FreeRTOS-MAVID (EVB) Quick Start Guide

Revision: 1.2

Libre Wireless Technologies Private Limited

librewireless.com

Table of Contents

Table of Figures

1. Document Information

1.1. Abstract

This document explains the content of MAVID package for FreeRTOS.

1.2. Document Convention

Icon	Meaning	Description
Note:	Note	Provides information good to know
CAUTION	Caution	Indicates situation that might result in loss of data or hardware damage

1.3. Document Revision History

Revision	Date	Description of change	Author
1.2	July 19, 2020	Added section Update Credentials	Sachin
1.1	June 25, 2020	Made changes to section 3.4, 3.4.1, 3.4.3, 3.4.4	Sachin
1.0	May 18, 2020	Added section 5	Ramya
0.1	May 08, 2020	Initial Draft	Ramya

2. FreeRTOS MAVID (EVB)

2.1 MAVID Evaluation Board (EVB)

EVB is a MAVID device-based reference design, used for customer evaluation purpose.

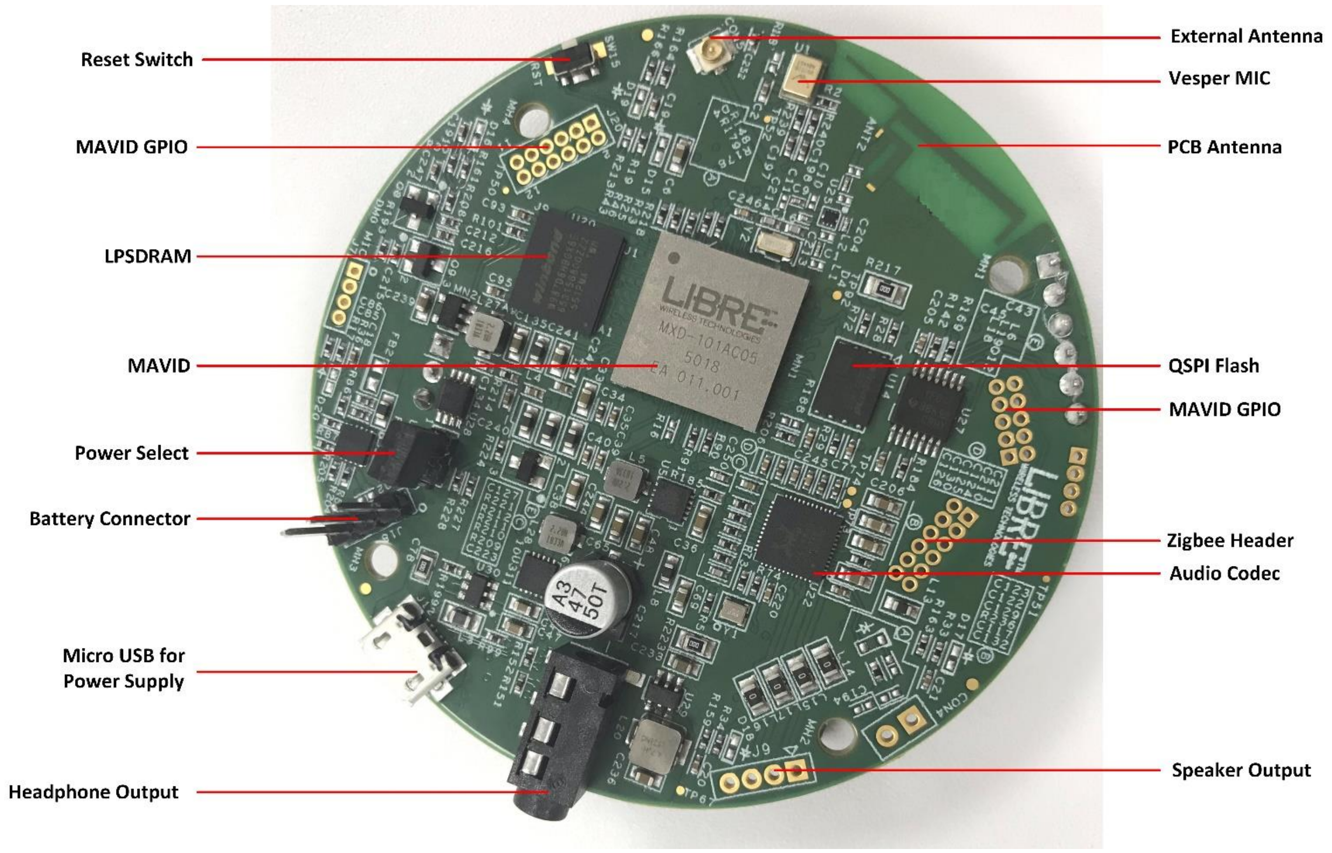


Figure 2.1-1: EVB PCBA Top View

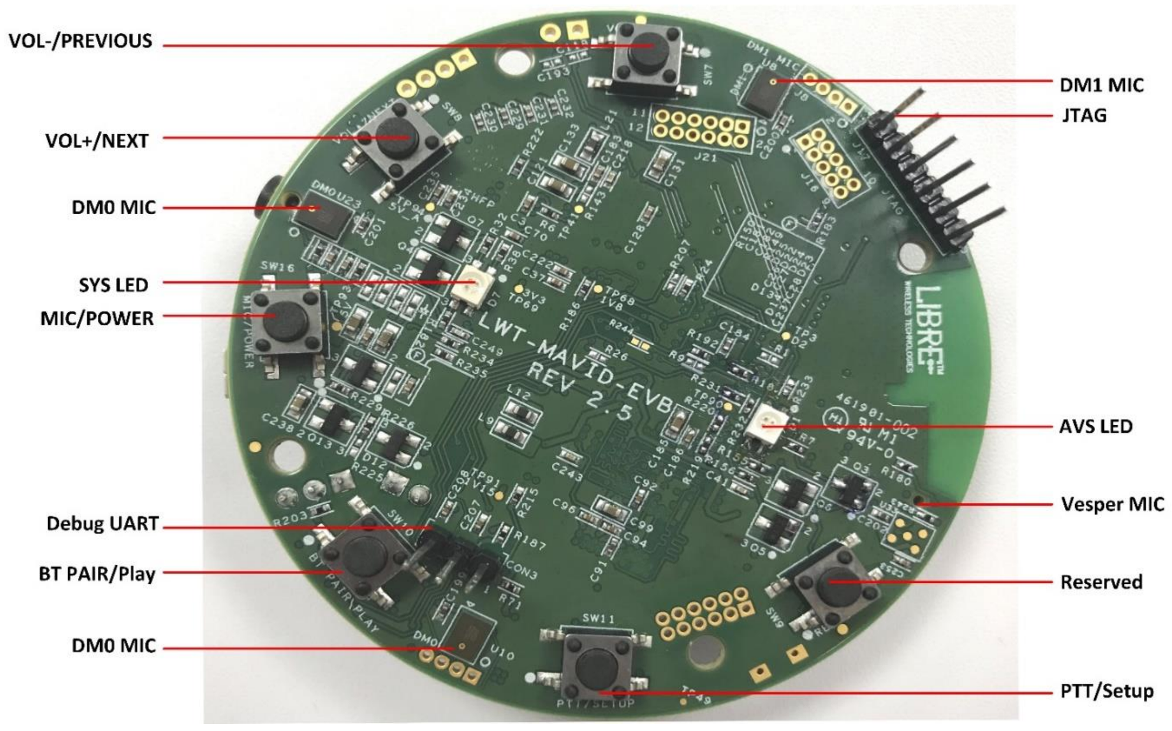


Figure 2.1-2: EVB PCBA Bottom View

Refer the sections: UART debugging, Network setup, MAC ID update, and APP usage for more details.

2.2 EVB Product UI

The MAVID EVB board has the following Switches and defined functionality as mentioned below:

Switch#	Short press Functionality	Long press Functionality	MAVID MCU port	RTC GPIO	MAVID Pin#
SW11	Push-To-Talk (Action)	SETUP	PI11	YES	P57
SW16	MIC ON/OFF	POWER ON/OFF	PA0	YES	P109
SW10	Play/Pause	BT pairing	PB4	NO	P84
SW7	VOL-	PREV	PD13	NO	P110
SW8	VOL+	NEXT	PF10	NO	P111
SW9	RESERVED	RESERVED	PD6	NO	P86

- The Top Side of EVB board there is right angled button for Reset.
- Simultaneous press of SW10 and SW11 results in Factory Default Reset (FDR) functionality.
- SW9 is reserved for user to define the functionality.

Here are the supported features in Libre's current EVB Rev 2.0.

- Micro USB for charging battery/Micro USB for powering up the EVB
- Battery charging circuit
- 6 nos. of buttons + 1 reset button
  - MIC mute/POWER ON-POWER OFF
  - Network setup/PTT
  - Play-pause/BT paring
  - VOL-/PREV
  - VOL+/NEXT
  - Pin hole for RST
  - SW11: Network setup (long press) and short press for (action – push to talk)
  - SW16: Power ON/POWER OFF (long press) and short press for Mic mute (when the unit is ON)
  - SW10: BT pair (long press) and short press for play/pause
  - SW15: Only reset (to reboot MAVID device)
  - J17: JTAG (it is for programming and debugging)
  - CON3: Debug port connection
  - J21: Zigbee Interface connector header
  - J20, J16: MAVID for unused GPOIs
  - TP3: AEC tuning
  - TP92: D4 debug port

3. Libre FreeRTOS IoT Application Development

The Libre application development platform provides the tools and development environment for custom application development.

It is based on the latest AWS FreeRTOS kernel.

3.1. Hardware Requirement

The hardware requirements are as follows:

- MAVID Development Kit
- STlink debugger
- USB TTL Serial cable
- Power adapter of 5V /2A
  for operating the device

3.2. Software Requirement

The software requirements are as follows:

- IAR version 7.70.1

  Download the IAR version 7.70.1 from www.iar.com

  Note: This SDK is compatible with IAR version 7.70.1

- STM32 ST-LINK Utility v4.2.0

  Download the ST-Link utility from https://stm32-st-link-utility.software.informer.com/download/

- Teraterm/Putty/GTK-term or any similar serial port terminal application

  Download Teraterm from https://tera-term.en.softonic.com/download

- Libre software development kit (contains software development SDK and ENV tool to generate custom ENV binary)

3.3. Hardware Setup

Follow the below steps to setup the hardware components:

Step 1. Connect the USB TTL cable to the device as shown below:

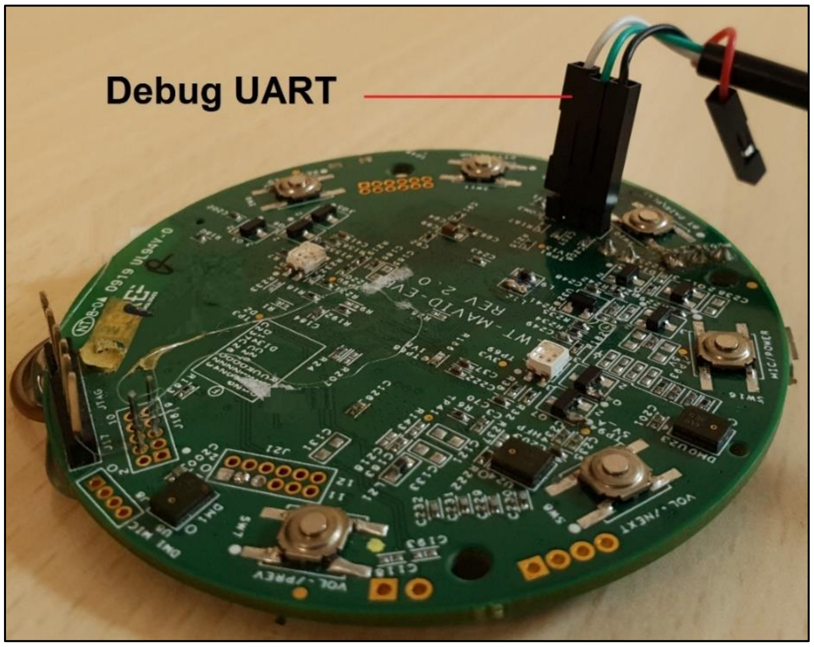


Figure 3.3-1: EVB PCBA Bottom View

Step 2. Connect the micro USB power cable to the device.

Step 3. Launch the serial port terminal on the PC with the following settings:

Baud Rate	460800
Data	8 bits
Parity	none
Stop Bits	1 bit
Flow Control	none
Enable LF on the serial receive	

Step 4. Press button SW15 to Power ON the device.

3.3.1. UART debugging

Libre does run command line interpreter (CLI) over UART, it includes the following capabilities:

- Does have the boot up logs with appropriate initialization state of each module, that point out the errors or exception with details.
- Relevant runtime debugging Information for every module, this will be looked up on to trace bugs.
- The Command Interpreter over UART provides **custom AT commands** such as,
  - Memory analysis -to monitor system memory
  - CPU utilization - to monitor CPU load
  - Modify Config parameters - To change the configuration parameters (NV-items) to change mode

3.4. Application Project Development

3.4.1 Download SDK from GitHub

To clone using HTTPS:

```
git clone https://github.com/aws/amazon-freertos.git --recurse-submodules
```

Using SSH:

```
git clone git@github.com:aws/amazon-freertos.git --recurse-submodules
```

If you have downloaded the repo without using the --recurse-submodules argument, you need to run:

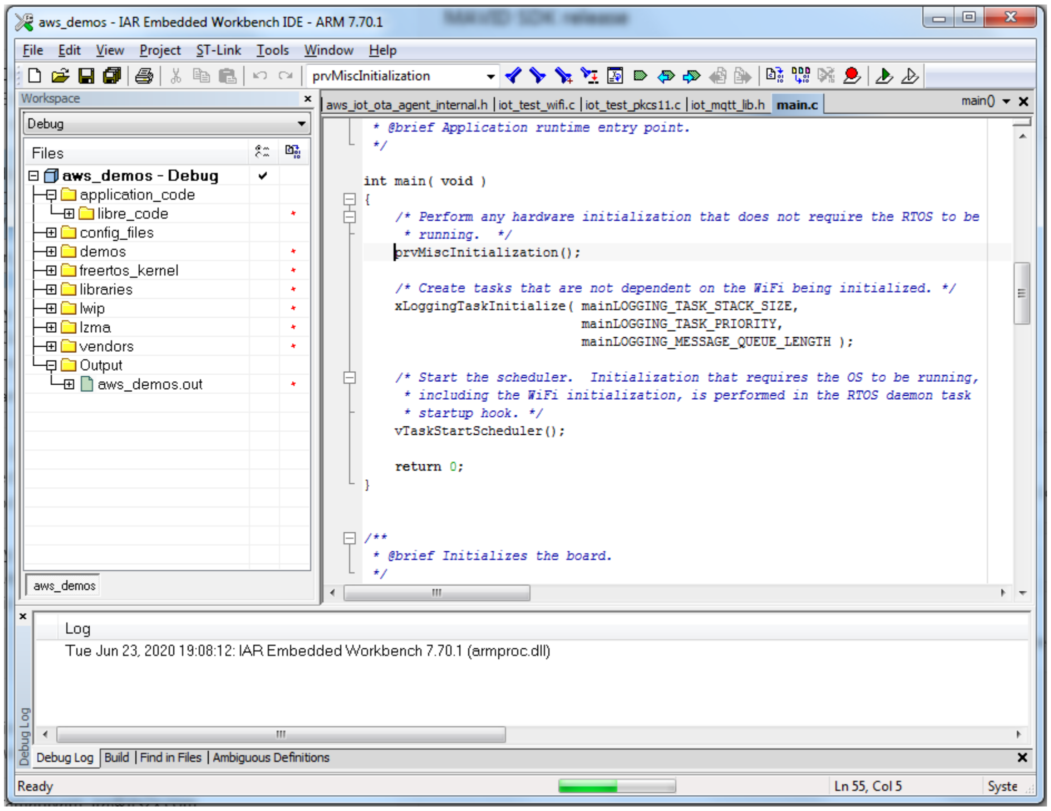
```
git submodule update --init --recursive
```

3.4.2 Launching demo project

Launch the demo project by following the below method:

Run **FREERTOS_BASE_DIR\projects\libre\mavid\iar\ aws_demos\aws_demos.eww**

***FREERTOS_BASE_DIR** is the root directory of the FreeRTOS source code.



```
 * @brief Application runtime entry point.
 */

int main( void )
{
    /* Perform any hardware initialization that does not require the RTOS to be
     * running.  */
    prvMiscInitialization();

    /* Create tasks that are not dependent on the WiFi being initialized. */
    xLoggingTaskInitialize( mainLOGGING_TASK_STACK_SIZE,
                            mainLOGGING_TASK_PRIORITY,
                            mainLOGGING_MESSAGE_QUEUE_LENGTH );

    /* Start the scheduler.  Initialization that requires the OS to be running,
     * including the WiFi initialization, is performed in the RTOS daemon task
     * startup hook. */
    vTaskStartScheduler();

    return 0;
}

/**
 * @brief Initializes the board.
 */
```

```
Log
Tue Jun 23, 2020 19:08:12: IAR Embedded Workbench 7.70.1 (armproc.dll)
```

Figure 3.4.2-1: IAR Project View

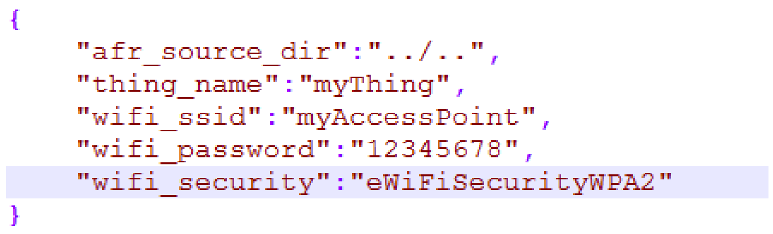
3.4.3 Update Credentials

- Before compiling the demo application follow the steps below to configure the Wi-Fi and IoT Thing credentials
- Install and Configure AWS CLI as explained in https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html

- Confirm that the AWS CLI is successfully installed:

```
C:\> aws --version
aws-cli/2.0.23 Python/3.7.4 Windows/10 botocore/2.0.0
```

- Edit **FREERTOS_BASE_DIR\tools\aws_config_quick_start\configure.json** to update Wi-Fi credentials and thing name:

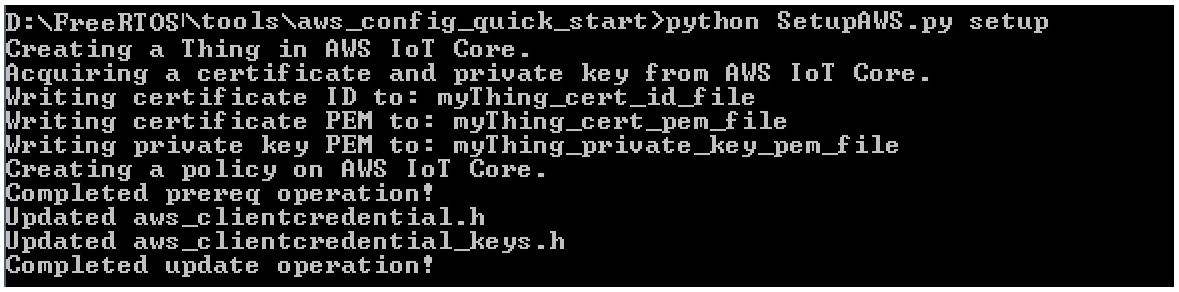


```
{
    "afr_source_dir":"../..",
    "thing_name":"myThing",
    "wifi_ssid":"myAccessPoint",
    "wifi_password":"12345678",
    "wifi_security":"eWiFiSecurityWPA2"
}
```

  Refer to FREERTOS_BASE_DIR\tools\aws_config_quick_start\README.md for more details.

- In the command terminal navigate to path **FREERTOS_BASE_DIR\tools\aws_config_quick_start\**

- From the command shell, run the python script SetupAWS.py with setup option as shown below:



```
D:\FreeRTOS\tools\aws_config_quick_start>python SetupAWS.py setup
Creating a Thing in AWS IoT Core.
Acquiring a certificate and private key from AWS IoT Core.
Writing certificate ID to: myThing_cert_id_file
Writing certificate PEM to: myThing_cert_pem_file
Writing private key PEM to: myThing_private_key_pem_file
Creating a policy on AWS IoT Core.
Completed prereq operation!
Updated aws_clientcredential.h
Updated aws_clientcredential_keys.h
Completed update operation!
```

- The script creates a thing, generates certificates and policies, attaches the certificates and policies to the thing, and updates the *aws_credentials.*h and *aws_credentials_keys.h* files at the path *FREERTOS_BASE_DIR\demos\include\ folder*, based on the inputs in *configure.json*

3.4.4 Compiling MQTT Demo Application

- In the Menu bar select **Project → Clean** to clean the project
- Create an IoT thing in AWS. Assign certificates and policies to the thing, as explained in **3.4.3.**
- To Build MQTT demo application enable **CONFIG_MQTT_DEMO_ENABLED** in *aws_demo_config.h*
- In the Menu bar select **Project → Make** to compile the project.
- After successful compilation, the output file **aws_demos.bin_JTAG.bin** is generated at **FREERTOS_BASE_DIR\ projects\libre\mavid\iar\ aws_demos\Debug\Exe ** folder.
- Load the binary file **aws_demos.bin_JTAG.bin** as explained in the section 3.4.6.Loading and Debugging Demo Application

3.4.4.1 Monitoring MQTT messages on the cloud

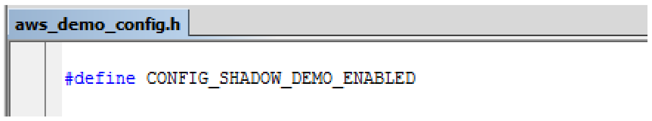
You can use the MQTT client in the AWS IoT console to monitor the messages that your device sends to the AWS Cloud. You might want to set this up before the device runs the demo project.

To subscribe to the MQTT topic with the AWS IoT MQTT client

1. Sign in to the AWS IoT console.
2. In the navigation pane, choose ***Test*** to open the MQTT client.
3. In **Subscription topic**, enter `iotdemo/#`, and then choose ***Subscribe to topic***.

3.4.5 Compiling Demo Application for Device Shadow

- In the Menu bar select **Project → Clean** to clean the project
- Create an IoT thing in AWS. Assign certificates and policies to the thing, as explained in **3.4.3.**
- To Build the Demo Application for MQTT Shadow enable the macro **CONFIG_SHADOW_DEMO_ENABLED** in *aws_demo_config.h*



```
#define CONFIG_SHADOW_DEMO_ENABLED
```

Figure 3.4.5-1: CONFIG_SHADOW_DEMO_ENABLED

- In the Menu bar select **Project → Make** to compile the project.
- After successful compilation, the output file **aws_demos.bin_JTAG.bin** is generated at **FREERTOS_BASE_DIR\projects\libre\mavid\iar\ aws_demos\Debug\Exe ** folder.
- Load the binary file **aws_demos.bin_JTAG.bin** as explained in the section 3.4.6.Loading and Debugging Demo Application

3.4.6 Loading and Debugging Demo Application

1) Loading Demo Application image to MAVID

- Connect the ST-link debugger to the MAVID EVK as shown below:

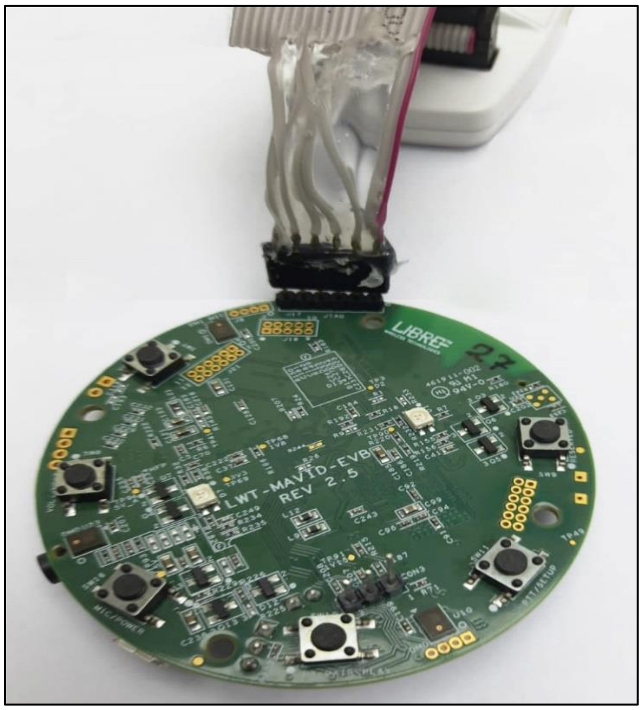


Figure 3.4.6-1: JTAG

- Go to “FREERTOS_BASE_DIR\vendors\libre\mavid\tools\" and run loadApp.bat and give the path of application binary aws_demos.bin_JTAG.bin generated as given in the section 3.4.3.

  E.g.:*loadApp.batD:\FreeRTOS\projects\libre\mavid\iar\aws_demos\Debug\Exe\aws_demos.bin_JTAG.bin*

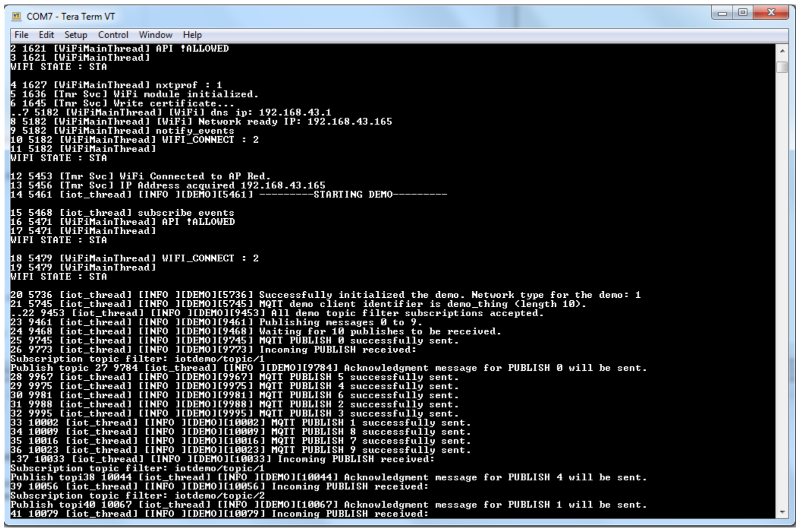
2) Loading ENV binary

- Connect the ST-link debugger to the MAVID EVK as shown in **Figure 3.4.5-1: JTAG**
- Run the bat file **loadENV.bat** located at “FreeRTOS\vendors\libre\mavid\tools\” and provide the path of env.bin as input argument.
  E.g.: loadENV.bat D:\freeRTOS\vendors\libre\mavid\tools\ENV_tool

Note: ENV binary needs to be flashed only when there are any changes made to *envitem.xml*

3) Debugging Application

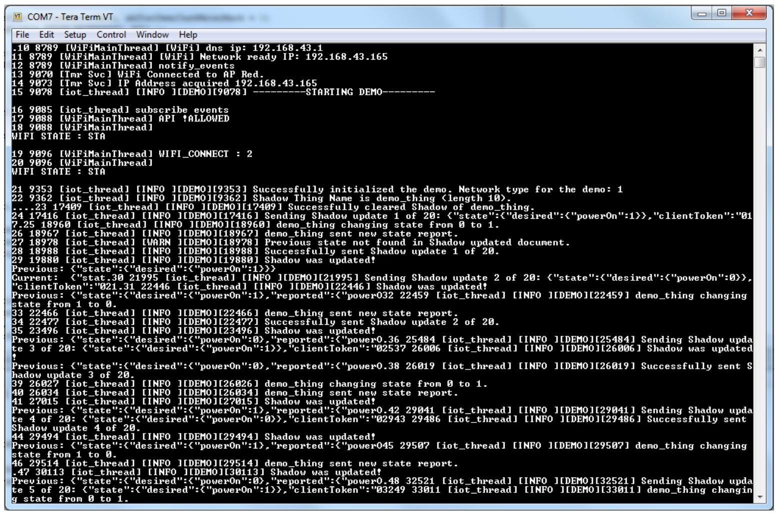
- After the application is programmed, open the project you wish to debug in IAR. Select Project → Attach to Running Target from IAR menu bar. This opens the Debug view.
- Select Debug → Break to break and Debug → Reset. This takes the execution to main(). Select Debug → Go to run.
- The MQTT Demo application runs with logs as shown below:



```
2 1621 [WiFiMainThread] API !ALLOWED
3 1621 [WiFiMainThread]
WIFI STATE : STA

4 1627 [WiFiMainThread] nxtprof : 1
5 1636 [Tmr Svc] WiFi module initialized.
6 1645 [Tmr Svc] Write certificate...
..7 5182 [WiFiMainThread] [WiFi] dns ip: 192.168.43.1
8 5182 [WiFiMainThread] [WiFi] Network ready IP: 192.168.43.165
9 5182 [WiFiMainThread] notify_events
10 5182 [WiFiMainThread] WIFI_CONNECT : 2
11 5182 [WiFiMainThread]
WIFI STATE : STA

12 5453 [Tmr Svc] WiFi Connected to AP Red.
13 5456 [Tmr Svc] IP Address acquired 192.168.43.165
14 5461 [iot_thread] [INFO ][DEMO][5461] ---------STARTING DEMO---------

15 5468 [iot_thread] subscribe events
16 5471 [WiFiMainThread] API !ALLOWED
17 5471 [WiFiMainThread]
WIFI STATE : STA

18 5479 [WiFiMainThread] WIFI_CONNECT : 2
19 5479 [WiFiMainThread]
WIFI STATE : STA

20 5736 [iot_thread] [INFO ][DEMO][5736] Successfully initialized the demo. Network type for the demo: 1
21 5745 [iot_thread] [INFO ][DEMO][5745] MQTT demo client identifier is demo_thing (length 10).
..22 9453 [iot_thread] [INFO ][DEMO][9453] All demo topic filter subscriptions accepted.
23 9461 [iot_thread] [INFO ][DEMO][9461] Publishing messages 0 to 9.
24 9468 [iot_thread] [INFO ][DEMO][9468] Waiting for 10 publishes to be received.
25 9745 [iot_thread] [INFO ][DEMO][9745] MQTT PUBLISH 0 successfully sent.
26 9773 [iot_thread] [INFO ][DEMO][9773] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/1
Publish topic 27 9784 [iot_thread] [INFO ][DEMO][9784] Acknowledgment message for PUBLISH 0 will be sent.
28 9967 [iot_thread] [INFO ][DEMO][9967] MQTT PUBLISH 5 successfully sent.
29 9975 [iot_thread] [INFO ][DEMO][9975] MQTT PUBLISH 4 successfully sent.
30 9981 [iot_thread] [INFO ][DEMO][9981] MQTT PUBLISH 6 successfully sent.
31 9988 [iot_thread] [INFO ][DEMO][9988] MQTT PUBLISH 2 successfully sent.
32 9995 [iot_thread] [INFO ][DEMO][9995] MQTT PUBLISH 3 successfully sent.
33 10002 [iot_thread] [INFO ][DEMO][10002] MQTT PUBLISH 1 successfully sent.
34 10009 [iot_thread] [INFO ][DEMO][10009] MQTT PUBLISH 8 successfully sent.
35 10016 [iot_thread] [INFO ][DEMO][10016] MQTT PUBLISH 7 successfully sent.
36 10023 [iot_thread] [INFO ][DEMO][10023] MQTT PUBLISH 9 successfully sent.
.37 10033 [iot_thread] [INFO ][DEMO][10033] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/1
Publish topi38 10044 [iot_thread] [INFO ][DEMO][10044] Acknowledgment message for PUBLISH 4 will be sent.
39 10056 [iot_thread] [INFO ][DEMO][10056] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/2
Publish topi40 10067 [iot_thread] [INFO ][DEMO][10067] Acknowledgment message for PUBLISH 1 will be sent.
41 10079 [iot_thread] [INFO ][DEMO][10079] Incoming PUBLISH received:
```

Figure 3.4.6-2: MQTT Demo Application logs

- The logs for the Shadow Demo application are shown below:



```
.10 8789 [WiFiMainThread] [WiFi] dns ip: 192.168.43.1
11 8789 [WiFiMainThread] [WiFi] Network ready IP: 192.168.43.165
12 8789 [WiFiMainThread] notify_events
13 9070 [Tmr Svc] WiFi Connected to AP Red.
14 9073 [Tmr Svc] IP Address acquired 192.168.43.165
15 9078 [iot_thread] [INFO ][DEMO][9078] ---------STARTING DEMO---------

16 9085 [iot_thread] subscribe events
17 9088 [WiFiMainThread] API !ALLOWED
18 9088 [WiFiMainThread]
WIFI STATE : STA

19 9096 [WiFiMainThread] WIFI_CONNECT : 2
20 9096 [WiFiMainThread]
WIFI STATE : STA

21 9353 [iot_thread] [INFO ][DEMO][9353] Successfully initialized the demo. Network type for the demo: 1
22 9362 [iot_thread] [INFO ][DEMO][9362] Shadow Thing Name is demo_thing (length 10).
....23 17409 [iot_thread] [INFO ][DEMO][17409] Successfully cleared Shadow of demo_thing.
24 17416 [iot_thread] [INFO ][DEMO][17416] Sending Shadow update 1 of 20: {"state":{"desired":{"powerOn":1}},"clientToken":"01
7.25 18960 [iot_thread] [INFO ][DEMO][18960] demo_thing changing state from 0 to 1.
26 18967 [iot_thread] [INFO ][DEMO][18967] demo_thing sent new state report.
27 18978 [iot_thread] [WARN ][DEMO][18978] Previous state not found in Shadow updated document.
28 18988 [iot_thread] [INFO ][DEMO][18988] Successfully sent Shadow update 1 of 20.
29 19880 [iot_thread] [INFO ][DEMO][19880] Shadow was updated!
Previous: {"state":{"desired":{"powerOn":1}}}
Current:  {"stat.30 21995 [iot_thread] [INFO ][DEMO][21995] Sending Shadow update 2 of 20: {"state":{"desired":{"powerOn":0}},
"clientToken":"021.31 22446 [iot_thread] [INFO ][DEMO][22446] Shadow was updated!
Previous: {"state":{"desired":{"powerOn":1},"reported":{"power032 22459 [iot_thread] [INFO ][DEMO][22459] demo_thing changing
state from 1 to 0.
33 22466 [iot_thread] [INFO ][DEMO][22466] demo_thing sent new state report.
34 22477 [iot_thread] [INFO ][DEMO][22477] Successfully sent Shadow update 2 of 20.
35 23496 [iot_thread] [INFO ][DEMO][23496] Shadow was updated!
Previous: {"state":{"desired":{"powerOn":0},"reported":{"power0.36 25484 [iot_thread] [INFO ][DEMO][25484] Sending Shadow upda
te 3 of 20: {"state":{"desired":{"powerOn":1}},"clientToken":"02537 26006 [iot_thread] [INFO ][DEMO][26006] Shadow was updated
!
Previous: {"state":{"desired":{"powerOn":0},"reported":{"power0.38 26019 [iot_thread] [INFO ][DEMO][26019] Successfully sent S
hadow update 3 of 20.
39 26027 [iot_thread] [INFO ][DEMO][26026] demo_thing changing state from 0 to 1.
40 26034 [iot_thread] [INFO ][DEMO][26034] demo_thing sent new state report.
41 27015 [iot_thread] [INFO ][DEMO][27015] Shadow was updated!
Previous: {"state":{"desired":{"powerOn":1},"reported":{"power0.42 29041 [iot_thread] [INFO ][DEMO][29041] Sending Shadow upda
te 4 of 20: {"state":{"desired":{"powerOn":0}},"clientToken":"02943 29486 [iot_thread] [INFO ][DEMO][29486] Successfully sent
Shadow update 4 of 20.
44 29494 [iot_thread] [INFO ][DEMO][29494] Shadow was updated!
Previous: {"state":{"desired":{"powerOn":1},"reported":{"power045 29507 [iot_thread] [INFO ][DEMO][29507] demo_thing changing
state from 1 to 0.
46 29514 [iot_thread] [INFO ][DEMO][29514] demo_thing sent new state report.
.47 30113 [iot_thread] [INFO ][DEMO][30113] Shadow was updated!
Previous: {"state":{"desired":{"powerOn":0},"reported":{"power0.48 32521 [iot_thread] [INFO ][DEMO][32521] Sending Shadow upda
te 5 of 20: {"state":{"desired":{"powerOn":1}},"clientToken":"03249 33011 [iot_thread] [INFO ][DEMO][33011] demo_thing changin
g state from 0 to 1.
```

Figure 3.4.6-3: Shadow Demo Application logs

3.4.7 Writing Custom Application

- The application execution starts at **main()**
- Enable the macro **configUSE_DAEMON_TASK_STARTUP_HOOK** in **FreeRTOSConfig.h** to start the *task startup* daemon and hook the custom tasks within *vApplicationDaemonTaskStartupHook().*
- New files can be added to **application_code** group in the project folder structure or new groups can be created.

3.4.8 Generating customized ENV

- The ENV_tool (Part of software SDK) contains the tools to modify and generate the env.bin.
- The values of ENV can be modified and new env can be added to **envitem.xml**.

  Refer "*LibreWirelessTechNote - MAVID_NV_Items_V0.4*" document for more details.
- Go to *FREERTOS_BASE_DIR\vendors\libre\mavid\tools\ENV_tool\* to generate new env.bin.

After editing the **envitem.xml**, run **CreateENV.bat.** The output **env.bin** and **env_uid.h** are generated in the same folder.

- In case a new ENV is added to **envitem.xml**, replace the **env_uid.h** at **FREERTOS\vendors\libre\mavid\components\Inc** with the newly generated **env_uid.h**.
- Load the **env.bin** on MAVID as explained in *Loading ENV binary* of the section 3.4.6.Loading and Debugging Demo Application.

4. LED Indication Status

The MAVID EVB board has the following System and AVS LED as mentioned below:

4.1 System LED

State	LED Indication
Power On	RED Solid
Power OFF	No Indication

5. Troubleshooting

For general troubleshooting information about Getting Started with FreeRTOS, see Troubleshooting Getting Started.