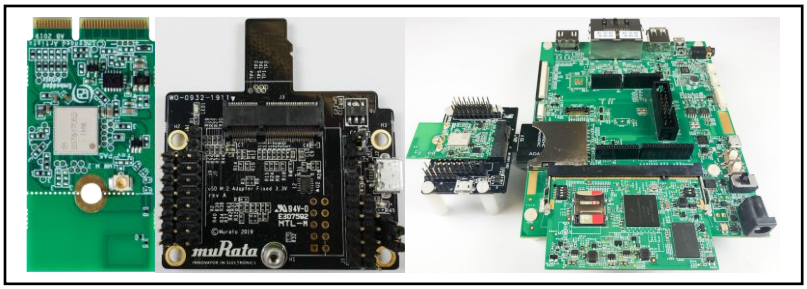
Murata Wi-Fi/BT Solution for i.MX

Quick Start Guide (Linux)



Revision History

Revision	Date	Author	Change Description
1.0	Sept 1, 2015	S Kerr G Mohiuddin	Initial Release
1.1	Sept 6, 2015	S Kerr	Removed software compile/build dependency. User can bring up NXP platform by downloading necessary files before flashing bootable SD card. Refer to Linux User Guide on software build procedures.
2.0	Nov 7, 2015	S Kerr	Modified Murata Wi-Fi/BT EVK definition. This simplifies bring-up on NXP i.MX6 Platforms. Incorporated changes for i.MX6UL 3.14.38 GA BSP Release.
4.0	Feb 14, 2017	S Kerr	Renamed document to “Murata Wi-Fi/BT Solution for i.MX Quick Start Guide (Linux)”. Incorporated changes for NXP Linux 4.1.15_2.0.0 GA BSP release. Modified NXP Linux 3.14.52_1.1.0 GA BSP release to build in bcmdhd WLAN driver, thereby matching 4.1.15_2.0.0 configuration. Added instructions for Murata binary patch release which addresses errata/features on both releases. Provided support for new i.MX 7Dual SDB, i.MX 6ULL EVK, and Murata Type 1CK. Expanded WLAN test verification section.
5.0	Feb 26, 2018	S Kerr J Kareem	Complete revision for integrating new Cypress FMAC driver release. Added support for SDIO/UART 1.8V VIO signaling on i.MX6UL(L) and i.MX6SX platforms. Rollout of Murata customized Yocto build for i.MX BSP’s. Add support for new i.MX 8MQuad EVK.
6.0	May 29, 2019	S Kerr B Chen P Sah	Added support for Murata's uSD-M.2 Adapter, Embedded Artists’ M.2 Wi-Fi/BT modules, and i.MX Linux Kernel versions 4.9.88/4.9.123. Streamlined support for (currently) following Murata modules: 1DX, 1MW, 1LV, and 1CX.

This page intentionally left blank.

Table of Contents

REVISION HISTORY **1**

TABLE OF CONTENTS **3**

1 INTRODUCTION **5**

1.1 NXP i.MX 6 Platform Support 6
1.2 NXP i.MX 8 Platform Support 7
1.3 Acronyms 8
1.4 References 9
1.4.1 Murata Linux User Manual 9
1.4.2 Murata Hardware User Manual 9
1.4.3 Murata uSD-M.2 Adapter Documentation 9
1.4.4 Embedded Artists' Reference Documentation 10
1.4.5 NXP Reference Documentation 10

2 WI-FI/BT HARDWARE SOLUTION FOR I.MX **11**

2.1 Embedded Artists' Wi-Fi/BT M.2 EVB's 11
2.2 Murata's uSD-M.2 Adapter 12
2.3 NXP i.MX versus Murata Module InterConnect 14

3 WI-FI/BT SOFTWARE SOLUTION FOR I.MX **15**

3.1 "fmac" Solution Overview 15
3.2 Specific i.MX Target Support Details 15
3.3 Murata "fmac" Customized i.MX Yocto Image Build 17
3.3.1 Install Ubuntu 17
3.3.2 Download Murata's Script Files 17
3.3.3 Configure Ubuntu for i.MX Yocto Build 18
3.3.4 Murata's i.MX Yocto Build Script 19
3.4 Additional Hardware/Software Considerations 21
3.4.1 Out-Of-Band (OOB) IRQ Support on NXP i.MX EVK's 21
3.4.2 UHS SDIO 3.0 operation on i.MX Platforms with uSD-M.2 Adapter 21

4 PREPARING BOOTABLE SD CARD FOR I.MX WITH MURATA WI-FI/BT EVK **22**

4.1 Linux PC Steps to Flash SD Card 22
4.2 Windows PC Steps to Flash SD Card 23

5 MURATA WI-FI/BT BRING-UP ON I.MX 6 PLATFORMS **24**

5.1 Connecting to i.MX 6SoloX SDB 25
5.2 Connecting to i.MX 6SoloLite EVK 26
5.3 Connecting to i.MX 6Q/DL SDB/SDP or i.MX 6 QP SDB 27
5.3.1 Specific Hardware Considerations for i.MX 6Quad/DualLite SDB/SDP 27
5.3.2 Specific Hardware Considerations for i.MX 6QP SDB 27
5.3.3 Wi-Fi/Bluetooth Bring-Up on i.MX 6Q/DL SDB/SDP or i.MX 6 QP SDB 27
5.4 Connecting to i.MX 6UltraLite EVK or i.MX 6ULL EVK 29

6 MURATA WI-FI/BT BRING-UP ON I.MX 8 PLATFORMS **30**

6.1 Bringing up Wi-Fi/BT on i.MX 8MQuad EVK 30
6.2 Bringing up Wi-Fi/BT on i.MX 8M Mini EVK 31

7 TEST/VERIFICATION OF WI-FI AND BLUETOOTH **32**

7.1 Wi-Fi Interface Test/Verification 33
7.1.1 Useful Environment Setup on NXP Linux 33
7.1.2 Bringing Up Wi-Fi Interface 33
7.1.3 STA/Client Mode: Scan for Visible Access Points 34
7.1.4 STA/Client Mode: Connecting to Unsecured Access Point or Wireless Router 38
7.1.5 STA/Client Mode: Connecting to Secured Access Point or Wireless Router (WPA2-PSK) 40
7.1.6 STA/Client Mode: Basic WLAN Connectivity Testing 43

7.1.7 Wi-Fi Direct Testing 44
7.1.8 Soft AP or Wi-Fi Hot Spot Testing 46
7.1.9 WLAN Manufacturing or RF Testing 47
7.2 Bluetooth Interface Test/Verification 49
8 USD-M.2 ADAPTER HIGH-LEVEL DESCRIPTION 51
9 EMBEDDED ARTISTS' WI-FI/BT M.2 MODULES 54
10 TECHNICAL SUPPORT CONTACT 54
11 ADDITIONAL USEFUL LINKS 55

LIST OF TABLES

Table 1: Acronyms used in Quick Start Guide 8
Table 2: Documents for uSD-M.2 Adapter 9
Table 3: Embedded Artists Documentation Listing 10
Table 4: NXP Reference Documentation Listing 11
Table 5: Embedded Artists' Wi-Fi/BT M.2 Modules Supported 12
Table 6: NXP i.MX/Murata Module InterConnect 14
Table 7: NXP i.MX EVK / Yocto (MACHINE) target / Kernel Version Matrix 16
Table 8: i.MX6/8 Targets supported by Murata 16
Table 9: Embedded Wi-Fi/Bluetooth Files 32
Table 10: GPIO and UART Settings for Bluetooth Tests 49
Table 11: uSD-M.2 Adapter Features 51
Table 12: List of Support Resources 54
Table 13: Additional Useful Links 55

LIST OF FIGURES

Figure 1: i.MX 6 EVK Wi-Fi/BT Interconnect Block Diagram 6
Figure 2: i.MX 8MQuad EVK Wi-Fi/BT Interconnect Block Diagram 7
Figure 3: i.MX 8M Mini EVK Wi-Fi/BT Interconnect Block Diagram 7
Figure 4: Murata uSD-M.2 Adapter Kit Contents (LBEE0ZZ1WE-TEMP) 13
Figure 5: USB to SD Card Reader/Writer Adapter 22
Figure 6: uSD-M.2 Adapter with type 1DX/1MW/1LV M.2 EVB options 24
Figure 7: i.MX 6SoloX SDB with uSD-M.2 Adapter and Type 1MW M.2 EVB 25
Figure 8: i.MX 6SoloLite EVK with uSD-M.2 Adapter and Type 1DX M.2 EVB 26
Figure 9: i.MX 6Quad/DualLite SDB (Inverted) with uSD-M.2 Adapter and Type 1MW M.2 EVB 28
Figure 10: i.MX 6UltraLite EVK with uSD-M.2 Adapter and Type 1LV M.2 EVB 29
Figure 11: i.MX 8MQuad with Type 1CX (bottom view) 30
Figure 12: i.MX 8M Mini with Type 1CX (bottom view) 31
Figure 13: uSD-M.2 Adapter Features (Top View) 52
Figure 14: uSD-M.2 Adapter Features (Bottom View) 53

1 Introduction

Murata has partnered with NXP Semiconductors N.V., Cypress Semiconductor Corporation, and Embedded Artists AB to offer a complete Wi-Fi and Bluetooth connectivity environment for building world class Internet-connected products on NXP's family of i.MX Processors. The Murata Connectivity Modules enable developers to minimize the development time and effort for connectivity function implementation. This document details enabling Murata Wi-Fi/BT solutions on reference i.MX platforms. Current Linux i.MX releases supported include NXP's L4.1.15_2.0.0_BSP, L4.9.11_1.0.0_BSP, L4.9.88_2.0.0_BSP, and L4.9.123_2.3.0_8MMini_GA. Following steps are described:

- How to build SD card image (using Murata customized script file) which enables Cypress "***fmac***" WLAN driver (while disabling the legacy "bcmdhd" driver in specific NXP i.MX Linux releases). This "***fmac***" driver and associated components (WPA supplicant, Hostapd, WLAN firmware and NVRAM files, Bluetooth patchfiles, etc.) are currently not enabled in the i.MX Linux Yocto images prior to kernel version 4.14.62.
- As part of the build process, the user may configure customized i.MX image to support 1.8V VIO signaling on the 6UL(L) reference platforms (also required to interface to Type 1LV EVB).
- Connect/configure Murata/Embedded Artists M.2-based Wi-Fi/BT EVK to i.MX Reference platform and power up.
- Initialize/configure WLAN and Bluetooth interfaces.
- Exercise WLAN and Bluetooth functionality.

The NXP Platforms currently supported are based on i.MX 8 and i.MX 6 with no Cypress-based Murata modules soldered down. The wireless solution for the following platforms is either just the Embedded Artists' Wi-Fi/BT M.2 EVB (i.MX 8), or a combined solution with Murata's uSD-M.2 Adapter and Embedded Artists' Wi-Fi/BT M.2 EVB:

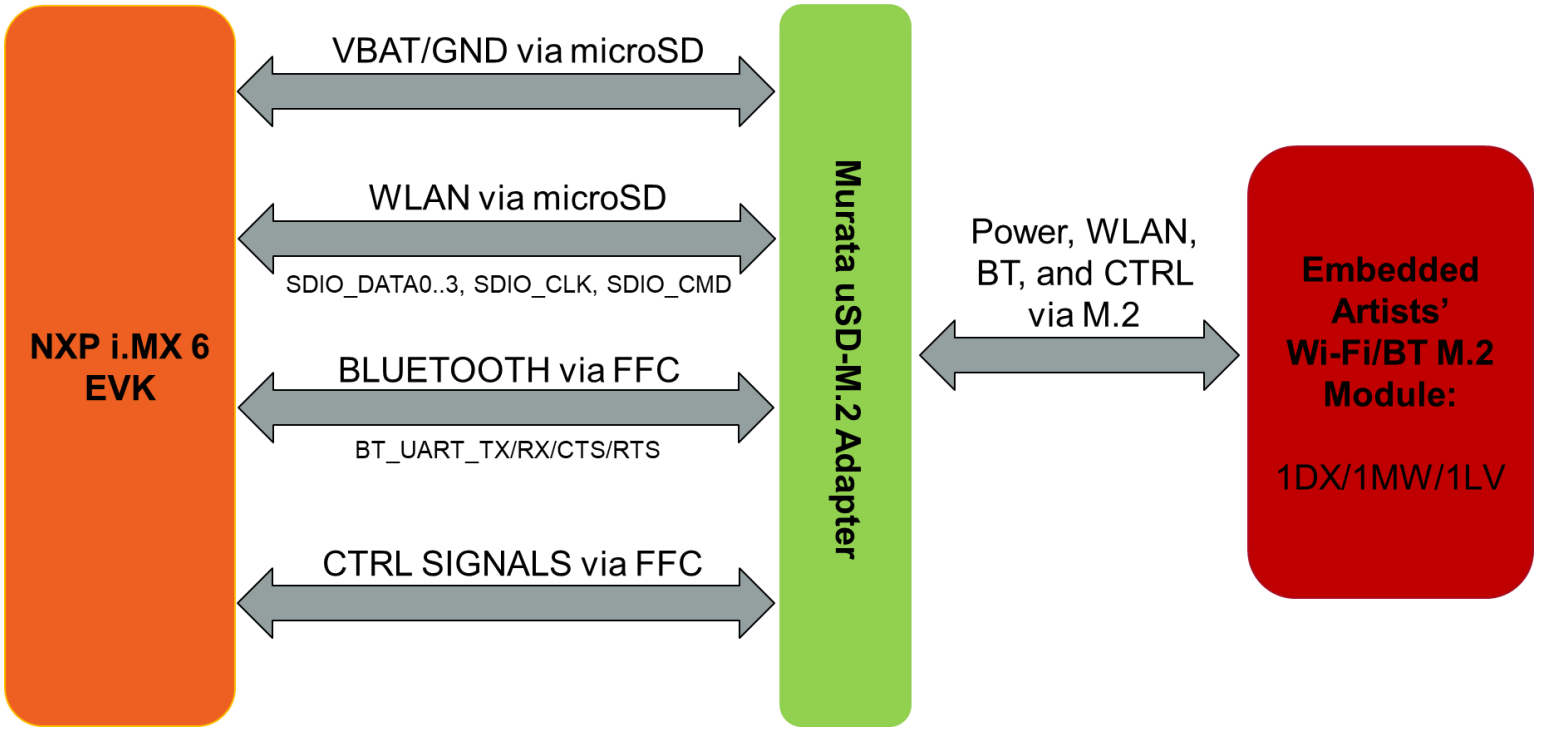
- i.MX 8MQuad Evaluation Kit
- i.MX 8M Mini Evaluation Kit
- i.MX 6QuadPlus SABRE Development Board
- i.MX 6Quad/DualLite SABRE Development Board
- i.MX 6SoloX SABRE Development Board
- i.MX 6SoloLite Evaluation Kit
- i.MX 6SLL Evaluation Kit
- i.MX 6UltraLite (or i.MX 6UL) Evaluation Kit
- i.MX 6ULL Evaluation Kit

1.1 NXP i.MX 6 Platform Support

A high-level connection Diagram for the Murata Interconnect kit (i.MX 6 platforms) is provided in **.** All the Murata Wi-Fi/BT modules enabled by this release are shown. The wireless solution is arrived at by combining Murata's uSD-M.2 Adapter with Embedded Artists' Wi-Fi/BT M.2 EVB. Refer to **Sections 8** and **9** for more details on the uSD-M.2 Adapter and Wi-Fi/BT M.2 Modules. It is important to note that Murata no longer promotes the legacy i.MX V1/V2 InterConnect Kit which used 60-pin Samtec connectors. Murata has collaborated very closely with Embedded Artists to arrive at the new Wi-Fi/BT M.2 EVB (Module) solution. As evident from the Embedded Artists' documentation, the M.2 EVB is optimized for evaluation with the following features:

- PCI Express M.2 (key "E") compliant – Industry standard.
  Comprehensive interface support for WLAN SDIO/PCIe, Bluetooth UART/PCM/I2S, WLAN-Bluetooth coexistence, all necessary WLAN/Bluetooth control signals, and additional WLAN/Bluetooth debug signals.
- Relatively low-cost form factor.
- Easy for customers to run prototype builds with Wi-Fi/BT M.2 Modules.
- Can also be used in lower-volume production runs (i.e. <10K). Contact Embedded Artists for higher volume pricing (i.e. 100, 500, 1000, and more).
- Reference certified PCB trace antenna.
- Snap-off option for customers needing to adhere to MAX 30mm length (U.FL connector used).
- U.FL connector for external antenna or conducted testing.
- Comprehensive test points (including SDIO DATA, CLK, and CMD lines).

Figure 1: i.MX 6 EVK Wi-Fi/BT Interconnect Block Diagram



1.2 NXP i.MX 8 Platform Support

Figure 2 shows a simplified block diagram for the i.MX 8MQuad EVK Wi-Fi/BT interconnect, Currently Type 1CX is supported with 2x2 802.11ac MIMO and WLAN-PCIe interface. Note that **no** Adapter is used for i.MX 8MQuad EVK – just the Wi-Fi/BT M.2 EVB (Module).

Figure 2: i.MX 8MQuad EVK Wi-Fi/BT Interconnect Block Diagram

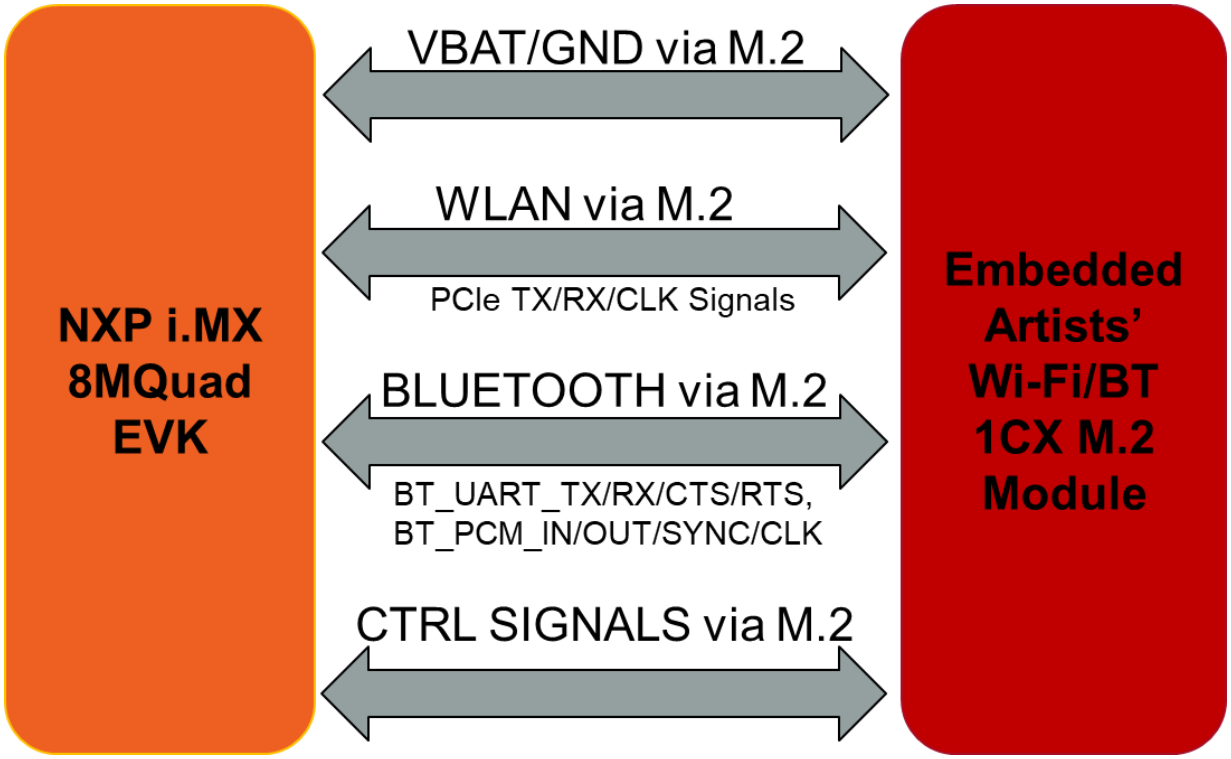
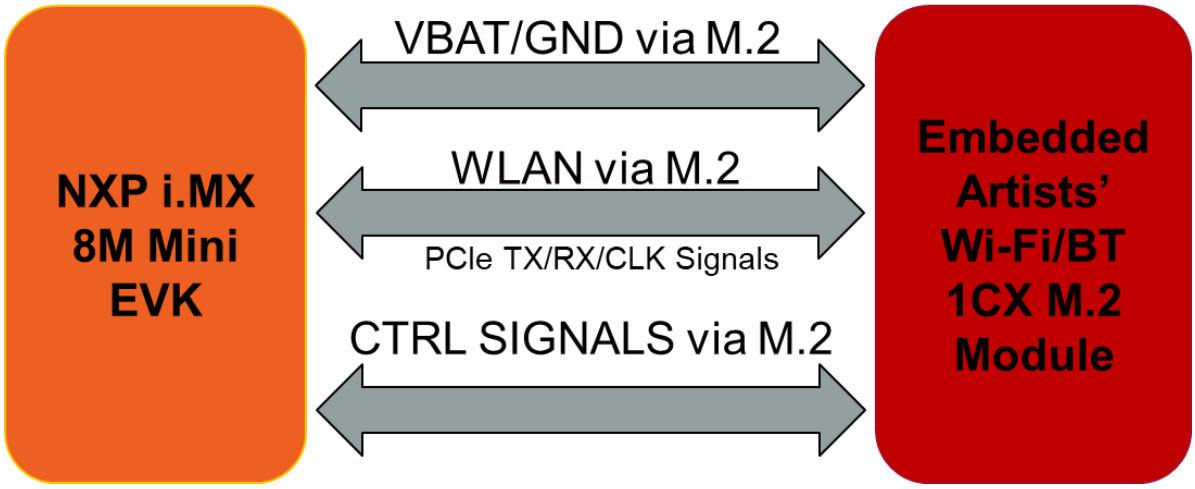


Figure 3 shows a simplified block diagram for the i.MX 8M Mini EVK Wi-Fi/BT interconnect, Currently Type 1CX module (WLAN Only) is supported with 2x2 802.11ac MIMO and WLAN-PCIe interface. Note that **no** Adapter is used for i.MX 8MQuad EVK – just the Wi-Fi/BT M.2 EVB (Module).

Figure 3: i.MX 8M Mini EVK Wi-Fi/BT Interconnect Block Diagram



1.3 Acronyms

Table 1: Acronyms used in Quick Start Guide

Acronym	Meaning
AP	Access Point
API	Application Programming Interface
BSP	Board Support Package
BT	Bluetooth
CTRL	Control
CTS	Clear to Send
DHCP	Dynamic Host Configuration Protocol
DTB	Device Tree Blob: Kernel reads in at boot time for configuration.
EA	Embedded Artists designs, manufactures and distributes current Wi-Fi/BT M.2 EVB's. Refer to this link for more information.
EULA	End User License Agreement
EVB	Evaluation Board (Embedded Artists' Wi-Fi/BT module)
EVK	Evaluation Kit (includes EVB + Adapter)
FFC	Flat Flex Cable
FW	Firmware
GPIO	General Purpose Input/Output
IRQ	Interrupt Request Line
MIMO	Multiple Input Multiple Output
NVRAM	Non-Volatile Random-Access Memory
OOB	Out of Band
O/S	Operation System
PC	Personal Computer
PCIe	PCI Express
PCM	Pulse Code Modulation
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
RTS	Request to Send
SABRE	Smart Application Blueprint for Rapid Engineering
SDIO	Secure Digital Input Output
STA	Station
SW	Software
UART	Universal Asynchronous Receiver/Transmitter
UHS	Ultra-High Speed
USB	Universal Serial Bus
uSD	Micro SD
uSD-M.2	Micro SD to M.2 Adapter
VBAT	Voltage of the Battery
VIO	Input Offset Voltage
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access

1.4 References

This section reviews all the key reference documents that the user may like to refer to. Note that the references also include Embedded Artists and NXP links.

1.4.1 Murata Linux User Manual

Murata Wi-Fi/BT Solution for i.MX Linux User Manual 6.0, "Murata Wi-Fi & BT Solution for i.MX Linux User Manual 6.0.pdf".

This manual describes all steps necessary to build the file system, kernel, DTB files, and WLAN "***fmac***" driver necessary for supporting NXP i.MX Platforms and the Murata Wi-Fi/BT EVK. This manual is a key document given the hard requirement for users to build the i.MX image and flash a (micro) SD card prior to booting the i.MX platform. The Murata Linux User Manual is available on the main Murata i.MX Support site (landing page): https://wireless.murata.com/imx.

1.4.2 Murata Hardware User Manual

Murata Wi-Fi/BT Solution for i.MX Hardware User Manual 3.0, "Murata Wi-Fi & BT Solution for i.MX Hardware User Manual 3.0.pdf".

This manual describes the hardware interconnect for the Murata Wi-Fi/BT solution on all i.MX Platforms. These platforms include i.MX RT, 6, 7[1], and 8. Specifics on interfacing each i.MX Platform to Murata Wi-Fi/BT EVK are provided. The Murata Hardware User Manual is available on the main Murata i.MX Support site (landing page): https://wireless.murata.com/imx.

1.4.3 Murata uSD-M.2 Adapter Documentation

The Murata uSD-M.2 Adapter interfaces Embedded Artists' Wi-Fi/BT M.2 EVB's to NXP's i.MX 6 Platforms. **Table 2** provides the hyperlink of two documents for this adapter. Murata landing page for uSD-M.2 Adapter is here: https://wireless.murata.com/usd-m2. Note that this adapter can be potentially used on any platform that supports 4-bit WLAN SDIO via a microSD or SD connector. The additional Bluetooth and control signals are available via 20-pin FFC connector or Arduino headers.

Table 2: Documents for uSD-M.2 Adapter

Documentation Filename	Note
uSD-M.2 Adapter Product Brief	Brief introduction of uSD-M.2
uSD-M.2 Adapter Datasheet	Datasheet of uSD-M.2 Adapter

[1] The NXP i.MX 7Dual SDB is not covered in this document – because the Murata module (Type ZP) on this platform is no longer promoted. Customers are advised to use Type 1MW instead. Embedded Artists sells an i.MX 7Dual Developer's Kit which supports all (currently promoted) Murata modules.

1.4.4 Embedded Artists' Reference Documentation

Embedded Artists designed the 1DX/1MW/1LV/1CX M.2 EVB's in close collaboration with Murata. It is ***important to note*** that Embedded Artists manufactures and distributes the Wi-Fi/BT M.2 EVB's. Refer to this main landing page for more information: www.embeddedartists.com/m2. **Table 3** lists some relevant documents published by Embedded Artists.

Table 3: Embedded Artists Documentation Listing

Documentation Filename	Note
Wi-Fi/BT M.2 EVB Primer	Introduction and drill-down on M.2 interface
M.2 SDIO Interface Schematic	Reference schematic for customers designing in WLAN-SDIO M.2 EVB.
M.2 PCIe Interface Schematic	Reference schematic for customers designing in WLAN-PCIe M.2 EVB.
1DX M.2 Module Datasheet	Comprehensive details on 1DX Wi-Fi/BT M.2 Module.
1MW M.2 Module Datasheet	Comprehensive details on 1MW Wi-Fi/BT M.2 Module.
1LV M.2 Module Datasheet	Comprehensive details on 1LV Wi-Fi/BT M.2 Module.
1CX M.2 Module Datasheet	Comprehensive details on 1CX Wi-Fi/BT M.2 Module.

1.4.5 NXP Reference Documentation

Some of the key NXP reference documentation for Linux includes the following:

- Yocto Project User's Guide: This document describes how to build an image for an NXP i.MX platform by using a Yocto Project build environment. It describes the NXP release layer and the NXP-specific usage.
- i.MX Linux User's Guide: This document explains how to build and install the NXP Linux O/S BSP on the i.MX platform. It also covers special NXP features and how to use them.
- i.MX Linux Reference Manual: This document supports porting the i.MX Linux O/S BSP to customer-specific products. Intended audience should have a working knowledge of Linux O/S kernel internals, driver models and i.MX processors.
- i.MX Linux Release Notes: This document contains important information about the package contents, supported features, known issues, and limitations in the release.

Table 4 provides the following information on all releases supported:

- document filename
- document number
- document release revision (also indicates Linux kernel version)
- hyperlink to NXP document download

Table 4: NXP Reference Documentation Listing

Document Filename / Number	4.9.123_2.3.0	4.9.88_2.0.0	4.9.11_1.0.0	4.1.15_2.0.0
i.MX_Yocto_Project_User's_Guide.pdf / IMXLXYOCTOUG	Rev. L4.9.123_2.3.0_8MMini_GA	Rev. L4.9.88_2.0.0_BSP	Rev. L4.9.11_1.0.0_BSP	Rev. L4.1.15_2.0.0_BSP
i.MX_Linux_User's_Guide.pdf / IMXLUG				
i.MX_Linux_Reference_Manual.pdf / IMXLXRM				
i.MX_Linux_Release_Notes.pdf / IMXLXRN				

2 Wi-Fi/BT Hardware Solution for i.MX

As already outlined in the **Introduction,** the i.MX hardware solution is arrived at using Embedded Artists' Wi-Fi/BT M.2 EVB's in conjunction with Murata's uSD-M.2 Adapter[2]. This section provides additional details on the hardware solution.

2.1 Embedded Artists' Wi-Fi/BT M.2 EVB's

Embedded Artists designs, manufactures and distributes the Wi-Fi/BT M.2 EVB's based on Murata modules. These new M.2 EVB's are now Murata's ***official*** evaluation board for these modules in the Distribution Channel.

Embedded Artists has excellent documentation support with a main landing page at: https://www.embeddedartists.com/m2/. **Table 5** shows the details of Embedded Artists' Wi-Fi/BT M.2 Modules. Note that Type 1DX (CYW4343W), 1MW (CYW43455), and 1LV (CYW43012) support a WLAN-SDIO interface; whereas Type 1CX has a WLAN-PCIe interface. Also note that Type 1LV's interface voltage only supports 1.8V; whereas other EVB's interface voltage can be either 3.3V or 1.8V[3]. To learn specifics on any of the M.2 EVB's, just click on Embedded Artists' M.2 Module Part Number (hyperlink included in table). To learn more specifics on the Murata module, just click Murata part number and you will be redirected to Murata's module landing page.

[2] Murata uSD-M.2 Adapter is not used for NXP platforms with M.2 Slot – i.e. i.MX 8MQuad EVK or i.MX 8M Mini EVK. The Adapter is used for NXP i.MX 6 Platforms (also i.MX RT – see https://wireless.murata.com/usd-m2.html for more information).

[3] The M.2 standard specifies 1.8V VIO voltage for SDIO, UART, and PCM interfaces. The Embedded Artists' M.2 EVB incorporates a key design feature which allows the default 1.8V VIO to be overridden with 3.3V VIO if necessary (only on specific modules which support 3.3V). Given the default NXP i.MX voltage rails, 3.3V VIO is a requirement on most of the i.MX 6 platforms.

Table 5: Embedded Artists' Wi-Fi/BT M.2 Modules Supported

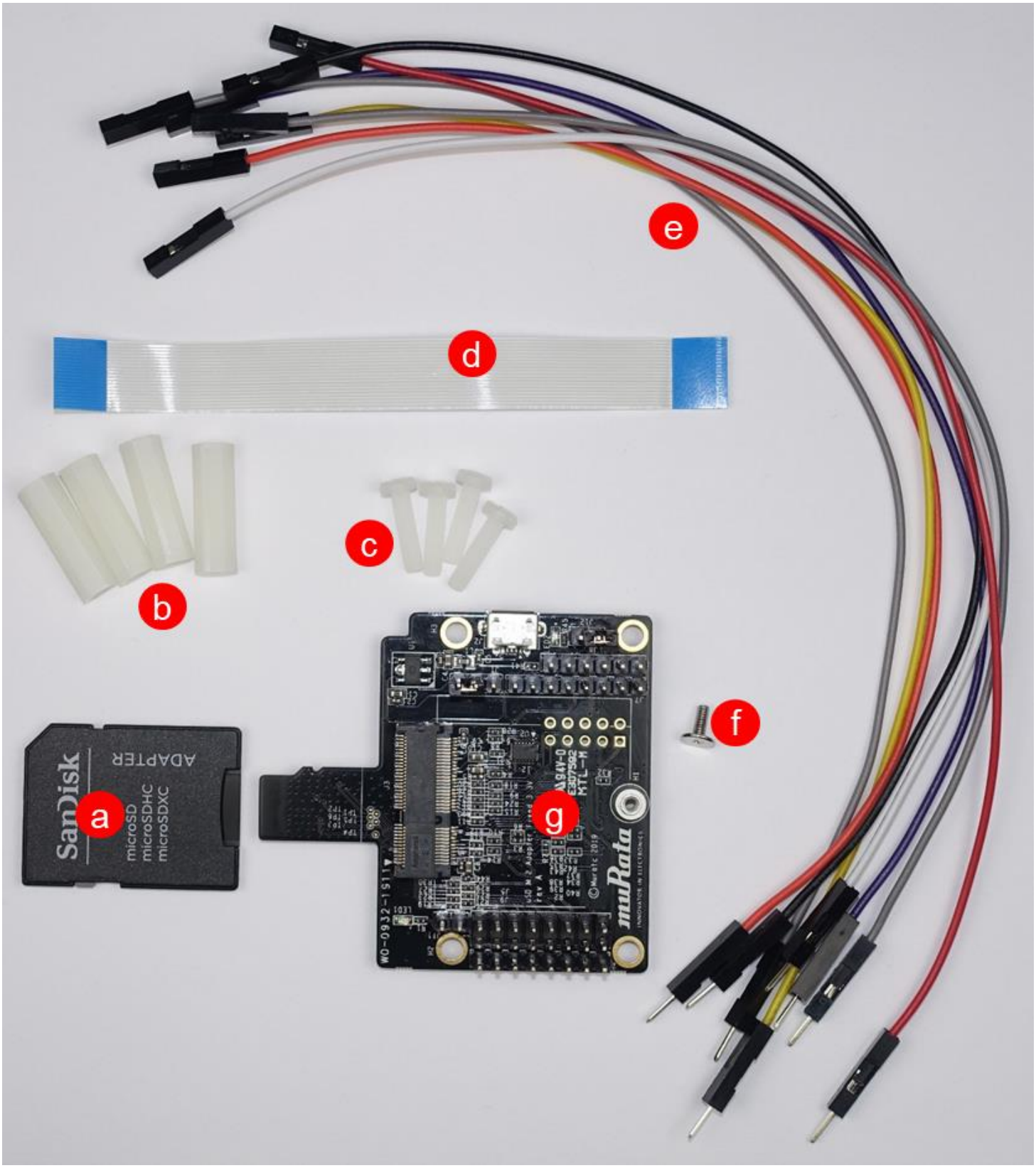
Murata Module	CYW Chipset	Wi-Fi/BT Support	Murata Part Number	EA M.2 Module Part Number	VIO	Interface	EVB Picture
1DX	4343W	802.11b/g/n BT/BLE 4.2	LBEE5KL1DX	EAR00318	1.8V/ 3.3V	WLAN: SDIO Bluetooth: UART	
1MW	43455	802.11a/b/g/n/ac (1x1 SISO) BT/BLE 5.0	LBEE5HY1MW	EAR00315	1.8V/ 3.3V	WLAN: SDIO Bluetooth: UART	
1LV	43012	802.11a/b/g/n (ac friendly) BT/BLE 5.0	LBEE59B1LV	EAR00323	1.8V only	WLAN: SDIO Bluetooth: UART	
1CX	4356	802.11a/b/g/n/ac (2x2 MIMO) BT/BLE 5.0	LBEH5UL1CX	EAR00321	1.8V/ 3.3V	WLAN: PCIe Bluetooth: UART	

2.2 Murata's uSD-M.2 Adapter

The Wi-Fi/BT solution for NXP i.MX 6 Platform requires the use of a Murata uSD-M.2 Adapter Kit in conjunction with Embedded Artists' Wi-Fi/BT M.2 EVB. **Figure 4** shows the contents of Murata's uSD-M2 Adapter Kit (Part Number **LBEE0ZZ1WE-TEMP**):

a) microSD to SD Card Adapter
b) 4 x 19mm M3 stand-offs (nylon)
c) M3 screws (nylon)
d) 75mm 20-pos, 0.5mm pitch flat/flex cable
e) 13 pieces 200mm long male-to-female jumper cables (compatible with Arduino header)
f) M.2 screw for attaching Wi-Fi/Bluetooth M.2 Evaluation Board (EVB)
g) uSD-M.2 Adapter (Revision A)

Figure 4: Murata uSD-M.2 Adapter Kit Contents (LBEE0ZZ1WE-TEMP)



For more information on the uSD-M.2 Adapter, refer to **Section 8** or go to the Adapter landing page at: https://wireless.murata.com/usd-m2.html.

2.3 NXP i.MX versus Murata Module InterConnect

Table 6 provides an i.MX Reference Platform versus Murata module matrix. An additional column is included to provide a quick Cypress chipset lookup. For a given i.MX platform and Murata module, you can quickly look up the compatibility. Providing more details on the terminology used in the table:

- "**NC**" means not compatible. This is due to one or both of the following reasons:
  - VIO incompatible: module requires VIO voltage level that the i.MX HW cannot provide.
  - Bus (SDIO, PCIe, UART) interconnect not available.
- "**uSD-M.2-3.3V**" means that the configuration **is** supported. uSD-M.2 Adapter is needed with EA's M.2 EVB. WLAN SDIO, BT UART, WLAN/BT control signals are configured for 3.3V VIO on both i.MX EVK and Adapter/M.2.
- "**uSD-M.2-1.8V**" means that the configuration **is** supported. uSD-M.2 Adapter is needed with EA's M.2 EVB. WLAN SDIO, BT UART, WLAN/BT control signals are configured for 1.8V VIO on both i.MX EVK and Adapter/M.2. Note that due to i.MX EVK limitations, there may be some mixed 3.3V signaling on the WLAN/BT control lines and BT UART.
- "**M.2**" for i.MX8MQuad means that the configuration is supported. The EVK has M.2 connector. No uSD-M.2 Adapter required. However, the M.2 slot only supports WLAN PCIe interface and Bluetooth UART. Currently only Type 1CX M.2 EVB works in this configuration.
- "***M.2**" for i.MX 8M Mini means that the configuration is supported. The EVK has M.2 connector. No uSD-M.2 Adapter required. However, the M.2 slot only supports WLAN PCIe interface. Currently only Type 1CX M.2 EVB works in this configuration. Since no UART is connected to the M.2 slot, only WLAN works with no Bluetooth support.

NOTE: When using uSD-M2 adapter, the maximum SDIO clock frequency is only 50MHz for both 1.8V and 3.3V VIO. For UHS mode support (i.e. MAX SDIO clock is 200 MHz for Type 1MW) and for comprehensive signal support, ***Murata recommends*** the Embedded Artists' i.MX Developer Kits.

Table 6: NXP i.MX/Murata Module InterConnect

Murata Module	CYW Chipset	6UL(L)	6SL(L)	6SX	6Q(P)/DL	8MQuad	8M Mini
1DX	4343W	uSD-M.2-3.3V	uSD-M.2-3.3V	uSD-M.2-3.3V	uSD-M.2-3.3V	NC	NC
1MW	43455	uSD-M.2-3.3V	uSD-M.2-3.3V	uSD-M.2-3.3V	uSD-M.2-3.3V	NC	NC
1LV	43012	uSD-M.2-1.8V	NC	NC	NC	NC	NC
1CX	4356	NC	NC	NC	NC	M.2	*M.2

uSD-M.2-3.3V = work with uSD-M.2 Adapter on 3.3V (J12 is closed)
uSD-M.2-1.8V = work with uSD-M.2 Adapter on 1.8V (J12 is open)
M.2 = work with onboard M.2 slot
*M.2 = work with onboard M.2 slot, but only WLAN works

3 Wi-Fi/BT Software Solution for i.MX

3.1 "fmac" Solution Overview

The default NXP's Linux 4.1.15_2.0.0, Linux 4.9.11_1.0.0, Linux 4.9.88_2.0.0, and Linux 4.9.123 for i.MX 8MMini GA BSP integrate the legacy Cypress WLAN ***"bcmdhd"*** driver and have limited Bluetooth support. The new implementation of Cypress' WLAN driver is referred to as ***"fmac"***. Please note that there is a distinct difference between the ***"brcmfmac"*** open source community drivers integrated into kernel.org Linux releases. The ***"fmac"*** driver (as customized by Murata) is an official open source release from Cypress that is tested and verified. The ***"fmac"*** release leverages the Linux Backports implementation to integrate the WLAN driver into the desired Linux kernel version.

Murata's customized Yocto layer ***"meta-murata-wireless***" seamlessly disables the existing ***"bcmdhd"*** WLAN driver and pulls in the ***"fmac"*** (officially supported) driver implementation. More specifically it provides the following enhancements/customizations:

- Pull Cypress ***"fmac"*** driver and run backports tool during Yocto build to generate necessary driver modules.
- Additional/necessary patches to Cypress ***"fmac"*** driver for i.MX implementation.
- i.MX Linux kernel customizations to support ***"fmac"*** driver with OOB IRQ interrupts.
- Support 1.8V VIO signaling with NXP i.MX6UL(L) EVK – necessary for Type 1LV.
- WLAN production firmware files. For manufacturing test firmware (necessary for RF/regulatory testing), please contact Murata directly. General email link is wirelessFAQ@murata.com.
- Murata NVRAM files for correctly configuring WLAN RF.
- Example Bluetooth patch files.
- WL tool binary necessary for interoperability and RF testing.
- Hostapd (Version 2.6) configuration with specific patch release.
- Hostap-conf enablement.
- Hostap-utils enablement.
- WPA-supplicant (Version 2.6) configuration with specific patch release.
- Wi-Fi Direct (P2P) enablement.

There are four versions of ***"fmac"*** currently supported: ***"v4.12 orga"***, ***"v4.14 battra"***, ***"v4.14 mothra", and***, ***"v4.14 manda"***. ***"orga", "battra", "mothra", and "manda"*** are the Cypress codenames denoting ***"fmac"*** release version. ***"v4.12"*** and ***"v4.14"*** are the latest kernel versions supported by either release (either ***"fmac"*** release can be backported to kernel version 3.0). To abbreviate references to specific versions of ***"fmac"***, Murata uses ***just*** the Cypress codename – i.e. ***"orga"***, ***"battra", "mothra" or "manda"***. It is strongly recommended to use the latest ***"fmac"*** release version: currently this is ***"manda"***.

3.2 Specific i.MX Target Support Details

Murata's customized Yocto layer supports the following NXP i.MX EVK's as outlined in **Table 7**. ***"MACHINE=target"*** is a direct reference to Yocto build. The ***"target"*** string is the keyword used to select hardware configuration for the build (important knowledge when running Murata build script). With the newer EVK's (i.MX 8MQuad and i.MX8M Mini) only certain kernel versions are supported.

From this table, you can find a proper version of Linux Kernel for your target platform. You can then refer to **Table 8** for the support of interrupt configuration, corresponding DTB (Device Tree Blob) files, hardware interconnect configuration (either uSD-M.2 Adapter & M.2 EVB, or just M.2 EVB), and the support of SDIO signaling 1.8V VIO option. **NOTE:** please refer to **Section 3.4.1** and/or the Hardware User Manual to ensure that i.MX EVK hardware configuration supports OOB IRQ signaling if that is your desired WLAN interrupt mode.

Table 7: NXP i.MX EVK / Yocto (MACHINE) target / Kernel Version Matrix

i.MX EVK	MACHINE=target	Linux Kernel 4.1.15	Linux Kernel 4.9.11	Linux Kernel 4.9.88	Linux Kernel 4.9.123
i.MX 8MMini EVK	imx8mmevk	N	N	N	Y
i.MX 8MQuad EVK	imx8mqevk	N	N	Y	Y
i.MX 6QuadPlus SDB	imx6qpsabresd	Y	Y	Y	Y
i.MX 6Quad SDB	imx6qsabresd	Y	Y	Y	Y
i.MX 6DualLite SDB	imx6dlsabresd	Y	Y	Y	Y
i.MX 6SX SDB	imx6sxsabresd	Y	Y	Y	Y
i.MX 6SL EVK	imx6slevk	Y	Y	Y	N
i.MX 6UL EVK (14x14, 9x9)	imx6ulevk	Y	Y	Y	Y
i.MX 6ULL EVK (14x14)	imx6ull14x14evk	Y	Y	Y	Y
i.MX 6ULL EVK (9x9)	imx6ull9x9evk	Y	Y	Y	Y

Table 8: i.MX6/8 Targets supported by Murata

Target (MACHINE)	SDIO Interrupt Configuration	NXP i.MX DTB File	Hardware Config	1.8V VIO SDIO
imx8mmevk	N/A	fsl-imx8mm-evk.dtb	M.2	N/A
imx8mqevk	N/A	fsl-imx8mq-evk-pcie1-m2.dtb	M.2	N/A
imx6qpsabresd	OOB IRQ	imx6qp-sabresd-btwifi-m2-oob.dtb	uSD-M.2	No
imx6qpsabresd	SDIO in-band	imx6qp-sabresd-btwifi-m2.dtb	uSD-M.2	No
imx6qsabresd	OOB IRQ	imx6q-sabresd-btwifi-m2-oob.dtb	uSD-M.2	No
imx6qsabresd	SDIO in-band	imx6q-sabresd-btwifi-m2.dtb	uSD-M.2	No
imx6dlsabresd	OOB IRQ	imx6dl-sabresd-btwifi-m2-oob.dtb	uSD-M.2	No
imx6dlsabresd	SDIO in-band	imx6dl-sabresd-btwifi-m2.dtb	uSD-M.2	No
imx6sxsabresd	OOB IRQ	imx6sx-sdb-btwifi-m2-oob.dtb	uSD-M.2	No
imx6sxsabresd	SDIO in-band	imx6sx-sdb-btwifi-m2.dtb	uSD-M.2	No
imx6slevk	SDIO in-band	imx6sl-evk-btwifi-m2.dtb	uSD-M.2	No
imx6ulevk	OOB IRQ	imx6ul-14x14-evk-btwifi-m2-oob.dtb imx6ul-9x9-evk-btwifi-m2-oob.dtb	uSD-M.2	Yes
imx6ulevk	SDIO in-band	imx6ul-14x14-evk-btwifi-m2.dtb imx6ul-9x9-evk-btwifi-m2.dtb	uSD-M.2	Yes
imx6ull14x14evk	OOB IRQ	imx6ull-14x14-evk-btwifi-m2-oob.dtb	uSD-M.2	Yes
Imx6ull14x14evk	SDIO in-band	imx6ull-14x14-evk-btwifi-m2.dtb	uSD-M.2	Yes
imx6ull9x9evk	OOB IRQ	imx6ull-14x14-evk-btwifi-m2-oob.dtb	uSD-M.2	Yes
imx6ull9x9evk	SDIO in-band	imx6ull-9x9-evk-btwifi-m2.dtb	uSD-M.2	Yes

3.3 Murata "fmac" Customized i.MX Yocto Image Build

The NXP i.MX image contains third party IP which is sub-licensed via a click-through EULA (when either downloading an i.MX validation/demo image or building the image from source). As such Murata cannot make this image available directly to customers. Previously, customers could download the "bcmdhd" enabled i.MX image directly from NXP's website. However, given the transition to "***fmac***", the user must now build the Yocto Linux image (for any kernel prior to 4.14.62[4]). As detailed by the Murata Linux User Manual, Murata employs a customized "meta-murata-wireless" layer to make this customized Yocto build as simple as possible. Nonetheless end users still must configure a Linux build environment and follow specific steps to arrive at the desired image for a given i.MX target and Murata WLAN/BT configuration.

Murata has greatly simplified the build requirement by providing scripts for Ubuntu host setup and customized Yocto build – scripts easily downloadable from Murata's Github. Steps for downloading, configuring and invoking these scripts are detailed here.

3.3.1 Install Ubuntu

First step is to install Ubuntu 12.04, 14.04 or 16.04 (Murata's build is verified on Ubuntu 16.04 64-bit install) on the host - native PC or virtual environment like VMware. Host PC typically used has Ubuntu 16.04/14.04/12.04 installed with 50 GB free disk space (80 GB needed for i.MX8 build).

NOTE: Murata has verified these build steps using Ubuntu 16.04 (x64). For more information on the Ubuntu download, please refer to this link: https://www.ubuntu.com/download/desktop. The Ubuntu installation manual is provided here.

3.3.2 Download Murata's Script Files

With Ubuntu installed, we need to get the script files downloaded. There are two options:

a) Using "web browser" option to download "meta-murata-wireless" zip file and extract:

- Click on "clone or download" button at: https://github.com/murata-wireless/meta-murata-wireless.
- Now select "Download ZIP" option.
- Once the file is downloaded, extract it with "unzip" command or folder UI.
- Now go to the "meta-murata-wireless-master/cyw-script-utils/latest" folder where the necessary README and script files are contained.

[4] Starting with Linux kernel 4.14.62 version, NXP integrated the Cypress "fmac" driver directly into their kernel release (i.e. "drivers/net/wireless/broadcom/brcm80211/brcmfmac/" folder). For kernels prior to Linux 4.14.62, the only solution is to use "backports" as part of "meta-murata-wireless" implementation. Note that the "backports" implementation compiles "fmac" driver code against selected kernel, generating loadable modules which are later written to the file system. With built-in "fmac", the driver code can be either compiled into the kernel image or generated as loadable modules. However, the key differentiator is that the built-in "fmac" compiles code included in the kernel release. For specific details refer to Murata Linux User Manual.

OR:

b) Use "***wget***" command to pull specific files from Murata Github (**NOTE:** we need to set script files as executable afterwards with "***chmod a+x***" command because "wget" does not maintain the file permissions correctly):

```
wget --no-check-certificate --content-disposition https://github.com/murata-wireless/meta-murata-wireless/raw/master/cyw-script-utils/latest/README.txt

wget --no-check-certificate --content-disposition https://github.com/murata-wireless/meta-murata-wireless/raw/master/cyw-script-utils/latest/Host_Setup_for_Yocto.sh

wget --no-check-certificate --content-disposition https://github.com/murata-wireless/meta-murata-wireless/raw/master/cyw-script-utils/latest/Murata_Wireless_Yocto_Build.sh

chmod a+x *.sh
```

3.3.3 Configure Ubuntu for i.MX Yocto Build

Next step is configuring Ubuntu for Yocto build. Please run Murata's host setup script (should already be downloaded at this stage): "*Host_Setup_for_Yocto.sh*". To examine the plain ASCII text version, you can go to this link or just hit the "Raw" button. For more information (README file), just go to the main folder: https://github.com/murata-wireless/meta-murata-wireless/tree/master/cyw-script-utils/latest. The "latest" folder is used to maintain the most recent/up-to-date script.

Murata's script installs necessary additional packages required for the Yocto build. For additional information, refer to NXP Yocto Project User's Guide (part of NXP Reference Documents release). Murata's script will prompt user for password – as supervisory access is needed to install various packages. GIT is also configured so it can be used later during the build process. For more information on first-time GIT setup, you can refer to this link. Running the script file is straightforward. Simply invoke at Ubuntu "terminal" prompt (folder location is not important):

```
./Host_Setup_for_Yocto.sh
```

The script goes through the following stages:

```
1) Verifying Host Environment
2) Verifying Host Script Version
3) Installing Essential Yocto host packages
4) GIT Configuration: verifying User name and email ID
```

For an example input/output sequence, refer to Appendix C of Linux User Manual.

3.3.4 **Murata's i.MX Yocto Build Script**

With Ubuntu installed and configured to build i.MX Yocto, please run the build script (should already be downloaded at this stage): "Murata_Wireless_Yocto_Build.sh". For plain ASCII text version, you can go to this link or just hit the "Raw" button. For more information (README file), just go to the main folder: https://github.com/murata-wireless/meta-murata-wireless/tree/master/cyw-script-utils/latest. The "latest" folder is used to maintain the most recent/up-to-date script.

Prior to running Murata's build script, make sure you have completed the following:

- Installed 64-bit version of Ubuntu 16.04 (preferred), 14.04, or 12.04.
- Ran Murata's host setup script in **Section 3.3.3** to add necessary packages for Yocto build and configure GIT.
- Created a i.MX BSP folder ***specific*** to the desired i.MX Yocto Release. The i.MX Yocto distribution **cannot** build different versions of Yocto (Linux kernel) in the same folder. Currently the following Yocto releases are supported:
  - 4.9.123_2.3.0 GA
  - 4.9.88_2.0.0 GA
  - 4.9.11_1.0.0 GA
  - 4.1.15_2.0.0 GA
- Once the build script successfully completes, the i.MX BSP folder will contain:
  - Yocto "***sources***" and "***downloads***" folder.
  - "***meta-murata-wireless***" folder – is a sub-folder of "***sources***".
  - One or more i.MX build folders.

NOTE: when creating a i.MX BSP folder (***$BSP_DIR*** or "***murata-imx-bsp***" used to reference this all-important folder later in this document), make sure that no parent folder contains a "***.repo***" folder. Creating the i.MX BSP folder is straightforward:

```
cd ~
mkdir murata-imx-bsp
cd murata-imx-bsp
cp <Script Path>/Murata Wireless Yocto_Build.sh
```

Murata's build script performs the following tasks:

- Verifies host environment (i.e. Ubuntu 12.04/14.04/16.04).
- Check to make sure script being run is the latest version.
- Prompts the user to select release type:
  - "**Stable**" corresponds to "***meta-murata-wireless***" release/tag (rather than a branch). Murata tests wireless functionality on i.MX platforms for each release/tag. This release type is recommended for baseline image builds or initial bring-up testing.
  - "**Developer**" corresponds to a branch which can be a "moving target". When performing the automated build, the script file pulls the latest branch contents – as opposed to a specific GIT commit on that branch. If the user wants the latest fixes, then this is the best option to go with. **NOTE**: Murata only runs "spot" tests before submitting fixes/enhancements to the branch.

- Select the "***fmac***" release. The script displays both the "***fmac***" codename and latest kernel version supported by that release. Currently, four "***fmac***" releases are supported: "***orga***", "***battra***", "***mothra***", and "***manda***" (most recent and up-to-date regarding fixes and enhancements). Murata strongly recommends using "***manda***" release.
- Select i.MX Yocto release. As already pointed out the current i.MX BSP folder (from which script is being executed) can only support **one** i.MX Yocto release. If you need to test/evaluate different Yocto/kernel versions, then **you must** create additional folders.
- Select i.MX target: refer to **Table 7** for more details.
- If the i.MX target selected supports alternative VIO options (i.e. i.MX6UL(L) supports 1.8V VIO signaling), then the script prompts the user for the desired configuration. With the limitation of uSD-M2 adapter, SDIO clock can only reach 50MHz for both 1.8V and 3.3V.
- Select "DISTRO and image". This configures the graphical driver and Yocto image. For more details refer to the Yocto documentation. It is recommended to go with Murata defaults on this step – Murata has tested/validated with these images.
- Name desired build target folder name. If re-running the Murata build script, this folder name must be unique.
- Review the final configuration and accept before moving forward.
- Accept the NXP/Freescale End User's License Agreement (EULA). There is 3rd party IP included in the i.MX Yocto build. This step addresses the sub-licensing issue. During this step, the user must review a fair bit of legal documentation (by repeatedly entering space bar) or if already familiar with the EULA language, enter 'q' to bypass displaying the complete agreement. The final step of this EULA step prompts the user to enter "y" to accept.
- Last and final step is to confirm that user wants to kick off the final build process (invoke "***bitbake* <image>**" command).

Running the script file is straightforward. Simply invoke from your i.MX BSP folder (**$BSP_DIR** or "**murata-imx-bsp**" – already created by this point):

```
./Murata_Wireless_Yocto_Build.sh
```

The script goes through the following stages:

```
1) Verifying Host Environment
2) Verifying Script Version
3) Select Release Type:
      a) Stable: Murata tested/verified release tag. Stable is the recommended default.
      b) Developer: Includes latest fixes on branch. May change at any time.
4) Select "fmac" version
5) Select i.MX Yocto Release
6) Select Target
   6.1) Select VIO Signaling
7) Select DISTRO & Image
8) Creation of Build directory
9) Verify your selection
10) Acceptance of End User License Agreement(EULA)
11) Starting Build Now. Note: depending on machine type, build may take 1-7 hours to complete.
```

For an example input/output sequence, refer to Appendix D of Linux User Manual. Once the Murata-customized i.MX image is built, it will be located at the following location:

```
<$BSP_DIR>/<build target folder – selected during script>/tmp/deploy/images/<$target>/
```

Or, if using i.MX 6UL EVK as example with "murata-imx-bsp" folder:

```
~/murata-imx-bsp/imx6ulevk_build/tmp/deploy/images/imx6ulevk/
```

i.MX 6UL EVK validation SD card image name would be:

```
fsl-image-validation-imx-imx6ulevk.sdcard
```

With SD card image built and located, refer to **Section 4** for preparing bootable SD card.

3.4 Additional Hardware/Software Considerations

3.4.1 Out-Of-Band (OOB) IRQ Support on NXP i.MX EVK's

The preferred interrupt configuration is OOB IRQ. Murata recommends that the user runs OOB IRQ (if possible) on all i.MX reference platforms. Otherwise power-save mechanisms are limited. Specifically, the host cannot shut down the SDIO bus when using SDIO in-band signaling for WLAN interrupt mechanism.

The following i.MX 6 Platforms don't support WLAN OOB IRQ configuration out-of-box:

- **i.MX 6Q/DL SDB/SDP or i.MX 6 QP SDB**: all these platforms require rework to enable Bluetooth and WLAN/Bluetooth control signals. Refer to the Hardware User Manual for specifics (some of which are described in **Section 5.3**).
- **i.MX 6SoloX SDB**: this platform will work with SDIO in-band signaling. However, to provide correct WL_REG_ON (WLAN core enable/disable) **OR** OOB IRQ support, rework must be done. Refer to the Hardware User Manual for specifics.
- **i.MX 6UL(L) EVK's**: these platforms require one resistor to be move to support OOB IRQ support. **NOTE:** For OOB IRQ configuration on i.MX6UL/ULL EVK, the second (of two) Ethernet ports is disabled due to hardware conflict (documented in Hardware User Manual).
- **i.MX 6SL EVK**: this platform does not support WLAN/BT control signals or BT UART. There is no option for 20-pin FFC interconnect (ribbon cable). **NOTE:** the i.MX 6SLL EVK does not suffer the same limitation – it does provide 20-pin FFC interconnect.

In summary, SDIO in-band interrupts are recommended unless necessary rework is done. Refer to the Murata Hardware User Manual for necessary modifications on all NXP i.MX 6 Platforms.

3.4.2 UHS SDIO 3.0 operation on i.MX Platforms with uSD-M.2 Adapter

When using uSD-M2 adapter, the maximum SDIO clock frequency is only 50MHz for both 1.8V and 3.3V VIO. For UHS mode support (i.e. MAX SDIO clock is 200 MHz for Type 1MW) and for comprehensive signal support, ***Murata recommends*** the Embedded Artists' i.MX Developer Kits.

4 Preparing Bootable SD Card for i.MX with Murata Wi-Fi/BT EVK

4.1 Linux PC Steps to Flash SD Card

Now that the SD card image is built, we can now flash the (micro) SD card used for booting the i.MX platform. Insert the (micro) SD card into a host machine (PC). It is imperative that the (micro) SD card comes up as "/dev/sdx" device. If it does not, then you may require a USB to (micro) SD card adapter as shown in **Figure 5**. This Kingston device (MobileLite G4) provides direct plug-ins for microSD and SD cards. It supports USB 3.0 and UHS SD cards – allowing very fast transfer speeds. With the "right" (micro) SD Card Reader/Writer and UHS (micro) SD card, flashing a 1 GB i.MX image can be done in 10~20 seconds versus 1~2 minutes (or more).

Figure 5: USB to SD Card Reader/Writer Adapter

Once the (micro) SD card has been inserted into the PC, run the "***dmesg***" command to find which "/dev/sdx" device was just enumerated:

```
dmesg
```

The enumeration log of the *just* inserted (micro) SD card should look like:

```
[285317.464075] usbcore: registered new interface driver usb-storage
[285318.472525] scsi 6:0:0:0: Direct-Access     Generic- USB3.0 CRW    -0 1.00 PQ: 0 ANSI: 4
[285318.473143] sd 6:0:0:0: Attached scsi generic sg2 type 0
[285319.263194] sd 6:0:0:0: [sdc] 15597568 512-byte logical blocks: (7.98 GB/7.43 GiB)
[285319.264368] sd 6:0:0:0: [sdc] Write Protect is off
[285319.264379] sd 6:0:0:0: [sdc] Mode Sense: 2f 00 00 00
[285319.265413] sd 6:0:0:0: [sdc] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[285319.274779]  sdc: sdc1 sdc2
```

Referencing this example log, the correct device for the (micro) SD card is "***/dev/sdc***".

NOTE: Before running next command, make sure you have selected the correct device. Otherwise you may unintentionally **WIPE/ERASE YOUR HARD DRIVE!!** Substitute the correct (micro) SD device name for "***/dev/sdx***" in "***dd***" command line below.

Following the "***imx6ulevk***" target example, the command for flashing the (micro) SD is:

```
sudo dd if=$BUILD_DIR/tmp/deploy/images/imx6ulevk/fsl-image-validation-imx-imx6ulevk.sdcard of=/dev/sdx bs=1M && sync
```

⇨ **SD Card is now flashed with customized Murata wireless image which integrates "*fmac*" driver and other components listed in Section 3.1.**

4.2 Windows PC Steps to Flash SD Card

In case you need to later flash the same (micro) SD card image using a Windows PC, the following steps have been included. Windows utilities such as "Win32 Disk Imager" or "NetBSD Disk Image Tool" can be used to flash the (micro) SD card. For example, when using "Win32 Disk Imager", follow these steps:

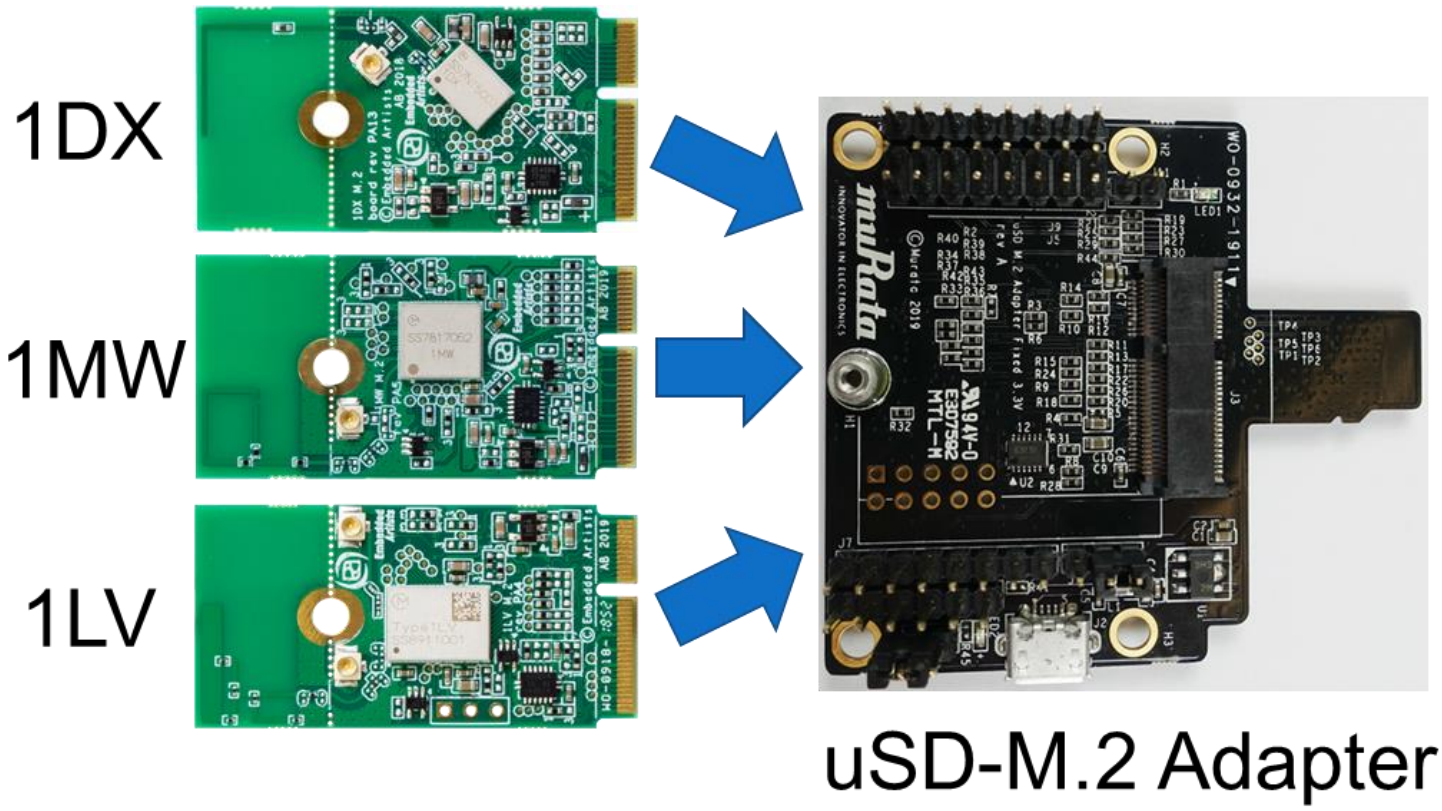
- After bringing up "Win32 Disk Imager" program[5], click on the folder icon/button and navigate to the location of the desired "*.sdcard" file. You need to change "*.img" file type to "*.*" to select/open the "*.sdcard" file.
- Select the "Device" button and select the drive letter corresponding to the (micro) SD card: formatting SD card may be necessary beforehand with Windows low-level utilities[6].
- Now click the "Write" button. A warning window will pop up that warns that the device being written to may be corrupted.
- Upon completion, a window with "Write Successful" should appear.
- Click "OK" on the "Write Successful" window.
- Now click "Exit" on "Win32 Disk Imager" window.
- To be safe, you may elect to "eject" the SD card removable memory device before removing it.

[5] "Win32 Disk Imager" is an open source tool that can be downloaded from websites such as "sourceforge.net".
[6] Unlike Linux environment, Windows PC does not require use of "USB to SD Card Reader/Writer" adapter.

5 Murata Wi-Fi/BT Bring-Up on i.MX 6 Platforms

Any of the (WLAN-SDIO) Embedded Artists' Wi-Fi/BT M.2 EVB's listed in **Table 5** (currently 1DX/1MW/1LV) can be connected to the i.MX 6 Platforms through Murata's uSD-M.2 Adapter as shows. The following sub-sections details steps for bringing up Embedded Artists' Wi-Fi/BT EVB's on the four major variants of the i.MX 6 Platform. The specific steps described only vary when switching WLAN SDIO VIO between 3.3V (default configuration on most i.MX platforms) and 1.8V (necessary for Type 1LV). Note that only NXP i.MX 6UL/ULL EVK's support 1.8V VIO signaling.

Figure 6: uSD-M.2 Adapter with type 1DX/1MW/1LV M.2 EVB options



5.1 Connecting to i.MX 6SoloX SDB

The Murata Wi-Fi/BT EVB is connected via the SD2 slot: see **Figure 7** below.

[1] Ensure no power is applied to i.MX 6SoloX SDB. Connect J16 micro-USB port to PC and start terminal emulator: “Minicom” on Linux or “Tera Term” on Windows. Set port to 115200-N-8-1.
[2] Set Jumper J1 to position 2-3 (VBAT from microSD connector). Make sure Jumper J11 is removed (BT core not disabled). Insert Jumper J12 to set VIO to 3.3V (Blue LED2 illuminates).
[3] Connect the Wi-Fi/BT M.2 EVB to the M.2 connector on the uSD-M2 Adapter board.
[4] Connect the ribbon cable at both ends before inserting uSD-M.2 Adapter board into SD2 slot. Note the orientation as shown in **Figure 7**.
[5] Prepare SD card to boot platform per **Section 4**. Insert SD card, power on the platform and interrupt at u-boot. Now type (“oob” string configures OOB IRQ configuration – if i.MX Platform has been configured correctly; see **Section 3.4.1** for more details):

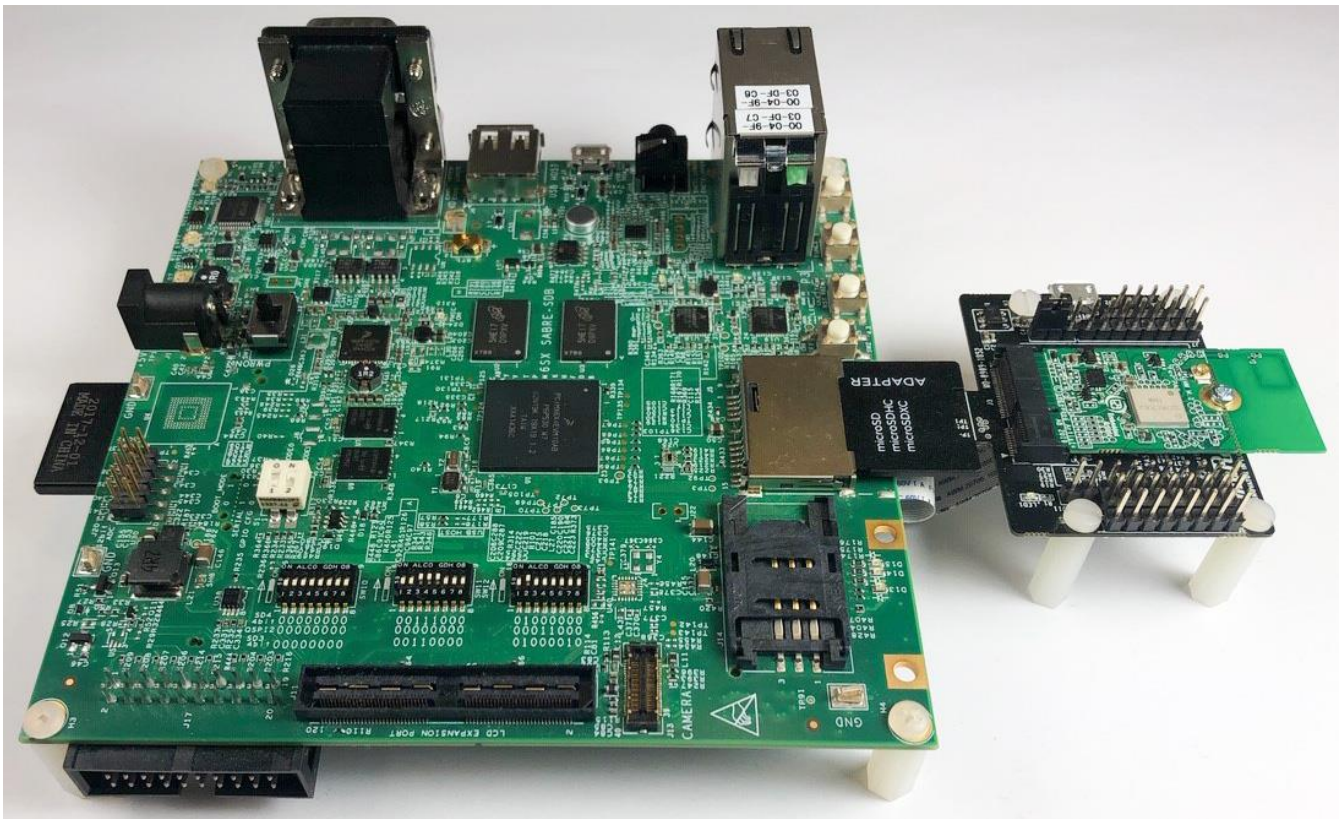
```
setenv fdt_file imx6sx-sdb-btwifi-m2<-oob>.dtb
```

Now save the u-boot configuration and boot the platform:

```
saveenv
boot    ← causes platform to boot kernel
```

[6] Refer to **Section 7** to test/verify Wi-Fi and Bluetooth functionality.

Figure 7: i.MX 6SoloX SDB with uSD-M.2 Adapter and Type 1MW M.2 EVB



5.2 Connecting to i.MX 6SoloLite EVK

Murata Wi-Fi/BT EVB is connected via SD1 slot: see **Figure 8: i.MX 6SoloLite EVK with uSD-M.2 Adapter and Type 1DX M.2 EVB**.

[1] Ensure no power is applied to i.MX 6SL EVK. Connect J26 micro-USB port to PC and start terminal emulator: “minicom” on Linux or “tera term” on Windows. Set port to 115200-N-8-1.
[2] Set Jumper J1 to position 2-3 (VBAT from microSD connector). Jumper J11 can optionally be inserted to disable BT core. Insert Jumper J12 to set VIO to 3.3V (Blue LED2 illuminates).
[3] Connect the Wi-Fi/BT M.2 EVB to the M.2 connector on the uSD-M2 Adapter board.
[4] Insert uSD-M.2 Adapter board into SD1 slot (no ribbon cable interconnect is possible). Note the orientation as shown in the figure.
[5] Prepare SD card to boot platform per **Section 4**. Insert SD card, power on platform and interrupt at u-boot. Now type (“oob” is not an option as only SDIO in-band interrupt configuration is supported; see **Section 3.4.1** for more details):

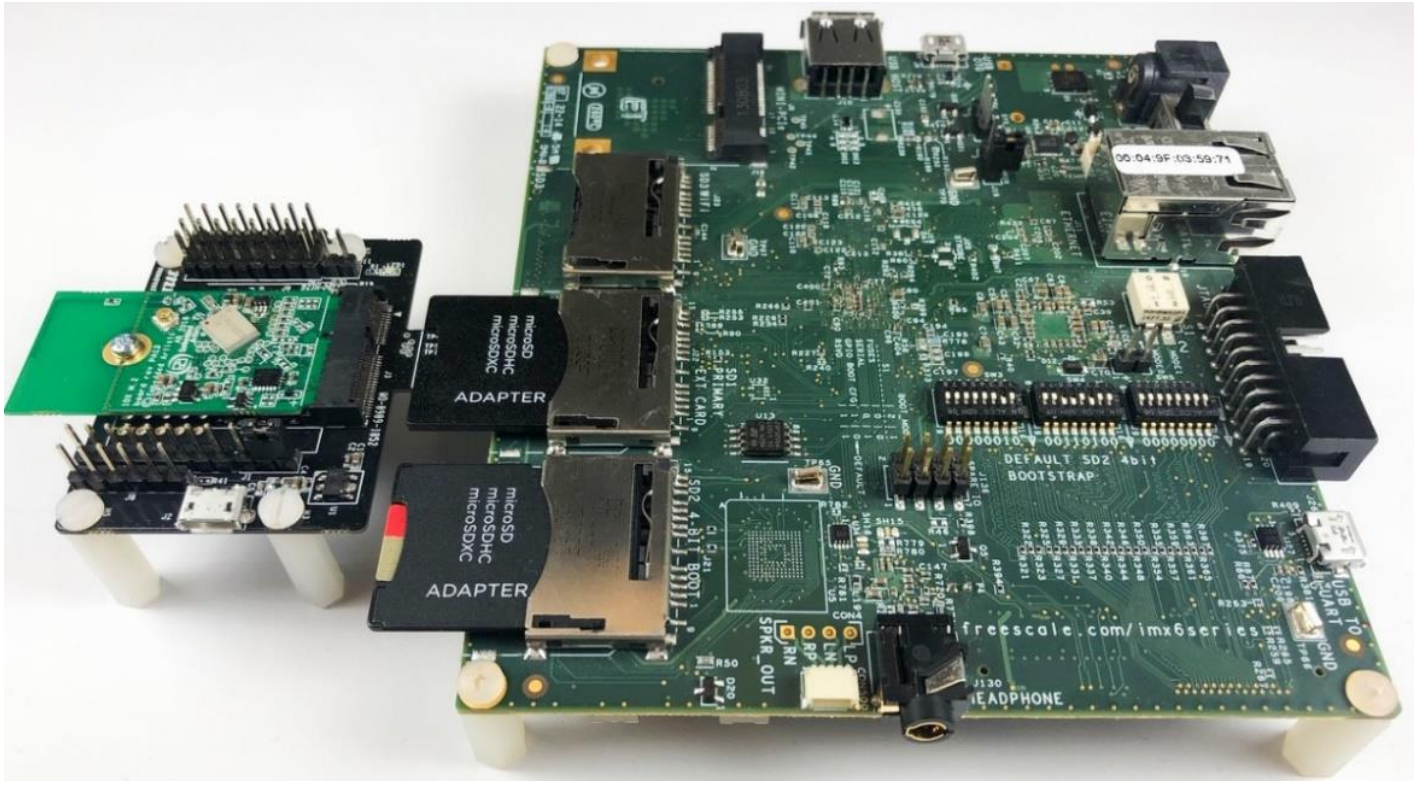
```
setenv fdt_file imx6sl-evk-btwifi-m2.dtb
```

Now save the u-boot configuration and boot the platform:

```
saveenv
boot    ← causes platform to boot kernel
```

[6] Refer to **Section 7** to test/verify Wi-Fi and Bluetooth functionality.

Figure 8: i.MX 6SoloLite EVK with uSD-M.2 Adapter and Type 1DX M.2 EVB



5.3 Connecting to i.MX 6Q/DL SDB/SDP or i.MX 6 QP SDB

The following section provides bring-up instructions for i.MX 6QuadPlus SDB, i.MX 6Quad/DualLite SDB, and i.MX 6Quad/DualLite SDP. The i.MX 6QuadPlus SDB has a modified schematic from the (essentially identical) i.MX 6Quad/DualLite SDB/SDP platforms.

5.3.1 Specific Hardware Considerations for i.MX 6Quad/DualLite SDB/SDP

Although the Murata Wi-Fi/BT EVK is designed to be “plug ‘n play”, **rework is required** for the i.MX 6Quad/DualLite SDB/SDP platforms. As shipped from the factory, the i.MX 6Q/DL SDB/SDP **do not** connect the J13 Bluetooth ribbon cable connector to the necessary UART and control signals. Refer to the Hardware User Manual for necessary rework. NXP also details the board rework in their schematic file (Bluetooth page). Page 15 of the NXP schematic (SPF-27516_C3.pdf) correctly captures the necessary rework to be done. Those schematic notes are repeated below.

NOTE: To use J13, populate resistors R209 - R213 and depopulated the SPI NOR FLASH U14. Resistors R214 and R215 should not be populated because both UART outputs (TXDs) have been crossed together and both UART inputs (RXDs) have been crossed together. To make the UART work correctly, solder a jumper wire from R215 pad 1 to R214 pad 2 and from R215 pad 2 to R214 pad 1.

5.3.2 Specific Hardware Considerations for i.MX 6QP SDB

Depending on the revision, rework **may be required** for the i.MX 6QuadPlus SDB. This rework connects the Bluetooth UART and Wi-Fi/BT control signals (WL_REG_ON, BT_REG_ON, and WL_HOST_WAKE). Revision B of i.MX 6QP SDB populates the necessary resistors for connecting BT UART and Wi-Fi/BT control signals. If your board has revision earlier than B (i.e. A2) then you will have to populate the necessary resistors. Refer to the Hardware User Manual for necessary rework. For the Rev A2 board, page 15 of the NXP schematic (SPF-28857_A2.pdf) correctly captures the necessary rework to be done. Note the much simpler rework on the i.MX 6QB SDB given that the no special “crossing” of TX and RX resistor pads is necessary. Those schematic notes are repeated below.

NOTE: To use J13, populate resistors R209 - R213 and depopulate the SPI NOR FLASH U14.

5.3.3 Wi-Fi/Bluetooth Bring-Up on i.MX 6Q/DL SDB/SDP or i.MX 6 QP SDB

Note: The following steps will only pass if NXP Platform has been correctly reworked The NXP i.MX6 platform has been inverted. This makes working with Wi-Fi/BT EVK much easier. The one drawback is Ethernet port access. To properly match Wi-Fi/BT EVK and i.MX6 platform heights, additional nylon standoffs are required.

[1] Ensure no power is applied to i.MX 6Q(P)/DL SDB/SDP. Connect J509 micro-USB port to PC and start emulator: “Minicom” on Linux or “Tera Term” on Windows. Set port to 115200-N-8-1.
[2] Set Jumper J1 to position 2-3 (VBAT from microSD connector). Make sure Jumper J11 is removed (BT core not disabled). Insert Jumper J12 to set VIO to 3.3V (Blue LED2 illuminates).
[3] Connect the Wi-Fi/BT M.2 EVB to the M.2 connector on the uSD-M2 Adapter board.
[4] Connect the ribbon cable at both ends before inserting uSD-M.2 Adapter board into SD2 slot. Note the orientation as shown in **Figure 9**.
[5] Prepare SD card to boot platform per **Section 4**. Insert SD card, power on the platform and interrupt at u-boot. Now type (“oob” string configures OOB IRQ configuration; see **Section 3.4.1** for more details):

```
setenv fdt_file imx6q-sabresd-btwifi-m2<-oob>.dtb
                (OR imx6dl-sabresd-btwifi-m2<-oob>.dtb,
                    imx6qp-sabresd-btwifi-m2<-oob>.dtb)
```

Now save the u-boot configuration and boot the platform:

```
saveenv
boot  ← causes platform to boot kernel
```

[6] Refer to **Section 7** to test/verify Wi-Fi and Bluetooth functionality.

Figure 9: i.MX 6Quad/DualLite SDB (Inverted) with uSD-M.2 Adapter and Type 1MW M.2 EVB

5.4 Connecting to i.MX 6UltraLite EVK or i.MX 6ULL EVK

[1] Ensure no power is applied to i.MX 6UltraLite or i.MX 6ULL EVK. Connect J1101 micro-USB port to PC and start terminal emulator: “Minicom” on Linux or “Tera Term” on Windows. Set port to 115200-N-8-1.
[2] Set Jumper J1 to position 2-3 (VBAT from microSD connector). Make sure Jumper J11 is removed (BT core not disabled). Depending on desired VIO; either insert Jumper J12 for 3.3V (Blue LED2 illuminates) or remove it for 1.8V VIO. Note that the i.MX 6UL(L) image is specific for either 1.8V or 3.3V signaling.
[3] Connect the Wi-Fi/BT M.2 EVB to the M.2 connector on the uSD-M2 Adapter board.
[4] Connect ribbon cable at both ends before inserting Murata EVK into SD1 slot. Note the orientation as shown in **Figure 10**. Make sure that the adapter clicks in correctly.
[5] Prepare microSD card to boot platform per **Section 4**. Insert microSD card, power on the platform and interrupt at u-boot. Now type (“oob” string configures OOB IRQ configuration; see **Section 3.4.1** for more details):

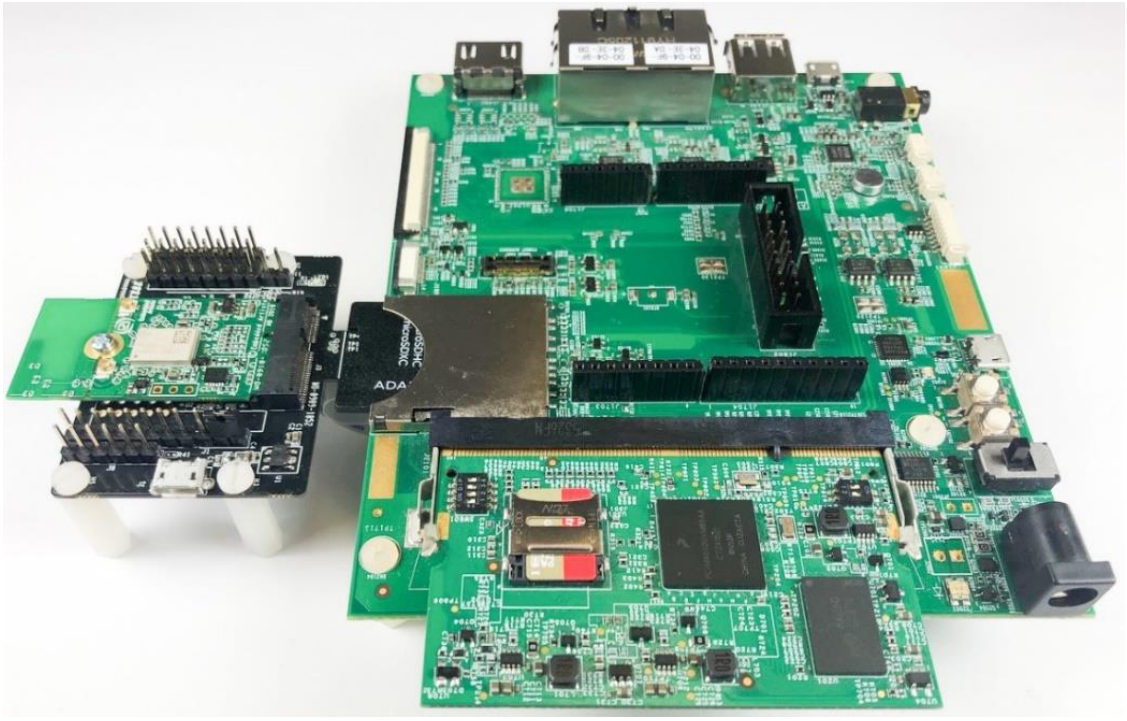
```
setenv fdt_file imx6ul-14x14-evk-btwifi-m2<-oob>.dtb
                (OR imx6ul-9x9-evk-btwifi-m2<-oob>.dtb,
                    imx6ull-14x14-evk-btwifi-m2<-oob>.dtb,
                    imx6ull-9x9-evk-btwifi-m2<-oob>.dtb)
```

Now save the u-boot configuration and boot the platform:

```
saveenv
boot  ← causes platform to boot kernel
```

[6] Refer to **Section 7** to test/verify Wi-Fi and Bluetooth functionality.

Figure 10: i.MX 6UltraLite EVK with uSD-M.2 Adapter and Type 1LV M.2 EVB



6 Murata Wi-Fi/BT Bring-Up on i.MX 8 Platforms

6.1 Bringing up Wi-Fi/BT on i.MX 8MQuad EVK

The NXP i.MX 8MQuad EVK (see **Figure 11**) provides a secondary Wi-Fi/BT solution on the underside via a M.2 connector. The M.2 interconnect provides WLAN PCIe, BT UART, control signals, and optionally BT PCM. Currently the only supported M.2 EVB is Type 1CX (WLAN-PCIe).

[1] Connect J1701 micro-USB port to PC and start terminal emulator: “Minicom” on Linux or “Tera Term” on Windows. Set port to 115200-N-8-1.
[2] Connect Embedded Artists Type 1CX M.2 EVB to M.2 connector as shown in **Figure 11**. Attach two dual-band (2.4/5GHz) antennas with U.FL connector as shown – otherwise there is severe signal attenuation. Now you can flip platform right-side-up. Try and keep antennas at right angles for optimum WLAN throughput performance.
[3] Prepare micro SD card to boot platform per **Section 4**.
[4] Insert SD card and power on platform and interrupt at u-boot. Now type:

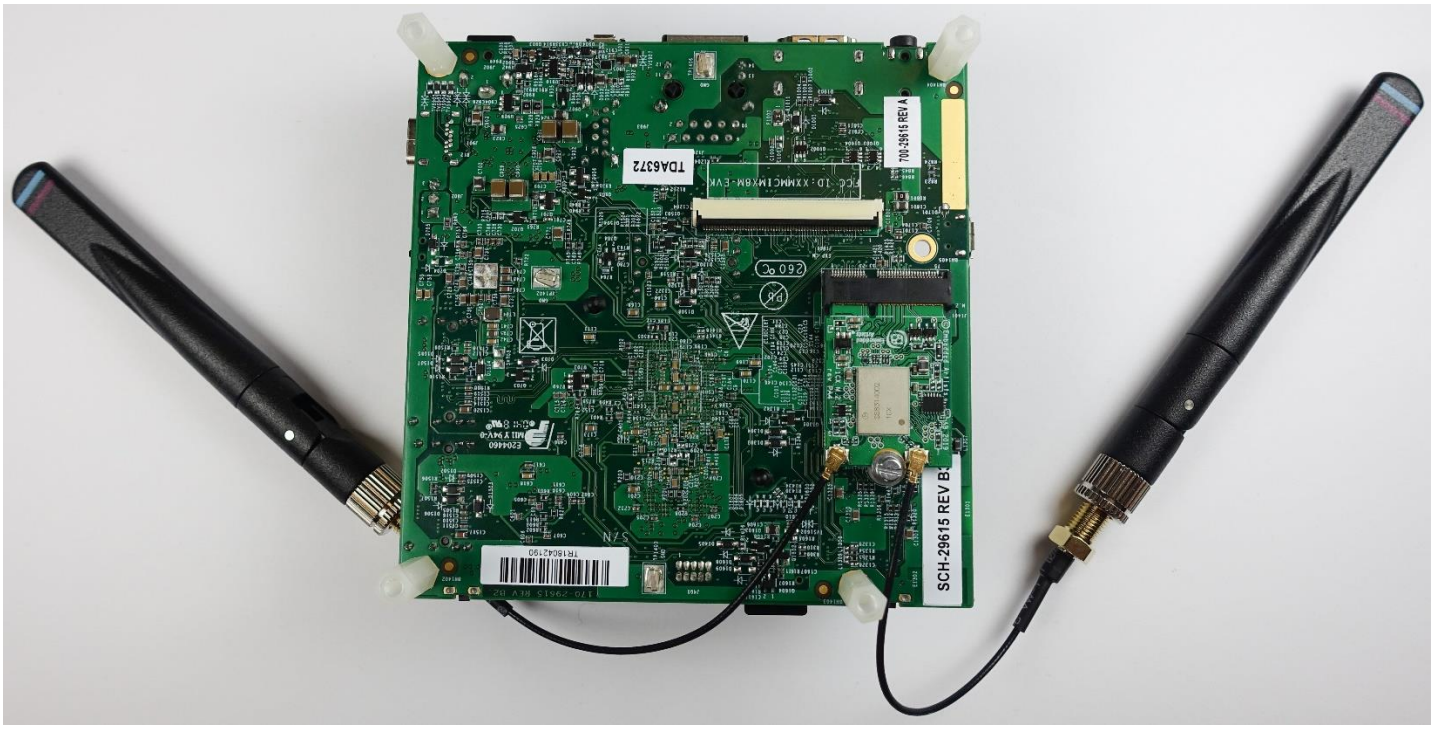
```
setenv fdt_file fsl-imx8mq-evk-pcie1-m2.dtb
```

Now save the u-boot configuration and boot the platform:

```
saveenv
boot   ← causes platform to boot kernel
```

[5] Refer to **Section 7** to test/verify Wi-Fi and Bluetooth functionality.

Figure 11: i.MX 8MQuad with Type 1CX (bottom view)



6.2 Bringing up Wi-Fi/BT on i.MX 8M Mini EVK

The NXP i.MX 8M Mini EVK (see **Figure 12**) provides a secondary Wi-Fi solution on the underside via a M.2 connector. The M.2 interconnect provides WLAN PCIe, and WLAN control signals **with no** Bluetooth signal support (no BT UART/PCM/Control signals). Currently the only supported M.2 EVB is Type 1CX (WLAN-PCIe).

[1] Connect J901 micro-USB port to PC and start terminal emulator: “Minicom” on Linux or “Tera Term” on Windows. Set port to 115200-N-8-1.
[2] Connect Embedded Artists Type 1CX M.2 EVB to M.2 connector as shown in **Figure 12**. Attach two dual-band (2.4/5GHz) antennas with U.FL connector as shown – otherwise there is severe signal attenuation. Now you can flip platform right-side-up. Try and keep antennas at right angles for optimum WLAN throughput performance.
[3] Prepare micro SD card to boot platform per **Section 4**.
[4] Insert the power supply came with the EVK into J302 USB Type C connector.
[5] Insert SD card and power on platform and interrupt at u-boot. Now type:

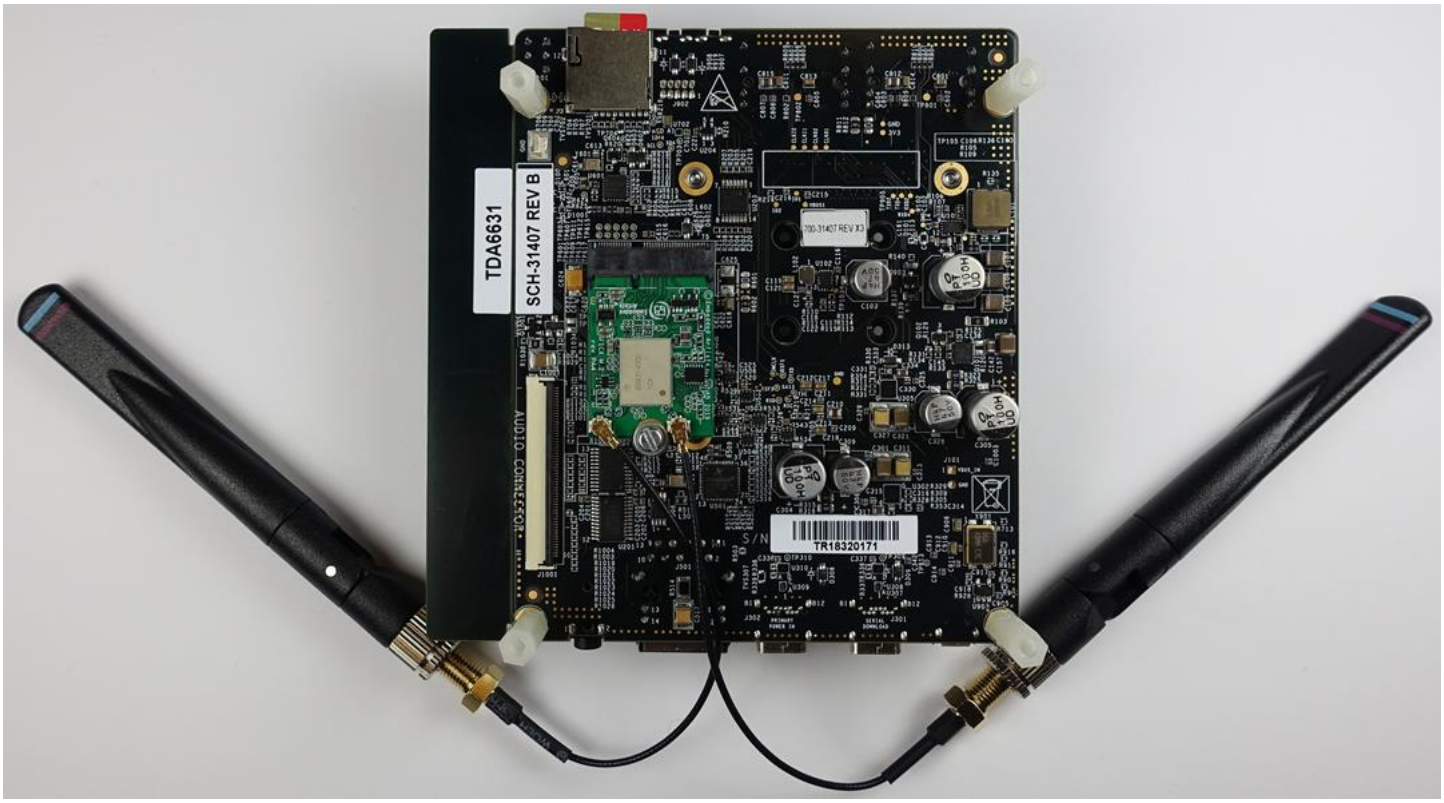
```
setenv fdt_file fsl-imx8mm-evk.dtb
```

a. Now save the u-boot configuration and boot the platform:

```
saveenv
boot  ← causes platform to boot kernel
```

[6] Refer to **Section 7** to test/verify Wi-Fi and Bluetooth functionality.

Figure 12: i.MX 8M Mini with Type 1CX (bottom view)



7 Test/Verification of Wi-Fi and Bluetooth

Now the kernel should be booting correctly on the platform with Murata Wi-Fi/BT EVK plugged in (i.e. already set correct DTB file when interrupting u-boot). Next steps are to verify Wi-Fi and Bluetooth functionality. The NXP i.MX (Murata modified) images include all the necessary files to support Wi-Fi and Bluetooth bring-up/testing/verification. The relevant folders and files are summarized in **Table 9**.

Table 9: Embedded Wi-Fi/Bluetooth Files

Filename or Folder	Details
/lib/firmware/brcm/brcmfmac<chipset>-sdio.bin	“fmac” firmware file for WLAN SDIO chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-sdio.txt	“fmac” NVRAM file for WLAN SDIO chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-sdio.clm_blob	“fmac” regulatory binary for WLAN SDIO chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-pcie.bin	“fmac” firmware file for WLAN PCIe chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-pcie.clm_blob	“fmac” regulatory binary for WLAN PCIe chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-pcie.txt	“fmac” NVRAM file for WLAN PCIe chipset/module.
/etc/firmware/BCM4354A2.1CX.hcd	Type 1CX (CYW4356) Bluetooth patchfile.
/etc/firmware/BCM43012C0.1LV.hcd	Type 1LV (CYW43012) Bluetooth patchfile.
/etc/firmware/BCM4345C0.1MW.hcd	Type 1MW (CYW43455) Bluetooth patchfile.
/etc/firmware/BCM43430A1.1DX.hcd	Type 1DX (CYW4343W) Bluetooth patchfile.
/usr/sbin/wl	WL tool is used for Wi-Fi RF and manufacturing tests. Also useful tool for initial bring-up and debug.
/usr/sbin/iw	Linux “iw” executable.
/usr/sbin/wpa_supplicant	WPA supplicant executable.
/usr/sbin/wpa_cli	WPA CLI tool.
/usr/bin/wpa_passphrase	WPA Passphrase generator.
/etc/wpa_supplicant.conf	WPA supplicant configuration file.
/usr/sbin/hostapd	Hostapd executable – manages wireless link in Soft AP mode.
/usr/sbin/hostapd_cli	Hostapd CLI tool.
/etc/hostapd.conf	Hostapd configuration file.
/etc/udhcpd.conf	DHCP server configuration file.
/etc/network/interfaces	Modify this file to automatically bring up “wlan0” interface. Also assign static IP address if desired. Currently not configured. Only applies to Kernel 4.1.15.
/usr/bin/hciattach	“hciattach” binary – used for initializing Bluetooth UART connection.
/usr/bin/hciconfig	“hciconfig” binary – used for configuring Bluetooth interface.
/usr/bin/hcitool	“hcitool” binary – used for controlling Bluetooth interface.
/usr/bin/iperf3	iPerf throughput test tool.

7.1 Wi-Fi Interface Test/Verification

7.1.1 Useful Environment Setup on NXP Linux

Once the kernel has booted and user has logged in as “root” (no password), there are a couple of quick commands (sequence is important) which make the terminal console easier to work on:

```
$ stty rows 80 cols 132      ← set your favorite row and column width here
$ export TERM=ansi           ← invoke this command after “stty”
```

7.1.2 Bringing Up Wi-Fi Interface

As i.MX kernel boots, the “fmac” WLAN driver is loaded automatically. As part of driver loading sequence, the WLAN device is probed over the SDIO bus. As an example, we will bring up Wi-Fi when using following configuration:

- i.MX 6ULL EVK
- uSD-M.2 Adapter configured for 1.8V VIO (J12 is open)
- Type 1MW (CYW43455) EVB
- Murata i.MX image compiled for i.MX6ULL @1.8V

Expected output as kernel boots (can use “***dmesg***” later to display):

```
sdhci: Secure Digital Host Controller Interface driver
sdhci: Copyright(c) Pierre Ossman
sdhci-pltfm: SDHCI platform and OF driver helper
sdhci-esdhc-imx 2190000.usdhc: allocated mmc-pwrseq
sdhci-esdhc-imx 2190000.usdhc: assigned as wifi host   ← “wifi-host” descriptor in “mmc0” entry
Murata: mmc_power_up: Setting 1.8V for Index: 0        ← forces 1.8V VIO operation in kernel
mmc0: SDHCI controller on 2190000.usdhc [2190000.usdhc] using ADMA
sdhci-esdhc-imx 2194000.usdhc: could not get ultra high speed state, work on normal mode ← SDIO clock is limited to
50MHz in dtb files for imx6ul(l)evk, so usdhc can’t reach ultra high speed state
Murata: mmc_sdio_init_card: Skipping 1.8V setting for Index: 0
mmc0: queuing unknown CIS tuple 0x80 (2 bytes)         ← WLAN device enumeration
mmc0: queuing unknown CIS tuple 0x80 (3 bytes)
mmc0: queuing unknown CIS tuple 0x80 (3 bytes)
mmc0: queuing unknown CIS tuple 0x80 (7 bytes)
mmc0: queuing unknown CIS tuple 0x80 (6 bytes)
mmc0: new ultra high speed SDR104 SDIO card at address 0001   ← switch to UHS mode (SDR104)
…
brcmfmac: brcmf_fw_map_chip_to_name: using brcm/brcmfmac43455-sdio.bin for chip 0x004345(17221) rev 0x000006
usbcore: registered new interface driver brcmfmac
brcmfmac: brcmf_c_preinit_dcmds: Murata Customized Version: imx-rocko-manda_r1.0;
brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: Sep 21 2018 04:08:34 version 7.45.173 (r707987 CY) FWID
01-d2799ea2
```

In addition to the documented log messages, there are highlighted sections:

- "***brcmfmac***" string identifies "fmac" driver log messages
- "***brcmfmac43455-sdio.bin***" indicates the WLAN firmware file being loaded.
- "***Murata Customized Version***" indicates that "***meta-murata-wireless***" was used to generate this image. The branch or release/tag information is included.
- "***Firmware version***" indicates specific version of firmware being loaded by "***fmac***".

Now invoke "***ifconfig wlan0 up***" command to initialize the "***wlan0***" interface. Example output shown below:

```
$ ifconfig wlan0 up
IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready

$ ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr b8:d7:af:56:61:fc      ← WLAN MAC Address
          UP BROADCAST MULTICAST  MTU:1500  Metric:1         ← "wlan0" interface is UP
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

NOTE: no IP address is assigned yet to the "wlan0" interface. That will be done later **Section 7.1.6**.

7.1.3 STA/Client Mode: Scan for Visible Access Points

In this section, two different/simple methods for scanning are presented: one uses the Cypress "***wl***" tool; the other uses the Linux "***iw***" command. If you don't see a list of SSID's and there are broadcasting Access Points (Wireless Routers) in range, then something is wrong. Please check antenna connection, setup, etc. Also note that the strength of received signals is important to do connectivity testing (i.e. "***ping***", "***iperf***", etc.). Very attenuated signals will be in the high 80's or 90's (see "***RSSI***" value). If close to an Access Point, the returned "***RSSI***" value should be between -30 and -50 dBm for a properly configured setup.

7.1.3.1 Using "iw" Linux Command

"***iw***" is the default Linux command line tool for controlling a WLAN interface. It provides an alternative to "***wl***" tool. One useful link to learn more about "***iw***" is on the "**Linux Wireless wiki**" here: https://wireless.wiki.kernel.org/en/users/documentation/iw. In the following example of listing WLAN devices and performing a scan, one active AP has SSID of "***Murata_5G***". Here are expected results:

```
$ iw dev  ← list available WLAN devices
phy#0
        Interface wlan0
                ifindex 6
                wdev 0x1
                addr b8:d7:af:56:61:fc
                type managed
                channel 36 (5180 MHz), width: 20 MHz, center1: 5180 MHz
                txpower 31.00 dBm


$iw dev wlan0 scan  ← perform scan on “wlan0” interface

BSS 84:1b:5e:f6:a7:60(on wlan0)
        TSF: 0 usec (0d, 00:00:00)
        freq: 5180
        beacon interval: 100 TUs
        capability: ESS (0x0001)
        signal: -41.00 dBm
        last seen: 0 ms ago
        SSID: Murata_5G
        Supported rates: 6.0* 9.0 12.0* 18.0 24.0* 36.0 48.0 54.0
        BSS Load:
                * station count: 0
                * channel utilisation: 3/255
                * available admission capacity: 0 [*32us]
        HT capabilities:
                Capabilities: 0x96f
                        RX LDPC
                        HT20/HT40
                        SM Power Save disabled
                        RX HT20 SGI
                        RX HT40 SGI
                        RX STBC 1-stream
                        Max AMSDU length: 7935 bytes
                        No DSSS/CCK HT40
                Maximum RX AMPDU length 65535 bytes (exponent: 0x003)
                Minimum RX AMPDU time spacing: 4 usec (0x05)
                HT RX MCS rate indexes supported: 0-23
                HT TX MCS rate indexes are undefined
        HT operation:
                * primary channel: 36
                * secondary channel offset: above
                * STA channel width: any
                * RIFS: 1
                * HT protection: no
```

```
        * non-GF present: 0
        * OBSS non-GF present: 0
        * dual beacon: 0
        * dual CTS protection: 0
        * STBC beacon: 0
        * L-SIG TXOP Prot: 0
        * PCO active: 0
        * PCO phase: 0
    Extended capabilities: BSS Transition, 6
    VHT capabilities:
        VHT Capabilities (0x0f825932):
            Max MPDU length: 11454
            Supported Channel Width: neither 160 nor 80+80
            RX LDPC
            short GI (80 MHz)
            SU Beamformer
            SU Beamformee
        VHT RX MCS set:
            1 streams: MCS 0-9
            2 streams: MCS 0-9
            3 streams: MCS 0-9
            4 streams: not supported
            5 streams: not supported
            6 streams: not supported
            7 streams: not supported
            8 streams: not supported
        VHT RX highest supported: 0 Mbps
        VHT TX MCS set:
            1 streams: MCS 0-9
            2 streams: MCS 0-9
            3 streams: MCS 0-9
            4 streams: not supported
            5 streams: not supported
            6 streams: not supported
            7 streams: not supported
            8 streams: not supported
        VHT TX highest supported: 0 Mbps
    VHT operation:
        * channel width: 1 (80 MHz)
        * center freq segment 1: 42
        * center freq segment 2: 0
        * VHT basic MCS set: 0x0000
    WPS:    * Version: 1.0
        * Wi-Fi Protected Setup State: 2 (Configured)
        * Response Type: 3 (AP)
        * UUID: 1b52c4d5-ffb1-0ad1-63f3-9b91a979382c
```

```
        * Manufacturer: NETGEAR, Inc.
        * Model: R6300
        * Model Number: R6300
        * Serial Number: 4536
        * Primary Device Type: 6-0050f204-1
        * Device name: R6300
        * Config methods: Display
        * RF Bands: 0x3
        * Unknown TLV (0x1049, 6 bytes): 00 37 2a 00 01 20
    WMM:    * Parameter version 1
        * u-APSD
        * BE: CW 15-1023, AIFSN 3
        * BK: CW 15-1023, AIFSN 7
        * VI: CW 7-15, AIFSN 2, TXOP 6016 usec
        * VO: CW 3-7, AIFSN 2, TXOP 3264 usec…… etc.  ← More SSID listings follow here.
```

7.1.3.2 Using "wl" Tool

The Cypress "***wl***" tool is a powerful tool which allows the user to configure any number of radio characteristics. It is most commonly used for RF testing/evaluation. However, it is also convenient to do initial bring-up testing. The "***wl***" tool is integrated into the Murata-customized i.MX image and provides a quick way to check/verify functionality. For more information on "***wl***" tool please reference the following documents:

- "WL Tool Instructions" on "My Murata->Common Documents"
- "WL Tool for Embedded 802.11 Systems" on Cypress Linux Support Portal.

Now that Wi-Fi interface is up and running (having loaded the default "***brcmfmac<chipset>-sdio.bin***" – STA/Client mode firmware), let's do some basic testing to verify functionality. The "***wl scan***" command will initiate an active probe of all visible SSID's. The "***wl scanresults***" will return a list of visible SSID's.

In following example, one active AP has SSID of "***Murata_5G***". Here are expected results:

```
$ wl scan
$ wl scanresults
brcmfmac: brcmf_cfg80211_vndr_cmds_dcmd_handler: oversize return buffer 130048
SSID: "Murata_5G"  ← Broadcast SSID, with RSSI (received signal strength) one line below
Mode: Managed   RSSI: -38 dBm   SNR: 0 dB      noise: 0 dBm    Flags: RSSI on-channel  Channel: 36/80
BSSID: 84:1B:5E:F6:A7:60       Capability: ESS
Supported Rates: [ 6(b) 9 12(b) 18 24(b) 36 48 54 ]
Extended Capabilities: BSS_Transition
VHT Capable:
      Chanspec: 5GHz channel 42 80MHz (0xe02a)
      Primary channel: 36     ← Channel Number
      HT Capabilities: 40Mhz SGI20 SGI40  ← 802.11ac bandwidth (40 MHz in this case)
```

```
        Supported HT MCS : 0-23
        Supported VHT MCS:
                NSS1 Tx: 0-9   Rx: 0-9
                NSS2 Tx: 0-9   Rx: 0-9
                NSS3 Tx: 0-9   Rx: 0-9
WPS: V2.0 Configured
VS_IE:dd7c0050f204104a000110104400010210 3b000103104700101b52c4d5ffb10ad163f39b91a979382c102
1000d4e4554474541522c20496e632e1023000552363330301024000552363330301042000434353336105400
00800060050f2040001101100055236333030100800022008103c000103104900060037 2a000120
VS_IE:dd090010180200001c0000
VS_IE:dd180050f20201018c0003a4000027a400004243bc0062326600

…… etc.  ← More SSID listings follow here.
```

NOTE: "***oversize return buffer***" log message (right after "***wl scanresults***" command) can be ignored.

7.1.4 STA/Client Mode: Connecting to Unsecured Access Point or Wireless Router

NOTE: In the following test sequences of using "***wl***" tool or "***iw***" command, the WLAN interface is not assigned an IP address. That is done later in **Section 7.1.6** where connectivity testing is performed.

7.1.4.1 Using "iw" Linux Command

Following example with "***Murata_5G***" SSID, now invoke "***iw***" connect command:

```
$ iw dev wlan0 connect Murata_5G
```

Check status of connection with "***iw***" link command:

```
$ iw dev wlan0 link
Connected to 84:1b:5e:f6:a7:60 (on wlan0)
        SSID: Murata_5G
        freq: 5180
        RX: 1944 bytes (8 packets)
        TX: 0 bytes (0 packets)
        signal: -44 dBm
        bss flags:
        dtim period:    2
        beacon int:     100
```

7.1.4.2 Using "wl" Tool

To verify connectivity quickly, associating to an unsecured Access Point is a quick test. Here is the syntax for the "***wl join***" command:

```
$ wl join
join    Join a specified network SSID.
        Usage: join <ssid> [key <0-3>:xxxxx] [imode bss|ibss] [amode
open|shared|openshared|wpa|wpapsk|wpa2|wpa2psk|wpanone] [options]
        Options:
        -b MAC, --bssid=MAC     BSSID (xx:xx:xx:xx:xx:xx) to scan and join
        -c CL, --chanspecs=CL   chanspecs (comma or space separated list)
        prescanned      uses channel and bssid list from scanresults
        -p, -passive: force passive assoc scan (useful for P2P)
```

Following example with "***Murata_5G***" SSID, now invoke "***wl join***":

```
$ wl join Murata_5G  ← connect to "Murata_5G" Access Point
IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
```

Check status of connection with "***wl assoc***" command:

```
$ wl assoc  ← check association status
SSID: "Murata_5G"
Mode: Managed   RSSI: -44 dBm   SNR: 0 dB       noise: -91 dBm  Flags: RSSI on-channel  Channel: 36/80
BSSID: 84:1B:5E:F6:A7:60        Capability: ESS
Supported Rates: [ 6(b) 9 12(b) 18 24(b) 36 48 54 ]
Extended Capabilities: BSS_Transition
VHT Capable:
        Chanspec: 5GHz channel 42 80MHz (0xe02a)
        Primary channel: 36
        HT Capabilities: 40Mhz SGI20 SGI40
        Supported HT MCS : 0-7
        Supported VHT MCS:
                NSS1 Tx: 0-9   Rx: 0-9
                NSS2 Tx: 0-9   Rx: 0-9
                NSS3 Tx: 0-9   Rx: 0-9
WPS: V2.0 Configured
VS_IE:dd310050f204104a000110104400010210470010 1b52c4d5ffb10ad163f39b91a979382c103c000103104
9000600372a000120
VS_IE:dd090010180200001c0000
VS_IE:dd180050f20201018c0003a4000027a400004243bc0062326600
VS_IE:dd7c0050f204104a000110104400010210 3b000103104700101b52c4d5ffb10ad163f39b91a979382c102
1000d4e4554474541522c20496e632e10230005523633303010240005523633303010420004343533361054 0
00800060050f204000110110005523633303010080002200 8103c000103104900060037 2a000120
```

7.1.5 STA/Client Mode: Connecting to Secured Access Point or Wireless Router (WPA2-PSK)

In this section, we will cover two different approaches to accomplish STA/Client association to a WPA2-PSK secured Access Point. One is using the embedded "***wpa_cli***" tool, the other is configuring the "***/etc/wpa_supplicant.conf***" file.

Important Dependencies:

- WLAN device must be configured correctly (refer to **Section 7.1.2**).
- WPA supplicant must be up and running.

7.1.5.1 Using "wpa_cli" Command

"***wpa_cli***" can only be invoked once the "***wlan0***" interface is configured and the WPA supplicant is running. The user can follow this command sequence to establish a secure client connection to an Access Point with WPA2-PSK authentication.

Prior to running "***wpa_cli***", you might like to back up default/previous "***/etc/wpa_supplicant.conf***" file in case you overwrite it:

```
$ cp /etc/wpa_supplicant.conf  /etc/wpa_supplicant.conf.bak
```

To make sure WPA supplicant process is in a "known state", kill and re-start it:

```
$ killall wpa_supplicant
wpa_supplicant: no process found
$ wpa_supplicant -i wlan0 -D nl80211 -c /etc/wpa_supplicant.conf -B
Successfully initialized wpa_supplicant
rfkill: Cannot open RFKILL control device
rfkill: Cannot get wiphy information
```

Now invoke "***wpa_cli***" which brings up the tool in interactive mode:

```
$ wpa_cli -i wlan0
wpa_cli v2.6
Copyright (c) 2004-2016, Jouni Malinen <j@w1.fi> and contributors
This software may be distributed under the terms of the BSD license.
See README for more details.
Interactive mode
```

Following previous example, we configure the "***Murata_5G***" AP with WPA2-PSK security and associate to it using "wpa_cli" tool with following commands:

```
> remove_network all   ← tear down any existing network connections
OK
<3>CTRL-EVENT-DISCONNECTED bssid=b0:00:b4:65:e0:60 reason=3 locally_generated=1
> status                ← check status
wpa_state=INACTIVE
p2p_device_address=ba:d7:af:56:61:fc
address=b8:d7:af:56:61:fc
uuid=a3178764-c106-57ee-8ec2-6bf2e0607166
> scan                  ← initiate a scan
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>CTRL-EVENT-NETWORK-NOT-FOUND
> scan_results          ← list results of scan
bssid / frequency / signal level / flags / ssid
84:1b:5e:f6:a7:60    5180   -38   [WPA2-PSK-CCMP][WPS][ESS]    Murata_5G
84:1b:5e:f6:a7:61    2412   -36   [WPA2-PSK-CCMP][WPS][ESS]    Murata_2G
…
> add_network   ← add a network. This returns integer value which is then used for setting parameters.
0
> set_network 0 ssid "Murata_5G"          ← set SSID to “Murata_5G”
OK
> set_network 0 psk "your_passphrase"     ← set WPA passphrase
OK
> enable 0  ← enable network connection. If ssid and passphrase set correctly, connection will be established.
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>Trying to associate with SSID 'Murata_5G'
<3>Associated with 84:1b:5e:f6:a7:60     ← connection established
<3>CTRL-EVENT-CONNECTED - Connection to 84:1b:5e:f6:a7:60 completed [id=0 id_str=]
<3>CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
 > status              ← now verify that connection is established
bssid=84:1b:5e:f6:a7:60
freq=5180
ssid=Murata_5G
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
p2p_device_address=ba:d7:af:56:61:fc
address=b8:d7:af:56:61:fc
```

```
uuid=a3178764-c106-57ee-8ec2-6bf2e0607166
> save_config      ← save current configuration: this overwrites "/etc/wpa_supplicant.conf" file!
OK
> quit             ← exit wpa_cli interactive mode
```

Now let's check contents of "/etc/wpa_supplicant.conf" file:

```
$ more /etc/wpa_supplicant.conf

ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
	ssid="Murata_5G"
	psk="your_passphrase"
}
```

NOTE: Using "***save_config***" command in "***wpa_cli***" interactive mode allows us to easily generate the "***/etc/wpa_supplicant.conf***" file for a specific/desired configuration.

7.1.5.2 Using "wpa_supplicant.conf" file

Another approach to establishing a WPA2-PSK secure connection is to properly configure the "/etc/wpa_supplicant.conf" file and let the wpa_supplicant establish the connection. The default content of "/etc/wpa_supplicant.conf" file is:

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
	key_mgmt=NONE
}
```

With the default configuration, the WPA supplicant will establish a connection with any random Access Point that has no authentication scheme enabled (i.e. "open"). Using "***Murata_5G***" SSID example, the relevant/modified contents of the "***/etc/wpa_supplicant.conf***" file (already shown in **Section 7.1.5.1)** is:

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
	ssid="Murata_5G"
	psk="your_passphrase"
}
```

To establish a secured WPA2-PSK connection by only modifying "***/etc/wpa_supplicant.conf***" file, we need to follow these steps:

- Modify "***/etc/wpa_supplicant.conf"*** file to configure desired connection.
- Kill WPA supplicant process and re-start it.
- Re-started WPA supplicant will read in modified configuration file and associate to AP (Wireless Router) accordingly.

Expected output when killing and re-starting the WPA supplicant process:

```
$ killall wpa_supplicant
$ wpa_supplicant -i wlan0 -D nl80211 -c /etc/wpa_supplicant.conf -B
Successfully initialized wpa_supplicant
rfkill: Cannot open RFKILL control device
rfkill: Cannot get wiphy information
IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
```

Verify that connection is re-established with Access Point:

```
$ iw dev wlan0 link
Connected to 84:1b:5e:f6:a7:60 (on wlan0)
        SSID: Murata_5G
        freq: 5180
        RX: 1659 bytes (7 packets)
        TX: 264 bytes (2 packets)
        signal: -45 dBm
        tx bitrate: 24.0 MBit/s

        bss flags:
        dtim period:    2
        beacon int:     100
```

7.1.6 **STA/Client Mode: Basic WLAN Connectivity Testing**

Prior to running connectivity tests, we need to assign an IP address to the "***wlan0***" interface. If the subnet address is known, one option is to use manual "***ifconfig***" command to assign an IP address to "***wlan0***". Here is an example "***ifconfig***" command assuming subnet of 192.168.1.255:

```
$ ifconfig wlan0 192.168.1.111 netmask 255.255.255.0
```

Alternatively, we can use the DHCP client to obtain an address (assuming wireless network associated to has a DHCP server):

```
$ udhcpc -i wlan0   ← Command to invoke DHCP client and obtain IP address.
udhcpc (v1.23.1) started
Sending discover...
Sending select for 192.168.1.100...
Lease of 192.168.1.100 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 192.168.1.1
```

The most basic connectivity test is to use the "***ping***" command. In this example, we assume the wireless router (associated to) has an IP address of 192.168.1.1:

```
$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=12.686 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=10.227 ms
64 bytes from 192.168.1.1: seq=2 ttl=64 time=10.053 ms
64 bytes from 192.168.1.1: seq=3 ttl=64 time=10.341 ms
64 bytes from 192.168.1.1: seq=4 ttl=64 time=12.364 ms
^C      ← Enter <CTRL-C> to terminate ping session
--- 192.168.1.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss   ← Indicates that no packets were dropped
round-trip min/avg/max = 10.053/11.134/12.686 ms
```

If we want to do more sophisticated connectivity tests, the **"iperf3"** tool is available in the i.MX image. To run throughput performance tests with "***iperf3***" you need at least one client and one server. Typically, the user will install the "***iperf3***" utility on a Windows or Linux PC which is wired to the associated wireless router. For more information on the "***iperf3***" tool refer to this link: https://iperf.fr/.

7.1.7 Wi-Fi Direct Testing

In this section we use the "***wpa_cli***" tool to configure the i.MX6/Murata Wi-Fi platform as a P2P Group Owner (P2P GO). Another device (i.e. another i.MX platform or smartphone) is required to act as the P2P Client. Together the P2P GO and Client devices establish a P2P Group.

Important Dependencies:

- WLAN device must be configured correctly (refer to **Section 7.1.2**).
- WPA supplicant must be up and running.

Prior to running "***wpa_cli***", you might like to back up default/previous "***/etc/wpa_supplicant.conf***" file in case you overwrite it:

```
$ cp /etc/wpa_supplicant.conf  /etc/wpa_supplicant.conf.bak
```

Now we can invoke "***wpa_cli***" tool to configure the P2P interface:

```
$ wpa_cli -i wlan0
> remove_network all        ← Let's remove any network association
OK
<3>CTRL-EVENT-DISCONNECTED bssid=84:1b:5e:f6:a7:60 reason=3 locally_generated=1
> > status                  ← Check status now
wpa_state=INACTIVE
ip_address=192.168.1.3
p2p_device_address=ba:d7:af:56:61:fc
address=b8:d7:af:56:61:fc
uuid=a3178764-c106-57ee-8ec2-6bf2e0607166
> p2p_group_add             ← Add P2P Group
IPv6: ADDRCONF(NETDEV_UP): p2p-wlan0-0: link is not ready
OK
<3>P2P-GROUP-STARTED p2p-wlan0-0IPv6: ADDRCONF(NETDEV_CHANGE): p2p-wlan0-0: link becomes ready
 GO ssid="DIRECT-Ih" freq=2437 passphrase="sJjJ4JUR" go_dev_addr=ba:d7:af:56:61:fc
> > quit          ← Quit "wpa_cli" tool
```

After running "***p2p_group_add***" command, the following are set:

- P2P virtual interface (see results of "***ifconfig***" command below)
- P2P SSID, with selected channel and secure passphrase needed by another P2P client to associate.

To verify new virtual P2P interface, we just invoke "ifconfig" command:

```
$ ifconfig
...
p2p-wlan0-0 Link encap:Ethernet  HWaddr ba:d7:af:56:e1:fc      ← new P2P interface
        inet6 addr: fe80::b8d7:afff:fe56:e1fc/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:23 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B)  TX bytes:4525 (4.4 KiB)
wlan0     Link encap:Ethernet  HWaddr b8:d7:af:56:61:fc      ← existing "wlan0" interface
        inet addr:192.168.1.3  Bcast:192.168. 1.255  Mask:255.255.255.0
        UP BROADCAST MULTICAST  MTU:1500  Metric:1
        RX packets:1903 errors:0 dropped:0 overruns:0 frame:0
        TX packets:769 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:367182 (358.5 KiB)  TX bytes:76384 (74.5 KiB)
```

To test connectivity, we can assign manual IP address to P2P interface:

```
$ ifconfig p2p-wlan0-0 192.168.2.1 netmask 255.255.255.0
```

Now connect a P2P Client such as smartphone (or similar) and force same IP address with same subnet address. We can now ping from either interface.

7.1.8 Soft AP or Wi-Fi Hot Spot Testing

In this section we use the "***hostapd***" supplicant to configure the i.MX/Murata Wi-Fi platform as a "Soft AP" or "Wi-Fi hot spot". Both unsecured and secured configurations are presented.

Important Dependencies:

- "***hostapd***", "***hostapd.conf***", and "***udhcpd.conf***" files are present in file system: these are part of standard Murata i.MX customized release.

First off, we need to kill the WPA supplicant:

```
$ killall wpa_supplicant
```

Using the default settings in "***hostapd.conf***" file, the configuration is setup for no authentication. We can start up the SoftAP with following commands:

```
$ ifconfig wlan0 192.168.1.1 up
$ udhcpd -S -I 192.168.1.1 /etc/udhcpd.conf
$ hostapd -B /etc/hostapd.conf
```

After invoking the "hostapd" command, the following log is expected:

```
Configuration file: /etc/hostapd.conf
Failed to create interface mon.wlan0: -95 (Operation not supported)
rfkill: Cannot open RFKILL control device
wlan0: Could not connect to kernel driver
Using interface wlan0 with hwaddr b8:d7:af:56:61:fc and ssid "test"
wlan0: interface state UNINITIALIIPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
ZED->ENABLED
wlan0: AP-ENABLED
```

We can now associate from another client device and ping the "wlan0" interface (in this example 192.168.1.1). For authenticating in secure fashion, just change the "/etc/hostapd.conf" file to uncomment/configure the "wpa" and "wpa_passphrase" lines correctly:

```
"#wpa=1" ➔ "wpa=1"
"#wpa_passphrase=secret passphrase" ➔ "wpa_passphrase=password123"
```

7.1.9 WLAN Manufacturing or RF Testing

Running manufacturing, RF, or regulatory testing is straightforward with the "***fmac***" driver. The only necessary step is to switch over to manufacturing test firmware and reboot the platform. The "***production***" version of WLAN firmware is used on default image. To switch over to manufacturing test firmware, the user needs to contact Murata or Cypress to obtain the manufacturing test firmware files. Here are the necessary steps:

- Obtain manufacturing test firmware tarball. If you need assistance then email Murata support at: wirelessFAQ@murata.com.
- Mount flashed (micro) SD card on Linux host PC.
- "***cd***" to "***lib/firmware/brcm***" folder.
- Switch user to "***root***".
- Create "***mfgtest***" and "***production***" sub-folders in "***/lib/firmware/brcm***" folder.
- Backup existing "****.bin***" and ***"*.clm_blob***" files to "***production***" sub-folder.
- Copy manufacturing test firmware files (from "mfgtest" sub-folder) into "***lib/firmware/brcm***" folder.
- Insert (micro) SD card on i.MX platform and boot.
- Verify that "***WLTEST***" is logged when "***fmac***" driver loads.

On host, insert (micro) SD card and configure sub-folder for production and manufacturing test firmware:

```
$ cd <path to mounted rootfs>/lib/firmware/brcm
$ sudo mkdir mfgtest
$ sudo mkdir production
$ sudo cp *.bin production/
$ sudo cp *.clm_blob production/
$ sudo cp <path to mfgtest binaries>/*.bin mfgtest/
$ sudo cp <path to mfgtest binaries>/*.clm_blob mfgtest/
```

On i.MX Target, insert modified (micro) SD card, boot platform and note firmware log message:

```
brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: Sep 21 2018 04:08:34 version 7.45.173 (r707987 CY) FWID 01-d2799ea2
```

Now copy over "***mfgtest***" sub-folder contents to "***/lib/firmware/brcm***":

```
$ cd /lib/firmware/brcm
$ cp mfgtest/* .
```

Reboot platform and note different log message (with "***WLTEST***" string) for manufacturing test firmware:

```
brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: Sep 21 2018 04:02:19 version 7.45.173 (r707987 CY WLTEST) FWID 01-3c82dde4
```

NOTE: the version number of firmware for production and manufacturing test **should not differ**. In this example, both firmware versions are "***7.45.173***".

Before running any manufacturing tests with "***wl***" tool, make sure that WPA supplicant is not running:

```
$ killall wpa_supplicant
```

Now you can run RF testing. ***Note*** that manufacturing test firmware ***does not support*** some interoperability modes that production firmware does. The manufacturing test firmware is a specific release and is intended ***only to be used*** for RF testing.

Once RF testing is done, you can easily revert to the normal production firmware:

```
$ cd /lib/firmware/brcm
$ cp production/* .
```

7.2 Bluetooth Interface Test/Verification

For Murata modules supporting Bluetooth, we can verify the HCI UART connection by invoking "***hciattach***", bringing up the interface with "***hciconfig***" and then invoking "***hcitool scan***" to see what Bluetooth devices are visible. The Bluetooth test commands vary depending on which UART port is connected to Bluetooth. With the new "***modem_reset***" construct in DTS(I) files, the Bluetooth core should come up in the correct state to be initialized (i.e. as kernel boots, the Bluetooth core is reset and is taken out of reset). When the BlueZ "***hciattach***" call is invoked, a Bluetooth patch file is pulled from the "/etc/firmware/" folder – refer to **Table 9** for more details. Here is the default command sequence to verify Bluetooth functionality:

```
hciattach /dev/ttymxc[UART# -1]  bcm43xx 3000000 flow -t 20
hciconfig hci0 up
hcitool scan
```

NOTE: if the platform is correctly configured, then the user should ONLY have to execute "***hciattach***" command followed by "***hciconfig***", and "***hcitool***".

Table 10 lists the BT_REG_ON GPIO and UART ports for the various i.MX platforms. Here is example output using i.MX 6ULL EVK with Type 1MW module (no BT_REG_ON toggle):

```
$ hciattach /dev/ttymxc1 bcm43xx 3000000 flow -t 20
bcm43xx_init
Set Controller UART speed to 3000000 bit/s
Flash firmware /etc/firmware/BCM4345C0.1MW.hcd
Set Controller UART speed to 3000000 bit/s
Setting TTY to N_HCI line discipline
Device setup complete
$ hciconfig hci0 up
$ hcitool scan
Scanning ...
        34:F3:9A:A6:00:53       SCOTTK-HPZBOOK
```

Table 10: GPIO and UART Settings for Bluetooth Tests

i.MX6 Platform	GPIO/UART Configuration	Notes
i.MX 8MQuad EVK	GPIO69; UART3	
i.MX 6Q(P)/DL SDB/SDP	GPIO2; UART5	
i.MX 6SoloX SDB	GPIO171; UART5	
i.MX 6UL/ULL EVK	GPIO508; UART2	GPIO508 does not allow its direction to be set. Always output.

If there is an issue with the BT core not being reset correctly (after kernel boot), then the corresponding DTS(I) file can be modified to disable the “modem_reset” construct. Typically, this should **NOT** be necessary. Once the ***modified*** DTB file is loaded at kernel boot time, the command sequence to toggle BT reset/enable line and verify BT functionality is now:

```
echo [GPIO #] > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio[GPIO #]/direction      ← SKIP this step for i.MX 6UL/ULL EVK
echo 0 > /sys/class/gpio/gpio[GPIO #]/value
sleep 0.1
echo 1 > /sys/class/gpio/gpio[GPIO#]/value
hciattach /dev/ttymxc[UART# -1] bcm43xx 3000000 flow -t 20
hciconfig hci0 up
hcitool scan
```

Regarding some more details on modifying DTS(I) files to obtain complete control over BT_REG_ON signal, code excerpt below is from i.MX 8MQuad DTS file (“fsl-imx8mq-evk.dts” which is included by “fsl-imx8mq-evk-pcie1-m2.dtb”). Just comment out the “modem_reset” line (in BT UART descriptor) and recompile the DTS files. Refer to Linux User Manual for specifics on changing code and compiling.

```
&uart3 { /* BT */
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_uart3>;
    assigned-clocks = <&clk IMX8MQ_CLK_UART3_SRC>;
    assigned-clock-parents = <&clk IMX8MQ_SYS1_PLL_80M>;
    fsl,uart-has-rtscts;
//  resets = <&modem_reset>;   ← Comment out “resets” line on BT UART descriptor
    status = "okay";
};
```

8 uSD-M.2 Adapter High-Level Description

Figure 13: uSD-M.2 Adapter Features (Top View) and **Figure 14: uSD-M.2 Adapter Features (Bottom View)** show the features on the uSD-M.2 Adapter; with text explanation in **Table 11.** The uSD-M.2 Adapter supports additional signals (for i.MX 6) to WLAN-SDIO using the 20 pin FFC connector (J6). For more details on the uSD-M.2 Adapter, refer to the Hardware User Manual.

Table 11: uSD-M.2 Adapter Features

A	microSD connector provides Power (VBAT, GND) and WLAN-SDIO
B	SDIO bus test points (CLK, CMD, DAT0, DAT1, DAT2, DAT3)
C	Power LED Indicator (green): if not illuminated then no power applied to M.2 EVB
D	J11 = Optional BT Disable Jumper for WLAN-Only Mode (close this jumper to drive BT_REG_ON low and disable Bluetooth Core; thereby optimizing power consumption for WLAN-Only Mode)
E	J9 = BT UART TX/RX and WLAN/BT CTRL Arduino Header
F	J5 = Optional BT PCM and WLAN/BT Debug Signals
G	Threaded mount for M.2 screw: 30mm distance from M.2 connector
H	J7 = Optional Arduino Header Power Supply (can connect either 5V or 3.3V VBAT)
I	J8 = BT UART RTS/CTS Arduino Header
J	J12 = VDDIO Override: Short for 3.3V VDDIO; Open for 1.8V (default)
K	LED2 = 3.3V VDDIO Override Indicator (Blue)
L	J2 = Optional 5V USB Power Supply via Micro-AB USB Connector
M	J1 = Power Supply Selector Jumper must be installed to power Adapter (unless J5 Arduino Header Pins #15/16 are connected to external GND/3.3V VBAT). **Position 1-2:** 5V/3.3V VBAT supply from micro-USB (J2) or Arduino (J7); **Position 2-3:** VBAT supply (typical 3.1~3.3V) from microSD connector
N	Regulator to step down optional 5V VBAT from USB or Arduino header to 3.3V
O	M.2 Connector: type 2230-xx-E
P	microSD connector pins: provides Power and WLAN-SDIO
Q	WLAN JTAG header (header pins not populated)
R	20 pin FFC connector (BT UART, BT PCM, WLAN/BT Control signals)
S	Additional test points from 20pin flat/flex connector

Figure 13: uSD-M.2 Adapter Features (Top View)

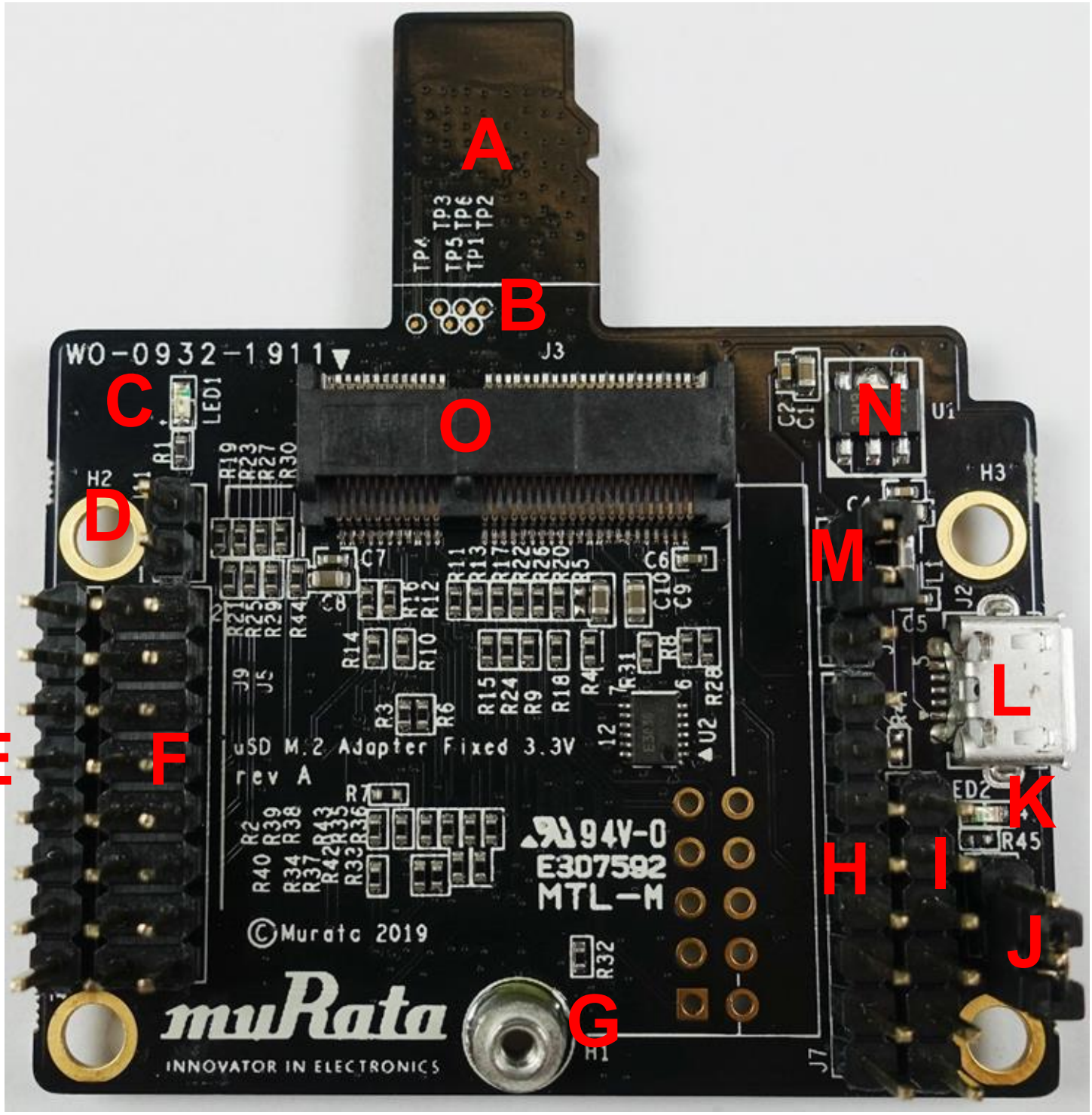
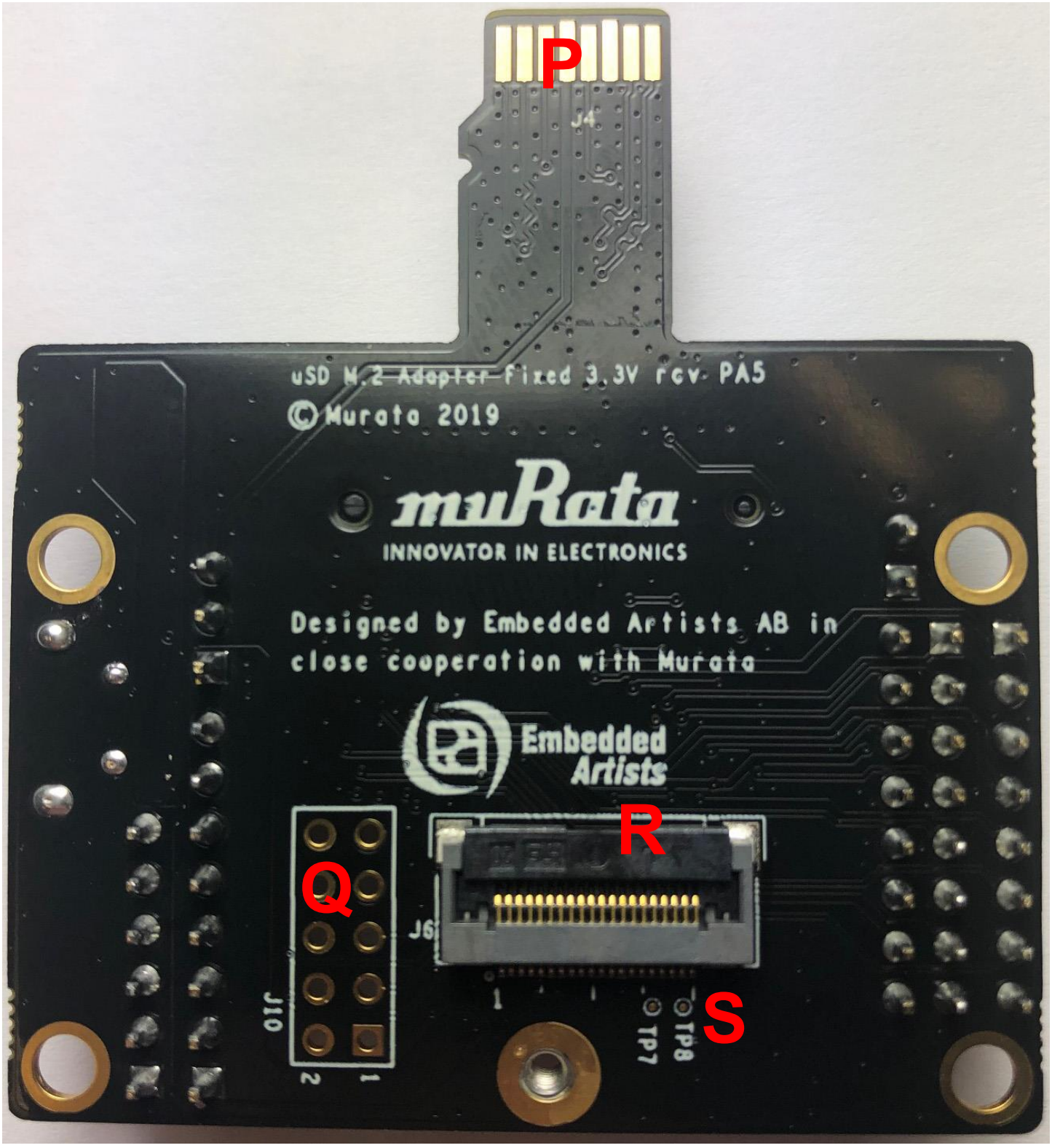


Figure 14: uSD-M.2 Adapter Features (Bottom View)



9 Embedded Artists' Wi-Fi/BT M.2 Modules

For specifics on the M.2 Wi-Fi/BT M.2 EVB's, refer to www.embeddedartists.com/m2/.

10 Technical Support Contact

Table 12 below lists all the support resources available for the Murata/Cypress i.MX Wi-Fi/BT solution. There are two dedicated Murata websites (main landing page and i.MX support portal) in addition to a dedicated imxfaq@murata.com email alias. All website/email addresses are hyperlinked in the "Support Site" column below.

Table 12: List of Support Resources

Support Site	Notes
Murata i.MX Landing Page	**No** login credentials required. This is an excellent starting point to understand all hardware/software configurations supported. Quick Start Guides and Murata Module Datasheets provided.
My Murata i.MX Support Portal	Login credentials required. More detailed Murata documentation provided: such as Linux User Manual, Hardware User Manual, RF Regulatory Test Manual, etc. For registering refer to this guide.
Murata uSD-M.2 Adapter Landing Page	Landing page for uSD-M.2 Adapter. In conjunction with Murata i.MX Landing Page, this should provide the user with comprehensive getting started documentation.
Cypress Linux Community	Login credentials required. Support for Linux wireless solutions based on Cypress chipsets. Cypress and Murata team support queries on this forum.
NXP i.MX Community	Login credentials required. Support forum for NXP i.MX Processors. NXP and Murata team support queries on this forum.
Murata i.MX FAQ Email	i.MX FAQ email. Supported by Murata Team. Typically used to support issues accessing support sites, this email address is used for questions not addressed on support forums.

11 Additional Useful Links

In addition to **Table 12** listings of support resources, **Table 13** provides some useful links.

Table 13: Additional Useful Links

Link	Notes
“iw” Command Line	“iw” is default Linux command to configure WLAN interface.
iPerf Performance Test Tool	“iPerf” test tool is built into NXP Linux BSP image.