



IMPORTANT NOTICE

Dear Customer,

Effective on 6 December 2019, NXP Semiconductors ("NXP") has acquired the wireless connectivity assets of Marvell Technology Group, Ltd.

Where a Marvell reference remains in the attached document, read "NXP". Additionally, use the NXP link as shown below.

- Instead of <http://www.marvell.com>, use <http://www.nxp.com>.

Any copyright notice in the document ("© (year) Marvell. All rights reserved.") should be read to be "2019 NXP B.V. All rights reserved."

Any references to Marvell part numbers and ordering information within the document have changed. Please refer to the customer communication dated 17 September 2019, which includes the Product Change Notification for the relevant devices. If you have any questions related to the document, please contact our nearest sales office or consult www.nxp.com. Thank you for your cooperation and understanding.

NXP Semiconductors

UM11386

Getting Started with 88MW320-88MW322 for Amazon AWS IOT SDK

Rev. 1 — 18 May 2020

User manual

Document information

Information	Content
Keywords	Wireless Microcontroller, GNU toolchain, Linux development host, Eclipse setup, Amazon FreeRTOS
Abstract	Guidelines to install some software, and to build and run a demo project



Revision History

Revision history

Rev	Date	Description
Rev.1	18-May-2020	Initial version

1 Introduction

The AWS IoT Starter Kit is a development kit based on the 88MW320/88MW322, the latest integrated Cortex M4 Microcontroller from NXP, which integrates 802.11b/g/n Wi-Fi on a single microcontroller chip. The development kit is FCC certified and available for sale. The 88MW320/88MW322 modules are also FCC certified and available for customization and volume sale.

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

Ubuntu 16.04 is the host platform supported for development and debugging. You may be able to use other platforms, but those are not officially supported. You need to have permissions to install software on the host platform.

The external tools required to build the SDK are the following:

- Ubuntu 16.04 host platform
- The ARM toolchain version 4_9_2015q3
- Eclipse 4.9.0 IDE

The ARM toolchain is required 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. The supported version of the toolchain at the time of writing this document is 4_9_2015q3.

The development kit is pre-flashed with Wireless Microcontroller Demo Project Firmware.

2 Development Toolchain Requirements

For development purposes, the minimum requirement is the ARM toolchain in addition to the tools bundled with the SDK.

2.1 GNU Toolchain

The SDK officially supports the GCC Compiler toolchain. The cross-compiler toolchain for GNU ARM is available at the following URL: <https://launchpad.net/gcc-arm-embedded/4.9/4.9-2015-q3-update>

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.

The current version of the gcc-arm-embedded toolchain at the time of writing this document is 4_9_2015q3. And this is the version we recommend.

2.2 Linux Toolchain Setup Procedure

Setting up the GCC toolchain in Linux requires the steps outlined below.

- Download the toolchain tarball available at <https://launchpad.net/gcc-arm-embedded/4.9/4.9-2015-q3-update>
Linux: `gcc-arm-none-eabi-4_9-2015q3-20150921-linux.tar.bz2`
- Copy the file to a directory of your choice. Please ensure there are no spaces in the directory name.
- Untar the file using the command:

```
tar -vxf <filename>
```
- Add the path of the installed toolchain to system PATH. For example, append the following line at the end of the `.profile` file located in `/home/<user>` directory:

```
PATH=$PATH:<path to>gcc-arm-none-eabi-4_9_2015_q3/bin
```

Note: Newer distributions of Ubuntu might come with a Debian version of the GCC Cross Compiler. It is imperative that the native Cross Compiler is removed. The above setup procedure should be followed.

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 can be used, however it is recommended to upgrade to the most recent release. The following steps are explained and tested to work on Ubuntu 16.04.

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 auto detect the machine type and install the appropriate software viz. C libraries, USB library, FTDI library, ncurses, python and latex.

In this document, the generic folder name `amzsdk_bundle-x.y.z` is used to indicate Amazon SDK root folder. The actual folder name may be different.

Make sure you have root privileges, then go to `amzsdk_bundle-x.y.z/` directory and run the following command:

```
#./lib/third_party/mcu_vendor/marvell/WMSDK/mw320/sdk/tools/bin/installpkgs.sh
```

3.1.1 Avoiding 'sudo'

In this document, "flashprog" operation refers to the act of flashing the NAND of the board using *flashprog.py* script, as explained further in the document. Similarly, "ramload" operation refers to copying the firmware image from the host directly to the RAM of the microcontroller, without flashing the NAND, and using *ramload.py* script.

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 running the following command:

```
#./lib/third_party/mcu_vendor/marvell/WMSDK/mw320/sdk/tools/bin/perm_fix.sh
```

Note: Fixing these permissions is mandatory for ensuring a smooth Eclipse-IDE-based experience.

3.2 Setting up the Serial Console

1. Insert the USB cable into the Linux host USB slot. This triggers 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 shown above, two `ttyUSB` devices have been created. The second `ttyUSB` is the serial console, in our case `ttyUSB1`.

3. Use the following command to execute `minicom` in setup mode:

```
minicom -s
```

Alternatively, use other serial programs such as `putty`.

4. Go to Serial Port Setup and capture 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.

5. Hit enter on the serial console window. This should display a hash (#) on the screen.

Note: The development boards include an FTDI silicon device. The FTDI device exposes two USB interfaces for the host. The first interface is associated to the JTAG functionality of the MCU and the second interface is associated with the physical UARTx port of the MCU.

3.3 Installing OpenOCD

OpenOCD is a software that aims to provide debugging, in-system programming and boundary-scan testing for embedded target devices.

OpenOCD version 0.9 is required. It is also required for Eclipse functionality. If an earlier version such as version 0.7 was installed on your Linux Host, please remove that repository with the appropriate command for the Linux distribution you are currently using.

OpenOCD can be installed with the standard linux command:

```
apt-get install openocd
```

If the above-mentioned command does not install v0.9 or higher, use the following procedure to download the source for openocd:

- Install libusb-1.0: `sudo apt-get install libusb-1.0`
- Download openocd 0.9.0 (We get it in the form of source code) from <http://openocd.org/>
- Extract openocd and navigate to its directory
- Configure openocd using the following command: `./configure --enable-ftdi --enable-jlink`
- Run the make utility to compile openocd: `make install`

4 Setting up Eclipse

Note: Please make sure that you have performed the steps listed in [Section 3.1.1 "Avoiding 'sudo'"](#).

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

4.1 Download and Installation

4.1.1 Java Run Time Environment

Eclipse requires Java Run Time Environment (JRE) to be installed. It is recommended to install this first, although it can be installed after the installation of Eclipse. The JRE version (32/64 bit) must match the version of Eclipse (32/64 bit) that will be installed. JRE can be downloaded from the URL: <http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>

4.1.2 Eclipse

Download and install "Eclipse IDE for C/C++ Developers" from <http://www.eclipse.org>. The supported Eclipse version is Eclipse 4.9.0 or above. The installation only requires to extract the downloaded archive. The platform-specific Eclipse executable can be run to start the application. Read more about Eclipse installation and usage in [Section 5.3 "Working with Eclipse"](#).

5 Building and Running Amazon FreeRTOS Demo Project

There are two ways to run the Amazon FreeRTOS Demo Project:

1. Using the command line, covered in [Section 5.2](#).
2. Using Eclipse IDE, covered in [Section 5.3](#).

5.1 Provisioning

Depending whether you want to use the test or demo application, set the provisioning data in the following files:

- `./tests/common/include/aws_clientcredential.h`
- `./demos/common/include/aws_clientcredential.h`

For example:

```
#define clientcredentialWIFI_SSID      "Paste Wi-Fi SSID here"
#define clientcredentialWIFI_PASSWORD "Paste Wi-Fi password here"
#define clientcredentialWIFI_SECURITY "Paste Wi-Fi Security"
```

Notes:

- Possible values are: `eWiFiSecurityOpen`, `eWiFiSecurityWEP`, `eWiFiSecurityWPA`, and `eWiFiSecurityWPA2`
- SSID and Passphrase should be enclosed in quotes " "

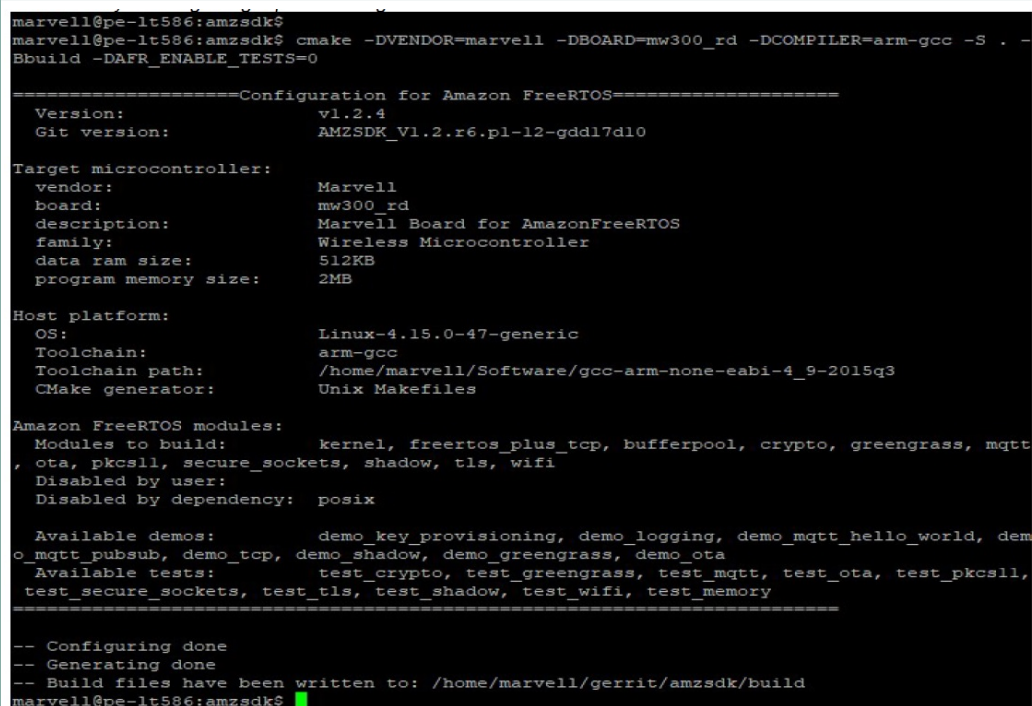
5.2 Working with Command Line

5.2.1 Building

Use the following command to start building the demo application:

```
$ cmake -DVENDOR=marvell -DBOARD=mw300_rd -DCOMPILER=arm-gcc -S . -Bbuild -DAFR_ENABLE_TESTS=0
```

Make sure you get the same output as shown in the following figure.



```
marvell@pe-lt586:amzsdk$ cmake -DVENDOR=marvell -DBOARD=mw300_rd -DCOMPILER=arm-gcc -S . -Bbuild -DAFR_ENABLE_TESTS=0
=====Configuration for Amazon FreeRTOS=====
Version:          v1.2.4
Git version:      AMZSDK_V1.2.r6.pl-12-gdd17d10

Target microcontroller:
  vendor:         Marvell
  board:          mw300_rd
  description:    Marvell Board for AmazonFreeRTOS
  family:         Wireless Microcontroller
  data ram size:  512KB
  program memory size: 2MB

Host platform:
  OS:             Linux-4.15.0-47-generic
  Toolchain:      arm-gcc
  Toolchain path: /home/marvell/Software/gcc-arm-none-eabi-4_9-2015q3
  CMake generator: Unix Makefiles

Amazon FreeRTOS modules:
  Modules to build: kernel, freertos_plus_tcp, bufferpool, crypto, greengrass, mqtt, ota, pkcs11, secure_sockets, shadow, tls, wifi
  Disabled by user:
  Disabled by dependency: posix

  Available demos: demo_key_provisioning, demo_logging, demo_mqtt_hello_world, demo_mqtt_pubsub, demo_tcp, demo_shadow, demo_greengrass, demo_ota
  Available tests:  test_crypto, test_greengrass, test_mqtt, test_ota, test_pkcs11, test_secure_sockets, test_tls, test_shadow, test_wifi, test_memory

-- Configuring done
-- Generating done
-- Build files have been written to: /home/marvell/gerrit/amzsdk/build
marvell@pe-lt586:amzsdk$
```

Figure 1. Building the demo application

Navigate to *build* directory:

```
cd build
```

Run the make utility to build the application:

```
make all -j4
```

Make sure you get the same output as shown in the following figure.

```
[ 92%] Building C object CMakeFiles/afr_ota.dir/lib/third_party/tinycbor/cborencoder.c.obj
[ 93%] Building C object CMakeFiles/afr_ota.dir/lib/third_party/tinycbor/cborencoder_close_container_checked.c.obj
[ 93%] Building C object CMakeFiles/afr_ota.dir/lib/third_party/tinycbor/cborerrorstrings.c.obj
[ 93%] Building C object CMakeFiles/afr_ota.dir/lib/third_party/tinycbor/cborparser.c.obj
[ 94%] Building C object CMakeFiles/afr_ota.dir/lib/third_party/tinycbor/cborparser_dup_string.c.obj
[ 94%] Building C object CMakeFiles/afr_ota.dir/lib/third_party/tinycbor/cborpretty.c.obj
[ 94%] Building C object CMakeFiles/afr_ota.dir/lib/third_party/jsmn/jsmn.c.obj
[ 95%] Building C object CMakeFiles/afr_ota.dir/demos/common/logging/aws_logging_task_dynamic_buffers.c.obj
[ 95%] Building C object CMakeFiles/afr_ota.dir/demos/common/demo_runner/aws_demo_runner.c.obj
[ 95%] Linking C static library afr_ota.a
[ 95%] Built target afr_ota
[ 96%] Linking C executable aws_demos.axf
[100%] Built target aws_demos
marvell@pe-lt586:build$
```

Figure 2. Building the demo application

Use the following commands to build a test application:

```
$ cmake -DVENDOR=marvell -DBOARD=mw300_rd -DCOMPILER=arm-gcc -S .
-Bbuild -DAFR_ENABLE_TESTS=1

$ cd build

$ make all -j4
```

Run the `cmake` command every time you switch between the `aws_demos` project and the `aws_tests` project.

5.2.2 Loading to Flash

This method writes the firmware image to the flash of the development board. The firmware then gets executed after a reset of the development board.

Note that you need to build the SDK before flashing the image to the microcontroller.

5.2.2.1 Loading Layout and Boot2

Before flashing the firmware image, prepare the development board flash with some common components, namely Layout and Boot2. Use the following commands:

```
$ cd amzsdk_bundle-x.y.z

$ ./vendors/marvell/WMSDK/mw320/sdk/tools/OpenOCD/flashprog.py
-l ./vendors/marvell/WMSDK/mw320/sdk/tools/OpenOCD/mw300/
layout.txt --boot2 ./vendors/marvell/WMSDK/mw320/boot2/bin/
boot2.bin
```

The command initiates the following:

- **Layout:** The flashprog utility is first instructed to write a layout to the flash. The layout is like a partition information of the flash. The default layout is located at `/lib/third_party/mcu_vendor/marvell/WMSDK/mw320/sdk/tools/OpenOCD/mw300/layout.txt`.
- **boot2:** This is the boot-loader used by the WMSDK. The flashprog is also writing a bootloader to the flash. It is the boot-loader's job to load the microcontroller's firmware image once we flash it subsequently.

Make sure you get the same output as shown in the figure below.

```
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00007f14 msp: 0x20001000
29088 bytes written at address 0x00100000
downloaded 29088 bytes in 0.245458s (115.728 KiB/s)
verified 29088 bytes in 0.350004s (81.160 KiB/s)
semihosting is enabled

Flashprog version: 2.1.0
Erasing primary flash...done
Writing new flash layout...done
Writing "boot2" @0x0 (primary)...done
semihosting: *** application exited ***
Flashprog Complete
shutdown command invoked

target state: halted
target halted due to breakpoint, current mode: Thread
xPSR: 0x21000000 pc: 0x00100658 msp: 0x0015ffe4, semihosting
```

Figure 3. Output of the command to flash Layout and boot2 elements

5.2.2.2 Flashing the Wi-Fi Firmware

The firmware uses the Wi-Fi chipset for its functionality. The Wi-Fi chipset has its own firmware that must also be present in the flash. The *flashprog.py* utility is used to flash the Wi-Fi firmware, as it was used earlier to flash the *boot2* boot-loader and the MCU firmware. Use the following commands to flash the Wi-Fi firmware:

```
$ cd amzsdk_bundle-x.y.z
$ ./vendors/marvell/WMSDK/mw320/sdk/tools/OpenOCD/flashprog.py
--wififw ./vendors/marvell/WMSDK/mw320/wifi-firmware/mw30x/
mw30x_uapsta_W14.88.36.p135.bin
```

Make sure the output of the command is similar to the one shown in the following figure.

```
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00007f14 msp: 0x20001000
29088 bytes written at address 0x00100000
downloaded 29088 bytes in 0.245498s (115.709 KiB/s)
verified 29088 bytes in 0.350229s (81.108 KiB/s)
semihosting is enabled

Flashprog version: 2.1.0
Writing "wififw" @0x12a000 (primary).....done
semihosting: *** application exited ***
Flashprog Complete
shutdown command invoked

target state: halted
target halted due to breakpoint, current mode: Thread
xPSR: 0x21000000 pc: 0x00100658 msp: 0x0015ffe4, semihosting
```

Figure 4. Flashing the Wi-Fi Firmware

5.2.2.3 Loading the MCU Firmware

Once you have flashed *layout* and *boot2* as detailed in [Section 5.2.2.1 "Loading Layout and Boot2"](#), use the following commands to flash the MCU firmware.

```
$ cd amzsdk_bundle-x.y.z  
  
$ ./vendors/marvell/WMSDK/mw320/sdk/tools/OpenOCD/flashprog.py --  
mcufw build/cmake/vendors/marvell/mw300_rd/aws_demos.bin -r
```

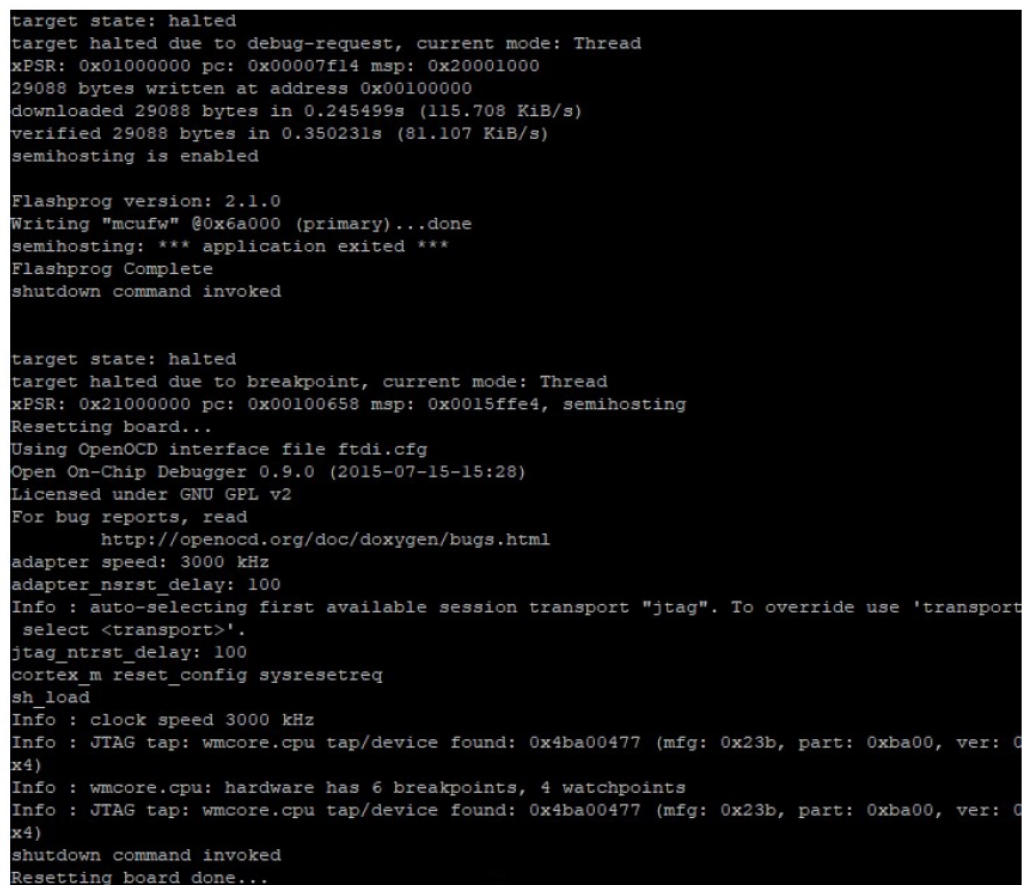
Reset the board.

You should see the logs for the demo app.

To run the test app, flash the *aws_tests.bin* binary located at the same location.

```
$ cd amzsdk_bundle-x.y.z  
  
$ ./vendors/marvell/WMSDK/mw320/sdk/tools/OpenOCD/flashprog.py --  
mcufw build/cmake/vendors/marvell/mw300_rd/aws_tests.bin -r
```

Your command output should be similar to the one shown in the figure below.



```
target state: halted  
target halted due to debug-request, current mode: Thread  
xPSR: 0x01000000 pc: 0x00007f14 msp: 0x20001000  
29088 bytes written at address 0x00100000  
downloaded 29088 bytes in 0.245499s (115.708 KiB/s)  
verified 29088 bytes in 0.350231s (81.107 KiB/s)  
semihosting is enabled  
  
Flashprog version: 2.1.0  
Writing "mcufw" @0x6a000 (primary)...done  
semihosting: *** application exited ***  
Flashprog Complete  
shutdown command invoked  
  
target state: halted  
target halted due to breakpoint, current mode: Thread  
xPSR: 0x21000000 pc: 0x00100658 msp: 0x0015ffe4, semihosting  
Resetting board...  
Using OpenOCD interface file ftdi.cfg  
Open On-Chip Debugger 0.9.0 (2015-07-15-15:28)  
Licensed under GNU GPL v2  
For bug reports, read  
    http://openocd.org/doc/doxygen/bugs.html  
adapter speed: 3000 kHz  
adapter_nsrst_delay: 100  
Info : auto-selecting first available session transport "jtag". To override use 'transport  
select <transport>'.  
jtag_ntrst_delay: 100  
cortex_m reset_config sysresetreq  
sh_load  
Info : clock speed 3000 kHz  
Info : JTAG tap: wmc core.cpu tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0xba00, ver: 0  
x4)  
Info : wmc core.cpu: hardware has 6 breakpoints, 4 watchpoints  
Info : JTAG tap: wmc core.cpu tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0xba00, ver: 0  
x4)  
shutdown command invoked  
Resetting board done...
```

Figure 5. Flashing the MCU Firmware

5.2.3 Starting the Demo App

Once you have flashed the firmware and reset the board, the Demo app should start as shown in the figure below.

```
Network connection successful.
Wi-Fi Connected to AP. Creating tasks which use network...
2 6293 [Startup Hook] Write certificate...
3 6296 [Startup Hook] Write device private key...
4 6362 [Startup Hook] Creating MQTT Echo Task...
6 11668 [MQTTEcho] MQTT echo connected, to connect to a2wtm15blvjjs8-ats.iot.us-east-2.amazonaws.com
7 11668 [MQTTEcho] MQTT echo test echoing task created.
8 11961 [MQTTEcho] MQTT Echo demo subscribed to freertos/demos/echo
9 12248 [MQTTEcho] Echo successfully published 'Hello World 0'
10 12591 [Echoing] Message returned with ACK: 'Hello World 0 ACK'
11 17633 [MQTTEcho] Echo successfully published 'Hello World 1'
12 17927 [Echoing] Message returned with ACK: 'Hello World 1 ACK'
13 22953 [MQTTEcho] Echo successfully published 'Hello World 2'
14 23276 [Echoing] Message returned with ACK: 'Hello World 2 ACK'
15 28245 [MQTTEcho] Echo successfully published 'Hello World 3'
16 28575 [Echoing] Message returned with ACK: 'Hello World 3 ACK'
17 33542 [MQTTEcho] Echo successfully published 'Hello World 4'
18 33980 [Echoing] Message returned with ACK: 'Hello World 4 ACK'
19 38823 [MQTTEcho] Echo successfully published 'Hello World 5'
20 39279 [Echoing] Message returned with ACK: 'Hello World 5 ACK'
21 44139 [MQTTEcho] Echo successfully published 'Hello World 6'
22 44501 [Echoing] Message returned with ACK: 'Hello World 6 ACK'
23 49516 [MQTTEcho] Echo successfully published 'Hello World 7'
24 50270 [Echoing] Message returned with ACK: 'Hello World 7 ACK'
25 54796 [MQTTEcho] Echo successfully published 'Hello World 8'
26 55129 [Echoing] Message returned with ACK: 'Hello World 8 ACK'
27 60080 [MQTTEcho] Echo successfully published 'Hello World 9'
28 60389 [Echoing] Message returned with ACK: 'Hello World 9 ACK'
29 65378 [MQTTEcho] Echo successfully published 'Hello World 10'
30 65998 [Echoing] Message returned with ACK: 'Hello World 10 ACK'
31 70658 [MQTTEcho] Echo successfully published 'Hello World 11'
32 70964 [Echoing] Message returned with ACK: 'Hello World 11 ACK'
33 75958 [MQTTEcho] MQTT echo demo finished.
34 75958 [MQTTEcho] ----Demo finished----
```

Figure 6. Demo App Start

5.2.4 Loading to SRAM

As an alternative method to test your image, use the flashprog utility to copy the microcontroller image from the host directly into the microcontroller RAM. The image is not copied in the flash therefore it gets lost after rebooting the microcontroller.

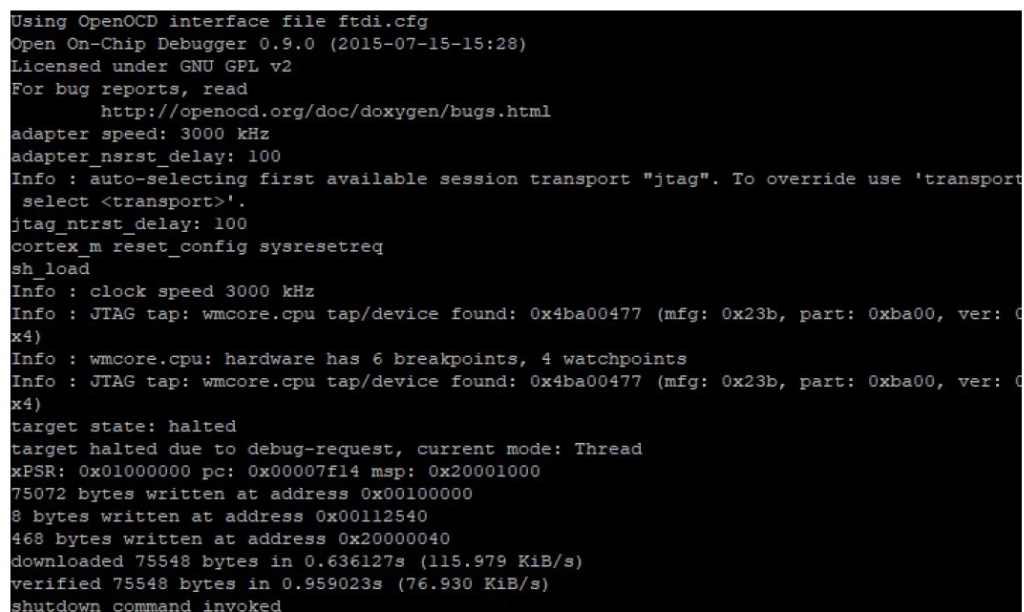
Loading the firmware image into the SRAM is a faster operation as it launches the execution straight away. This method is most used for iterative development.

Use the following commands to load the firmware into the SRAM.

```
$ cd amzsdk_bundle-x.y.z

$ ./lib/third_party/mcu_vendor/marvell/WMSDK/mw320/sdk/tools/
OpenOCD/ramload.py build/cmake/vendors/marvell/mw300_rd/
aws_demos.axf
```

The command output is shown in the figure below.



```
Using OpenOCD interface file ftdi.cfg
Open On-Chip Debugger 0.9.0 (2015-07-15-15:28)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
adapter speed: 3000 kHz
adapter_nsrst_delay: 100
Info : auto-selecting first available session transport "jtag". To override use 'transport
select <transport>'.
jtag_ntrst_delay: 100
cortex_m reset_config sysresetreq
sh_load
Info : clock speed 3000 kHz
Info : JTAG tap: wmc0re.cpu tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0xba00, ver: 0
x4)
Info : wmc0re.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : JTAG tap: wmc0re.cpu tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0xba00, ver: 0
x4)
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00007f14 msp: 0x20001000
75072 bytes written at address 0x00100000
8 bytes written at address 0x00112540
468 bytes written at address 0x20000040
downloaded 75548 bytes in 0.636127s (115.979 KiB/s)
verified 75548 bytes in 0.959023s (76.930 KiB/s)
shutdown command invoked
```

Figure 7. Loading the Firmware Image into the SRAM

When the command execution is complete, you should see the logs of the Demo app.

5.3 Working with Eclipse

Before setting up an Eclipse workspace, run the `cmake` command as explained in [Section 5.2.1 "Building"](#)

To work with the `aws_demos` Eclipse project, use the command line to run the following command:

```
$ cmake -DVENDOR=marvell -DBOARD=mw300_rd -DCOMPILER=arm-gcc -S .  
-Bbuild -DAFR_ENABLE_TESTS=0
```

To work with the `aws_tests` Eclipse project, use the command line to run the following command:

```
$ cmake -DVENDOR=marvell -DBOARD=mw300_rd -DCOMPILER=arm-gcc -S .  
-Bbuild -DAFR_ENABLE_TESTS=1
```

Make sure to run the `cmake` command every time you switch between the `aws_demos` project and the `aws_tests` project.

When prompted, select your Eclipse workspace as shown in the figure below.

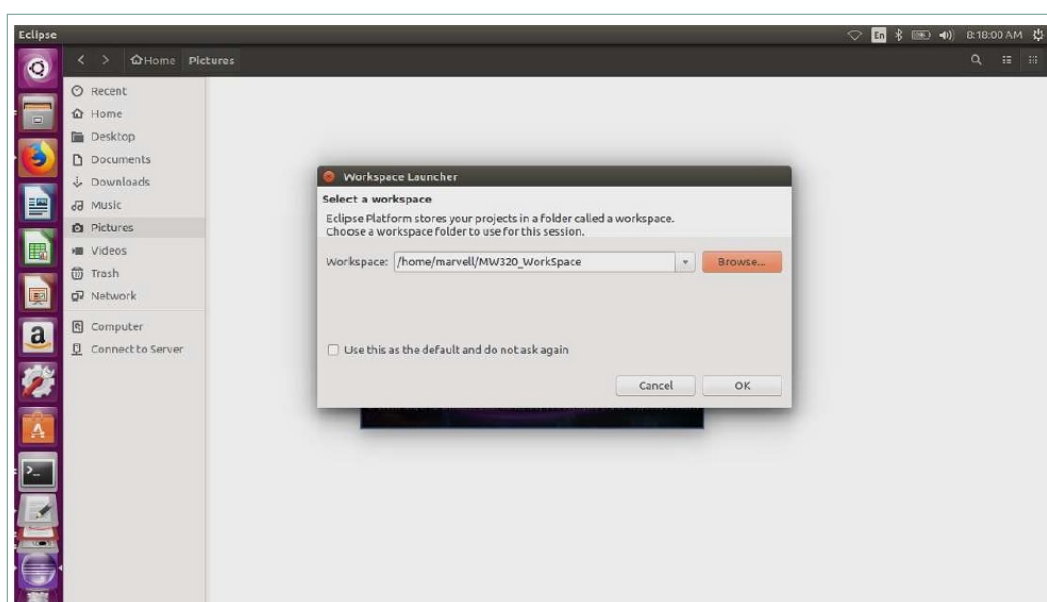


Figure 8. Selecting the Eclipse Workspace

Select the option to create a Makefile project with existing code as shown in the figure below.

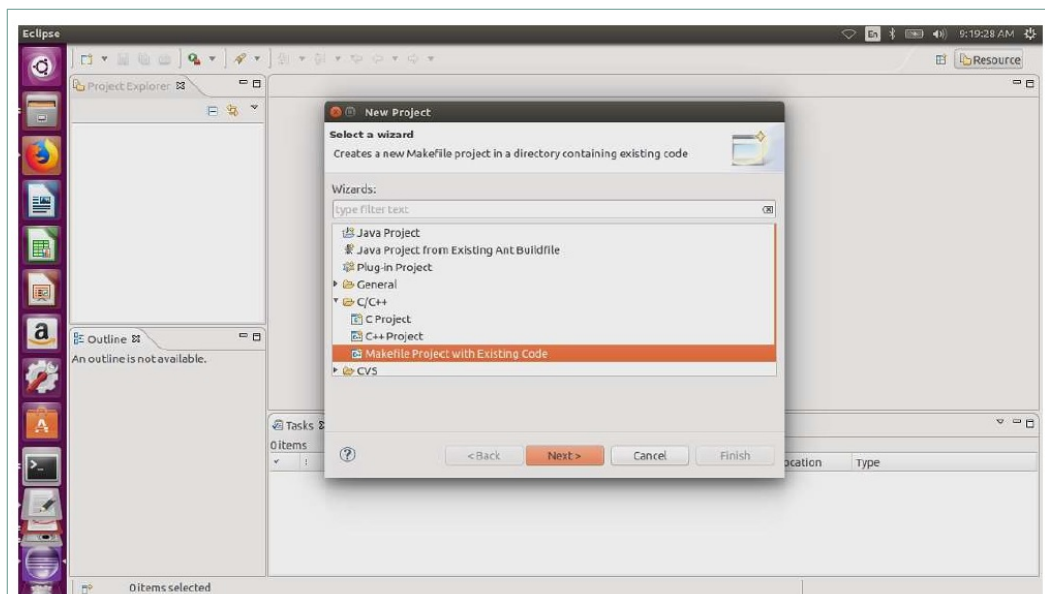


Figure 9. Creating Makefile Project with Existing Code

Browse to locate the directory of the existing code and click on **Finish**.

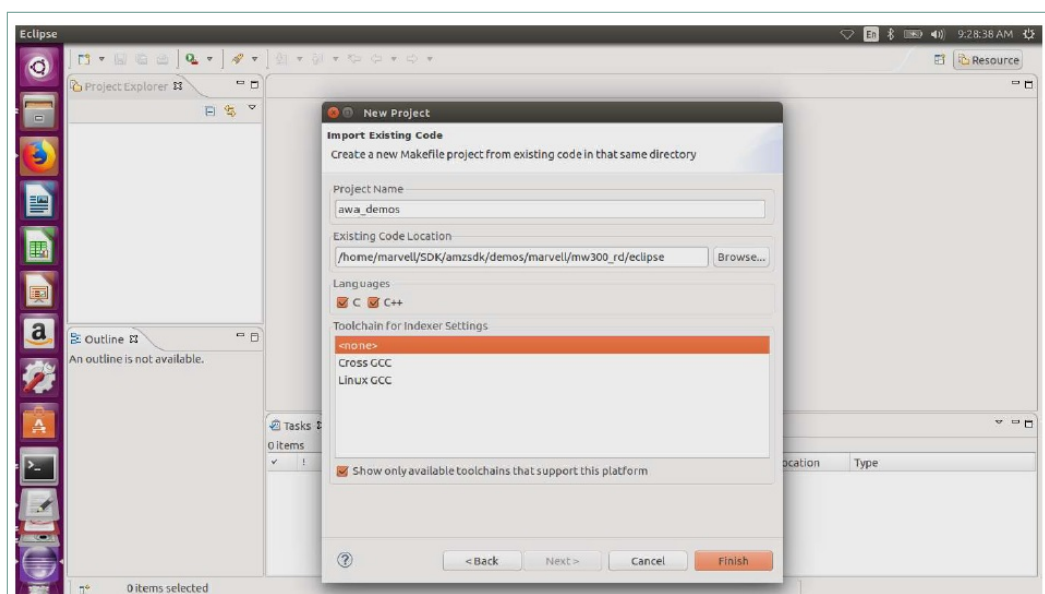


Figure 10. Browsing to Locate the Existing Code

Select `aws_demos` in the project explorer shown on the left navigation pane. Apply a right-mouse click to open the menu and select **Build**.

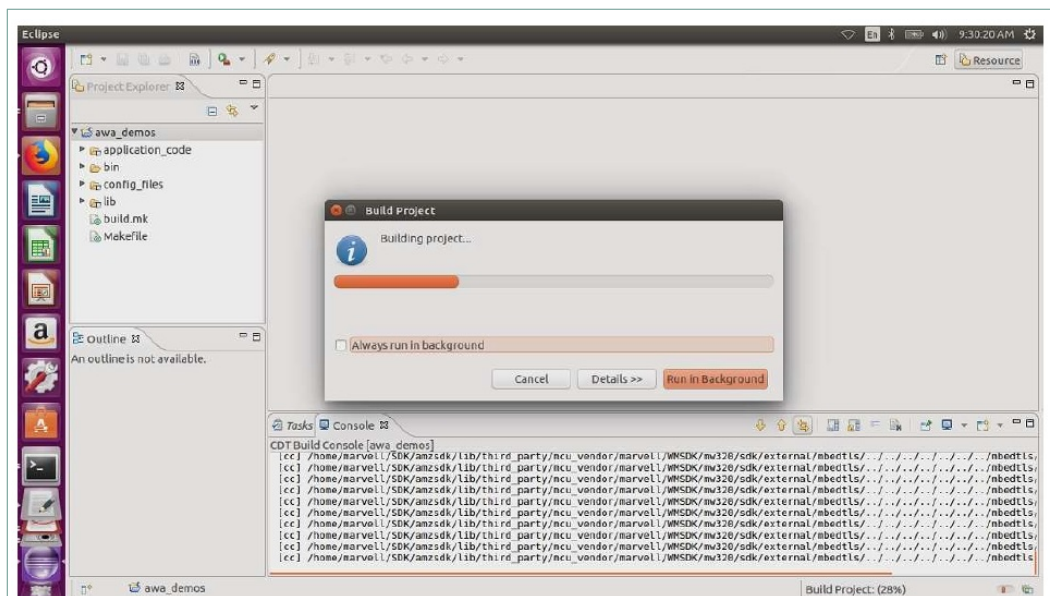


Figure 11. Building aws_demos Project

Upon successful build, *aws_demos.bin* is generated at the following location:

build/cmake/vendors/marvell/mw300_rd/aws_demos.bin

Use the tools to flash the Layout file (*layout.txt*), the Boot2 binary (*boot2.bin*), the MCU Firmware Binary (*aws_demos.bin*), and the WiFi Firmware.

6 Troubleshooting

6.1 Network Issue

Double check your network credentials. Refer to [Section 5.1 "Provisioning"](#).

6.2 Enabling Additional Logs

Enabling board specific logs

Enable calls to `wmstdio_init(UART0_ID, 0)` in the function *prvMiscInitialization* file *main.c* file (tests or demos)

Enabling Wi-Fi logs

Enable macro `CONFIG_WLCMGR_DEBUG` in *AFR_HOME/lib/third_party/mcu_vendor/marvell/WMSDK/mw320/sdk/src/incl/autoconf.h* file.

6.3 Using GDB

It is recommended to use *arm-none-eabi-gdb* and *gdb* command files packaged along with the SDK.

Navigate to the relevant directory.

```
cd AFR_HOME/lib/third_party/mcu_vendor/marvell/WMSDK/mw320
```

Run the following command to connect to GDB.

```
arm-none-eabi-gdb - x ./sdk/tools/OpenOCD/  
gdbinit ../../../../../../build/cmake/vendors/marvell/mw300 _rd/  
aws_demos.axf
```

7 Abbreviations

Table 1. Abbreviations

Acronym	Description
AWS	Amazon Web Services
IDE	Integrated Development Environment
JRE	Java Run Time Environment
SDK	Software Development Kit
WMSDK	Wireless Microcontroller SDK

8 Legal information

8.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

8.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors. In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory. Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of

customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer. In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out of the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages. Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US\$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — While NXP Semiconductors has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP Semiconductors accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

8.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

Tables

Tab. 1.	Abbreviations	18
---------	---------------------	----

Figures

Fig. 1.	Building the demo application	9	Fig. 6.	Demo App Start	13
Fig. 2.	Building the demo application	10	Fig. 7.	Loading the Firmware Image into the SRAM ...	14
Fig. 3.	Output of the command to flash Layout and boot2 elements	11	Fig. 8.	Selecting the Eclipse Workspace	15
Fig. 4.	Flashing the Wi-Fi Firmware	11	Fig. 9.	Creating Makefile Project with Existing Code ...	16
Fig. 5.	Flashing the MCU Firmware	12	Fig. 10.	Browsing to Locate the Existing Code	16
			Fig. 11.	Building aws_demos Project	17

Contents

1	Introduction	3
2	Development Toolchain Requirements	3
2.1	GNU Toolchain	3
2.2	Linux Toolchain Setup Procedure	4
3	Working with a Linux Development Host	4
3.1	Installing Packages	4
3.1.1	Avoiding 'sudo'	5
3.2	Setting up the Serial Console	6
3.3	Installing OpenOCD	7
4	Setting up Eclipse	7
4.1	Download and Installation	7
4.1.1	Java Run Time Environment	7
4.1.2	Eclipse	7
5	Building and Running Amazon FreeRTOS Demo Project	8
5.1	Provisioning	8
5.2	Working with Command Line	9
5.2.1	Building	9
5.2.2	Loading to Flash	10
5.2.2.1	Loading Layout and Boot2	10
5.2.2.2	Flashing the Wi-Fi Firmware	11
5.2.2.3	Loading the MCU Firmware	12
5.2.3	Starting the Demo App	13
5.2.4	Loading to SRAM	14
5.3	Working with Eclipse	15
6	Troubleshooting	18
6.1	Network Issue	18
6.2	Enabling Additional Logs	18
6.3	Using GDB	18
7	Abbreviations	18
8	Legal information	19

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.
