Microchip Debugger (MDB) 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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Trademarks

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

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

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

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

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

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

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

♻ Printed on recycled paper.

ISBN: 978-1-63276-064-7

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.

Table of Contents

NOTES:

Preface

NOTICE TO CUSTOMERS

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

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

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

INTRODUCTION

This chapter contains general information that will be helpful to know before using the Microchip Debugger (MDB). Items that are discussed include:

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

DOCUMENT LAYOUT

This document is organized as follows:

- **Chapter 1. How to Use MDB** – describes how to get started with the MDB, invoking the MDB, and the debugging methods available.
- **Chapter 2. MDB Reference** – details the classes of commands and describes the available lists of commands.
- **Appendix A. Revision History** – identifies the changes that have been made to the document.

CONVENTIONS USED IN THIS GUIDE

The following conventions may appear in this documentation:

TABLE 1: DOCUMENTATION CONVENTIONS

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

Square brackets [ ]	Optional arguments	`mpasmwin [options]`
		`file` `[options]`

Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments; an OR selection	`errorlevel {0|1}`
Ellipses...	Replaces repeated text	`var_name [, var_name...]`

	Represents code supplied by user	`void main (void)`
		`{ ...`
		`}`

RECOMMENDED READING

This document describes how to use the MDB. Other useful documents are listed below. The following Microchip presentation and documents are available and recommended as supplemental reference resources.

Microchip Command-line Debugger Webinar

This is a great webinar that gives an introduction to the command-line debugger and provides some useful examples. The webinar is available on Microchip's web site:

www.microchip.com/webinars.microchip.com/WebinarDetails.aspx?dDocName=en565588

Multi-Tool Design Advisory (DS51764)

A small document on guidelines and implementation considerations to ensure proper interfacing to the various development tools.

Processor Extension Pak and Header Specification (DS51292)

This booklet describes how to install and use Processor Extension Paks (PEPs) and related debug headers to better debug selected devices without the loss of pins or resources. See also the PEP and Header online help file.

Transition Socket Specification (DS51194)

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

NOTES:

Chapter 1. How to Use MDB

1.1 INTRODUCTION

The Microchip Debugger (MDB) is a command-line interface to Microchip’s hardware and software development tools. As an alternative to using the Microchip MPLAB® X IDE (Integrated Development Environment) graphical interface, the MDB facilitates programming and debugging devices through a Command Prompt interface.

The MDB is designed for engineers who prefer to use the Command Prompt. The command-line interface to the debuggers is faster and allows more extensive testing to be performed. This is especially helpful when a task is repetitive, such as debugging an issue that is difficult to resolve, or when there is automation of a testing procedure.

The MDB can be used with a script or batch file. The MDB can be used with these tools:

- MPLAB ICD 3 In-Circuit Debugger
- PICkit™ 3 In-Circuit Debugger/Programmer
- MPLAB REAL ICE™ In-Circuit Emulator
- MPLAB PM3 Device Programmer
- MPLAB SIM Software Simulator
- Licensed third party programmers and debuggers

1.2 GETTING STARTED

Install MPLAB X IDE.

The MDB is automatically installed with the MPLAB X IDE. To download the latest version, go to the Microchip web site (www.microchip.com).

Generate a .cof or .elf file for debugging. (If simply programming a device, a hex file is sufficient.) The project can be built with MPLAB X IDE or using third-party compilers, as long as a .cof or .elf file is generated. The .cof/.elf file is a linked executable file that contains symbolic debugging information.

1.3 INVOKING THE MDB

Use the Command Prompt to invoke MDB.

In Windows 7, the Command Prompt must be opened in Administrator mode: *Start>All Programs>Accessories>Command Prompt*, right click and select “Run as Administrator.” This opens the Administrator: Command Prompt.

The path to the MDB may vary depending on where the MPLAB X IDE is installed and which operating system is installed. See the following table for the various operating systems and paths. These paths are long so you may want to add them to your path variable.

TABLE 1-1: PATHS TO THE MDB BY OPERATING SYSTEM

Windows 32-bit Operating System
`c:\Program Files\Microchip\MPLABX\mplab_ide\bin>mdb.bat`
Windows 64-bit Operating System
`c:\Program Files (x86)\Microchip\MPLABX\mplab_ide\bin>mdb.bat`
Linux Operating System
`/opt/microchip/mplabx/mplab_ide/bin/mdb.sh`
Apple OS X
`/Applications/microchip/mplabx/mplab_ide.app/Contents/Resources/mplab_ide/bin/mdb.sh`

Note: The mdb.bat and mdb.sh scripts do not need to be run from the directory where they were installed. If the directory where these scripts are installed is added to the system path, then mdb.bat and msb.sh may run from any directory.

1.4 DEBUGGING METHODS

You can run a test using either of these methods:

- Entering Commands Method
- Running a Command File Method

Entering commands is the preferred method to run a test with MDB. It allows you to interact with the target application as it executes in simulation or on actual hardware. The result of each command is displayed one at a time, so that mistakes are more easily understood and corrected. See “Entering Commands Method”.

The Running a Command File method cannot be used after invoking the MDB. The command file is included as a parameter in the command line when invoking the MDB. See “Running a Command File Method”.

1.4.1 Entering Commands Method

> **Note:** Although the MPLAB X IDE can run multiple tools simultaneously, the MDB will only run one tool at a time. But, you can have multiple instances of the MDB running.
> Refer to **Section 1.6 “Using Multiple Instances of the MDB”**.

Entering commands is a step-by-step method to run a test with MDB. Once the MDB is running, you can start entering commands. Please note that while the MDB commands are not case-sensitive, the property options and file names are case-sensitive.

Type `help` for a list of classes of commands in MDB. Refer to **Chapter 2. “MDB Reference”**, **Table 2-1 “MDB Classes of Commands”**.

For other commands available, see **Chapter 2. “MDB Reference”**, Table 2-2 through Table 2-10.

The following sections describe these topics:

- Programming a Device
- Debugging a Device

1.4.2 Programming a Device

The file or hardware tool you need to use for MDB cannot be active or open simultaneously in the MPLAB X IDE, IPE, or a third party program. Make sure you close (or make inactive) the file or hardware tool before you attempt to use it with the MDB.

> **Note:** When programming a device, you must select a device first.

1. Select the device by entering the command:

   ```
   Device [device name]
   ```

   For example: `Device PIC18F66K22`
2. Use the set command to select any options you wish to use. See **Table 2-6 "Tool-Property-Name Options Used with the Set Command"** or **Table 2-7 "Simulator Options Used With the Set Command"**.
3. Select the hardware tool. To verify the supported tools, type:

   ```
   Help Hwtool
   ```

   The MPLAB ICD 3, MPLAB REAL ICE, PICkit 3 and Simulator are for programming and debugging, while the MPLAB PM3 is for programming only. To select the hardware tool, type the command:

   ```
   Hwtool [tool name]
   ```

   For example: `Hwtool SIM`
4. If the project was already built, a cof or elf file was generated. To program the device with the cof, elf or hex file, enter the command:

   ```
    Program "[location of the cof or elf or hex file]".
   ```

   For example:

   ```
   Program
   "C:\MDBTestExample\Build\test\preprocess\files\dist\
     \test_IO_Button.cof".
   ```

   If you are using SIM (Simulator) as the hardware tool and the project needs an scl file, it can be set up by using the command:

   ```
    Stim "[location of the scl file]"
   ```

   For more information, use the command Help Stim. You can use Stimulus to set pin injection and/or register injection.

A "Program succeeded" message displays after programming is complete. A verify is automatically performed during a programming sequence.

1.4.3 Debugging a Device

You can use the following commands to debug a device.

- MCLR Reset - Refer to the device data sheet for Reset information. If an MCLR Reset is needed for debugging purposes, enter the command:

  ```
  Reset MCLR
  ```

- Target Device Reset - Refer to the device data sheet for Reset information. If you need to reset the target device, first halt the target, then use the command:

  ```
  Help Reset
  ```

  The `Help Reset` command provides information on the usage of the `Reset` Command. It does not actually reset the target.

- Set Breakpoint - There are two ways to set a breakpoint for debugging:
  - Set a breakpoint by source-line-number using the command:

    ```
    Break filename: linenumber
    ```
    For example: `Break main.c:53`

  - Set a breakpoint at an absolute address using command:

    ```
    Break *address
    ```
    For example: `Break *0x108`

- Set Watchpoint - To set a watchpoint for debugging:
  - Set a watchpoint by specifying an address and the type of watch using the command:

    ```
    Watch address breakontype
    ```
    For example: `Watch 0xa0007ff0 R`

    or

    ```
    Watch address breakontype[:value] [passcount]
    ```
    For example: `Watch 0xa0007ff0 R:0xf 1`

- Delete Breakpoint - To delete a breakpoint, use the command:

  ```
  Delete [breakpoint number]
  ```

  If no argument is specified in this command, it will delete all breakpoints.

- Run Program - The `Run` command can be used to run the program until it reaches a breakpoint.
- Step Through - To step through the program, use the `Step` command or `Next` command.
- See Variable Value - A `Print [variable]` command can be used to see the value of a variable or an SFR.
- Exit - Use the `Quit` command to exit the MDB.

1.5 RUNNING A COMMAND FILE METHOD

> **Note:** Although the MPLAB X IDE can run multiple tools simultaneously, the MDB will only run one tool at a time. But, you can have multiple instances of the MDB running. Refer to **Section 1.6 "Using Multiple Instances of the MDB"**.

If programming and debugging needs to be done frequently or multiple times, run the test by running a command file. This will be more efficient than entering the commands repeatedly. Put all the commands in a file and run MDB using this command in the Command Prompt, for example:

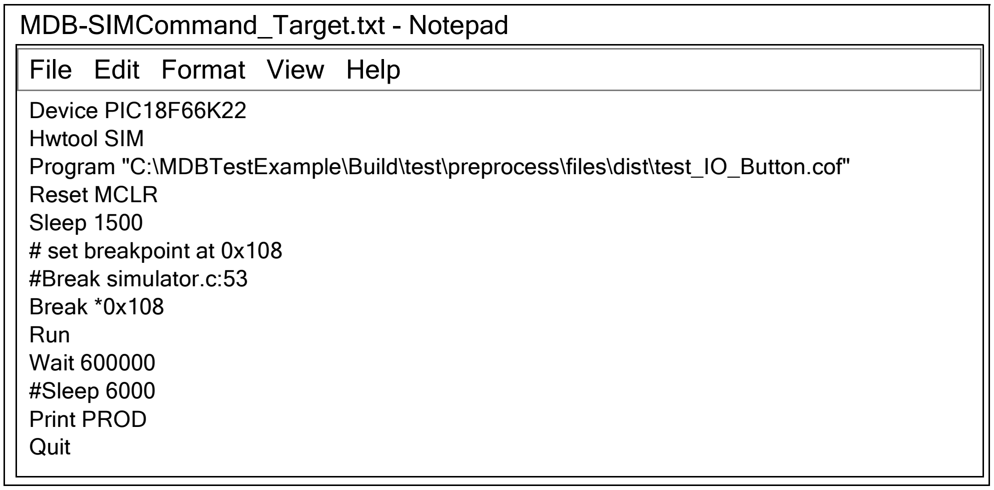
```
C:\Program Files\Microchip\MPLABX\mplab_ide\bin>mdb.bat
<commandfile.txt>
```

The following is an example of a command file:

`C:\MDB-SIMCommand_Target.txt`

A line starting with # means that it is a comment. A `Sleep` command should be added to make sure the MDB has enough time to finish the previous command before it executes the next command. MDB will run all the commands in the command file sequentially.

FIGURE 1-1: EXAMPLE OF RUNNING A COMMAND FILE



MDB-SIMCommand_Target.txt - Notepad
File Edit Format View Help

```
Device PIC18F66K22
Hwtool SIM
Program "C:\MDBTestExample\Build\test\preprocess\files\dist\test_IO_Button.cof"
Reset MCLR
Sleep 1500
# set breakpoint at 0x108
#Break simulator.c:53
Break *0x108
Run
Wait 600000
#Sleep 6000
Print PROD
Quit
```

Creating a Printable Log File

By default, the MDB generates xml log files into the MPLAB IDE binary log directory `<MPLAB X installation location>\mplab_ide\bin\log`. Redirecting output to a file is a general option that can be executed from the command prompt and is not specific to the MDB batch file. Redirecting output to a printable text file can be more useful for examining errors than looking at the Command Prompt window.

To create a printable file, open the `MDB.bat` file, and modify it by adding `>>%mplabx_dir%\bin\mdblog.txt` at the end of the batch file. This instructs the batch file to create the `mdblog.txt` file, which can be printed.

1.6 USING MULTIPLE INSTANCES OF THE MDB

Using multiple instances of the MDB is similar to using multiple instances of the MPLAB X IDE. Some set up is required before using hardware tools (PICkit 3, etc.) with an instance of the MDB. Refer to the MPLAB X IDE online help “Before You Begin”, “Launch Multiple Instances of the IDE” for instructions on setting up the hardware tools and formatting the MCHPDEFPORT file. After any hardware tool setup is complete to assign the appropriate driver for the tool, an instance of the MDB may be invoked from the bin directory of the installation.

NOTES:

Chapter 2. MDB Reference

2.1 CLASSES OF COMMANDS

Type `help` for a list of classes of commands in MDB.

TABLE 2-1: MDB CLASSES OF COMMANDS

breakpoints	Making program stop at certain points
data	Examining/changing data
deviceandtool	Selecting debug tool and device
others	Miscellaneous commands
programming	Programming device and its relative functions
running	Running the program
stack	Examining stack

2.2 LIST OF COMMANDS

> **Note:** MDB commands are not case-sensitive. However, when using the SET command, where tool option properties are passed as parameters, the parameter portion of the command line entered is case-sensitive.

For a list of all commands within a particular class, type `help` followed by the class name. The `help` command can also be abbreviated to h. See the following tables for information about each list of commands.

For documentation on a particular command, type `help [command]` to display information about the command. For example, if you type:

```
help breakpoints or h Breakpoints
```

the MDB displays information about the break, watch, delete and halt command.

The following tables provide information on these commands:

- Breakpoint Commands
- Data Commands
- Device and Tool Commands
- Others Commands
- Tool Property Options Used with the Set Command
- Simulator Options Used with the Set Command
- Programming Commands
- Running Commands
- Stack Commands

2.2.1 Breakpoint Commands

To display information about the breakpoint commands available in MDB, type `help breakpoints`. Table 2-2 provides additional information for this command.

TABLE 2-2: BREAKPOINT COMMANDS

Break	Sets a breakpoint at the specified source line number. **Command format:** `break filename:linenumber [passCount]` ***Example:*** `break newmain.c:142 4`

	Sets a breakpoint at an absolute address.
	Command format: `break *address [passCount]`
	• `address` - the address of the program memory to break on – use the command: `'print /a'` to get a symbol address.
	• `passCount` - the parameter is optional. The number of times the break ‘on condition’ is met before the program halts.
	Example:
	`break *0x9d0000cc 5`
	MDB assigns a breakpoint number and returns:
	`Breakpoint 0 at 0x9d0000cc: file newmain.c, line 16.`

	Sets a breakpoint at the beginning of the function. **Command format:** `break function_name [passCount]` ***Example:*** `break function_foo 5`

Delete	Deletes a breakpoint – if no argument is specified, delete all breakpoints. You can abbreviate this command as d.
	Command format: `delete [breakpoint number]`
	`d [breakpoint number]`
	The breakpoint number is generated by MDB for the break and/or watch commands.
	Examples:
	`delete or D`
	`delete 1` or `d 1`

Halt	Stops the debugged program.

Watch	Sets a data breakpoint at the specified memory address, variable name, or an SFR (special function register):
	Command format: `Watch address breakonType[:value] [passCount]`
	• `address` – the name of a global variable, SFR, or data memory address to be watched. Use command `'print /a'` to get a variable address.
	• `breakonType:`
	`R` -- Read.
	`W` -- Write.
	`RW` -- Read or Write.
	• `value` – this parameter is optional. If it is specified, the program will break only when the value held in the data memory matches the specified value matches the specified value.
	• `passCount` – this parameter is optional. The number of times the breakon condition is met before the program breaks.
	Examples:
	`watch 0xa0007ff0 R:0xf 1`
	`watch 0xa0007ff0 R:10 1`
	`watch my_Variable W 4`
	MDB will assign and return the watchpoint number, for example: `Watchpoint 1`

2.2.2 Data Commands

To display information about the data commands available in MDB, type `help data`. Table 2-3 provides additional information for this command.

TABLE 2-3: DATA COMMANDS

Print	Prints a variable with optional formatting. **Command format:** `print [/f] [/datasize:value] variable` • `/f` - Optional format letter. The format letters supported are: `x` - Print as integer in signed hexadecimal. `d` - Print as integer in signed decimal. `a` - Print the address of a symbol. • `/datasize:value` – optional data size. Variable in assembly code might not have data size information. The user can specify the data size if the .cof or .elf file does not have the size information. The values supported are: `1` - The data size is 1 byte. `2` - The data size is 2 bytes. `4` - The data size is 4 byte.
Stim	Specifies a simulator SCL stimulus file to use. Loads the specified SCL stimulus file into the simulator, or if no path to the file is specified, it clears a loaded file. (Note, if the path or filename has spaces in it, you must use the quotation marks as shown below. If there are no spaces in the path of filename, the quotation marks are not needed.) **Command format:** `Stim “[path to file]”` `Stim`
Write	Use this command to write to memory. **Command format:** `write [/t] addr word1 word2 ... wordn` • `/t` - the type of memory The type of memory is any of: `r` - File Registers (RAM) memory. This is the initial default. `p` - Program (flash) memory. `e` - EE Data memory. Each time you specify a memory type with `write`, that type becomes the default memory the next time you use write. • `addr` - the starting address where you want MDB to begin writing to memory. • `word` - the following values will be written to successive words of memory.

TABLE 2-3: DATA COMMANDS (CONTINUED)

x	Examine memory. You can use the command x (for examine) to examine memory in any of several formats, independent of your program’s data types. **Command format:** `x [/tnfu] [addr]` • `/t` - the type of memory. Each time you specify a memory type with x, that type becomes the default memory the next time you use x.The type of memory is any of the following: `r` File Registers (RAM) memory. This is the initial default. `p` Program (flash) memory. `e` EE Data memory. • n - the repeat count. The repeat count is a decimal integer; the default is 1. It specifies how much memory (counting by units u) to display. • `f` - the display format. The display format is one of the formats used by print (`x, d, o, f, s`), and in addition “`i`” (for machine instructions). The default is ‘x’ (hexadecimal) initially. The default changes each time you use x. • u - the unit size. Each time you specify a unit size with x, that size becomes the default unit the next time you use x. (For the ‘s’ and ‘`i`’ formats, the unit size is ignored and is normally not written.) The unit size is any of: `b` Bytes. `h` Halfwords (two bytes). `w` Words (four bytes). This is the initial default. • `addr` - the starting display address where you want MDB to begin displaying memory. The `addr` can be a literal or a symbol name. The default for `addr`, if not specified, is taken as the value just after the last address examined.

2.2.3 Device and Tool Commands

To display information about the device and tool commands available in MDB, type `help device` or `help hwtool`. Table 2-4 provides additional information for these commands.

TABLE 2-4: DEVICE AND TOOL COMMANDS

Device	Sets the name of the target device. **Command format:** `Device devicename` ***Example:*** `Device PIC32MX795F512L`

Hwtool	Sets the debug tool or list all the available hardware tools on the system. (The device must be set with the `Device` command before a tool can be used/set.)
	Command format: `Hwtool` [toolType] [-p] [index]
	
	Following are the supported tool names (not case-sensitive):
	• `ICD3` – MPLAB ICD 3 In-Circuit Debugger
	• `RealICE` – MPLAB REAL ICE In-Circuit Emulator
	• `PICkit3` – PICkit 3 In-Circuit Debugger
	• `SIM` – Simulator
	• `PM3` – MPLAB PM3 Programmer
	• `LicensedDebugger` – third party debugger
	• `LicensedProgrammer` – third party programmer
	
	To set the tool for programming only, a space must precede the `-p` option.
	Command format: `Hwtool` [toolType] -p
	Example:
	`Hwtool ICD3 -p`
	
	Use the index option to select the tool if there are more than one instance of a tool type. If you have two ICD3 units connected to the PC, use the command `hwtool` to find the assigned index number of the tool.
	Example:
	`>hwtool`
	`index     Description`
	`0        MPLAB ICD 3 tm (MRK1000000000)`
	`1        MPLAB ICD 3 tm (MRK1000001111)`
	
	Example:
	`Hwtool ICD3 -p 1`
	Note: By default, when a hardware tool is selected, it is loaded as a debugger. This means that it always programs the device and adds the necessary debug requirements to enable the image to be debugged.
	To use a tool for programming only, use the -p option when setting the hardware tool.

2.2.4 Others Commands

To display information about the other commands available in MDB, type `help others`. Table 2-5 provides additional information for this command.

TABLE 2-5: OTHERS COMMANDS

Help	`help other` - Prints a list of commands.
Quit	`quit` - Exits the debugger.
Set	The tool property name and value are from the project properties that are selected when creating the project in MPLAB X IDE. **Command format:** `Set tool-property-name tool-property-value` ***Example:*** `Set programoptions.eraseb4program true` Refer to Table 2-6 for other tool properties options that can used with the `Set` command. Refer to Table 2-7 for simulator options that can be used with the `Set` command.
Sleep	Makes the current script processor sleep until specified milliseconds have elapsed. **Command format:** `Sleep milliseconds` ***Example:*** `Sleep 10`
Wait	The `Wait` command makes the current script processor wait until the debugger halts before processing the next command. **Command format:** `Wait` `Wait Milliseconds` makes the processor process the next command if the debugger does not halt and milliseconds have elapsed. **Command format:** `Wait [milliseconds]`
Info	Prints a table of all breakpoints that have been set and not deleted. Optional argument n means print information only about the specified breakpoint. For each breakpoint the following columns are printed: • Breakpoint Numbers • Enabled or Disabled - Enabled breakpoints are marked with `‘y’`. `‘n’` marks breakpoints that are not enabled. • Address - Where the breakpoint is in your program, as a memory address. • What - Where the breakpoint is in the source for your program, as a file and line number. **Command format:** `info breakpoints [n]` `info break [n]`

2.2.4.1 TOOL PROPERTY OPTIONS USED WITH THE SET COMMAND

Table 2-6 provides additional information for the set command used with tool property options.

> **Note:** MDB commands are not case-sensitive. However, when using the SET command, where tool option properties are passed as parameters, the parameter portion of the command line entered is case-sensitive.

TABLE 2-6: TOOL-PROPERTY-NAME OPTIONS USED WITH THE SET COMMAND

Tool Property Name	Value	Tool

`AutoSelectMemRanges`
Determines whether the debugger will automatically select the areas of memory and program memory ranges to program. If set to `auto` the debugger will automatically select the memory and ranges. `Manual` means the memories and ranges will be determined by the memories properties below.
Example:
`set AutoSelectMemRanges auto`
	`auto` or `manual`	MPLAB ICD3, REAL ICE, PICkit 3, MPLAB PM3

`debugoptions.useswbreakpoints`
`True` indicates that software breakpoints will be used for program address breakpoints, `false` indicates that hardware breakpoints will be used (does not apply to PICkit 3).
Example:
`set debugoptions.useswbreakpoints true`
	`true` or `false`	MPLAB ICD3, REAL ICE

`memories.programmemory`
If `true`, the program memory will be programmed; if `false`, it will not.
Example:
`set memories.programmemory true`
	`true` or `false`	MPLAB ICD3, REAL ICE, PICkit 3, MPLAB PM3

`memories.programmemory.start`
The value represents the starting program memory address that the debug tool will begin programming.
Example:
`set memories.programmemory.start 0x0000`
	a string representing a long value	MPLAB ICD3, REAL ICE, PICkit 3, MPLAB PM3

`memories.programmemory.end`
The value represents the ending program memory address that the debug tool will end programming.
Example:
`set memories.programmemory.end 0xFFFF`
	a string representing a long value	MPLAB ICD3, REAL ICE, PICkit 3, MPLAB PM3

`memories.eeprom`
If `true`, the EEPROM memory will be programmed; if `false`, it will not.
Example:
`set memories.eeprom true`
	`true` or `false`	MPLAB ICD3, REAL ICE, PICkit 3, MPLAB PM3

TABLE 2-6: TOOL-PROPERTY-NAME OPTIONS USED WITH THE SET COMMAND (CONTINUED)

`memories.id`
If `true`, the user ID memory will be programmed; if `false`, it will not.
Example:
`set memories.id true`
	`true` or `false`	MPLAB ICD3, REAL ICE, PICkit 3, MPLAB PM3

`memories.bootflash`
If `true`, the boot flash (PIC32 only) memory will be programmed; if `false`, it will not.
Example:
`set memories.bootflash true`
	`true` or `false`	MPLAB ICD3, REAL ICE, PICkit 3, MPLAB PM3

`memories.aux`
If `true`, the auxiliary program memory (dsPIC/PIC24 EP parts only) will be programmed; if `false`, it will not.
Example:
`set memories.aux true`
	`true` or `false`	MPLAB ICD3, REAL ICE, PICkit 3, MPLAB PM3

`programoptions.eraseb4program`
If `true`, the device will be erased before it is programmed; if `false` it will not.
Example:
`set programoptions.eraseb4program true`
	`true` or `false`	MPLAB ICD3, REAL ICE, PICkit 3, MPLAB PM3

`poweroptions.powerenable`
If `true`, the debug tool will supply target power at the default voltage for the tool. If `false` it will not supply target power.
Note: This property does not apply to REAL ICE.
To set a non-default voltage for the target power, first set the `poweroptions.powerenable` to `true`, then set the voltage value where `n.n` represents the desired voltage:
`set voltagevalue n.n`
Example:
`set poweroptions.powerenable true`
`set voltagevalue 3.3`
	`true` or `false`	MPLAB ICD3, PICkit 3, MPLAB PM3

`SecureSegment.SegmentProgramming SegmentProgrammingAll`
If `true`, it permits programming to “Program Over Secure and Protected FLASH”. This property must be set prior to using the program operation on the MDB. Use the `-p` command to set the tool as a programmer if it’s for a production final image and not just a debug image.
Example:
`set SecureSegment.SegmentProgramming SegmentProgrammingAll true`
	`true` or `false`	MPLAB ICD3, REAL ICE, PICkit 3

`system.disableerrormsg`
If `true`, the system will disable warnings and error messages; if `false` the system will enable warning and error messages (this is the default).
Example:
`set system.disableerrormsg true`
	`true` or `false`	Not tool dependent

2.2.4.2 SIMULATOR OPTIONS USED WITH THE SET COMMAND

Table 2-7 provides additional information for the set command used with the simulator options.

> **Note:** For the following table, the break options allow you to set the conditions that will cause program execution to halt. In general, the program will either break on option, ignore the option, or report the option.

TABLE 2-7: SIMULATOR OPTIONS USED WITH THE SET COMMAND

Command name	Values	Device or Runtime Dependent

`breakoptions.coreerrors`
Sets the condition if core errors occur.
Example:
`set breakoptions.coreerrors Break`
	`Break, Ignore, Report`	No

`breakoptions.corewarnings`
Sets the condition if core warnings occur.
Example:
`set breakoptions.corewarnings Ignore`
	`Break, Ignore, Report`	No

`breakoptions.peripheralerrors`
Sets the condition if peripheral errors occur.
Example:
`set breakoptions.peripheralerrors Report`
	`Break, Ignore, Report`	No

`breakoptions.peripheralwarnings`
Sets the condition if peripheral warnings occur.
Example:
`set breakoptions.peripheralwarnings Break`
	`Break, Ignore, Report`	No

`breakoptions.stimulusmessages.notes`
Sets the condition if stimulus notes occur.
Example:
`set breakoptions.stimulusmessages.notes Ignore`
	`Break, Ignore, Report`	No

`breakoptions.stimulusmessags.errors`
Sets the condition if stimulus errors occur.
Example:
`set breakoptions.stimulusmessages.errors Report`
	`Break, Ignore, Report`	No

`breakoptions.stimulusmessags.warnings`
Sets the condition if stimulus warnings occur.
Example:
`set breakoptions.stimulusmessages.warnings Ignore`
	`Break, Ignore, Report`	No

`breakoptions.wdtwarnings`
Sets the condition if watchdog timer warnings occur.
Example:
`set breakoptions.wdtwarnings Ignore`
	`Break, Ignore, Report`	No

`codecoverage.enabled`
Enables or disables code coverage.
Example:
`set codecoverage.enabled Disable`
	`Disable,`
	`Enabled/Reset on POR,`
	`Enabled/Reset on Run`
		No

TABLE 2-7: SIMULATOR OPTIONS USED WITH THE SET COMMAND (CONTINUED)

Command name	Values	Device or Runtime Dependent

`codecoverage.enableoutputtofile`
Enables write to file.
Example:
`set codecoverage.enableoutputtofile true`
	`true, false`	No

`codecoverage.outputtofile`
Absolute path to output file.
Example:
`set codecoverage.outputtofile`
`“c:\path\to\file.txt”`
	`String path`	No

`oscillator.auxfrequency`
Auxiliary PLL Frequency, used by PWM and ADC.
Example:
`set oscillator.auxfrequency 4400`
	`Numeric`	Yes

`oscillator.auxfrequencyunit`
Auxiliary PLL Frequency Units.
Example:
`set oscillator.auxfrequencyunit None`
	`Mega, Kilo, None`	Yes

`oscillator.frequency`
Instruction Execution Frequency.
Example:
`set oscillator.frequency 4700`
	`numeric`	No

`oscillator.frequencyunit`
Instruction Frequency Units.
Example:
`set oscillator.frequencyunit Kilo`
	`Mega, Kilo, None`	No

`oscillator.rcfrequency`
RC Oscillator Frequency.
Example:
`set oscillator.rcfrequency 4500`
	`Numeric`	No

`oscillator.rcfrequencyunit`
RC Oscillator Frequency Units.
Example:
`set oscillator.rcfrequencyunit None`
	`Mega, Kilo, None`	No

`performancedata.show`
Shows performance data in output window.
Example:
`set performancedata.show true`
	`true, false`	No

`periphADC1.altscl`
Use MPLAB 8 style ADC.
Example:
`set periphADC1.altscl true`
	`true, false`	Yes

`periphADC1.minTacq`
Specifies minimum acquisition time (Tacq) in seconds.
Example:
`set periphADC1.minTacq 10`
	`Numeric`	Yes

TABLE 2-7: SIMULATOR OPTIONS USED WITH THE SET COMMAND (CONTINUED)

Command name	Values	Device or Runtime Dependent
periphADC1.tacqunits Units for minimum acquisition time (Tacq). ***Example:*** set periphADC1.tacqunits nanoseconds	milliseconds, microseconds, nanoseconds	Yes
periphADC2.altscl Use MPLAB 8 style ADC. ***Example:*** set periphADC2.altscl true	true, false	Yes
periphADC2.minTacq Specifies minimum acquisition time (Tacq) in seconds. ***Example:*** set periphADC2.minTacq 20	Numeric	Yes
periphADC2.tacqunits Units for minimum acquisition time (Tacq). ***Example:*** set periphADC2.tacqunits milliseconds	milliseconds, microseconds, nanoseconds	Yes
uartNio.output Specifies location of UART output, where N represents the UART number 1 through 6. ***Example:*** set uart1io.output file	file, window	Yes
uartNio.uartioenabled If true, the system will enable the UART I/O; if false the system will disable it. N represents the UART number 1 through 6. ***Example:*** set uart1io.uartioenabled false	true, false	Yes
uartNio.outputfile Passes in a string containing the root (absolute path) of the file system to the file used for UART output. N represents the UART number 1 through 6. ***Example:*** set uart1io.outputfile “c:\path\to\outputfile.txt”	Absolute path to file	Yes

2.2.5 Programming Commands

To display information about the programming commands available in MDB, type `help [programming option]`. Table 2-5 provides additional information for these commands.

TABLE 2-8: PROGRAMMING COMMANDS

Program	Programs device memory with the image specified by the file. (Note, if the path or filename has spaces in it, you must use the quotation marks. If there are no spaces in the path of filename, the quotation marks are not needed, as shown below.) **Command format:** `Program executableImageFile`
Upload	Uploads the executable image to MDB memory. The source of the instructions to be loaded is the contents of the memory of an attached PIC device through the programmer or debugger. **Command format:** `Upload`
Dump	Writes the MDB memory to a hex file. **Command format:** `Dump [-m] filename` The `m` is an optional argument that specifies which memories to write to the hex file. It can be any combination of the following: • p - Program Memory (Flash) • e - EE Data • c - Configuration Bits • u - User ID memory • b - Boot Memory • `f` - Flash Data The `filename` is the full path and name to the hex file.

2.2.6 Running Commands

To display information about the running commands available in MDB, type `help running`. Table 2-5 provides additional information for these commands.

TABLE 2-9: RUNNING COMMANDS

Continue	Resumes program being debugged, after breakpoint. **Command format:** `Continue`
Halt	Stops the debugged program. **Command format:** `Halt`
Next	Step program, proceeding through subroutine calls. Like the "step" command as long as subroutine calls do not happen; when they do, the call is treated as one instruction. **Command format:** `Next`
Run	Start the debugged program. **Command format:** `Run`
Step	Step program until it reaches a different source line. The step command only enters a function if there is a line number information for the function. **Command format:** `Step`

2.2.7 Stack Commands

To display information about the stack commands available in MDB, type `help backtrace`. Table 2-5 provides additional information for these commands.

TABLE 2-10: STACK COMMANDS

Backtrace	Print a backtrace of the entire stack; one line per frame for all frames in the stack. **Command format:** `Backtrace [full] [<n, -n>]` • `full` - prints the values of local variables • n - prints the innermost n frames • `-n` - prints the outermost n frames

NOTES:

Appendix A. Revision History

REVISION HISTORY

Revision A (November 2012) - initial release of this document.

Revision B (April 2013)

- added note in Invoking the MDB section
- added Tool Property Name Options for the Set command
- added Simulator Options for the Set command
- added `-p` option
- added note on running multiple tools
- removed example of using commands to debug a project
- added section on creating a printable log file

Revision C (March 2014)

- relocated Revision History from Preface to it's own appendix.
- added a Document Layout section to the Preface.
- added new **Section 1.6 "Using Multiple Instances of the MDB"**.
- moved reference tables to **Chapter 2. "MDB Reference"**.
- added notes about case-sensitivity for commands in **Chapter 2. "MDB Reference"**.
- added tool column to **Table 2-6: "Tool-Property-Name Options Used with the Set Command"**
- added new **Table 2-7: "Simulator Options Used With the Set Command"**.

NOTES:

Index

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: 852-2943-5100
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: 86-571-8792-8115
Fax: 86-571-8792-8116

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

03/25/14