



**MPLAB<sup>®</sup> REAL ICE<sup>™</sup>**  
**In-Circuit Emulator**  
**User's Guide**

---

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

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

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

---

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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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

**Trademarks**

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

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

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

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

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

© 2006, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

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

*Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona, Gresham, Oregon and Mountain View, California. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, 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.*



# MPLAB® REAL ICE™ IN-CIRCUIT EMULATOR USER'S GUIDE

---

---

## Table of Contents

---

---

<b>Preface</b> .....	<b>1</b>
<b>Chapter 1. Overview</b>	
1.1 Introduction .....	7
1.2 MPLAB REAL ICE In-Circuit Emulator Defined .....	7
1.3 How the MPLAB REAL ICE In-Circuit Emulator Helps You .....	7
1.4 MPLAB REAL ICE In-Circuit Emulator Kit Components .....	8
<b>Chapter 2. Operation</b>	
2.1 Introduction .....	9
2.2 MPLAB REAL ICE In-Circuit Emulator vs. MPLAB ICE 2000/4000 Emulators .....	9
2.3 MPLAB REAL ICE In-Circuit Emulator vs. MPLAB ICD 2 Debugger .....	9
2.4 System Configurations .....	10
2.5 Communication Connections .....	13
2.6 Debug Mode .....	17
2.7 Requirements For Debug Mode .....	17
2.8 Program Mode .....	19
2.9 Resources Used by the MPLAB REAL ICE In-Circuit Emulator .....	20
<b>Chapter 3. Installation</b>	
3.1 Introduction .....	21
3.2 Installing the Software .....	21
3.3 Installing the USB Device Drivers .....	21
3.4 Selecting Target Communications .....	21
3.5 Connecting the Logic Probes .....	22
3.6 Connecting and Powering the Emulator .....	22
3.7 Connecting and Powering the Target .....	22
<b>Chapter 4. General Setup</b>	
4.1 Introduction .....	23
4.2 Starting the MPLAB IDE Software .....	23
4.3 Creating a Project .....	24
4.4 Viewing the Project .....	24
4.5 Building the Project .....	24
4.6 Setting Configuration Bits .....	24
4.7 Setting the Emulator as the Debugger or Programmer .....	25
4.8 Settings Dialog .....	25

# MPLAB® REAL ICE™ In-Circuit Emulator User's Guide

---

## Chapter 5. Using the Emulator as a Debugger

5.1 Introduction .....	27
5.2 Debugger Overview .....	27
5.3 Breakpoints .....	27
5.4 Triggers .....	27
5.5 Trace .....	28
5.6 Debugging Functions .....	30
5.7 Debugging Dialogs/WIndows .....	32

## Chapter 6. Using the Emulator as a Programmer

6.1 Introduction .....	39
6.2 Programmer Overview .....	39
6.3 Programming Functions .....	39

## Chapter 7. Hardware Specification

7.1 Introduction .....	41
7.2 Highlights .....	41
7.3 Declaration of Conformity .....	41
7.4 USB Port/Power .....	42
7.5 Emulator Pod .....	42
7.6 Standard Communication Board .....	44
7.7 High-Speed Communication Boards .....	45
7.8 Other Emulator Boards (Future) .....	48
7.9 Target Board .....	48

## Appendix A. Revision History

A.1 Revision History .....	49
----------------------------	----

<b>Glossary .....</b>	<b>51</b>
-----------------------	-----------

<b>Index .....</b>	<b>65</b>
--------------------	-----------

<b>Worldwide Sales and Service .....</b>	<b>68</b>
--	-----------



# MPLAB® REAL ICE™ IN-CIRCUIT EMULATOR USER'S GUIDE

---

---

## Preface

---

---

### NOTICE TO CUSTOMERS

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

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

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

## INTRODUCTION

This chapter contains general information that will be useful to know before using the MPLAB REAL ICE in-circuit emulator. Items discussed include:

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

## DOCUMENT LAYOUT

This document describes how to use the MPLAB REAL ICE in-circuit emulator 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:

- **Chapter 1: Overview** – What the MPLAB REAL ICE in-circuit emulator is, and how it can help you develop your application.
- **Chapter 2: Operation** – The theory of MPLAB REAL ICE in-circuit emulator operation.
- **Chapter 3: Installation** – How to install the emulator software and hardware.
- **Chapter 4: General Setup** – How to get started using the emulator.
- **Chapter 5: Using the Emulator as a Debugger** – A description of emulator functions available in MPLAB IDE when the MPLAB REAL ICE in-circuit emulator is chosen as the debug tool.
- **Chapter 6: Using the Emulator as a Programmer** – A description of emulator functions available in MPLAB IDE when the MPLAB REAL ICE in-circuit emulator is chosen as the programming tool.

# MPLAB® REAL ICE™ In-Circuit Emulator User's Guide

- **Chapter 7: Hardware Specification** – The hardware and electrical specifications of the emulator system.

## CONVENTIONS USED IN THIS GUIDE

The following conventions may appear in this documentation:

### DOCUMENTATION CONVENTIONS

Description	Represents	Examples
<b>Arial font:</b>		
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</i></u> >Save
Bold characters	A dialog button	Click <b>OK</b>
	A tab	Click the <b>Power</b> tab
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
<b>Courier font:</b>		
Plain Courier	Sample source code	#define START
	Filenames	autoexec.bat
	File paths	c:\mcc18\h
	Keywords	_asm, _endasm, static
	Command-line options	-Opa+, -Opa-
	Bit values	0, 1
	Constants	0xFF, 'A'
Italic Courier	A variable argument	<i>file.o</i> , where <i>file</i> can be any valid filename
Square brackets [ ]	Optional arguments	mpasmwin [options] <i>file</i> [options]
Curly brackets and pipe character: {   }	Choice of mutually exclusive arguments; an OR selection	errorlevel {0 1}
Ellipses...	Replaces repeated text	var_name [, var_name...]
	Represents code supplied by user	void main (void) { ... }

## WARRANTY REGISTRATION

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

## RECOMMENDED READING

This document describes how to use the MPLAB REAL ICE in-circuit emulator. Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

### **Readme for MPLAB REAL ICE In-Circuit Emulator**

For the latest information on using the MPLAB REAL ICE in-circuit emulator, read the "Readme for MPLAB REAL ICE Emulator.txt" file (an ASCII text file) in the Readmes subdirectory of the MPLAB IDE installation directory. The Readme file contains update information and known issues that may not be included in this user's guide.

### **MPLAB REAL ICE In-Circuit Emulator Setup (DS51615)**

A mini-poster showing you how to hook up the hardware and install the software for the MPLAB REAL ICE in-circuit emulator using standard communications and the Explorer 16 demo board.

### **MPLAB REAL ICE In-Circuit Emulator On-line Help File**

A comprehensive help file for the emulator. Usage, troubleshooting and hardware specifications are included.

### **Header Board Specification (DS51292)**

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

### **Transition Socket Specification (DS51194)**

Consult this document for information on transition sockets available for use with MPLAB ICE 2000/4000 device adaptors, MPLAB ICD 2 headers and MPLAB REAL ICE in-circuit emulator headers.

## THE MICROCHIP WEB SITE

Microchip provides online support via our web site at [www.microchip.com](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 (FAQs), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

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

To register, access the Microchip web site at [www.microchip.com](http://www.microchip.com), click on Customer Change Notification and follow the registration instructions.

The Development Systems product group categories are:

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

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

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

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

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

# MPLAB® REAL ICE™ In-Circuit Emulator User's Guide

---

NOTES:



# MPLAB® REAL ICE™ IN-CIRCUIT EMULATOR USER'S GUIDE

---

---

## Chapter 1. Overview

---

---

### 1.1 INTRODUCTION

An overview of the MPLAB REAL ICE in-circuit emulator system is given.

- MPLAB REAL ICE In-Circuit Emulator Defined
- How the MPLAB REAL ICE In-Circuit Emulator Helps You
- MPLAB REAL ICE In-Circuit Emulator Kit Components

### 1.2 MPLAB REAL ICE IN-CIRCUIT EMULATOR DEFINED

MPLAB REAL ICE in-circuit emulator is an in-circuit emulator that is controlled by a PC running MPLAB IDE software on a Windows® platform. The MPLAB REAL ICE in-circuit emulator is an integral part of the development engineer's toolsuite. The application usage can vary from software development to hardware integration to manufacturing test to field service.

The MPLAB REAL ICE in-circuit emulator is a modern emulator system that supports hardware and software development for selected Microchip PICmicro® microcontrollers (MCUs) and dsPIC® Digital Signal Controllers (DSCs) that are based on In-Circuit Serial Programming™ (ICSP™) programming capability and Standard DUT Programming (STDP) 2-wire serial interfaces.

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

The MPLAB REAL ICE emulation concept was developed for emulating complex processors that differ from conventional system processors in several aspects:

- Processors run at maximum speeds
- Capability to incorporate I/O port data input
- Instrumented Trace (MPLAB IDE and Compiler Assisted)

In addition to emulator functions, the MPLAB REAL ICE in-circuit emulator system also may be used as a development programmer.

### 1.3 HOW THE MPLAB REAL ICE IN-CIRCUIT EMULATOR HELPS YOU

The MPLAB REAL ICE in-circuit emulator system allows you to:

- Debug your application on your own hardware in real time
- Debug with hardware breakpoints
- Debug with software breakpoints (future)
- Set breakpoints based on internal and/or external signals
- Monitor internal file registers
- Emulate full speed
- Program your device
- Trace lines of code or log variable/expression values

## 1.4 MPLAB REAL ICE IN-CIRCUIT EMULATOR KIT COMPONENTS

The components of the MPLAB REAL ICE in-circuit emulator system kit are listed below.

1. MPLAB IDE Quick Start Guide (DS51281)
2. CD-ROM with MPLAB IDE software and on-line documentation
3. Emulator pod
4. USB cable to provide communications between the emulator and a PC and to provide power to the emulator
5. Standard driver board (MPLAB ICD 2 compatible) and cable to connect the emulator pod to a header module or target board
6. Logic probes
7. Self-Test board

Additional hardware that may be ordered separately:

8. Processor Extension Pack: High-speed driver board, ICE header/receiver board and cables to connect the emulator pod to a target board
9. Performance Pack: High-speed driver board, high-speed receiver board and cables to connect the emulator pod to a target board
10. High-speed to standard converter board
11. Transition socket

---

---

## Chapter 2. Operation

---

---

### 2.1 INTRODUCTION

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

- MPLAB REAL ICE In-Circuit Emulator vs. MPLAB ICE 2000/4000 Emulators
- MPLAB REAL ICE In-Circuit Emulator vs. MPLAB ICD 2 Debugger
- System Configurations
- Communication Connections
- Debug Mode
- Requirements For Debug Mode
- Program Mode
- Resources Used by the MPLAB REAL ICE In-Circuit Emulator

### 2.2 MPLAB REAL ICE IN-CIRCUIT EMULATOR VS. MPLAB ICE 2000/4000 EMULATORS

The MPLAB REAL ICE in-circuit emulator system is a next generation In-Circuit Emulator (ICE) system. It differs from classical in-circuit emulator systems (e.g., MPLAB ICE 2000/4000) in a single, but important way: the production device and emulation device are the same. This means that the actual device/emulated device differences are all but eliminated. For example, speed bottlenecks caused by bringing internal busses off-chip and using external memories on classical emulator systems are eliminated by using the actual device for emulation.

Another significant benefit is that there is no time lag from when the device is released to when an emulator module to support the device can be released. If a header board is required, it can be developed to coincide with the device release, or lag it by a small amount, which is a great improvement over the longer processor module development times.

### 2.3 MPLAB REAL ICE IN-CIRCUIT EMULATOR VS. MPLAB ICD 2 DEBUGGER

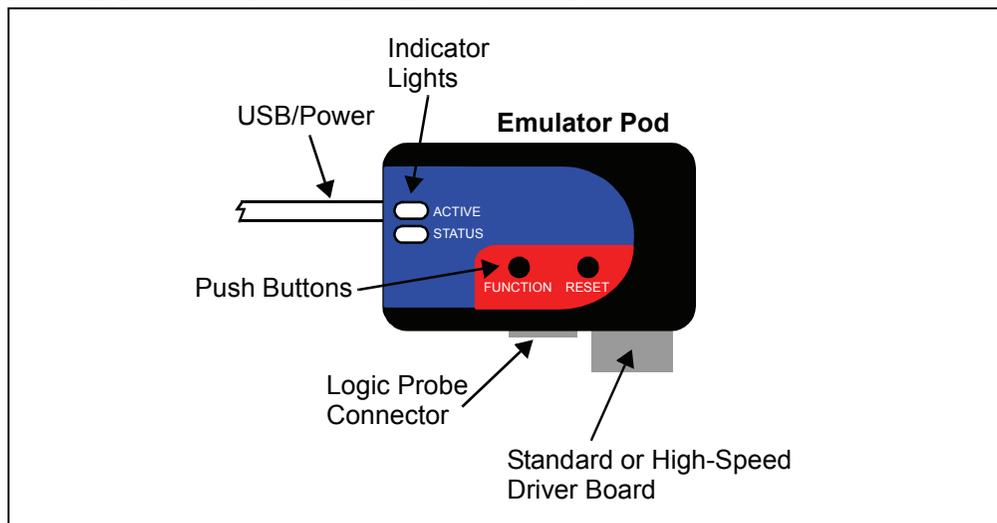
The MPLAB REAL ICE in-circuit emulator system is similar to the MPLAB ICD 2 in-circuit debugger system, but surpasses it in speed and functionality. Even with standard communication, the MPLAB REAL ICE in-circuit emulator is faster than the MPLAB ICD 2. With the high-speed communication option, it is much faster. Also, in addition to basic debug functions, the MPLAB REAL ICE in-circuit emulator incorporates “emulation” functions, such as trace.

## 2.4 SYSTEM CONFIGURATIONS

The MPLAB REAL ICE in-circuit emulator system consists of these basic items:

- Emulator pod with indicator lights, push buttons and a logic probe connector
- USB cable to connect a PC to the emulator pod and power the pod
- Driver board and modular cable(s) to connect the emulator pod to an ICE header or target board

**FIGURE 2-1: BASIC EMULATOR SYSTEM**



The emulator system configurations are discussed in the following sections.

### CAUTION

**Do not connect the hardware before installing the software and USB drivers, or whenever the pod or target is powered.**

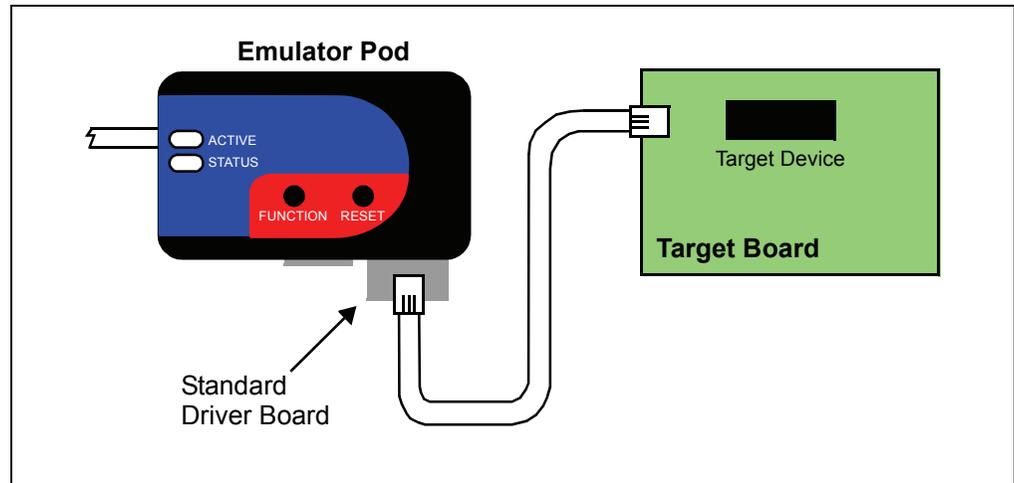
#### 2.4.1 Standard Communication

The emulator system can be configured to use standard communication for both programming and debugging functions. This 6-pin connection is the same one used by the MPLAB ICD 2 in-circuit debugger, and provides the same amount of functionality.

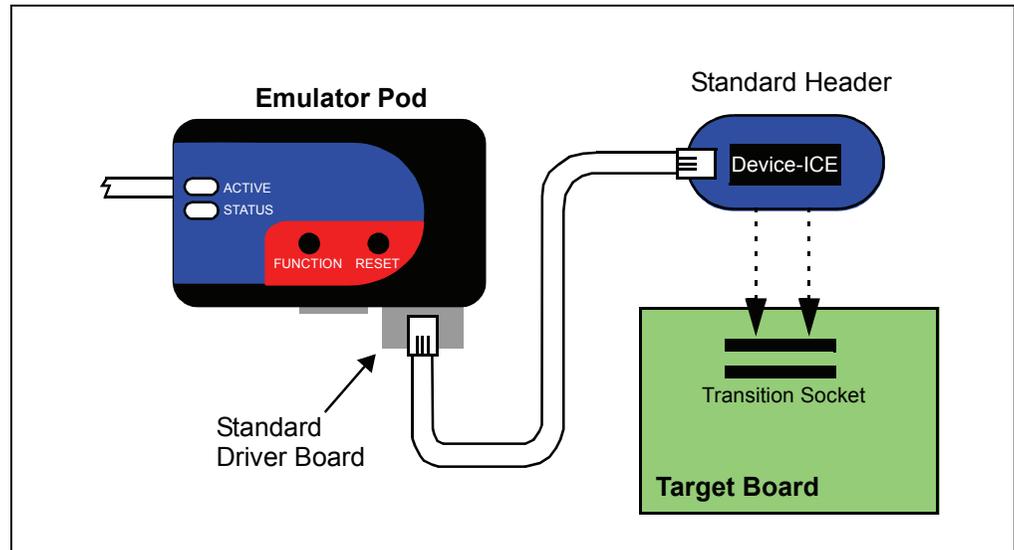
The standard driver board is plugged into the emulator pod to configure the system for this type of communication with the target. The modular cable can be either (1) inserted into a matching socket at the target, where the target device is on the target board (Figure 2-2), or (2) inserted into a header board, which is then plugged into the target board (Figure 2-3).

For more on standard communication, see **Chapter 7. "Hardware Specification"**.

**FIGURE 2-2: STANDARD EMULATOR SYSTEM – DEVICE WITH ON-BOARD ICE CIRCUITRY**



**FIGURE 2-3: STANDARD EMULATOR SYSTEM – ICE DEVICE**



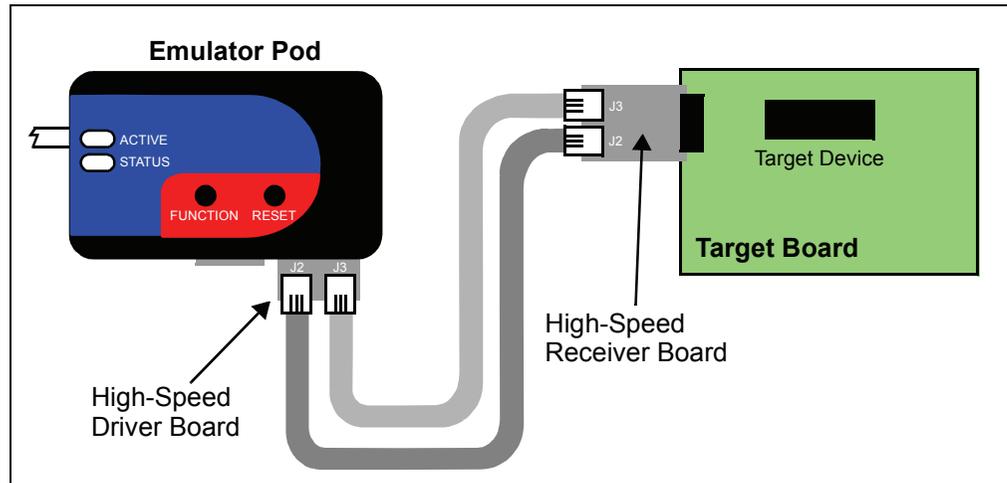
## 2.4.2 High-Speed Communication

The emulator system can be configured to use high-speed communication for both programming and debugging functions. This connection allows for high-speed operations, a longer distance between the emulator and target and additional tracing functionality over a standard connection.

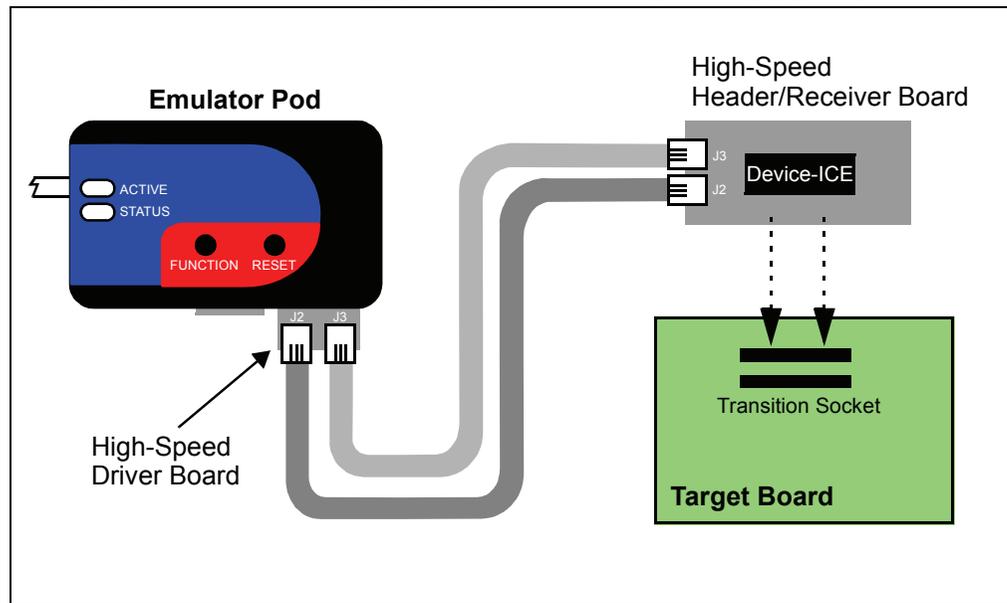
The high-speed driver board is plugged into the emulator pod to configure the system for this type of communication with the target. The modular cables can be inserted into matching sockets either (1) at the high-speed receiver board, which is plugged into the target board, with an on-board target device, via an 8-pin connector (Figure 2-4), or (2) at the header/receiver board, which is then plugged into the target board (Figure 2-5).

For more on high-speed communication, see **Chapter 7. "Hardware Specification"**.

**FIGURE 2-4: HIGH-SPEED EMULATOR SYSTEM – DEVICE WITH ON-BOARD ICE CIRCUITRY**



**FIGURE 2-5: HIGH-SPEED EMULATOR SYSTEM – ICE DEVICE**



## 2.4.3 Hybrid Communication (Future)

The emulator system can be configured to use high speed to standard communication via a converter. If you need to use high-speed to facilitate emulator/target communication over a large distance, but do not need the additional debug features, then this configuration can be useful.

To configure the system for this type of communication, plug the high-speed driver board into the emulator and then attach the modular cables. Plug these cables into the sockets on the converter board. Plug one end of the standard cable into the socket on the converter board. Plug the other end of the standard cable into either the target board or header board, as shown in **Section 2.4.1 “Standard Communication”**.

## 2.5 COMMUNICATION CONNECTIONS

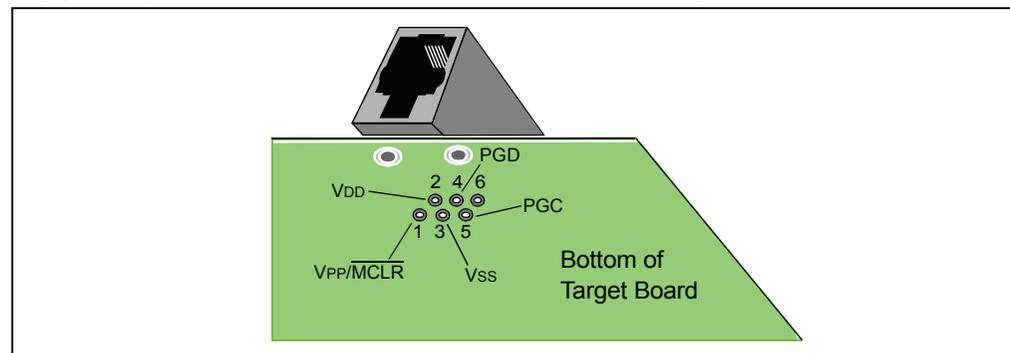
There are two driver boards available to closely match most application requirements. The standard driver board (MPLAB ICD 2 compatible) can be used to connect to the myriad of demo boards and applications that contain the RJ11 connector. The high-speed driver/receiver board combination can be used for high-speed applications, for additional trace features, for large (several feet) emulator-to-target distances and for noisy environments.

### 2.5.1 Standard Communications Target Connection

Using the standard driver board, the MPLAB REAL ICE in-circuit emulator is connected to the target device with the modular interface (six-conductor) cable. The pin numbering for the connector is shown from the bottom of the target PC board in Figure 2-6.

**Note:** Cable connections at the emulator and target are mirror images of each other, i.e., pin 1 on one end of the cable is connected to pin 1 on the other end of the cable.

**FIGURE 2-6: STANDARD CONNECTION AT TARGET**

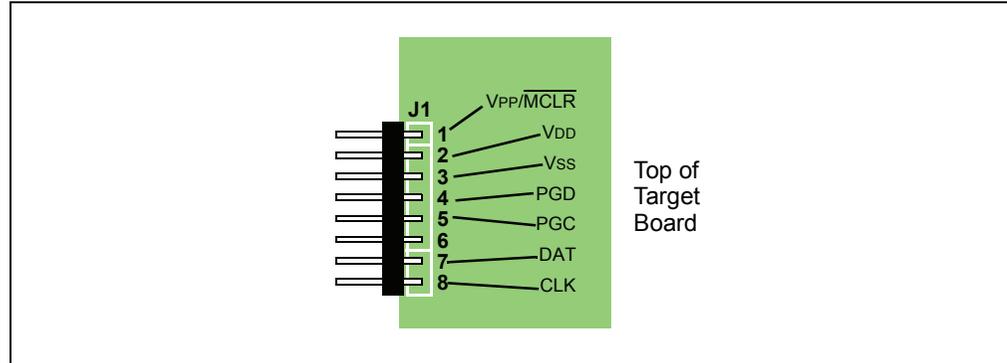


## 2.5.2 High-Speed Communications Target Connection

Using the high-speed driver/receiver board combination, the MPLAB REAL ICE in-circuit emulator is connected to the target device with an 8-pin interface. The pin numbering for the connector is shown from the top of the target PC board in Figure 2-7.

**Note:** Connections at the emulator and target are mirror images of each other, i.e., pin 1 on the receiver board is connected to pin 1 on the target board.

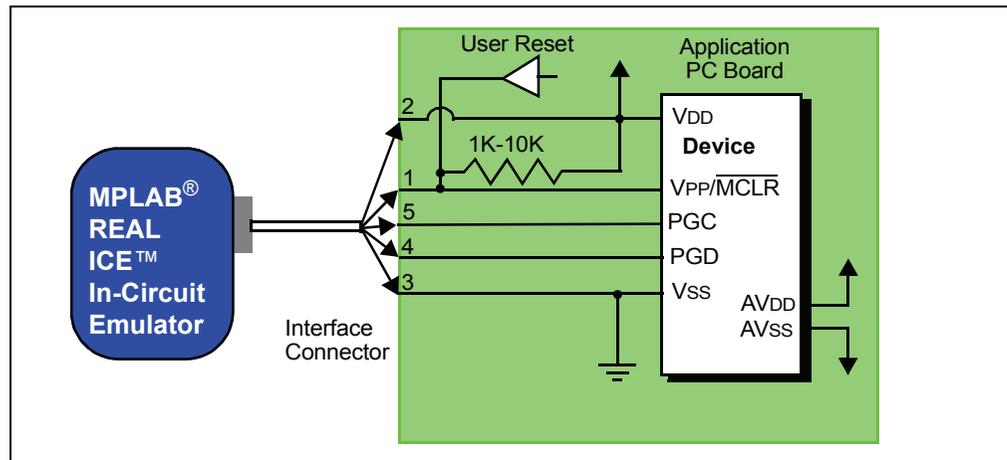
**FIGURE 2-7: HIGH-SPEED CONNECTION AT TARGET**



## 2.5.3 Target Connection Circuitry

Figure 2-8 shows the interconnections of the MPLAB REAL ICE in-circuit emulator to the connector on the target board. The diagram also shows the wiring from the connector to a device on the target PC board. A pull-up resistor (usually around 10 kΩ) is recommended to be connected from the VPP/MCLR line to VDD so that the line may be strobed low to reset the device.

**FIGURE 2-8: STANDARD CONNECTION TARGET CIRCUITRY**



In the following descriptions, only three lines are active and relevant to core emulator operation: pins 1 (VPP/MCLR), 5 (PGC) and 4 (PGD). Pins 2 (VDD) and 3 (VSS) are shown on the above diagram for completeness, but are only sensed, not provided or controlled, by the emulator.

Be aware that the target VDD is sensed by the emulator to allow level translation for target low-voltage operation. If the emulator does not sense voltage on its VDD line (pin 2 of the interface connector), it will not operate.

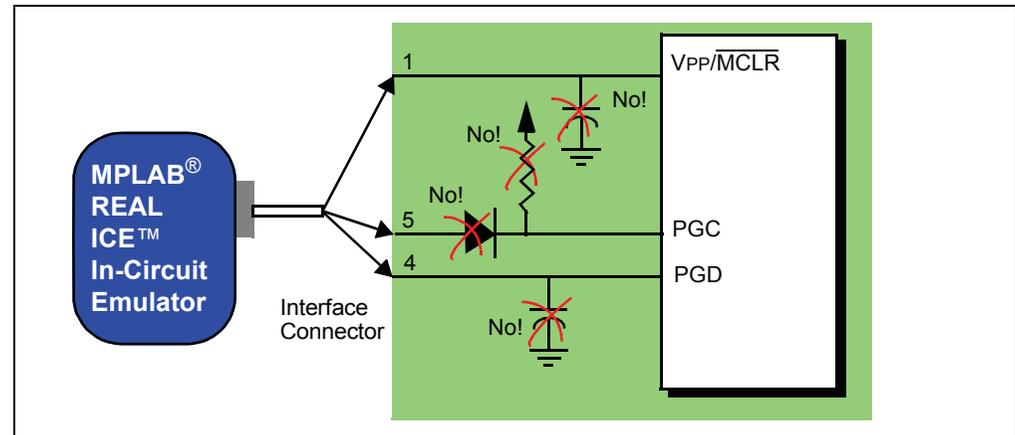
Not all devices have the AVDD and AVSS lines, but if they are present on the target device, all must be connected to the appropriate levels in order for the emulator to operate.

The interconnection is very simple; any problems experienced are often caused by other connections or components on these critical lines that interfere with the operation of the MPLAB REAL ICE in-circuit emulator system, as discussed in the next section.

## 2.5.4 Circuits That Will Prevent the Emulator From Functioning

Figure 2-9 shows the active emulator lines with some components that will prevent the MPLAB REAL ICE in-circuit emulator system from functioning.

**FIGURE 2-9: IMPROPER CIRCUIT COMPONENTS**



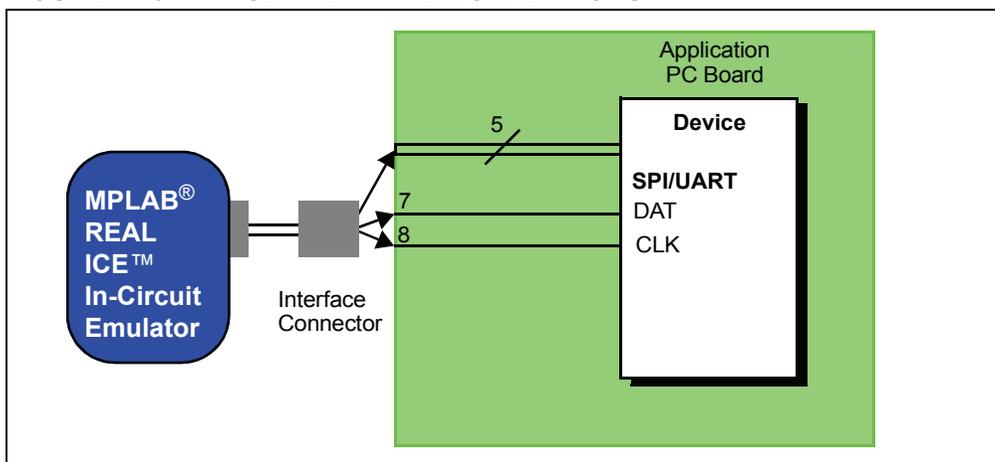
Specifically, these guidelines must be followed:

- No pull-ups on PGC/PGD – they will divide the voltage levels, since these lines have 4.7 k $\Omega$  pull-down resistors in the emulator.
- No capacitors on PGC/PGD – they will prevent fast transitions on data and clock lines during programming and debug communications.
- No capacitors on MCLR – they will prevent fast transitions of VPP. A simple pull-up resistor is generally sufficient.
- No diodes on PGC/PGD – they will prevent bidirectional communication between the emulator and the target device.

## 2.5.5 SPI/UART Trace Connections (Future)

When using high-speed communications, streaming serial trace is possible using the device SPI/UART and pins 7 (DAT) and 8 (CLK). Figure 2-10 shows these additional connections. As with pins 4 (PBD) and 5 (PGC) (**Section 2.5.4 “Circuits That Will Prevent the Emulator From Functioning”**), do not use pull-ups, capacitors or diodes.

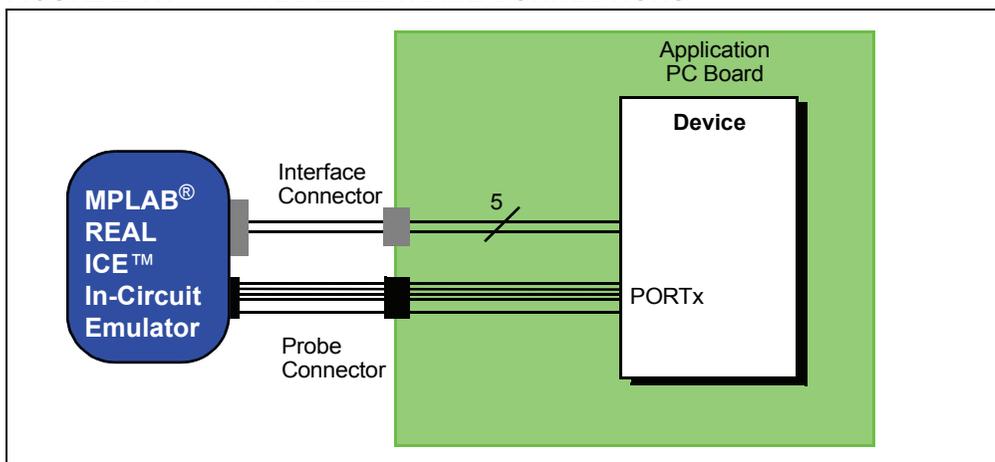
FIGURE 2-10: SERIAL TRACE CONNECTIONS



## 2.5.6 I/O Port Trace Connections

Streaming parallel trace is possible using a device I/O port and the emulator logic probe connector. This provides greater trace speed and data quantity, but limits emulator-to-target distance by the length of the parallel cable. Figure 2-11 shows these additional connections.

FIGURE 2-11: PARALLEL TRACE CONNECTIONS



For this trace configuration, seven (7) lines of data and one (1) line for clock are transmitted (see **Section 7.5.4 “Logic Probe/External Trigger Interface”**.)

As with pins 4 (PBD) and 5 (PGC) (**Section 2.5.4 “Circuits That Will Prevent the Emulator From Functioning”**), do not use pull-ups, capacitors or diodes. However, to use probe pins as inputs, you must provide the circuitry to drive them. Unused pins should either be pulled up or grounded. Floating pins may produce false triggers.

For more on this type of trace, see **Section 5.5.2 “I/O Port Trace”**.

## 2.6 DEBUG MODE

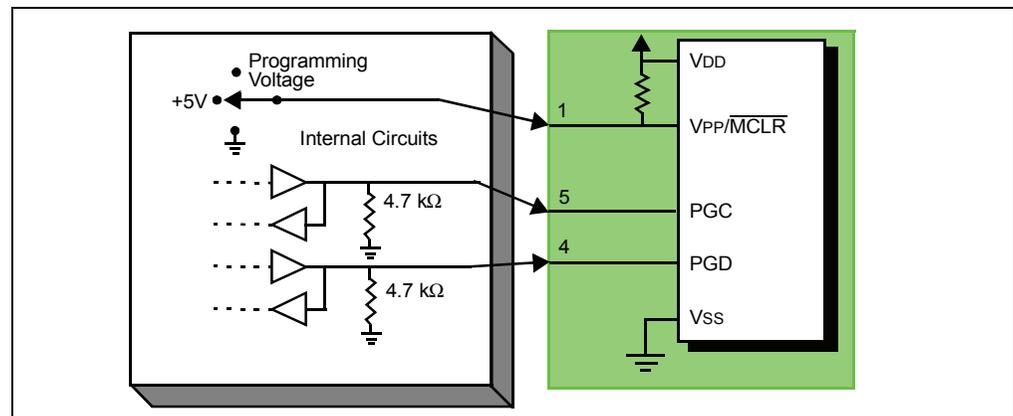
There are two steps to using the MPLAB REAL ICE in-circuit emulator system as a debugger. The first requires that an application be programmed into the target device. The second uses the internal in-circuit debug hardware of the target Flash device to run and test the application program. These two steps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

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

If the target device cannot be programmed correctly, the MPLAB REAL ICE in-circuit emulator will not be able to debug.

Figure 2-12 shows the basic interconnections required for programming. Note that this is the same as Figure 2-8, but for the sake of clarity, the VDD and VSS lines from the emulator are not shown.

**FIGURE 2-12: PROPER CONNECTIONS FOR PROGRAMMING**



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

## 2.7 REQUIREMENTS FOR DEBUG MODE

To debug (set breakpoints, see registers, etc.) with the MPLAB REAL ICE in-circuit emulator system, there are critical elements that must be working correctly:

- The emulator must be connected to a PC. It must be powered by the PC via the USB cable, and it must be communicating with MPLAB IDE software via the USB cable. See **Chapter 3. "Installation"** for details.
- The emulator must be connected as shown to the VPP, PGC and PGD pins of the target device with the modular interface cable (or equivalent). VSS and VDD are also required to be connected between the emulator and target device.
- The target device must have power and a functional, running oscillator. If the target device does not run, for whatever reason, the MPLAB REAL ICE in-circuit emulator cannot debug.

- The target device must have its configuration words programmed correctly:
  - The oscillator Configuration bits should correspond to RC, XT, etc., depending upon the target design.
  - For some devices, the Watchdog Timer is enabled by default and needs to be disabled.
  - The target device must not have code protection enabled.
  - The target device must not have table read protection enabled.

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

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

## 2.7.1 Sequence of Operations Leading to Debug Mode

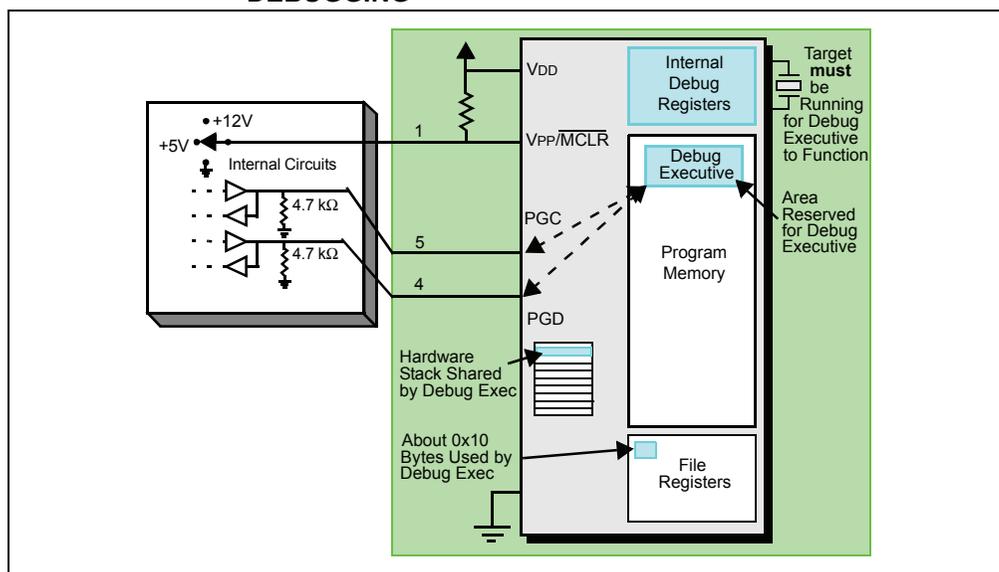
Given that the Requirements For Debug Mode are met, these actions can be performed when the MPLAB REAL ICE in-circuit emulator is set as the current debugger (*Debugger>Select Tool*):

- When *Debugger>Program* is selected, the application code is programmed into the device's memory via the ICSP protocol as described above.
- A small "debug executive" program is loaded into the high area of program memory of the target device. Since the debug executive must reside in program memory, the application program must not use this reserved space. The debug executive typically needs about 0x120 words of program memory. Some devices have special memory areas dedicated to the debug executive. Check your device data sheet for details.
- Special "in-circuit debug" registers in the target device are enabled. These allow the debug executive to be activated by the emulator.
- The target device is held in Reset by keeping the  $\overline{VPP/MCLR}$  line low.

## 2.7.2 Debug Mode Details

Figure 2-13 illustrates the MPLAB REAL ICE in-circuit emulator system when it is ready for debugging.

**FIGURE 2-13: MPLAB<sup>®</sup> REAL ICE<sup>™</sup> IN-CIRCUIT EMULATOR READY FOR DEBUGGING**



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

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

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

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

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

## 2.8 PROGRAM MODE

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

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

**Note:** A header board is required to debug some devices. These parts can be programmed without the header by connecting the VPP, PGC and PGD lines as described previously.

## 2.9 RESOURCES USED BY THE MPLAB REAL ICE IN-CIRCUIT EMULATOR

The MPLAB REAL ICE in-circuit emulator may use on-chip resources when debugging, depending on the device.

### 2.9.1 dsPIC DSC/PIC24 Devices

The emulator device (*Device-ICE*) and header board may be used with the emulator for debugging functions without loss of pins. For more information, see the "*Header Board Specification*" (DS51292).

#### General Resources Used

- $\overline{\text{MCLR}}/\text{VPP}$  shared for programming

#### Program/Data Memory Used

In MPLAB IDE, program memory and/or data memory (file register) displays marked with an "R" represent reserved registers.

Device	Program Memory Used	File Registers Used
dsPIC DSC/PIC24 ( <i>device-ICE</i> )	None*	0x800-0x822

\* No user program memory space used for debug.



## Chapter 3. Installation

### 3.1 INTRODUCTION

How to install the MPLAB REAL ICE in-circuit emulator system is discussed.

- Installing the Software
- Installing the USB Device Drivers
- Selecting Target Communications
- Connecting the Logic Probes
- Connecting and Powering the Emulator
- Connecting and Powering the Target

### 3.2 INSTALLING THE SOFTWARE

To install the MPLAB IDE software, first acquire the latest MPLAB IDE installation executable (*MPxxxxxx.exe*, where *xxxxxx* represents the version of MPLAB IDE) from either the Microchip web site ([www.microchip.com](http://www.microchip.com)) or the MPLAB IDE CD-ROM (DS51123). Then run the executable and follow the screens to install MPLAB IDE.

**Note:** MPLAB IDE v7.43 or greater is required for MPLAB REAL ICE in-circuit emulator functionality.

### 3.3 INSTALLING THE USB DEVICE DRIVERS

MPLAB REAL ICE in-circuit emulator USB device drivers must be installed before the emulator can use USB communications. Follow the html instructions found at the location below to install the drivers:

*MPLAB IDE installation directory\REAL ICE\Drivers\ddri.htm*

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

### 3.4 SELECTING TARGET COMMUNICATIONS

If you have not already done so, insert the desired driver board into the emulator pod and attach the cables. To change a driver board, unplug the USB and target power, remove the board, insert the other board, plug in the USB and power the target.

#### CAUTION

Neither the emulator nor target should be powered when inserting or removing a driver board or damage to the driver board could result.

A driver board is inserted into the pod to select the type of communication with the target, either standard (for header boards and many demo boards) or high speed (for target boards over six inches away from the emulator). See **Section 2.4 “System Configurations”** for more details.

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

**Note:** In the future, ICD header boards with ICD devices (*Device-ICD*) may be used, though only standard debug, and not emulator debug, functions will be available.

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

For regular devices, the emulator may be connected directly to the target by choosing the appropriate driver board for the connector on the target board. For a standard connector, choose the standard driver board and cable. For an 8-pin connector, choose the high-speed driver board and cables, connected to the high-speed receiver board, which is then plugged in to the 8-pin target connector.

The device on the target must have built-in debug circuitry in order for the MPLAB REAL ICE in-circuit emulator to perform emulation with it. Consult the device data sheet to see if the device has the needed debug circuitry, i.e., it should have a “Background Debugger Enable” Configuration bit.

**Note:** In the future, devices with circuitry that support ICD may be used, though only standard debug, and not emulator debug, functions will be available.

## 3.5 CONNECTING THE LOGIC PROBES

The logic probes may be connected into the logic probe connector on the emulator pod. These probes will allow halting the MPLAB REAL ICE in-circuit emulator by external triggers, and will provide output triggers to synchronize external equipment such as oscilloscopes and logic analyzers.

This connector can also be used for I/O Port trace. See **Section 2.5.6 “I/O Port Trace Connections”**.

## 3.6 CONNECTING AND POWERING THE EMULATOR

If you have not already done so, connect the emulator pod to the PC through a USB port using the provided cable. The USB connection provides communication between the emulator and PC, and power to the emulator.

The emulator pod contains the hardware necessary to perform the common emulator functions, such as trace, break and emulate.

## 3.7 CONNECTING AND POWERING THE TARGET

If you have not already done so, connect the emulator pod to the target using the appropriate cables for the driver board selected (see **Section 3.4 “Selecting Target Communications”**). Then power the target.

**Note:** The emulator cannot power the target.

---

---

## Chapter 4. General Setup

---

---

### 4.1 INTRODUCTION

How to get started using the MPLAB REAL ICE in-circuit emulator is discussed.

- Starting the MPLAB IDE Software
- Creating a Project
- Viewing the Project
- Building the Project
- Setting Configuration Bits
- Setting the Emulator as the Debugger or Programmer
- Settings Dialog

### 4.2 STARTING THE MPLAB IDE SOFTWARE

After installing the MPLAB IDE software (**Section 3.2 “Installing the Software”**), invoke it by using any of these methods:

- Select *Start>Programs>Microchip>MPLAB IDE vx.xx>MPLAB IDE*, where vx.xx is the version number.
- Double click the MPLAB IDE desktop icon.
- Execute the file `mplab.exe` in the `\core` subdirectory of the MPLAB IDE installation directory.

For more information on using the software, see:

- *“MPLAB IDE User’s Guide”* (DS51519) – Comprehensive guide for using MPLAB IDE.
- *“MPLAB IDE Quick Start Guide”* (DS51281) – Chapters 1 and 2 of the user’s guide.
- The on-line help files – The most up-to-date information on MPLAB IDE and MPLAB REAL ICE in-circuit emulator.
- Readme files – Last minute information on each release is included in `Readme for MPLAB IDE.txt` and `Readme for MPLAB REAL ICE Emulator.txt`. Both files are found in the `Readmes` subdirectory of the MPLAB IDE installation directory.

## 4.3 CREATING A PROJECT

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

- Select your device (e.g., PIC24FJ128GA010)
  - Select a language toolsuite (e.g., Microchip C30 Toolsuite)
  - Name the project
  - Add files (e.g., program.c, support.s, p24FJ128GA010.gld)
- Don't forget to add a linker script file to your project. Linker scripts may be found, by default, in the following directories:

### **MPLAB ASM30**

- C:\Program Files\Microchip\MPLAB ASM30 Suite\Support\gld

### **MPLAB C30**

- C:\Program Files\Microchip\MPLAB C30\support\gld

- C:\pic30\_tools\support\gld

## 4.4 VIEWING THE PROJECT

After the Project Wizard has created a project, the project and its associated files are visible in the Project window. Additional files can be added to the project using the Project window. Right click on any line in the project window tree to pop up a menu with additional options for adding and removing files.

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

## 4.5 BUILDING THE PROJECT

After the project is created, the application needs to be built. This will create object (hex) code for the application that can be programmed into the target by the MPLAB REAL ICE in-circuit emulator.

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

**Note:** On the Project Manager toolbar, select “Debug” from the drop-down list.

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

## 4.6 SETTING CONFIGURATION BITS

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

On most devices, the Watchdog Timer is enabled initially. It is usually a good idea to disable this bit.

## 4.7 SETTING THE EMULATOR AS THE DEBUGGER OR PROGRAMMER

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

Select *Programmer>Select Programmer>MPLAB REAL ICE* to choose the MPLAB REAL ICE in-circuit emulator as the programmer tool. The Programmer menu and MPLAB IDE toolbar will change to display programmer options once the tool is selected. Also, the Output window will open and messages concerning ICE status and communications will be displayed on the **MPLAB REAL ICE** tab.

## 4.8 SETTINGS DIALOG

Select either *Debugger>Settings* or *Programmer>Settings* to open the Settings dialog and set up the MPLAB REAL ICE in-circuit emulator.

- Program Memory Tab
- Configuration Tab
- Instrumented Trace Tab

### 4.8.1 Program Memory Tab

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

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

**TABLE 4-1: MANUAL SELECTION OPTIONS**

<b>Memories</b>	
Program	Check to program Program Memory into target.
Configuration	Check to program Configuration bits into target. <b>Note:</b> This memory is always programmed when in Debug mode.
EEPROM	Check to erase and then program EEPROM memory on target, if available. Uncheck to erase EEPROM memory on target.
ID	Check to program ID Memory into target.
<b>Program Options</b>	
Erase all before Program	Check to erase all memory before programming begins. Unless programming new or already erased devices, it is important to have this box checked. If not checked, the device is not erased and program code will be merged with the code already in the device.
<b>Program Memory</b>	
Start, End	The starting and ending hex address range in program memory for programming, reading, or verification. If you receive a programming error due to an incorrect end address, you need to perform a reconnect, correct the end address and program again. <b>Note:</b> The address range does not apply to the Erase function. The Erase function will erase all data on the device.

## 4.8.2 Configuration Tab

Set up firmware options.

- Check “Auto Download Latest Firmware” (Recommended.)
- Click **Manual Download** to manually select a firmware file to download to the target device.

## 4.8.3 Instrumented Trace Tab

Set the size of the trace buffer.

**Note:** There is currently a 256k trace line maximum.

For more on tracing, see **Chapter 5. “Using the Emulator as a Debugger”**.

---

---

## Chapter 5. Using the Emulator as a Debugger

---

---

### 5.1 INTRODUCTION

How to use the MPLAB REAL ICE in-circuit emulator as a debugger is discussed.

- Debugger Overview
- Breakpoints
- Triggers
- Trace
- Debugging Functions
- Debugging Dialogs/Windows

### 5.2 DEBUGGER OVERVIEW

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

Select *Debugger>Settings* to open the Settings dialog and set up options as needed.

### 5.3 BREAKPOINTS

Select *Debugger>Breakpoints* to open the Breakpoints dialog and set up multiple breakpoints and breakpoint conditions, or right click on a line of code to set up an individual breakpoint.

Breakpoints and triggers use the same resources. Therefore, the available number of breakpoints is actually the available number of combined breakpoints/triggers.

See **Section 5.7.1 “Breakpoints Dialog”** for more information.

### 5.4 TRIGGERS

Select *Debugger>Triggers* to open the Triggers dialog to set up:

- Real-time data capture triggers. Use real-time data capture for variables in MPLAB IDE windows to see real-time data updates of values instead of updates on halt.
- External triggers. Use external triggers to set up hardware triggers using the logic probe port.

Breakpoints and data capture triggers use the same resources. Therefore, the available number of breakpoints is actually the available number of combined breakpoints/triggers.

<p><b>Note:</b> There is a 60-instruction-cycle delay between data captures.</p>
--

See **Section 5.7.7 “Triggers Dialog”** for more information.

## 5.5 TRACE

This section will discuss the types of available instrumented trace and how to use them. For information on the trace window, see **Section 5.7.10 “Trace Window”**.

- Capture Trace
- I/O Port Trace
- Requirements for Trace
- Setting Up the Project for Trace
- Setting Up Trace in MPLAB IDE
- Running Trace
- Disabling Trace

### 5.5.1 Capture Trace

Capture trace can be used with either standard or high-speed communications, with no additional connections. This two-wire interface uses the instrumented trace macro format (see **Section 5.5.5 “Setting Up Trace in MPLAB IDE”**).

If capture trace is used, then real-time data capture triggers cannot be used because of hardware constraints. However, breakpoints are still available. To use data capture triggers, you must disable capture trace (see **Section 5.5.7 “Disabling Trace”**).

### 5.5.2 I/O Port Trace

I/O Port trace can be used with either standard or high-speed communications. Trace clock and data are provided from a device I/O port through the MPLAB REAL ICE in-circuit emulator logic probe connector. For hardware connections, see **Section 2.5.6 “I/O Port Trace Connections”**.

The port interface uses the instrumented trace macro format (see **Section 5.5.5 “Setting Up Trace in MPLAB IDE”**).

### 5.5.3 Requirements for Trace

The following is required to use trace:

- MPLAB IDE v7.43 and above
- MPLAB C30 v2.04 and above

<b>Note:</b> Instrumented trace is only available when using C code, not assembly.
--

### 5.5.4 Setting Up the Project for Trace

Refer to **Chapter 4. “General Setup”** for a discussion of how to set up MPLAB IDE and an MPLAB IDE project to use the MPLAB REAL ICE in-circuit emulator.

To enable trace:

- Select *Project>Build Options>Project, Trace* tab. Click “Enable Instrumented Trace” and then select the Transport for trace data, i.e., “Capture Trace” for standard communication transport or “I/O Port” for I/O port data transport. If using I/O port transport, select the port you will be using from the pull-down list. Click **OK**.

# Using the Emulator as a Debugger

## 5.5.5 Setting Up Trace in MPLAB IDE

The trace buffer can hold up to 256K bytes of information and can be set to a value up to that maximum (see **Section 4.8.3 “Instrumented Trace Tab”**). The trace buffer is circular, so data will wrap if the maximum is exceeded.

- To record a PC location, click on or highlight a line of code and then right click to select “Insert Line Trace” from the pop-up menu. This causes the following line to be inserted above to the selected line:

```
__TRACE(id);
```

where *id* is a line trace number auto-generated during the build.

**Note:** Inserting a macro into code may modify the logic flow of the program. Please be sure that braces are present where necessary.

- The recording of a variable value is performed much in the same way. First highlight the variable name or expression and then right click to select “Log Selected Value” from the pop-up menu. This causes the following line to be inserted above the line containing the variable:

```
__LOG(id,selected variable);
```

where *id* is a log number auto-generated during build and *selected variable* is the highlighted variable.

- To remove a trace point, simply highlight and then delete the Trace/Log macro.

## 5.5.6 Running Trace

1. Rebuild the project (*Project>Build All*).

**Note:** On the Project Manager toolbar, select “Debug” from the drop-down list.

2. After rebuilding, if there are trace macros in code, a Warning dialog will ask, “File has been modified. Do you want to reload?”. Click **Yes**. When you examine your code, you will find that all *ids* have been replaced with unique numbers.

**Note:** To disable this warning and automatically reload, select *Configure>Settings*, **Other** tab, and check “Automatically reload files that were modified outside of the IDE”. Then click **OK**.

3. Reprogram the device (*Debugger>Program*).
4. Run the program and then halt, or set a breakpoint to halt.
5. Select *View>Trace* to view the trace data (**Section 5.7.10 “Trace Window”**) or right click and in the Trace window and select “Reload”.

Repeat these steps each time you change a trace point.

## 5.5.7 Disabling Trace

To disable the trace capability:

1. Remove all trace and log macros from code.
2. Select *Project>Build Options>Project, Trace* tab. Uncheck "Enable Instrumented Trace". Click **OK**.
3. Rebuild the project (*Project>Build All*).
4. Reprogram the device (*Debugger>Program*).

To temporarily disable trace transport:

- Select *Project>Build Options>Project, Trace* tab. Select "Off" for Transport. Click **OK**.

## 5.6 DEBUGGING FUNCTIONS

When you select the MPLAB REAL ICE in-circuit emulator from the Debugger menu, debug items will be added to the following MPLAB IDE functions:

- Debugger Menu
- Right Mouse Button Menu
- Toolbars/Status Bar

### 5.6.1 Debugger Menu

#### Run F9

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

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

#### Animate

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

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

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

#### Halt F5

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

#### Step Into F7

Single step through program code.

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

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

<b>Note:</b> Do not step into a <code>SLEEP</code> instruction.
---

#### Step Over F8

In C code, steps over the current line of code.

# Using the Emulator as a Debugger

---

## Step Out

Not available.

## Reset F6

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

## Breakpoints

Open the Breakpoint dialog (see **Section 5.7.1 “Breakpoints Dialog”**). Set multiple breakpoints in this dialog.

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

## Triggers

Set up real-time data capture triggers (see **Section 5.7.7 “Triggers Dialog”**).

## Program

Download your code to the target device.

## Read

Read target memory. Information uploaded to MPLAB IDE.

## Erase Flash Device

Erase all Flash memory.

## Abort

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

## Settings

Open the Programmer dialog (see **Section 4.8 “Settings Dialog”**). Set up program and firmware options.

## 5.6.2 Right Mouse Button Menu

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

### Log Selected Value

Log the value of the highlighted variable in the trace window. See **Section 5.5.5 “Setting Up Trace in MPLAB IDE”**.

### Insert Line Trace

Log the occurrence of the selected line in the trace window. See **Section 5.5.5 “Setting Up Trace in MPLAB IDE”**.

### Set/Remove Breakpoint

Set or remove a breakpoint at the currently selected line.

### Enable/Disable Breakpoint

Enable or disable a breakpoint at the currently selected line.

### Breakpoints

Remove, enable or disable all breakpoints.

## Run To Cursor

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

## Set PC at Cursor

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

### 5.6.3 Toolbars/Status Bar

When the MPLAB REAL ICE in-circuit emulator is selected as a debugger, these toolbars are displayed in MPLAB IDE:

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

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

## 5.7 DEBUGGING DIALOGS/WINDOWS

Open the following debug dialogs and windows using the menu items mentioned in **Section 5.6 “Debugging Functions”**.

- Breakpoints Dialog
  - Set Breakpoint Dialog
  - Stopwatch Dialog
  - Event Breakpoints Dialog
  - Sequenced Breakpoints Dialog
  - AND Dialog
- Triggers Dialog
  - Data Capture Properties Dialog
  - Add External Trigger Dialog
- Trace Window

### 5.7.1 Breakpoints Dialog

Set up different types of breakpoints in this dialog. Click on **Add Breakpoint** to add breakpoints to the dialog window. Then use the other buttons for more advanced breakpoint options.

#### 5.7.1.1 BREAKPOINT DIALOG WINDOW

Information about each breakpoint is visible in this window.

**TABLE 5-1: BREAKPOINT DIALOG WINDOW**

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

# Using the Emulator as a Debugger

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

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

## 5.7.1.2 BREAKPOINT DIALOG BUTTONS

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

**TABLE 5-2: BREAKPOINT DIALOG BUTTONS**

Control	Function	Related Dialog
Add Breakpoint	Add a breakpoint	Section 5.7.2 “Set Breakpoint Dialog”
Stopwatch	Set up the stopwatch	Section 5.7.3 “Stopwatch Dialog”
Event Breakpoints	Set up break on an event	Section 5.7.4 “Event Breakpoints Dialog”
Sequenced Breakpoints	Set up a sequence till break	Section 5.7.5 “Sequenced Breakpoints Dialog”
ANDED Breakpoints	Set up ANDED condition till break	Section 5.7.6 “AND Dialog”

## 5.7.2 Set Breakpoint Dialog

Select a breakpoint for the Breakpoints Dialog here.

### 5.7.2.1 PROGRAM MEMORY TAB

Set up a program memory breakpoint here.

**TABLE 5-3: PROGRAM MEMORY BREAKPOINT**

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

## 5.7.2.2 DATA MEMORY TAB

Set up a data memory breakpoint here.

**TABLE 5-4: DATA MEMORY BREAKPOINT**

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

## 5.7.3 Stopwatch Dialog

The stopwatch allows timing from one breakpoint/trigger condition to the next.

**TABLE 5-5: STOPWATCH SETUP**

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

## 5.7.4 Event Breakpoints Dialog

Select a condition where the program will always break:

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

## 5.7.5 Sequenced Breakpoints Dialog

Set up a sequential occurrence of breakpoints. Sequence execution of breakpoints is bottom-up; the last breakpoint in the sequence occurs first.

To add a breakpoint to a sequence:

- Select a breakpoint from the list of “Available Breakpoints”. Available breakpoints/triggers are those previously added to the breakpoint dialog.
- Select a sequence for the list of “Sequences”.
- Click **Add**.

To change the order of breakpoints in a sequence, drag-and-drop the breakpoint in the “Sequences list”.

To remove a breakpoint from a sequence:

- Select the breakpoint in the “Sequences” list.
- Click **Remove**.

## 5.7.6 AND Dialog

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.

To add a breakpoint to the AND condition:

- Select a breakpoint from the list of “Available Breakpoints”. Available breakpoints/triggers are those previously added to the breakpoint dialog.
- Click **Add**.

To remove a breakpoint from a sequence:

- Select the breakpoint in the “ANDed Breakpoints” list.
- Click **Remove**.

## 5.7.7 Triggers Dialog

Set up triggers in this dialog. Click on **Add Data Capture** to open the Data Capture Properties Dialog to set up a real-time data capture. Click on **Add External Triggers** to open the Add External Trigger Dialog to set up hardware triggers.

When complete, trigger information will appear in this dialog. Each trigger may be enabled/disabled individually.

## 5.7.8 Data Capture Properties Dialog

Use this dialog to set up real-time data capture for a data address. Real-time data capture provides updating of variables in the Watch, File Register and Special Function Register window in real time instead of on halt.

- Enter the hex address of the data address for capture.
- Specify if the capture happens on read or write from the X bus.

Data capture can only be performed on register-sized variables. For dsPIC DSC/PIC24 devices, only 16-bit variables can be captured, such as shorts and ints, but not floats and longs.

Triggers use the same resources as breakpoints, so the maximum available number of breakpoints applies to the maximum number of available triggers and breakpoints. As an example, if 4 breakpoints are available, and two breakpoints are set, up to two triggers may be set also.

If Capture trace is used, then real-time data capture triggers cannot be used because of hardware constraints. However, breakpoints are still available.

## 5.7.9 Add External Trigger Dialog

Use this dialog to set up external triggering via the logic probe port. Depending on the processor speed, the amount of skid after the trigger can be significant.

- Select a pin to use as the trigger from the pull-down menu. Once selected, the corresponding pin is highlighted on a figure of the logic probe port as viewed from the front of the emulator.
- Next select the type of trigger.

**TABLE 5-6: TRIGGER SETUP**

Trigger Type	Trigger	Action
Input	Positive or negative edge triggered	Halt or Reset on trigger
Output	High-to-low or Low-to-high pulse	Assert on Halt or Run

## 5.7.10 Trace Window

View trace information in this window (*View>Trace*). See **Section 5.5.1 “Capture Trace”** for more on tracing.

Traced code or variable values are visible in the top half of the window.

- Line – line number of item traced
- Address – line address of item traced
- Value – value of traced item, if applicable

For each trace item selected/highlighted, the corresponding code line will be shown in the bottom half of the window, if “Show Source” has been selected.

### Trace Window Menu

Below are the menu items in the Trace window right click menu.

#### Close

Close this window.

#### Find

Opens the Find dialog. In the Find What field, enter a string of text you want to find, or select text from the drop-down list. You can also select text in the edit window or place the cursor over a word you want to search for, before you open the Find dialog.

In the Find dialog you may select any of the available options and the direction you want to search. Up searches backward from the insertion point, Down searches forward.

#### Find Next

Find the next instance of Find text.

<F3> repeats the last Find.

<Shift> + <F3> reverses the direction of the last Find.

#### Go To

Jump to the specified item:

- Trigger – Jump to the location of the trigger.
- Top – Jump to the top of the window.
- Bottom – Jump to the bottom of the window.
- Go To Trace Line – Go to the trace line specified in the dialog.
- Go To Source Line – Open a File window and go to the source code line corresponding to the selected trace line.

# Using the Emulator as a Debugger

---

## **Show Source**

Show/hide the source code listing on the bottom of the window. The window bar dividing the trace and source code may be dragged to resize each portion.

## **Reload**

Reload the trace memory window with the contents of the trace buffer.

## **Output to File**

Export the contents of the trace memory window to a file. Uses a Save As dialog, with the addition of cycle and tab information. Enter a "Start" and "End" cycle to write to the file. Also specify if the text is to be tab-delimited.

## **Print**

Print the contents of the trace memory window.

## **Refresh**

Refresh the viewable contents of the window.

## **Properties**

Set up window properties.

# MPLAB<sup>®</sup> REAL ICE<sup>™</sup> In-Circuit Emulator User's Guide

---

NOTES:

---

---

## Chapter 6. Using the Emulator as a Programmer

---

---

### 6.1 INTRODUCTION

How to use the MPLAB REAL ICE in-circuit emulator as a programmer is discussed.

- Programmer Overview
- Programming Functions

### 6.2 PROGRAMMER OVERVIEW

Select *Programmer>Select Programmer>MPLAB REAL ICE* to choose the MPLAB REAL ICE in-circuit emulator as the programmer tool. The Programmer menu and MPLAB IDE toolbar will change to display programmer options once the tool is selected. Also, the Output window will open and messages concerning ICE status and communications will be displayed on the **MPLAB REAL ICE** tab.

Select *Programmer>Settings* to open the Settings dialog and set up options as needed.

### 6.3 PROGRAMMING FUNCTIONS

When you select the MPLAB REAL ICE in-circuit emulator from the Programmer menu, program items will be added to the following MPLAB IDE functions:

- Programmer Menu
- Toolbars/Status Bar

#### 6.3.1 Programmer Menu

##### **Program**

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

##### **Verify**

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

##### **Read**

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

##### **Blank Check All**

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

##### **Erase Flash Device**

Erase all Flash memory.

##### **Settings**

Open the Programmer dialog (see **Section 4.8 “Settings Dialog”**). Set up program and firmware options.

## 6.3.2 Toolbars/Status Bar

When the MPLAB REAL ICE in-circuit emulator is selected as a programmer, these toolbars are displayed in MPLAB IDE:

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

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



# MPLAB® REAL ICE™ IN-CIRCUIT EMULATOR USER'S GUIDE

---

---

## Chapter 7. Hardware Specification

---

---

### 7.1 INTRODUCTION

The hardware and electrical specifications of the MPLAB REAL ICE in-circuit emulator system are detailed.

### 7.2 HIGHLIGHTS

This chapter discusses:

- Declaration of Conformity
- USB Port/Power
- Emulator Pod
- Standard Communication Board
- High-Speed Communication Boards
- Other Emulator Boards (Future)
- Target Board

### 7.3 DECLARATION OF CONFORMITY

We

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

hereby declare that the product:

MPLAB® REAL ICE™ In-Circuit Emulator

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

Standards: EN61010-1      Laboratory Equipment

Microchip Technology, Inc.

Date: August 2006

#### Important Information Concerning the Use of the MPLAB REAL ICE In-Circuit Emulator

Due to the special nature of the MPLAB REAL ICE in-circuit emulator, the user is advised that it can generate higher than normal levels of electromagnetic radiation which can interfere with the operation of all kinds of radio and other equipment.

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

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

## 7.4 USB PORT/POWER

The MPLAB REAL ICE in-circuit emulator is connected to the host PC via a Universal Serial Bus (USB) port, version 2.0 compliant. The USB connector is located on the back of the pod. A USB port on the host PC is required.

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

System power is derived from the USB interface. The emulator is classified as a high power system per the USB specification, and requires 300 mA of power from the USB to function in all operational modes (emulator/programmer).

**Cable Length** – The PC-to-emulator cable length for proper operation has been tested for each driver board and is shipped in the emulator kit.

## 7.5 EMULATOR POD

The emulator pod consists of a main board enclosed in the casing with a port for two optional driver boards (for standard or high-speed communication with a target). On the emulator enclosure are push buttons, indicator lights (LEDs) and a logic probe connector interface.

### 7.5.1 Main Board

This component has the interface processor (dsPIC DSC), the USB 2.0 interface capable of USB speeds of 480 Mb/sec, a Field Programmable Gate Array (FPGA) for general system control and increased communication throughput, an SRAM for holding the program code image for programming into the emulation device on-board Flash, the external trigger logic, user interface push buttons and LED indicators.

The MPLAB REAL ICE in-circuit emulator system supports two types of interfaces to the target processor. They consist of the standard driver board and an optional high-speed driver board. These boards are inserted into the emulator pod via a card guide.

Durability/insertion life cycle of the card guide: 10,000 cycles

### 7.5.2 Push Buttons

The push buttons have the following significance.

Push Button	Related LED	Description
Reset	Status	Push to Reset the device.
Function	Status	Halt – When running, push to put the emulator in the Break or halted condition.

## 7.5.3 Indicator Lights (LEDs)

The indicator lights have the following significance.

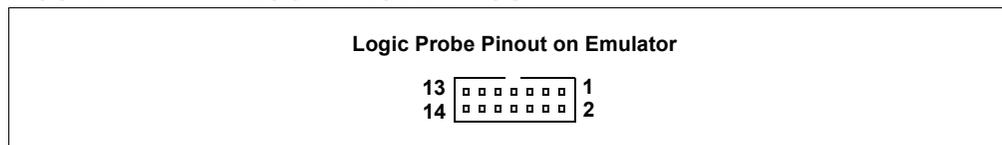
LED	Color	Description
Active	Blue	Lit when power is first applied or when target is connected.
Status	Green	Lit when the emulator is operating normally – standby.
	Red	Lit when an operation has failed.
	Orange	Lit when the emulator is busy.

## 7.5.4 Logic Probe/External Trigger Interface

Probes can be connected to the 14-pin header on the side of the unit for processing external signals that are used for triggering external equipment. This header contains 8 input/output connections that are user selectable as inputs or outputs with logic levels that are proportional to the target operating voltage. The outputs can be used for triggering an external logic analyzer or oscilloscope to allow the developer capture events of interest based on trigger criteria set within MPLAB IDE.

The inputs are part of a trigger bus.

**FIGURE 7-1: LOGIC PROBE PINOUT**



Logic probes may be attached to this connector to give the functionality described in Table 7-1. The probes are color coded for easy identification.

**TABLE 7-1: LOGIC PROBE PINOUT DESCRIPTION**

Pin	I/O	Name	Function	Color
1	O	VDD*	VDD reference	Red
2	O	NC	No connection	Gray
3	O	NC	No connection	Gray
4	I	TCLK	External synchronous clock	Gray
5	I/O	EXT7**	External input/output bit 7	White
6	I/O	EXT6	External input/output bit 6	White
7	I/O	EXT5	External input/output bit 5	White
8	I/O	EXT4	External input/output bit 4	White
9	I/O	EXT3	External input/output bit 3	White
10	I/O	EXT2	External input/output bit 2	White
11	I/O	EXT1	External input/output bit 1	White
12	I/O	EXT0**	External input/output bit 0	White
13	Gnd	GND	System Ground	Black
14	Gnd	GND	System Ground	Black

\*Do not connect VDD to the target.

\*\*Do not connect EXT0 to EXT7. This is dedicated to self test.

The electrical specifications for logic probes are listed in Table 7-2.

**TABLE 7-2: LOGIC PROBE ELECTRICAL SPECIFICATIONS**

Logic Inputs	VIH = VDD x 0.7V (min)			
	VIL = VDD x 0.3V (max)			
Logic Outputs	VDD = 5V	VDD = 3V	VDD = 2.3V	VDD = 1.65V
	VOH = 3.8V min	VOH = 2.4V min	VOH = 1.9V min	VOH = 1.2V min
	VOL = 0.55V max	VOL = 0.55V max	VOL = 0.3V max	VOL = 0.45V max

## 7.6 STANDARD COMMUNICATION BOARD

For standard emulator communication with a target (**Section 2.4.1 “Standard Communication”**), use the standard driver board.

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

The VPP high-voltage lines can produce a variable voltage that can swing from 14V down to 0 volts to satisfy the voltage requirements for the specific emulation processor.

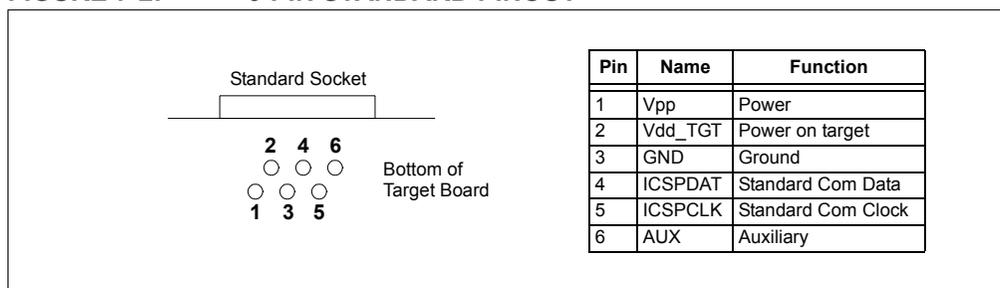
The VDD sense connection draws very little current from the target processor. The actual power comes from the MPLAB REAL ICE in-circuit emulation system as the VDD sense line is used as a reference only to track the target voltage. The VDD connection is isolated with an optical switch.

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 REAL ICE in-circuit emulator system)
- Clock and data signals are protected from high voltages caused by faulty targets systems, or improper connections
- Clock and data signals are protected from high current caused from electrical shorts in faulty target systems

**Note:** When using the standard driver board, the rate for real-time streaming data and tracing is limited to 20 MIPS.

**FIGURE 7-2: 6-PIN STANDARD PINOUT**



## 7.7 HIGH-SPEED COMMUNICATION BOARDS

For high-speed emulator communication with a target (Section 2.4.2 “High-Speed Communication”), use the following boards:

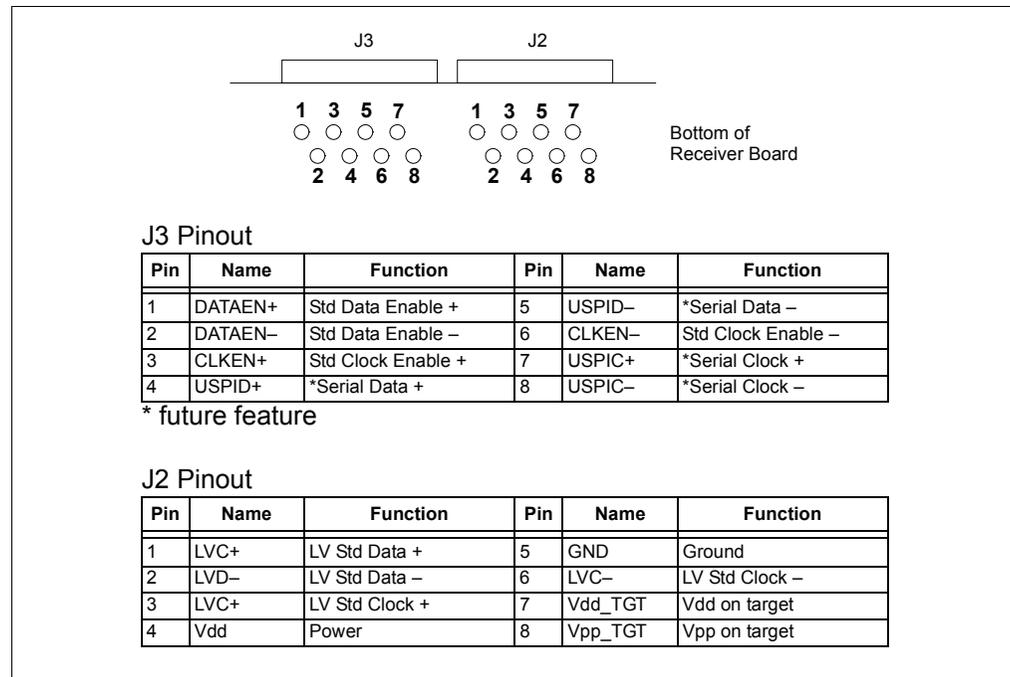
- High-Speed Driver Board
- High-Speed Receiver Board

### 7.7.1 High-Speed Driver Board

The high-speed driver board consists of two separate multipoint LVDS (Low Voltage Differential Signal) transmitters and receivers for clock and data. Multipoint LVDS requires 100 ohm terminations at each driver output and receiver input, per the standard, and multipoint configurations type 2 receivers are used, as these are intended for control signals or where fail-safe provisions are needed. Even though the standard allows for any combination of drivers, receivers and/or transceivers of up to 32 on the line, only two will be used. The driver board has a port expansion which is controlled by an I<sup>2</sup>C™ interface for sending and receiving status information to the emulator. The high-speed driver board assembly is inserted into the emulator pod via the card guide.

**Note:** Data rates up to 40 MIPS are possible.

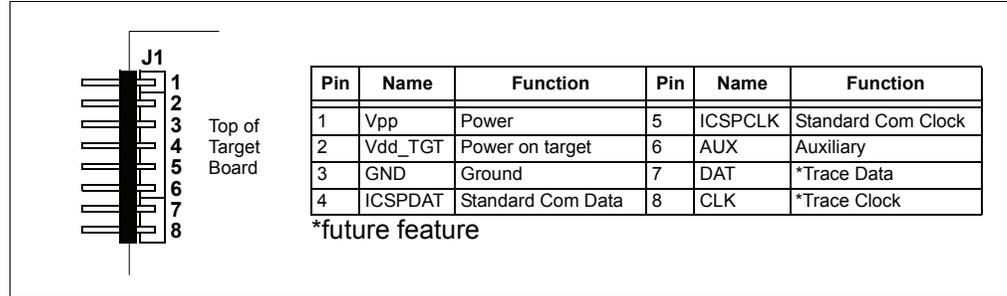
**FIGURE 7-3: DUAL 8-PIN PINOUT**



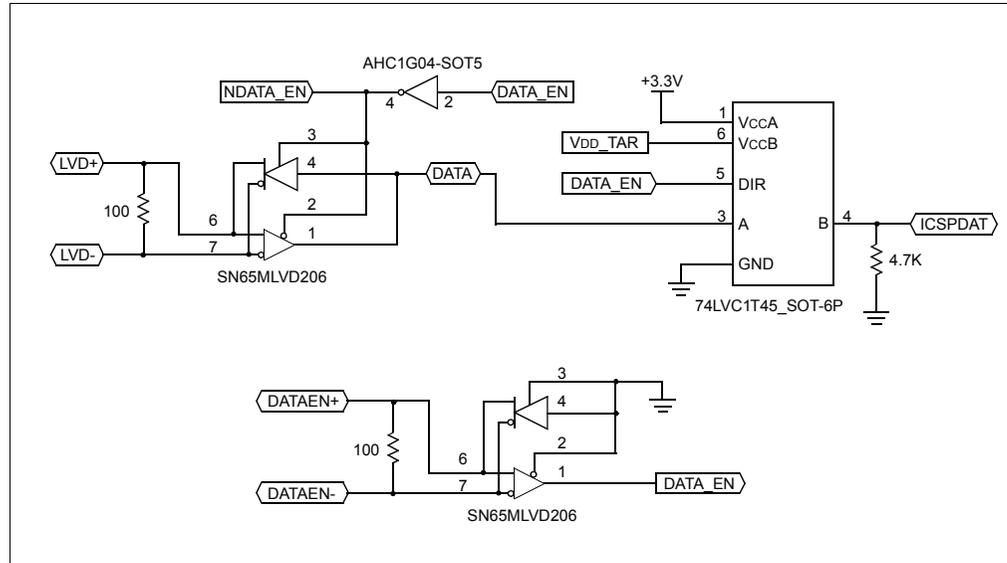
## 7.7.2 High-Speed Receiver Board

A high-speed receiver board assembly is also required when using LVDS connectivity. This board is a counterpart to the high-speed driver board assembly in the pod. When the driver is active on the pod, the receiver is active in the receiver board. Alternatively, when the driver is active on the receiver board, the corresponding receiver is active in the driver board, providing transmitting and receiving capability at both extremes. The receiver board contains an 8-pin, 0.100 inch centers header, and is used to connect to the target board. The receiver board circuit may be absorbed into the target system, avoiding the usage of the receiver board.

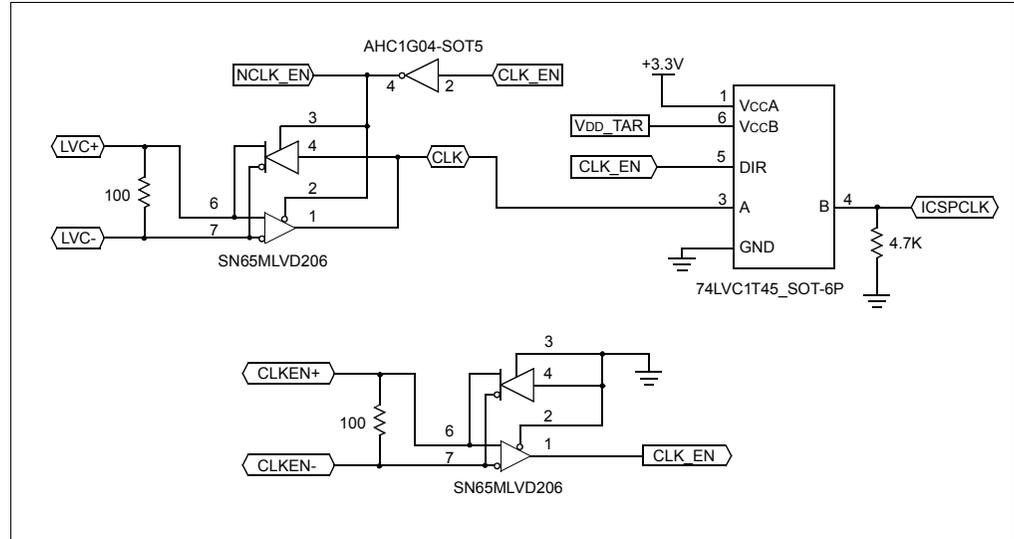
**FIGURE 7-4: 8-PIN HEADER PINOUT**



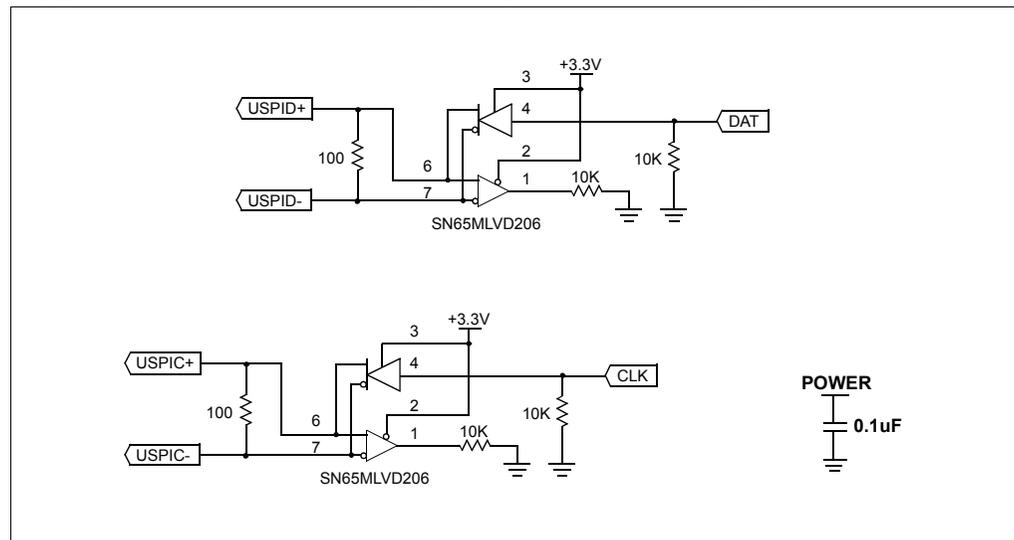
**FIGURE 7-5: RECEIVER BOARD SCHEMATIC – ICSPDAT**



**FIGURE 7-6: RECEIVER BOARD SCHEMATIC – ICSPCLK**



**FIGURE 7-7: RECEIVER BOARD SCHEMATIC – DAT & CLK**



## 7.8 OTHER EMULATOR BOARDS (FUTURE)

Additional boards that will be available soon for use with the emulator system are:

- Header Board
- Self-Test Board
- High-Speed to Standard Converter Board

### 7.8.1 Header Board

To use an ICE device (*Device-ICE*), a header board is required. Simply plug the communication connector(s) into the header board and then connect the header to the target board either directly or through a transition socket.

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

### 7.8.2 Self-Test Board

This board can be used to verify that the emulator is functioning properly. To use this board:

1. Disconnect the emulator from the target and the PC.
2. Insert the standard driver board if it is not already installed.
3. Connect the self-test board to the emulator using the modular cable.
4. Connect the emulator to the PC.
5. Select the MPLAB REAL ICE in-circuit emulator as either a debugger or programmer in MPLAB IDE.
6. MPLAB IDE will automatically detect and run the complete self test and give you a status (pass/fail). Instructions on what to do will follow.

### 7.8.3 High-Speed to Standard Converter Board

This board is a combined high-speed receiver board and standard conversion board. The board allows high-speed communication to be used with a target that has a standard connector.

Plug the high-speed connectors into one end of the board and the standard connector into the other end. This is useful for connecting to a target board with a standard connector which is a large distance away from the emulator, thus calling for LVDS communication.

## 7.9 TARGET BOARD

The target board should be powered according to the requirements of the selected device (1.6V-5.5V) and the application.

**Note:** The emulator cannot power the target.

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

- **Section 2.5.3 “Target Connection Circuitry”**
- **Section 2.5.4 “Circuits That Will Prevent the Emulator From Functioning”**



# MPLAB® REAL ICE™ IN-CIRCUIT EMULATOR USER'S GUIDE

---

---

## Appendix A. Revision History

---

---

### A.1 REVISION HISTORY

#### Revision A (September 2006)

- Initial release of this document.

# MPLAB<sup>®</sup> REAL ICE<sup>™</sup> In-Circuit Emulator User's Guide

---

NOTES:

---

---

## Glossary

---

---

### **Absolute Section**

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

### **Access Memory (PIC18 Only)**

Special registers on PIC18XXXXX devices that allow access regardless of the setting of the Bank Select Register (BSR).

### **Address**

Value that identifies a location in memory.

### **Alphabetic Character**

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

### **Alphanumeric**

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

### **ANSI**

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

### **Application**

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

### **Archive**

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

### **Archiver**

A tool that creates and manipulates libraries.

### **ASCII**

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

### **Assembler**

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

### **Assembly Language**

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

### **Asynchronous Stimulus**

Data generated to simulate external inputs to a simulator device.

### **Breakpoint, Hardware**

An event whose execution will cause a halt.

## **Breakpoint, Software**

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

## **Build**

Compile and link all the source files for an application.

## **C**

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

## **Calibration Memory**

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

## **COFF**

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

## **Command Line Interface**

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

## **Compiler**

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

## **Configuration Bits**

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

## **Control Directives**

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

## **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.

## **Device Programmer**

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

## **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.

**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.

**Emulation**

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

**Emulation Memory**

Program memory contained within the emulator.

**Emulator**

Hardware that performs emulation.

**Emulator System**

The MPLAB ICE 2000 and 4000 emulator systems include the pod, processor module, device adapter, cables and MPLAB IDE software.

**EPROM**

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

**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.

**Export**

Send data out of MPLAB IDE in a standardized format.

**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 PIC17CXXX or PIC18CXXX device.

**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.

## **File Registers**

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

## **Flash**

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

## **FNOP**

Forced No Operation. A forced `NOOP` cycle is the second cycle of a two cycle instruction. Since the PICmicro 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 `NOOP` cycle.

## **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.

## **HEX Code**

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

## **HEX File**

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

## **High Level Language**

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

## **ICD**

In-Circuit Debugger. MPLAB ICD and MPLAB ICD 2 are Microchip's in-circuit debuggers for PIC16F87X and PIC18FXXX devices, respectively. These ICDs work with MPLAB IDE.

## **ICE**

In-Circuit Emulator. MPLAB ICE 2000 and 4000 are Microchip's in-circuit emulators that work with MPLAB IDE.

## **ICSP**

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

## **IDE**

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

## **Import**

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

## **Instruction Set**

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

## **Instructions**

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

**Internal Linkage**

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

**International Organization for Standardization**

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

**Interrupt**

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

**Interrupt Handler**

A routine that processes special code when an interrupt occurs.

**Interrupt Request**

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

**Interrupt Service Routine**

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.

**IRQ**

See Interrupt Request.

**ISO**

See International Organization for Standardization.

**ISR**

See Interrupt Service Routine.

**Librarian**

See Archiver.

**Library**

See Archive.

**Linker**

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

**Linker Script Files**

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

**Listing Directives**

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

**Listing File**

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

## 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.

## LVDS

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

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

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

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

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

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

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

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

## Machine Code

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

## Machine Language

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

## Macro

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

## Macro Directives

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

## 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.

**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 the PIC17CXXX and PIC18CXXX families of 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 the PIC17CXXX and PIC18CXXX families of microcontrollers. In Microprocessor mode, the on-chip program memory is not used. The entire program memory is mapped externally.

**Mnemonics**

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

**MPASM™ Assembler**

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

**MPLAB ASM30**

Microchip's relocatable macro assembler for dsPIC30F digital signal controller devices.

**MPLAB C1X**

Refers to both the MPLAB C17 and MPLAB C18 C compilers from Microchip. MPLAB C17 is the C compiler for PIC17CXXX devices and MPLAB C18 is the C compiler for PIC18CXXX and PIC18FXXXX devices.

**MPLAB C30**

Microchip's C compiler for dsPIC30F digital signal controller devices.

**MPLAB ICD 2**

Microchip's in-circuit debugger for PIC16F87X, PIC18FXXX and dsPIC30FXXXX devices. The ICD works with MPLAB IDE. The main component of each ICD is the module. A complete system consists of a module, header, demo board, cables and MPLAB IDE Software.

**MPLAB ICE 2000**

Microchip's in-circuit emulator for PICmicro MCUs that works with MPLAB IDE.

**MPLAB ICE 4000**

Microchip's in-circuit emulator for dsPIC DSCs that works with MPLAB IDE.

**MPLAB IDE**

Microchip's Integrated Development Environment.

**MPLAB LIB30**

MPLAB LIB30 archiver/librarian is an object librarian for use with COFF object modules created using either MPLAB ASM30 or MPLAB C30 C compiler.

**MPLAB LINK30**

MPLAB LINK30 is an object linker for the Microchip MPLAB ASM30 assembler and the Microchip MPLAB C30 C compiler.

## **MPLAB SIM**

Microchip's simulator that works with MPLAB IDE in support of PICmicro MCU devices.

## **MPLAB SIM30**

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

## **MPLIB<sup>™</sup> Object Librarian**

MPLIB librarian is an object librarian for use with COFF object modules created using either MPASM assembler (mpasm or mpasmwin v2.0) or MPLAB C1X C compilers.

## **MPLINK<sup>™</sup> Object Linker**

MPLINK linker is an object linker for the Microchip MPASM assembler and the Microchip MPLAB C17 or C18 C compilers. MPLINK linker also may be used with the Microchip MPLIB librarian. MPLINK linker is designed to be used with MPLAB IDE, though it does not have to be.

## **MRU**

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

## **Nesting Depth**

The maximum level to which macros can include other macros.

## **Node**

MPLAB IDE project component.

## **Non Real Time**

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

## **Nonvolatile Storage**

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

## **NOP**

No Operation. An instruction that has no effect when executed except to advance the Program Counter.

## **Object Code**

The machine code generated by an assembler or compiler.

## **Object File**

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

## **Object File Directives**

Directives that are used only when creating an object file.

## **Off-Chip Memory**

Off-chip memory refers to the memory selection option for the PIC17CXXX or PIC18CXXX 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 IBM® or compatible personal computer running a supported Windows operating system.

## **PICmicro MCUs**

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

## **PICSTART® Plus**

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

## **Pod, Emulator**

The external emulator box that contains emulation memory, trace memory, event and cycle timers and trace/breakpoint logic.

## **Power-on Reset Emulation**

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

## **PRO MATE® II**

A device programmer from Microchip. Programs all PICmicro microcontrollers and most memory and Keeloq devices. Can be used with MPLAB IDE or stand-alone.

## **Program Counter**

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

## **Program Memory**

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

## **Project**

A set of source files and instructions to build the object and executable code for an application.

## **Prototype System**

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

## **PWM Signals**

Pulse Width Modulation Signals. Certain PICmicro 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.

## **Real Time**

When released from the halt state in the Emulator or MPLAB ICD mode, 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 MPLAB ICE is enabled and constantly captures all selected cycles, and all break logic is enabled. In the emulator or MPLAB ICD, the processor executes in real time until a valid breakpoint causes a halt, or until the user halts the emulator. In the simulator, real time simply means execution of the microcontroller instructions as fast as they can be simulated by the host CPU.

## **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.

## **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.

## **SFR**

See Special Function Registers.

## **Shell**

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

## **Simulator**

A software program that models the operation of devices.

## **Single Step**

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

## **Skew**

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

## **Skid**

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

## **Source Code**

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

## **Source File**

An ASCII text file containing source code.

## **Special Function Registers**

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

## **Stack, Hardware**

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

## **Stack, Software**

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

## **Static RAM or SRAM**

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

## **Status Bar**

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

## **Step Into**

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

## **Step Over**

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

## **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.

## **Symbol**

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

## **System Window Control**

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

## **Target**

Refers to user hardware.

## **Target Application**

Software residing on the target board.

## **Target Board**

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

## **Target Processor**

The microcontroller device on the target application board.

## **Template**

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

## **Tool Bar**

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

## **Trace**

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

## **Trace Memory**

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

## **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.

## **Uninitialized Data**

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

```
int myVar;
```

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

**Upload**

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

**USB**

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

**Warning**

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

**Watch Variable**

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

**Watch Window**

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

**Watchdog Timer**

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

**WDT**

See Watchdog Timer.

# MPLAB® REAL ICE™ In-Circuit Emulator User's Guide

---

NOTES:



# MPLAB<sup>®</sup> REAL ICE<sup>™</sup> IN-CIRCUIT EMULATOR USER'S GUIDE

## Index

<b>A</b>	
Abort .....	31
Animate .....	30
<b>B</b>	
Blank Check .....	39
Breakpoints .....	27, 31, 32
<b>C</b>	
Cables	
Length .....	42
USB .....	10
Capacitors .....	15
Capture Trace .....	28
CD-ROM .....	8
Circuits That Will Prevent the Emulator From Functioning .....	15
Code Protect .....	18
Configuration Bits .....	18, 24
Converter Board, High-Speed to Standard .....	8, 13, 48
Customer Notification Service .....	4
Customer Support .....	5
<b>D</b>	
Data Capture .....	27, 31, 35
Debug	
Executive .....	19
Registers .....	19
Debug Mode	
Sequence of Operations .....	18
Documentation	
Conventions .....	2
Layout .....	1
Driver Board	
High-Speed .....	8, 10, 45
Standard .....	8, 10, 44
Durability, Card Guide .....	42
<b>E</b>	
Erase .....	39
Erase Flash Device .....	31
External Triggers .....	36, 43
<b>H</b>	
Halt .....	30
Header Board .....	19, 48
Specification .....	3
High-Speed Communication .....	12
Connections .....	14
Converter Board .....	48
Driver Board .....	45
Receiver Board .....	46
Hybrid Communication .....	13
<b>I</b>	
I/O Port Trace .....	28
ICSP .....	17, 19, 44
Indicator Lights .....	43
Internet Address, Microchip .....	4
<b>K</b>	
Kit Components .....	8
<b>L</b>	
LEDs .....	43
Logic Probes .....	8, 22, 43
I/O Electrical Specifications .....	44
Pinout .....	43
LVDS .....	45, 46
<b>M</b>	
Memory Used .....	20
Modular Interface Cable .....	17
MPLAB IDE .....	21
MPLAB REAL ICE Defined .....	7
<b>P</b>	
Parallel Trace .....	16
Pod .....	8, 10
Program .....	31, 39
Project Wizard .....	24
Pull-ups .....	15
Push Buttons .....	42
<b>R</b>	
Read .....	31, 39
Reading, Recommended .....	3
Readme .....	3
Real Time Data Capture .....	35
Real Time Watch .....	35
Receiver Board	
High-Speed .....	46
Reset	
Processor .....	31
Resistors .....	15
Run .....	30

## S

Self-Test Board .....	8, 48
Serial Trace .....	16
SPI Trace .....	16
Standard Communication .....	10
Connections .....	13
Converter Board .....	48
Driver Board .....	44
Step .....	30

## T

Table Read Protect .....	18
Target Connection	
Circuitry .....	14
High-Speed .....	14
I/O Port .....	16
Improper Circuits .....	15
SPI/UART .....	16
Standard .....	13
Target Device .....	17, 20
Trace .....	7, 22, 46, 62
Capture .....	28
I/O Port .....	16, 28
SPI/UART .....	16
Trace Window .....	36
Transition Socket .....	8
Specification .....	3, 22
Triggers .....	27, 35
External .....	43

## U

UART Trace .....	16
USB .....	42, 63
Cables .....	8, 10
Device Drivers .....	21

## V

Verify .....	39
--------------	----

## W

Watchdog Timer .....	18
Web Site, Microchip .....	4

NOTES:



---

---

## WORLDWIDE SALES AND SERVICE

---

---

### AMERICAS

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

**Atlanta**  
Alpharetta, GA  
Tel: 770-640-0034  
Fax: 770-640-0307

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

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

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

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

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

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

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

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

### ASIA/PACIFIC

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

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

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

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

**China - Fuzhou**  
Tel: 86-591-8750-3506  
Fax: 86-591-8750-3521

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

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

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

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

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

**China - Shunde**  
Tel: 86-757-2839-5507  
Fax: 86-757-2839-5571

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

**China - Xian**  
Tel: 86-29-8833-7250  
Fax: 86-29-8833-7256

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-4182-8400  
Fax: 91-80-4182-8422

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

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

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

**Korea - Gumi**  
Tel: 82-54-473-4301  
Fax: 82-54-473-4302

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

**Malaysia - Penang**  
Tel: 60-4-646-8870  
Fax: 60-4-646-5086

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

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

**Taiwan - Hsin Chu**  
Tel: 886-3-572-9526  
Fax: 886-3-572-6459

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

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

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

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-3910  
Fax: 43-7242-2244-393

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

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

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

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

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

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

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

# Mouser Electronics

Authorized Distributor

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

[Microchip:](#)

[AC244006](#)