Emulation Extension Pak (EEP) and Emulation Header

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.

Trademarks

Printed on recycled paper.

ISBN: 978-1-62077-860-9

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, 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.

EEP AND EMULATION HEADER USER'S GUIDE

Table of Contents

Chapter 1. EEP and Emulation Header Overview

NOTICE TO CUSTOMERS

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

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

For the most up-to-date information on development tools, see the MPLAB® IDE or MPLAB X IDE online Help (Help menu).

This chapter contains the following topics:

- Emulation Extension Pak and Emulation Header Defined
- Why Do I Need An Emulation Header?
- Programming Notes
- General Emulation Header Setup
- Device vs. Optional Header Features
- MPLAB X IDE Use with Headers
- Calibration Bits
- Performance Issues
- Related Debug Tools
- Customer Support

1.1 EMULATION EXTENSION PAK AND EMULATION HEADER DEFINED

An emulation header is a circuit board that allows an emulator to debug code for a specific device. A special version of the device (-ME2) with on-board emulation circuitry is located on the header. Connectors on the side of the header allow it to connect directly to or through an adapter to the emulator. Connectors on the bottom of the header allow it to connect directly to or through a transition socket to a target board.

An Emulation Extension Pak (EEP) contains an emulation header, gold single in-line pins, and a trace cable and trace adapter board.

1.2 WHY DO I NEED AN EMULATION HEADER?

Although some devices have on-board debug circuitry to allow you to debug your code, you often lose device resources to debugging, i.e., debugging requires the use of two I/O lines, plus Vdd, Vss and Vpp, to communicate with the device. Using a debug header can free up these resources for your application.

Using an emulation header gives you the benefits of a debug header plus new and powerful debugging features. These features give you more choices to pick the right debugging feature(s) to efficiently solve the debugging task at hand.

1.2.1 Advances Debug Features

Advanced debug features and benefits are:

- Real-time Hardware Instruction Trace (**RI**)
  - Provides full instruction execution information up to 32 MHz
  - Trace through Reset conditions
  - Trace buffer with optional stall
- Hardware Address/Data Breakpoints - 32 maximum
  - Break on program memory fetch or data read/writes
  - Program or data address range breakpoints
  - Data masking for bit-field breakpoints
  - Data comparison breakpoints
  - Break on ISR and/or main code
  - Break on Pass Count
  - Break and Trigger Out or just Trigger Out
  - Data break on normal or linear modes
  - Break without halting or as a trigger for other events
- Enhanced Event Breakpoints
  - Break on execution out of bounds - watches PC
  - Break on MCLR Reset
  - Break on trigger in signal
- Background Debug
  - Settings and breakpoints may be changed in runtime or sleep time
  - Faster stepping for lower MCU frequencies compared to -ICE/-ICD parts
- Event Combiners - Four (4) available
  - Each event combiner can combine up to eight (8) events
  - Generates a halt or Trigger Out
  - Modeled on MPLAB ICE 2000 In-Circuit Emulator complex triggers
- Execution Out-Of-Bounds Detection
  - Watch for PC values that exceed the available program memory
- Enhanced Stopwatch Cycle Counter
  - 32-bit instruction cycle counter
- External Trigger In/Out
  - Trigger In: Signal falling edge can cause a Halt or Trigger Out
  - Trigger Out: On an event with an enabled trigger, positive pulse is generated

RI = This feature applies to the MPLAB REAL ICE in-circuit emulator only.

For more information on these features, see: **Chapter 2. "Emulation Header Features"**.

1.2.2 Standard Debug Features

For information on standard debug features, see the user's guide or online help file for your hardware debug tool:

- PICkit 3™ in-circuit debugger
- MPLAB ICD 3 in-circuit debugger
- MPLAB REAL ICE™ in-circuit emulator

Also see the *Processor Extension Pak and Debug Header Specification* (DS51292).

1.3 PROGRAMMING NOTES

The emulation header is designed to be used with the in-circuit emulator in debugger mode, *Debug>Debug Project* (not in programmer mode, *Run>Run Project*) in MPLAB X IDE. Any programming of the special -ME2 device on the header is for debug purposes.

To program production (non-special) devices with your debug tool, use the Universal Programming Module (AC162049) or design a modular interface connector on the target. See the appropriate specification for connections. For the most up-to-date device programming specifications, see the Microchip website www.microchip.com.

Also, production devices can be programmed with the following tools:

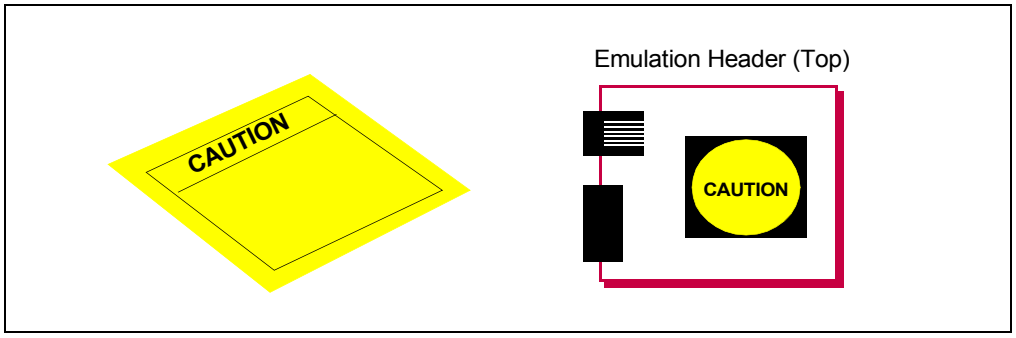
- MPLAB PM3 device programmer
- PICkit 3 development programmer
- MPLAB ICD 3 in-circuit debugger (select as a programmer)
- MPLAB REAL ICE in-circuit emulator (select as a programmer)

1.4 GENERAL EMULATION HEADER SETUP

To set up your header, follow these instructions, before doing anything else:

1. Check the header box for any paper inserts that specify special operating instructions and the emulation header for any stickers (Figure 1-1).

FIGURE 1-1: SPECIAL HEADER INSTRUCTIONS



2. Set any jumpers or switches on the header to determine device functionality or selection as specified for that header. See the section "Emulation Header List" for information on how to set up individual headers.
3. Connect the header to your desired debug tool by consulting the tool documentation for connection options. Example connections are shown in Figures 1-2 through 1-4.

FIGURE 1-2: PICkit 3™ IN-CIRCUIT DEBUGGER CONNECTIONS

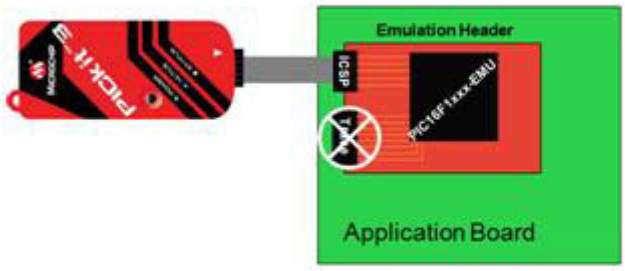


FIGURE 1-3: MPLAB ICD® 3 IN-CIRCUIT DEBUGGER CONNECTIONS

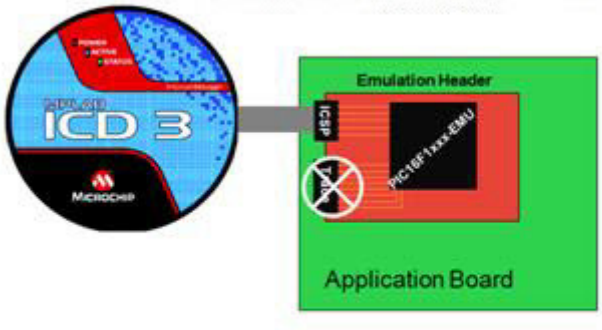
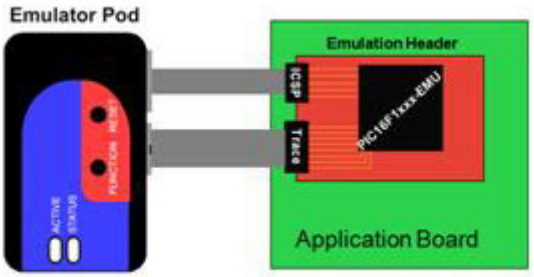


FIGURE 1-4: MPLAB® REAL ICE™ IN-CIRCUIT EMULATOR CONNECTIONS

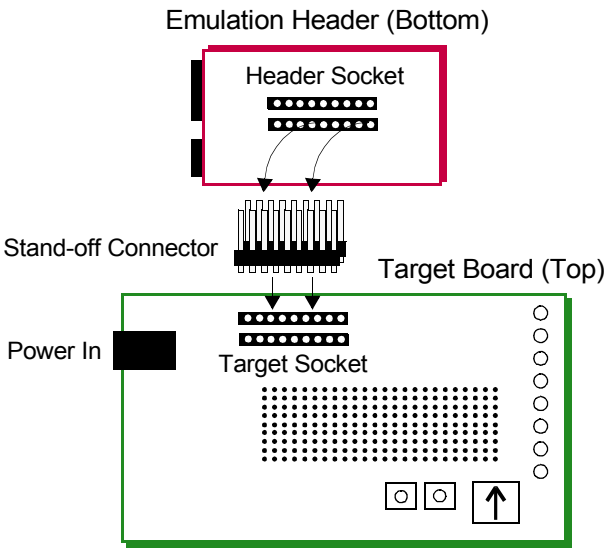


4. Connect the header to the target board. On the bottom of the header is a socket that is used to connect to the target board. The header can be connected to the target board as follows:
   a) PDIP header socket to PDIP target socket with a stand-off (male-to-male) connector or single in-line pins
   b) Header socket to plug on the target board
   c) Header socket to target socket with a transition socket (see the "*Transition Socket Specification*", DS51194)

   An example connection is shown in Figure 1-5.

   The header socket will have the same pin count as your selected device. The -ME2 device on the top of the header usually has a larger pin count because it has additional pins that are dedicated to debug.

FIGURE 1-5: CONNECT HEADER TO TARGET



5. If using a debug tool that can power the target, power that tool now.
6. Power the target, if needed.

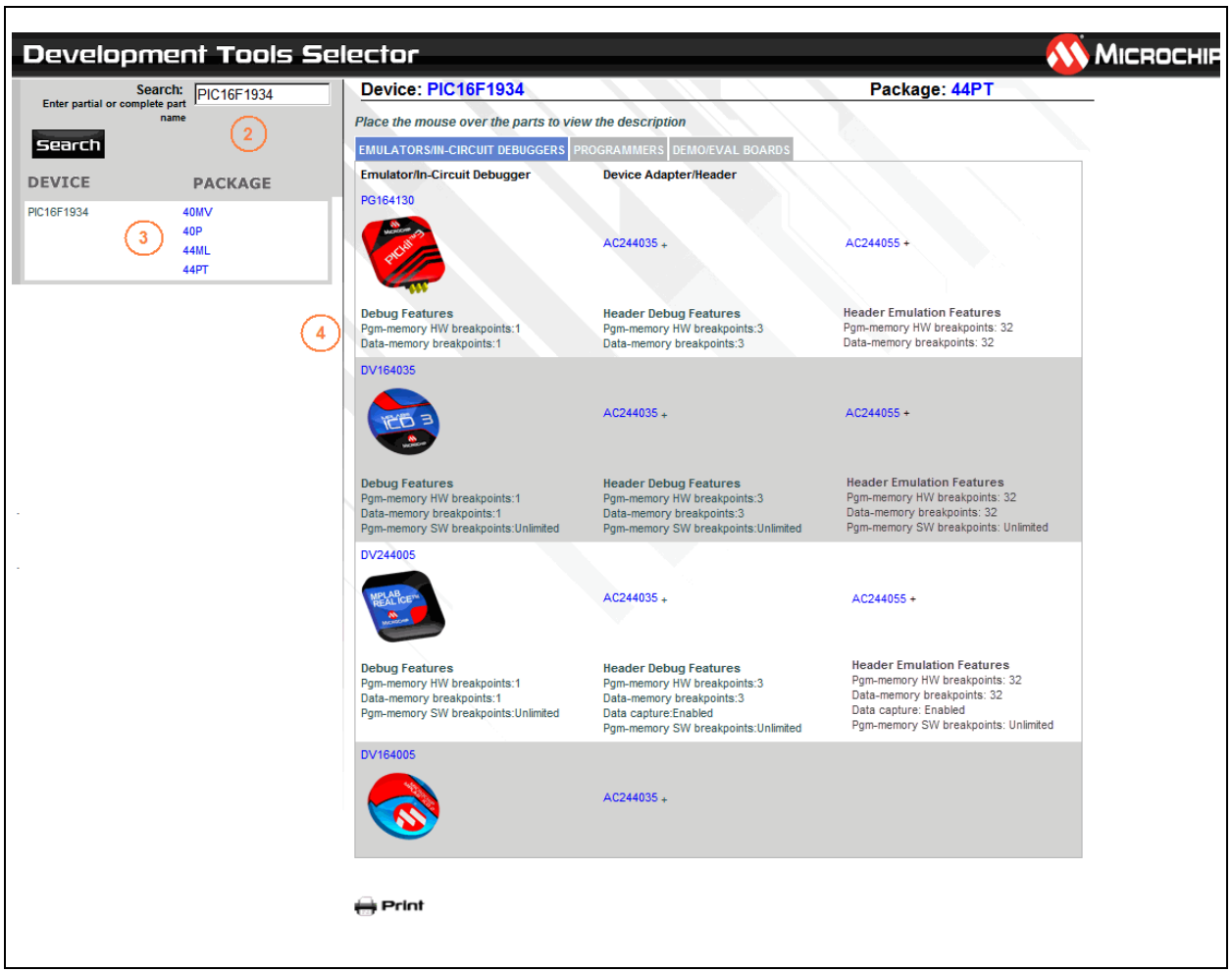
1.5 DEVICE VS. OPTIONAL HEADER FEATURES

This document discusses the benefits of using an emulation header versus devices that have on-board debug capability or debug headers. Another resource for determining the differences in debug features is the Development Tool Selector (DTS).

To find features by device:

1. In a web browser, go to: http://www.microchip.com/dtsapp/
2. Enter your device and click the **Search** button.
3. Select the package you will use.
4. Compare the device under "Debug Features", "Header Debug Features", and "Header Emulation Features".

FIGURE 1-6: DTS DEVICE INFORMATION



1.6 MPLAB X IDE USE WITH HEADERS

Emulation header functionality is supported on MPLAB X IDE, but not on MPLAB IDE v8. Please use debug headers if you are still using MPLAB IDE v8.

You need to do the following to use an emulation header on MPLAB X IDE:

1. Set up the emulation header as specified in “General Emulation Header Setup”.
2. Begin creating a project for a device supported by your emulation header using the Projects wizard (*File>New Project*). See MPLAB X IDE documentation for more on Projects.
3. In one step of the wizard you will have an opportunity to specify the header.
4. In another step you will specify the hardware (debug) tool to which your header is attached.
5. Once the wizard is complete, write code for your project.
6. Select *Debug>Debug Project* to run and debug your code.

Note: An emulation header can only be used to debug (Debug menu), not to program (Run menu). See “Programming Notes”.

1.7 CALIBRATION BITS

The calibration bits for the band gap and internal oscillator are always preserved to their factory settings.

1.8 PERFORMANCE ISSUES

The PIC® MCU devices do not support partial program memory erase; therefore, users may experience slower performance than with other devices.

Also, see the in-circuit emulator Help file for information on specific device limitations that could affect performance.

1.9 RELATED DEBUG TOOLS

The following tools support the use of emulation headers:

- PICkit 3 in-circuit debugger
- MPLAB ICD 3 in-circuit debugger
- MPLAB REAL ICE in-circuit emulator

See the Microchip website for the latest documentation: http://www.microchip.com

1.10 CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

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

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

Documentation errors or comments may be sent to: docerrors@microchip.com.

NOTES:

Chapter 2. Emulation Header Features

2.1 INTRODUCTION

Emulation headers provide an array of debug features. The following requirements and advantages are discussed here:

- Hardware and Software Requirements
- Real Time Hardware Instruction Trace
- Hardware Address/Data Breakpoints
- Enhanced Event Breakpoints
- Background Debug Mode
- Event Combiners
- Stopwatch Cycle Counter
- Interrupt Context Detection
- Trigger In/Out

2.2 HARDWARE AND SOFTWARE REQUIREMENTS

To use emulation header features, the following are required.

Hardware

- an emulation header that is set up as per **Section 1.4 "General Emulation Header Setup"**.
- for Trace, the trace cable and interface board (included with emulation header) also need to be connected as per above.

Software

- MPLAB X IDE v1.90 or greater
- In the Project Properties window (right click on the project name and select "Properties"), and under "Supported Debug Header", ensure that the emulation header is selected.

2.3 REAL TIME HARDWARE INSTRUCTION TRACE

Note: This feature is only supported on the MPLAB REAL ICE in-circuit emulator.

Real Time Hardware Instruction Trace is a real-time dump of the program execution stream that can be captured and analyzed.

The functionality that is provided includes:

- full instruction execution information up to 32 MHz
- trace through Reset conditions
- trace buffer with optional stall

2.3.1 How Trace Works

Emulation header trace is similar to PIC32 instruction trace, which is a non-intrusive hardware instruction trace. You can use trace to capture every instruction executed by the device. The trace data is output from the device (using the pins TRCLK, TRDAT[6:0], and TRSTALL) to the emulator. The emulator streams this data to a trace buffer, that acts like a rolling first-in/first-out (FIFO) buffer, on the PC.

The amount of trace data is limited only by the size of the trace buffer. This buffer can fill quickly even when set to the maximum size, so it is wise to determine exactly what you need to capture.

Enable and set trace options in the Project Properties dialog in the following menu selections:

Categories – "REAL ICE"

Option categories – "Trace and Profiling"
From there, you can set:

"Data selection" – enable/disable trace and select the type of trace.

"Data file path and name" – location of trace file

"Data file maximum size" – size of trace file

"Data Buffer maximum size" – size of the trace buffer

MPLAB X IDE will announce when the trace buffer has overflowed.

Note: Execution will **not** halt when this external buffer is full.

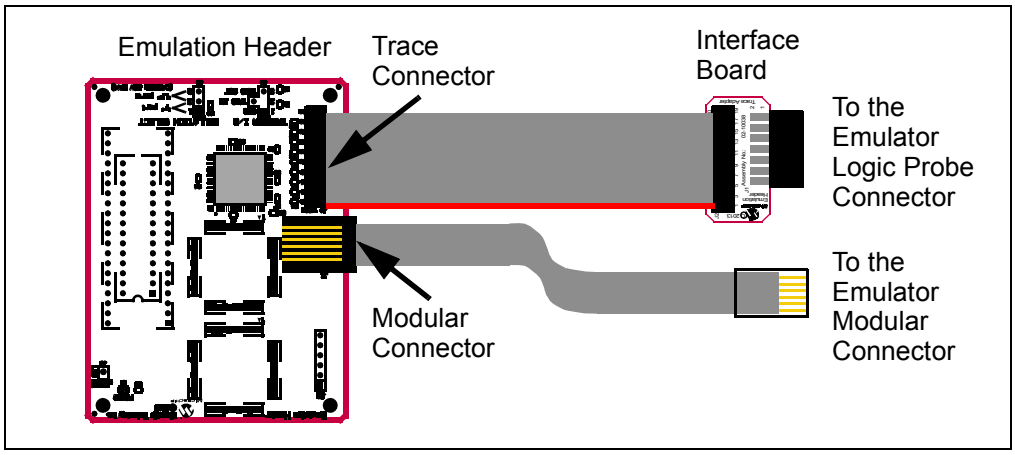
2.3.2 Setting Up and Using Trace

Trace requires hardware and software setup:

2.3.2.1 HARDWARE SETUP

To use trace, you will need the ribbon cable and emulator interface board that comes with the emulation header. Connect one end to the emulation header (Figure 2-1). Connect the other end to the emulator. For more information on complete hardware setup and connections, refer to **Section 1.4 “General Emulation Header Setup”**.

FIGURE 2-1: EMULATION TRACE CABLE CONNECTED TO HEADER



2.3.2.2 MPLAB X IDE SETUP

To set up MPLAB X IDE to use trace for the MPLAB REAL ICE in-circuit emulator, perform the following steps:

1. Right click on the project name and select “Properties” to open the Project Properties window.
2. Click on “Real ICE” under “Categories”.
3. Under “Option categories”, select “Clock”.
   For data capture and trace, the emulator needs to know the instruction cycle speed.
4. Under “Option categories”, select “Trace and Profiling”.
5. Under “Data Collection Selection”, choose “Instruction Trace/Profiling”.
6. Set up any other trace-related options.
7. Click **OK**.

On a debug run, trace will continue to fill the trace buffer with data, rolling over when the buffer is full, until a program Halt.

2.3.2.3 VIEWING TRACE DATA

When trace is enabled and code is run, trace data will be collected by the emulator. Once the device is halted, trace data will be decoded and displayed in the Trace window (*Window>Debugging>Trace*).

FIGURE 2-2: TRACE WINDOW

Line	Address	Op	Label	Instruction
1	000	0x3180		MOVLP 0x0
2	001	0x2802		GOTO 0x2
3	002	0x0064	MainProgram	CLRWDT
4	003	0x30FE	Loop_202	MOVLW 0xFE
5	004	0x0023		MOVLB 0x3
6	005	0x008E		MOVWF PORTC
7	006	0x0022		MOVLB 0x2
8	007	0x100E		BCF PORTC, 0x0
9	008	0x0021		MOVLB 0x1
10	009	0x100E		BCF PORTC, 0x0
11	00A	0x0022		MOVLB 0x2
12	00B	0x140E		BSF PORTC, 0x0
13	000	0x3180		MOVLP 0x0
14	001	0x2802		GOTO 0x2

2.3.3 Improving the Trace Experience

Remove as many USB devices from your PC USB ports as you can. This should improve trace throughput.

2.3.4 Trace Hardware

Hardware details are shown in the following figures.

FIGURE 2-3: TRACE CONNECTOR ON EMULATION HEADER

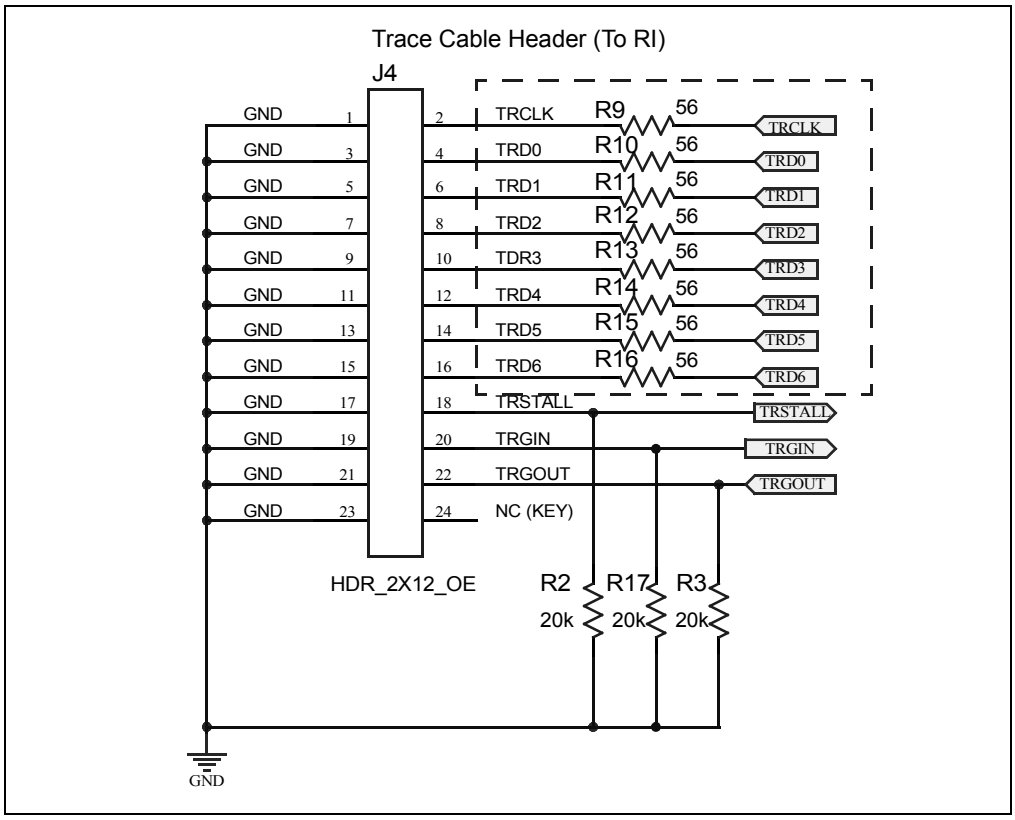
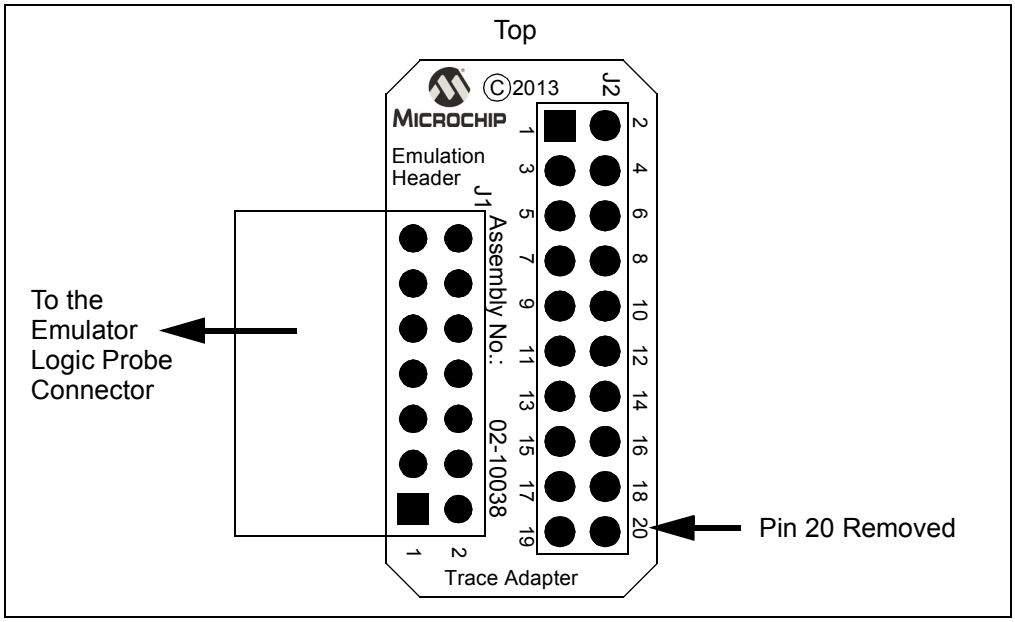


FIGURE 2-4: INTERFACE BOARD



2.4 HARDWARE ADDRESS/DATA BREAKPOINTS

Up to 32 hardware address/data breakpoints are available to use on the emulation header. Software breakpoints are useful, but real hardware breakpoints are incomparable when you need unfettered control of qualifying the breakpoint/event conditions (beyond the simple address matching). Consider the addition of a special interrupt contextual qualifier and these hardware breakpoints offer a high degree of configurability.

Some notable breakpoint features are listed here:

- 32 available address/data hardware breakpoints
- Address range breakpoints (break within data or program memory address ranges)
- Data-Masking qualifier for data breakpoints (allows bit-field breakpoints)
- Data-Comparison qualifier for data breakpoints (equality to)
- Interrupt Context qualifier for address/data breakpoints –
  Select from:
  - Always break (break in both ISR and main code)
  - Break in main line (non-interrupt) context only - break in main code only
  - Break in interrupt context only - break in ISR code only
- Trigger Out qualifier for address/data breakpoints (generate a trigger out signal on event condition) –
  Select from:
  - Do not trigger out when breakpoint is hit
  - Trigger out when breakpoint is hit
- Pass Count qualifier for address/data breakpoints (break on event condition occurring N times)
- Data breakpoints trigger on both normal and linear address modes
- Breakpoints and other events can trigger without halting execution and could be used as trigger events to other features

Address/Data Breakpoints may be found and set up on the New Breakpoint Dialog (*Debug>New Breakpoint*) by choosing either "Address" or "Data" as the "Breakpoint Type". After the breakpoint is created, it can be edited by right clicking and selecting "Customize".

2.5 ENHANCED EVENT BREAKPOINTS

For a definition of event breakpoints, see the MPLAB X IDE Help, “New Breakpoint Dialog”. When creating a new breakpoint or customizing an existing breakpoint using an emulation header, additional actions are available for event breakpoints:

Action	Description
Break	break (halt) execution per option specified
Trigger out	emit a trigger out pulse per option specified
Break and trigger out	break (halt) execution AND emit a trigger out pulse per option specified

Event breakpoints may be found and set up on the New Breakpoint Dialog (*Debug>New Breakpoint*) by choosing “Event” as the “Breakpoint Type”. After the breakpoint is created, it may be edited by right clicking and selecting “Customize”.

2.6 BACKGROUND DEBUG MODE

The emulation header’s on-board -ME2 device contains a Background Debug Mode control interface that allows you read/write access to RAM memory, SFRs, and emulation registers while your program is running or even sleeping.

Background Debug Mode capability includes the following advantages:

- Allows runtime/sleep time changes of In-Circuit Debug (ICD) settings and breakpoints (i.e., runtime address/data/complex/event breakpoints).
- Compared to debug headers (with -ICE or -ICD devices), yields noticeably faster single-stepping speeds at lower MCU operating frequencies.

2.7 EVENT COMBINERS

An event combiner monitors multiple event inputs (currently breakpoints only) and can generate a halt or a trigger out that is based on combinations and sequences of those inputs.

Emulation headers have four (4) Event Combiners and each combines eight (8) combinational or sequential events into a single event trigger. Individual breakpoints may be grouped into sequences, logical 'AND' lists, or a nested combination of these for more complex control. The Event Combiners were modeled after the legacy MPLAB® ICE 2000 complex triggers.

Set up the following complex breakpoints through selections in the Breakpoint window (*Window>Debugging>Breakpoints*).

- Complex Breakpoint Sequence
- Complex Breakpoint Latched-And
- Complex Breakpoint Nesting

2.7.1 Complex Breakpoint Sequence

A breakpoint sequence is a list of breakpoints that execute but do not halt until the last breakpoint is executed. Sequenced breakpoints can be useful when there are more than one execution path leading to a certain instruction, and you only want to exercise one specific path.

To create a Breakpoint Sequence:

1. Right click on an existing breakpoint or shift click to select a group of existing breakpoints and right click on the group.
2. From the pop-up menu, go to "Complex Breakpoint" and select "Add to New Sequence".
3. Enter a name for your sequence in the dialog box, and click **OK**.
4. The breakpoint(s) will appear under the new sequence.

To add Existing Breakpoints to a Sequence:

1. Right click on an existing breakpoint or shift click to select a group of existing breakpoints and right click on the group.
2. From the pop-up menu, go to "Complex Breakpoint" and select "Move to *Name*", where *Name* is the name of the sequence.

To add a New Breakpoint to a Sequence:

Right click on the sequence and select "New Breakpoint".

For more on setting up a new breakpoint, see **Section 2.4 "Hardware Address/Data Breakpoints"** and **Section 2.5 "Enhanced Event Breakpoints"**.

To select the Sequence Order:

1. Expand on a sequence to see all items.
2. Right click on an item and select *Complex Breakpoints>Move Up* or *Complex Breakpoints>Move Down*. Sequence execution of breakpoints is bottom-up; the last breakpoint in the sequence occurs first.

To remove Breakpoints from a Sequence:

1. Right click on an existing breakpoint or shift click to select a group of existing breakpoints and right click on the group.
2. From the pop-up menu, go to "Complex Breakpoint" and select "Remove from *Name*", where *Name* is the name of the sequence.

2.7.2 Complex Breakpoint Latched-And

In addition to breakpoint sequences, a Latched-And (hardware AND) is available to AND a list of breakpoints. ANDed breakpoints can be useful when a variable is modified in more than one location and you need to break only when that variable is modified in one particular location.

To create a Breakpoint Latch-And:

1. Right click on an existing breakpoint or shift click to select a group of existing breakpoints and right click on the group.
2. From the pop-up menu, go to “Complex Breakpoint” and select “Add a New Latched-And”.
3. Enter a name for your Latched-And in the dialog box and click **OK**.
4. The breakpoint(s) will appear under the new Latched-And.

To add Existing Breakpoints to a Latch-And:

1. Right click on an existing breakpoint or shift click to select a group of existing breakpoints and right click on the group.
2. From the pop-up menu, go to “Complex Breakpoint” and select “Move to *Name*”, where *Name* is the name of the Latched-And.

To add a New Breakpoint to a Latch-And:

Right click on the Latched-And and select “New Breakpoint”.

To remove Breakpoints from a Latch-And:

1. Right click on an existing breakpoint or shift click to select a group of existing breakpoints and right click on the group.
2. From the pop-up menu, go to “Complex Breakpoint” and select “Remove from *Name*”, where *Name* is the name of the Latched-And.

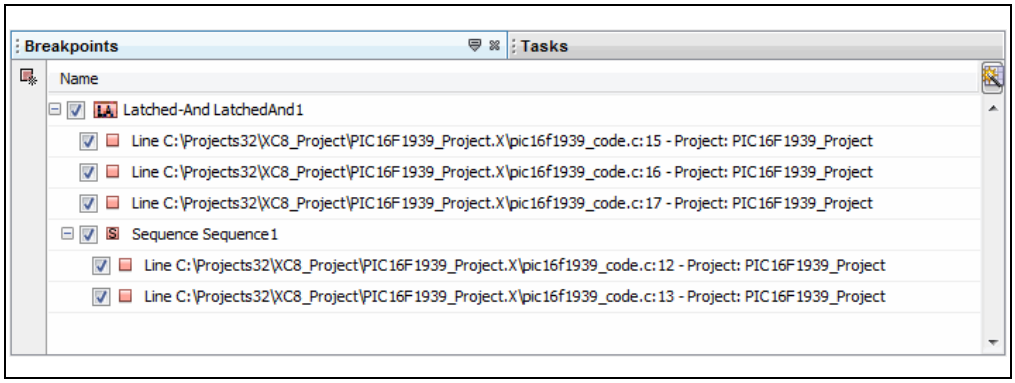
2.7.3 Complex Breakpoint Nesting

Complex breakpoints may be nested to create even more complex breaking schemes.

To nest one group of complex breakpoints into another:

1. Create two groups of complex breakpoints (Sequenced, Latched-And or one of each).
2. Right click on the complex breakpoint group you wish to nest.
3. From the pop-up menu, go to “Complex Breakpoint” and select “Move to *Name*”, where *Name* is the name of the other complex breakpoint group.
4. The first group will appear under the second group, thus creating a scheme.

FIGURE 2-5: NESTED COMPLEX BREAKPOINTS



2.8 STOPWATCH CYCLE COUNTER

The Stopwatch (32-bit Instruction) Cycle Counter has the ability to perform the existing basic instruction cycle counting (all instruction cycles counted) that exists on standard Enhanced Midrange parts.

> **Note:** The count units are in instruction cycles, not in instructions (as not all instructions execute in a single cycle).

The stopwatch is available under *Window>Debugging>Stopwatch*.

2.9 EXECUTION OUT-OF-BOUNDS DETECTION

An emulation header can be used to detect out-of-bounds execution of code. Out-of-bounds code execution is detected by an event breakpoint that watches for PC values that exceed the available program memory of the emulated MCU. The out-of-bounds code execution condition is typically caused by a computed GOTO or CALL that erroneously computes the index, or by loading PCLATH with an incorrect value. Once code is halted due to the execution out-of-bounds event breakpoint the 'Previous PC' functionality can be used to identify the offending instruction.

The Out-of-bounds break option may be found and set up on the New Breakpoint Dialog (*Debug>New Breakpoint*) by choosing "Event" as the "Breakpoint Type" and checking "Break on execution out of bounds". After the breakpoint is created, it may be edited by right clicking and selecting "Customize".

See also, **Section 2.5 "Enhanced Event Breakpoints"**.

2.10 INTERRUPT CONTEXT DETECTION

An address or data breakpoint can be set based on the context of an interrupt. You can set up the breakpoint so it only breaks when it is in the interrupt section of code (ISR), only when it is in main line code, or when it is in either ISR or main code. This can assist when attempting to narrow down issues in code regions.

Address/Data Breakpoints may be found and set up on the New Breakpoint Dialog (*Debug>New Breakpoint*) by choosing either "Address" or "Data" as the "Breakpoint Type". After the breakpoint is created, it may be edited by right clicking and selecting "Customize".

See also, **Section 2.4 "Hardware Address/Data Breakpoints"**.

2.11 TRIGGER IN/OUT

The emulation header is capable of producing an output pulse for external triggering and detecting an input pulse for internal triggering.

2.11.1 Trigger In/Out Hardware

Pins on the emulation header may be used for Trigger In/Out. Pin functions are labeled on the board silkscreen, namely:

- GND – ground
- TRIG IN – used for Trigger In
- TRIG OUT – used for Trigger Out

2.11.2 Trigger In Operation

A pulse on the Trigger In (TRIG IN) pin can be used to generate a trigger condition, halt and/or trigger out signal. Set up Trigger In by selecting *Window>Debugging>Triggers*.

Selection	Description
Polarity	Trigger-in pin pulse polarity: Positive: a positive-going pulse Negative: a negative-going pulse
Noise Reduction Filter	Reduce the noise on the trigger-in pin using a filter, which helps mitigate spurious triggers from halting code. Enable or disable this feature.
Trigger Trace	Trigger trace when a pulse is received on the trigger-in pin. Enable or disable this feature.

An Event Breakpoint can be set up to break when a Trigger In pulse is detected.

See **Section 2.5 “Enhanced Event Breakpoints”**.

2.11.3 Trigger Out Operation

A pulse can be output on the Trigger Out (TRIG OUT) pin based on a setting of either a Program or Data Breakpoint (see **Section 2.4 "Hardware Address/Data Breakpoints"**). The breakpoint may be set up so that the pulse is emitted without halting device execution.

Set up Trigger Out by selecting *Window>Debugging>Triggers*.

Selection	Description
Polarity	Trigger-in pin pulse polarity: **Positive:** a positive-going pulse **Negative:** a negative-going pulse
Slew Rate Limiting	Limit the slew rate on the Trigger Out pulse. **Enable:** slew rate is slow and limited **Disable:** Slew rate is as fast as possible
One Shot	The trigger out pulse duration is: **Enable:** a fixed width, regardless of MCU operating frequency **Disable:** the length of the event itself, which is MCU frequency-dependent
Force Trigger Out	Click this button to force a Trigger Out pulse.

A handy feature of the trigger out signal is that its duration can last as long as the occurring event. For example, if the customer needs to time the duration of the watchdog timer (whose timeout period is user-programmable); the following code, in conjunction with the trigger out and SLEEP event breakpoint features, can make timing an event very simple without employing the old-school technique of writing a single line of (special, non-production) code to wiggle an I/O pin.

The following code is an example:

```
    ; ------------------------------------------------------
    ;   T R I G G E R   O U T   T E S T :
    ;   T I M I N G   T H E   W A T C H D O G   T I M E R
    ; ------------------------------------------------------

    ; 1) Ensure the watchdog timer configuration bit is enabled.
    ; 2) Set an Event Breakpoint (Break on SLEEP) to initiate a
    :    'Trigger out' action only.
    ; 3) Connect your oscilloscope probe to the TRIGGER OUT pin and
    ;    set up your oscilloscope to trigger on the rising edge.
    ; 4) Run the following code:

    CLRWDT
    NOP
    NOP
    SLEEP

    ; The expiration if the watchdog timer will wake up the MCU from
    : sleep after ~2 seconds. The trigger out pulse high-time
    ; duration is the duration of the watchdog timer: ___---___
    ; Since the default watchdog timer period value is 2 seconds
    : typical, the trigger out pulse measured on the oscilloscope
    ; should be approximately 2 seconds.
    NOP
    NOP
    NOP
Loop_101:
    BRA         Loop_101
    ; ------------------------------------------------------
```

Chapter 3. Emulation Header List

3.1 INTRODUCTION

Currently available emulation headers and their associated -ME2 devices are shown below, organized by supported device.

TABLE 1: OPTIONAL DEBUG HEADERS - PIC12/16 DEVICES

Device	Pin Count	Header Part Number	-ME2 Device Used	VDD Max
PIC16F1782 PIC16F1783 PIC16F1784 PIC16F1786 PIC16F1787 PIC16F1788 PIC16F1789	28	AC244064	PIC16F1789-ME2	5.5V
PIC16LF1782 PIC16LF1783 PIC16LF1784 PIC16LF1786 PIC16LF1787 PIC16LF1788 PIC16LF1789	28	AC244064	PIC16F1789-ME2	3.6V
PIC12F1822 PIC12F1840 PIC16F1823 PIC16F1824 PIC16F1825 PIC16F1826 PIC16F1827 PIC16F1828 PIC16F1829 PIC16F1847	8 8 14 14 14 18 18 20 20 18	AC244063	PIC16F1829-ME2	5.5V
PIC12LF1822 PIC12LF1840 PIC16LF1823 PIC16LF1824 PIC16LF1825 PIC16LF1826 PIC16LF1827 PIC16LF1828 PIC16LF1829 PIC16LF1847	8 8 14 14 14 18 18 20 20 18			3.6V

TABLE 1: OPTIONAL DEBUG HEADERS - PIC12/16 DEVICES (CON'T)

Device	Pin Count	Header Part Number	-ME2 Device Used	VDD Max
PIC16F1933 PIC16F1934 PIC16F1936 PIC16F1937 PIC16F1938 PIC16F1939	28 40/44 28 40/44 28 40/44	AC244055	PIC16F1939-ME2	5.5V
PIC16LF1933 PIC16LF1934 PIC16LF1936 PIC16LF1937 PIC16LF1938 PIC16LF1939	28 40/44 28 40/44 28 40/44			3.6V

3.2 AC244055

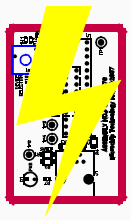
3.2.1 Header Identification

The header AC number is used for ordering the header. However, this number is not on the emulation header, as the board is often used for multiple headers by inserting different -ME2 devices. To identify this debug header, use the following information.

Header	-ME2 Device	Board Assembly Number
AC244055	PIC16F1939-ME2	02-10039

3.2.2 Header Setup and Operation

For this header, you will need to select your device type using J5.



CAUTION

Header Damage.
Selecting the wrong type of device may cause device damage, especially when Vdd is greater than 3.6 V (with respect to Vss).

You can also enable the Power LED by using J3. J6 is not a jumper but a group of pins that you can use for trigger in and/or trigger out signals.

Jumper	Name	Setting	Function	Notes
J3	LED Enable	Open	disable power LED indicator	
		Short	enable power LED indicator	
J5	Emulation Select	pins 1-2	emulate 'F' variants (5.5 V MAX)	Note **1**
		pins 2-3	emulate 'LF' variants (3.6 V MAX**!**)	
J6	Trigger I/O	pin 1	Ground	Note **2**
		pin 2	used for Trigger In	
		pin 3	used for Trigger Out	
Note 1:	Do not change this setting when the header is powered. Power down first.			
Note 2:	See **Section 2.11 "Trigger In/Out"**.			

3.2.3 Header Limitations

See the "Limitations" section in your debug tool online Help file for details.

Additionally, the following silicon errata applies to this header:

Hardware Breakpoint Issue

When using this emulation header with C language or Assembly language code, hardware breakpoints will not function past memory locations 0x3FF for the following emulated devices:

- PIC16F1936, PIC16LF1936
- PIC16F1937, PIC16LF1937

Workarounds:

1. Software breakpoints for these four parts could be used (with the caveat that software breakpoints do not offer the advanced capability of hardware breakpoints).
2. Any function(s) that need to be debugged with hardware breakpoints could be explicitly (and temporarily) located in the program memory area below 0x400 until the function is properly debugged.

   In C language for the MPLAB XC8 C compiler, this can be done with the `@ address` construct as follows, which will locate `function_1` at program memory base address `0x200`.

```
void function_1(void) @ 0x200
{
    asm("NOP");
    asm("NOP");
    asm("NOP");
}
```

   In assembly language for the MPASM assembler, this can be done with the `CODE` directive as follows, which will locate `function_1` at program memory base address `0x200`.

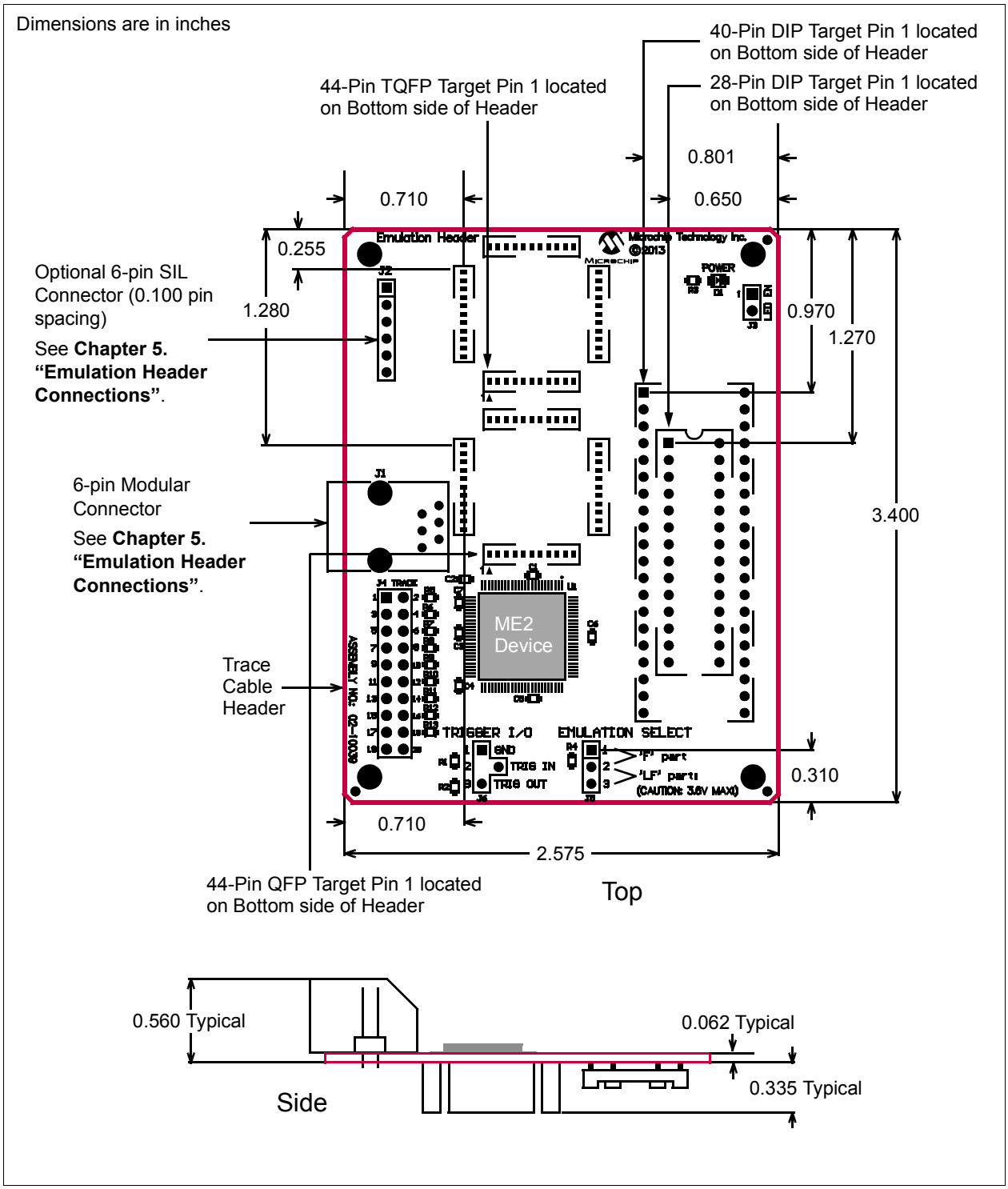
```
HW_BRKPT CODE 0x0200

function_1:

    NOP
    NOP
    NOP

    RETURN
```

3.2.4 Header Dimensions

The figure below lists the dimensions for the emulation header. Dimensions are design values in inches.

If the length and/or width of the emulation header is too large a footprint for the target board, consider using stand-offs, single in-line pins, transition sockets or other extenders in the header connection socket to raise the header above the target.

FIGURE 3-1: DIMENSIONS - AC244055



3.3 AC244063

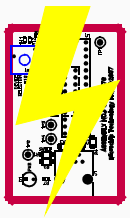
3.3.1 Header Identification

The header AC number is used for ordering the header. However, this number is not on the emulation header, as the board is often used for multiple headers by inserting different -ME2 devices. To identify this debug header, use the following information.

Header	-ME2 Device	Board Assembly Number
AC244063	PIC16F1829-ME2	02-10212

3.3.2 Header Setup and Operation

For this header, you will need to select your device type using J5.



CAUTION

Header Damage.
Selecting the wrong type of device may cause device damage, especially when Vdd is greater than 3.6 V (with respect to Vss).

You can also enable the Power LED by using J3. J6 is not a jumper but a group of pins that you can use for trigger in/trigger out signals.

Jumper	Name	Setting	Function	Notes
J3	LED Enable	Open	disable power LED indicator	
		Short	enable power LED indicator	
J5	Emulation Select	pins 1-2	emulate 'F' variants (5.5 V MAX)	Note **1**
		pins 2-3	emulate 'LF' variants (3.6 V MAX!)	
J6	Device Package	pins 1-2	18-pin devices	Note **1**
		pins 2-3	8/14/20-pin devices	
J7	Trigger I/O	pin 1	Ground	Note **2**
J8		pin 2	used for Trigger In	
J9		pin 3	used for Trigger Out	
Note 1:	Do not change this setting when the header is powered. Power down first.			
Note 2:	See **Section 2.11 "Trigger In/Out"**.			

3.3.3 Header Limitations

See the "Limitations" section in your debug tool online Help file for details.

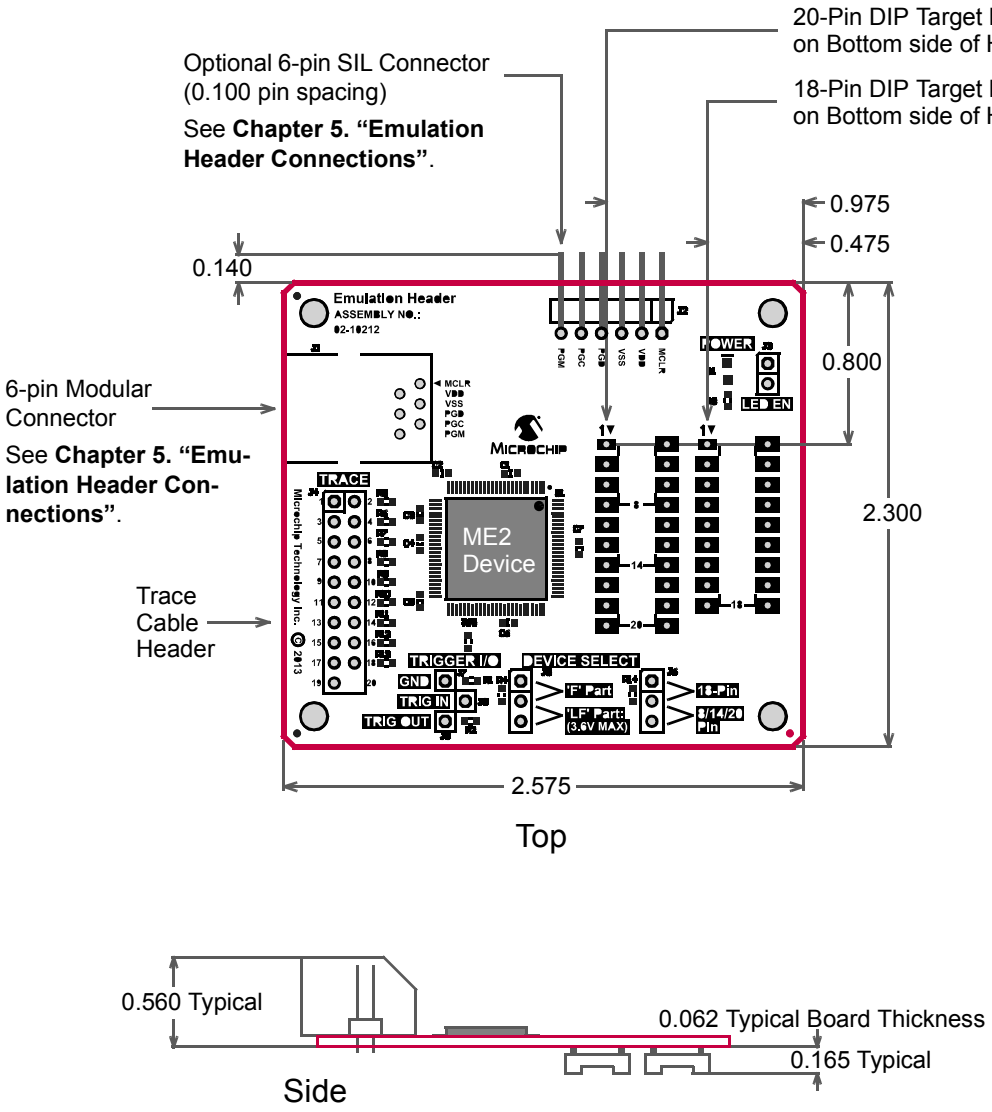
3.3.4 Header Dimensions

The figure below lists the dimensions for the emulation header. Dimensions are design values in inches.

If the length and/or width of the emulation header is too large a footprint for the target board, consider using stand-offs, single in-line pins, transition sockets or other extenders in the header connection socket to raise the header above the target.

FIGURE 3-2: DIMENSIONS - AC244063

Dimensions are in inches



3.4 AC244064

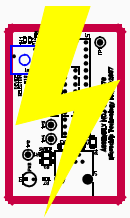
3.4.1 Header Identification

The header AC number is used for ordering the header. However, this number is not on the emulation header, as the board is often used for multiple headers by inserting different -ME2 devices. To identify this debug header, use the following information.

Header	-ME2 Device	Board Assembly Number
AC244064	PIC16F1789-ME2	02-10039

3.4.2 Header Setup and Operation

For this header, you will need to select your device type using J5.



CAUTION

Header Damage.
Selecting the wrong type of device may cause device damage, especially when Vdd is greater than 3.6 V (with respect to Vss).

You can also enable the Power LED by using J3. J6 is not a jumper but a group of pins that you can use for trigger in/trigger out signals.

Jumper	Name	Setting	Function	Notes
J3	LED Enable	Open	disable power LED indicator	
		Short	enable power LED indicator	
J5	Emulation Select	pins 1-2	emulate 'F' variants (5.5 V MAX)	Note **1**
		pins 2-3	emulate 'LF' variants (3.6 V MAX!)	
J6	Trigger I/O	pin 1	Ground	Note **2**
		pin 2	used for Trigger In	
		pin 3	used for Trigger Out	
Note 1:	Do not change this setting when the header is powered. Power down first.			
Note 2:	See **Section 2.11 "Trigger In/Out"**.			

3.4.3 Header Limitations

See the "Limitations" section in your debug tool online Help file for details.

Additionally, the following silicon errata applies to this header:

CCP3 Capture (PIC16(L)F1784/7 only)

When the input threshold control for RE0 is configured for TTL, the CCP3 capture input is ignored.

Work-around: Use ST Threshold.

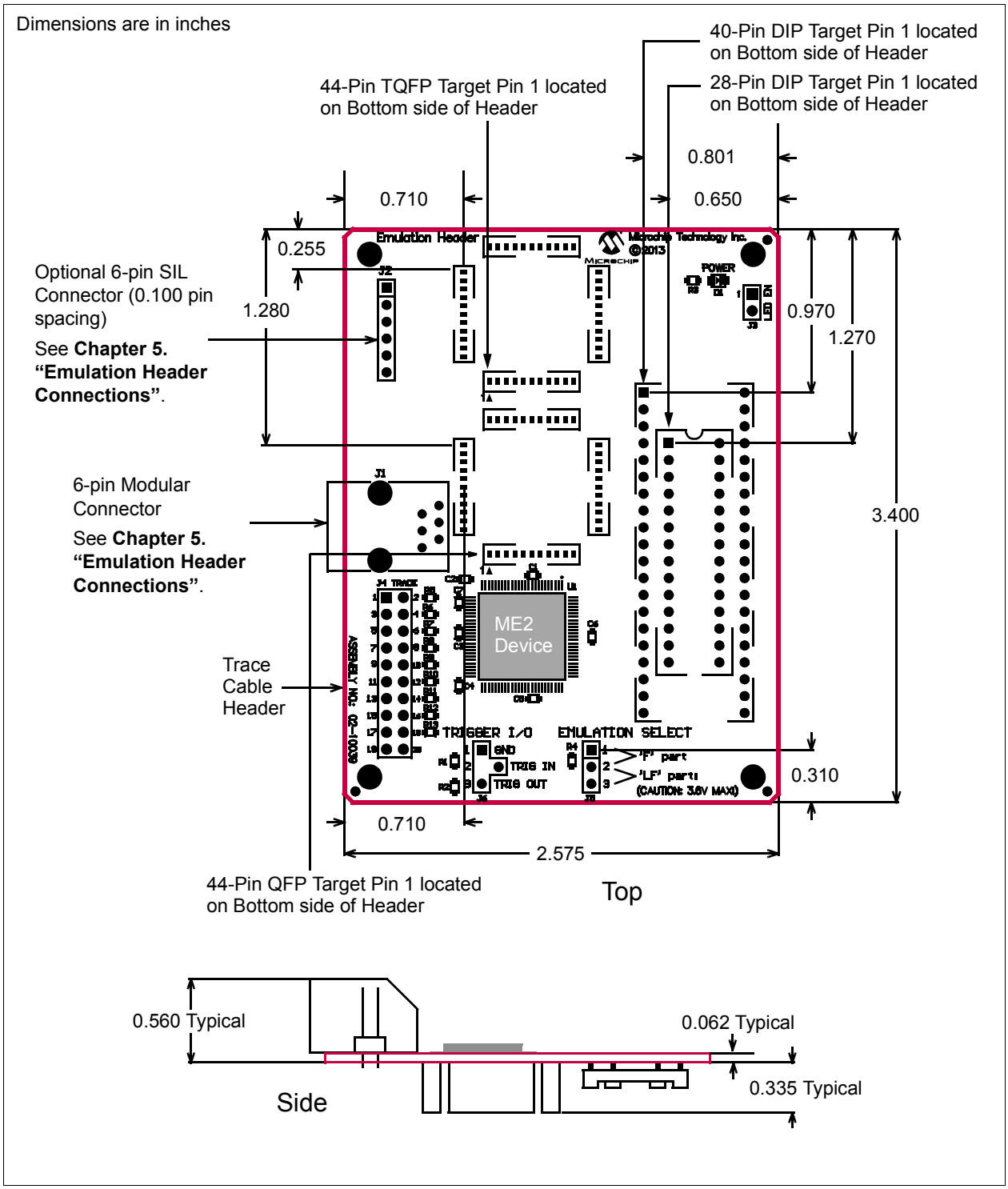
Affected Silicon Revisions: C0

3.4.4 Header Dimensions

The figure below lists the dimensions for the emulation header. Dimensions are design values in inches.

If the length and/or width of the emulation header is too large a footprint for the target board, consider using stand-offs, single in-line pins, transition sockets or other extenders in the header connection socket to raise the header above the target.

FIGURE 3-3: DIMENSIONS - AC244064



NOTES:

Chapter 4. Emulation Header Target Footprints

4.1 INTRODUCTION

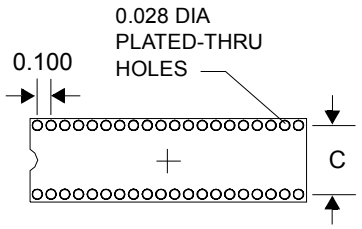
To connect a debug header directly to a target board (without the use of a transition socket) the following information will be helpful.

- DIP Device Footprints
- TQFP/PLCC Device Footprints

4.2 DIP DEVICE FOOTPRINTS

DIP device adapter footprints shown will accept adapter plugs like Samtec series APA plugs. These plugs can be soldered into place during development/emulation and eliminate the need for any other sockets.

FIGURE 4-1: DIP FOOTPRINT



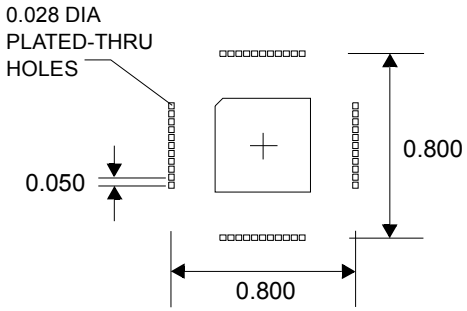
UNLESS OTHERWISE SPECIFIED,
DIMENSIONS ARE IN INCHES.
Drawing of DIP is 40-pin.

DIP	C
8-Pin	0.300
14-Pin	0.300
18-Pin	0.300
20-Pin	0.300
28-Pin	0.300
40-Pin	0.600

4.3 TQFP/PLCC DEVICE FOOTPRINTS

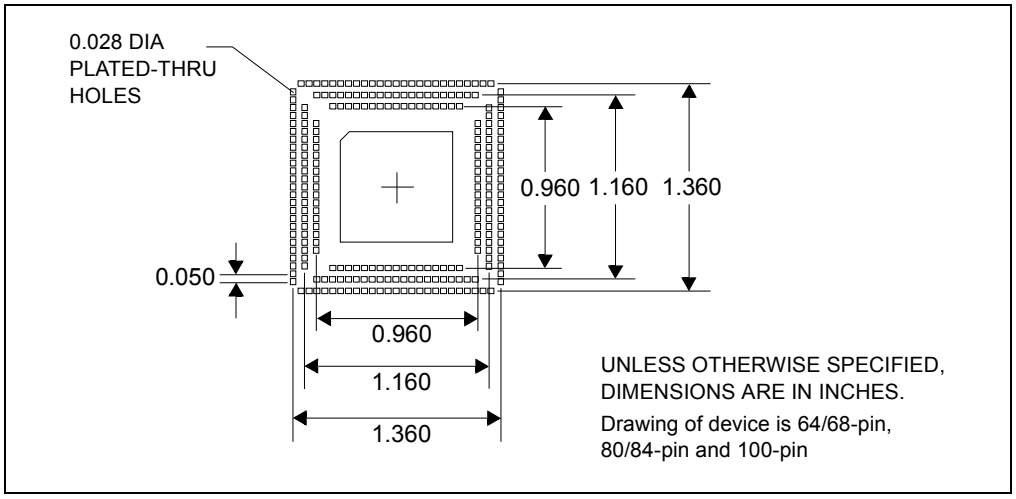
TQFP/PLCC device adapter footprints shown will accept board stackers like Samtec series DWM 0.050 Pitch Stackers. These stackers can be soldered into place during development/emulation and eliminate the need for any other sockets.

FIGURE 4-2: SINGLE-ROW TQFP/PLCC FOOTPRINT



UNLESS OTHERWISE SPECIFIED,
DIMENSIONS ARE IN INCHES.
Drawing of device is 44-pin TQFP/PLCC.

FIGURE 4-3: DOUBLE AND TRIPLE-ROW TQFP/PLCC FOOTPRINT



Header pin-out matches the PLCC package. PLCC will map to TQFP as follows:

Header to 44-pin TQFP – one-to-one mapping.

Chapter 5. Emulation Header Connections

5.1 INTRODUCTION

The different types of emulation header connectors are shown here, as well as information on connecting development tools to the header.

The topics discussed here are:

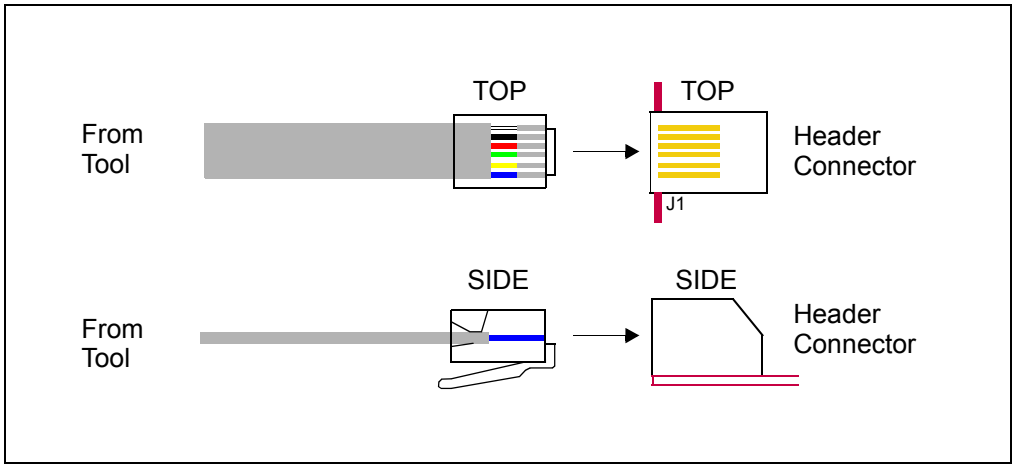
- 6-Pin Modular Connector
- 6-Pin SIL Connector
- Modular-to-SIL Adapter
- Ordering Information

5.2 6-PIN MODULAR CONNECTOR

Debug headers with 6-pin modular (RJ-11/ICSP) connectors can connect directly with the following tools:

- MPLAB REAL ICE in-circuit emulator (Standard Driver Board)
- MPLAB ICD 2 or 3

FIGURE 1: MODULAR CONNECTION



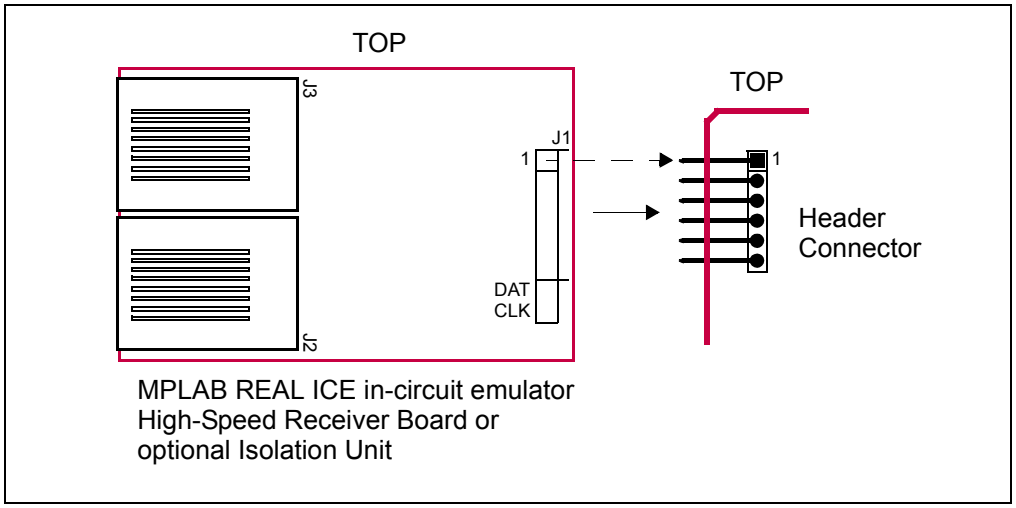
5.3 6-PIN SIL CONNECTOR

Emulation headers with 6-pin SIL (Single In-Line) connectors are compatible with the PICkit 3.

The 6-pin modular cable attached to the Standard Driver Board may be connected to the 6 header pins through the Modular-to-SIL Adapter.

The 8-pin socket of the High Speed Driver Board or optional Isolation Unit may be directly connected to the 6 header pins. Be sure to line up pin 1 on the board with pin 1 on the header.

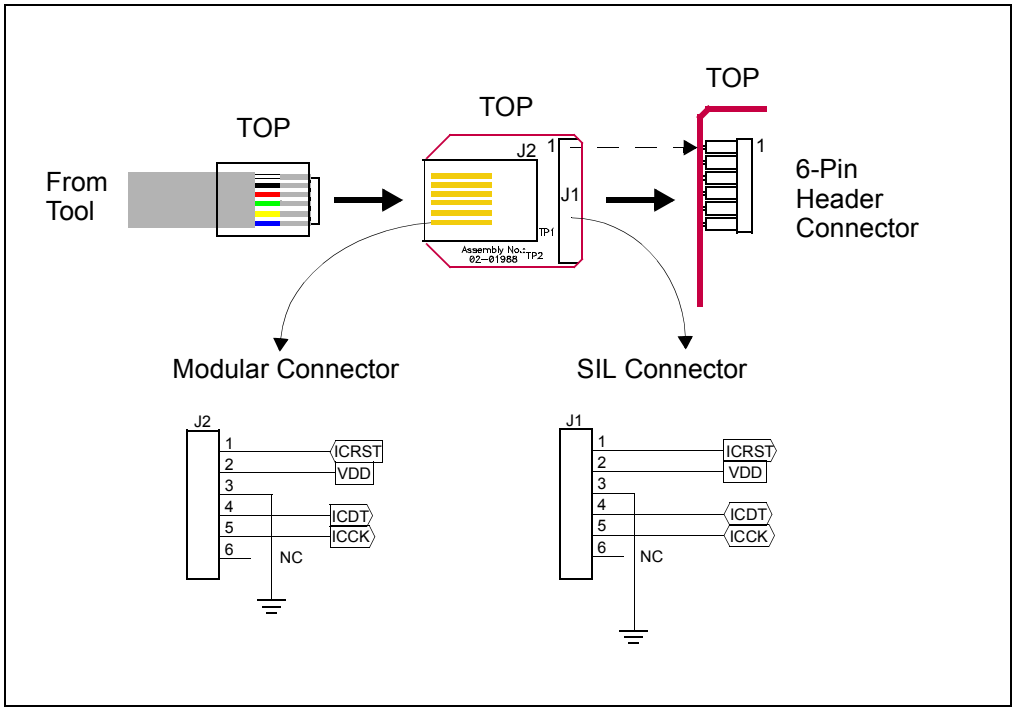
FIGURE 5-1: 6-PIN SIL CONNECTION TO AN EMULATOR



5.4 MODULAR-TO-SIL ADAPTER

You can use this adapter for a 6-pin modular connector to an 6-pin SIL connector. Ensure that you line up pin 1 of J1 with pin 1 of the 6-pin header connector.

FIGURE 5-2: MODULAR-TO-SIL ADAPTER CONNECTION



5.5 ORDERING INFORMATION

To order the development tools and other hardware shown here, please refer to the table below.

TABLE 5-1: MICROCHIP HARDWARE ORDERING NUMBERS

Hardware	Order #
MPLAB REAL ICE in-circuit emulator (Standard Communication)	DV244005
MPLAB REAL ICE in-circuit emulator (High-Speed Communication) – Performance Pak	AC244002
MPLAB REAL ICE Isolation Unit (works with High-Speed Communication)	AC244005
Modular-to-SIL Adapter	AC164110

NOTES:

Index

V

NOTES:

Worldwide Sales and Service

AMERICAS

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

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

Austin, TX
Tel: 512-257-3370

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

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

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

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

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

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

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

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110

Canada - Toronto
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

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

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

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

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

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

China - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

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

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

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

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

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

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

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

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

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

ASIA/PACIFIC

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

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

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

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

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

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

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

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

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

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

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

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

EUROPE

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

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

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

Germany - Dusseldorf
Tel: 49-2129-3766400

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

Germany - Pforzheim
Tel: 49-7231-424750

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

Italy - Venice
Tel: 39-049-7625286

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

Poland - Warsaw
Tel: 48-22-3325737

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

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

10/28/13