Marvell
Wireless Microcontroller
Development Host Setup
Guide

Not Approved by Document Control.

August 24, 2015
CONFIDENTIAL
Document Classification: Proprietary Information

Document Conventions

	**Note:** Provides related information or information of special importance.
	**Caution:** Indicates potential damage to hardware or software, or loss of data.
	**Warning:** Indicates a risk of personal injury.

For more information, visit our website at: www.marvell.com

Date	Revision	Changelog
Aug 24 2015	V2.0	• Include Eclipse setup details • Libusb-win32 is obsolete, include instructions for winusb • Add support for fixing Mac driver conflict
Apr 7 2015	V1.0	Create independent document

Table of Contents

1 Overview

This document mentions the configurations and settings that are required to use the Marvell's Wireless Microcontroller family of SDKs on your development host. Most parts of this document are common across all SDK-variants. If certain parts are applicable to only a particular SDK variant, a note mentioning this behavior has been included.

The first step for development of your application for the SDK board is to cross compile the application along with SDK on a host computer. Once compiled the generated binary file is loaded onto the board using tools provided along with SDK. Once the application starts running on the board you can debug or interact with it from the Serial console on your host computer.

The following host platforms are supported for development:

- Windows 7/8
- Fedora 20, 21
- Ubuntu 12.04, 14.04
- Mac OS-X Yosemite

You may be able to use other platforms, but those platforms are not supported officially. This system will act as the host platform for development and debugging. You need to have permissions to install software on this host system. Following are the external tools required for building SDK successfully.

- Any of the above platforms.
- A recent version of the ARM toolchain to cross compile your application and the SDK. The SDK takes advantage of the latest versions of the toolchain to optimize the image footprint and fit more functionality into less space. Using older toolchains is not recommended.
- Python. This is required for building FTFS images.
- Host GCC. This is required for compiling host tools.

See Section 2 for instructions on how to set up your Windows system.

See Section 3 for instructions on how to set up your Linux system.

See Section 4 for instructions on how to set up your Mac system.

The development kit is pre-flashed with Wireless Microcontroller Demo Project Firmware and an associated flash file system to allow you to exercise many of the features of the SDK before you set up a full development toolchain. However, you will still need to set up a host system so you can get access to the serial console.

1.1 Development Toolchain Requirements

For development purposes, at a minimum you will need an ARM compiler toolchain (in addition to the tools bundled with the SDK). The SDK bundles a number of tools to facilitate various development tasks. In particular, OpenOCD software is supported and bundled with the SDK to access various JTAG functions such as programming the flash and downloading and running firmware images in memory.

1.1.1 GNU Toolchain

The SDK officially supports the GCC Compiler toolchain. The cross compiler toolchain for GNU ARM is available from the following URL: https://launchpad.net/gcc-arm-embedded

The build system is configured to use the GNU toolchain by default. The Makefiles assume that the GNU compiler toolchain binaries are available on the user's PATH and can be invoked from the Makefiles. The Makefiles also assume that the GNU toolchain binaries are prefixed with `arm-none-eabi-`.

The GCC toolchain can be used with GDB for debugging with OpenOCD (bundled with the SDK) providing the software interfacing to JTAG.

2 Working with Windows Development Host

There are two ways in which Windows can be used as a development host. Either MinGW or Cygwin can be used. Experimental MinGW support has recently been added to simplify the initial setup procedure.

2.1 Setup Cygwin

Cygwin is required for building the SDK. If you do not have Cygwin already installed, please refer to Section 6.2 for the instructions. The Cygwin installation should have at least the following packages:

- **make:** make is required to build the SDK.
- **binutils:** the binutils package is used by the SDK components to interpret generated artifacts.
- **gcc:** gcc is used by the build system to build binaries that are executed on the development host.
- **python (v2.7)**: Python language interpreter is required to create FTFS file-system image
- **openssl-devel**: OpenSSL developer libraries are required by certain development tools for crypto routines.

Open the Cygwin terminal and add the following line at the end of the */etc/fstab* file.

```
none /cygdrive cygdrive binary,noacl,posix=0,user 0 0
```

If a similar line already exists in your fstab file, please ensure that the word *noacl* is part of that line as shown above.

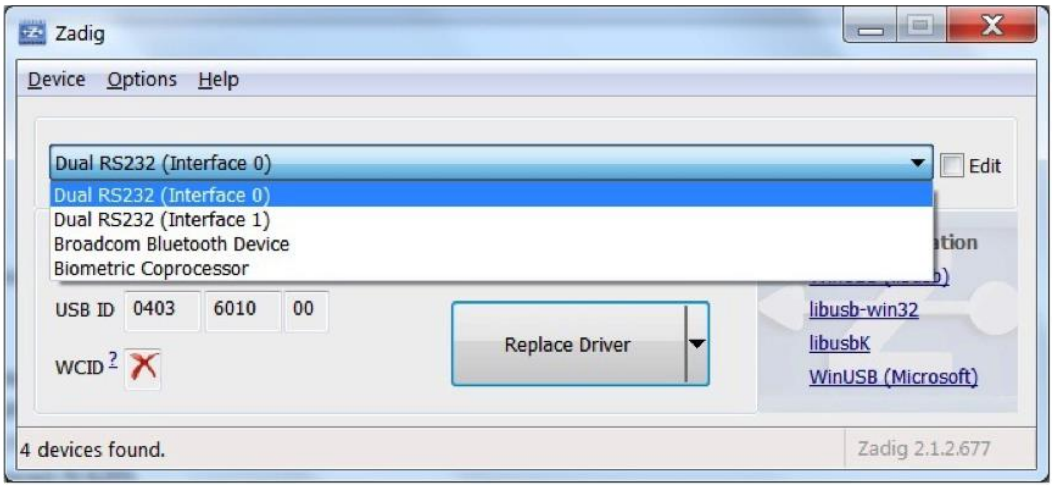
You could edit this file from Windows using applications like notepad. The file is present at the location *c:\cygwin\etc\fstab*, assuming that you installed Cygwin at the *c:\cygwin* location.

This ensures that the original permissions of the files are retained when the tarball is untarred in Cygwin.

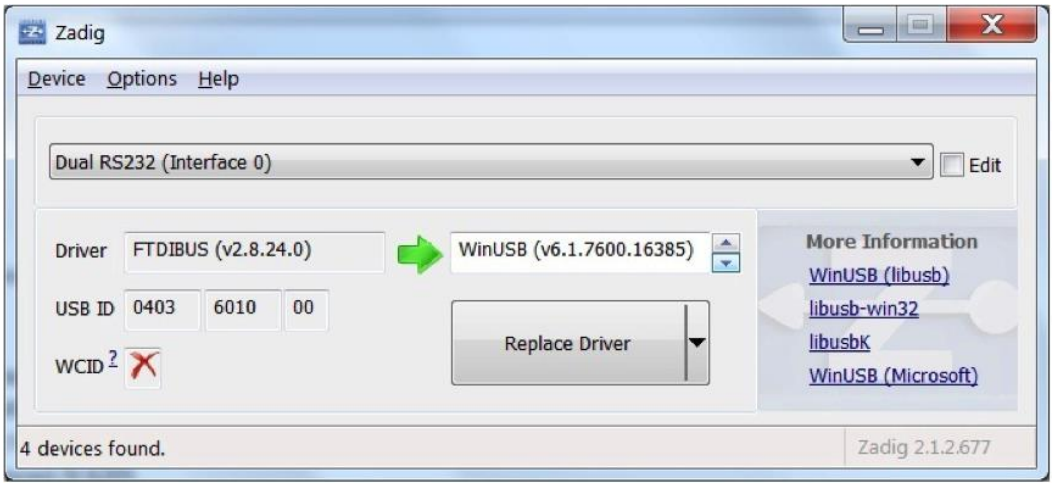
2.2 Install WinUSB Driver

1. Download the Zadig driver management application from: http://zadig.akeo.ie . This application eases the driver management and installation for WinUSB.
2. Please connect the development board to the development host by plugging in the USB cable. Within a minute or two, Windows will detect and install the default driver for this device. Once this is done, a pop-up that says this device is now ready to use is displayed. Please proceed after this pop-up is seen.
3. Execute the downloaded Zadig executable. Select Options → List All Devices

4. In the device drop-down, please select the *Dual RS232 (Interface 0)* option as shown in the image below. If this interface is not visible, please disconnect and reconnect the development board to your development host.
    a. Note that if you use the J-Link connector, this will be seen as J-Link
    b. Note that if you use the MA-Link connector, this will be seen as *Dual RS232-HS*



5. In the list of drivers to be installed please select WinUSB as show below:



6. Click on *Replace Driver*, to replace the default driver that is installed.

2.3 Preparing for Eclipse

In order to use Eclipse IDE for development and debugging, please make sure that you have **Cygwin bin path** and **Toolchain path** added to the system path. This can be done by clicking on "Control Panel" -> "System and Security" -> "System" -> "Advanced system settings" -> "Environment Variables" -> "User Variables" -> "PATH".

- The typical Cygwin path is c:\cygwin\bin or c:\cygwin64\bin; but it may change depending on where you install Cygwin.
- Similarly typical toolchain path is c:\Program Files\Gnu Tools ARM Embedded\<version> but may change depending on where you install toolchain.

Once these changes are done, you can open Windows command shell and check you can execute "bash" and "arm-none-eabi-gcc" commands.

2.4 Open Serial Console

Microsoft no longer includes the Hyperterminal application for Windows 7. Freely available Hyperterminal applications are available for Windows 7 users on the Internet.

Once all the above components have been installed, follow these steps to get the serial console:

1. On your Windows host, right click on "My Computer" and go to Properties.
1. Go to Hardware -> Device Manager
2. Under "Ports (COM & LPT)", you will find two entries "USB Serial Port (COMx)", where x will be a number. These are the COM ports that are assigned to your device. If you don't find any such port, check if you have rightly installed the drivers mentioned in the previous sections.
3. Launch the Hyperterminal application. For Windows XP, this is available at Start-> All Programs-> Accessories-> Communications-> HyperTerminal. Alternatively, you may use other serial programs such as putty.
4. Go to File -> Properties -> Settings → ASCII Setup. Ensure that only the option "*Wrap lines that exceed terminal width*" is checked.All the other options should be unchecked.
5. Enter a name for this Connection, and press OK.
6. Under *Connect Using* select *COMx* which is the second COM Port out of the two that you saw in point 4 above.
7. Set the following parameters:
   - Bits per second: 115200
   - Data bits: 8
   - Parity: None
   - Stop bits: 1
   - Flow control: None
8. The Connection is now established.
9. Hit enter on the Hyper Terminal window. This should show you a hash (#) on the screen. If that is not the case, repeat step 7 onwards but select the second COM port instead of the first one.

Note

The openocd binary bundled with the SDK doesn't work cleanly with the usb-serial console. You may experience that the console becomes inaccessible after using the openocd to perform any operations on the board. After flashing a new image, unplugging and plugging the USB cable into your development host will get the console back.

Because of licensing issues, we cannot provide a binary which does work. You will

need to build openocd yourself with the proprietary FTD2xx drivers to make the usb-serial console work. Refer to the README (Section “Building OpenOCD with FTD2xx drivers”) in the OpenOCD directory in the SDK for instructions on how to do it.

2.5 Setup MinGW

Please skip this section if you will be using Cygwin. Note that support for MinGW is experimental in nature. Also, Eclipse based development and debugging is presently not supported with MinGW.

MinGW toolset can be downloaded separately from the same place where you downloaded the SDK release. It is in the form of a zip file. Once unzipped it needs no other changes and can be used as it is. It has a collection of all the tools (except the cross compiler toolchain and python) required for the SDK build. Follow the below steps for using MinGW:

1. Download and unzip MinGW.zip to C:\MinGW
2. Add the following variables to the PATH environment variable.
   2.1. C:\MinGW\bin
   2.2. C:\MinGW\msys\1.0\bin

   ```
   C:> PATH C:\MinGW\bin\;C:\MinGW\msys\1.0\bin\;%PATH%
   ```
3. Download and install python from https://www.python.org/download/. Any other Python installation may also work.

Now you are all set to build the SDK. Please jump to Section 1 for toolchain installation.

3 Working with a Linux Development Host

Linux development hosts can be used in lieu of Windows development hosts. Any modern Linux Desktop distribution such as Ubuntu or Fedora is supported. You may find it beneficial to upgrade to the most recent release.

3.1 Installing Packages

To enable quick setup of development environment on a newly setup Linux machine, a script is provided along with the SDK. The script will try to autodetect the machine type and install the appropriate software viz. C libraries, USB library, FTDI library, ncurses, python and latex.

Please go to the wmsdk_bundle-x.y.z/ directory and run the following command:

```
# ./wmsdk/build/utils/installpkgs.sh
```

If you are using the HAP SDK variant, please use the following command instead:

```
# ./hap_sdk/build/utils/installpkgs.sh
```

3.1.1 Avoiding ‘sudo’

Your Linux development host can also be configured to perform ‘flashprog’ and ‘ramload’ operations without requiring the ‘sudo’ command to be executed each time. This can be done by executing the following command:

```
# ./wmsdk/build/utils/installpkgs.sh perm_fix
```

Note that fixing these permissions is mandatory for ensuring a smooth Eclipse IDE based experience.

3.2 Setup Serial Console

1. Insert the USB cable into the Linux host’s USB slot as mentioned above. This will trigger the detection of the device and you should see messages like the following in the */var/log/messages* file (or after executing the *dmesg* command).

```
Jan  6 20:00:51 localhost kernel: usb 4-2: new full speed USB device
using uhci_hcd and address 127
Jan  6 20:00:51 localhost kernel: usb 4-2: configuration #1 chosen
from 1 choice
Jan  6 20:00:51 localhost kernel: ftdi_sio 4-2:1.0: FTDI USB Serial
Device converter detected
Jan  6 20:00:51 localhost kernel: ftdi_sio: Detected FT2232C
Jan  6 20:00:51 localhost kernel: usb 4-2: FTDI USB Serial Device
converter now attached to ttyUSB0
Jan  6 20:00:51 localhost kernel: ftdi_sio 4-2:1.1: FTDI USB Serial
Device converter detected
Jan  6 20:00:51 localhost kernel: ftdi_sio: Detected FT2232C
Jan  6 20:00:51 localhost kernel: usb 4-2: FTDI USB Serial Device
converter now attached to ttyUSB1
```

2. As can be seen two ttyUSB devices have been created. The second of this device is the serial console, in our case ttyUSB1
3. Execute minicom in setup mode (*minicom –s*). Alternatively, you can use other serial programs such as putty.
4. Go to Serial Port Setup
5. Perform the following settings:

```
| A -    Serial Device      : /dev/ttyUSB1
| B - Lockfile Location     : /var/lock
| C -   Callin Program      :
| D -  Callout Program      :
| E -    Bps/Par/Bits       : 115200 8N1
| F - Hardware Flow Control : No
| G - Software Flow Control : No
```

You can save these settings in minicom for future use. The minicom window will now show messages from the serial console.

6. Hit enter on the serial console window. This should show you a hash (#) on the screen.

Note	The development boards from Marvell have an FTDI chip. The FTDI chip exposes two USB interfaces for the host. The first interface is associated to JTAG functionality of the MCU and the second interface is associated with physical UARTx port of the MCU.

4 Working with Mac OS-X Development Host

Mac OS-X Mavericks or later can be used as a development host. The Xcode application should be installed on OS X. If you do not have Xcode installed, please refer to Section 6.1 for the instructions. Please use the following steps for the setup.

4.1 Install OpenOCD

1. Install macports from http://www.macports.org/install.php. Restart your terminal window so that it finds the port utility in its execution path. Then

   ```
   $ which port
   ```

   ```
   $ sudo port selfupdate
   ```

   NOTE: Some networks do not allow rsync traffic through their gateways. If your selfupdate command times out, it might be better to run this command by connecting to another network that doesn't block such traffic.

2. Install OpenOCD as:

   ```
   $ sudo port install openocd
   ```

   Note: Please make sure that the version of installed openocd is 0.9.0. You can instruct the port to install 0.9.0 as:

   ```
   $ sudo port install openocd @0.9.0
   ```

4.2 Manage driver conflict

The Yosemite release includes all the drivers required for accessing the development board.

If you have already installed FTDI VCP drivers, please remove them. The existence of these drivers can be verified by checking if */System/Library/Extensions/FTDIUSBSerialDriver.kext/Contents/Info.plist* file exists. If it does, these drivers are installed. You can uninstall them as follows:

- Move this directory to some backup folder.

```
$ sudo mv  /System/Library/Extensions/FTDIUSBSerialDriver.kext/Contents/
~/backup
```

- Reboot the Mac

The development board exposes both a serial console as well as JTAG over the USB interface connected to the development host. It has been observed that the driver's in the OS-X Yosemite end-up acquiring both the ports thus not allowing JTAG access for our use. Please perform the following steps to use both serial console and JTAG from your development host.

1. Disconnect the development board from your Mac.
2. Unload Apple's USB FTDI driver as:

   ```
   $ sudo kextunload -p -b com.apple.driver.AppleUSBFTDI
   ```

3. Now load only the Serial Terminal personality

```
$ sudo kextutil -b com.apple.driver.AppleUSBFTDI -p AppleUSBEFTDI-
6010-1
```

4. This ensures that the driver can only then claim the available interface that is used for serial console.
5. You can verify that only a single serial port is enumerated by running *'ls –l /dev/tty.usb*'* command. This should show only a single file.

Now you are set to use both serial console and JTAG, as long as the Mac is not rebooted, or the board is not physically disconnected.

4.3 Setup Serial Console

1. Once the above settings are done, and the board is plugged in to your development host, a virtual USB device will be created for you. Please lookup the name of this device. The device will be of the form /dev/tty.usbserial<id-string>.
2. Install any application that can read serial console (e.g. minicom) as:

   $ sudo port install minicom
3. Execute minicom in setup mode (*minicom* –s). Alternatively, you can use other serial programs such as putty.
4. Go to Serial Port Setup
5. Perform the following settings:

```
| A -    Serial Device      : /dev/tty.usb-<id-string>
| B - Lockfile Location     : /var/lock
| C -   Callin Program      :
| D -  Callout Program      :
| E -    Bps/Par/Bits       : 115200 8N1
| F - Hardware Flow Control : No
| G - Software Flow Control : No
```

You can save these settings in minicom for future use. The minicom window will now show messages from the serial console.

6. Hit enter on the serial console window. This should show you a hash (#) on the screen. Try hitting the 'reset' button of the board, bootup logs should appear on the screen.

5 Eclipse Setup

Note

If you are using Windows host, please make sure that you have performed steps mentioned in “Preparing for Eclipse” section. Also, note that Eclipse is not supported through MinGW, and please use Cygwin.

If you are using Linux host, please make sure that you have performed steps mentioned in “Avoiding sudo” section.

Eclipse is a preferred IDE for WMSDK based application development and debugging. It provides a rich, user-friendly IDE with integrated debugging support including thread aware debugging. This section describes common Eclipse setup for all the development hosts supported.

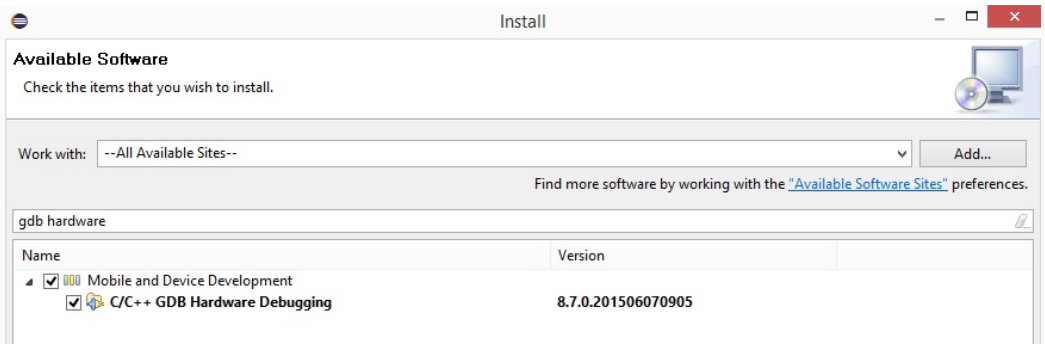
5.1 Download and Install

5.1.1 Eclipse

Download and install “Eclipse IDE for C/C++ Developers” from http://www.eclipse.org. The supported Eclipse version is Eclipse Mars. Installation is just extracting the downloaded archive. Platform specific Eclipse executable can be executed to start the eclipse

5.1.2 C/C++ GDB Hardware Debugging Plugin

Start Eclipse and go to Help -> Install new software. Select “All Available Sites” in “Work with” menu. Then enter filter text as “GDB Hardware”. Select “C/C++ GDB Hardware Debugging” option and install the plugin



Note

If you are using Mac OS-X, please start Eclipse from the terminal command line and not through Finder. This is because openocd is not present in system default paths. You can start eclipse as:

$ /path/to/Eclipse.app/Contents/MacOS/eclipse

6 Appendix: Dependent Installations

6.1 Installing Xcode on Mac OS X

1. Install Xcode from the app store: https://itunes.apple.com/in/app/xcode/id497799835?mt=12
2. Click on the *Install* button by going to:

   Xcode menu > Preferences > Downloads > Command line tools > Install
3. Verify that the development utilities *make* and *gcc* are available through the console

   $ gcc –version

   $ which make

6.2 Installing Cygwin on Windows

If you don't already have Cygwin installed on your system, the first thing you need to do is to download and install Cygwin.

1. Install Cygwin from: http://www.cygwin.com/setup_x86.exe (for x86 32-bit systems) or http://www.cygwin.com/setup_x86_64.exe (for x86 64-bit systems)

- Select the option *Install from Internet*
- Use default installation path: *c:\cygwin*. If you chose an alternate installation directory, please make sure that there are no spaces in the path.
- Pick the Local Package Directory (this is the download cache directory)
- Select the option *Direct Connection*
- Select any mirror you want to use
- Add additional packages to the default selection:

  You will need to add the following additional packages to work with the SDK.
  - **make:** make is used by the build system of the SDK
  - **binutils:** binutils is used by flash programmer to determine the application entry offset
  - **gcc:** gcc is used by the build system to build native packages
  - **python:** python scripts are used by some of the optional utilities
  - **openssl-devel**: openssl's headers and libraries are used for generation of the manufacturing image. The pairing pin is stored in the manufacturing partition in a secure manner.

  The following packages are required if you wish to rebuild the reference manual from the SDK. It may take a considerably longer time for installing these packages. If you do not wish to rebuild the documentation, these packages can be safely skipped.
  - **doxygen (optional):** doxygen is used to build the developer manual in the SDK.
  - **tetex, texlive-collection-latexextra, texlive-collection-latexrecommend (optional):** These tools are used to build a pdf version of the developer manual

Click "Next". The Cygwin Setup window will show the progress as each package gets installed.

Note

If you are not familiar with cygwin, please visit http://cygwin.com/ for additional information and details. In particular, the Cygwin User Guide (http://cygwin.com/cygwin-ug-net/) is a good resource for new users.

Open the Cygwin terminal and add the following line at the end of the */etc/fstab* file.

```
none /cygdrive cygdrive binary,noacl,posix=0,user 0 0
```

If a similar line already exists in your fstab file, please ensure that the word *noacl* is part of that line as shown above.

You could edit this file from Windows using applications like notepad. The file is present at the location *c:\cygwin\etc\fstab*, assuming that you installed Cygwin at the *c:\cygwin* location.

This ensures that the original permissions of the files are retained when the tarball is untarred in Cygwin.

Marvell Semiconductor, Inc.
5488 Marvell Lane
Santa Clara, CA 95054, USA

Tel: 1.408.222.2500
Fax: 1.408.752.9028

www.marvell.com

Marvell.Moving Forward Faster