



MPLAB® PICKit™ 4 In-Circuit Debugger User's Guide

Notice to Customers



Important:

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 website (www.microchip.com) to obtain the latest documentation available.

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

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



Table of Contents

Notice to Customers.....	1
1. Introduction.....	4
1.1. Document Layout.....	4
1.2. Conventions Used in This Guide.....	4
1.3. Recommended Reading.....	5
2. About the Debugger.....	7
2.1. MPLAB PICKit 4 In-Circuit Debugger Description.....	7
2.2. MPLAB PICKit 4 In-Circuit Debugger Advantages.....	7
2.3. MPLAB PICKit 4 In-Circuit Debugger Components.....	8
2.4. MPLAB PICKit 4 Block Diagram.....	9
3. Operation.....	10
3.1. Debugger to Target Communication.....	10
3.2. Target Communication Connections.....	11
3.3. Debugging.....	14
3.4. Requirements for Debugging.....	15
3.5. Programming.....	17
3.6. Resources Used by the Debugger.....	17
4. Debugger Usage.....	18
4.1. Installation and Setup.....	18
4.2. Debug Tutorial.....	18
4.3. Quick Debug/Program Reference.....	18
4.4. Debugger Limitations.....	19
4.5. Common Debug Features.....	19
4.6. Connecting the Target Board.....	20
4.7. Setting Up the Target Board.....	20
4.8. Setting Up MPLAB X IDE.....	21
4.9. Starting and Stopping Debugging.....	21
4.10. Viewing Processor Memory and Files.....	22
4.11. Breakpoints and Stopwatch.....	22
5. MPLAB PICKit 4 Programmer-To-Go.....	24
5.1. Power Requirements for Programmer-To-Go.....	24
5.2. Limitations for Programmer-To-Go.....	25
5.3. Setting up PICKit 4 for Programmer-To-Go Mode.....	26
5.4. Using Programmer-To-Go.....	30
5.5. Exiting Programmer-To-Go Mode.....	31
6. Troubleshooting.....	32
6.1. Some Questions to Answer First.....	32
6.2. Top Reasons Why You Can't Debug.....	32
6.3. Other Things to Consider.....	33
7. Frequently Asked Questions.....	36

7.1.	How Does it Work?	36
7.2.	What's Wrong?	36
8.	Error Messages	38
8.1.	Types of Error Messages	38
8.2.	General Corrective Actions	47
9.	Engineering Technical Notes (ETNs)	49
10.	Debugger Function Summary	50
10.1.	Debugger Selection and Switching	50
10.2.	Debugger Options Selection	50
11.	Hardware Specification	56
11.1.	USB Connector	56
11.2.	MPLAB PICKit 4 In-Circuit Debugger	56
11.3.	Communication Hardware	58
11.4.	Target Board Considerations	61
12.	Revision History	63
12.1.	Revision A (May 2018)	63
12.2.	Revision B (August 2018)	63
12.3.	Revision C (October 2018)	63
12.4.	Revision D (January 2020)	63
13.	Support	64
13.1.	Warranty Registration	64
13.2.	myMicrochip Personalized Notification Service	64
14.	Glossary	65
	The Microchip Web Site	83
	Customer Change Notification Service	83
	Customer Support	83
	Microchip Devices Code Protection Feature	83
	Legal Notice	84
	Trademarks	84
	Quality Management System Certified by DNV	84
	Worldwide Sales and Service	85

1. Introduction

This chapter contains general information that will be useful to know before using the MPLAB® PICKit™ 4 In-Circuit Debugger.

1.1 Document Layout

This document describes how to use the MPLAB PICKit 4 In-Circuit Debugger as a development tool to emulate and debug firmware on a target board, as well as how to program devices. The document is organized as follows:

- [2. About the Debugger](#) – What the MPLAB PICKit 4 In-Circuit Debugger is and how it can help you develop your application.
- [3. Operation](#) – The theory of MPLAB PICKit 4 In-Circuit Debugger operation. Explains configuration options.
- [4. Debugger Usage](#) – A description of basic debug features available in MPLAB X IDE when the MPLAB PICKit 4 In-Circuit Debugger is chosen as the debug tool. This includes the debug features for breakpoints and stopwatch.
- [6. Troubleshooting](#) – The first things you should try if you are having issues with debugger operation.
- [7. Frequently Asked Questions](#) – A list of frequently asked questions, useful for troubleshooting.
- [8. Error Messages](#) – A list of error messages and suggested resolutions.
- [10. Debugger Function Summary](#) – A summary of debugger functions available in MPLAB X IDE when the MPLAB PICKit 4 In-Circuit Debugger is chosen as the debug or program tool.
- [11. Hardware Specification](#) – The hardware and electrical specifications of the debugger system.
- [Revision History](#) – A summary of changes to the document and when they were made.

1.2 Conventions Used in This Guide

This manual uses the following documentation conventions:

Table 1-1. Documentation Conventions

Description	Represents	Examples
Arial font:		
Italic characters	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 text with right angle bracket	A menu path	<u><i>File>Save</i></u>
Bold characters	A dialog button	Click OK
	A tab	Click the Power tab
N'Rnnnn	A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		

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

1.3 Recommended Reading

This user's guide describes how to use MPLAB PICKit 4 In-Circuit Debugger. Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

Multi-Tool Design Advisory (DS51764)

Please read this first! This document contains important information about operational issues that should be considered when using the MPLAB PICKit 4 In-Circuit Debugger with your target design.

MPLAB X IDE Online Help

This is an essential document to be used with any Microchip hardware tool.

This is an extensive help file for the MPLAB X IDE. It includes an overview of embedded systems, installation requirements, tutorials, details on creating new projects, setting build properties, debugging code, setting configuration bits, setting breakpoints, programming a device, etc. This help file is generally more up-to-date than the printable PDF of the user's guide (DS50002027) available as a free download at <https://www.microchip.com/mplabx/>.

Release Notes for MPLAB PICKit 4 In-Circuit Debugger

For the latest information on using MPLAB PICKit 4 In-Circuit Debugger, read the notes under "Release Notes and Support Documentation" on the MPLAB X IDE Start Page. The release notes contain update information and known issues that may not be included in this user's guide.

MPLAB PICKit 4 Quick Start Guide Poster (DS50002721)

This poster shows you how to connect the hardware and install the software for the MPLAB PICKit 4 In-Circuit Debugger using standard communications and a target board.

MPLAB PICKit 4 In-Circuit Debugger Online Help File

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

Processor Extension Pak and Header Specification (DS50001292)

This booklet describes how to install and use headers. Headers are used to better debug selected devices, without the loss of pins or resources. See also the PEP and Header online Help file.

2. About the Debugger

An overview of the MPLAB® PICKit™ 4 In-Circuit Debugger system is provided here.

2.1 MPLAB PICKit 4 In-Circuit Debugger Description

The MPLAB PICKit 4 In-Circuit Debugger (PG164140) allows fast and easy debugging and programming of Microchip PIC®, dsPIC®, AVR, SAM and CEC (Arm® Cortex®-M7-based) microcontrollers using the powerful graphical user interface of MPLAB X Integrated Development Environment (IDE).

The MPLAB PICKit 4 is connected to the design engineer's computer using a high-speed 2.0 USB interface and can be connected to the target via a Microchip debug 8-pin Single In-Line (SIL) connector. The connector uses two device I/O pins and the reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™). An additional microSDHC card slot and the ability to be self-powered from the target means you can take your code with you and program on the go.

The MPLAB PICKit 4 programs faster than its predecessor (PICKit 3) and comes ready to support PIC, dsPIC, AVR, SAM and CEC MCU devices. Along with a wider target voltage, the MPLAB PICKit 4 supports advanced interfaces such as 4-wire JTAG, Serial Wire Debug (SWD), and streaming Data Gateway¹, while being backward compatible for demo boards, headers and target systems using 2-wire JTAG and ICSP. The MPLAB PICKit 4 also has a unique Programmer-To-Go function with the addition of a microSDHC card slot to hold project code and the ability to be powered by the target board.

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

The MPLAB PICKit 4 In-Circuit Debugger is compatible with any of these platforms:

- Microsoft Windows® 7 or later
- Linux®
- macOS™

The MPLAB PICKit 4 In-Circuit Debugger was developed for debugging embedded processors with rich debug facilities which are different from conventional system processors in the following aspects:

- Processors run at maximum speeds.
- Capability to incorporate I/O port data input.
- Advanced host communication interfaces (Windows, macOS and Linux).
- Advanced communication mediums and protocols.
- Faster programming times.

In addition to debugger functions, the MPLAB PICKit 4 In-Circuit Debugger system also may be used as a device production programmer.

Note: ¹ The functionality will be available in a future firmware update of the product through MPLAB X IDE.

2.2 MPLAB PICKit 4 In-Circuit Debugger Advantages

The MPLAB PICKit 4 In-Circuit Debugger system provides the following advantages:

Features/Capabilities:

- Connects to computer via high-speed USB 2.0 (480 Mbps/s) cable.
- An 8-pin SIL programming connector and the option to use various interfaces.
- Programs devices using MPLAB X IDE or MPLAB IPE.
- Supports multiple hardware and software breakpoints, stopwatch, and source code file debugging.
- Debugs your application on your own hardware in real time.
- Sets breakpoints based on internal events.

- Monitors internal file registers.
- Debugs at full speed.
- Configures pin drivers.
- Field-upgradeable through an MPLAB X IDE firmware download.
- Adds new device support and features by installing the latest version of MPLAB X IDE (available as a free download at <https://www.microchip.com/mplabx/>).
- Indicates debugger status via the indicator light strip.
- Operates within a temperature range of 0-70 degrees Celsius.

Performance/Speed:

- More and faster memory.
- A Real-Time Operating System (RTOS).
- No firmware download delays incurred when switching devices.
- A 32-bit MCU running at 300 MHz.

Safety:

- Receive feedback from debugger when external power supply is needed for target.
- Supports target supply voltages from 1.2 to 2.8V for low voltage program mode entry and 2.8 to 5.0V for low voltage and high voltage program mode entry.
- Safely power up to 1A with an optional 9V DC power supply.
- Protection circuitries are added to the probe drivers to guard from power surges from the target.
- V_{DD} and V_{PP} voltage monitors protect against overvoltage conditions/all lines have over-current protection.
- Programming/debugging pins with a programmable range of resistor values, plus direction (pull-up, pull-down, or nonexistent).
- Controlled programming speed provides flexibility to overcome target board design issues.
- CE and RoHS compliant – conforms to industry standards.

2.3 MPLAB PICKit 4 In-Circuit Debugger Components

The components of the MPLAB PICKit 4 In-Circuit Debugger system are:

- A rectangular-shaped MPLAB PICKit 4 unit housed in a durable, black plastic case with a brushed metal top which is accented with an indicator light strip, button area.
- A Micro-B USB connector.
- MicroSD card slot.
- Emergency recovery button.
- Lanyard connector.
- A Micro-B USB cable to provide communications between the debugger and a computer, as well as providing power to the debugger.

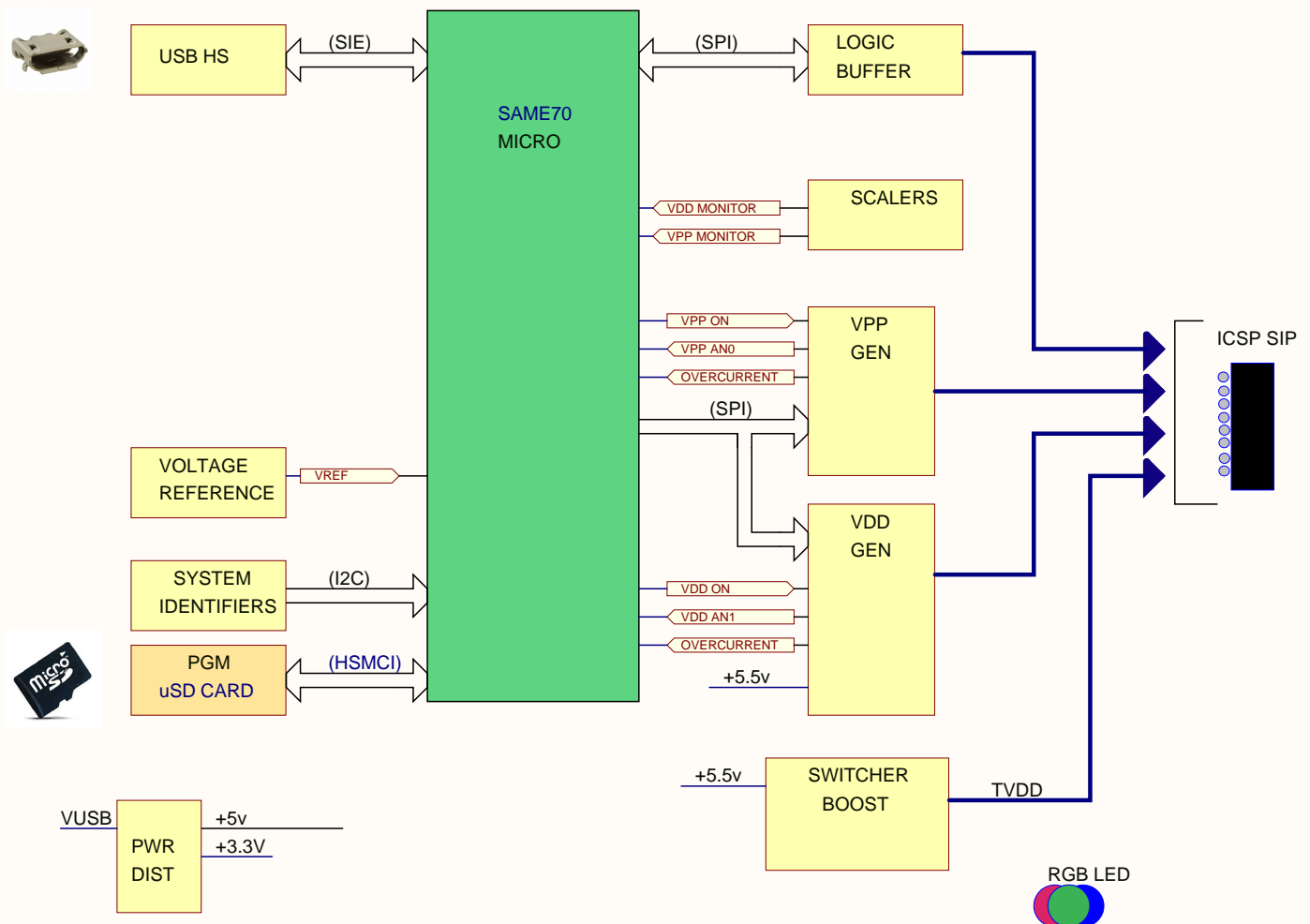
Figure 2-1. Basic Debugger System



Additional hardware and accessories may be ordered separately from Microchip Direct (<https://www.microchipdirect.com>).

- (Part Number AC002015) - a connectivity board that supports JTAG, SWD and ICSP protocols, useful for debugging legacy AVR® with MPLAB PICKit 4 (<https://www.microchipdirect.com/product/search/all/AC102015>).
- Transition sockets.
- ICD headers.
- MPLAB processor extension paks.

2.4 MPLAB PICKit 4 Block Diagram



3. Operation

A simplified theory of operation of the MPLAB PICKit 4 In-Circuit Debugger system is provided here. It is intended to provide enough information so that a target board can be designed that is compatible with the debugger for both debugging and programming operations. The basic theory of in-circuit debugging and programming is discussed so that problems, if encountered, are quickly resolved.

3.1 Debugger to Target Communication



Important: The MPLAB X IDE software must be installed prior to connecting the MPLAB PICKit 4 In-Circuit Debugger.

The debugger is connected to the computer via a USB cable for communication and debugger power.

The debugger is connected to the target application for communication and data collection and optional debugger power.

The debugger system configurations are discussed in the following sections.

	<p style="text-align: center;">CAUTION</p> <p>Communication Failure. Do not connect the hardware before installing the software and USB drivers.</p>
	<p style="text-align: center;">CAUTION</p> <p>Debugger or Target Damage. Do not change hardware connections while the debugger or target is powered.</p>

Note: The MPLAB PICKit 4 In-Circuit Debugger is warranted for operation using the provided cable. Cables from other vendors may result in communication errors.

3.1.1 Standard ICSP™ Device Communication

The debugger system can be configured to use standard ICSP communication connection for both programming and debugging functions.

Make sure to align the Pin 1 on the debugger to Pin 1 on the target. The programming connector can be inserted into either:

- A matching connector at the target, where the target device is on the target board ([Figure 3-1](#)).
- A standard adapter/header board combo (available as a Processor Extension Pak), which is then plugged into the target board ([Figure 3-2](#)).

For more on standard communication, see [Standard Communication](#).

Figure 3-1. Standard Debugger System – Device With On-board ICE Circuitry

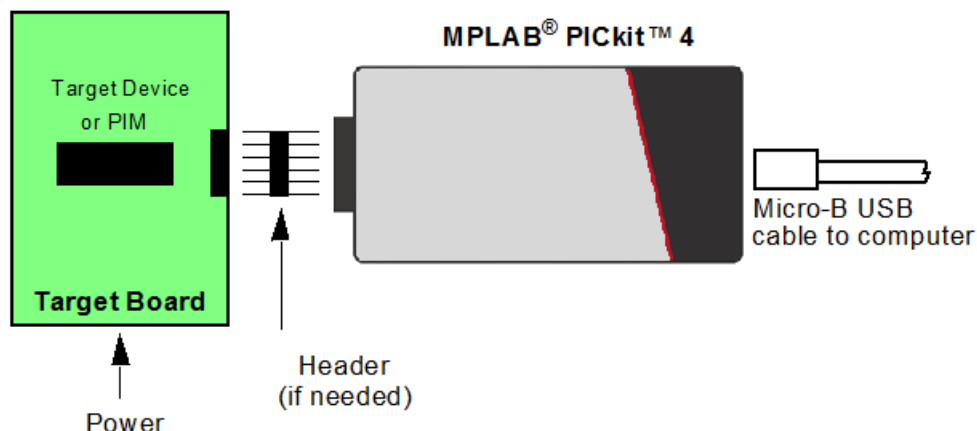
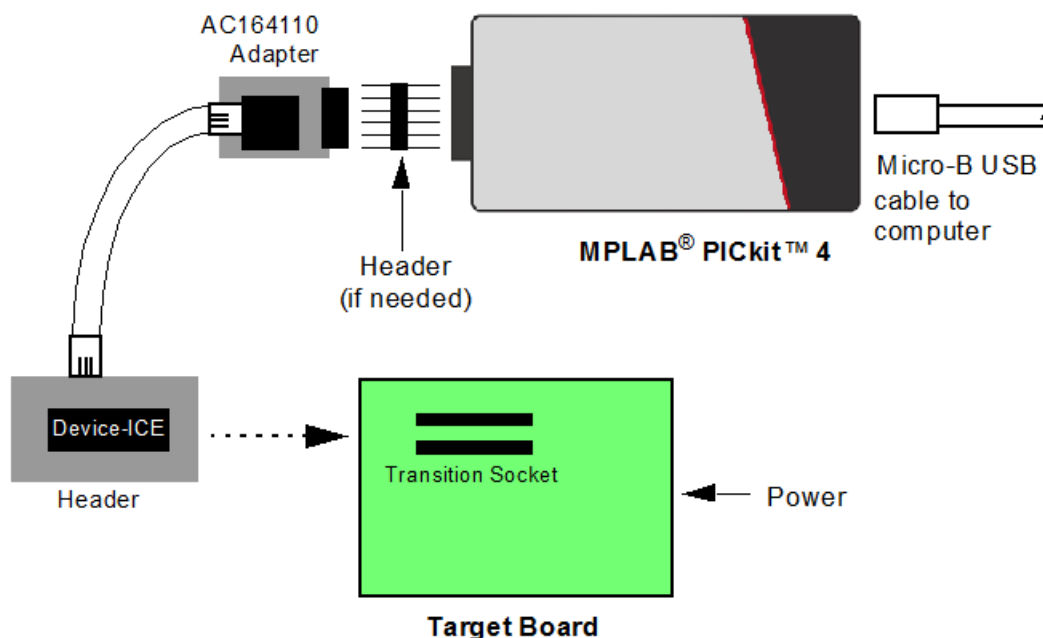


Figure 3-2. Standard Debugger System – ICE Device



3.2 Target Communication Connections



Important: Refer to the data sheet for the device you are using as well as the application notes and the specific interface for additional information and diagrams.

3.2.1 Standard Communication Target Connection

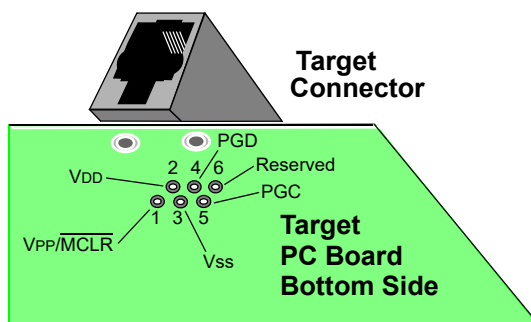
USING SINGLE IN-LINE CONNECTOR

Use the single in-line connector between the MPLAB PICKit 4 In-Circuit Debugger and the target board connector (see [Figure 3-1](#) and [Standard Communication](#)).

USING AN ADAPTER

Use the AC164110 adapter between the MPLAB PICKit 4 In-Circuit Debugger and the target device with the modular interface (six conductor) cable. The pin numbering for the connector is shown from the bottom of the target PCB in Figure 3-3.

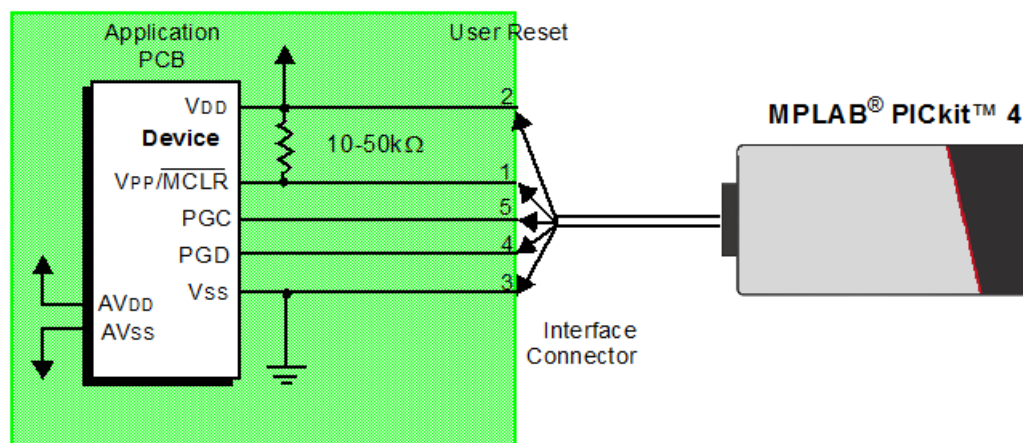
Figure 3-3. Standard RJ-11 Connection at Target



3.2.2 Target Connection Circuitry

Figure 3-4 below shows the interconnections of the MPLAB PICKit 4 In-Circuit Debugger to the connector on the target board. The diagram also shows the wiring from the connector to a device on the target PCB. A pull-up resistor (usually around 10-50 kΩ) is recommended to be connected from the V_{PP}/MCLR line to V_{DD} so that the line may be strobed low to reset the device.

Figure 3-4. Standard Connection to Target Circuitry

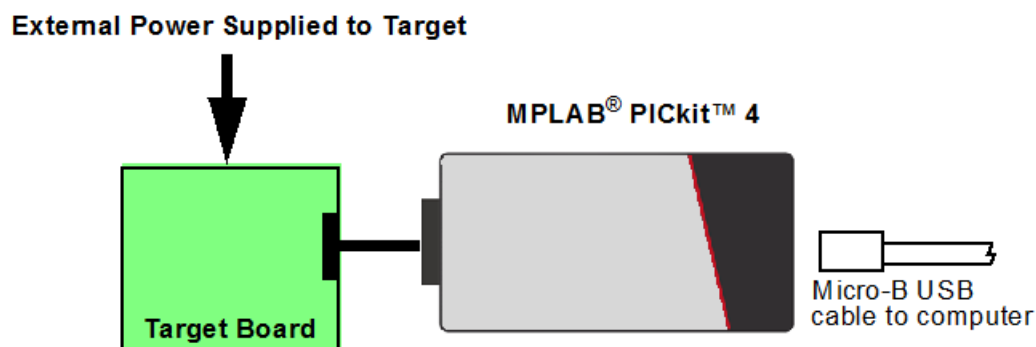


3.2.3 Target Powered

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

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

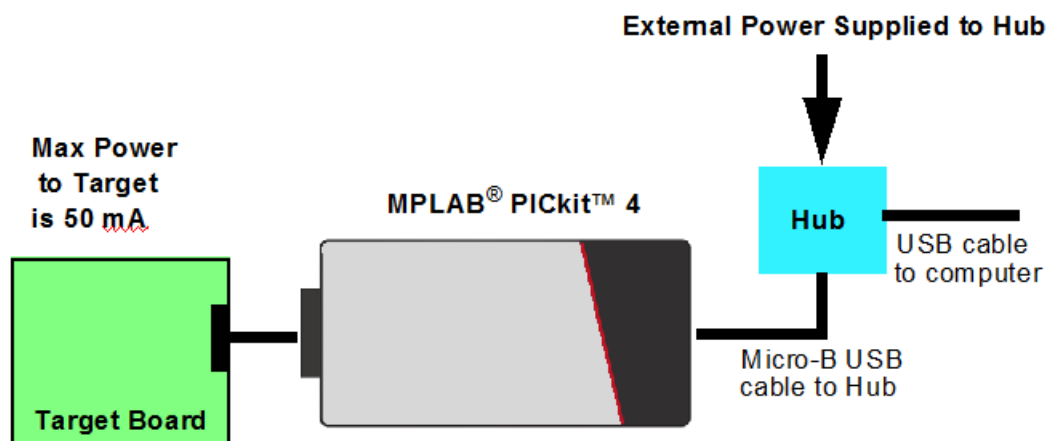
Figure 3-5. Target Powered from External Source



3.2.4 Debugger Powered

If the target is powered through the debugger with an externally powered hub as shown below, the power available to the target is limited to 50 mA.

Figure 3-6. Target Powered Through Self-Powered Hub



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

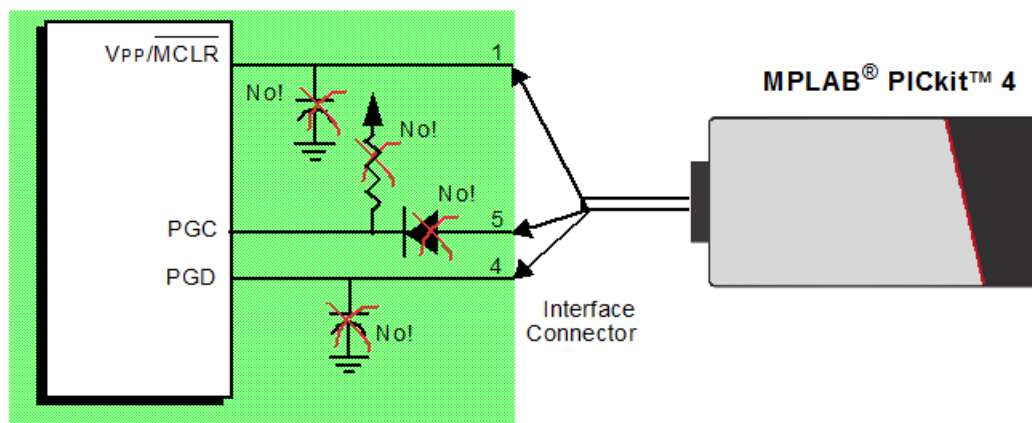
Also, devices with a V_{CAP} line (PIC18FXXJ for example) should be connected to the appropriate capacitor or level.

Note: The interconnection is very simple. Any problems experienced are often caused by other connections or components on these critical lines that interfere with the operation of the MPLAB PICKit 4 In-Circuit Debugger, as discussed in the following section.

3.2.5 Circuits That Will Prevent the Debugger From Functioning

The figure below shows the active debugger lines with some components that will prevent the MPLAB PICKit 4 In-Circuit Debugger system from functioning.

Figure 3-7. Improper Circuit Components



Specifically, these guidelines must be followed:

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

3.3 Debugging

There are two steps to using the MPLAB PICKit 4 In-Circuit Debugger system as a debugger. The first requires that an application is programmed into the target device (usually with the MPLAB PICKit 4 itself). The second uses the internal in-circuit debug hardware of the target Flash device to run and test the application program. These two steps are directly related to the MPLAB X IDE operations:

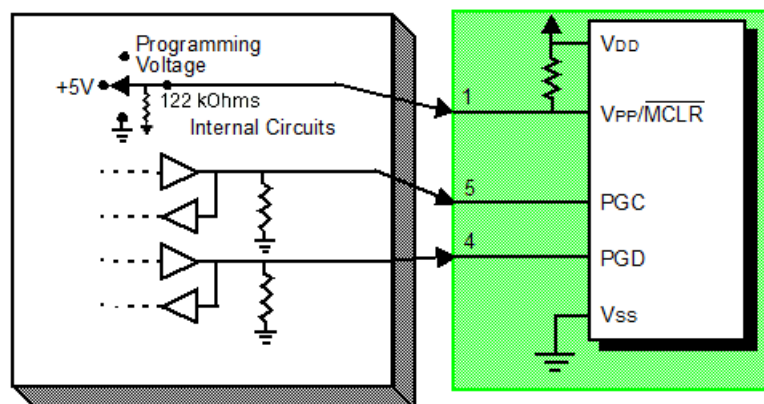
1. Programming the code into the target and activating special debug functions.
2. Debugging the code using features such as breakpoints.

Note: For more information, refer to the MPLAB X IDE online Help.

If the target device cannot be programmed correctly, the MPLAB PICKit 4 In-Circuit Debugger will not be able to debug.

A simplified diagram of some of the internal interface circuitry of the MPLAB PICKit 4 In-Circuit Debugger is shown in the figure below.

Figure 3-8. Proper Connections for Programming



For programming, no clock is needed on the target device, but power must be supplied. When programming, the debugger puts programming levels on V_{PP}/\overline{MCLR} , sends clock pulses on PGC, and serial data via PGD. To verify that the part has been programmed correctly, clocks are sent to PGC and data is read back from PGD. This sequence confirms the debugger and device are communicating correctly.

3.4 Requirements for Debugging

To debug (set breakpoints, see registers, etc.) with the MPLAB PICKit 4 In-Circuit Debugger system, there are critical elements that must be working correctly:

- The debugger must be connected to a computer. It must be powered by the computer via the USB cable and it must be communicating with the MPLAB X IDE software via the Micro-B USB cable. Refer to the MPLAB X IDE Help file titled “Getting Started with MPLAB X IDE,” and navigate through the “Tutorial” to the “Running and Debugging Code” section.
- The debugger must be connected (as shown in the figure in [3.3 Debugging](#)) to the V_{PP} , PGC and PGD pins of the target device with the modular interface cable (or equivalent).
- The target device must have power and a functional, running oscillator. If for any reason, the target device does not run, the MPLAB PICKit 4 In-Circuit Debugger will not be able to debug.
- The target device must have its Configuration words programmed correctly. These are set using the MPLAB X IDE.
 - The oscillator Configuration bits should correspond to RC, XT, etc., depending on the target design.
 - For some devices, the Watchdog Timer is enabled by default and needs to be disabled.
 - The target device must not have code protection enabled.
 - The target device must not have table read protection enabled.
 - For some devices with more than one PGC/PGD pair, the correct pair needs to be selected in the device's configuration word settings. This only refers to debugging, since programming will work through any PGC/PGD pair.

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

- [3.4.1 Sequence of Operations Leading to Debugging](#)
- [3.4.2 Debugging Details](#)

3.4.1 Sequence of Operations Leading to Debugging

Given that the [3.4 Requirements for Debugging](#) are met, set the MPLAB PICKit 4 In-Circuit Debugger as the current tool in MPLAB X IDE. Go to *File > Project Properties* to open the dialog, then under “Hardware Tool,” click **PICKit 4**. The following actions can now be performed.

- When *Debug > Debug Main Project* is selected, the application code is programmed into the device's memory via the ICSP protocol as described at the beginning of this section.
- A small “debug executive” program is loaded into the high area of program memory of the target device. Since the debug executive must reside in program memory, the application program must not use this reserved space.

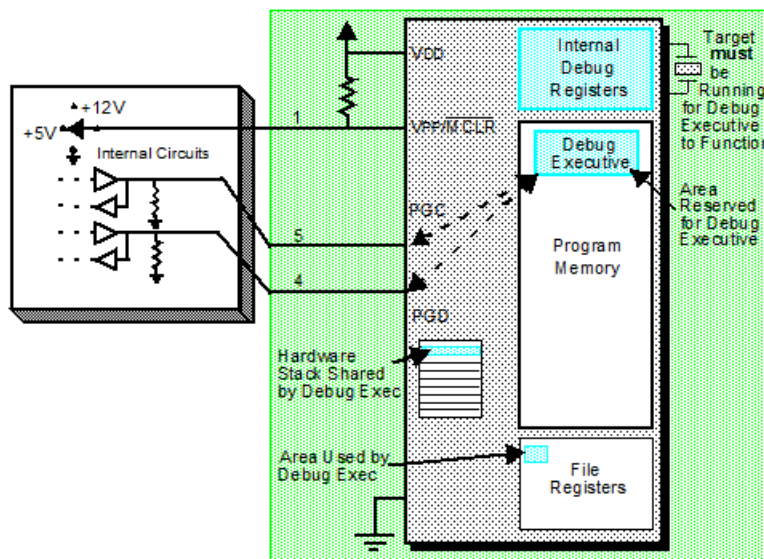
Some devices have special memory areas dedicated to the debug executive. Check your device data sheet for details.

- Special “in-circuit debug” registers in the target device are enabled by MPLAB X IDE. These allow the debug executive to be activated by the debugger. For more information on the device’s reserved resources, see [3.6 Resources Used by the Debugger](#).
- The target device is run in Debug mode.

3.4.2 Debugging Details

The figure below illustrates the MPLAB PICKit 4 In-Circuit Debugger system when it is ready to begin debugging.

Figure 3-9. MPLAB® PICKit™ 4 In-Circuit Debugger Ready to Begin Debugging



To find out whether an application program will run correctly, a breakpoint is typically set early in the program code. When a breakpoint is set from the user interface of MPLAB X IDE, the address of the breakpoint is stored in the special internal debug registers of the target device. Commands on PGC and PGD communicate directly to these registers to set the breakpoint address.

Next, the *Debug > Debug Main Project* function is usually selected in MPLAB X IDE. The debugger tells the debug executive to run. The target starts from the Reset vector and executes until the Program Counter reaches the breakpoint address that was stored previously in the internal debug registers.

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

The debug executive runs like an application in program memory. It uses some locations on the stack for its temporary variables. If the device does not run, for whatever reason (no oscillator, faulty power supply connection, shorts on the target board, etc.), then the debug executive cannot communicate to the MPLAB PICKit 4 In-Circuit Debugger, and MPLAB X IDE will issue an error message.

Another way to set a breakpoint is to select *Debug > Pause*. This toggles the PGC and PGD lines so that the in-circuit debug mechanism of the target device switches the Program Counter from the user’s code in program memory to the debug executive. Again, the target application program is effectively halted, and MPLAB X IDE uses the debugger communications with the debug executive to interrogate the state of the target device.

3.5 Programming

Note: For information on programming, refer to the MPLAB X IDE online Help.

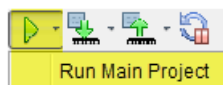


Notice: Headers are not supported at this time.

In MPLAB X IDE, use the MPLAB PICKit 4 as a programmer to program a non-ICE/-ICD device, such as a device not on a header board. Set the MPLAB PICKit 4 In-Circuit Debugger as the current tool (click the Debug Tool PICKit 4 in the navigation window, select *File > Project Properties*, then under “Hardware Tool,” click **PICKit 4**) to perform these actions:

- When Run Main Project icon (see below) is selected, the application code is programmed into the device's memory via the ICSP protocol. No clock is required while programming and all modes of the processor can be programmed – including code protect, Watchdog Timer enabled, and table read protect.

Figure 3-10. Run Main Project Icon



- A small “program executive” program may be loaded into the high area of program memory for some target devices.
- Special “in-circuit debug” registers in the target device are disabled by MPLAB X IDE, along with all debug features. This means that a breakpoint cannot be set and register contents cannot be seen or altered.
- The target device is run in Release mode. As a programmer, the debugger can only toggle the $\overline{\text{MCLR}}$ line to Reset and start the target device.

The MPLAB PICKit 4 In-Circuit Debugger system programs the target using ICSP. V_{PP} , PGC and PGD lines should be connected as described previously. No clock is required while programming and all modes of the processor can be programmed, including code protection, Watchdog Timer and table read protection.

3.6 Resources Used by the Debugger

For a complete list of resources used by the debugger for your device, see the online Help file in MPLAB X IDE for the MPLAB PICKit 4 In-Circuit Debugger. From the MPLAB X IDE “Learn & Discover” page, click **Users Guide & Release Notes** and then click the link for the “Reserved Resources for MPLAB PICKit 4.”

4. Debugger Usage

How to install and setup the MPLAB PICKit 4 In-Circuit Debugger system is discussed in this section. For instructions on using MPLAB X IDE with the debugger, refer to the online Help accessible from the MPLAB X IDE main menu bar [Help > Tool Help Contents > MPLAB X IDE Help](#).

4.1 Installation and Setup

In MPLAB X IDE, refer to the online Help file “Getting Started with MPLAB X IDE” for details on installing the IDE and setting up the debugger to work with it.

In summary:

1. Install MPLAB X IDE.



Tip: Tutorial topics are available in the MPLAB X IDE online Help that is accessible from the main menu bar [Help > Tool Help Contents > MPLAB X IDE Help > Tutorial](#).

2. Connect the MPLAB PICKit 4 to the computer and allow the default USB drivers to install. For more information on target connections, see [Operation](#).



Important: The debugger can power a target board only up to 50 mA.

3. Select which language toolsuite/compiler you want to use for development and install it on your computer.
4. Launch MPLAB X IDE and open the online Help ([Help > Tool Help Contents > MPLAB X IDE Help](#)) for detailed instructions on creating and setting up a new project and running and debugging code.

Items of note:

1. Each debugger contains a unique identifier which, when first installed, will be recognized by the operating system, regardless of the computer USB port used.
2. MPLAB X IDE operation connects to the hardware tool at run time (Run or Debug Run). To always be connected to the hardware tool, go to [Tools > Options](#), **Embedded** button, **Generic Settings** tab, and check the “Maintain active connection to hardware tool” check box.
3. Configuration bits can only be viewed in the Configuration Bits window. To set them in code, select [Window > Target Memory Views](#). Then select “Configuration Bits” from the Memory drop list and select “Read/Write” from the Format drop list to enable access to the settings.

4.2 Debug Tutorial

Refer to the MPLAB X IDE Help file titled “Getting Started with MPLAB X IDE,” and navigate through the “Tutorial” to the “Running and Debugging Code” section.



4.3 Quick Debug/Program Reference

The following table is a quick reference for using the MPLAB PICKit 4 In-Circuit Debugger as either a debugging or programming tool.



Notice: For header support, see the Release Notes for MPLAB PICKit 4 in MPLAB X IDE v5.25 or greater.

Table 4-1. Debug vs. Program Operation

Item	Debug	Program
Needed Hardware	A computer and target application (Microchip demo board or your own design).	
	Debugger, USB cable, and power supply (if needed).	
	Device with on-board debug circuitry or debug header with special -ICE device.	Device (with or without on-board debug circuitry).
MPLAB X IDE selection	Project Properties, ICD 4 as Hardware Tool.	
	Debug Main Project icon 	Make and Program Device icon 
Program Operation	Programs application code into the device. Depending on the selections on the Project Properties dialog, this can be any range of program memory. In addition, a small debug executive is placed in program memory and other debug resources are reserved.	Programs application code into the device. Depending on the selections on the Project Properties dialog, this can be any range of program memory.
Debug Features Available	All for device – breakpoints, etc.	N/A
Serial Quick-Time Programming (SQTP)	N/A	Use the MPLAB IPE to generate the SQTP file.
Command-line Operation	Use MDB command line utility, found by default in: C:\Program Files (x86)\Microchip\MPLABX\vx.xx\mplab_platform\bin\mdb.bat	Use IPECMD, found by default in: C:\Program Files (x86)\Microchip\MPLABX\<vx.xx\mplab_platform\mplab_ipe\ipecmd.exe.
Programmer-To-Go	N/A	Programs application code stored on a microSDHC card inserted in the MPLAB PICKit 4 into the device.

4.4 Debugger Limitations

For a complete list of debugger limitations for your device, see the online Help file in MPLAB X IDE ([Help > Tool Help Contents > Hardware Tool Reference Help > Limitations - Emulators and Debuggers](#)).

4.5 Common Debug Features

Refer to the online Help file “Getting Started with MPLAB X IDE,” Running an Debugging Code section, for details on debug features. This sections includes:

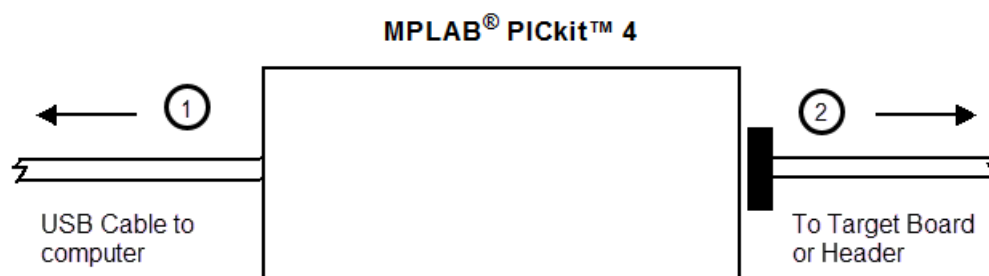
1. Debug Running the project (build, program and run) from [Debug > Debug Main Project](#).
2. Using breakpoints.
3. Stepping through code.
4. Using the Watches window.
5. Viewing Memory, Variables and the Call Stack.

6. Using the Call Graph.

4.6 Connecting the Target Board

1. Connect the Micro-B USB cable between the debugger and the computer, if not already connected.
2. Connect the appropriate cable(s) between the debugger to the target.
3. Connect power to target if needed.

Figure 4-1. Insert Communications and USB Cables



See [Debugger To Target Communication](#) for more details and a diagram.

4.7 Setting Up the Target Board

The target must be set up for the type of target device to be used.

4.7.1 Using Production Devices

For production devices, the debugger may be connected directly to the target board. The device on the target board must have built-in debug circuitry in order to debug with the MPLAB PICKit 4 In-Circuit Debugger.

The target board must have a connector to accommodate the communications chosen for the debugger. For connection information, see [Debugger to Target Communication](#).

4.7.2 Using ICE Devices



Notice: For header support, see the latest Release Notes for MPLAB PICKit 4 in MPLAB X IDE v5.25 or greater.

For ICE devices, an ICE header board is required. The header board contains the hardware that is necessary to emulate a specific device or family of devices. For more information on ICE headers, see the "Processor Extension Pak and Header Specification" (DS50001292).

A transition socket is used with the ICE header to connect the header to the target board. Transition sockets are available in various styles to allow a common header to be connected to one of the supported surface mount package styles. For more information on transition sockets, see the "Transition Socket Specification" (DS50001194).

Header board layout will be different for headers or processor extension paks. For connection information, see [Debugger to Target Communication](#).

4.7.3 Using an ICD Header



Notice: For header support, see the latest Release Notes for MPLAB PICKit 4 in MPLAB X IDE v5.25 or greater.

All Baseline and some Mid-Range PIC microcontrollers require a special –ICD device mounted on a debug header circuit board to enable the debugging feature. For a list of these devices and the required ICD header board part number, see the “*Processor Extension Pak and Header Specification*” (DS50001292).

Each ICD header board comes with the necessary – ICD device and is used on the target board instead of the production microcontroller. However, most header boards have an RJ-11 debug connector which requires the AC164110 RJ-11 to ICSP™ adapter kit to connect it to MPLAB PICKit 4.

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

4.7.4 Powering the Target

These are configuration essentials:

- When using the USB connection, MPLAB PICKit 4 can be powered from the computer but it can only provide a limited amount of current, up to 50 mA, at V_{DD} from 1.2-5V to a small target board.
- The desired method is for the target to provide V_{DD} since it can provide a higher current. The additional benefit is that plug-and-play target detection facility is inherited, for instance, MPLAB X IDE will let you know in the Output window when it has detected the target and has detected the device.

If you have not already done so, connect the MPLAB PICKit 4 to the target board using the appropriate cables (see [4.6 Connecting the Target Board](#)). Then power the target.

4.8 Setting Up MPLAB X IDE

Once the hardware is connected and powered, MPLAB X IDE may be set up for use with the MPLAB PICKit 4 In-Circuit Debugger.






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



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

4.9 Starting and Stopping Debugging

Note: Refer to the MPLAB X IDE online Help for information on menu option icons.

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

- To run your code, select either Debug > Debug Main Project or  from the Run toolbar.
- To halt your code, select either Debug > Pause or  from the Debug toolbar.
- To run your code again, select either Debug > Continue or  from the Debug toolbar.
- To step through your code, select either Debug > Step Into or  from the Debug toolbar. Be careful not to step into a Sleep instruction or you will have to perform a processor Reset to resume debugging.
- To step over a line of code, select either Debug > Step Over or  from the Debug toolbar.

- To end code execution, select either Debug > Finish Debugger Session or  from the Debug toolbar.
- To perform a processor Reset on your code, select either Debug > Reset or  from the Debug toolbar.

Depending on the device, additional Resets, such as POR/BOR, MCLR, and System, may be available. Refer to the product data sheet for more information.

4.10 Viewing Processor Memory and Files

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

- Window > Target Memory Views - view data (Data Memory) and code (Execution Memory) in device memory. Other memory can also be viewed as defined by the device including Peripherals, Configuration Bits, CPU Registers, External EBI Memory, External SQI Memory, User ID Memory, etc.
- Window > Debugging - view debug information. Select from Variables, Watches, Call Stack, Breakpoints, Stopwatch, and many others.

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

4.11 Breakpoints and Stopwatch

Use breakpoints to halt code execution at specific lines in your code. Use the stopwatch with breakpoints to time code execution.

4.11.1 Breakpoint Resources

In 16-bit devices - breakpoints, data captures, and run-time watches use the same resources. Therefore, the available number of breakpoints is actually the available number of combined breakpoints/triggers.

In 32-bit devices - breakpoints use different resources than data captures and run-time watches. Therefore, the available number of breakpoints is independent of the available number of triggers.

The number of hardware and software breakpoints available and/or used is displayed in the Dashboard window (Window > Dashboard). See the MPLAB X IDE online Help file for more on this feature. *Not all devices have software breakpoints.*

For limitations on breakpoint operation, including the general number of hardware breakpoints per device, and hardware breakpoint skidding amounts, see the online Help file in MPLAB X IDE for the debugger limitations (Help > Help Contents > Hardware Tool Reference > Limitations - Emulators and Debuggers).

4.11.2 Hardware or Software Breakpoint Selection

To select hardware or software breakpoints:

1. Select your project in the Projects window. Then select File > Project Properties or right click and select **Properties**.
2. In the Project Properties dialog under "Categories," select **PICKit 4**.
3. Under "Option Categories," select **Debug Options**.
4. Select **Use software breakpoints** to use software breakpoints. Clear the selection to use hardware breakpoints.



Tip: Using software breakpoints for debugging impacts device endurance. Therefore, it is recommended that devices used in this manner are not be used as production parts.

To help you decide which type of breakpoints to use (hardware or software), the following table compares the features of each.

Table 4-2. Hardware vs. Software Breakpoints

Feature	Hardware Breakpoints	Software Breakpoints
Number of breakpoints	Limited	Unlimited
Breakpoints written to*	Internal Debug Registers	Flash Program Memory
Breakpoints applied to**	Program Memory/Data Memory	Program Memory only
Time to set breakpoints	Minimal	Dependent on oscillator speed, time to program Flash Memory, and page size
Breakpoint skidding	Most devices. See the online Help, Limitations section, for details.	No
* Where information about the breakpoint is written in the device.		
** What kind of device feature applies to the breakpoint. This is where the breakpoint is set.		

4.11.3 Breakpoint and Stopwatch Usage

Breakpoints halt execution of code. To determine the time between the breakpoints, use the stopwatch.

Refer to the MPLAB X IDE online Help for instructions on how to set up and use breakpoints and the stopwatch.

5. MPLAB PICKit 4 Programmer-To-Go

The MPLAB PICKit 4 Programmer-To-Go (PTG) functionality allows a device memory image to be downloaded into a microSDHC card inserted in the MPLAB PICKit 4 tool for later programming into a specific device. That image contains the programming algorithm information. No software or PC is required to program devices once the MPLAB PICKit 4 programmer is set up for Programming-To-Go. You can use the MPLAB X IDE (see [5.3.1 Setting Up PTG Mode Using MPLAB X IDE](#)) or the MPLAB IPE (see [5.3.2 Setting Up PTG Mode with MPLAB IPE](#)) to set up the MPLAB PICKit 4 for Programmer-To-Go mode.

Note:

No debugging capabilities are available in Programmer-To-Go mode.

Figure 5-1. MPLAB® PICKit™ 4 In-Circuit Debugger Diagram



Prerequisites for Programmer-To-Go

- MPLAB X IDE or MPLAB IPE (v5.25 or greater) must be installed on your computer.
- MicroSDHC Card - a formatted FAT32-compatible microSDHC card must be inserted correctly in the MPLAB PICKit 4 tool in order to use the Programmer-To-Go feature.
- Power source - for setting up the MPLAB PICKit 4 for Programmer-To-Go mode and for programming devices remotely. See the following section for specific power requirements.

5.1 Power Requirements for Programmer-To-Go

When Connected to Computer

When the MPLAB PICKit 4 is connected to the computer and in the MPLAB X IDE or MPLAB IPE application, use the supplied USB cable between the computer and the Micro-B USB connector located on the top of the tool.

When Programming a Device Using PTG

When the MPLAB PICKit 4 is connected to the remote target board for programming a device using Programmer-To-Go, the minimum power required from the target board to the MPLAB PICKit 4 is 350 mA.

If sufficient power cannot be supplied from the target board, then MPLAB PICKit 4 must be powered by a 5V power supply through the Micro-B USB connector on the top of the MPLAB PICKit 4 tool. There are several options for providing power, such as, using:

- Any available PC USB port or USB hub port. (No USB communication is necessary; it is only used to provide power.)
- A USB host port on a portable device.
- A USB power adapter or charger with a USB Micro-B connector, either from an automotive power jack or an AC wall plug.
- A portable battery charge or power source for cell phones or other portable devices with USB Micro-B connector.
- A custom battery pack that supplies regulated 5V into the MPLAB PICKit 4 USB Micro-B connector.

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

- Is able to supply at least 350 mA of current to the MPLAB PICKit 4 tool.
- Provides a steady, regulated 4.5V to 5.5V output.

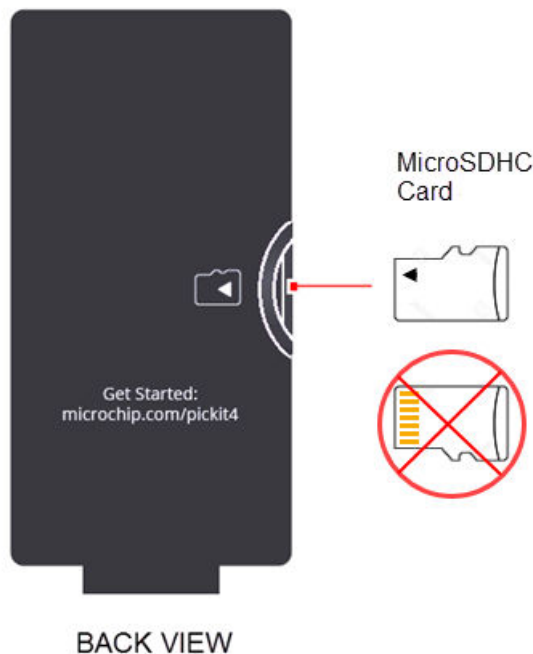
Note:

1. Most portable chargers/power devices with their own batteries will not give an indication when their internal battery voltage gets low and the output drops below 4.5V. Therefore, you must be sure the device's battery has sufficient remaining capacity to power the MPLAB PICKit 4 above 4.5V.
2. Any battery-based power sources should be disconnected from the MPLAB PICKit 4 unit when it is not in use. Otherwise, the MPLAB PICKit 4 unit will drain the power source battery.

5.2 Limitations for Programmer-To-Go

1. You must have a formatted FAT32-compatible microSDHC card **inserted** correctly into the MPLAB PICKit 4 in order to use the Programmer-To-Go feature.

Figure 5-2. Inserting the MicroSDHC Card Correctly



2. Devices Supported: Check the MPLAB PICKit 4 Release Notes (Readme) for device support information.
3. In Programmer-To-Go mode, if the target board is powering the PICKit 4 it must be capable of providing 350 mA of power for the PICKit 4 to operate properly. If the target board cannot provide enough power, you will need to provide power directly to the PICKit 4 through its USB port with either a power supply, computer, or USB power bank. See 5.1 Power Requirements for Programmer-To-Go for suggested power sources.
4. If high voltage (HV) programming is not required, it is recommended to use low voltage programming (LVP) if the device supports it.
5. When using PTG in HV programming mode and target power, a 100-ohm resistor is required in series with NMCLR if the PICKit 4 will be powered from the target. Or, follow the procedure described in ETN-37 MPLAB PICKit 4 VPP Overshoot Modification document found on the MPLAB PICKit 4 product web page (<https://www.microchip.com/Developmenttools/ProductDetails/PG164140>).
6. Target Voltage is limited to 5.0V maximum.

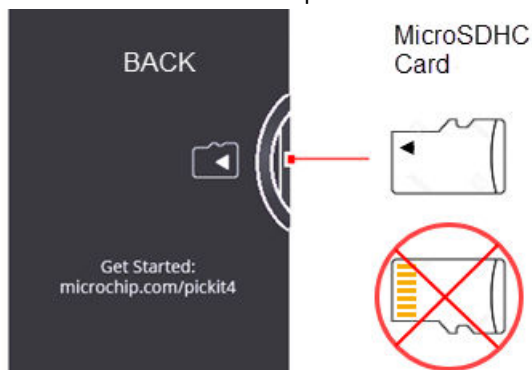
5.3 Setting up PICKit 4 for Programmer-To-Go Mode

Before downloading a memory image to the MPLAB PICKit 4 for PTG operation, the PICKit 4 programmer options should be set up for Programmer-To-Go operation. In fact, it is highly recommended to test programming a target device from the software first, with all desired options, to ensure the device programs as expected before downloading an image to Programmer-To-Go. Then you can put the PICKit 4 into PTG mode. Refer to the following sections for instructions.

5.3.1 Setting Up PTG Mode Using MPLAB X IDE

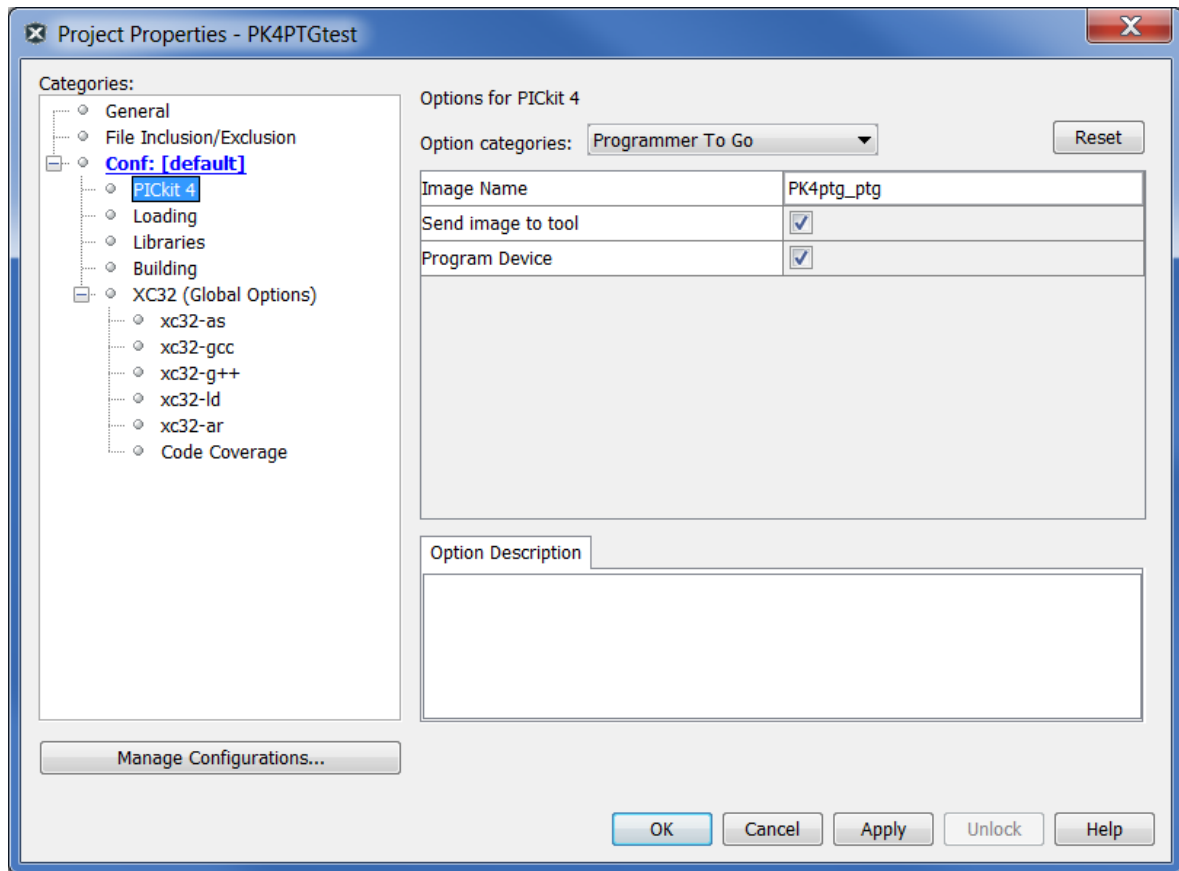
Using MPLAB X IDE, follow these steps to download the project files into the microSDHC card in the MPLAB PICKit 4 and enter Programmer-To-Go mode.

1. Insert a formatted FAT32-compatible microSDHC card into the PICKit 4.



2. Ensure that you have the appropriate connections to the device for Programmer-To-Go:
 - PICKit 4 is connected to the computer via the USB cable.
 - PICKit 4 is connected via the appropriate programming interface connector to the target board.
 - The target board is powered from either the PICKit 4 or a power supply, depending on your Project Properties selection.
3. In MPLAB X IDE, open the project you want to use and select the PICKit 4 tool for programming.
4. Right-click on your project name to open the Project Properties dialog. Then click on **PICKit 4** under *Categories* to display the *Options for PICKit 4* on the right side of the display. Select the **Programmer-To-Go** option category.

Figure 5-3. MPLAB X IDE Programmer-To-Go Options



5. In the *Image Name* field, the default is "<your project name>_ptg," though you can edit the name, if you wish. This will be the folder name on the microSDHC card that contains the appropriate files for Programmer-To-Go.
6. In the *Send image to tool*, the check box is selected by default. With the box checked, the PTG image is created and then sent to the microSDHC card in the connected MPLAB PICKit 4.
7. The *Program Device* check box is selected by default. When the check box selected, the device connected to the MPLAB PICKit 4 is programmed.
Note: If both the *Send image to tool* and *Program Device* check boxes are unchecked, see [5.3.3 Setting Up PTG Mode Without a Memory Card](#).
8. Click **Apply**, then **OK**. Use the Make and Program Device Main Project icon on the toolbar and select **Programmer-To-Go PICKit3/PICKit4 Main Project**.

Figure 5-4. MPLAB X IDE - Download Image to the MicroSDHC Card



During this process, the device is programmed, then the microSDHC card is populated with the appropriate files for the Programmer-To-Go operation. The Output window displays a status message "Programming/Verify complete" when the process finishes successfully.

Note: The PTG settings on the microSDHC card are the same as in the project (for example, memory, power, etc.).

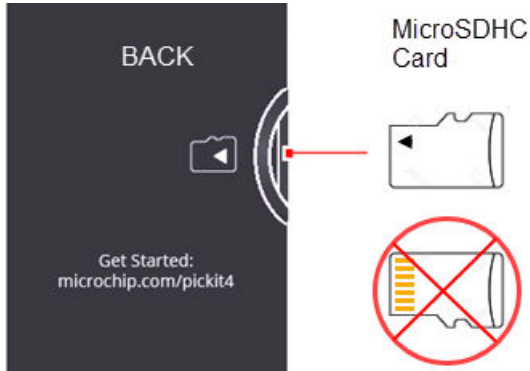
The PICKit 4 is now in Programmer-To-Go mode. The LED should blink green to indicate the tool has been configured successfully for Programmer-To-Go.

9. Disconnect the PICKit 4 and you're ready to use Programmer-To-Go.

5.3.2 Setting Up PTG Mode with MPLAB IPE

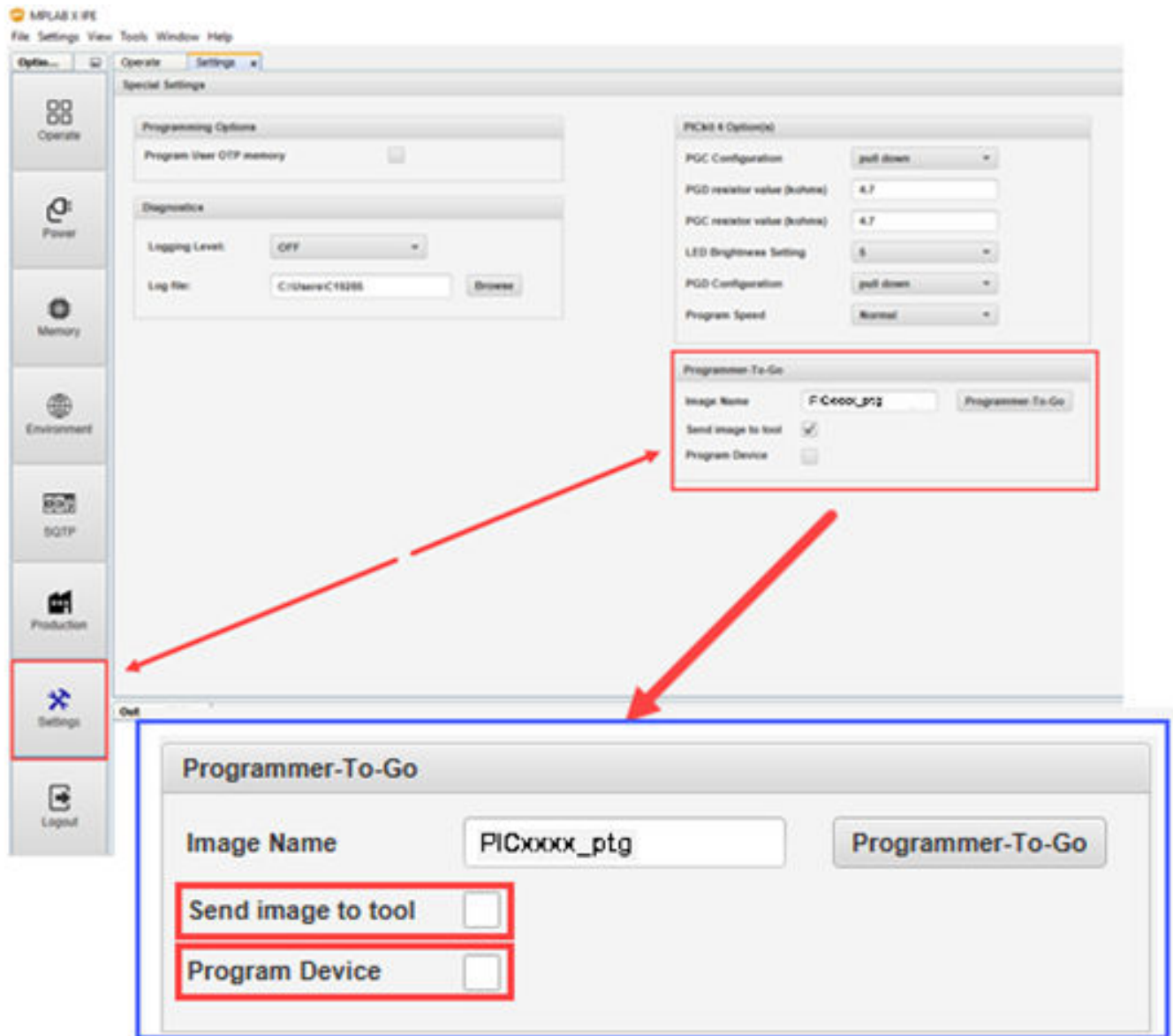
Using MPLAB IPE, follow these steps to download the project files into the microSDHC card in the MPLAB PICKit 4 and enter Programmer-To-Go mode.

1. Insert a formatted FAT32-compatible microSDHC card into the PICKit. 4.



2. Ensure that you have the appropriate connections to the device for Programmer-To-Go:
 - PICKit 4 is connected to the computer via the USB cable.
 - PICKit 4 is connected via the appropriate programming interface connector to the target board.
 - The target board is powered from either the PICKit 4 or a power supply, depending on your Project Properties selection.
3. From the MPLAB IPE menu, select **Settings, Advance Mode** and type in the password to log in. Click the **Settings** icon on the left to open the Special Setting dialog and locate the Programmer-To-Go settings.

Figure 5-5. MPLAB IPE Programmer-To-Go Options



4. In the *Image Name* field, the default is "<your project name>_ptg," though you can edit the name if you wish. This will be the folder name on the microSDHC card that contains the appropriate files for Programmer-To-Go.
5. In the *Send image to tool*, the check box is selected by default. With the check box selected, the PTG image is created and then sent to the microSDHC card in the connected MPLAB PICKit 4.
6. The *Program Device* the check box is selected by default. With the box checked, the device connected to the MPLAB PICKit 4 is programmed.
Note: If both the *Send image to tool* and *Program Device* check boxes are unchecked, see [5.3.3 Setting Up PTG Mode Without a Memory Card](#).
7. Click the **Programmer-To-Go** button.
 During this process, the device is programmed, then the Programmer-To-Go directory is populated with the appropriate files for the Programmer-To-Go operation into the microSDHC card. The Output window displays a status message "Programming/Verify complete" when the process finishes successfully.
Note: The PTG settings on the microSDHC card are the same as in the project (for example, memory, power, etc.).

The PICKit 4 is now in Programmer-To-Go mode. The LED should blink green to indicate the tool has been configured successfully for Programmer-To-Go.

8. Disconnect the PICKit 4 and you're ready to use Programmer-To-Go.

5.3.3 Setting Up PTG Mode Without a Memory Card

If you do not want to send an image or program a device, you do not need to have a microSDHC card in the PICKit 4 to put it into PTG mode. This method puts the PICKit 4 into PTG mode and presumes that the necessary image is already on a microSDHC card that will be inserted in the PICKit 4 prior to using PTG to program devices.

There are several cases where you may want to do this, for example, the files for programming a device are put in a zip file and sent to a different location where they can be unzipped and downloaded to a microSDHC card which will be inserted into a PICKit 4 that's already in PTG mode. Or you may have a situation where multiple instances of PICKit 4 are put in PTG mode and microSDHC cards for a variety of devices are used to program these devices. These are some examples of why you would want to set up PTG mode without a memory card installed.

Refer to the instructions in [5.3.1 Setting Up PTG Mode Using MPLAB X IDE](#) or [5.3.2 Setting Up PTG Mode with MPLAB IPE](#) but leave both the *Send Image to Tool* and *Program Device* check boxes unchecked. Continue with the instructions provided in those sections to put the PICKit 4 into PTG mode.

Remember, in order to use PTG, you must have a microSDHC card, with the necessary image inserted in the PICKit 4.

5.4 Using Programmer-To-Go

When you are ready to start programming devices using the MPLAB PICKit 4 in PTG mode, complete the following steps:

1. Connect the PICKit 4 tool, with a microSDHC card inserted, to the target board with the device specified in your project.
2. Ensure that you have the appropriate connections to the device for Programmer-To-Go:
 - PICKit 4 is connected via appropriate programming interface connector on the target board. Ensure you match pin 1 on the target board with the pin 1 indicator on the PICKit 4.
 - The target board is powered from either the PICKit 4 or a power supply, depending on your project properties selection.

Note: In Programmer-To-Go mode, if the target is providing power to the MPLAB PICKit 4, the target board must be capable of providing 350 mA of power for the PICKit 4 tool to operate properly.

3. When the PICKit 4 LED changes to a blinking green state, it is ready to program. If no image is on the microSDHC card, the MPLAB PICKit 4 tool blinks red to indicate a PTG error.
4. To start programming the device, **firmly press (not hold) on the center of PICKit 4 shield (logo)** on the front of the tool. After the tool checks the device ID, the LED changes to blinking purple indicating that it is programming the device.
5. When programming is complete, the LED changes back to a flashing green to indicate a successful programming/verify operation. It is now ready for the next programming operation.

Note: A long press on PICKit 4 pushbutton reinitializes the tool. This can be used to reinitialize the PICKit 4 after detecting an error. It can also be used if you want to swap another microSDHC card with a different Programmer-To-Go image.

LED Status Sequence

When the PICKit 4 is in Programmer-To-Go mode and is properly connected to the target board, the following sequence occurs:

Status	Meaning
Fast blinking yellow	Initializing power settings.
Blinking green	PICKit 4 is ready to program or programming completed successfully.
Blinking purple	Programming in progress.

Status	Meaning
Blinking red	<p>Errors during initializing:</p> <ul style="list-style-type: none"> • A microSDHC card is not detected. • The microSDHC card format is not supported. This should be reported by MPLAB X IDE or MPLAB IPE when the “send image to too” was set. • Initialization files are not found on the microSDHC card. Check if Programmer-To-Go image is present in the microSDHC card. <p>Errors during programming:</p> <ul style="list-style-type: none"> • The power settings are not set properly. For example, if the PICKit 4 tool is supplying power, but it detects Vdd from the target. Or if the target is supposed to be supplying power but the PICKit tool does not detect Vdd from the target, the LED will blink red to indicate an unexpected event. • Memory verify errors. • Device ID does not match. • If the data files for programming are not found or are corrupted.

5.5 Exiting Programmer-To-Go Mode

To exit from Programmer-To-Go mode, plug the MPLAB PICKit 4 unit into a PC USB port and connect to MPLAB X IDE or MPLAB IPE. Initiate any non-PTG operation (for example, Program, Erase, etc.) and the following message displays:

“The PICKit 4 is currently in programmer to go mode. If you continue with this operation, the PICKit 4 will exit programmer to go mode. Would you like to continue?”

Select **Yes** to exit Programmer-To-Go mode.

6. Troubleshooting

If you are having problems with MPLAB PICKit 4 In-Circuit Debugger operation, start here.

6.1 Some Questions to Answer First

1. **Which device are you working with?**
Often an upgrade to a newer version of MPLAB X IDE is required to support newer devices.
2. **Are you using a Microchip demo board or one of your own design? And, have you followed the guidelines for resistors/capacitors for communications connections?**
See [3. Operation](#).
3. **Have you powered the target?**
The debugger cannot power the target if greater than 50 mA. For applications needing more than 50 mA, use an external power supply to power the target board.
4. **Are you using a USB hub in your setup? Is it powered?**
If you continue to have problems, try using the debugger without the hub (plugged directly into the computer).
5. **Are you using the USB cable shipped with the debugger?** Other USB cables may be of poor quality, too long or do not support USB Communication.

6.2 Top Reasons Why You Can't Debug

1. **Oscillator not working.** Check your Configuration bits setting for the oscillator. If you are using an external oscillator, try using an internal oscillator. If you are using an internal PLL, make sure your PLL settings are correct.
2. **No power to the target board.** Check the power cable connection.
3. **Incorrect V_{DD} voltage.** The V_{DD} voltage is outside the specifications for this device. See the device programming specification for details.
4. **Physical disconnect.** The debugger has become physically disconnected from the computer and/or the target board. Check the communications cables' connections.
5. **Communications lost.** Debugger to PC communication has somehow been interrupted. Reconnect to the debugger in MPLAB X IDE or MPLAB IPE.
6. **Device not seated.** The device is not properly seated on the target board. If the debugger is properly connected and the target board is powered, but the device is absent or not plugged in completely, you may receive the message:
`Target Device ID (0x0) does not match expected Device ID (0x%x)`
, where %x is the expected device ID.
7. **Device is code-protected.** Check your Configuration bits settings for code protection.
8. **No device debug circuitry.** The production device may not have debugging capabilities. Use a debug header instead. (See the *"Processor Extension Pak and Debug Header Specification"* (DS50001292) in [1.3 Recommended Reading](#).)
9. **Application code corrupted.** The target application has become corrupted or contains errors. Try rebuilding and reprogramming the target application. Then initiate a Power-On-Reset of the target.
10. **Incorrect programming pins.** The PGC/PGD pin pairs are not correctly programmed in your Configuration bits (for devices with multiple PGC/PGD pin pairs).
11. **Additional setup required.** Other configuration settings are interfering with debugging. Any configuration setting that would prevent the target from executing code will also prevent the debugger from putting the code into Debug mode.
12. **Incorrect brown-out voltage.** Brown-out Detect voltage is greater than the operating voltage V_{DD}. This means the device is in Reset and cannot be debugged.
13. **Incorrect connections.** Review the guidelines in [3. Operation](#) for the correct communication connections.
14. **Invalid request.** The debugger cannot always perform the action requested. For example, the debugger cannot set a breakpoint if the target application is currently running.

6.3 Other Things to Consider

6.3.1 General


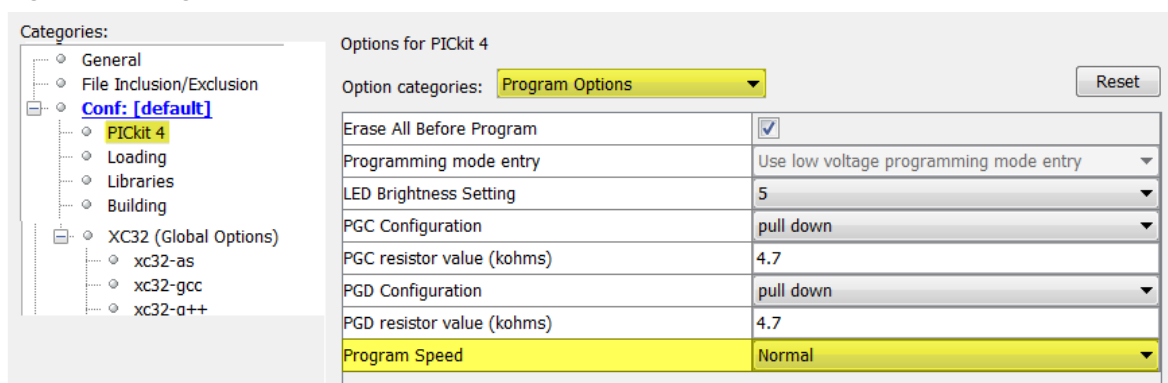
1. It is possible the error was a one-time event. Try the operation again.
2. There may be a problem programming in general. As a test, switch to Run mode using the  icon and program the target with the simplest application possible (for example, a program to blink an LED). If the program will not run, then you know that something is wrong with the target setup.
3. It is possible that the target device has been damaged in some way (for example, over current). Development environments are notoriously hostile to components. Consider trying another target board. Microchip Technology Inc. offers demonstration boards to support most of its microcontrollers. Consider using one of these applications, which are known to work, to verify correct MPLAB® PICKit™ 4 In-Circuit Debugger functionality.
4. Review debugger setup to ensure proper application setup. For more information, see 3. [Operation](#).
5. Your program speed may be set too high for your circuit. In MPLAB X IDE, go to *File > Project Properties*, select **PICKit 4** in *Categories*, then *Program Options*, *Program Speed* and select a slower speed from the drop-down menu. The default is Normal (see figure below).

Figure 6-1. Program Speed Option



6. There may be certain situations where the debugger is not operating properly and firmware may need to be downloaded or the debugger needs to be reprogrammed. See the following sections to determine additional actions.

6.3.2 How to Invoke the Bootload Mode

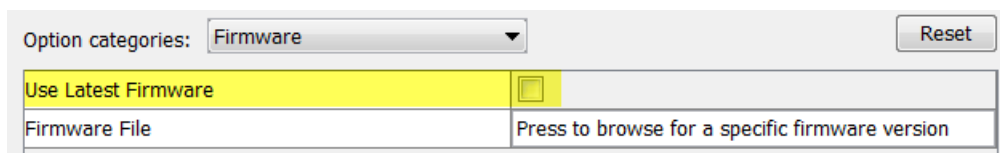
If the MPLAB X IDE or MPLAB IPE cannot communicate with the debugger, the debugger may need to be forced into bootloader mode (download new firmware). Some possible reasons could be the following:

- If steps 1-5 in 6.3.1 [General](#) did not correct the debugger issue.
- If the MPLAB X IDE Output window shows an asterisk (*) next to the Application version number, the debugger's firmware is not the newest.

Currently loaded versions:

```
Application version.....00.00.51*
FPGA version.....00.00.11
Script version.....00.01.79
Script build number.....d1a127856a
Application build number.....cef06ab8e2
```

This can occur if the Project Properties Firmware options has the "Use Latest Firmware" box unchecked and there is a new firmware version available with the MPLAB X IDE version.



In this case, check the “Use Latest Firmware” box and click the Refresh Debug Tool Status Icon in the MPLAB X IDE dashboard display. If there is still an asterisk next to the Application version number, or the debugger issue is not resolved, proceed to the following steps for bootloader mode.

Also, refer to [11.2.2 Indicator Lights Strip](#) for more information on light strip modes and bootloader errors.

Perform the following steps to force the debugger into bootloader mode:

1. Disconnect the Micro-B USB cable from the debugger.
2. Press down on the MPLAB PICKit 4 logo and hold while plugging in the Micro-B USB cable. The light strip flashes purple. Continue pressing the logo until the light strip stops flashing and changes to steady on purple. You are now in bootloader mode.
3. Try to reestablish communication with the MPLAB X IDE or MPLAB IPE. If successful, the latest firmware is downloaded. When complete, the LED is steady on blue and the debugger is ready for operation.

6.3.3 How to Use the Hardware Tool Emergency Boot Firmware Recovery Utility



Only use this utility to restore hardware tool boot firmware to its factory state. Use only if your hardware tool no longer functions on any machine.

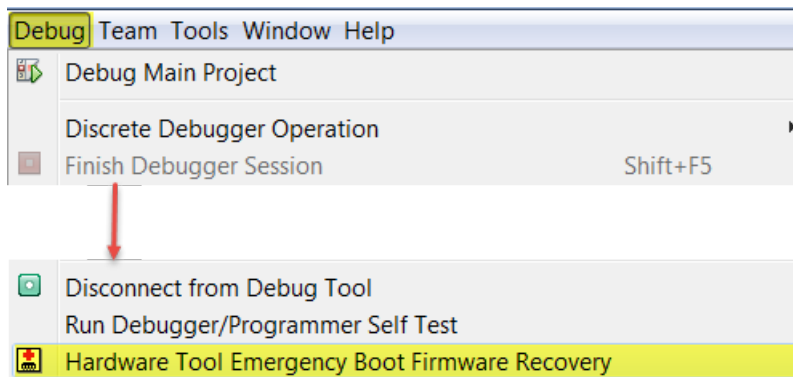
The debugger may need to be forced into recovery boot mode (reprogrammed) in rare situations; for example, if any of the following occurs when the debugger is connected to the computer:

- If the debugger has no LED lit.
- If the procedure described in [6.3.2 How to Invoke the Bootload Mode](#) was not successful.

YOU MUST USE MPLAB X IDE V4.15 OR GREATER TO USED THE EMERGENCY RECOVERY UTILITY.

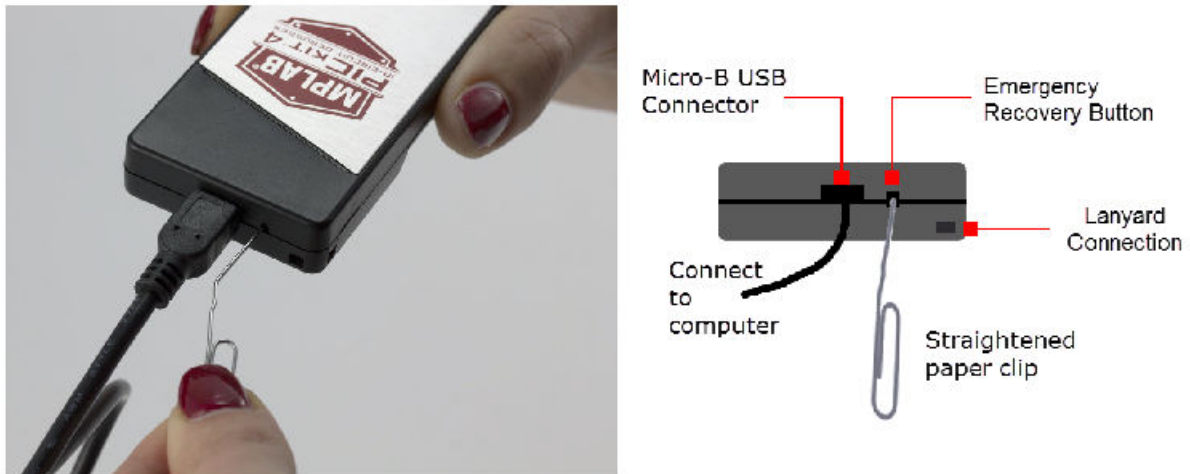
Carefully follow the instructions found in MPLAB X IDE under the main menu options *Debug > Hardware Tool Emergency Boot Firmware Recovery*.

Figure 6-2. Selecting Emergency Utility



The figure below shows where the emergency recovery button is located on the MPLAB PICKit 4 In-Circuit Debugger.

Figure 6-3. Emergency Recovery Button



If the procedure was successful, the recovery wizard displays a success screen. The MPLAB PICKit 4 will now be operational and able to communicate with the MPLAB X IDE.

If the procedure failed, try it again. If it fails a second time, contact Microchip Support at <https://support.microchip.com>.

7. Frequently Asked Questions

Look here for answers to frequently asked questions about the MPLAB PICKit 4 In-Circuit Debugger system.

7.1 How Does it Work?

What's in the silicon that allows it to communicate with the MPLAB PICKit 4 In-Circuit Debugger?

MPLAB PICKit 4 In-Circuit Debugger can communicate with Flash silicon via the ICSP™ interface. It uses the debug executive downloaded into program or test memory.

How is the throughput of the processor affected by having to run the debug executive?

The debug executive doesn't run while in Run mode, so there is no throughput reduction when running your code, that is, the debugger doesn't 'steal' any cycles from the target device.

Does the MPLAB PICKit 4 In-Circuit Debugger have complex breakpoints like other in-circuit emulators/debuggers?

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

Does the MPLAB PICKit 4 In-Circuit Debugger have complex breakpoints?

Yes. You can break based on a value in a data memory location. You can also do sequenced breakpoints, where several events have to occur before it breaks. However, you can only do two sequences. You can also do the AND condition and do PASS counts.

Is the MPLAB PICKit 4 In-Circuit Debugger optoisolated or electrically isolated?

No. You cannot apply a floating or high voltage (120V) to the current system.

Will the MPLAB PICKit 4 In-Circuit Debugger slow down the running of the program?

No. The device will run at any device speed as specified in the data sheet.


Is it possible to debug a dsPIC DSC device running at any speed?

The MPLAB PICKit 4 In-Circuit Debugger is capable of debugging at any device speed as specified in the device's data sheet.

7.2 What's Wrong?

Things to consider:

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

If **Run Main Project** icon () is selected, the device will automatically run immediately after programming. Therefore, if your code changes the Flash memory, verification could fail. To prevent the code from running after programming, select **Hold in Reset**.

My computer went into power-down/hibernate mode and now my debugger won't work. What happened?

When using the debugger for prolonged periods of time, especially as a debugger, be sure to disable the Hibernate mode in the Power Options Dialog window of your computer's operating system. Go to the **Hibernate** tab and uncheck the "Enable hibernation" check box. This will ensure that all communication is maintained across all the USB subsystem components.

I set my peripheral to NOT freeze on halt, but it is suddenly freezing. What's going on?

For dsPIC30F/33F and PIC24F/H devices, a reserved bit in the peripheral control register (usually either bit 14 or 5) is used as a Freeze bit by the debugger. If you have performed a write to the entire register, you may have overwritten this bit (the bit is user-accessible in Debug mode).

To avoid this problem, write only to the bits you wish to change for your application (BTS, BTC) instead of to the entire register (MOV).

When using a 16-bit device, an unexpected Reset occurred. How do I determine what caused it?

Some things to consider:

- To determine a Reset source, check the RCON register.
- Handle traps/interrupts in an Interrupt Service Routine (ISR). You should include `trap.c` style code, for example,

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

- Use ASSERTs. For example: `ASSERT (IPL==7)`

8. Error Messages

The MPLAB PICKit 4 In-Circuit Debugger produces various error messages; some are specific and others can be resolved with general corrective actions. In general, read any instructions under your error message. If these fail to fix the problem or if there are no instructions, refer to the following sections.

8.1 Types of Error Messages

8.1.1 Debugger-to-Target Communications Errors

Failed to send database

If you receive this error:

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

If these fail to fix the problem or if there are no instructions, see [8.2.3 Debugger to Computer Communication Error Actions](#).

8.1.2 Corrupted/Outdated Installation Errors

Failed to download firmware

If the hex file exists:

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

If the hex file does not exist:

- Reinstall MPLAB X IDE or MPLAB IPE.

Unable to download debug executive

If you receive this error while attempting to debug:

1. Deselect the debugger as the debug tool.
2. Close your project and then close MPLAB X IDE or MPLAB IPE.
3. Restart MPLAB X IDE or MPLAB IPE and reopen your project.
4. Reselect the debugger as the debug tool and attempt to program the target device again.

Unable to download program executive

If you receive this error while attempting to program:

1. Deselect the debugger as the programmer.
2. Close your project and then close MPLAB X IDE or MPLAB IPE.
3. Restart MPLAB X IDE or MPLAB IPE and reopen your project.
4. Reselect the debugger as the programmer and attempt to program the target device again.

If these actions fail to fix the problem or if there are no instructions, see [Corrupted Installation Actions](#).

8.1.3 Debug Failure Errors

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

You will receive this message if you try to Run before programming your device for the first time and try to Run. If you receive this message after this, or immediately after programming your device, please refer to [8.2.6 Debug Failure Actions](#).

The device is code protected.

The device on which you are attempting to operate (read, program, blank check or verify) is code protected, in other words, the code cannot be read or modified. Check your Configuration bits setting for code protection ([Windows > Target Memory Views > Configuration Bits](#)).


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

If these actions fail to fix the problem, see [Debugger to Target Communication Error Actions](#) and [8.2.6 Debug Failure Actions](#).

8.1.4 Miscellaneous Errors

MPLAB PICKIT 4 is busy. Please wait for the current operation to finish.

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

1. Wait. Give the debugger time to finish any application tasks. Then try to deselect the debugger again.
2. Select  (Finish Debugger Session) to stop any running applications. Then, try to deselect the debugger again.
3. Unplug the debugger from the computer. Then, try to deselect the debugger again.
4. Shut down MPLAB X IDE.

8.1.5 List of Error Messages

Table 8-1. Alphabetized List Of Error Messages

AP_VER=Algorithm Plugin Version
AREAS_TO_PROGRAM=The following memory area(s) will be programmed:
AREAS_TO_READ=The following memory area(s) will be read:
AREAS_TO_VERIFY=The following memory area(s) will be verified:
BLANK_CHECK_COMPLETE=Blank check complete, device is blank.
BLANK_CHECK_FAILED=Blank check failed. The device is not blank.
BLANK_CHECKING=Blank Checking...
BOOT_CONFIG_MEMORY=boot config memory
BOOT_VER=Boot Version
BOOTFLASH=boot flash
BP_CANT_B_DELETED_WHEN_RUNNING=software breakpoints cannot be removed while the target is running. The selected breakpoint will be removed the next time the target halts.
CANT_CREATE_CONTROLLER=Unable to find the tool controller class.
CANT_FIND_FILE=Unable to locate file %s.
CANT_OP_BELOW_LVPTHRESH=The voltage level selected %f, is below the minimum erase voltage of %f. The operation cannot continue at this voltage level.
CANT_PGM_USEROTP=The debug tool cannot program User OTP memory because it is not blank. Please exclude User OTP memory from the memories to program or switch to a device with blank User OTP memory.
CANT_PRESERVE_PGM_MEM=Unable to preserve program memory: Invalid range Start = %08x, End = %08x.
CANT_READ_REGISTERS=Unable to read target register(s).
CANT_READ_SERIALNUM=Unable to read the device serial number.
CANT_REGISTER_ALTERNATE_PNP=Unable to register for PNP events for multiple USB product IDs.

CANT_REMOVE_SWPS_BUSY=The ICD 4 is currently busy and cannot remove software breakpoints at this time.
CHECK_4_HIGH_VOLTAGE_VPP=CAUTION: Check that the device selected in MPLAB IDE (%s) is the same one that is physically attached to the debug tool. Selecting a 5V device when a 3.3V device is connected can result in damage to the device when the debugger checks the device ID. Do you wish to continue?
CHECK_PGM_SPEED=You have set the program speed to %s. The circuit on your board may require you to slow the speed down. Please change the setting in the tool properties to low and try the operation again.
CHECK_SLAVE_DEBUG=Debugging may have failed because the, "Debug" check box in the Slave Core settings of the master project has not been enabled. Please make sure this setting is enabled.
COMM_PROTOCOL_ERROR=A communication error with the debug tool has occurred. The tool will be reset and should re-enumerate shortly.
COMMAND_TIME_OUT=PICKit 4 has timeout out waiting for a response to command %02x.
CONFIGURATION=configuration
CONFIGURATION_MEMORY=configuration memory
CONNECTION_FAILED=Connection Failed
CORRUPTED_STREAMING_DATA=Invalid streaming data has been detected. Run time watch or trace data may no longer be valid. It is recommended that you restart your debug session.
CPM_TO_TARGET_FAILED=An exception occurred during ControlPointMediator.ToTarget().
DATA_FLASH_MEMORY=Data Flash memory
DATA_FLASH=data flash
DEBUG_INFO_PGM_FAILED=Could not enter debug mode because programming the debug information failed. Invalid combinations of config bits may cause this problem.
DEBUG_READ_INFO=Reading the device while in debug mode may take a long time due to the target oscillator speed. Reducing the range that you'd like to read (under the ICD 4 project properties) can mitigate the situation. The abort operation can be used to terminate the read operation if necessary.
DEVICE_ID_REVISION=Device ID Revision
DEVICE_ID=Device ID
DEVICE_INFO_CONFIG_BITS_MASK=Address = %08x, Mask = %08x
DEVICE_INFO_MEMBERS=DeviceInfo: pcAddress = %08x, Vpp = %.2f, useRowEraseIfVttagelsLow = %s, voltageBelowWhichUseRowErase = %.2f, deviceName = %s, programmerType = %s
DEVICE_INFO_MEMINFO_MEMBERS= DeviceInfo: mask = %04x, exists = %s, startAddr = %08x, endAddr = %08x, rowSize = %04x, rowEraseSize = %04x, addrInc = %04x, widthProgram = %04x
DEVICE_INFO=DeviceInfo: Values:
DEVID_MISMATCH=Target Device ID (0x%x) is an Invalid Device ID. Please check your connections to the Target Device.
DFU_NOT_SUPPORTED=MPLAB X has detected the tool connected has capabilities that this version does not support. Please download the latest version of MPLAB X to use this tool.
DISCONNECT_WHILE_BUSY=The tool was disconnected while it was busy.
EEDATA_MEMORY=EEData memory
EEDATA=EEData
EMPTY_PROGRAM_RANGES=The programming operation did not complete because no memory areas have been selected.

EMULATION_MEMORY_READ_WRITE_ERROR=An error occurred while trying to read/write MPLAB's emulation memory: Address=%08x
END=end
ENSURE_SELF_TEST_READY=Please ensure the RJ-11 cable is connected to the test board before continuing.
ENSURE_SELF_TEST_READY=Please ensure the RJ-11 cable is connected to the test board before continuing. Would you like to continue?
ENV_ID_GROUP=Device Identification
ERASE_COMPLETE=Erase successful
ERASING=Erasing...
FAILED_2_PGM_DEVICE=Failed to program device.
FAILED_CREATING_COM=Unable create communications object.
FAILED_CREATING_DEBUGGER_MODULES=Initialization failed: Failed creating the debugger module.
FAILED_ERASING=Failed to erase the device.
FAILED_ESTABLISHING_COMMUNICATION=Unable to establish tool communications.
FAILED_GETTING_DBG_EXEC=A problem occurred while trying to load the debug executive.
FAILED_GETTING_DEVICE_INFO=Initialization failed: Failed while retrieving device database (.pic) information
FAILED_GETTING_EMU_INFO=Initialization failed: Failed getting emulation database information
FAILED_GETTING_HEADER_INFO=Initialization failed: Failed getting header database information
FAILED_GETTING_PGM_EXEC=A problem occurred while trying to load the program executive.
FAILED_GETTING_TEX=Unable to obtain the ToolExecMediator
FAILED_GETTING_TOOL_INFO=Initialization failed: Failed while retrieving tool database (.ri4) information
FAILED_INITING_DATABASE=Initialization failed: Unable to initialize the too database object
FAILED_INITING_DEBUGHANDLER=Initialization failed: Unable to initialize the DebugHandler object
FAILED_PARSING_FILE=Failed to parse firmware file: %s
FAILED_READING_EMULATION_REGS=Failed to read emulation memory.
FAILED_READING_MPLAB_MEMORY=Unable to read %s memory from %0x08 to %0x08.
FAILED_READING_SECURE_SEGMENT=A failure occurred while reading secure segment configuration bits
FAILED_SETTING_PC=Unable to set PC.
FAILED_SETTING_SHADOWS=Failed to properly set shadow registers.
FAILED_SETTING_XMIT_EVENTS=Unable to synchronize run time data semaphores.
FAILED_STEPPING=Failed while stepping the target.
FAILED_TO_GET_DEVID=Failed to get Device ID. Please make sure the target device is attached and try the operation again.
FAILED_TO_INIT_TOOL=Failed to initialize PICKit 4
FAILED_UPDATING_BP=Failed to update breakpoint:\nFile: %s\naddress: %08x
FAILED_UPDATING_FIRMWARE=Failed to properly update the firmware.
FILE_REGISTER=file register
FIRMWARE_DOWNLOAD_TIMEOUT=PICKit 4 timeout out during the firmware download process.

FLASH_DATA_MEMORY=Flash data memory
FLASH_DATA=flash data
FRCINDEBUG_NEEDS_CLOCKSWITCHING=To use FRC in debug mode the clock switching configuration bits setting must be enabled. Please enable clock switching and retry the requested operation.
FW_DOESNT_SUPPORT_DYNBP=The current PICKit 4 firmware does not support setting run time breakpoints for the selected device. Please download firmware version %02x.%02x.%02x or higher.
GOOD_ID_MISMATCH=Target Device ID (0x%x) is a valid Device ID but does not match the expected Device ID (0x%x) as selected.
HALTING=Halting...
HIGH=High
HOLDMCLR_FAILED=Hold in reset failed.
IDS_SELF_TEST_BOARD_PASSED=PICKit 4 is functioning properly. If you are still having problems with your target circuit please check the Target Board Considerations section of the online help.
IDS_ST_CLKREAD_ERR=Test interface PGC clock line read failure.
IDS_ST_CLKREAD_NO_TEST=Test interface PGC clock line read not tested.
IDS_ST_CLKREAD_SUCCESS=Test interface PGC clock line read succeeded.
IDS_ST_CLKWRITE_ERR=Test interface PGC clock line write failure. Please ensure that the tester is properly connected.
IDS_ST_CLKWRITE_NO_TEST=Test interface PGC clock line write not tested.
IDS_ST_CLKWRITE_SUCCESS=Test interface PGC clock line write succeeded.
IDS_ST_DATREAD_ERR=Test interface PGD data line read failure.
IDS_ST_DATREAD_NO_TEST=Test interface PGD data line read not tested.
IDS_ST_DATREAD_SUCCESS=Test interface PGD data line read succeeded.
IDS_ST_DATWRITE_ERR=Test interface PGD data line write failure.
IDS_ST_DATWRITE_NO_TEST=Test interface PGD data line write not tested.
IDS_ST_DATWRITE_SUCCESS=Test interface PGD data line write succeeded.
IDS_ST_LVP_ERR=Test interface LVP control line failure.
IDS_ST_LVP_NO_TEST=Test interface LVP control line not tested.
IDS_ST_LVP_SUCCESS=Test interface LVP control line test succeeded.
IDS_ST_MCLR_ERR=Test interface MCLR level failure.
IDS_ST_MCLR_NO_TEST=Test interface MCLR level not tested.
IDS_ST_MCLR_SUCCESS=Test interface MCLR level test succeeded.
IDS_TEST_NOT_COMPLETED=Interface test could not be completed. Please contact your local FAE/CAE to SAR the unit.
INCOMPATIBLE_FW=The REAL ICE firmware is not compatible with the current version of MPLAB X software.
INVALID_ADDRESS=The operation cannot proceed because the %s address is outside the devices address range of 0x%08x - 0x%08x.
JTAG_NEEDS_JTAGEN=The JTAG Adapter requires the JTAG enable configuration bit to be turned on. Please enable this configuration bit before continuing.

MCLR_HOLD_RESET_NO_MAINTAIN_POWER=WARNING: You are powering the target device from PICKit 4 and have not selected the, "Maintain active power" option on the PICKit 4's Power property page. Without this option, the state of MCLR (hold/release from reset) cannot be guaranteed after the current session has ended.
MCLR_OFF_ID_WARNING=If you are using low voltage programming and the MCLRE config bit on the target device is set to OFF, this may explain why the device ID is incorrect. In this case, please switch to the "\"Use high voltage programming mode entry\" Program mode entry setting on the PICKit 4 Program Options property page and try the operation again.
MCLR_OFF_WARNING=If you wish to continue with MCLRE configuration bit set to OFF, switch to the "\"Use high voltage programming mode entry\" Program mode entry setting on the PICKit 4 Program Options property page.
MEM_INFO=DeviceInfo: MemInfo values:
MEM_RANGE_ERROR_BAD_END_ADDR=Invalid program range end address %s received. Please check the manual program ranges on the debug tool's, "Memories to Program" property page.
MEM_RANGE_ERROR_BAD_START_ADDR=Invalid program range start address %s received. Please check the manual program ranges on the debug tool's, "Memories to Program" property page.
MEM_RANGE_ERROR_END_LESSTHAN_START=Invalid program range received: end address %s < start address %s. Please check the manual program ranges on the debug tool's, "Memories to Program" property page.
MEM_RANGE_ERROR_ENDADDR_NOT_ALIGNED=Invalid program range received: end address %s is not aligned on a proper 0x%x address boundary. Please check the manual program ranges on the debug tool's, "Memories to Program" property page.
MEM_RANGE_ERROR_STARTADDR_NOT_ALIGNED=Invalid program range received: start address %s is not aligned on a proper 0x%x address boundary. Please check the manual program ranges on the debug tool's, "Memories to Program" property page.
MEM_RANGE_ERROR_UNKNOWN=An unknown error has occurred while trying to validate the user entered memory ranges.
MEM_RANGE_ERROR_WRONG_DATABASE=Unable to access data object while validating user entered memory ranges.
MEM_RANGE_OUT_OF_BOUNDS=The selected program range, %s, does not fall within the proper range for the memory area selected. Please check the manual program ranges on the debug tool's, "Memories to Program" property page.
MEM_RANGE_STRING_MALFORMED=The memory range(s) entered on the, "Memories to Program" property page (%s) is not formatted properly.
MISSING_BOOT_CONFIG_PARAMETER=Unable to find boot config start/end address in database.
MUST_NOT_USE_LVP_WHEN_LVPCFG_OFF=MPLAB has detected that the low voltage configuration bit on the device is off and you have selected the low voltage programming option on the debug tool's property page. If you wish to use the low voltage programming option you must first do the following:\n* Turn off the low voltage programming option on the debug tool's Program Options property page\n* Program the low voltage configuration bit to on\n* Turn on the low voltage programming option on the debug tool's Program Options property page.
MUST_SET_LVPBIT_WITH_LVP=The low voltage programming feature requires the LVP configuration bit to be enabled on the target device. Please enable this configuration bit and try the operation again.
NEW_FIRMWARE_NO_DEVICE=Downloading firmware.
NEW_FIRMWARE=Now Downloading new Firmware for target device: %s
NMMR=NMMR
NO_DYNAMIC_BP_SUPPORT_AT_ALL=The current device does not support the ability to set breakpoints while the device is running. The breakpoint will be applied prior to the next time you run the device.
NO_PGM_HANDLER=Cannot program software breakpoints. The program handler has not been initialized.

NO_PROGRAMMING_ATTEMPTED=MPLAB's memory is blank so no programming operation was attempted.
NORMAL=Normal
OP_FAILED_FROM_CP=The requested operation failed because the device is code protected.
OpenIDE-Module-Name=PICKit 4
OPERATION_INFO_MEMBERS=OperationInfo: Type = %s, Mask = %08x, Erase = %s, Production Mode = %s.
OPERATION_INFO_TRANSFER_INFO_MEMBERS=OperationInfo: Start = %x, End = %x, Buffer Length = %d, Type = %s, Mask = %08x.
OPERATION_INFO=OperationInfo: Values:
OPERATION_NOT_SUPPORTED=This operation is not supported for the selected device
OUTPUTWIN_TITLE=PICKit 4
PERIPHERAL=Peripheral
POWER_ERROR_NO_9V=The configuration is set for the tool to provide power to the target but the 9V power jack is not detected. Please ensure the external 9V barrel jack is connected to the tool.
POWER_ERROR_NO_POWER_SRC=The configuration is set for the target board to supply its own power but no voltage has been detected on VDD. Please ensure you have your target powered up and try again.
POWER_ERROR_POWER_SRC_CONFLICT=The configuration is set for the tool to provide power to the target but there is voltage already detected on VDD. This is a conflict. Please ensure your target is not supplying voltage to the tool and try again.
POWER_ERROR_SLOW_DISCHARGE= There seems to be excessive capacitance on VDD causing a slower system discharge and shutdown. Consider minimizing overall capacitance loading or use power from your target to avoid discharge delays.
POWER_ERROR_UNKNOWN=An unknown power error has occurred.
POWER_ERROR_VDD_TOO_HIGH=The VDD voltage desired is out of range. It exceeds the maximum voltage of 5.5V.
POWER_ERROR_VDD_TOO_LOW=The VDD voltage desired is out of range. It is below the minimum voltage of 1.5V.
POWER_ERROR_VPP_TOO_HIGH=The VPP voltage desired is out of range. It exceeds the maximum voltage of 14.2V.
POWER_ERROR_VPP_TOO_LOW=The VPP voltage desired is out of range. It is below the minimum voltage of 1.5V.
PRESERVE_MEM_RANGE_ERROR_BAD_END_ADDR=Invalid preserve range end address %s received. Please check the manual program ranges on the debug tool's, "Memories to Program" property page.
PRESERVE_MEM_RANGE_ERROR_BAD_START_ADDR=Invalid preserve range start address %s received. Please check the manual program ranges on the debug tool's, "Memories to Program" property page.
PRESERVE_MEM_RANGE_ERROR_END_LESSTHAN_START=Invalid preserve range received: end address %s < start address %s. Please check the manual program ranges on the debug tool's, "Memories to Program" property page.
PRESERVE_MEM_RANGE_ERROR_ENDADDR_NOT_ALIGNED=Invalid preserve range received: end address %s is not aligned on a proper 0x%x address boundary. Please check the manual program ranges on the debug tool's, "Memories to Program" property page.
PRESERVE_MEM_RANGE_ERROR_STARTADDR_NOT_ALIGNED=Invalid preserve range received: start address %s is not aligned on a proper 0x%x address boundary. Please check the manual program ranges on the debug tool's, "Memories to Program" property page.

PRESERVE_MEM_RANGE_ERROR_UNKNOWN=An unknown error has occurred while trying to validate the user entered preserve ranges.
PRESERVE_MEM_RANGE_ERROR_WRONG_DATABASE=Unable to access data object while validating user entered memory ranges.
PRESERVE_MEM_RANGE_MEM_NOT_SELECTED=You have selected to preserve an area of memory but have not selected to program that area. Please check the preserved ranges on the debug tool's "Memories to Program" property page and make sure that any preserved memory is also designated to be programmed.
PRESERVE_MEM_RANGE_OUT_OF_BOUNDS=The selected preserve range, %s, does not fall within the proper range for the memory area selected. Please check the manual program ranges on the debug tool's "Memories to Program" property page.
PRESERVE_MEM_RANGE_STRING_MALFORMED=The preserve memory range(s) entered on the, "Memories to Program" property page (%s) is not formatted properly.
PRESERVE_MEM_RANGE_WONT_BE_PROGRAMMED_AUTO_SELECT=Some or all of the preserve memory ranges (%s) entered on the, "Memories to Program" property page, do not fall under the indicated program range(s) (%s) for the memory selected. Please deselect the \u201cAuto select memories and ranges\u201d option on the \u201cMemories to Program\u201d property page, change to manual mode and adjust your range(s) accordingly.
PRESERVE_MEM_RANGE_WONT_BE_PROGRAMMED=Some or all of the preserve memory ranges (%s) entered on the, "Memories to Program" property page, do not fall under the indicated program range(s) (%s) for the memory selected. Please check the preserved ranges on the debug tool's, "Memories to Program" property page.
PROGRAM_CFG_WARNING=WARNING: You have selected to program configuration memory. Programming invalid values into any of the configuration fields may have unintended consequences. Please make sure that EVERY configuration field has a valid value. If you are not sure, you can read the configuration values off of device first and then change only the fields you are concerned with. Would you like to continue programming?
PROGRAM_COMPLETE=Programming/Verify complete
PROGRAM_MEMORY=program memory
PROGRAM=program
PROGRAMMING_DID_NOT_COMPLETE=Programming did not complete.
READ_COMPLETE=Read complete
READ_DID_NOT_COMPLETE=Read did not complete.
RELEASEMCLR_FAILED=Release from reset failed.
REMOVING_SWBPS_COMPLETE=Removing software breakpoints complete
REMOVING_SWBPS=Removing software breakpoints...
RESET_FAILED=Failed to reset the device.
RESETTING=Resetting...
RISKY_CFG_RANGE_REMOVED=The configuration memory will not be included in the program operation because the, "Exclude configuration memory from programming" option is set. To change this, go to the Memories to Program property page and uncheck the setting.\nWARNING; Programming configuration values on this device can cause unintended consequences if all of the configuration values are not properly set. It is advised that you read the configuration values off of device first and then change only the fields you are concerned with.
RUN_INTERRUPT_THREAD_SYNCH_ERROR=An internal run error has occurred. It is advised that you restart your debug session. You may continue running but certain run time features may no longer work properly.
RUN_TARGET_FAILED=Unable to run the target device.
RUNNING=Running
SERIAL_NUM=Serial Number:\n

SETTING_SWBPS=Setting software breakpoints.....
STACK=stack
START_AND_END_ADDR=start address = 0x%x, end address = 0x%x
START=start
TARGET_DETECTED=Target voltage detected
TARGET_FOUND=Target device %s found.
TARGET_HALTED=Target Halted
TARGET_NOT_READY_4_DEBUG=The target device is not ready for debugging. Please check your configuration bit settings and program the device before proceeding. The most common causes for this failure are oscillator and/or PGC/PGD settings.
TARGET_VDD=Target VDD:
TEST=test
TOOL_INFO_MEMBERS=ToolInfo: speedLevel = %d, PGCResistance = %d, PGDResistance = %d, PGCPullDir = %s, PGDPullDir = %s, ICSPSelected = %s.
TOOL_INFO=ToolInfo: Values:
TOOL_IS_BUSY=PICKit 4 is busy. Please wait for the current operation to finish.
TOOL_SUPPLYING_POWER=PICKit 4 is supplying power to the target (%.2f volts).
TOOL_VDD=VDD:
TOOL_VPP=VPP:
UNABLE_TO_OBTAIN_RESET_VECTOR=PICKit 4 was unable to retrieve the reset vector address. This indicates that no _reset symbol has been defined and may prevent the device from starting up properly.
UNKNOWN_MEMTYPE=Unknown memory type
UNLOAD_WHILE_BUSY=PICKit 4 was unloaded while still busy. Please unplug and reconnect the USB cable before using PICKit 4 again.
UPDATING_APP=Updating firmware application...
UPDATING_BOOTLOADER=Updating firmware bootloader.
USE_LVP_PROGRAMMING=NOTE: If you would like to program this device using low voltage programming, select Cancel on this dialog. Then go to the PICKit 4 node of the project properties and check the Enable Low Voltage Programming check box of the Program Options Option Category pane (low voltage programming is not valid for debugging operations).
USERID_MEMORY=User Id Memory
USERID=user Id
VERIFY_COMPLETE=Verification successful.
VERIFY_FAILED=Verify failed
VERSIONS=Versions
VOLTAGE_LEVEL_BAD_VALUE_EX=You have entered an invalid value %s for the Voltage Level on the PICKit 4 Power property page. Please fix this before continuing.
VOLTAGE_LEVEL_BAD_VALUE=Unable to parse the voltage level %s. Please enter a valid voltage entry.
VOLTAGE_LEVEL_OUT_OF_RANGE=The target voltage level you have entered, %.3f, is outside the range of the device %.3f - %.3f.

VOLTAGES=Voltages

WOULD_YOU_LIKE_TO_CONTINUE=Would you like to continue?

WRONG_PICKIT_4_FLAVOR=Your PICKit 4 hardware needs updating please contact PICKit 4_Update@microchip.com to get a replacement.

8.2 General Corrective Actions

8.2.1 Read/Write Error Actions

If you receive a read or write error:

1. Did you click *Debug > Reset*? This may produce read/write errors.
2. Try the action again. It may be a one-time error.
3. Ensure that the target is powered and at the correct voltage levels for the device. See the device data sheet for required device voltage levels.
4. Ensure that the debugger-to-target connection is correct (PGC and PGD are connected).
5. For write failures, ensure that “Erase all before Program” is checked on the Program Options for the debugger (see [10.2.2 Debug](#)).
6. Ensure that the cable(s) are of the correct length.

8.2.2 Debugger to Target Communication Error Actions

If the MPLAB PICKit 4 In-Circuit Debugger and the target device are *not* communicating with each other:

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

8.2.3 Debugger to Computer Communication Error Actions

If the MPLAB PICKit 4 In-Circuit Debugger and MPLAB X IDE or MPLAB IPE are not communicating with each other:

1. Unplug and then plug in the debugger.
2. Reconnect to the debugger.
3. Try the operation again. It is possible the error was a one-time event.
4. The version of MPLAB X IDE or MPLAB IPE installed may be incorrect for the version of firmware loaded on the MPLAB PICKit 4 In-Circuit Debugger. Follow the steps outlined in [8.2.4 Corrupted Installation Actions](#).
5. There may be an issue with the computer USB port. See section [8.2.5 USB Port Communication Error Actions](#).

8.2.4 Corrupted Installation Actions

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

1. Uninstall all versions of MPLAB X IDE or MPLAB IPE from the computer.
2. Reinstall the desired MPLAB X IDE or MPLAB IPE version.
3. If the problem persists, contact Microchip Support.

8.2.5 USB Port Communication Error Actions

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

1. Reconnect to the MPLAB PICKit 4 In-Circuit Debugger.
2. Make sure the debugger is physically connected to the computer on the appropriate USB port.
3. Make sure the appropriate USB port has been selected in the debugger options (see [10.2 Debugger Options Selection](#)).
4. Make sure the USB port is not in use by another device.
5. If using a USB hub, make sure it is powered.
6. Make sure the USB drivers are loaded.

8.2.6 Debug Failure Actions

The MPLAB PICKit 4 In-Circuit Debugger was unable to perform a debugging operation. There are numerous reasons why this might occur. See [6.2 Top Reasons Why You Can't Debug](#) and [6.3 Other Things to Consider](#).

8.2.7 Internal Error Actions

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

The most likely cause is a corrupted installation ([8.2.4 Corrupted Installation Actions](#)).

Another likely cause is exhausted system resources.

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

If the problem persists, contact Microchip Support.

9. Engineering Technical Notes (ETNs)

The following ETNs are related to the MPLAB® PICKit™ 4 In-Circuit Debugger. Please see the product web page for details.

ETN37: MPLAB PICKit 4 V_{PP} Overshoot Modification can be found on the MPLAB PICKit 4 product web page at <https://www.microchip.com/Developmenttools/ProductDetails/PG164140>.

10. Debugger Function Summary

A summary of the MPLAB PICKit 4 In-Circuit Debugger functions are summarized below.

10.1 Debugger Selection and Switching

Use the Project Properties dialog to select or switch debuggers for a project. To switch you must have more than one debugger connected to your computer. MPLAB X IDE will differentiate between the debuggers by displaying different serial numbers.

To select or change the debugger used for a project:

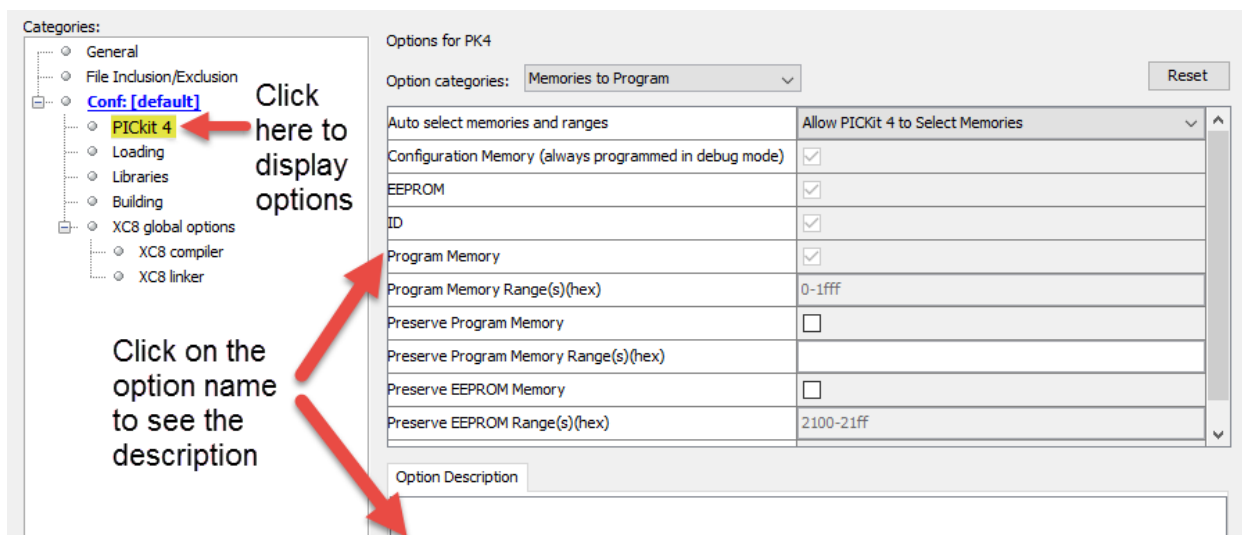
1. Open the Project Properties dialog by doing one of the following:
 - 1.1. Click on the project name in the Projects window and select *File > Project Properties*.
 - or
 - 1.2. Right click on the project name in the Projects window and select **Properties**.
2. Under **Categories** on the left side, expand “Conf:[default]” to show PICKit 4.
3. Under **Hardware Tools**, find **PICKit 4** and click on a serial number (SN) to select a debugger for use in the project, then click **Apply**.


10.2 Debugger Options Selection

Debugger options are set in the Project Properties dialog. Click on **PICKit 4** under **Categories** to display the **Options for PICKit 4** (see figure below). Use the **Options categories** drop list to select various options. Click on an option name to see its description in the Option Description box below. Click to the right of an option name to select or change it.

Note: The available option categories and the options within those categories are dependent on the device you have selected.

Figure 10-1. Options for MPLAB® PICKit™ 4



After setting the options, click **Apply** or **OK**. Also click the Refresh Debug Tool status icon  in the MPLAB X IDE dashboard display to update any changes made.

The possible option categories are:

- [10.2.1 Memories to Program](#)
- [10.2.2 Debug](#)

- [10.2.3 Program](#)
- [10.2.4 Freeze Peripherals](#)
- [10.2.5 Power](#)
- [10.2.6 Programmer-To-Go](#)
- [10.2.7 Secure Segment](#)
- [10.2.8 Firmware](#)
- [10.2.9 Clock](#)
- [10.2.10 Communication](#)

10.2.1 Memories to Program

Select the memories to be programmed into the target. The table below shows all the possible options, however, only those options available for your selected device will be displayed in MPLAB X IDE.

Note: If **Erase All Before Program** is selected as shown in [10.2.3 Program](#) then all device memory will be erased before programming.

Table 10-1. Memories to Program Option Category

Auto select memories and ranges	Allow PICKit 4 to Select Memories - The debugger uses your selected device and default settings to determine what to program. Manually select memories and ranges - You select the type and range of memory to program (see below).
Configuration Memory	Check to include <i>Configuration Memory</i> in the area(s) to be programmed. This is always programmed in Debug mode.
Boot Flash	Check to include <i>Boot Flash</i> memory in the area(s) to be programmed. This is always programmed in Debug mode.
EEPROM	Check to include <i>EEPROM</i> memory in the area(s) to be programmed.
ID	Check to program the user ID.
Program Memory	Check to program the target program memory range specified below.
Program Memory Range(s) (hex)	The range(s) of program memory to be programmed. These are the starting and ending hex address range(s) in program memory for programming, reading, or verification. Each range must be two hex numbers (the start and end addresses of the range) separated by a dash. Multiple ranges must be separated by a comma (for example, 0-ff, 200-2ff). Ranges must be aligned on a 0x800 address boundary. Note: The address range does not apply to the Erase function. The Erase function will erase all data on the device.
Preserve Program Memory	Enabling this option will cause the current program memory on the device to be read into MPLAB X IDE's memory and then reprogrammed back to the target device when programming is done. The range(s) of program memory that will be preserved is determined by the Preserve Program Memory Range(s) option below. Ensure that code is NOT code protected.
Preserve Program Memory Range(s) (hex)	The range(s) of program memory to be preserved. Each range must be two hex numbers, representing the start and end addresses of the range, separated by a dash. Ranges must be separated by a comma (for example, 0-ff, 200-2ff). Areas are reserved by reading them into MPLAB X IDE and then programming them back down when a program operation occurs. Thus the preserved areas must lie within a memory range that will be programmed.

Preserve (Type of) Memory	Enabling this option will cause the current memory type on the device to be read into MPLAB X IDE's memory and then reprogrammed back to the target device when programming is done. Check to preserve <i>Memory</i> for reprogramming, where <i>Memory</i> is the type of memory. Types include: EEPROM, ID, Boot Flash, and Auxiliary. Ensure that code is NOT code protected.
Preserve (Type of) Memory Range(s) (hex)*	The range(s) of the memory type to be preserved. Each range must be two hex numbers, representing the start and end addresses of the range, separated by a dash. Ranges must be separated by a comma (for example, 0-ff, 200-2ff). Areas are reserved by reading them into MPLAB X IDE and then programming them back down when a program operation occurs. Thus the preserved areas must lie within a memory range that will be programmed. <i>Memory</i> is the type of memory, which includes EEPROM, ID, Boot Flash, and Auxiliary. Ensure that code is NOT code protected.
* If you receive a programming error due to an incorrect range, ensure the range does not exceed available/remaining device memory.	

10.2.2 Debug

If this option is available for the project device, you can select to use software breakpoints.

Table 10-2. Debug Option Category

Use Software Breakpoints	Check to use software breakpoints. Uncheck to use hardware breakpoints. See the following table to determine which type is best for your application.
--------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------

Table 10-3. Software vs. Hardware Breakpoints

Features	Software Breakpoints	Hardware Breakpoints
Number of breakpoints	Unlimited	Limited
Breakpoints are written to	Program Memory	Debug Registers
Time to set breakpoints	Oscillator Speed Dependent – can take minutes	Minimal
Skidding	No	Yes
Note: Using software breakpoints for debugging impacts device endurance. Therefore, it is recommended that devices used in this manner not be used as production parts.		

10.2.3 Program

Choose to erase all memory before programming or to merge code.

Table 10-4. Program Option Category

Erase All Before Program	Enabling this option will cause the entire device to be erased prior to programming the data from MPLAB X IDE. Any memory areas designated to be preserved will be read before the device is erased and reprogrammed on the device when the device is programmed. Unless programming new or already erased devices, it is important to have this box checked. If not checked, the device is not erased and program code will be merged with the code already in the device.
--------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Programming mode entry	<p>This option designates the method the debugger will use to put the target device in programming mode. For the low-voltage method, V_{PP} will not exceed the V_{DD} supply voltage. Instead a test pattern will be used on V_{PP}. For the high-voltage method, a voltage in excess of 9 volts will be placed on V_{PP}.</p> <p>Note: High voltage programming requires V_{DD} above 2.8V. Select low voltage programming if your target voltage is below 2.8V.</p> <p>Low voltage and high voltage program mode entry - 2.8 to 5.0V.</p> <p>Low voltage program mode entry only - 1.2 to 5.0V.</p>
LED Brightness Setting	Select the level of brightness from 1 (darkest) to 10 (brightest).
PGC Configuration	This option determines the type of resistance that will be applied to the PGC line (pull down, pull up or none). The default is pull down. The value of the resistance is determined by the PGC resistor value option below.
PGC resistor value (k Ω)	Type in a resistor value from 0-50. The default value is 4.7 k Ω . If the PGC configuration is set to none, this value is ignored.
PGD Configuration	Select either none, pull up or pull down. The default is pull down. The value of the resistance is determined by the PGD resistor value option below.
PGD resistor value (k Ω)	Type in a resistor value from 0-50. The default value is 4.7 k Ω . If the PGD configuration is set to none, this value is ignored.
Program Speed	Select the speed the debugger will use to program the target as either Low, Normal or High. The default is Normal. If programming should fail, using a slower speed may solve the problem.

10.2.4 Freeze Peripherals

Select from the list of peripherals to freeze or not freeze on program halt. The available peripherals are device dependent.

PIC12/16/18 MCU Devices

To freeze/unfreeze all device peripherals on halt, check/uncheck the "Freeze on Halt" check box. If this does not halt your desired peripheral, be aware that some peripherals do not have a freeze-on-halt capability and cannot be controlled by the debugger.

dsPIC, PIC24 and PIC32 Devices

Select the peripheral's check box in the "Peripherals to Freeze on Halt" list to freeze that peripheral on a halt. Uncheck the peripheral to let it run while the program is halted. If you do not see a peripheral on the list, check "All Other Peripherals." If this does not halt your desired peripheral, be aware that some peripherals do not have a freeze-on-halt capability and cannot be controlled by the debugger.

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

10.2.5 Power

Select power options.

Table 10-5. Power Option Category

Power Target Circuit from PICKit 4	If checked, this option will allow the PICKit 4 to power the target circuit. Otherwise an external power supply must be used (see 3.2.4 Debugger Powered).
Voltage Level	If the "Power Target Circuit from PICKit 4" check box is checked, select the target V_{DD} that the debugger will provide.

10.2.6 Programmer-To-Go

Select the Programmer-To-Go options.

Table 10-6. Programmer-To-Go Option Category

Image Name	The default image name is "<your project name>_ptg," though you can edit the name if you wish. This will be the folder name on the microSDHC card that contains the appropriate files for Programmer-To-Go.
Send image to tool	This check box is selected by default. With the box checked, the PTG image is created and then sent to the microSDHC card in the connected MPLAB PICKit 4.
Program Device	This check box is selected by default. With the box checked, the device connected to the MPLAB PICKit 4 is programmed.

10.2.7 Secure Segment

Select and load debugger firmware.

Table 10-7. Secure Segment Option Category

Segments to be Programmed	Select one of the following: <ol style="list-style-type: none"> 1. Full Chip Programming (default). 2. Boot, Secure and General Segments. 3. Secure and General Segments. 4. General Segment Only.
---------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

10.2.8 Firmware

Select and load debugger firmware.

Table 10-8. Firmware Option Category

Use Latest Firmware	Check to use the latest firmware. Uncheck to select the firmware version.
Firmware File	Click in the right-hand text box to search for a firmware file (.jam) to associate with the debugger.

10.2.9 Clock

Set the option to use the fast internal RC (FRC) clock for the selected device.

Table 10-9. Clock Option Category

Use FRC in Debug mode (dsPIC33F and PIC24F/H devices only)	When debugging, use the device fast internal RC (FRC) for clocking instead of the oscillator specified for the application. This is useful when the application clock is slow. Checking this check box will let the application run at the slow speed but debug at the faster FRC speed. Reprogram after changing this setting. Note: Peripherals that are not frozen will operate at the FRC speed while debugging.
------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

10.2.10 Communication

Set the option(s) to use for your device and type of target communication.

Table 10-10. Communication Option Category

Interface	Select the interface from the available options.
Speed (MHz)	Enter a speed based on the available range for the interface.

High Voltage Activation Mode	<i>This option displays only for AVR devices with this option.</i> No High Voltage - Default setting. Simple High Voltage Pulse - The tool will try to activate the interface by issuing a high voltage pulse. This procedure is safe if the pin is configured as an input. User Power Toggle - In this mode the user will be prompted to toggle power on the target device. Once the tool detects that the power returns it will issue a high voltage pulse before the target device pin is configured, making the activation procedure as gentle as possible.
------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Programming AVR Devices with UPDI

MPLAB PICKit 4 supports using the high-voltage mechanism to activate the AVR® Unified Program and Debug Interface (UPDI). On low pin count AVR devices with UPDI, the UPDI pin can be configured as GPIO or RESET by configuring the RSTPINCFG configuration bits. To do further programming, the MPLAB PICKit 4 will have to use a high voltage pulse to reactivate the UPDI interface. When using the high voltage pulse, you must make sure that all circuits connected to the UPDI wire can tolerate a pulse of at least 12V.

GPIO vs. UPDI Operation:

When using a high voltage pulse to reactivate the UPDI interface, the reactivation is only temporary, but it will retain the UPDI functionality until the next reset. After the next reset, the pin will go back to the configuration as specified by the RSTPINCFG configuration bits. To have the pin configured as UPDI after a reset, the user will have to change the RSTPINCFG configuration bits back to UPDI.

It is possible to perform a debug session when the RSTPINCFG is configured to GPIO, but the pin will be temporarily configured as UPDI, and the pin will not operate as a GPIO pin.

Table 10-11. SYSCFG0 RSTPINCFG[1:0] Configuration Bits

Values	Function
0x0	GPIO
0x1	UPDI
0x2	RESET
0x3	Reserved

11. Hardware Specification

The hardware and electrical specifications of the MPLAB PICKit 4 In-Circuit Debugger system are detailed in this section.

11.1 USB Connector

The MPLAB PICKit 4 In-Circuit Debugger is connected to the host computer via a Micro-B USB connector, version 2.0 compliant. The Micro-B USB connector is located on the top of the debugger.

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 50 mA of power from the USB to function in all operational modes (debugger/programmer).

Note: The MPLAB PICKit 4 In-Circuit Debugger is powered through its Micro-B USB connector. The target board is powered from its own supply. Alternatively, the debugger can power the target board only if the target consumes less than 50 mA.

Cable Length – The computer-to-debugger cable, shipped with the debugger kit, is the correct length for proper operation.

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

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

11.2 MPLAB PICKit 4 In-Circuit Debugger

The debugger consists of an internal main board and an external Micro-B USB connector and an 8-pin SIL connector. On the front of the debugger enclosure is an indicator light strip and a hidden push button located underneath the logo.

Figure 11-1. MPLAB® PICKit™ 4 In-Circuit Debugger



1. Lanyard Connection - An opening through the top and side for a lanyard (not included) to be attached.
2. Emergency Recovery Button - If needed, this recessed button is used for Recovery Boot Mode.
3. Micro-B USB Connector - Used to connect the debugger to the computer with the supplied USB cable.
4. Indicator Light Strip - Displays the operational modes of the debugger (see [11.2.2 Indicator Lights Strip](#)).
5. Button Area - The area in the center of the shield logo is used for the Programmer-To-Go¹ option and for invoking the bootloader mode (see [6.3.2 How to Invoke the Bootload Mode](#)).
6. Pin 1 Marker - This designates the pin 1 location for proper connector alignment.
7. Programming Connector - The connector is an 8-pin SIL header (0.100" spacing) that connects to the target device (see [11.3.2 Pinouts for Interfaces](#)).
8. MicroSDHC Card Slot¹ - The microSDHC card slot supports a large variety of microSDHC cards with various speed requirements.

Note: ¹ The functionality will be available in a future firmware update of the product through MPLAB X IDE.

11.2.1 Main Board

The main board includes the following features:

- A 32-bit microcontroller using an Arm® Cortex®-M4 core.
- A USB 2.0 interface capable of USB speeds of 480 Mbps.
- An SRAM for holding the program code image. This image is used for programming on-board Flash device.
- One LED strip.

11.2.2 Indicator Lights Strip

The expected start-up sequence for the MPLAB PICKit 4 debugger is:

1. Purple - steady on for approximately 4 seconds.
2. Blue - steady on. The debugger is ready.

The indicator light strip has the following significance.

Table 11-1. Typical Light Strip Descriptions

Color	Description
Blue	Power is connected; debugger in standby.
Orange	Power target circuit from PICkit 4 checked.
Green	Power target circuit from PICkit 4 unchecked.
Red	Lit when the debugger has failed.

The following tables provide descriptions of the indicator lights and bootloader errors.

Table 11-2. Additional Light Strip Descriptions

Color	Description
Blue	Power is connected; debugger in standby.
Orange	Power target circuit from PICkit 4 checked (see Table 10-5).
Green	Power target circuit from PICkit 4 unchecked (see Table 10-5).
Purple	Bootloader is running.
Yellow	Debugger is busy.
Red	An operation has failed.
Purple	Fast blink indicates the time window for forcing the debugger into Bootload mode.

Table 11-3. Bootloader Error Descriptions

Bootloader Errors	Description
Red, slow blink	Power accessing the debugger's serial EEPROM.
Red, fast blink	Bootloader API commands cannot be processed.
White, fast blink	A runtime exception occurred in the tool firmware.

11.3 Communication Hardware

For standard debugger communication with a target (see [3.1 Debugger to Target Communication](#) and [3.1.1 Standard ICSP Device Communication](#)), either connect the debugger directly to the target or use a header if needed. The debugger has an 8-pin SIL connector. If the target has a 6-pin connector, make sure to align the Pin 1 appropriately.

11.3.1 Standard Communication

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

The V_{PP} high-voltage lines can produce a variable voltage that can swing from 0-14V to satisfy the voltage requirements of the specific emulation processor.

The V_{DD} sense connection draws very little current from the target processor. The actual power comes from the MPLAB PICKit 4 In-Circuit Debugger system, as the V_{DD} sense line is used as a reference only to track the target voltage.

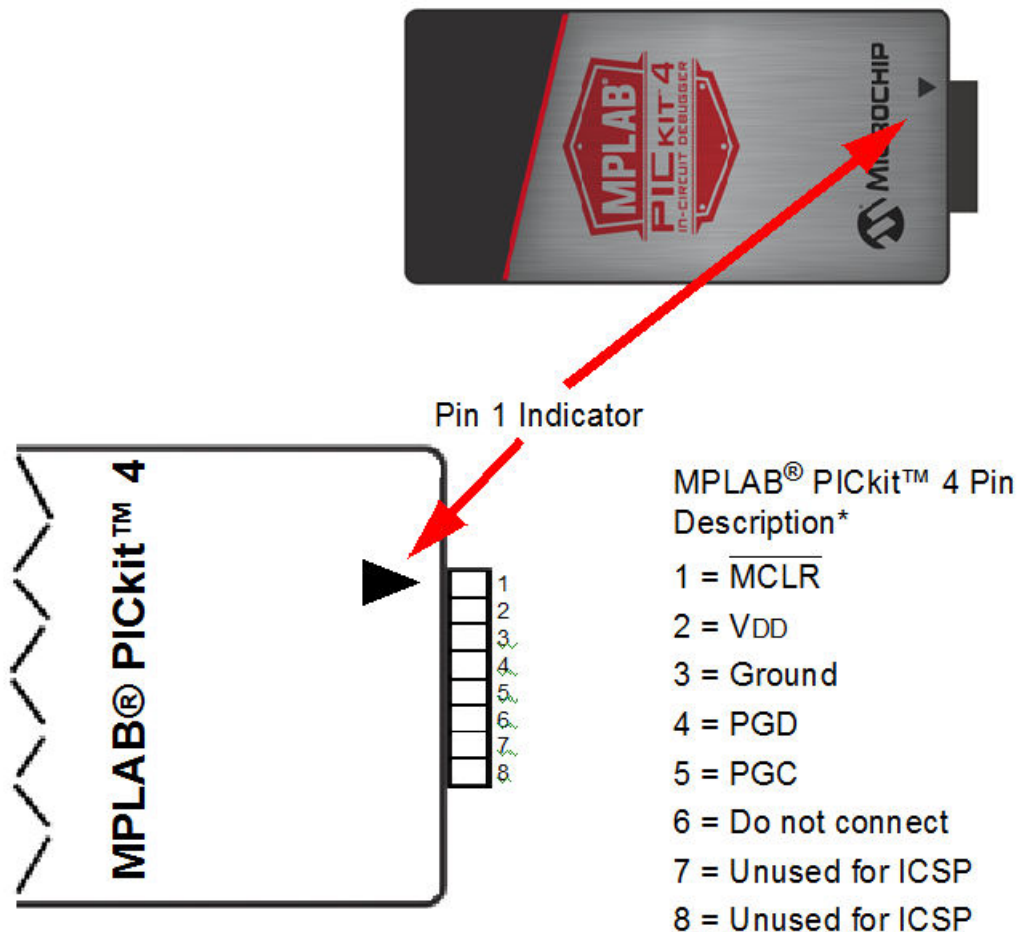
The clock and data connections are interfaces with the following characteristics:

- Clock and data signals are in high-impedance mode (even when no power is applied to the MPLAB PICKit 4 In-Circuit Debugger system).
- Clock and data signals are protected from high voltages caused by faulty target systems, or improper connections.
- Clock and data signals are protected from high current caused from electrical shorts in faulty target systems.

Table 11-4. Electrical Logic Table

Logic Inputs	$V_{IH} = V_{DD} \times 0.7V$ (min.)			
	$V_{IL} = V_{DD} \times 0.3V$ (max.)			
Logic Outputs	$V_{DD} = 5V$	$V_{DD} = 3V$	$V_{DD} = 2.3V$	$V_{DD} = 1.4V$
	$V_{OH} = 3.8V$ min.	$V_{OH} = 2.4V$ min.	$V_{OH} = 1.9V$ min.	$V_{OH} = 1.0V$ min.
	$V_{OL} = 0.55V$ max.	$V_{OL} = 0.55V$ max.	$V_{OL} = 0.3V$ max.	$V_{OL} = 0.1V$ max.

Figure 11-2. MPLAB® PICKit™ 4 Debugger Connector Pinout



11.3.2 Pinouts for Interfaces

The programming connector pin functions are different for various devices and interfaces. Refer to the following pinout tables for debug and data stream interfaces.

Note: Refer to the data sheet for the device you are using as well as the application notes for the specific interface for additional information and diagrams.

Table 11-5. Pinouts for Debug Interfaces


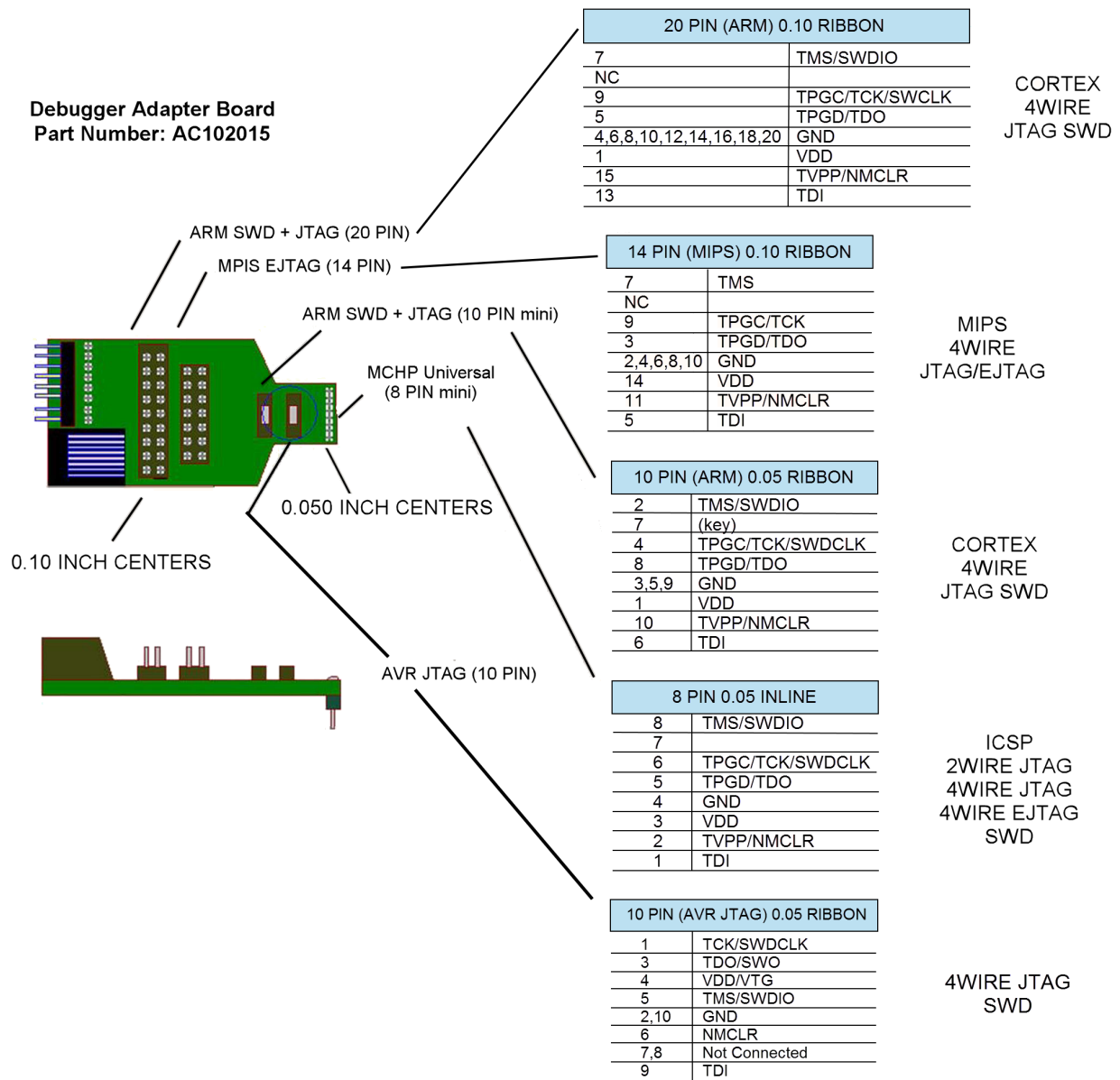
MPLAB PICKIT 4			DEBUG									
Connector	Pin #	Pin Name	ICSP (MCHP)	MIPS EJTAG	CORTEX® SWD	AVR® JTAG	AVR ISP(&DW)	UPDI	PDI	AW	DW(IRE)	TPI
	1	TVPP	MCLR	MCLR	MCLR							
	2	TVDD	VDD	VIO_REF	VTG	VTG	VTG	VTG	VTG	VTG	VTG	VTG
	3	GND	GND	GND	GND	GND	GND	GND	GND	GND	GND	GND
	4	PGD	DAT	TDO	SWO	TDO	MISO	DAT	DAT	DATA		DAT
	5	PGC	CLK	TCK	SWCLK	TCK	SCK					CLK
	6	TAUX	AUX			RESET	RESET		CLK		dW	RST
	7	TTDI		TDI		TDI	MOSI					
	8	TTMS		TMS	SWDIO	TMS						

Table 11-6. Pinouts for Data Stream Interfaces

MPLAB® PICKIT™ 4		DATA STREAM	
Pin #		DMCI / DGI U(S)ART / CDC	DGI SPI
1			
2		VTG	
3		GND	
4			MISO
5			SCK
6		(SCK)	
7		TX	MOSI
8		RX	SS

The following figure shows the debugger adapter board (AC102015) pinouts.

Figure 11-3. Debugger Adapter Board Pinouts



11.4 Target Board Considerations

The target board should be powered according to the requirements of the selected device and the application.

Note: Stresses above those listed under "Absolute Maximum Ratings" in the Electrical Characteristics chapter of the device's data sheet may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions, above those indicated in the operation listings of this specification, is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

The debugger does sense target voltage.

Depending on the type of debugger-to-target communication that is used, there are some considerations for target board circuitry:

- [3.2.2 Target Connection Circuitry](#)

- [3.2.5 Circuits That Will Prevent the Debugger From Functioning](#)

12. Revision History

12.1 Revision A (May 2018)

Initial release of this document.

12.2 Revision B (August 2018)

- Added information on the Debug Adapter Board in [2.3 MPLAB PICkit 4 In-Circuit Debugger Components](#).
- Expanded [Table 11-5](#) table to include pinouts for additional interfaces.
- Added a note to refer to device data sheets and application notes for debug interfaces in [11.3.2 Pinouts for Interfaces](#) and [3.2 Target Communication Connections](#).

12.3 Revision C (October 2018)

- Updated diagram in [Figure 11-2](#).
- Updated information in [Table 11-5](#).
- Added [Figure 11-3](#) to provide pinout information for the debugger adapter board (AC102015) for use with the PICkit 4 and various interfaces.

12.4 Revision D (January 2020)

- Added [10.2.6 Programmer-To-Go](#) section and updated references to Programmer-To-Go throughout document.
- Updated diagram of MPLAB PICkit 4.
- Added [9. Engineering Technical Notes \(ETNs\)](#) section.
- Modified target voltage values.

13. Support

13.1 Warranty Registration

If your development tool package includes a Warranty Registration Card, please complete the card and mail it in promptly. Sending in your Warranty Registration Card entitles you to receive new product updates. Interim software releases are available at the Microchip web site.

13.2 myMicrochip Personalized Notification Service

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

To begin the registration process and select your preferences to receive personalized notifications, go to:

<https://www.microchip.com/pcn>

A FAQ and registration details are available on the page, which can be opened by clicking the link above.

When you are selecting your preferences, choosing "Development Systems" will populate the list with available development tools. The main categories of tools are listed below:

- **Compilers** – The latest information on Microchip C compilers, assemblers, linkers and other language tools. These include all MPLAB C compilers; all MPLAB assemblers (including MPASM™ assembler); all MPLAB linkers (including MPLINK™ object linker); and all MPLAB librarians (including MPLIB™ object librarian).
- **Emulators** – The latest information on the MPLAB REAL ICE™ emulator.
- **In-Circuit Debuggers** – The latest information on Microchip in-circuit debuggers. These include the MPLAB ICD 3 and MPLAB ICD 4 in-circuit debuggers and PICKit™ 3 and MPLAB PICKit 4 in-circuit debuggers.
- **MPLAB® X IDE** – The latest information on Microchip MPLAB X IDE, the multi-platform (Windows®, Mac OS®, Linux®) Integrated Development Environment for development systems tools.
- **Programmers** – The latest information on Microchip programmers. These include the device (production) programmers MPLAB REAL ICE in-circuit emulator, MPLAB ICD 4 in-circuit debugger, MPLAB PICKit 4 in-circuit debugger, MPLAB PM3 and development (non-production) programmers PICKit 3.
- **Starter/Demo Boards** – These include MPLAB Starter Kit boards, PICDEM demo boards, and various other evaluation boards.

14. Glossary

Absolute Section

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

Absolute Variable/Function

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

Access Memory

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

Access Entry Points

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

Address

A value that identifies a location in memory.

Alphabetic Character

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

Alphanumeric

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

ANDed Breakpoints

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

Anonymous Structure

16-bit C Compiler – An unnamed structure.

PIC18 C Compiler – An unnamed structure that is a member of a C union. The members of an anonymous structure may be accessed as if they were members of the enclosing union. For example, in the following code, hi and lo are members of an anonymous structure inside the union caster.

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

ANSI

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

Application

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

Archive/Archiver

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

ASCII

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

Assembly/Assembler

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

Assigned Section

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

Asynchronously

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

Asynchronous Stimulus

Data generated to simulate external inputs to a simulator device.

Attribute

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

Attribute, Section

GCC Characteristics of sections, such as “executable”, “read-only”, 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.

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

C/C++

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

Calibration Memory

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

Central Processing Unit

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

Clean

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

COFF

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

Command Line Interface

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

Compiled Stack

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

Compiler

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

Conditional Assembly

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

Conditional Compilation

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

Configuration Bits

Special-purpose bits programmed to set PIC MCU and dsPIC DSC 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.

Data Monitor and Control Interface (DMCI)

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

Debug/Debugger

See *ICE/ICD*.

Debugging Information

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

Deprecated Features

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

Device Programmer

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

Digital Signal Controller

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

Digital Signal Processing\Digital Signal Processor

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

Directives

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

Download

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

DWARF

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

EEPROM

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

ELF

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

Emulation/Emulator

See *ICE/ICD*.

Endianness

The ordering of bytes in a multi-byte object.

Environment

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

Epilogue

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

EPROM

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

Error/Error File

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

Event

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

Executable Code

Software that is ready to be loaded for execution.

Export

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

Expressions

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

Extended Microcontroller Mode

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

Extended Mode (PIC18 MCUs)

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

External Label

A label that has external linkage.

External Linkage

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

External Symbol

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

External Symbol Resolution

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

External Input Line

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

External RAM

Off-chip Read/Write memory.

Fatal Error

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

File Registers

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

Filter

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

Fixup

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

Flash

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

FNOP

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

Frame Pointer

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

Free-Standing

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

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 run-time.

Hex Code/Hex File

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

Hexadecimal

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

High Level Language

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

ICE/ICD

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

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

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

ICSP

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

IDE

Integrated Development Environment, as in MPLAB IDE/MPLAB X IDE.

Identifier

A function or variable name.

IEEE

Institute of Electrical and Electronics Engineers.

Import

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

Initialized Data

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

```
int myVar=5;
```

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

Instruction Set

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

Instructions

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

Internal Linkage

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

International Organization for Standardization

An organization that sets standards in many businesses and technologies, including computing and communications. Also known as ISO.

Interrupt

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

Interrupt Handler

A routine that processes special code when an interrupt occurs.

Interrupt Service Request (IRQ)

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

Interrupt Service Routine (ISR)

Language tools: A function that handles an interrupt.

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

Interrupt Vector

Address of an interrupt service routine or interrupt handler.

L-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.

Library/Librarian

See *Archive/Archiver*.

Linker

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

Linker Script Files

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

Listing Directives

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

Listing File

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

Little Endian

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

Local Label

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

Logic Probes

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

Loop-Back Test Board

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

LVDS

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

With standard I/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/MPLAB X IDE, i.e., with a make.

Make Project

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

MCU

Microcontroller Unit. An abbreviation for microcontroller. Also uC.

Memory Model

For C compilers, a representation of the memory available to the application. For the PIC18 C compiler, a description that specifies the size of pointers that point to program memory.

Message

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

Microcontroller

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

Microcontroller Mode

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

Microprocessor Mode

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

Mnemonics

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

Module

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

MPASM™ Assembler

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

MPLAB Language Tool for Device

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

MPLAB ICD

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

MPLAB IDE/MPLAB X IDE

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

MPLAB PM3

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

MPLAB REAL ICE™ In-Circuit Emulator

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

MPLAB SIM

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

MPLAB Starter Kit for Device

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

MPLIB™ Object Librarian

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

MPLINK™ Object Linker

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

MRU

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

Native Data Size

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

Nesting Depth

The maximum level to which macros can include other macros.

Node

MPLAB IDE/MPLAB X IDE project component.

Non-Extended Mode (PIC18 MCUs)

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

Non Real Time

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

Non-Volatile Storage

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

NOP

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

Object Code/Object File

Object code is the machine code generated by an assembler or compiler. An object file is a file containing machine code and possibly debug information. It may be immediately executable or it may be relocatable, requiring linking with other object files, e.g., libraries, to produce a complete executable program.

Object File Directives

Directives that are used only when creating an object file.

Octal

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

Off-Chip Memory

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

Opcodes

Operational Codes. See *Mnemonics*.

Operators

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

OTP

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

Pass Counter

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

PC

Personal Computer or Program Counter.

PC Host

Any PC running a supported Windows operating system.

Persistent Data

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

Phantom Byte

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

PIC MCUs

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

PICKit 2 and 3

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

Plug-ins

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

Pod

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

Power-on-Reset Emulation

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

Pragma

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

Precedence

Rules that define the order of evaluation in expressions.

Production Programmer

A production programmer is a programming tool that has resources designed in to program devices rapidly. It has the capability to program at various voltage levels and completely adheres to the programming specification.

Programming a device as fast as possible is of prime importance in a production environment where time is of the essence as the application circuit moves through the assembly line.

Profile

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

Program Counter

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

Program Counter Unit

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

Program Memory

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

16-bit assembler/compiler: The memory area in a device where instructions are stored.

Project

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

Prologue

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

Prototype System

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

Psect

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

PWM Signals

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

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.

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.

Re-entrant

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 XC16 currently knows how to relax a CALL instruction into an RCALL instruction. This is done when the symbol that is being called is within +/- 32k instruction words from the current instruction.

Relocatable

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

Relocatable Section

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

Relocation

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

ROM

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

Run

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

Run-time Model

Describes the use of target architecture resources.

Run-time Watch

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

Scenario

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

Section

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

Section Attribute

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

Sequenced Breakpoints

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

Serialized Quick Turn Programming

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

Shell

The MPASM assembler shell is a prompted input interface to the macro assembler. There are two MPASM assembler shells: one for the DOS version and one for the Windows operating system 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/MPLAB X IDE updates register windows, watch variables, and status displays so you can analyze and debug instruction execution. You can also single step C compiler source code, but instead of executing single instructions, MPLAB IDE/MPLAB X IDE will execute all assembly level instructions generated by the line of the high level C statement.

Skew

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

Skid

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

Source Code

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

Source File

An ASCII text file containing source code.

Special Function Registers (SFRs)

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

SQTP

See *Serialized Quick Turn Programming*.

Stack, Hardware

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

Stack, Software

Memory used by an application for storing return addresses, function parameters, and local variables. This memory is dynamically allocated at run-time by instructions in the program. It allows for re-entrant function calls.

Stack, Compiled

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

Static RAM or SRAM

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

Status Bar

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

Step Into

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

Step Over

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

Step Out

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

Stimulus

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

Stopwatch

A counter for measuring execution cycles.

Storage Class

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

Storage Qualifier

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

Symbol

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

Symbol, Absolute

Description

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.

Term

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

Toolbar

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

Trace

An emulator or simulator function that logs program execution. The emulator logs program execution into its trace buffer which is uploaded to the MPLAB IDE/MPLAB X IDE 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 computer 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.

Volatile

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

Warning

Warning

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

16-bit assembler/compiler: Warnings report conditions that may indicate a problem, but do not halt processing.

Watch Variable

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

Watches Window

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

Watchdog Timer (WDT)

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

Workbook

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

The Microchip Web Site

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

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

Customer Change Notification Service

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

To register, access the Microchip web site at <http://www.microchip.com/>. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

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

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

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

Microchip Devices Code Protection Feature

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.

Legal Notice

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 unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Helder, JukeBlox, KeeLoq, Klear, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KlearNet, KlearNet logo, memBrain, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

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

GestIC is a registered trademark 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.

© 2018, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-5532-5

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, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
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	Australia - Sydney Tel: 61-2-9868-6733 China - Beijing Tel: 86-10-8569-7000 China - Chengdu Tel: 86-28-8665-5511 China - Chongqing Tel: 86-23-8980-9588 China - Dongguan Tel: 86-769-8702-9880 China - Guangzhou Tel: 86-20-8755-8029 China - Hangzhou Tel: 86-571-8792-8115 China - Hong Kong SAR Tel: 852-2943-5100 China - Nanjing Tel: 86-25-8473-2460 China - Qingdao Tel: 86-532-8502-7355 China - Shanghai Tel: 86-21-3326-8000 China - Shenyang Tel: 86-24-2334-2829 China - Shenzhen Tel: 86-755-8864-2200 China - Suzhou Tel: 86-186-6233-1526 China - Wuhan Tel: 86-27-5980-5300 China - Xian Tel: 86-29-8833-7252 China - Xiamen Tel: 86-592-2388138 China - Zhuhai Tel: 86-756-3210040	India - Bangalore Tel: 91-80-3090-4444 India - New Delhi Tel: 91-11-4160-8631 India - Pune Tel: 91-20-4121-0141 Japan - Osaka Tel: 81-6-6152-7160 Japan - Tokyo Tel: 81-3-6880-3770 Korea - Daegu Tel: 82-53-744-4301 Korea - Seoul Tel: 82-2-554-7200 Malaysia - Kuala Lumpur Tel: 60-3-7651-7906 Malaysia - Penang Tel: 60-4-227-8870 Philippines - Manila Tel: 63-2-634-9065 Singapore Tel: 65-6334-8870 Taiwan - Hsin Chu Tel: 886-3-577-8366 Taiwan - Kaohsiung Tel: 886-7-213-7830 Taiwan - Taipei Tel: 886-2-2508-8600 Thailand - Bangkok Tel: 66-2-694-1351 Vietnam - Ho Chi Minh Tel: 84-28-5448-2100	Austria - Wels Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 Denmark - Copenhagen Tel: 45-4450-2828 Fax: 45-4485-2829 Finland - Espoo Tel: 358-9-4520-820 France - Paris Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 Germany - Garching Tel: 49-8931-9700 Germany - Haan Tel: 49-2129-3766400 Germany - Heilbronn Tel: 49-7131-67-3636 Germany - Karlsruhe Tel: 49-721-625370 Germany - Munich Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 Germany - Rosenheim Tel: 49-8031-354-560 Israel - Ra'anana Tel: 972-9-744-7705 Italy - Milan Tel: 39-0331-742611 Fax: 39-0331-466781 Italy - Padova Tel: 39-049-7625286 Netherlands - Drunen Tel: 31-416-690399 Fax: 31-416-690340 Norway - Trondheim Tel: 47-72884388 Poland - Warsaw Tel: 48-22-3325737 Romania - Bucharest Tel: 40-21-407-87-50 Spain - Madrid Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 Sweden - Gothenberg Tel: 46-31-704-60-40 Sweden - Stockholm Tel: 46-8-5090-4654 UK - Wokingham Tel: 44-118-921-5800 Fax: 44-118-921-5820