MIPS Debug OpenOCD with Bus Blaster Probe and WiFIRE Probe-On-Board Getting Started Guide

Filename	:	MIPS Debug OpenOCD with Bus Blaster.Getting Started Guide.docx
Version	:	1.1.23 External Issue
Issue Date	:	31 May 2017
Author	:	MIPS

Contents

1. Introduction to OpenOCD, Bus Blaster, Codescape MIPS SDK Essentials, and the Digilent WiFIRE Board 3
1.1. OpenOCD 3
1.2. Bus Blaster v3c 3
1.3. Codescape MIPS SDK Essentials 3
1.4. Digilent WiFIRE Board 4
Technical Support 5
2. Installing OpenOCD 6
2.1. Installing the Windows USB Driver 6
2.2. Getting OpenOCD Up and Running on Linux 7
3. Installing Codescape MIPS SDK Essentials 8
4. Using OpenOCD 9
4.1. Running OpenOCD 9
4.2. Interfacing with OpenOCD 9
4.2.1. Telnet connection to OpenOCD 10
4.2.2. Common OpenOCD Commands 10
4.2.3. MIPS-specific OpenOCD Commands 11
5. Using GDB with OpenOCD 13
5.1. GDB useful commands 13
5.2. Additional GDB commands 14
5.3. MIPS-specific OpenOCD commands run from gdb ‘mon’ prefix 16
6. Connecting OpenOCD to the Probe-On-Board (POB) of the WiFIRE rev D for Windows 17
6.1. Installing WinUSB Windows Driver for the POB 17
6.2. Running OpenOCD Debug with the WiFIRE board and POB 18
7. Connecting a Bus Blaster probe to a chipKIT Wi-FIRE rev C development board 22

List of Figures

Figure 1 - Digilent Wi-FIRE Rev D board with 2 x 10 x .05” EJTAG/Trace Connector 17
Figure 2 - EJTAG Adapter Board 22
Figure 3 - Flying lead connections on the Wi-FIRE board 23
Figure 4 - Bus Blaster to Wi-FIRE adapter schematic 24

1. Introduction to OpenOCD, Bus Blaster, Codescape MIPS SDK Essentials, and the Digilent WiFIRE Board

This document provides assistance to set up and configure OpenOCD to work with a Bus Blaster (v3c) debug adapter (also called a JTAG probe) to debug MIPS targets.

1.1. OpenOCD

The Open On-Chip Debugger (OpenOCD) provides debugging, in-system programming, and boundary-scan testing for embedded target devices.

It does so with the assistance of a hardware debug adapter which provides an interface to the target being debugged.

Further details can be found at http://openocd.org.

1.2. Bus Blaster v3c

Bus Blaster v3c for MIPS is a high-speed debug adapter designed for supporting JTAG debug with various MIPS processors. It is controlled from a high-speed USB port of a PC and has a 14-pin target connector and interface cable with buffering logic suitable for MIPS EJTAG targets.

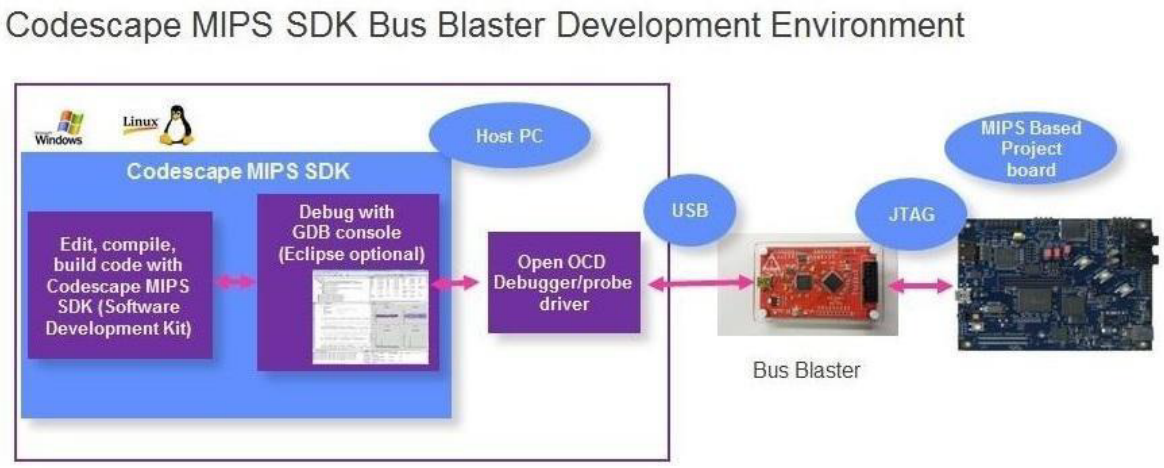
Bus Blaster v3c is available from Seeed Studio:

http://www.seeedstudio.com/depot/Bus-Blaster-v3c-for-MIPS-Kit-p-2258.html

1.3. Codescape MIPS SDK Essentials

The Codescape MIPS SDK Essentials provides tools for software development, compiling, building programs, and source debugging with GDB and is a free download available from:

https://www.mips.com/develop/tools/codescape-mips-sdk/



1.4. Digilent WiFIRE Board

This board is based on the Microchip PIC32MZ microcontroller. It includes a WiFi module, MicroSD card connector, USB 2.0 Hi-speed controller, 2 push buttons, potentiometer, and 4 LEDs.

The Resource Center for this board is available at:

https://reference.digilentinc.com/chipkit_wifire/chipkit_wifire

Technical Support

Technical Support and a community support forum is provided at
https://www.mips.com/forums/cat/mips-insider/

2. Installing OpenOCD

The general IMG webpage for Bus Blaster, OpenOCD, and other low-cost debug tools is located at:
https://www.mips.com/develop/tools/mips-debug-and-trace-probes/bus-blaster/

Scroll down to the OpenOCD section. There is a section “Getting started guide:” which provides a download of this document.

Next is the IMG OpenOCD installer which provides an installer for the latest Windows executable. It is based on the OpenOCD 0.10.0 build. The latest version of the IMG-specific version of OpenOCD is OpenOCD-0.10.0.2-img-installer.exe. There may be a more current release on this web page. This installer provides the option of installing Codescape Essentials, the compiler toolchain, which is prompted.

The Linux version of OpenOCD is provided as source code in a .tgz file format and provided in the “Source code:” section.

Detailed instructions needed to build OpenOCD from source files are available on the OpenOCD website:
http://openocd.org.

2.1. Installing the Windows USB Driver

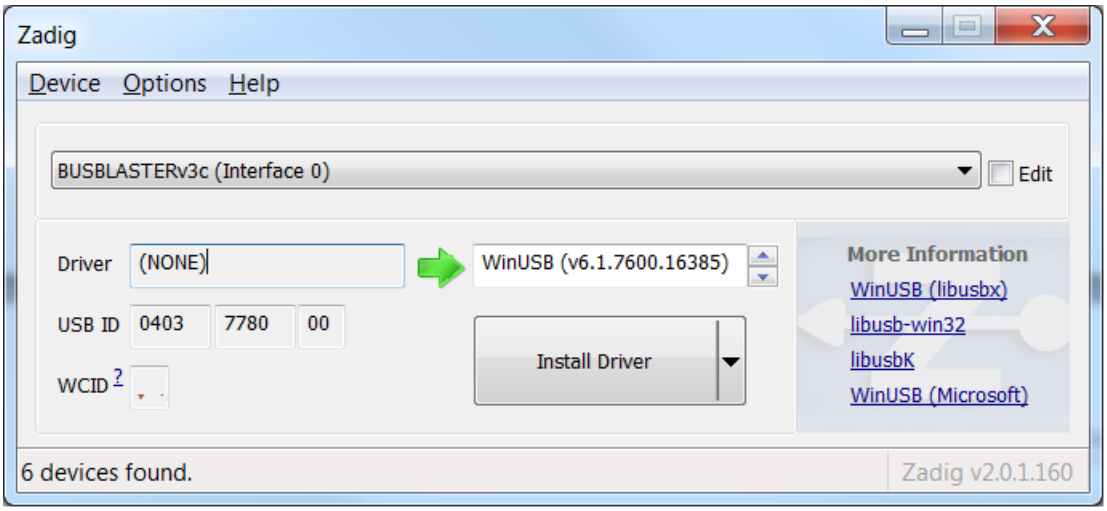
OpenOCD uses an installer program Zadig to install WinUSB, the probe’s Windows USB driver. The screenshots in this section were taken from running the program “zadig_2.2.exe” from the web site http://zadig.akeo.ie/ .

Older versions of the zadig program can be downloaded as a 7z compressed file from:
https://codescape.mips.com/components/probes/openocd/tools/zadig/

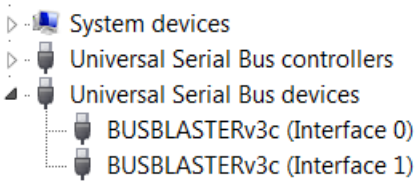
To install WinUSB follow the steps below:

1. Connect the Bus Blaster to your PC using the USB cable. Windows will attempt to locate driver software, cancel this.
2. Open the Zadic.exe installer program.
3. In the top field of the Zadig installer select ‘BUSBLASTERv3c (Interface 0)’.

   WinUSB is shown in the Driver field. If the field is blank and there are no selections when clicking on the down-arrow, click on Options > List All Devices.



4. Click 'Install Driver'.
   A progress bar is shown while the driver installs and confirmation of successful installation is given.
5. Repeat the above steps for '(Interface 1)'.
6. To confirm installation, in Windows 'Device Manager' which can be found in Control Panel > System and Security > System, you should see the entry below for 'Universal Serial Bus devices'.



Note: If you encounter problems with using a USB device with libusb on Windows, you may need to install a libusb device filter. To do so follow these instructions from Tin Can Tools: http://www.tincantools.com/wiki/Libusb_Device_Filter

2.2. Getting OpenOCD Up and Running on Linux

When invoking OpenOCD, include this command line option:

-f interface/mips_busblaster.cfg

mips_busblaster.cfg is the interface script that is matched to the particular VID/PID combination, and uses the open source FTDI driver linked into OpenOCD. That script is not part of upstream OpenOCD yet, but it can be found in the source code release found on

https://www.mips.com/develop/tools/mips-debug-and-trace-probes/bus-blaster/

Also, manually edit /etc/udev/rules.d/99-openocd.rules to include the following line; this is required for device permissions to be set correctly after the device is connected to USB:

ATTRS{idVendor}=="0403", ATTRS{idProduct}=="7780", MODE="664", GROUP="plugdev"

In the future the mips_busblaster.cfg and the rules patch will be included in the OpenOCD release.

3. Installing Codescape MIPS SDK Essentials

Codescape MIPS SDK Essentials provides you with the tools for developing software for MIPS targets including toolchain, QEMU (for Linux), and libraries.

Information about Codescape MIPS SDK Essentials is located at:

https://www.mips.com/develop/tools/codescape-mips-sdk/

You can make selections to install just those components you need for your target MIPS processor and for the type of applications you will be developing. This will speed up installation. During installation follow any instructions given on screen.

4. Using OpenOCD

4.1. Running OpenOCD

The generic command for opening OpenOCD is:

```
<install_path>/openocd.exe –s <path_to_scripts> -f <scriptpath>/<cfg_file1>
 -f <scriptpath>/<cfg_file2> –c "init"
```

When using OpenOCD, open a Windows command window (cmd.exe), enter the path and name of the OpenOCD executable, -s and the path to the scripts, enter one or more -f followed by the path to the specific configuration class and .cfg configuration file names, and -c "init". The `–c "init"` executes the configuration command 'init', which terminates the configuration stage and enters the run stage; it helps when the startup scripts manage tasks such as resetting the target or programming flash etc.

A batch file can be made for starting up OpenOCD. An example for the WiFIRE rev D board follows (see section 9 for more details):

```
set OpenOCD_Path=<install_dir>/OpenOCD-0.10.0-img/scripts
start "OpenOCD" cmd.exe /K openocd.exe –s "%OpenOCD Path%" -f interface/wifire-pob.cfg –f
target/wifire.cfg -c "init"
```

Note that the latest version of OpenOCD may have a different version number.

The order of the configuration files is important. One configuration file inclusion can be dependent on the previous configuration file. This is true for the two configuration files in the example above.

Here is an example running OpenOCD for the MIPS Bus Blaster probe connected to the WiFIRE rev C development board (see section 10 for more details). This example also starts a telnet session which provides a command window to issue OpenOCD commands (see section 5).

```
start "OpenOCD" cmd.exe /K openocd.exe -s "<install dir>/openocd-
0.10.0-img/scripts" -f interface/mips busblaster.cfg -f target/wifire.cfg -c "init"
start "telnet" cmd.exe /K telnet localhost 4444
```

A third method is to create an icon on your desktop and edit the target string to include the parameters as shown above. You can then assign a shortcut keystroke to that icon.

An html-based OpenOCD on-line manual can be found at:
http://openocd.sourceforge.net/doc/html/index.html

4.2. Interfacing with OpenOCD

OpenOCD runs as a daemon. It accepts connections from other programs, but does not provide any means for you to pass commands to it directly. Once OpenOCD is running on your computer you will need to connect to it through another program, such as Telnet. (GDB can also connect to the OpenOCD daemon).

4.2.1. Telnet connection to OpenOCD

Before you can run a Telnet client, it must be enabled in Windows. To do this, navigate to the Windows Features dialog (Control Panel > Programs > Programs and Features > Turn Window features on or off) and enable 'Telnet Client'.

To run Telnet and connect to OpenOCD, open a new command prompt (cmd.exe). From any directory, type:

```
telnet localhost 4444
```

You should see a simple prompt (>). From this prompt you will be able to send commands to OpenOCD. To exit the Telnet prompt, type 'exit' or press Ctrl+c.

At the > prompt enter:

```
> reset halt
```

The mips_busblaster.cfg 'adapter_khz 15000" sets the Bus Blaster adapter speed to a 15MHz TCK rate.

The wifire.cfg 'mips32 scan_delay 1500' line sets the minimum delay for the MIPS EJTAG fast data feature to operate. In some targets the value may need to be higher, for example, 3000.

4.2.2. Common OpenOCD Commands

Command	Description
`reset [run | halt | init]`	run (default) - reset and start the target running. halt - immediately halt and reset the target. init - immediately halt the target and execute reset-init script.
`halt`	Halt target and enter debug mode
`load_image` *`filename address`* `[[bin|ihex|elf|s19]` *`min_address max_length`*`]`	Load image into memory using fast load. *address* is target memory offset from its load address. *min_addr* - ignore data below *max_length* - max number of bytes to load
`resume [`*`address`*`]`	Resume target execution at current PC or optionally set the PC.
`step`	Single step target by one instruction.

`mdw` *`addr`* `[`*`count`*`]`
`mdh` *`addr`* `[`*`count`*`]`
`mdb` *`addr`* `[`*`count`*`]`
	Display memory contents, w=word, h=half, b=byte
	Optional '*count'* parameter for how many to display.

`mww` *`addr value`* `[`*`count`*`]`
`mwh` *`addr value`* `[`*`count`*`]`
`mwb` *`addr value`* `[`*`count`*`]`
	Write word, half, or byte into memory at specified address.
	Optional 'count' parameter for how many to write.

`reg [`*`num|name`* `[`*`val`*`]]`	Show register value by number or name, or change value of register. ex: reg, reg pc, reg r1, reg status, reg cause

Command	Description
bp [*address len* [hw]] rbp *address*	Show bp list or set a bp at address of len bytes. ex: bp 0x80100000 4 hw rbp – remove bp
wp [*address len* [(r|w|a)]] rwp *address*	Show data watchpoint list (also called data breakpoint) or set a wp at *address* of *len* bytes (4 bytes is supported). r=read, w=write, a=access ex: wp 0xbf800000 4 w
version	Display version of OpenOCD server
exit	Exits the current Telnet session.
shutdown	Close the OpenOCD daemon, disconnecting all clients.

4.2.3. MIPS-specific OpenOCD Commands

Command	Description
mips32 cpuinfo	Displays information for the current CPU core.
mips32 cp0 [[*reg_name* | *regnum select*] [*value*]]	Default is to display all cp0 registers *reg_name* – Name of cp0 register to be read or modified. Ex. mips32 cp0 status *regnum* - register number *select* - register select number ex: mips32 cp0 25 0 *value* - optional value to write into the register
mips32 dsp [*regname*] [*value*]	Default is to display all dsp registers. *regname* - name of register (ex: config) read or modified *value* - optional value to write into the register
mips32 dump_tlb [*entry*]	Default is to display all tlbs Command valid at any time *entry* - dump only the specified entry or index

Command	Description
`mips32 invalidate [all | inst | data | allnowb | datanowb]`	Invalidate either or both of the instruction and data caches. The MIPS core does not update the instruction cache if new code is written to memory. Typically, invalidate should be issued after writing to an instruction region of memory. The allnowb and datanowb options will step through the data cache clearing the cache tags. This is useful for initializing the cache before the memory controller is set up. all - writeback data and invalidate both inst and data caches. This is the default. inst - invalidate only the inst cache data - writeback and invalidate only the data cache allnowb - invalidate both inst and data cache without writeback datanowb - invalidate only the data cache without writeback
`mips32 scan_delay [value]`	*value* - delay in nsec between fast data writes; 3000 is typical. When enabled, downloading is faster. If the GDB 'load' command fails, try increasing this number. *value* >= 2000000 - turns off fast data and puts the probe download into legacy mode (which is slower but reliable).
`mips32 semihosting ['enable'|'disable']`	Activate support for semi-hosting operations. Command valid at any time.

5. Using GDB with OpenOCD

The OpenOCD command window is used primarily to get the probe connection working with the target. Once that is established, GDB is a more user-friendly debug environment, supporting symbolic debugging which includes viewing source lines, setting breakpoints on line numbers, stepping a line at a time, referencing function and variable names, and having the ability to view code in disassembled instruction format. Once GDB can be started up and run, there is little need to start up the OpenOCD telnet window because OpenOCD commands can be issued in the GDB command window by prefixing them with 'mon', short for 'monitor'.

Before starting GDB, here are several tips on GDB initialization. Please refer to GDB documentation for further details:

- GDB will read and execute the '.gdbinit' file during start-up if it is in the current working directory or in your home directory. The home directory on Windows is pointed to by the HOME environment variable.
- An alternative to creating '.gdbinit' is to specify an initialization file when starting GDB and use the -x option, for example: '**mips-mti-elf-gdb -x startup.txt**'
- the -ix option executes the commands in the file before loading the inferior (an inferior is an object in GDB that represents the state of each program execution).
- the -nx option prevents init files from being executed
- the -q (quiet) option will suppress intro messages
- To display the list of init files loaded by GDB at start-up (shown at the end of the output) use the –help switch

To connect GDB to OpenOCD use the following commands:

```
mips-mti-elf-gdb <elf file>
target remote localhost:3333
set endian little
```

These commands assume your path environment variable includes the path <install_dir>\
\Toolchains\mips-mti-elf\<version>\bin\.

A handy gdb command reference card is available in <install_dir>\Documentation directory.

5.1. GDB useful commands

Command	Description
Ctrl-c	gdb command to halt the target processor.
monitor reset halt	Reset and stop the processor. Notice the program stop running. Note: the gdb 'monitor' command passes the 'reset halt' text through to the OpenOCD command parser which executes the reset command. **Shortcut**: mo reset halt

Command	Description
b main	Set a breakpoint at the main function. (Short for: "break main".)
b *0x80000330	Set a breakpoint at instruction address 0x80000330.
i b	List the breakpoints. (Short for: "info breakpoint").
`c`	Continue the processor execution. (Short for: "continue".) It will stop at the first breakpoint, in this case, when it gets to `main`.
`c`	Continue to the next break point. (You can also simply press enter to repeat the last command.)
`p count`	Print the value of the variable `count`. (Short for: "print count".) For example, `count` is now 15.
`p/x count`	Prints the value of the `count` variable in hexadecimal (0xf).
`p/x &count`	Prints the address of count (for example, 0x8003ffd4).
`i r`	Print the value of all registers. (Short for: `info registers`.)
`i r v0`	Print the value of register v0 only.
`i r s0`	Print the value of register s0.
`stepi`	Executes a single instruction. (Type p/x $pc to print the value of the program counter or `i r pc`) **Shortcut:** `si`
`d 1`	Delete breakpoint 1 (type `i b` to list the breakpoints with their numbers)
`monitor reset run`	Reset and run the processor. This will run the processor without breakpoints, even if breakpoints have been set. **Shortcut:** `mo reset run` Note that gdb and OpenOCD may get out of sync with the target processor state. You may need to issue the continue command (`c`) to put gdb in the run state.

5.2. Additional GDB commands

Command	Example / Description
`load <elf_file_name>`	**Example 1:** `load program.elf` **Description:** Load an executable file (in ELF format) into the MIPS target. **Note:** the processor must be halted (Ctrl-c) before loading a new ELF file. After loading the executable, then run it by typing `c`. If `Ctrl-c` does not halt the target, it may be because gdb and OpenOCD are out of sync. Try '`c`' before `Ctrl-c`.

Command	Example / Description
`disas`	Disassemble instructions. **Examples:** `disas` or `disas (*$pc)` (disassemble from program counter) `disas 0xbfc00000,+100` (+100 is length in bytes) `disas /m main` (disassemble mixed source/assembly of the *main* function) `disas /r main` (show raw bytes as well as instructions) **Note:** If the above commands don't work (for example return `nops`), first halt the processor (Ctrl-c) and then enter the program: type `mo reset halt`, set a breakpoint at main (`b main`), and continue to that point (`c`). Then use the above commands.

`x/i 0xbfc00000`
`x/<n>i $pc`
 	
`x/16w $sp-0x10`
	Examine instruction – similar to `disas`
	Examine n instructions from current pc
	Example: `x/10i $pc`
	Examine 16 words down from top of stack

`set disassemble-next-line on|off`	If on, gdb will display disassembly of next source line when program halts (Ctrl-c)

`p/x <var>=<value>`
 	
`set var <varname>=<value>`
	Modifying memory
	`p/x count=0x1000FFFF` set the value of variable 'count'
	`set var count=59`

`set <reg>=<value>`	Modifying a register `set $pc=0xbfc00000` **Description:** set the pc to the reset vector; required after doing a `mon reset halt`, to synchronize OpenOCD and gdb
`monitor mdw addr <#words>`	Similar OpenOCD commands `mon mdw 0x80000100 16` **Description:** Display 16 words of memory starting at memory address 0x80000100. The default number of words is 1. Processor must be halted (Ctrl-c).

`monitor mww addr word [<#words>]`	Memory write word
	`mon mww 0x80000100 0xaaaaaaaa`
	Description: Write value 0xaaaaaaaa to memory address 0x80000100.
	Memory **fill**
	`mon mww 0x80010000 0xffffffff 0x100`
	Description: fill 0x100 words of memory with all ones
	Processor must be halted (Ctrl-c).

`<return>`	**Description:** Pressing the return/enter key in gdb with no command typed at the prompt will repeat the last command.

Command	Example / Description
`hbreak <address>`	**Hardware instruction breakpoint** `hb 23` Set hardware breakpoint on line 23 of current module. **Description:** Useful when debugging firmware or code that is installed/copied into memory after boot up. Use `i b` command to view breakpoints and watchpoints. To tell how many breakpoints are available, use the `mon mips32 cpuinfo` command. Example output is: Max Number of Instr Breakpoints: 8

`watch <address>`
`rwatch <address>`
`awatch <address>`
	Watchpoints which are hardware–based; program runs at full speed.
	Cause a breakpoint when the load or store address matches the address of a variable, accessed with the type selected.
	`watch *0xbf800000` break when this memory-mapped register is written to.
	Description: Useful to halt execution when a memory location is written (watch), read (rwatch), or accessed (read or written - awatch).
	Use `i b` command to view breakpoints and watchpoints
	Important Note: to continue execution, you must disable the watchpoint, step over the lw or sw instruction, then re-enable the watchpoint because the MIPS processor halts before the load or store instruction completes. For example:
	
	watch *0xbf800000 #set watch on write to mem-mapped register
	<assume assigned to watchpoint 1>
	c
	<processor halts>
	dis 1 #disable watchpoint 1
	si #single step over lw or sw instruction
	en 1 #enable watchpoint 1
	c #processor continues execution
	
	To tell how many watchpoints are available, use the
	`mon mips32 cpuinfo` command. Example output is:
	Max Number of Data Breakpoints: 4

5.3. MIPS-specific OpenOCD commands run from gdb 'mon' prefix

All the OpenOCD commands specific to MIPS can be issued in the GDB console by prefixing the command with 'mon' which stands for monitor. Refer to section 5.2.3.for the list of commands that start with `mips32`. Example: `mon mips32 cp0` displays the values of all the coprocessor 0 registers.

6. Connecting OpenOCD to the Probe-On-Board (POB) of the WiFIRE rev D for Windows

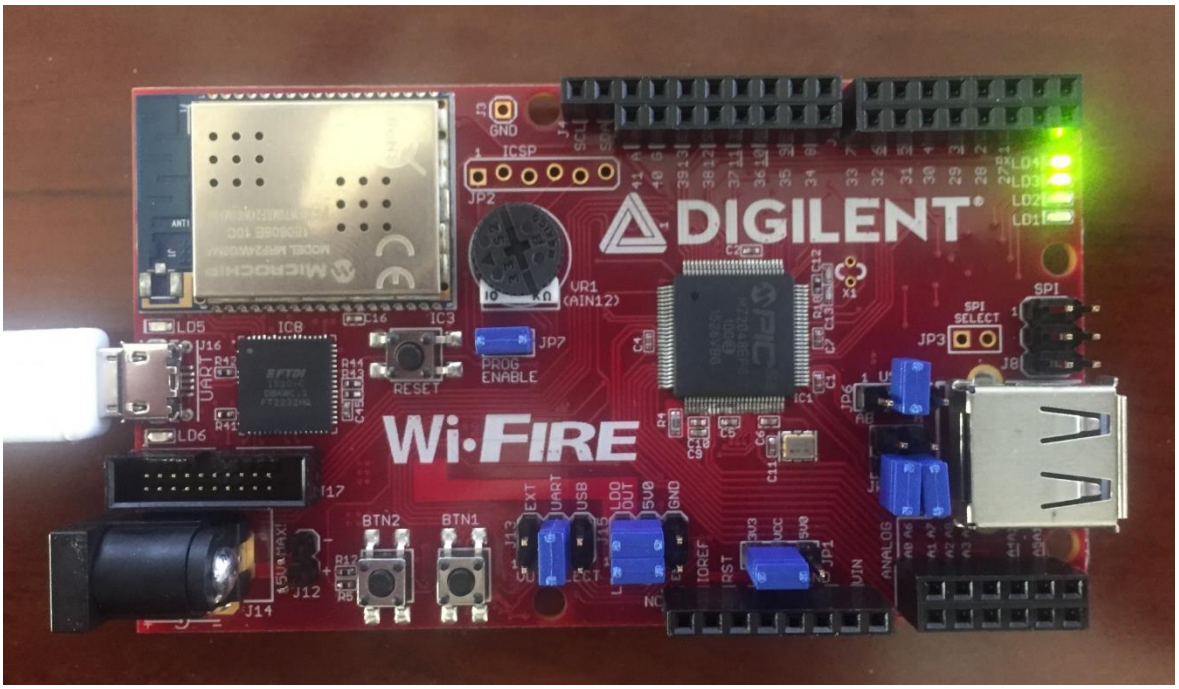


Figure 1 - Digilent Wi-FIRE Rev D board with 2 x 10 x .05" EJTAG/Trace Connector

OpenOCD uses an installer program Zadig to install WinUSB, the probe's Windows USB driver. The screenshots in this section were taken from running the program "zadig_2.2.exe" from the web site http://zadig.akeo.ie/ .

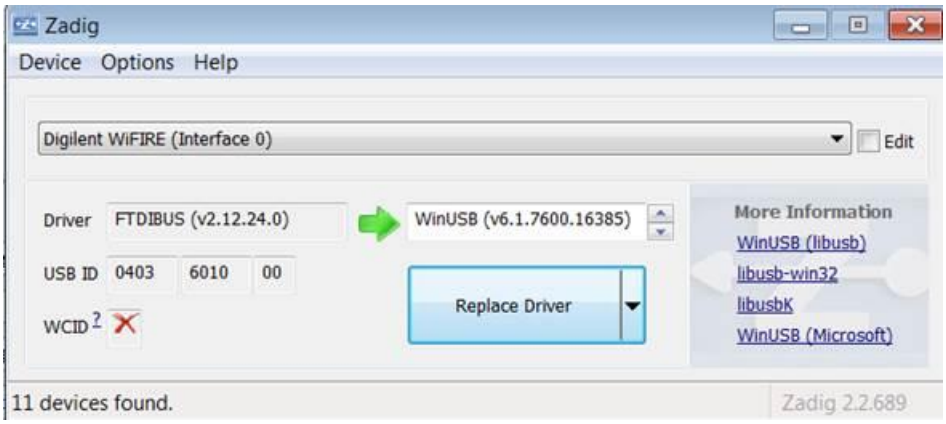
Older versions of the zadig program can be downloaded as a 7z compressed file from:

https://codescape.mips.com/components/probes/openocd/tools/zadig/

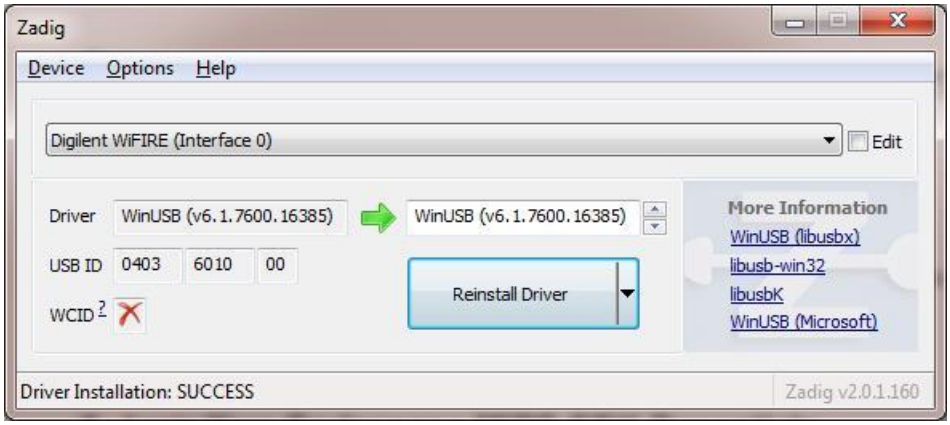
6.1. Installing WinUSB Windows Driver for the POB

To install WinUSB follow the steps below:

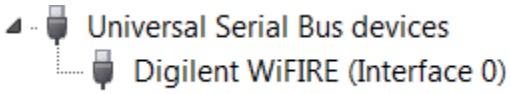
1. Connect a USB cable from your Windows PC to the micro USB connector on the left side of the board. This will power the board. Windows will attempt to locate driver software, cancel this.
2. Open the Zadic.exe installer program.
3. In the top field of the Zadig installer, if the field is blank and there are no selections when clicking on the down-arrow, click on Options > List All Devices.
4. Select 'Digilent WiFIRE (Interface 0)' then click on 'Replace Driver'. It should look similar to the following:



After the install the Zadig installer will look something like the following, with the bottom line of “Driver Installation: SUCCESS”:



5. The Device Manager should show the USB device as



6. Close the installer.

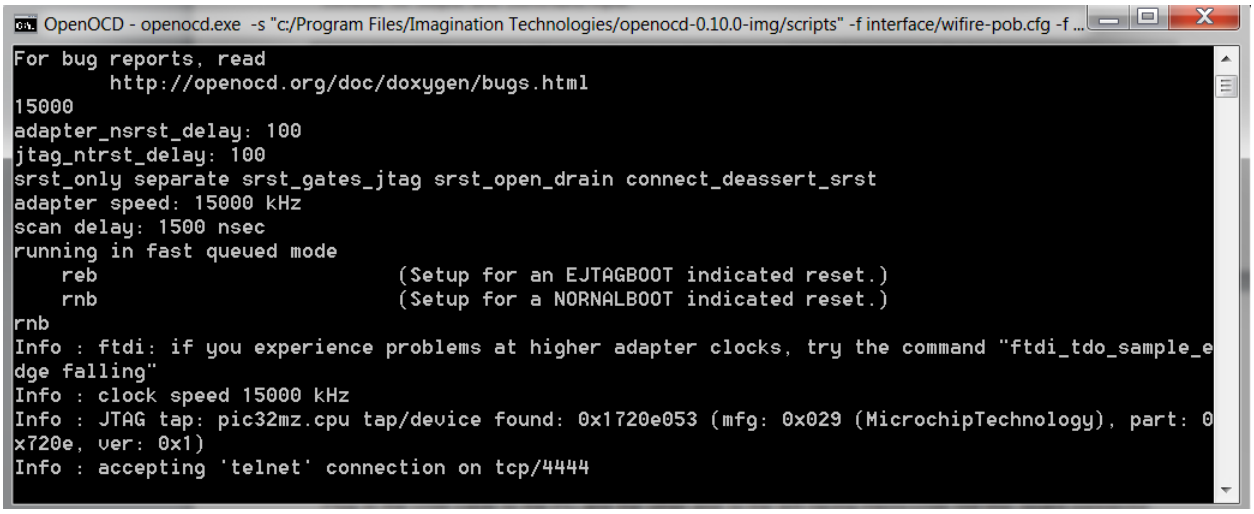
6.2. Running OpenOCD Debug with the WiFIRE board and POB

The OpenOCD connection uses two configuration files – WiFIRE-POB.cfg located in the **interface** directory and wifire.cfg located in the **target** directory. The following two commands can be put in a batch file and executed to 1) start up openocd and initialize the POB connection and 2) start a telnet window which allows openocd commands to be issued to the target, such as ‘halt’.

```
start "OpenOCD" cmd.exe /K openocd.exe -s "<install dir>/openocd-
0.10.0-img/scripts" -f interface/wifire-pob.cfg -f target/wifire.cfg -c "init"
start "telnet" cmd.exe /K telnet localhost 4444
```

Note: the –s and path sets up the path for OpenOCD to search for configuration files. This may not be required in your setup.

Plug in the USB cable to the PC and the other end to the left-facing micro-USB WiFIRE board connector. Run the above batch file. The initial output should resemble the following:



Following are examples of several OpenOCD commands that can be issued from the telnet console, and the resulting output. The target is a PIC32MZ-EF which includes a MIPS M5150 processor core.

Open On-Chip Debugger

```
> halt
target halted in MIPS32 mode due to debug-request, pc: 0x9d001c80
> mips32 cpuinfo
vzase: 1
cpuCore: MIPS_M5150
cputype: 4194480
 vendor: MIPS
  cpuid: 0
instr Set: MIPS32 (at reset) and microMIPS
prid: 1a720
rtl: 20.0.0
Instr Cache: 16384
 Data Cache: 4096
Max Number of Instr Breakpoints: 8
Max Number of  Data Breakpoints: 4
mta: false
MMU Type: TLB
TLB Entries: 16
dsp: true
Smart Mips ASE: false
msa: false
mvh: false
cdmm: true
> step
target halted in MIPS32 mode due to single-step, pc: 0x9d001c88
> reg
```

```
===== mips32 registers
(0) r0 (/32): 0x00000000
(1) r1 (/32): 0xFFBFFFFF
(2) r2 (/32): 0x00000000
(3) r3 (/32): 0x000000BF
(4) r4 (/32): 0x000000FA
(5) r5 (/32): 0x00000000
(6) r6 (/32): 0x00000000
(7) r7 (/32): 0x00000000
(8) r8 (/32): 0x9D003764
(9) r9 (/32): 0x00000000
(10) r10 (/32): 0x01000000
(11) r11 (/32): 0x00000001
(12) r12 (/32): 0x00000000
(13) r13 (/32): 0x00001000
(14) r14 (/32): 0x00000010
(15) r15 (/32): 0x00000004
(16) r16 (/32): 0xBF820000
(17) r17 (/32): 0x0BEBC200
(18) r18 (/32): 0x00000040
(19) r19 (/32): 0xBF860000
(20) r20 (/32): 0xBF860000
(21) r21 (/32): 0xBF820000
(22) r22 (/32): 0x80000000
(23) r23 (/32): 0xA0001000
(24) r24 (/32): 0x042075A3
(25) r25 (/32): 0x027909E9
(26) r26 (/32): 0x00000000
(27) r27 (/32): 0xB5DC76CA
(28) r28 (/32): 0x80008380
(29) r29 (/32): 0x8007FF60
(30) r30 (/32): 0x8007FF60
(31) r31 (/32): 0x9D003494
(32) status (/32): 0x25000001
(33) lo (/32): 0x05F5E100
(34) hi (/32): 0x00000000
(35) badvaddr (/32): 0xDF894225
(36) cause (/32): 0x00801C00
(37) pc (/32): 0x9D001C88
<floating point registers removed>
> resume
> halt
target halted in MIPS32 mode due to debug-request, pc: 0x9d001c80
> version
Open On-Chip Debugger 0.10.0-IMG-00055-g3aee415 (2017-01-27-15:19)
> mips32 cp0
     userlocal: 0xd5c35ffb
        hwrena: 0x00000000
      badvaddr: 0xdf894225
         count: 0x85ec9f2d
       compare: 0x85ecea60
        status: 0x25000001
        intctl: 0x00000020
        srsctl: 0x1c000000
      view_ipl: 0x00000000
         cause: 0x00801c00
     nestedexc: 0x00000000
```

```
            epc: 0x9d001c80
     nestedepc: 0x9d001c80
           prid: 0x0001a720
          ebase: 0x9d000000
       cdmmbase: 0x00000002
         config: 0x80240483
        config1: 0x9e9b0d9b
        config2: 0x80000000
        config3: 0x8ca2bd68
        config4: 0xa00c0000
        config5: 0x00000001
        config7: 0x80000000
         lladdr: 0x009b7542
          debug: 0x40128020
  tracecontrol: 0x00000000
 tracecontrol2: 0x00000000
usertracedata1: 0x00000000
      tracebpc: 0x00000000
          depc: 0x9d001c80
usertracedata2: 0x00000000
      perfctl0: 0x80000000
      perfcnt0: 0x5f5febfa
      perfctl1: 0x00000000
      perfcnt1: 0xeffee630
        errctl: 0x00000000
      errorepc: 0x9fc004c4
        desave: 0x00000004
```

7. Connecting a Bus Blaster probe to a chipKIT Wi-FIRE rev C development board

To use Bus Blaster probe with the older Digilent chipKIT Wi-FIRE rev C board use the Digilent/EJTAG Adapter Board as shown below. The adapter connection details are given at the end of this section in Table 1.

1. Connect the adapter board to the 6-pin JTAG connector JP3 on the Wi-FIRE board as shown in Figure 2.

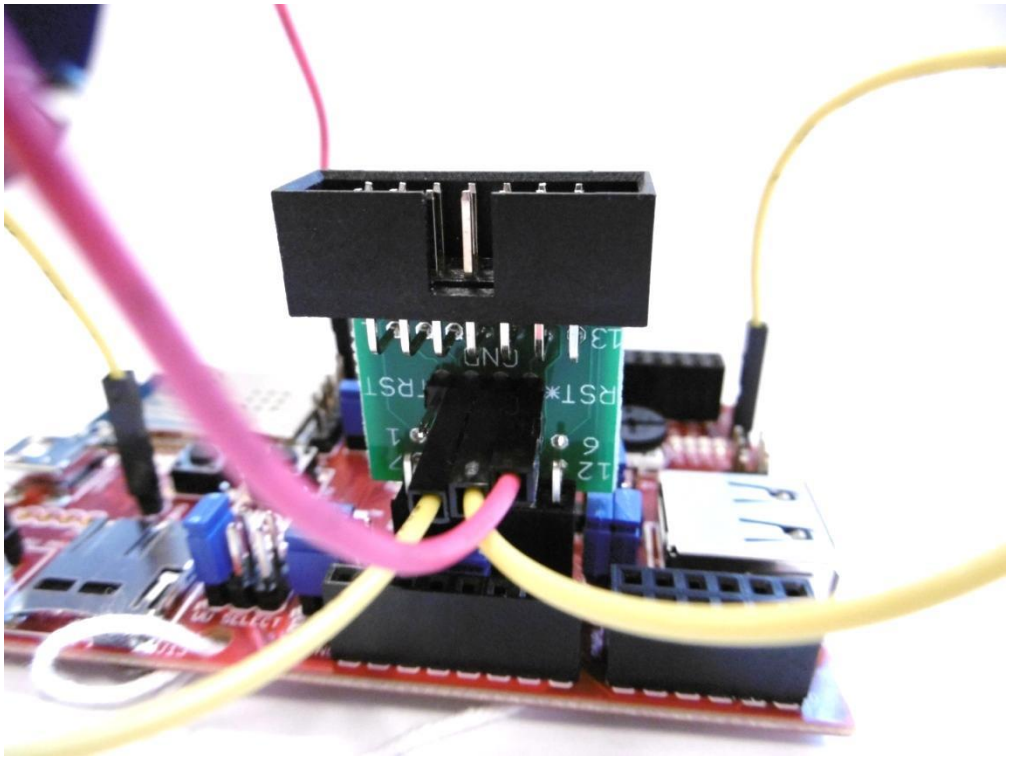


Figure 2 - EJTAG Adapter Board

2. Make sure that pins 1 to 6 on the adapter board connect to the JTAG connector on the Wi-FIRE board.
3. Using the flying leads supplied make the following connections as shown in Figure 3:
   - Pins 2 and 3 on the adapter board pin header to GND J2 and J3 on the Wi-FIRE board.
   - Pin 4 on the adapter board pin header to pin 1 of JP1 on the Wi-FIRE board (pink lead shown in figure).

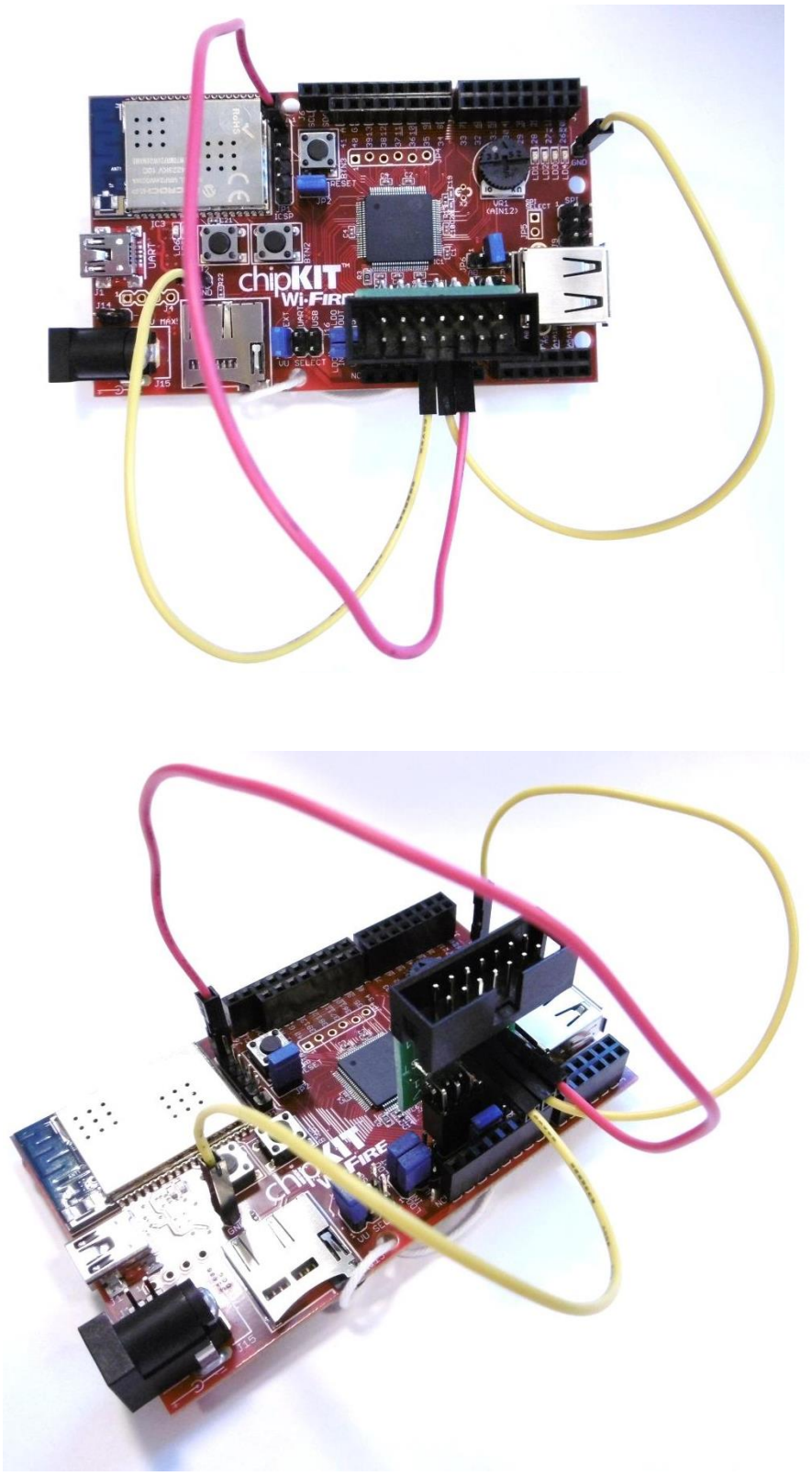


Figure 3 - Flying lead connections on the Wi-FIRE board

4. Connect the Bus Blaster to the 14 way IDC connector on the adapter board using the ribbon cable supplied.

Signal Name	Bus Blaster v3c EJTAG	Wi-FIRE JP3 JTAG	Wi-FIRE JP1
TRST*	1	NC	
TDI	3	2	
TDO	5	3	
TMS	7	1	
TCK	9	4	
RST*	11		1 (MCLR)
DINT	13	NC	
GND	2	5	

Table 1 - Bus Blaster to Wi-FIRE adapter connection details

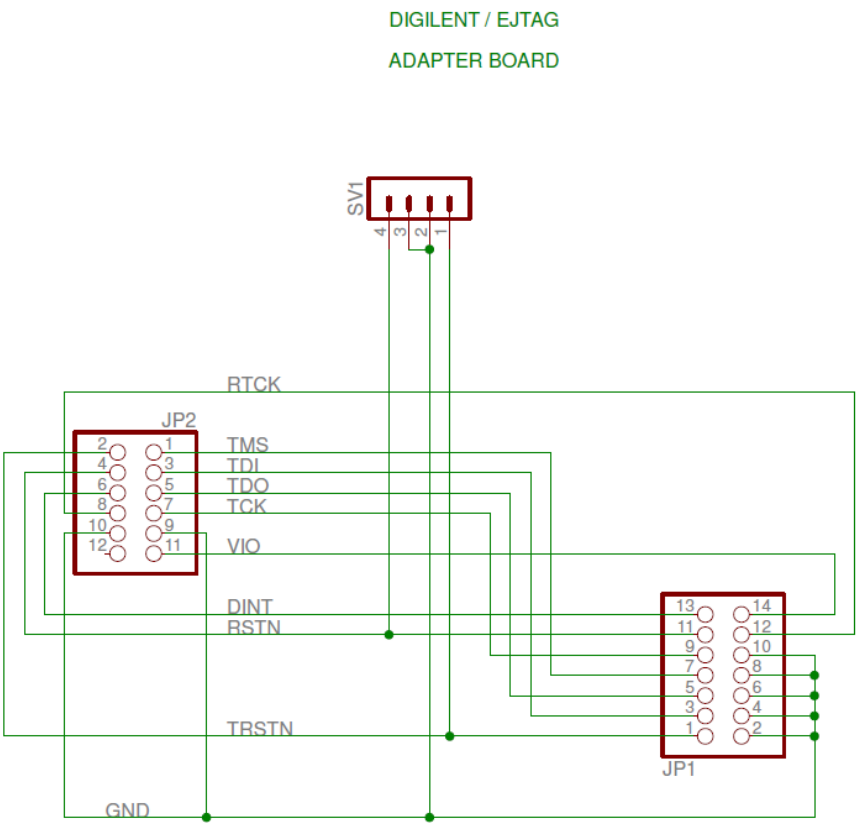


Figure 4 - Bus Blaster to Wi-FIRE adapter schematic