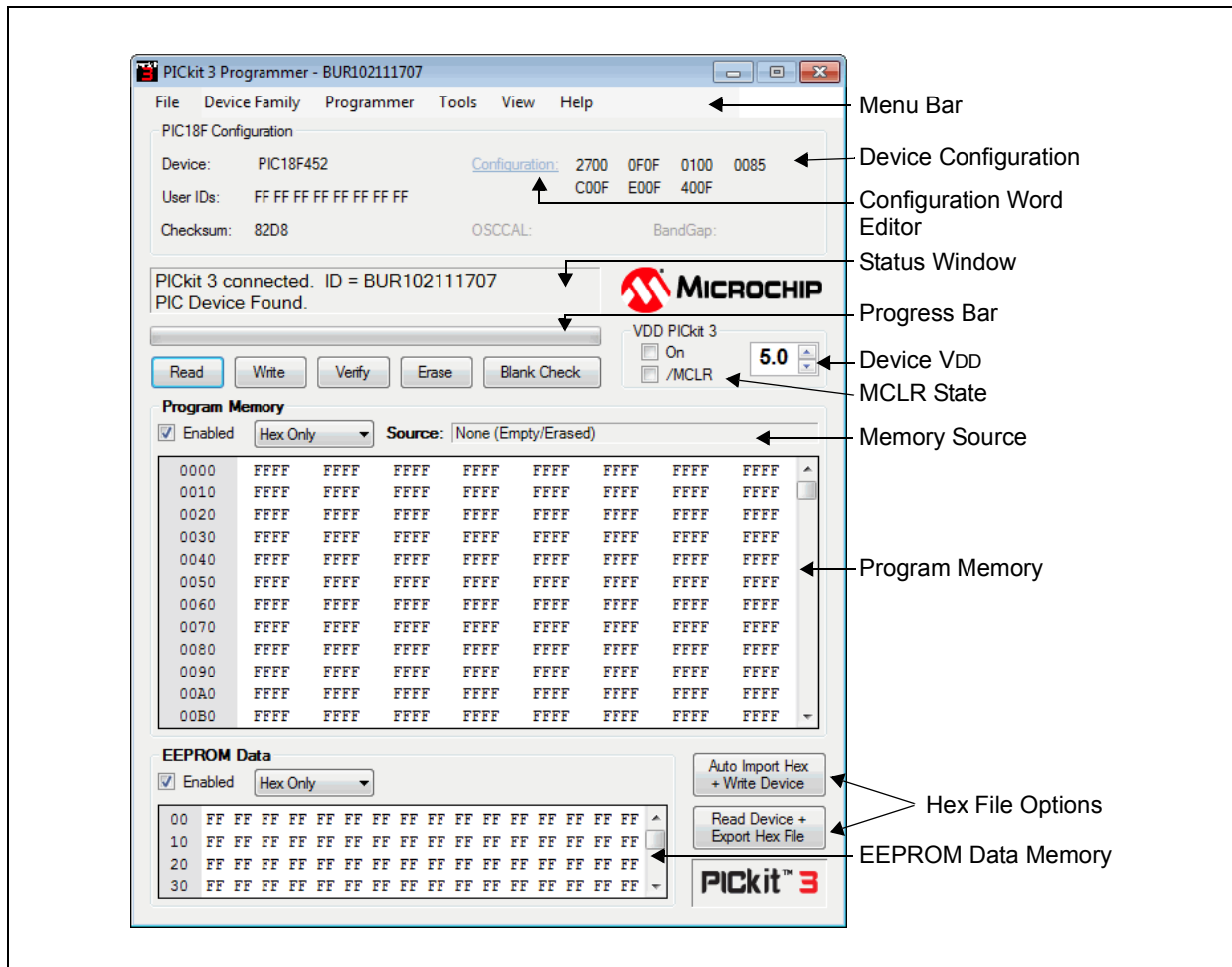
## Chapter 4. PICKit 3 Programmer Application

### 4.1 PICKit 3 PROGRAMMER APPLICATION DIALOG

The PICKit 3 Programmer Application allows you to program all supported devices listed in the PICKit 3 Readme file without using the MPLAB IDE. The programming interface appears, as shown in Figure 4-1. The menu items and controls are described in the following sections.

For more information on how to install and use the PICKit 3 Programmer Application, see Chapter 2. “Getting Started”.

**FIGURE 4-1: PICKIT™ 3 PROGRAMMER APPLICATION DIALOG**



## 4.2 PICKIT 3 PROGRAMMER APPLICATION MENU BAR

The menu bar selects various functions of the PICkit 3 Programmer Application.

### 4.2.1 File

- Import Hex – import a hex file for programming. The hex file format INHX32 is supported.
- Export Hex – export a hex file read from a device. The hex file is created in the INHX32 format.
- Exit – exit the program.

### 4.2.2 Device Family

Select a device family to search for a connected device in that family. Selecting the device family of the current part will clear all device data.

Some families that cannot be auto-detected (such as Baseline) will open a drop-down box so that supported devices can be selected.

### 4.2.3 Programmer

- Read Device – reads program memory, data EEPROM memory, ID locations and Configuration bits.
- Write Device – writes program memory, data EEPROM memory, ID locations and Configuration bits.
- Verify – verifies program memory, data EEPROM memory, ID locations and Configuration bits read from the target MCU against the code stored in the programming application.
- Erase – performs a Bulk Erase of the target MCU. OSCCAL and band gap values are preserved on parts with these features.
- Blank Check – performs a Blank Check of program memory, data EEPROM memory, ID locations and Configuration bits.
- Verify on Write – when checked, the device will be immediately verified after programming on a Write (recommended). When unchecked, the device will be programmed but not verified on a Write.
- Clear Memory Buffers on Erase – clears and resets the memory buffers on the target MCU when an erase command is performed.
- Hold Device in Reset – when checked, the MCLR (VPP) pin is held low (asserted). When unchecked, the pin is released (tri-stated), allowing an external pull-up to bring the device out of Reset.
- Alert Sounds... – opens the Alert Sounds dialog where you can enable and select sound files to indicate success, warning or error.
- Write on PICkit Button – when checked, a Write operation will be initiated by pressing the PICkit 3 push button.
- Manual Device Select – enables you to type a device to select.

## 4.2.4 Tools

- Enable Code Protect – enables code protection features of the microcontroller on future Write operations.

**Note:** To disable code protect, *all* device memory must be erased and rewritten.

- Enable Data Protect – enables data protection features of the microcontroller on future Write operations
- OSCCAL – allows the OSCCAL value to be changed for devices where it is stored in the last location of Program Memory.
- Target VDD Source
  - Auto-Detect – The PICkit 3 automatically detects whether the target device has its own power supply or needs to be powered by the programmer on each operation.
  - Force PICkit 3 – The PICkit 3 always attempts to supply VDD to the target device.
  - Force Target – The PICkit 3 always assumes that the target has its own power supply.
- Display Unimplemented Config Bits
  - As '0' bit value
  - As '1' bit value
  - As read or imported
- Set Unit ID – opens a dialog to optionally assign a Unit ID so that multiple PICkit 3 devices can be identified.
- Use VPP First Program Entry – when checked, it allows the PICkit 3 to connect to and program devices with configurations and code that interferes with the ICSP signal pins, preventing PICkit 3 from detecting them. Using this feature requires that the PICkit 3 supplies VDD to the target.
- Use LVP Program Entry – when checked, it allows the PICkit 3 to connect to and program devices with low voltage.
- Fast Programming – when checked, the PICkit 3 will attempt to program the device as fast as possible. When unchecked, the PICkit 3 will slow down ICSP communication. This may be helpful for targets with loaded ICSP lines.
- Check Communication – verifies USB communication with the PICkit 3 and ICSP communication with a target device by attempting to identify the connected device by its device ID.
- Logic Tool... - launches the Logic Tool.
- Check Communication - verifies communication with the PICkit 3.
- Download PICkit Operating System – performs a download of the PICkit 3 operating system (firmware).
- Revert to MPLAB mode - the PICkit 3 cannot operate in the Programmer Application and MPLAB IDE modes simultaneously. Selecting this option returns the PICkit 3 to bootloader mode so that the MPLAB IDE can update the PICkit 3 with compatible firmware.

## 4.2.5 View

- Single Window - displays active window
- Multiple Window - enables selectable viewing of separate windows for:
  - Program Memory
  - EEPROM data
  - Associate/Memory Displays in Front

# PICkit™ 3 Programmer Application User's Guide

## 4.2.6 Help

- PICkit 3 Programmer Application User's Guide – launches the user's guide PDF (Adobe® Reader must be installed).
- PICkit 3 ReadMe – opens the PICkit 3 Readme.txt file.
- PICkit 3 Programmer on the web – opens the link to the PICkit 3 in the default web browser.
- About – opens a dialog with the PICkit 3 Programmer Application version, device file version and firmware version.

## 4.3 DEVICE CONFIGURATION

The Device Configuration window displays the device, User ID, Configuration, and Checksum. It also displays OSCCAL and Band Gap for parts with those features.

For baseline (12-bit core) devices, serial EEPROM devices, and KEELOQ HCS devices, you must select the device from the Device drop-down menu.

All other part family devices will be detected by their device ID and the part name will be displayed on the Device line.

To edit the Configuration settings, click on **Configuration:** to open the Configuration Word Editor dialog (Figure 4-2), where you can edit the bit settings.

FIGURE 4-2: CONFIGURATION WORD EDITOR

Configuration Word Editor

Device Configuration Words may be edited here at the bit level. Refer to device datasheet for specific configuration bit functions.

- = Unimplemented bit    1 = Configuration bit. Click to toggle value.

Name	Address	Value	Bit Edit															
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONFIG1	300000	2700	-	-	1	-	-	1	1	1	-	-	-	-	-	-	-	-
CONFIG2	300002	0F0F	-	-	-	-	1	1	1	1	-	-	-	-	1	1	1	1
CONFIG3	300004	0100	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-
CONFIG4	300006	0085	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	1
CONFIG5	300008	C00F	1	1	-	-	-	-	-	-	-	-	-	-	1	1	1	1
CONFIG6	30000A	E00F	1	1	1	-	-	-	-	-	-	-	-	-	1	1	1	1
CONFIG7	30000C	400F	-	1	-	-	-	-	-	-	-	-	-	-	1	1	1	1

Unimplemented bits are displayed in the Value column as selected in menu Tools > Display Unimplemented Config Bits

Save Cancel



## 4.4 STATUS WINDOW

The status window displays text status of the operations in progress. If an operation is successful, the status window will display a green background. If an operation fails, the status window will display red. If an operation alerts a caution, the status window will display yellow.

## 4.5 PROGRESS BAR

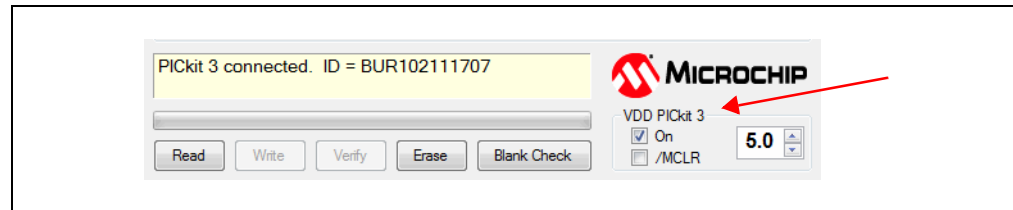
The progress bar displays the progress of an operation.

## 4.6 DEVICE VDD

The PICkit 3 VDD may be turned on and off by clicking the checkbox “On”. The voltage may be set in the box on the right either by typing it directly or using the up/down arrows to adjust it a tenth of a volt at a time. The maximum and minimum allowed voltages will vary depending on the target device.

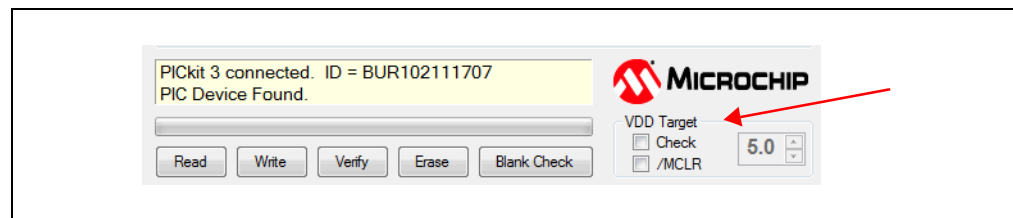
If the “On” checkbox is unchecked, PICkit 3 will automatically turn on the VDD at the set voltage during any requested programming operation.

**FIGURE 4-3: PICKIT™ 3 SUPPLIED VDD**



If the target device has its own power supply, then the PICkit 3 will display the detected VDD voltage in the box on the right, which will be grayed out to prevent being changed. The checkbox text changes to “Check”, and clicking on the checkbox will update the detected VDD voltage value. If *Target VDD > Auto-Detect* is selected, clicking on the checkbox will revert the VDD mode back to PICkit 3 supplied VDD if a target power supply is no longer detected.

**FIGURE 4-4: TARGET SUPPLIED VDD**



## 4.7 DEVICE MCLR STATE

The “/MCLR” checkbox shown in Figure 4-3 and Figure 4-4 has the same functionality as the menu selection *Programmer > Hold Device in Reset*. When the box is checked the target device will be held in Reset. When unchecked, the target circuit is allowed to pull MCLR up to VDD to release the device from Reset. This function can be used to prevent a device from executing code before and after programming.

**Note:** If the target device allows the MCLR pin to be configured as an input port, and it is configured as such, PICkit 3 will not be able to hold the device in Reset.

# PICkit™ 3 Programmer Application User's Guide

---

## 4.8 MEMORY SOURCE

The Source bar displays the source of the currently loaded device data. If read from a hex file, it will display the hex file name. If read from a device, it will display the part name. *None* (Empty/Erased) indicates the buffers are empty, and it will display *Edited* once Program Memory or Data EEPROM Memory has been edited in the window.

## 4.9 PROGRAM MEMORY

Program code can be loaded into the PICkit 3 Programmer Application by selecting File>Import HEX to import a hex file or by clicking **Read** to read the device memory. The origin of the code is displayed in the Source block. The Program Memory window displays the program code in hexadecimal. The code may be edited in the window.

The checkbox next to the Program Memory window is only available on devices with EEPROM data memory. If the box is checked, then Program Memory, User IDs, and Configuration Words are written to, read from, and verified on the device. If the box is unchecked, then Program Memory, User IDs, and Configuration Words will not be erased or altered during a Write Device operation, and will not be read or verified. The checkbox does not affect Erase Device or Blank Check operations. Both memory window checkboxes cannot be cleared at the same time.

For supported serial EEPROM devices, the device contents are displayed in the Program Memory window, instead of the Data EEPROM Memory window, for easier viewing in the larger display area.

## 4.10 HEX FILE OPTIONS

Two options are available to set up automating multiple functions:

- Auto Import Hex + Write Device — automatically imports a hex file and writes it to the device
- Read Device + Export Hex File — automatically reads the device and opens a dialog to export the hex file.

## 4.11 EEPROM DATA MEMORY

Similar to Program Memory, data EEPROM code can be loaded into the PICkit 3 Programmer Application by selecting File>Import HEX to import a hex file or by clicking **Read** to read the device memory. The origin of the code is displayed in the Source block. The Data EEPROM Memory window displays the program code in hexadecimal. The code may be edited in the window.

The check box next to the EEPROM Data window controls whether the EEPROM Data memory is written, read, and verified. If the box is checked, then the device EEPROM will be overwritten with the window data. If the box is not checked, then the device EEPROM will not be erased or altered during a Write Device operation. The checkbox does not affect Erase Device or Blank Check operations. Both memory window checkboxes cannot be cleared at the same time.

---

## Chapter 5. Troubleshooting

---

### 5.1 INTRODUCTION

This chapter provides questions and answers to common problems associated with using the PICKit 3 Programmer Application.

### 5.2 FREQUENTLY ASKED QUESTIONS

- Device is Not Recognized
- Current Limit Exceeded
- Verify and Read Return All Zeros
- VDD/VPP Errors
- Programming Errors
- Windows Error: Unrecognized USB Device
- PICKit 3 Not Found
- PICKit 3 Programmer Application Locks Up
- Programming Fails on Configuration
- Unable to Program PIC10F Devices
- The PICKit 3 PGM/LVP Pin
- PICKit 3 HEX File Format
- Window Display Problems

#### Device is Not Recognized

##### Question

Why am I receiving a “*No Device Found*” message?

##### Answer

Verify that the device is supported and that the target MCU is connected to the PICKit 3 in accordance with **Chapter 3. “Using In-Circuit Serial Programming (ICSP)”**. Verify that PIC18FXXJXX, PIC24X, and dsPIC33F devices have an appropriate capacitance on the VDDCORE/VCAP pin in accordance with the device data sheet.

Verify that the device is a member of the currently active family displayed at the top of the Status window. Select the correct family from the Device Family menu if needed.

See also the Programming Fails on Configuration topic.

#### Current Limit Exceeded

##### Question

Why am I receiving the error message “*USB Hub Current Limit Exceeded*” from the Microsoft® Windows® program?

##### Answer

Verify that the application circuit is not drawing more than 30 mA from the PICKit 3.

## Verify and Read Return All Zeros

### Question

When **Verify** or **Read** are clicked, the Program Memory window comes up with all zeros. What is wrong?

### Answer

The device may be code-protected. Ensure code protection has not been selected in the Configuration Word.

## VDD/VPP Errors

### Question

Why do I keep getting a “VDD Error” or “VPP Error”?

### Answer

This error indicates that the PICkit 3 is not able to drive VDD or VPP to the intended voltage. Check the circuit board for shorts, for large current draw and verify that the target device is connected to the PICkit 3 in accordance with **Chapter 3. “Using In-Circuit Serial Programming (ICSP)”**. Make sure that VDD capacitance is not reducing the VDD rise time longer than 500  $\mu$ s.

## Programming Errors

### Question

Why am I able to program some parts but not others?

### Answer

If some parts are configured for Low-Voltage Programming, a floating PGM pin can interfere with programming. Use a resistor to pull this pin low when programming.

Some Mid-Range parts, such as the PIC16F72/73/74/76/77 family and PIC16F737/747/767/777 family require a minimum programming VDD of +4.75V. Depending on the USB voltage, the PICkit 3 may not be able to supply +4.75V on VDD. Program these parts using an external +5.0V power supply.

Some PIC18F parts require significant bypass capacitance on VDD. Try increasing the total bypass capacitance up to 10  $\mu$ F.

PIC18FXXJXX, PIC24X, and dsPIC30F/33F devices require a 4.7  $\mu$ F capacitor on the VDDCORE/VCAP pin in order to function properly. If not using a separate regulator to supply VDDCORE, ensure that the ENVREG pin is tied to VDD.

## Windows Error: Unrecognized USB Device

### Question

Why do I get an “unrecognized device” error when plugging my PICkit 3 into USB?

### Answer

This error may occur if PICkit 3 is plugged into USB while connected to a target circuit board. When plugging PICkit 3 in a PC, restarting or booting up a PC, ensure the PICkit 3 is not connected to a target device.

This error may also occur when PICkit 3 is used with some USB hubs. If the PICkit 3 is plugged into a USB hub port, try plugging the PICkit 3 directly into a PC USB port.

## PICkit 3 Not Found

### Question

I have my PICkit 3 plugged into USB, but the PICkit 3 Programmer Application keeps saying "PICkit 3 Not Found?"

### Answer

Please see the answer for Windows Error: Unrecognized USB Device.

## PICkit 3 Programmer Application Locks Up

### Question

Why is the PICkit 3 Programmer window locking up?

### Answer

Often times, the software hasn't truly locked up. During a programming operation the PICkit 3 application user interface is inactive. Until the programming operation is complete, the Windows OS may stop updating the PICkit 3 if another application is brought into focus or the PICkit 3 has just been brought back into focus. Once the programming operation is complete, the application window will update. For large memory parts, programming may take several minutes. For 8-bit devices, try waiting at least 3 minutes before concluding the software has truly locked up. For 16-bit devices, try waiting 5 minutes for the operation to complete.

There are a few USB controller chipsets that appear to cause lockup problems with the PICkit 3, which seem to be more common in laptops. This can usually be worked around by connecting a USB hub between the PC and the PICkit 3 unit, or using a USB Cardbus adapter. Using a USB hub with an external power supply is recommended.

## Programming Fails on Configuration

### Question

Why does programming a device always fail on the Configuration Word(s), after which PICkit 3 won't recognize the device?

### Answer

This may be caused by a Configuration setting or program code that affects the ICSP PGD and PGC pins. This interference can prevent the target PIC MCU from entering programming mode.

Check the menu option *Tools>Use VPP First Program Entry* when attempting a programming operation on these devices. This program mode entry will usually get around the problem, but it requires that the target device be powered from the PICkit 3 unit VDD pin.

## Unable to Program PIC10F Devices

### Question

Why won't my PIC10F parts program?

### Answer

The PIC10F parts are 6-pin devices, even though they can be ordered in an 8-pin DIP. The AC163020 PIC10F2XX Programming Adapter should be used.

## The PICKit 3 PGM/LVP Pin

### Question

What is the PGM/LVP pin for? How do I connect it?

### Answer

The PICKit 3 PGM/LVP pin is not used for programming PIC microcontrollers, and should be left unconnected. The PGM/LVP pin is only used when programming some Serial EEPROMS. See the PICKit 3 Readme file ([Help>Readme](#)) for EEPROM connection information.

## PICKit 3 HEX File Format

### Question

What HEX file format does the PICKit 3 Programmer application use?

### Answer

The PICKit 3 Programmer application uses the Intel Hex 32 Format, often referred to as INHX32. However, PICKit 3 does not support record types 03 and 05. Least Significant Bytes are at lower hex file addresses (little Endian format).

## Window Display Problems

### Question

Why is the PICKit 3 Programmer application displaying Program Memory and/or EEPROM Data locations as all periods "..."? Why can't I close the [Help>About](#) dialog?

### Answer

These issues are frequently caused by non-standard monitor DPI settings. This can be corrected by setting the DPI to the Normal setting, 96 DPI. To do this

1. Right click on the Windows desktop background and select Properties from the pop-up menu. This will open the Display Properties dialog.
2. In the Display Properties dialog, select the **Settings** tab.
3. On this tab, click **Advanced** to open the Monitor dialog.
4. In the Monitor dialog, select the **General** tab.
5. On this tab, change the "DPI setting:" to "Normal size (96 DPI)
6. Click **Apply**, then the **OK** button to close the dialog.
7. Click **OK** to close the Display Properties dialog.

It may be necessary to restart the PC for the changes to take effect.

## Chapter 6. Updating and Reverting the PICKit 3 OS

### 6.1 INTRODUCTION

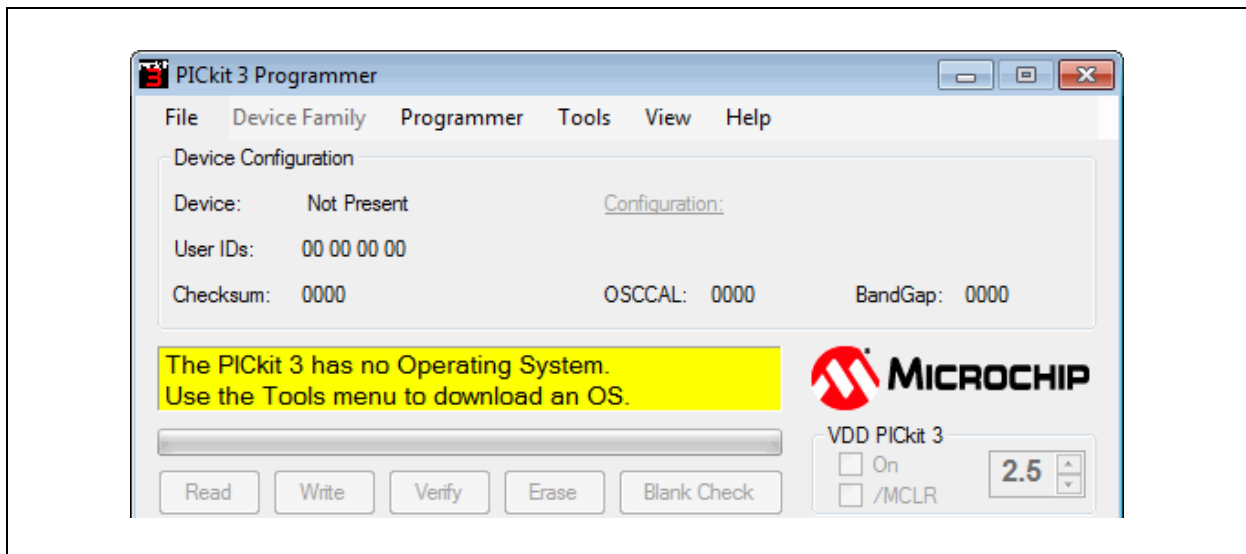
Updating and reverting the operating system (firmware) for the PICKit 3 programmer are discussed in this chapter. The differing operating systems are referred to as scripting mode or MPLAB mode.

**Note:** Do not have the MPLAB IDE open when updating the scripting mode OS or reverting to the MPLAB mode OS.

### 6.2 UPDATING THE PICKit 3 OS TO SCRIPTING MODE

Before launching the PICKit 3 Programmer Application, be certain that the MPLAB IDE is not open. When the PICKit 3 Programmer Application is launched, it will check the firmware version of the PICKit 3 to see if it is the latest version. A message is displayed with instructions to download an operating system (Figure 6-1).

**FIGURE 6-1: UPDATE PICKit™ 3 OS WARNING**



# PICKit™ 3 Programmer Application User's Guide

---

To update the PICKit 3 firmware/OS, refer to the PICKit 3 Programmer Application readme and complete the following steps:

1. When the PICKit 3 Programmer Application was initially downloaded, the latest firmware/OS was also downloaded. If this is the case, go to step 2.  
If you need to download the latest PICKit 3 firmware/OS from the Microchip web site, go to [www.microchip.com](http://www.microchip.com). Save the zip file in the PICKit 3 installation directory. By default, the location is: `C:\Program Files\Microchip\PICKit 3`
2. Select Tools>Download PICKit Operating System.
3. Select the hex file from the directory where the latest firmware/OS file was saved and click on the **Open** button.

The progress of the OS update will display in the status bar.

When the update completes, the status bar will display indicating a successful download. The firmware/OS update is then complete.

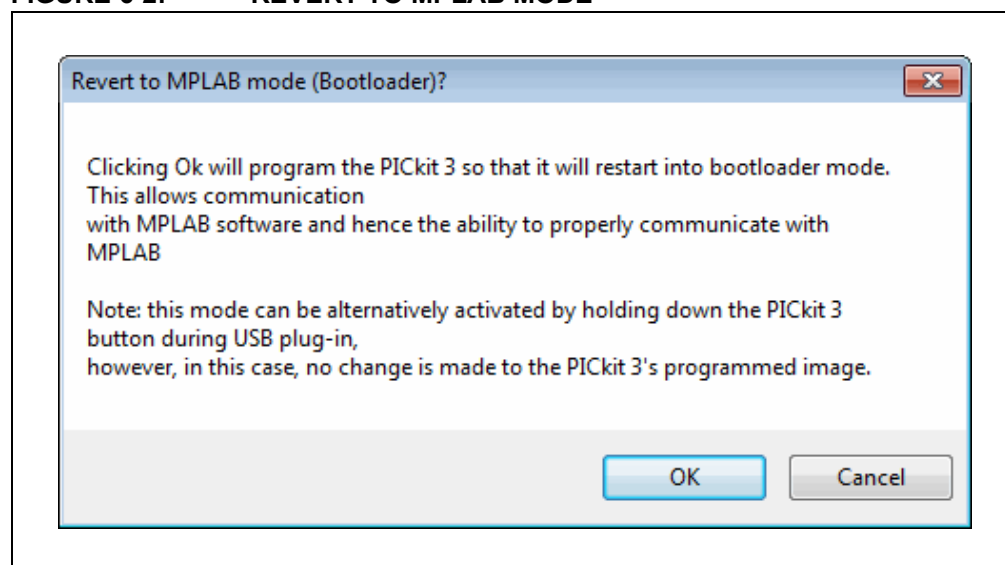
## 6.3 REVERTING THE PICKIT 3 OS TO MPLAB MODE

Before reverting to MPLAB mode, make sure MPLAB IDE is not open. The PICKit 3 OS used for the Programmer Application is not compatible with the MPLAB IDE.

1. To switch from using the PICKit 3 Programmer Application to the MPLAB IDE, select Tools>Revert to MPLAB Mode.

The following message displays.

**FIGURE 6-2: REVERT TO MPLAB MODE**



2. Click **OK** to proceed.  
This reverts the PICKit 3 to bootloader mode so MPLAB IDE can update the PICKit 3 with MPLAB IDE compatible firmware.
3. When complete, a message stating the PICKit 3 has been converted to MPLAB mode displays.
4. Click **OK** to exit the PICKit 3 Programmer Application.



---

## Chapter 7. Logic Tool

---

### 7.1 INTRODUCTION

The PICKit 3 Logic Tool allows the PICKit 3 ICSP connector pins to be used for stimulating and probing digital signals in a target circuit, and collectively as a simple 3-channel logic analyzer. The Logic Tool is opened by selecting *Tools > Logic Tool ...* in the main PICKit 3 Programmer Application window.

The Logic Tool has two operating modes. The Logic I/O mode is useful for triggering inputs to a PIC microcontroller or other digital circuitry, and can monitor digital signals to display their state. In essence, it provides an alternative for wiring buttons and LEDs to pins or signals while debugging or developing I/O functions.

The Analyzer mode can display waveforms of up to three digital signals, and trigger on specific events such as a rising edge on one signal when another signal is at a logic high level. This may be very useful for debugging serial communication buses such as UART, SPI and I<sup>2</sup>C™. It is also very applicable to monitoring the behavior of general microcontroller I/O.

Information in this guide covers:

- Logic I/O Mode
- Configuring the Logic Tool Logic I/O
- Logic Analyzer Mode
- The Logic Analyzer Window

# PICkit™ 3 Programmer Application User's Guide

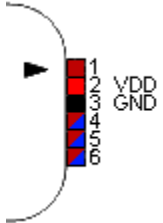
## 7.2 LOGIC I/O MODE

The PICkit 3 Logic Tool “Logic I/O” mode is the default mode when the Logic Tool is first opened. It allows simple stimulus and monitoring of digital signals.

The Logic Tool mode is set by the two buttons in the upper right of the Logic Tool window.

The 6-pin PICkit 3 ICSP connector has four signal pins that can be used inject a digital signal into a circuit or display the state of a digital signal from a circuit. The remaining two pins are dedicated for VDD and Ground connections.

The six ICSP pins can function as follows in Logic I/O mode:

	Pin	ICSP Function	Logic I/O Function
	1	VPP/MCLR	Digital Output
	2	VDD	VDD - must connect to or match target VDD
	3	GND	GND - must connect to target circuit ground
	4	PGD	Digital Output or Digital Input
	5	PGC	Digital Output or Digital Input
	6	PGM (LVP)	Digital Output or Digital Input

**Note 1:** The PICkit 3 VDD pin must be connected to the target circuit VDD supply, or set to provide a VDD output voltage in the main PICkit 3 application form.

The voltage level at the VDD pin sets the output high voltage for pins 4, 5, and 6 when used as outputs. For example, if using the PICkit 3 to provide digital stimulus to a 3.3 Volt circuit, the VDD pin should be either set to or connected to a 3.3 V supply to limit the output high voltage to 3.3 Volts.

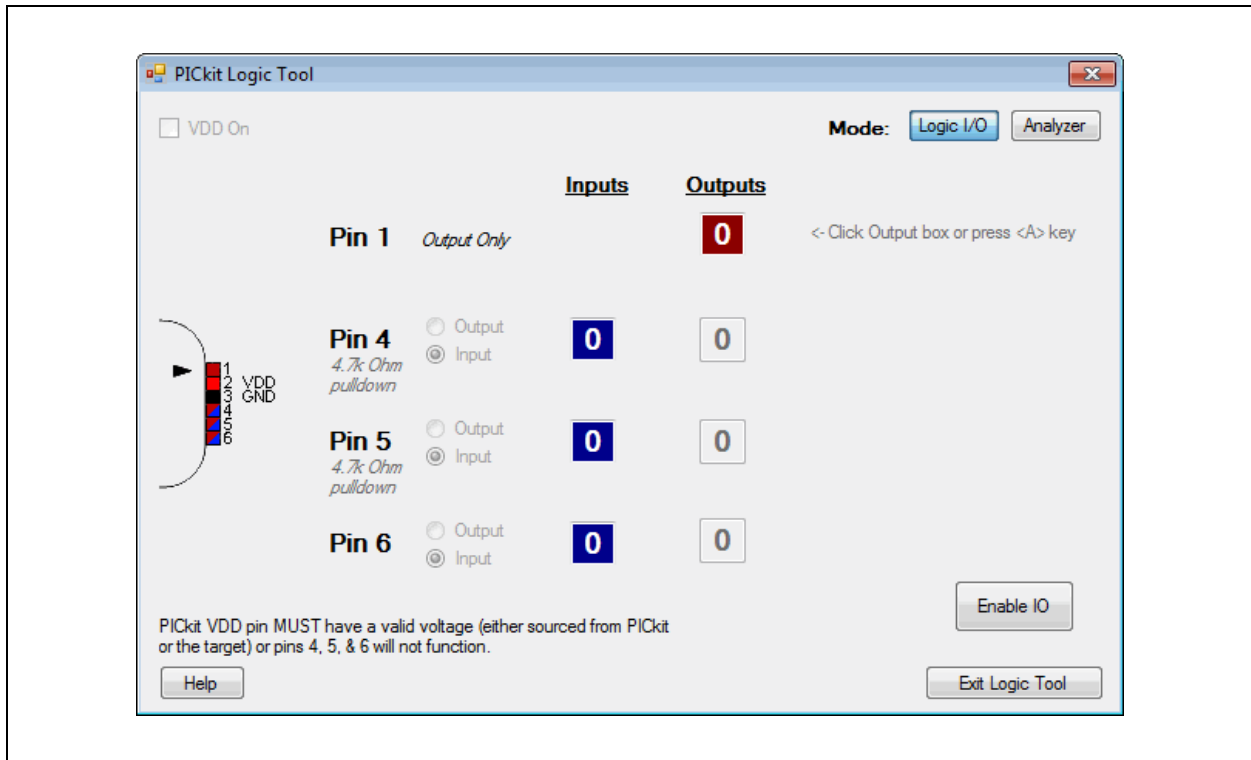
Pin 1's output voltage swing is determined by the voltage on the VDD pin when the Logic I/O is first enabled.

**2:** When used as inputs, pins 4 and 5 may be used to monitor signals down to 2.5 Volt logic, as these are TTL input buffers. Pin 6, as an input, may be used to monitor signals down to 3.6 Volt logic. It may not reliably report high signal states for lower voltage logic signals, as the input buffer is a Schmitt Trigger.

## 7.3 CONFIGURING THE LOGIC TOOL LOGIC I/O

First, to use the Logic I/O mode the Logic I/O mode button on the upper right of the logic window must be depressed, as shown in Figure 7-1.

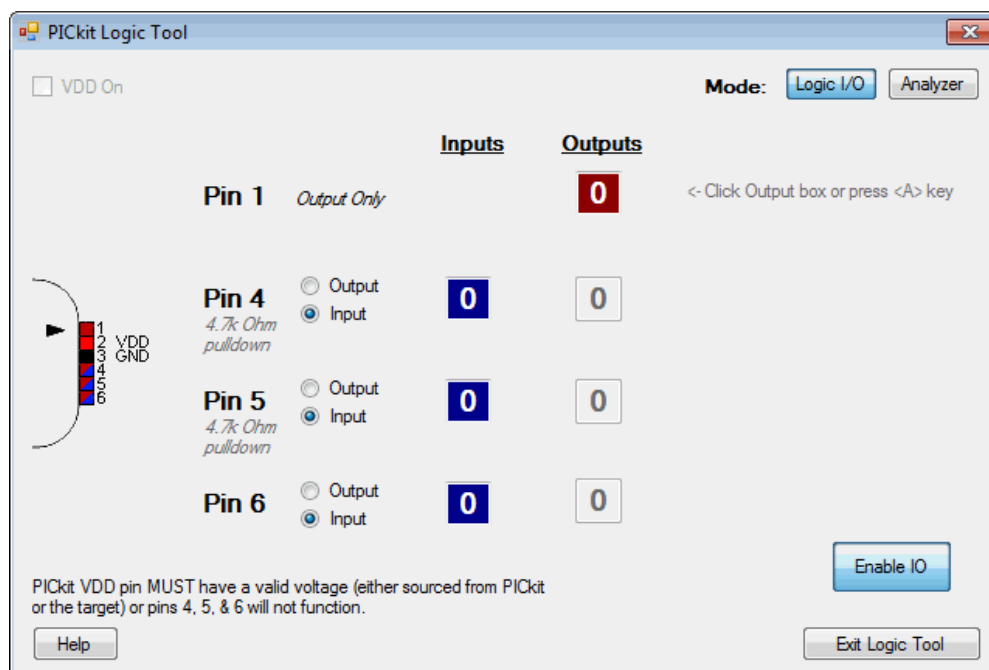
**FIGURE 7-1: INITIAL LOGIC I/O MODE DISPLAY**



# PICkit™ 3 Programmer Application User's Guide

Second, the four PICkit 3 pins used for Logic I/O digital signals (pins 1, 4, 5, and 6) will remain tri-stated (inactive) until the Enable IO button is pressed, as shown in Figure 7-2. When the IO is enabled, it becomes active and can be configured. If no valid voltage is detected on the VDD pin when clicking Enable IO a dialog will pop up to alert the user, and the PICkit 3 pins will remain disabled.

**FIGURE 7-2: LOGIC I/O MODE ENABLED**



Now that the pins of the PICkit 3 unit are enabled, the pin directions and output states can be configured.

## 7.3.1 Setting Pin Direction

Pins 4, 5, and 6 may be configured as Outputs (output a digital signal from PICkit 3) or Inputs (monitor a digital signal state connected to the pin). Pin 1 is only available as an Output.

Click the radio buttons next to the Pin # to set the pin as an Output or Input. When the pin is an Input, the connected signal state is displayed in the blue "Inputs" box, as shown in Figure 7-3 and Figure 7-4.

**FIGURE 7-3: LOGIC I/O INPUT SIGNAL IS LOGIC LOW ('0')**

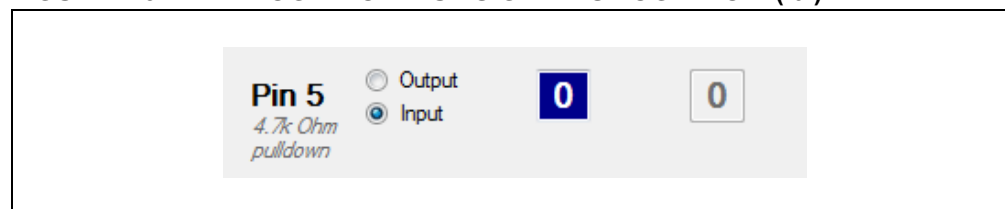
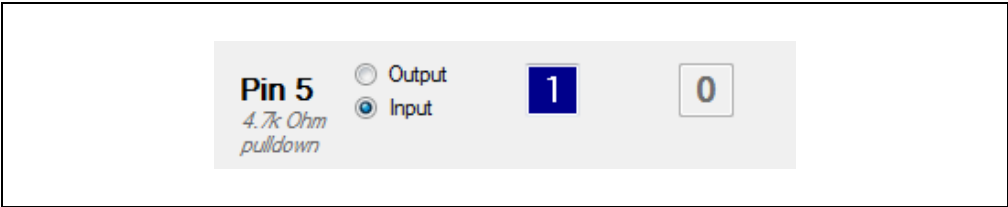


FIGURE 7-4: LOGIC I/O INPUT SIGNAL IS LOGIC HIGH ('1')



**Note:** Pin 4 and Pin 5 have a 4.7k Ohm pull-down resistor internal to the PICkit 3. This resistor is necessary for the PICkit 3 debugger functions, but note that this pull-down resistor will affect any digital signal it is connected to. Generally, this is only an issue when using Pin 4 or Pin 5 as an input. See the **Appendix B. "PICkit 3 Schematics"** for the pin circuit diagrams.

When a pin is selected as an Output, the pin will drive the logic level shown in the read "Output" box. Toggle the output state by clicking on the Output state box. Alternatively, a keyboard shortcut key can be used for each pin to toggle the output. The shortcut keys are:

Pin	Shortcut Key
1	<A>
4	<S>
5	<D>
6	<F>

FIGURE 7-5: LOGIC I/O OUTPUT LOGIC LOW ('0')



FIGURE 7-6: LOGIC I/O OUTPUT LOGIC HIGH ('1')



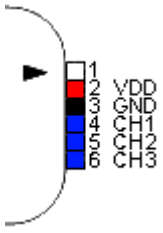
# PICkit™ 3 Programmer Application User's Guide

## 7.4 LOGIC ANALYZER MODE

The Analyzer mode of the PICkit 3 Logic Tool enables using the PICkit 3 as a simple 3-channel logic analyzer to capture, view, and measure the digital waveforms of up to three signals.

### 7.4.1 Connecting the PICkit 3 in Analyzer Mode

The PICkit 3 ISCP connector pins 4, 5, and 6 are used as the inputs for the three logic channels.

	Pin	ICSP Function	Logic Analyzer Function
	1	VPP/MCLR	No Connect
	2	VDD	VDD - must connect to or match target VDD
	3	GND	GND - must connect to target circuit ground
	4	PGD	Analyzer Channel 1
	5	PGC	Analyzer Channel 2
	6	PGM (LVP)	Analyzer Channel 3

For example, to monitor a SPI bus, the analyzer channel pins could be connected to monitor the three main bus signals as follows:

Logic Analyzer Pin	SPI Bus Signal
Analyzer Channel 1	SCK
Analyzer Channel 2	SDO (bus master output)
Analyzer Channel 3	SDI (bus master input)

- Note 1:** The PICkit 3 VDD pin must be connected to the target circuit VDD supply, or set to provide a VDD output voltage in the main PICkit 3 application form.
- Having the VDD pin connected is necessary as the PICkit 3 logic channel pins are clamped to the VDD pin voltage. If no voltage is present on VDD, the analyzer channel pins will be essentially clamped to ground!
- It is possible to have PICkit 3 output a VDD voltage without connecting pin 2 to the circuit VDD, as long as the VDD level is greater than or equal to the target circuit logic high voltage.
- 2:** Pin 4 and Pin 5 have a 4.7k Ohm pull-down resistor internal to the PICkit 3. This resistor is necessary for the PICkit 3 debugger functions, but note that this pull-down resistor will affect any digital signal these pins are connected to. See the PICkit 3 schematic in the PICkit 3 User's Guide appendix for the pin circuit diagrams.
- 3:** Channels 1 and 2 (pins 4 and 5) may be used to monitor signals down to 2.5 Volt logic, as these are TTL input buffers. Channel 3 (pin 6) may be used to monitor signals down to 3.6 Volt logic. It may not reliably report high signal states for lower voltage logic signals, as the input buffer is a Schmitt Trigger.

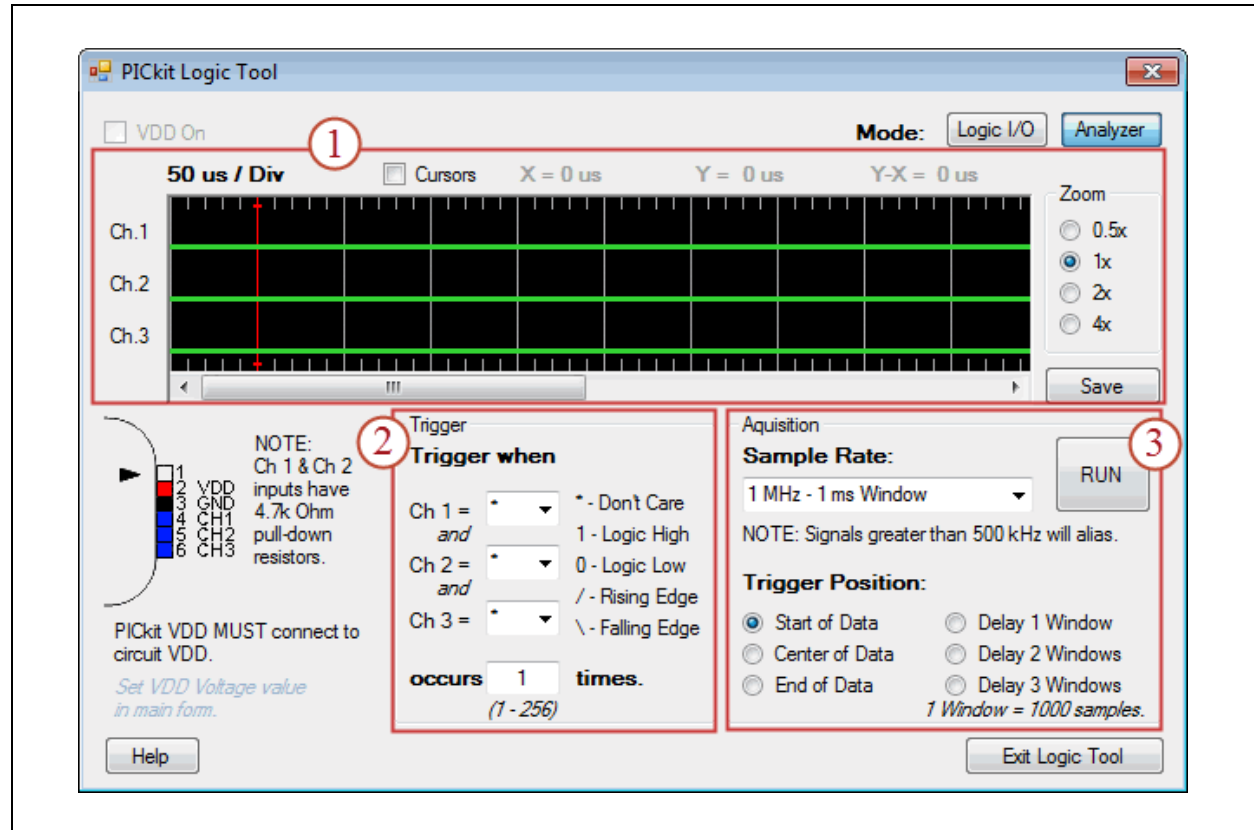
## 7.5 THE LOGIC ANALYZER WINDOW

The Logic Tool analyzer window is divided into three sections, as shown in Figure 7-7. These are

1. Display – for viewing and measuring captured waveforms.
2. Trigger – for setting trigger conditions for a capture
3. Acquisition – for setting the waveform sample rate and the waveform relation to the trigger sample.

Each section will be covered one at a time.

**FIGURE 7-7: ANALYZER WINDOW SECTIONS**

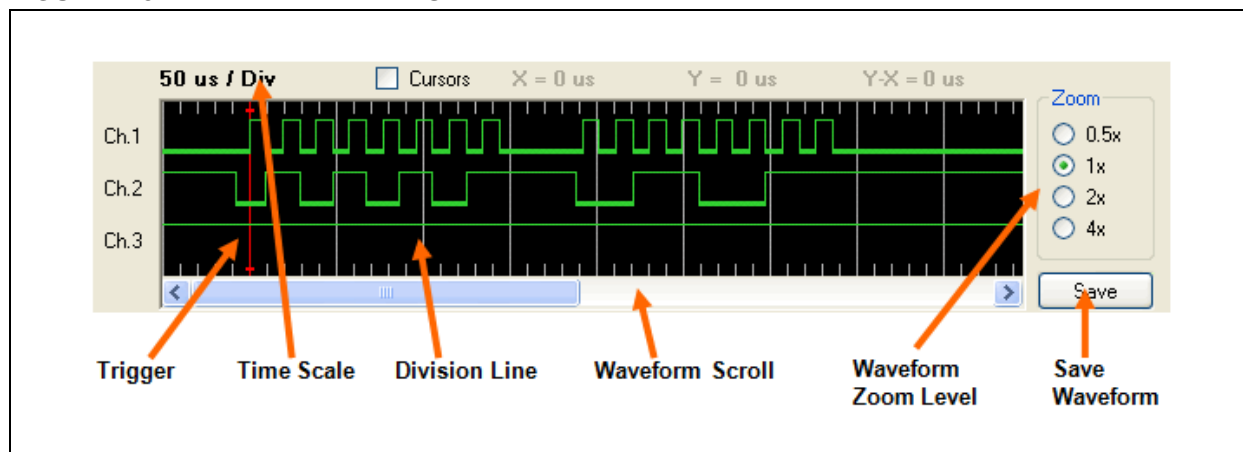


## 7.5.1 The Analyzer Display Section

The display section of the analyzer window allows the waveform to be viewed, zoomed, measured, and saved as a bitmap file.

Figure 7-8 shows a SPI bus waveform capture of a 2-byte transmission, and details the elements of the display window section.

**FIGURE 7-8: ANALYZER DISPLAY**



### 7.5.1.1 TRIGGER

The trigger is a predefined event in the monitored signals that causes a capture of the signal waveform. Triggering is discussed in detail in **Section 7.5.3 “The Analyzer Trigger Section”**.

On the waveform display, the point where the trigger occurred is indicated by a vertical red line. In Figure 7-8, the trigger was set to occur at the first rising edge of Channel 1, the SPI SCLK clock signal.

### 7.5.1.2 TIME SCALE

Above left of the waveform display is the time scale. This is how much time each Division Line in the waveform represents. In Figure 7-8, each division is 50 microseconds of time.

### 7.5.1.3 DIVISION LINE

A division line is a gray vertical line across the waveform display, which can be used to give a time reference to the displayed waveform. In Figure 7-8, the first 8 clocks on Channel 1 occupy about 3 divisions, so transmitting the first SPI byte took about  $3 \times 50 = 150$  us.

Smaller hash marks at the top and bottom of the display subdivide each time division into 5 smaller units. Since each time division in Figure 7-8 is 50 us, the smaller hash marks represent 10 us of time.

### 7.5.1.4 WAVEFORM SCROLL

The captured waveform is longer than can be shown all at once effectively in the display, so the horizontal scroll bar allows the display to scroll for viewing the entire waveform.



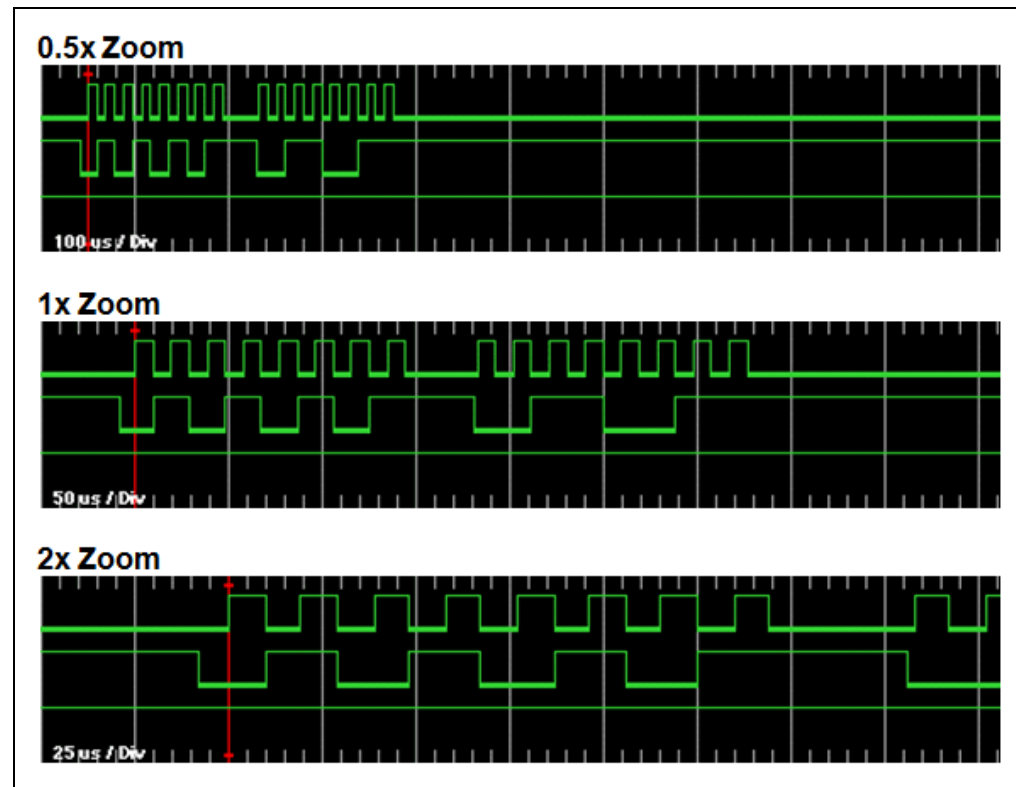
## 7.5.1.5 WAVEFORM ZOOM LEVEL

The waveform Zoom allows four levels of zoom to be selected. Normally, at 1x zoom each sample of a waveform is displayed as a pixel. A waveform is 1024 pixels, of which 500 can be displayed in the window. By selecting zoom level “0.5x”, the waveform is compressed so two samples are shown per pixel, which allows the entire waveform to be view at once, but with a loss of detail.

Zoom levels 2x and 4x display the waveform with 2 pixels per sample and 4 pixels per sample, respectively. This allows relative time details between the waveforms to be more easily seen.

Figure 7-9 shows the SPI waveform at zoom levels of 0.5x, 1x, and 2x. Note that the time scale changes as the zoom is changed.

**FIGURE 7-9: ANALYZER WINDOW SECTIONS**



## 7.5.1.6 SAVE WAVEFORM

Click the **Save** button to save the current waveform display in a bitmap file. The time scale will be added to the bottom of the display, as shown in Figure 7-9. If cursors are active, the cursors and their times will also be saved with the display.

Note that the entire waveform is saved. In Figure 7-9, the 1x and 2x waveforms were truncated after saving to only show the first portion.

## 7.5.2 The Analyzer Display Cursors

The display cursors are useful for making time and frequency measurements in the displayed waveform.

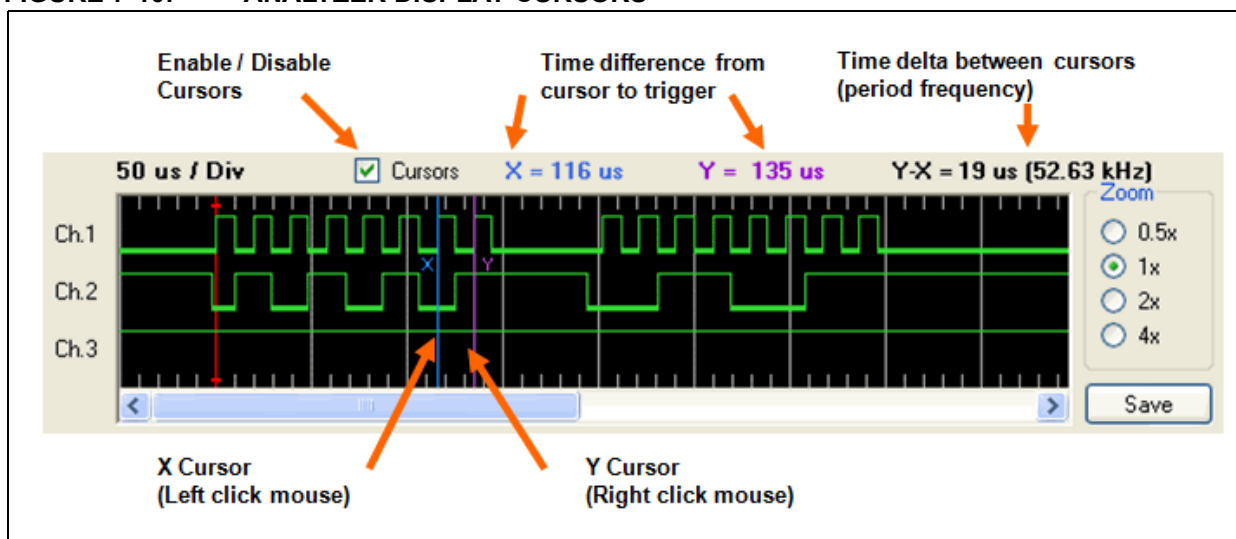
1. Click to check the “Cursors” checkbox and enable the cursors.
2. Place the X cursor by left-clicking in the waveform display.
3. Place the Y cursor by right-clicking in the waveform display.

It can be helpful to use Zoom for exact placement of the cursors. When zoomed, the cursors will get “wider” as they are the width of a sample and the sample width grows with increasing zoom.

Above the waveform display, the time difference between the Trigger and each cursor is displayed, along with the difference between the triggers. The time period between the cursor is also displayed as the related frequency.

In Figure 7-10, the cursors are used to measure the period (19  $\mu$ s) and frequency (53 kHz) of the SPI clock in channel 1. The X cursor sample is 116  $\mu$ s after the Trigger sample, and the Y cursor point is 135  $\mu$ s after the Trigger.

**FIGURE 7-10: ANALYZER DISPLAY CURSORS**



## 7.5.3 The Analyzer Trigger Section

The “trigger” is a user-defined set of events in the monitored signals that causes the capture of a waveform.

Each channel can be assigned one of the following trigger events:

'*' (Don't Care)	The analyzer channel is ignored for triggering purposes
'1' (Logic High)	The channel must be at a logic high state to trigger
'0' (Logic Low)	The channel must be at a logic low state to trigger
'/' (Rising Edge)	The channel must transition from low to high states to trigger
'\' (Falling Edge)	The channel must transition from high to low states to trigger

All trigger events on all channels must happen at once in order for the trigger to activate data capture. For example, for Figure 7-8, the trigger was set to simply detect the first rising edge on channel 1:

Figure 7-9 Trigger Conditions:

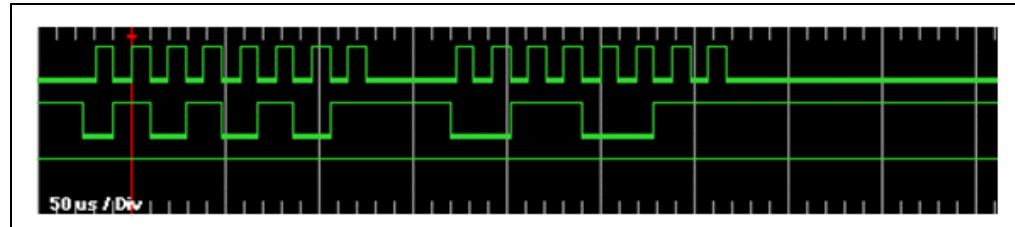
Ch 1 = /	(rising edge)
Ch 2 = *	(ignore)
Ch 3 = *	(ignore)

If the trigger conditions are changed as follows, where *both* a rising edge must be detected on channel 1 at the same time channel 2 is at a logic high state, the trigger will happen on the *second* clock instead, as shown in Figure 7-11. During first clock's rising edge, channel 2 is logic low; so, this does not fully satisfy the trigger condition.

Figure 7-11 Trigger Conditions:

Ch 1 = /	(rising edge)
Ch 2 = 1	(logic high)
Ch 3 = *	(ignore)

**FIGURE 7-11: TRIGGER CH 1 RISING EDGE WHEN CH 2 IS HIGH**

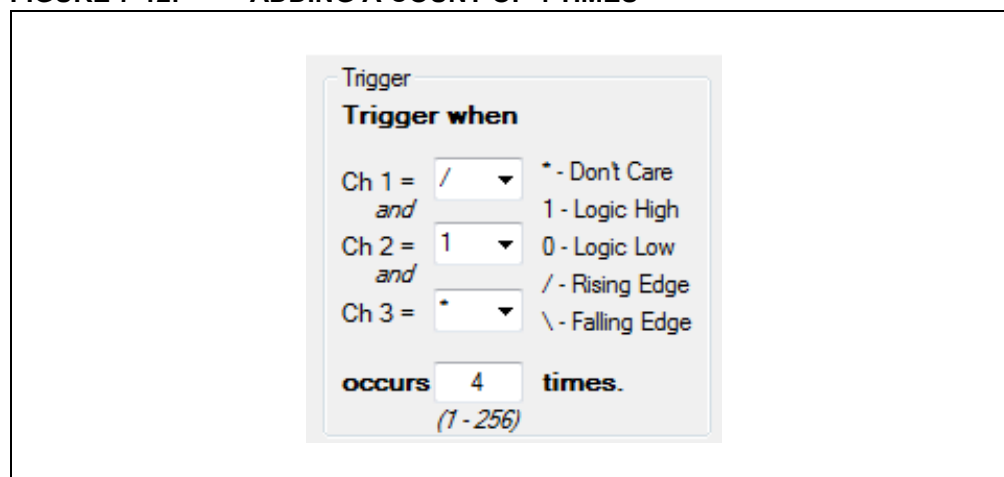


Finally, it is also possible to specify how many times the trigger condition must occur before waveform capture is initiated, up to 256 times. For example, suppose we wanted to capture the 16th byte of a long SPI transmission sequence. If we triggered on the first clock edge of the first byte, we probably wouldn't be able to see the 16th byte, as the analyzer would stop sampling before it occurred. However, we can set the analyzer to pass up the first 15 bytes, by setting the trigger count to 15 bytes \* 8 clocks + 1 = 121 times. This way, it will start counting clock edges on the first byte, but it won't trigger the data capture until the 16th byte is transmitted.

# PICkit™ 3 Programmer Application User's Guide

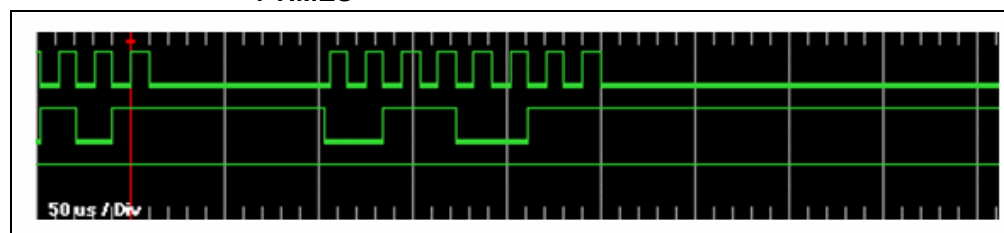
To illustrate, we'll add a count of 4 times to our trigger conditions of Figure 7-11:

**FIGURE 7-12:      ADDING A COUNT OF 4 TIMES**



Now, we'll trigger on the 4<sup>th</sup> rising edge of channel 1 that occurs while channel 2 is logic high. This will be the last clock of the first SPI byte, as shown in Figure 7-13.

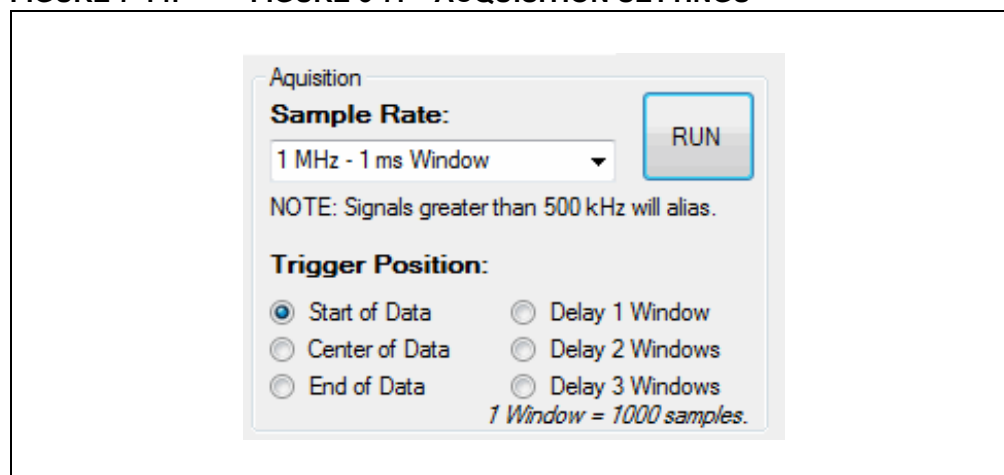
**FIGURE 7-13:      TRIGGER CH 1 RISING EDGE WHEN CH 2 IS HIGH OCCURS 4 TIMES**



## 7.5.4 The Analyzer Acquisition Section

The "Acquisition" section of the analyzer window is used to set the waveform sample rate, the position of the trigger relative to the captured waveform, and to start or "run" the analyzer.

**FIGURE 7-14:      FIGURE 3-7: ACQUISITION SETTINGS**



## 7.5.4.1 SAMPLE RATE

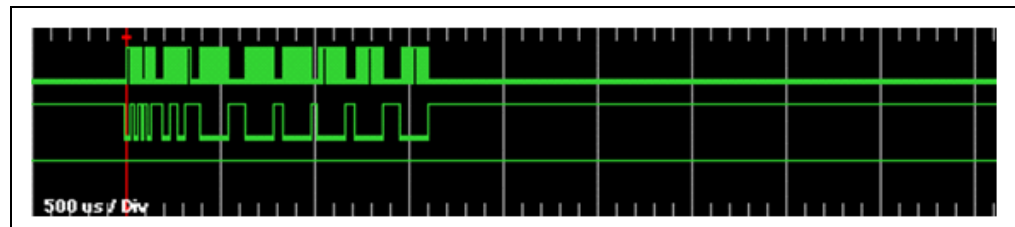
The sample rate is how often the analyzer channels are looked at. Each waveform capture is only 1024 samples long, so if we want to look at a longer period of time in the waveform display, we have to sample less often.

The trade-off is that at higher sample rates, we can see more detail and faster signals but only a small window of time. At lower sample rates, we can see a longer window of elapsed time, but at less detail and we may miss fast pulses.

Generally, the sample rate should be set at least 10 times the highest frequency or 5 times the fastest pulse width to get a decent representation of the waveform. Any waveform that has frequency higher than half the sample rate may *alias*. Aliasing means that waveform edges are missed, so the waveform can appear slower than it actually is.

Of course, the sample rate can always be set slower than these limits if the intention is simply to get a general idea of what's going on in the circuit without much detail. For example, the sample rate could be set much slower so it could be seen how many SPI bytes are being sent. In Figure 7-15 we can see that 8 bytes are being sent on the SPI bus. Now that we know how many bytes are sent, we could set the sample rate higher and adjust the trigger count to see each byte in detail to figure out the byte value.

**FIGURE 7-15: 8 SPI BYTES AT SLOW SAMPLE RATE**



In the PICkit 3 Logic Tool analyzer, the sample rate may be selected from those shown in Table .

**TABLE 7-1: SUPPORTED SAMPLE RATES**

Sample Rate	Time Between Samples	Waveform Limitations	
		Captured Waveform Length <sup>1</sup> (1024 samples)	Maximum Frequency (before aliasing)
1 MHz	1 us	1 ms	500 kHz
500 kHz	2 us	2 ms	250 kHz
250 kHz	4 us	4.1 ms	125 kHz
100 kHz	10 us	10.2 ms	50 kHz
50 kHz	20 us	20.5 ms	25 kHz
25 kHz	40 us	41 ms	12.5 kHz
10 kHz	100 us	102.4 ms	5 kHz
5 kHz	200 us	204.8 ms	2.5 kHz

**Note:** Waveform length is rounded to the nearest 0.1 decimal place.

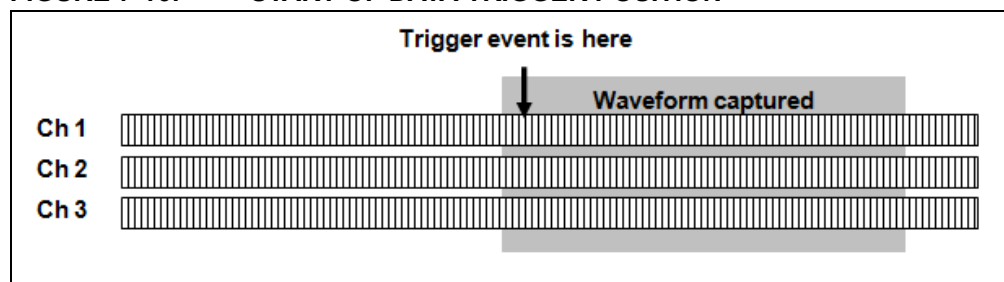
## 7.5.4.2 TRIGGER POSITION

Changing the trigger position allows more flexibility over how the captured data relates to the trigger event. For example, we might be more interested in what happened *before* the trigger, rather than after. There are six selectable trigger positions.

### Start of Data Trigger Position

This is the trigger position used in all the prior Figure waveforms. All the waveform data, except for one division, is captured after the trigger occurs. This is best used when all the waveform data of interest happens after the trigger.

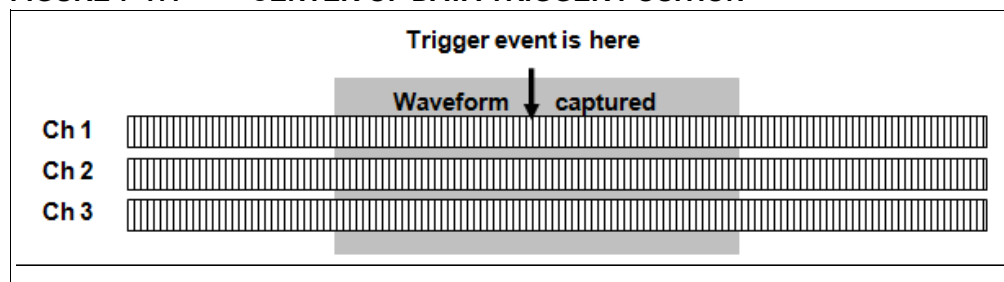
**FIGURE 7-16: START OF DATA TRIGGER POSITION**



### Center of Data Trigger Position

This is best used when all the waveform data of interest happens around the trigger. The trigger event is in the middle of the waveform display.

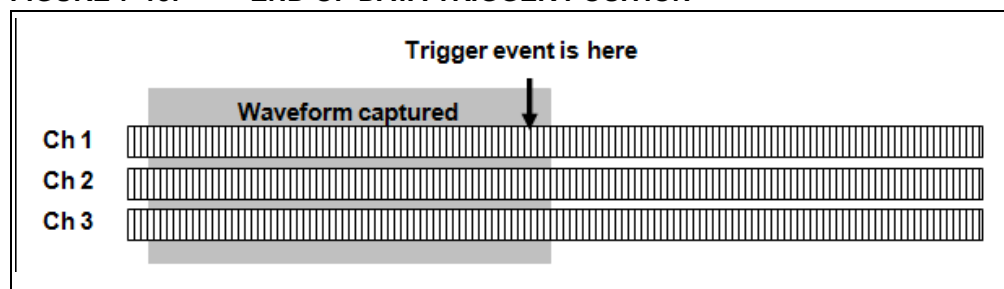
**FIGURE 7-17: CENTER OF DATA TRIGGER POSITION**



### End of Data Trigger Position

All the waveform data, except for just over one division, is captured prior to when the trigger occurs. This is best used when all the waveform data of interest happens before the trigger.

**FIGURE 7-18: END OF DATA TRIGGER POSITION**



## Delay 1 Window Trigger Position

## Delay 2 Windows Trigger Position

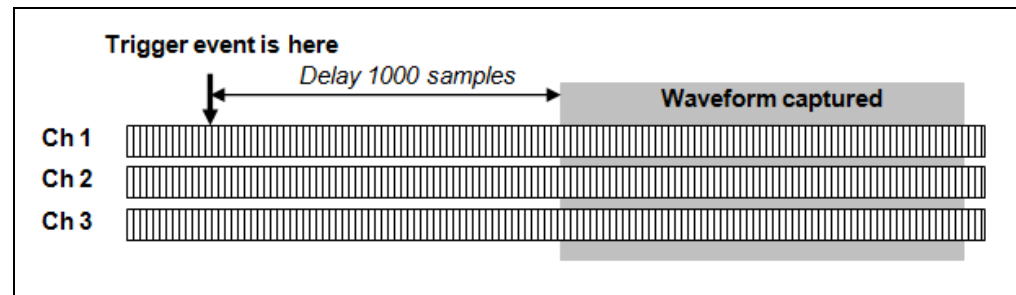
## Delay 3 Windows Trigger Position

In these cases, the trigger position is considered “Start of Data”; but, the waveform capture is delayed 1000 samples (nearly one waveform display-width) after the trigger. This allows a user to capture events that occur farther out than the display-width after a trigger happens, without reducing the sample rate.

In other words, when “Delay 1 Window” is selected, the analyzer will wait 1000 samples after the trigger event occurs before it begins recording waveform data. When “Delay 2 Windows” is selected, it will wait 2000 samples, etc.

Each waveform display is 1024 samples, so the 1000 sample delay increment gives a small overlap between successive delay captures. Assuming that the data of interest after the trigger is repeatable and consistent, this allows a total waveform of up to 4 times the sample-rate-window-width to be pieced together. For example, it would be possible to collect 4 ms worth of waveform data after a trigger event at the 1 MHz sample rate.

**FIGURE 7-19: DELAY 1 WINDOW TRIGGER POSITION**



# PICkit™ 3 Programmer Application User's Guide

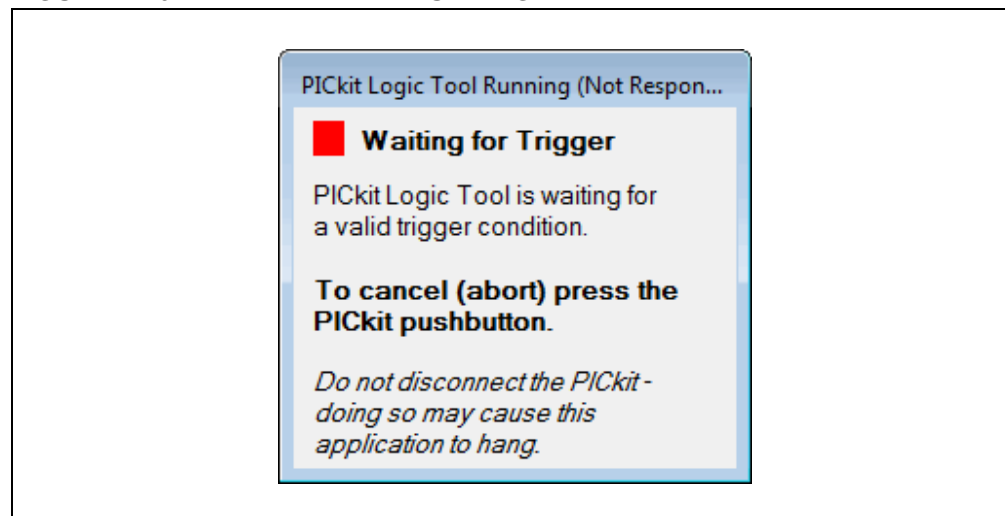
---

## 7.5.4.3 RUNNING THE ANALYZER

Once the trigger conditions, sample rate, and trigger position are set as desired, click the **RUN** button to begin collecting waveform data and looking for trigger events.

When the analyzer is running, it will show the dialog in Figure 7-20 and the “Busy” LED on the PICkit 3 unit will be lit.

**FIGURE 7-20: ANALYZER RUNNING**



When the trigger condition is met, the “Busy” LED turns off, the “Waiting for Trigger” dialog closes, and the analyzer waveform display updates with newly captured data.

If the analyzer is not triggering as expected; or if, for any other reason, it is necessary to stop the analyzer from running, press the PICkit 3 unit push button. When an analyzer run is canceled, the waveform display is not updated.

**Note:** When the analyzer is running, the PICkit 3 unit is unable to service USB requests. The application will wait until the run completes at a trigger condition or is canceled by the push button. If the PICkit 3 unit is unplugged from USB during a run, the PICkit 3 application may hang because it is waiting for a response from the PICkit 3 unit.



---

## **Appendix A. Hardware Specification**

---

### **A.1 INTRODUCTION**

The hardware and electrical specifications of the PICKit 3 programmer system are detailed.

### **A.2 HIGHLIGHTS**

This chapter discusses:

- Declaration of Conformity
- Device Support
- USB Port/Power
- PICKit 3 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 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 programmer

Due to the special nature of the PICKit 3 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.).

# PICkit™ 3 Programmer Application User's Guide

---

## A.4 DEVICE SUPPORT

Refer to the PICkit 3 Readme for a list of supported devices.

**Note:** The UNI/O (I<sup>2</sup>C) Serial EEPROM devices require the following PICkit 3 hardware consideration or change to work properly:

Application pull-up resistor needs to be 9.1k ohm or less  
or  
Remove R50 from the PICkit 3.

**Note:** The I<sup>2</sup>C (24LC) Serial EEPROM devices require the following PICkit 3 hardware changes to work properly:

Remove TR3 from the PICkit 3.  
Remove R50 from the PICkit 3.

## A.5 USB PORT/POWER

The PICkit 3 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 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.6 PICKIT 3 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.6.1 Main Board

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

### A.6.2 Indicator Lights (LEDs)

The indicator lights have the following significance.

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

# PICkit™ 3 Programmer Application User's Guide

## A.7 STANDARD COMMUNICATION HARDWARE

For standard debugger communication with a target (**Section 4.3 “PICkit 3 to Target Communication”**, and “Standard ICSP Device Communication”), use an adapter with an RJ-11 connector.

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

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

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

### A.7.1 Standard Communication

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

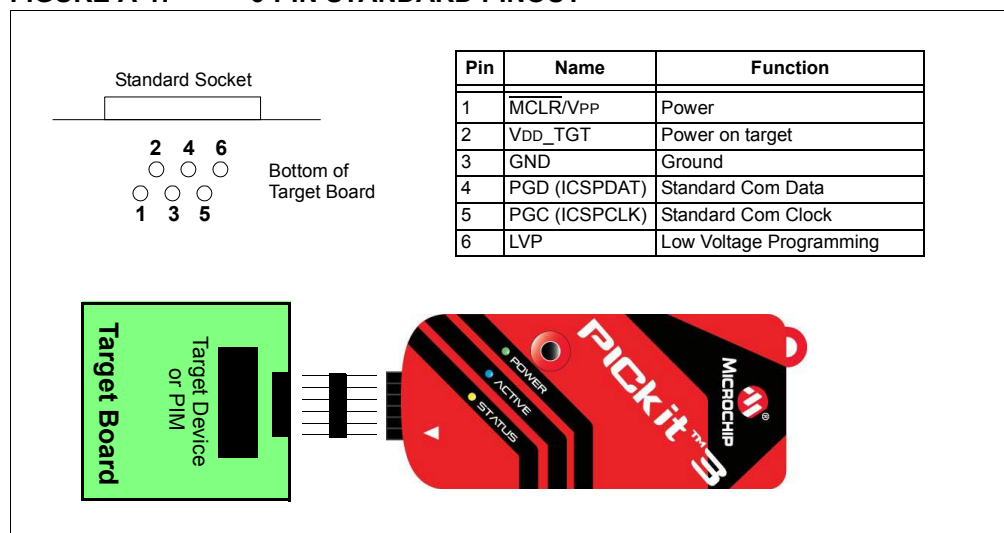
The VPP high-voltage lines can produce a variable voltage that can swing from 1.8 to 14 volts to satisfy the voltage requirements for the specific emulation processor.

The VDD sense connection draws current from the target processor.

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

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

**FIGURE A-1: 6-PIN STANDARD PINOUT**



## A.7.2 Modular Cable and Connector

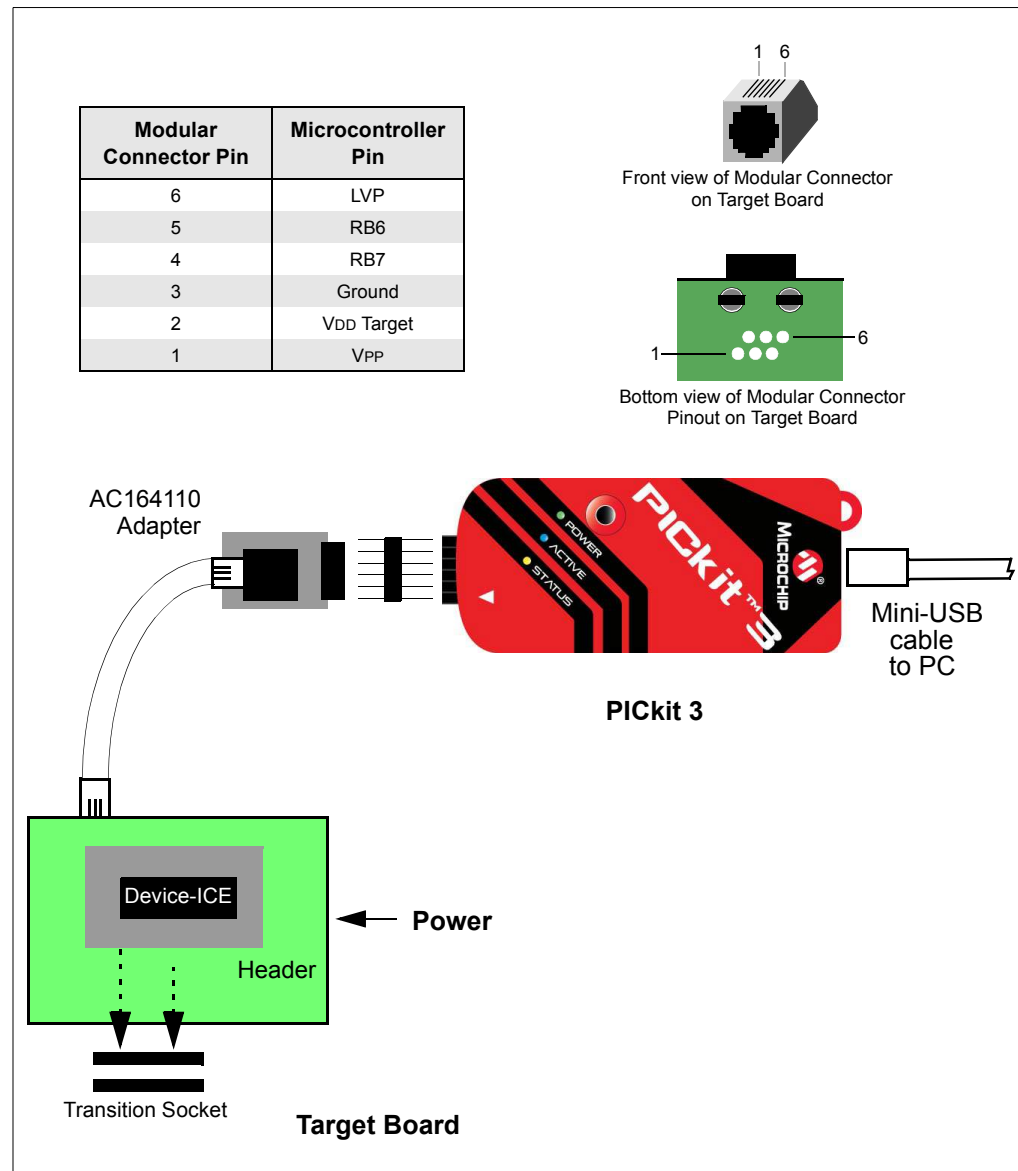
For standard communications, a modular cable connects the debugger and the target application. The specifications for this cable and its connectors are listed below.

### A.7.2.1 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**



## A.7.2.2 MODULAR PLUG SPECIFICATION

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

## A.7.2.3 MODULAR CABLE SPECIFICATION

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

## A.8 TARGET BOARD CONSIDERATIONS

The target board should be powered according to the requirements of the selected device (1.8V-5.5V) and the application.

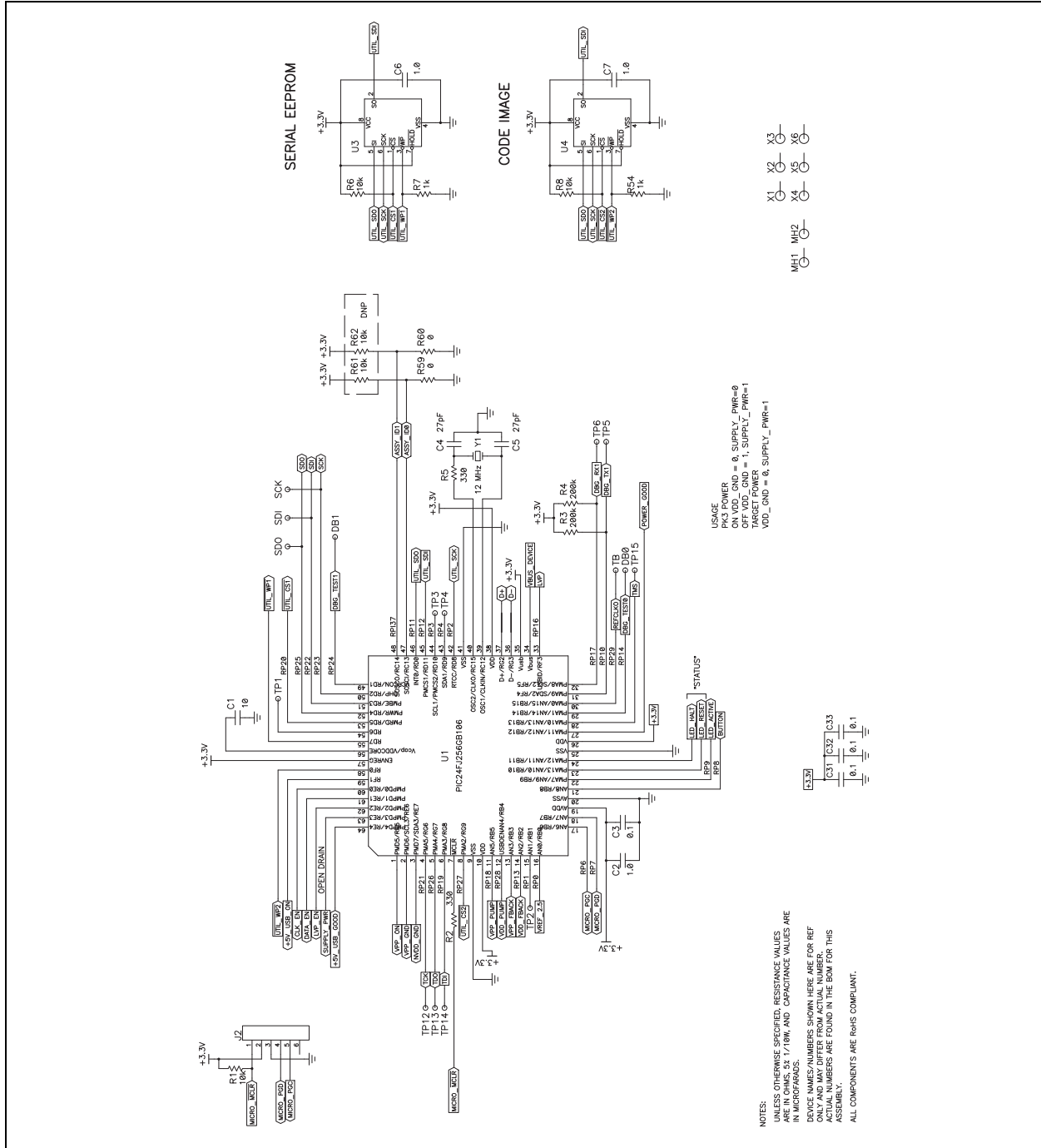
Depending on the type of debugger-to-target communications used, there will be some considerations for target board circuitry:

- **Section 4.4.2 “Target Connection Circuitry”**
- **Section 4.4.5 “Circuits That Will Prevent the PICkit 3 From Functioning”**

## Appendix B. PICKit 3 Schematics

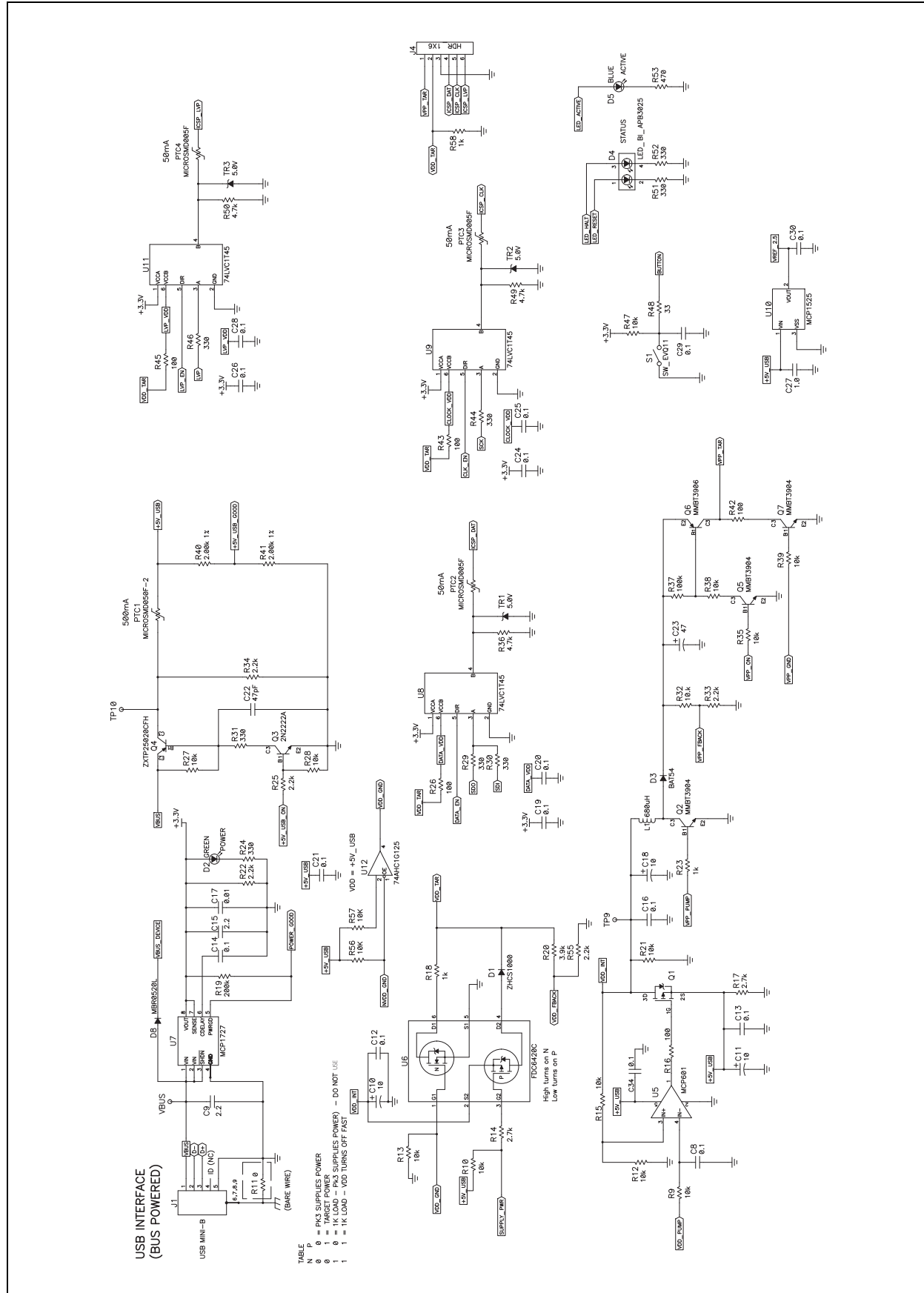
PICKit 3 Development Programmer schematic diagrams are shown here. Demo board schematics are found in their respective user's guides.

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



# PICKit™ 3 Programmer Application User's Guide

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





---

## Glossary

---

**Absolute Section**

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

**Access Memory**

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

**Access Entry Points**

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

**Address**

Value that identifies a location in memory.

**Alphabetic Character**

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

**Alphanumeric**

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

**ANDed Breakpoints**

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

**Anonymous Structure**

C30 – An unnamed structure.

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

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

**ANSI**

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

**Application**

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

**Archive**

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

**Archiver**

A tool that creates and manipulates libraries.

**ASCII**

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

**Assembler**

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

**Assembly Language**

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

**Assigned Section**

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

**Asynchronously**

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

**Asynchronous Stimulus**

Data generated to simulate external inputs to a simulator device.

**Attribute**

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

**Attribute, Section**

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

**Binary**

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

**Bookmarks**

Use bookmarks to easily locate specific lines in a file.

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

**Breakpoint**

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

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

**Build**

Compile and link all the source files for an application.

**C**

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

**Calibration Memory**

A special function register or registers used to hold values for calibration of a PIC microcontroller on-board RC oscillator or other device peripherals.

**Central Processing Unit**

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

**Clean**

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

**COFF**

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

**Command Line Interface**

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

**Compiler**

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

**Conditional Assembly**

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

**Conditional Compilation**

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

**Configuration Bits**

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

**Control Directives**

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

**CPU**

See Central Processing Unit.

**Cross Reference File**

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

**Data Directives**

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

## **Data Memory**

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

## **Debugger**

Hardware that performs debugging.

## **Debugger System**

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

## **Debugging Information**

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

## **Deprecated Features**

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

## **Device Programmer**

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

## **Digital Signal Controller**

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

## **Digital Signal Processing**

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

## **Digital Signal Processor**

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

## **Directives**

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

## **Download**

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

## **DSC**

See Digital Signal Controller.

## **DSP**

See Digital Signal Processor.

## **dsPIC DSCs**

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

## **DWARF**

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

## **EEPROM**

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

**ELF**

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

**Emulation**

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

**Emulation Memory**

Program memory contained within the emulator.

**Emulator**

Hardware that performs emulation.

**Emulator System**

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

**Endianness**

The ordering of bytes in a multi-byte object.

**Environment**

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

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

**Epilogue**

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

**EPROM**

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

**Error File**

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

**Errors**

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

**Event**

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

**Executable Code**

Software that is ready to be loaded for execution.

**Export**

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

## **Expressions**

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

## **Extended Microcontroller Mode**

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

## **Extended Mode**

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

## **External Label**

A label that has external linkage.

## **External Linkage**

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

## **External Symbol**

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

## **External Symbol Resolution**

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

## **External Input Line**

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

## **External RAM**

Off-chip Read/Write memory.

## **Fatal Error**

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

## **File Registers**

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

## **Filter**

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

## **Flash**

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

## **FNOP**

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

**Frame Pointer**

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

**Free-Standing**

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

**GPR**

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

**Halt**

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

**Heap**

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

**Hex Code**

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

**Hex File**

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

**Hexadecimal**

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

**High Level Language**

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

**ICD**

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

**ICE**

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

**ICSP**

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

**IDE**

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

**Identifier**

A function or variable name.

## **IEEE**

Institute of Electrical and Electronics Engineers.

## **Import**

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

## **Initialized Data**

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

```
int myVar=5;
```

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

## **Instruction Set**

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

## **Instructions**

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

## **Internal Linkage**

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

## **International Organization for Standardization**

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

## **Interrupt**

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

## **Interrupt Handler**

A routine that processes special code when an interrupt occurs.

## **Interrupt Request**

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

## **Interrupt Service Routine**

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

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

## **Interrupt Vector**

Address of an interrupt service routine or interrupt handler.

## **IRQ**

See Interrupt Request.

## **ISO**

See International Organization for Standardization.

## **ISR**

See Interrupt Service Routine.



**L-value**

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

**Latency**

The time between an event and its response.

**Librarian**

See Archiver.

**Library**

See Archive.

**Linker**

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

**Linker Script Files**

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

**Listing Directives**

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

**Listing File**

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

**Little Endian**

A data ordering scheme for multibyte data whereby the least significant byte is stored at the lower addresses.

**Local Label**

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

**Logic Probes**

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

**Loop-Back Test Board**

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

**LVDS**

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

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

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

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

Low voltage means that the standard 5 volts is replaced by either 3.3 volts or 1.5 volts. LVDS uses a dual wire system, running 180 degrees of each other. This enables noise to travel at the same level, which in turn can get filtered more easily and effectively. With standard I/O signaling, data storage is contingent upon the actual voltage level. Voltage level can be affected by wire length (longer wires increase resistance, which lowers voltage). But with LVDS, data storage is distinguished only by positive and negative voltage values, not the voltage level. Therefore, data can travel over greater lengths of wire while maintaining a clear and consistent data stream.

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

## **Machine Code**

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

## **Machine Language**

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

## **Macro**

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

## **Macro Directives**

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

## **Makefile**

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

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

## **Make Project**

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

## **MCU**

Microcontroller Unit. An abbreviation for microcontroller. Also `uC`.

## **Memory Model**

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

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

## **Message**

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

## **Microcontroller**

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

## **Microcontroller Mode**

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

## **Microprocessor Mode**

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

## **Mnemonics**

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

## **MPASM™ Assembler**

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

## **MPLAB Language Tool for Device**

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

## **MPLAB ICD**

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

## **MPLAB ICE 2000/4000**

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

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

## **MPLAB IDE**

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

## **MPLAB PM3**

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

## **MPLAB REAL ICE™ In-Circuit Emulator**

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

## **MPLAB SIM**

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

## **MPLIB™ Object Librarian**

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

## **MPLINK™ Object Linker**

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

## **MRU**

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

## **Native Data Size**

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

## **Nesting Depth**

The maximum level to which macros can include other macros.

## **Node**

MPLAB IDE project component.

## **Non-Extended Mode**

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

## **Non Real Time**

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

## **Non-Volatile Storage**

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

## **NOP**

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

## **Object Code**

The machine code generated by an assembler or compiler.

## **Object File**

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

## **Object File Directives**

Directives that are used only when creating an object file.

## **Octal**

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

## **Off-Chip Memory**

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

## **One-to-One Project-Workspace Model**

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

## **Opcodes**

Operational Codes. See Mnemonics.

## **Operators**

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

## **OTP**

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

## **Pass Counter**

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

## **PC**

Personal Computer or Program Counter.

## **PC Host**

Any PC running a supported Windows operating system.

## **Persistent Data**

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

## **Phantom Byte**

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

## **PIC MCUs**

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

## **PICKit 1, 2, and 3**

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

## **PICSTART Plus**

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

## **Plug-ins**

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

## Pod

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

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

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

## Power-on-Reset Emulation

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

## Pragma

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

## Precedence

Rules that define the order of evaluation in expressions.

## PRO MATE II

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

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

## Production Programmer

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

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

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

## Profile

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

## Program Counter

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

## Program Counter Unit

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

## **Program Memory**

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

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

## **Project**

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

## **Prologue**

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

## **Prototype System**

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

## **PWM Signals**

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

## **Qualifier**

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

## **Radix**

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

## **RAM**

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

## **Raw Data**

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

## **Read Only Memory**

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

## **Real Time**

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

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

## **Real-Time Watch**

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

## **Recursive Calls**

A function that calls itself, either directly or indirectly.

## **Recursion**

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

## **Reentrant**

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

## **Relaxation**

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

## **Relocatable**

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

## **Relocatable Section**

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

## **Relocation**

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

## **ROM**

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

## **Run**

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

## **Run-time Model**

Describes the use of target architecture resources.

## **Scenario**

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

## **Section**

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

## **Section Attribute**

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

## **Sequenced Breakpoints**

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

## **Serialized Quick Turn Programming**

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

## **SFR**

See Special Function Registers.



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

## **Skew**

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

## **Skid**

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

## **Source Code**

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

## **Source File**

An ASCII text file containing source code.

## **Special Function Registers**

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

## **SQTP**

See Serialized Quick Turn Programming.

## **Stack, Hardware**

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

## **Stack, Software**

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

## **MPLAB Starter Kit for Device**

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

## Static RAM or SRAM

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

## Status Bar

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

## Step Into

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

## Step Over

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

## Step Out

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

## Stimulus

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

## Stopwatch

A counter for measuring execution cycles.

## Storage Class

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

## Storage Qualifier

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

## Symbol

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

## Symbol, Absolute

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

## System Window Control

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

## Target

Refers to user hardware.

**Target Application**

Software residing on the target board.

**Target Board**

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

**Target Processor**

The microcontroller device on the target application board.

**Template**

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

**Tool Bar**

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

**Trace**

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

**Trace Memory**

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

**Trace Macro**

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

**Trigger Output**

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

**Trigraphs**

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

**Unassigned Section**

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

**Uninitialized Data**

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

```
int myVar;
```

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

**Upload**

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

**USB**

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

**Vector**

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

**Warning**

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

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

**Watch Variable**

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

**Watch Window**

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

**Watchdog Timer**

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

**WDT**

See Watchdog Timer.

**Workbook**

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

**Workspace**

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

### Index

<b>A</b>		Install Hardware .....	13
Auto Import Hex + Write Device.....	22	Install Software.....	14
Auto-Detect .....	31	<b>L</b>	
<b>B</b>		LEDs .....	11, 60
Blank Check .....	22, 30	<b>M</b>	
<b>C</b>		Microsoft Windows .....	35
Cable Length.....	28	<b>O</b>	
Cables		Operating System .....	31
Length .....	60, 64	Operating System, Update .....	41
Calibrate VDD & Set Unit ID .....	31	OSCCAL .....	31
Check Communication .....	31	<b>P</b>	
Code Protect .....	21, 31	PC, Power Down .....	60
Connector, 6-Pin .....	11	PGC/PGD.....	26
<b>D</b>		PICKit 3 Defined .....	9
Data EEPROM Memory .....	34	Power-Down mode.....	60
Device Selection .....	15, 32	Program Memory .....	34
Documentation		Programmer Application.....	29
Conventions .....	6	<b>R</b>	
Layout .....	5	Read.....	21
Download PICKit 3 Programmer Operating System. 31		Read Device.....	30
Driver Board		Read Device + Export Hex File .....	22
Standard .....	62	Reading, Recommended .....	7
Durability, Card Guide.....	60	Readme.....	7
<b>E</b>		<b>S</b>	
Enable Code Protect .....	31	Standard Communication	
Erase.....	22, 30	Driver Board.....	62
Export Hex .....	30	<b>T</b>	
<b>F</b>		Target Power.....	17
Fast Programming .....	31	Target Vdd Source .....	31
Firmware .....	31	Troubleshooting .....	35
Firmware, Update .....	41	<b>U</b>	
Force PICKit 2 .....	31	Update Firmware.....	41
Force Target .....	31	Update Operating System.....	41
<b>H</b>		USB.....	60, 85
Hex File, Import.....	19	Hubs.....	60
Hibernate mode .....	60	USB Port .....	10
Hold Device in Reset .....	30	Use VPP First Program Entry .....	31
Hubs, USB .....	60	<b>V</b>	
<b>I</b>		Vdd.....	18, 27, 33, 35
ICSP.....	25, 62	Verify .....	20, 30
ICSPCLK.....	62	Verify on Write.....	30
ICSPCLK/ICSPDAT .....	26	Vpp.....	26, 35
ICSPDAT .....	62	Vss .....	28
Import Hex .....	30		
Import Hex File.....	19		
Indicator Lights.....	60		

# PICkit™ 3 Programmer Application User's Guide

---

**W**

Watchdog Timer ..... 86

Write ..... 19

Write Device ..... 30

Write on PICkit Button ..... 30

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](http://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