

Wi-Fi®/Bluetooth® (CYW) for i.MX

Linux User Guide - Rev. 3.1



Table of Contents

1 Introduction	6
1.1 NXP i.MX 6 Platform Support	7
1.2 NXP i.MX 8 Platform Support	9
2 Wi-Fi/BT Hardware Solution for i.MX	11
2.1 Embedded Artists' Wi-Fi/BT M.2 EVBs	11
2.2 Murata's uSD-M.2 Adapter	13
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	16
3.3 Murata "FMAC" Customized i.MX Yocto Image Build	18
3.3.1 Install Ubuntu	18
3.3.2 Download Murata's Script Files	19
3.3.3 Configure Ubuntu for i.MX Yocto Build	19
3.3.4 Murata's i.MX Yocto Build Script	20
3.4 Additional Hardware/Software Considerations	22
3.4.1 Out-Of-Band (OOB) IRQ Support on NXP i.MX EVKs	22
3.4.2 1.8V versus 3.3V VIO Signaling using Murata's uSD-M.2 Adapter	23
3.4.3 UHS SDIO 3.0 operation on i.MX 6UL(L) EVKs with uSD-M.2 Adapter	23
3.4.4 WLAN/Bluetooth M.2 Direct Interconnect on NXP i.MX Platforms	23
3.4.5 Setting Correct Software Configuration before Testing Wi-Fi/BT Solution	24
4 Preparing NXP i.MX Platforms to Boot Linux Image	25
4.1 Flashing Murata-customized Linux Image to (micro) SD Card	25
4.1.1 Linux PC Steps to Flash SD Card	25
4.1.2 Windows PC Steps to Flash SD Card	26
4.2 Flashing Linux Image to NXP i.MX 8M Mini/Nano EVKs	27
4.2.1 Software File Preparation	27
4.2.2 i.MX 8M Mini or Nano EVK Hardware Configuration	28
4.2.3 Flash Linux Image to eMMC on i.MX 8M Mini/Nano EVK	30
4.2.4 Configure i.MX 8M Mini/Nano EVK to boot from eMMC	30
5 Murata Wi-Fi/BT Bring-Up on i.MX 6 Platforms	32
5.1 Connecting to i.MX 6Q(P)/DL SDB (3.3V WLAN-SDIO VIO Override)	34
5.1.1 Specific Hardware Considerations for i.MX 6Quad/DualLite SDB/SDP	34
5.1.2 Wi-Fi/Bluetooth Bring-Up on i.MX 6Quad/DualLite SDB/SDP	34
5.2 Connecting to i.MX 6UL EVK or i.MX 6ULL EVK (1.8V WLAN-SDIO VIO)	36

6 Murata Wi-Fi/BT Bring-Up on i.MX 8 Platforms	38
6.1 Bringing up Wi-Fi/BT on i.MX 8MQuad EVK.....	38
6.2 Bringing up Wi-Fi/BT on i.MX 8M Mini/Nano EVKs (Onboard 1MW or M.2)	39
6.3 Bringing up Wi-Fi/BT on i.MX 8M Mini/Nano EVKs (uSD-M.2 Adapter)	40
7 Test/Verification of Wi-Fi and Bluetooth	49
7.1 Wi-Fi Interface Test/Verification	50
7.1.1 Useful Environment Setup on NXP Linux	50
7.1.2 Set Module Script	50
7.1.3 Bringing Up Wi-Fi Interface.....	50
7.1.4 STA/Client Mode: Scan for Visible Access Points.....	52
7.1.5 STA/Client Mode: Connecting to Unsecured Access Point or Wireless Router	55
7.1.6 STA/Client Mode: Connecting to Secured Access Point or Wireless Router (WPA2-PSK)	56
7.1.7 STA/Client Mode: Basic WLAN Connectivity Testing.....	59
7.1.8 Wi-Fi Direct Testing	60
7.1.9 Soft AP or Wi-Fi Hot Spot Testing	61
7.1.10 WLAN Manufacturing, RF or Regulatory Testing	62
7.2 Bluetooth Interface Test/Verification.....	63
8 Murata's uSD-M.2 Adapter.....	65
8.1 Connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter	65
8.2 Configuring uSD-M.2 Adapter Jumpers for Correct VIO Signaling.....	65
8.3 Securing uSD-M.2 Adapter to NXP i.MX EVK	66
8.4 uSD-M.2 Adapter High-Level Description.....	68
9 Embedded Artists' Wi-Fi/BT M.2 Modules.....	71
10 Embedded Artists' i.MX + Wireless Solution	71
11 Useful Links	73
12 Appendix A	74
13 Acronyms.....	76
14 Technical Support Contact.....	77
15 References	78
15.1 Murata Wi-Fi/Bluetooth for i.MX Linux User Manual for CYW-based Module	78
15.2 Murata Wi-Fi/Bluetooth for i.MX Linux Quick Start Guide for CYW-based Module.....	78
15.3 Murata Wi-Fi/BT Solution for i.MX Hardware User Manual.....	78
15.4 Murata's Community Forum Support.....	78
15.5 Murata uSD-M.2 Adapter Datasheet (Rev B2)	78
15.6 Murata uSD-M.2 Adapter Datasheet (Rev B1)	78
15.7 Murata uSD-M.2 Adapter Datasheet (legacy Rev A)	78

15.8 Embedded Artists' Reference Documentation	78
15.9 Murata's i.MX Wireless Solutions Landing Page	79
15.10 NXP Reference Documentation	79
Revision History.....	81

Figures

Figure 1: i.MX 6 EVK Wi-Fi/BT Interconnect Block Diagram	8
Figure 2: i.MX 8QXP MEK or 8MQuad EVK Wi-Fi/BT PCIe Interconnect Block Diagram.....	9
Figure 3: i.MX 8MQuad EVK Wi-Fi/BT SDIO Interconnect Block Diagram	9
Figure 4: i.MX 8MPlus EVK Wi-Fi/BT Interconnect Block Diagram.....	10
Figure 5: i.MX 8M Mini EVK Wi-Fi/BT PCIe Interconnect Block Diagram	10
Figure 6: i.MX 8M Mini/Nano EVK Wi-Fi/BT SDIO Interconnect Block Diagram.....	11
Figure 7: Configuring dash.....	18
Figure 8: USB to SD Card Reader/Writer Adapter	25
Figure 9: Power, Download, and Debug port connection to board	28
Figure 10: i.MX 8M Mini EVK DIP Switches configured for Download.....	29
Figure 11: i.MX 8M Nano EVK DIP Switches configured for Download.....	29
Figure 12: i.MX 8M Mini EVK DIP Switches configured for eMMC Boot.....	31
Figure 13: i.MX 8M Nano EVK DIP Switches configured for eMMC Boot.....	31
Figure 14: uSD-M.2 Adapter with M.2 EVB options.....	33
Figure 15: i.MX 6Quad/DualLite SDB (Inverted) with uSD-M.2 Adapter and Type 1MW M.2 EVB	35
Figure 16: i.MX 6UL EVK with uSD-M.2 Adapter and Type 1LV M.2 EVB.....	37
Figure 17: i.MX 8MQuad with Type 1CX (bottom view).....	39
Figure 18: i.MX 8M Mini with Type 1XA (bottom view)	40
Figure 19: i.MX 8M Mini/Nano EVK with uSD-M.2 Adapter (WLAN Only)	40
Figure 20: i.MX 8M Mini/Nano EVK with uSD-M.2 Adapter (WLAN/Bluetooth).....	41
Figure 21: Cable connections on i.MX 8M Mini EVK (Even Number Connector View)	43
Figure 22: Cable connections on i.MX 8M Mini EVK (Odd Number Connector View).....	44
Figure 23: Cable connections on uSD-M.2 Adapter (J9 Header).....	45
Figure 24: Cable connections on uSD-M.2 Adapter (J8 Header).....	46
Figure 25: Low-Profile Jumper Wires (Digi-Key part number 1988-1178-ND)	47
Figure 26: NXP i.MX EVK with Low-Profile Jumper Wires	47
Figure 27: Additional Hex Standoff (Digi-Key part number RPC3570-ND)	48
Figure 28: Connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter	65
Figure 29: Host/M.2 IO Voltage Level Shift Options on Rev B1 Adapter	66
Figure 30: Securing uSD/SD Connection on i.MX 6 EVK.....	67
Figure 31: Securing uSD Connection on i.MX 8 EVK.....	67
Figure 32: uSD-M.2 Adapter Features (Top View)	69

Figure 33: uSD-M.2 Adapter Features (Bottom View)	70
Figure 34: Combine i.MX COM with Wi-Fi/BT M.2 EVB	71

Tables

Table 1: Document Conventions	5
Table 2: Current NXP i.MX Platforms Supported	7
Table 3: Embedded Artists' Wi-Fi/BT M.2 Modules Supported	12
Table 4: uSD-M.2 Adapter Kit Contents	13
Table 5: NXP i.MX/Murata Module Interconnect	15
Table 6: NXP i.MX EVK / Yocto (MACHINE) target / Kernel Version Matrix	17
Table 7: i.MX6/8 Targets supported by Murata	17
Table 8: Select Hardware Configurations which use eMMC Boot Configuration	25
Table 9: Files to flash i.MX 8M Mini & 8M Nano EVKs	27
Table 10: i.MX 8M Mini/Nano EVK Jumper connections to uSD-M.2 Adapter	42
Table 11: Embedded Wi-Fi/Bluetooth Files	49
Table 12: GPIO and UART Settings for Bluetooth Tests	64
Table 13: uSD-M.2 Adapter Features	68
Table 14: Embedded Artists' i.MX Interconnect	72
Table 15: Embedded Artists' Landing Pages	72
Table 16: Embedded Artists' Datasheets and Schematics	72
Table 17: Embedded Artists' User Manuals and Software	73
Table 18: Useful Links	73
Table 19: List of Support Resources	77
Table 20: Embedded Artists Documentation Listing	79
Table 21: NXP Reference Documentation Listing	80

About This Document

The document describes in detail the Wi-Fi/Bluetooth solution offered by Murata for i.MX, in terms of usage.

The document uses NXP's i.MX 6/8 series-based Evaluation Kits (EVKs) as examples and all software referenced in the document are assumed to be used on such EVKs. Custom hardware-based solutions and /or unsupported Linux versions/software may require additional modifications and are outside the scope of this document.

Audience & Purpose

This document is targeted towards system developers of NXP i.MX application processor-based solutions, running Linux operating system.

Document Conventions

Table 1 describes the document conventions.

Table 1: Document Conventions

Conventions	Description
	Warning Note Indicates very important note. Users are strongly recommended to review.
	Info Note Intended for informational purposes. Users should review.
	Menu Reference Indicates menu navigation instructions. Example: Insert → Tables → Quick Tables → Save Selection to Gallery 
	External Hyperlink This symbol indicates a hyperlink to an external document or website. Example: Embedded Artists AB  Click on the text to open the external link.
	Internal Hyperlink This symbol indicates a hyperlink within the document. Example: Introduction  Click on the text to open the link.
<code>Console input/output or code snippet</code>	Console I/O or Code Snippet This text Style denotes console input/output or a code snippet.
<code># Console I/O comment // Code snippet comment</code>	Console I/O or Code Snippet Comment This text Style denotes a console input/output or code snippet comment. <ul style="list-style-type: none"> • Console I/O comment (preceded by "#") is for informational purposes only and does not denote actual console input/output. • Code Snippet comment (preceded by "//") may exist in the original code.

1 Introduction

Murata has partnered with [NXP Semiconductors N.V.](#), [Infineon Technologies](#) 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 NXP i.MX platforms. Current NXP Linux i.MX releases supported include kernel versions:

- 5.15.32_2.0.0
- 5.10.52_2.1.0
- 5.4.47_2.2.0
- 4.14.98_2.3.0



For each kernel version, multiple Infineon "FMAC" WLAN driver versions are supported.

This document describes the following steps:

- How to build Murata's customized image and flash it to various NXP i.MX EVKs. This image enables Infineon "FMAC" WLAN driver (while disabling previous legacy drivers). It also enables/configures associated components such as WPA supplicant, Hostapd, WLAN firmware/regulatory/NVRAM files, Bluetooth patch files, etc.
- Connect and/or configure applicable Wi-Fi/BT solution for a given NXP i.MX EVK & 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 some Infineon-based Murata modules soldered down. However, in most cases the wireless solution is provided by connecting [Embedded Artists' Wi-Fi/BT M.2 EVB](#) directly to the NXP i.MX EVK; or using [Murata's uSD-M.2 Adapter](#) as an interconnect solution. Refer to **Table 2** for more details.



Some of the NXP i.MX Reference platforms have restricted hardware interfaces which do not permit additional Wi-Fi/BT interconnect options.



All the NXP i.MX 6 family EVKs (except i.MX 6UL and i.MX 6ULL) only provide 3.3V VIO signaling option. This limits the interfacing of specific Wi-Fi/BT M.2 EVBs which operate at the non-standard 3.3V VIO signaling level (i.e., currently only 1DX/1MW/2AE/2BC/2BZ/1YN M.2 EVBs support 3.3V VIO on WLAN-SDIO interface). These restricted platforms are not covered in this document.

Table 2: Current NXP i.MX Platforms Supported

NXP i.MX EVK Part number	NXP i.MX EVK	Murata Modules Supported	Interconnect
MCIMX8QXP-CPU ↗	i.MX 8QXP MEK	1XA	M.2
8MPLUSLPD4-EVK ↗	i.MX 8MPLUS EVK	1DX ¹ , 1MW ² , 1LV, 2AE, 2BC, 2BZ, 1YN, 1XA	M.2 (WLAN Only)
MCIMX8M-EVKB ↗	i.MX 8MQuad EVK	1CX ³ , 1XA 1DX, 1MW, 1LV, 2AE, 2BC, 2BZ, 1YN,	1CX Soldered down, 1XA via M.2 1DX, 1MW, 1LV, 2AE, 2BC, 2BZ, 1YN via uSD-M.2 Adapter (WLAN Only)
8MMINILPD4-EVKB ↗	i.MX 8M Mini EVK	1DX, 1MW, 1LV, 2AE, 2BC, 2BZ, 1YN, 1XA	1DX, 1MW, 1LV, 2AE, 2BC, 2BZ, 1YN via uSD-M.2 Adapter 1XA via M.2 (WLAN Only)
8MNANOD4-EVK ↗	i.MX 8M Nano EVK	1DX, 1MW, 1LV, 2AE, 2BC, 2BZ, 1YN	1MW Soldered down 1DX, 1MW, 1LV, 2AE, 2BC, 2BZ, 1YN via uSD-M.2 Adapter
MCIMX6Q-SDB ↗	i.MX 6Quad SDB	1DX, 1MW, 2AE, 2BC, 2BZ, 1YN	uSD-M.2 Adapter
MCIMX6UL-EVKB ↗	i.MX 6UL EVK	1DX, 1MW, 1LV, 2AE, 2BC, 2BZ, 1YN	uSD-M.2 Adapter
MCIMX6ULL-EVK ↗	i.MX 6ULL EVK ⁴	1DX, 1MW, 1LV, 2AE, 2BC, 2BZ, 1YN	uSD-M.2 Adapter

1.1 NXP i.MX 6 Platform Support

A high-level connection Diagram for the Murata uSD-M.2 Adapter (i.MX 6 platforms) is provided in **Figure 1**. 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 [Section 8](#) [↗](#) and [Section 9](#) [↗](#) for more details on the uSD-M.2 Adapter and Wi-Fi/BT M.2 Modules. 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., < 10 K). 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 30 mm length (u.FL connector used).
- u.FL connector for external antenna or conducted testing.

¹ This is a legacy module. Can use Type 1YN as suggested replacement.

² This is a legacy module. Can use Type 2AE/2BC as suggested replacement.

³ This is a legacy module and no longer supported. Use Type 1XA as suggested replacement.

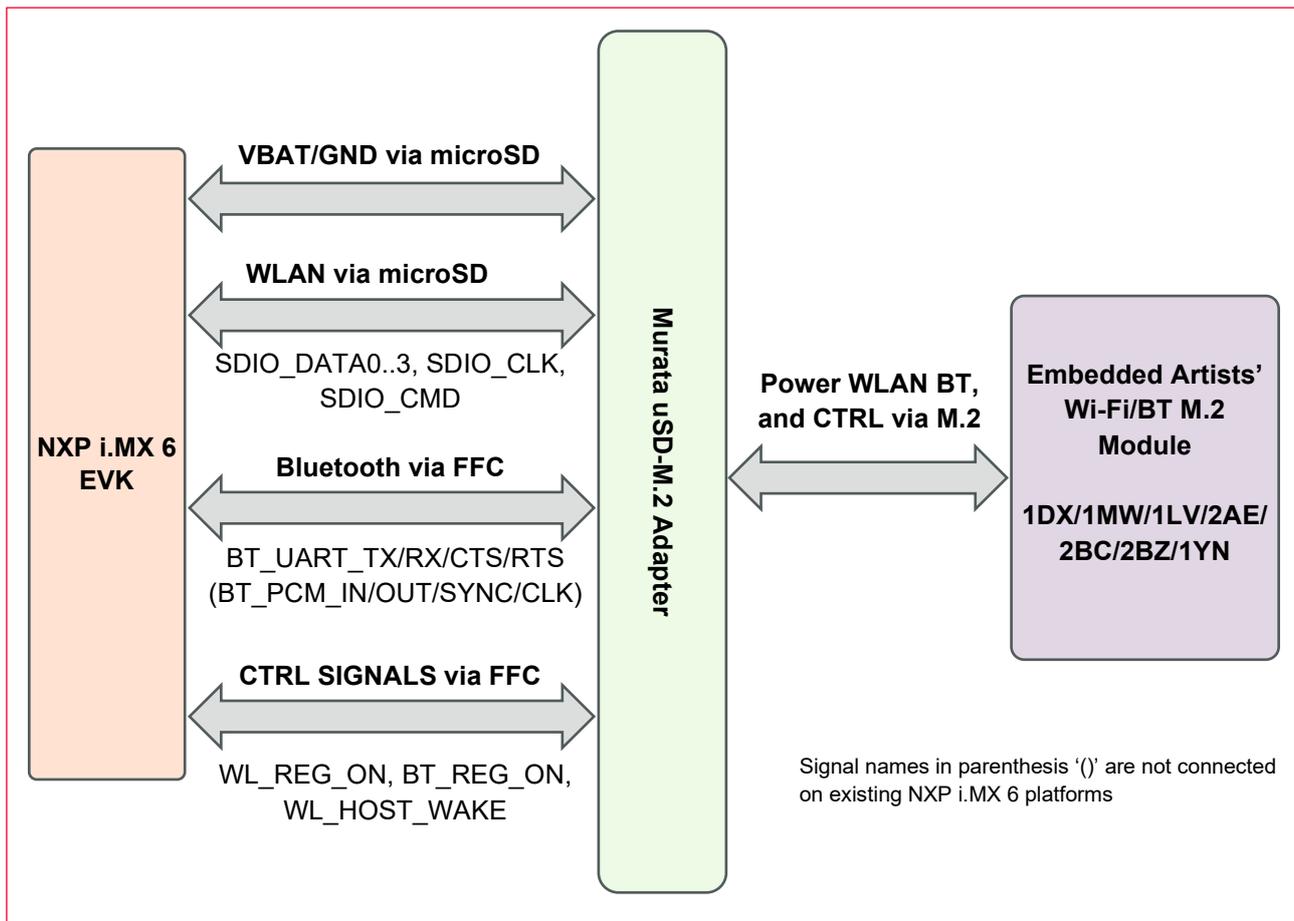
⁴ i.MX 6ULL is used for evaluation of i.MX 6ULZ.

- Comprehensive test points (including SDIO DATA, CLK, and CMD lines).



Murata no longer supports the legacy i.MX V1/V2 Interconnect Kit which used 60-pin Samtec connectors.

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 8QXP MEK/8MQuad EVK Wi-Fi/BT functionality. Currently, Type 1XA is supported with WLAN-PCIe interface.



No uSD-M.2 Adapter is used – just the Wi-Fi/BT M.2 EVB (Module).

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

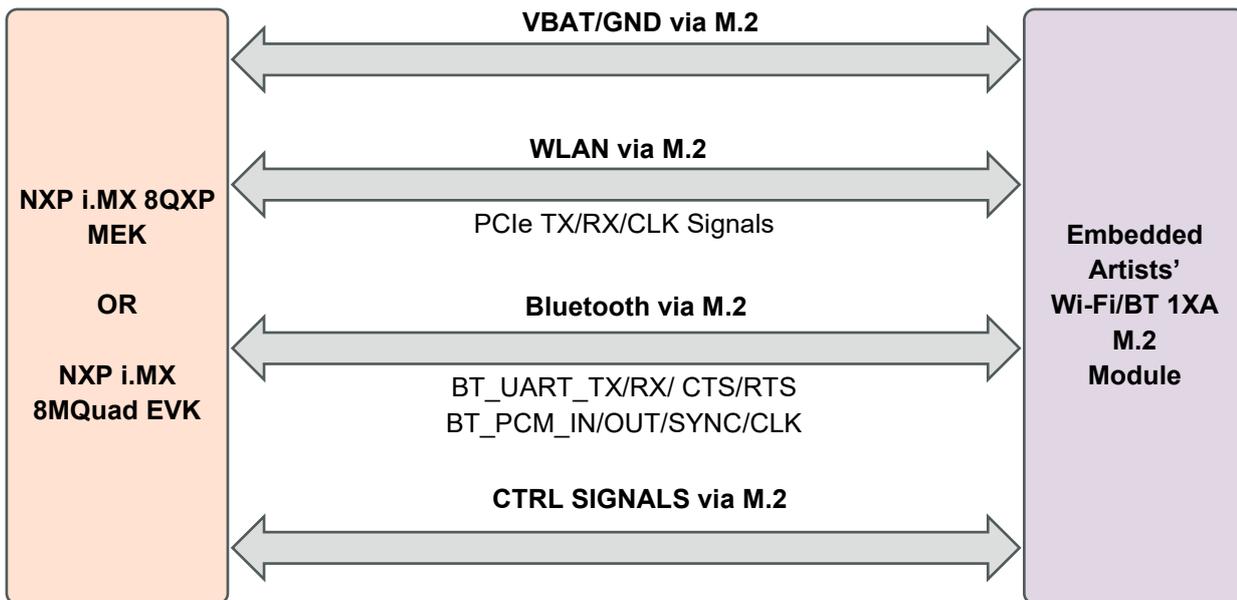


Figure 3 shows a simplified block diagram for the i.MX 8MQuad EVK Wi-Fi interconnect with Murata’s uSD-M.2 Adapter option (with Embedded Artists’ Wi-Fi/BT M.2 EVB). Only WLAN-SDIO based modules are supported in this configuration: 1MW, 1DX, 1LV, 2AE, 2BC, 2BZ and 1YN. Note that only Wi-Fi is supported in this configuration.

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

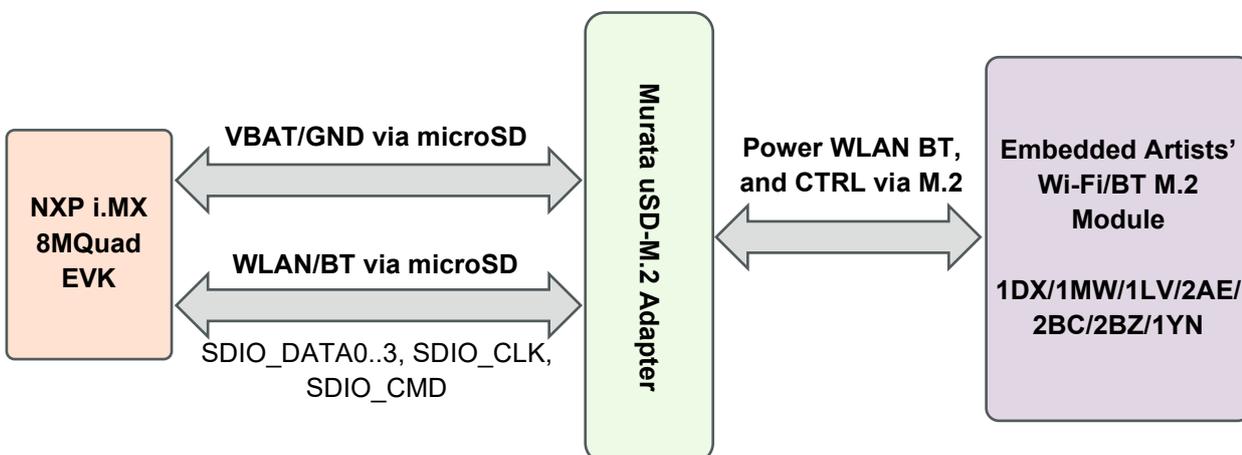


Figure 4 shows a simplified block diagram for the i.MX 8M Plus EVK Wi-Fi/BT interconnect via the M.2 slot. Only WLAN-SDIO based modules are supported in this configuration: 1MW, 1DX, 1LV, 2AE, 2BC, 2BZ and 1YN. Note that only Wi-Fi is supported in this configuration.

Figure 4: i.MX 8MPlus EVK Wi-Fi/BT Interconnect Block Diagram

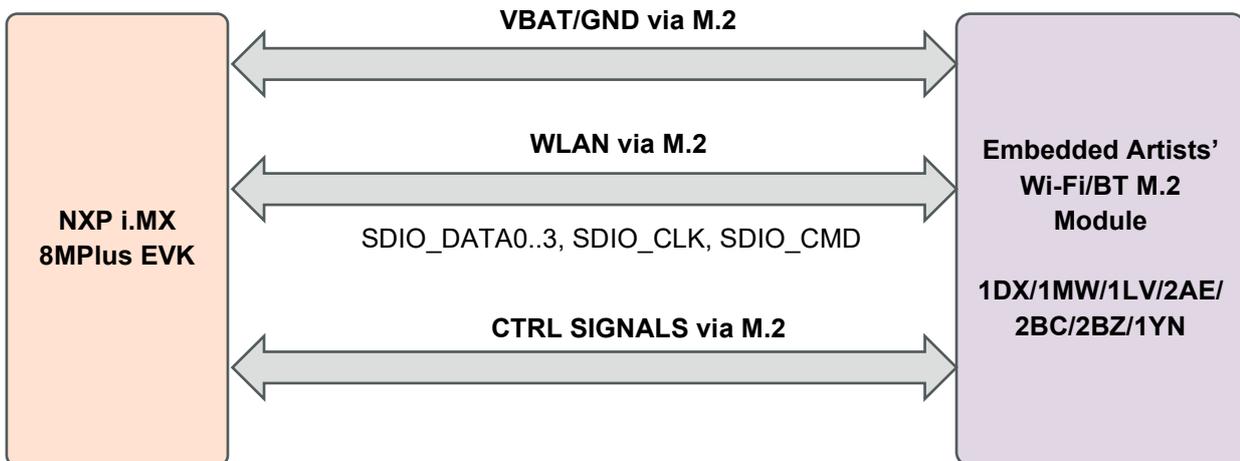


Figure 5 shows a simplified block diagram for the i.MX 8M Mini EVK Wi-Fi/BT interconnect via the M.2 slot. Currently, Type 1XA M.2 Module (WLAN Only) is supported with 2x2 802.11ac MIMO and WLAN-PCIe interface.



The NXP i.MX 8M Mini EVK does not bring out the Bluetooth signals to the M.2 connector – WLAN only.

Figure 5: i.MX 8M Mini EVK Wi-Fi/BT PCIe Interconnect Block Diagram

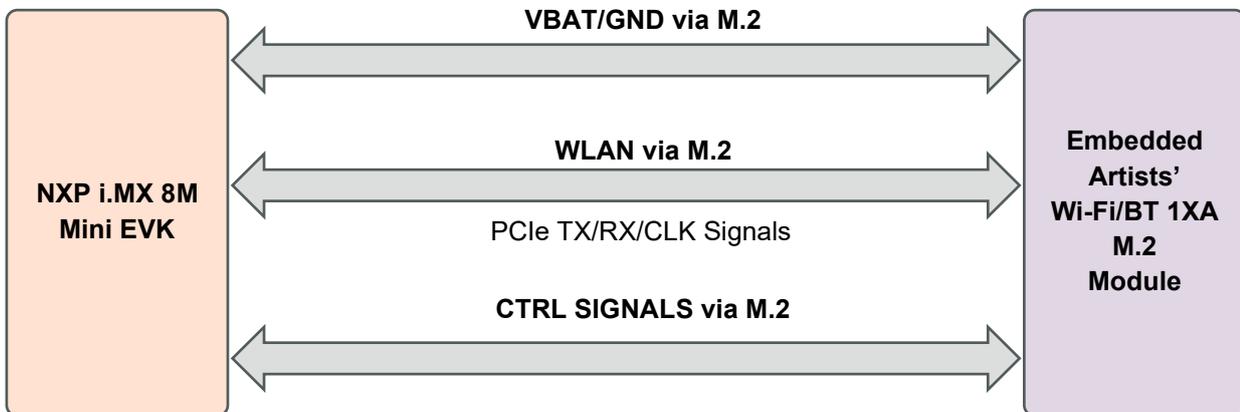
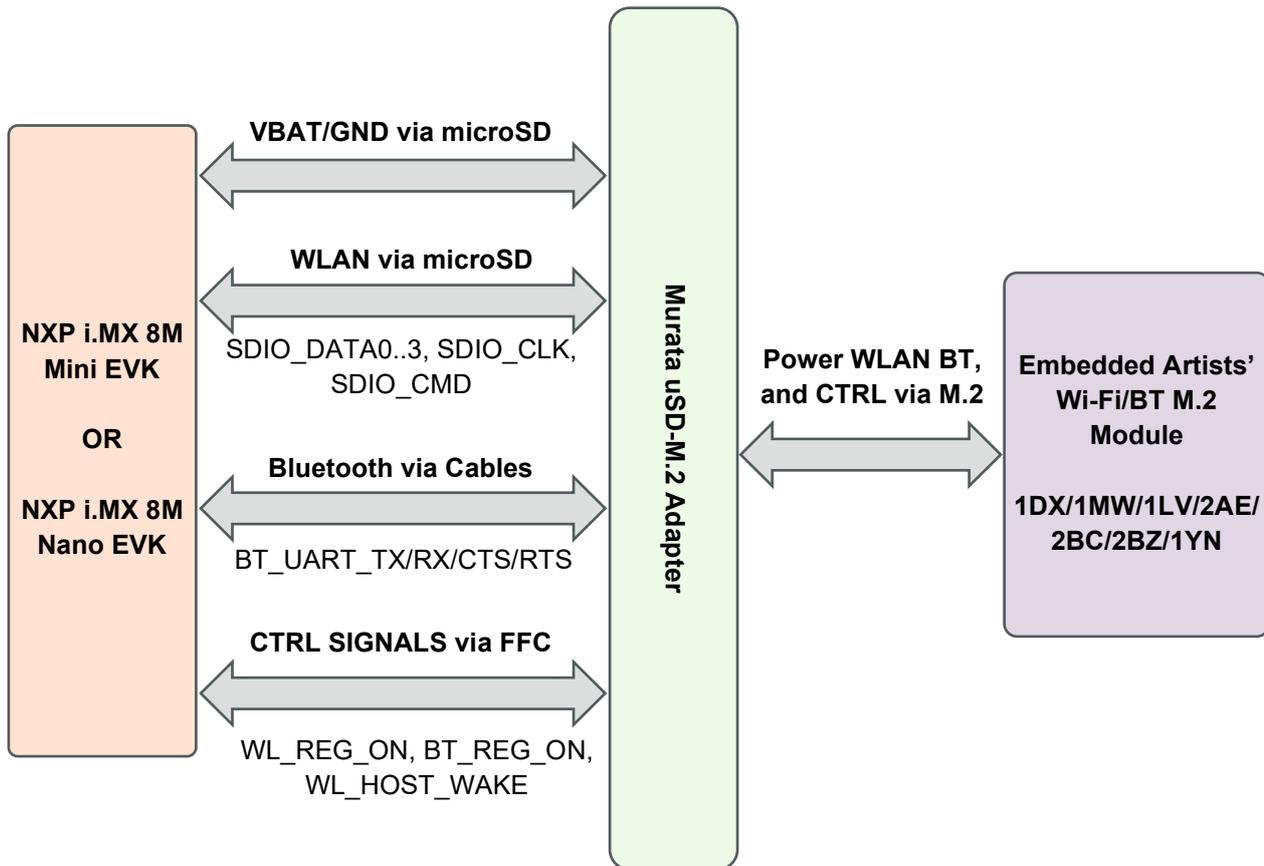


Figure 6 shows a simplified block diagram for the i.MX 8M Mini/Nano EVK interconnect with Murata's uSD-M.2 Adapter option (with Embedded Artists' Wi-Fi/BT M.2 EVB). Only WLAN-SDIO based modules are supported in this configuration: 1MW, 1DX, 1LV, 2AE, 2BC, 2BZ and 1YN. To properly support this (WLAN-SDIO VIO @1.8V; BT-UART VIO @3.3V) interconnect, Rev B1/B2 uSD-M.2 Adapter needs to be used given that it correctly level shifts the Bluetooth and WLAN/BT control signals between the M.2 EVB and the NXP i.MX 8M Mini EVK.

Figure 6: i.MX 8M Mini/Nano EVK Wi-Fi/BT SDIO Interconnect Block Diagram



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

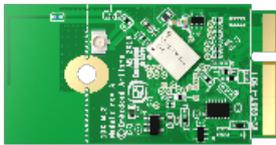
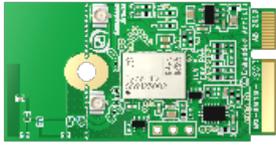
As already outlined in the [Introduction](#), the Murata Wi-Fi/BT solution is primarily arrived at using Embedded Artists' Wi-Fi/BT M.2 EVBs in conjunction with Murata's uSD-M.2 Adapter. This section provides additional details on the hardware solution.

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

Embedded Artists' designs, manufactures and distributes the Wi-Fi/BT M.2 EVBs based on Murata modules. These new M.2 EVBs 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](#).

Table 3 shows the details of Embedded Artists' Wi-Fi/BT M.2 Modules. Note that Type 1DX (CYW4343W), 1MW (CYW43455), 1LV (CYW43012), 2AE (CYW4373E), 2BC (CYW4373), 2BZ (CYW54590) and 1YN (CYW43439) support a WLAN-SDIO interface, whereas Type 1XA has a WLAN-PCIe interface. Also note that Type 1LV's interface voltage only supports 1.8V; whereas other EVBs interface voltage can be either 3.3V or 1.8V. To learn specifics on any of the M.2 EVBs, 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.

Table 3: Embedded Artists' Wi-Fi/BT M.2 Modules Supported

Murata Module	Chipset	Wi-Fi/BT support	Murata Part Number	EA M.2 Part Number	VIO	Interface	EVB Picture
1DX	CYW 4343W	802.11b/g/n BT/BLE 5.1	LBEE5KL1DX ↗	EAR00318 ↗	1.8V/ 3.3V	WLAN: SDIO Bluetooth: UART	
1MW	CYW 43455	802.11 a/b/g/n/ac (1x1 SISO) BT/BLE 5.0	LBEE5HY1MW ↗	EAR00315 ↗	1.8V/ 3.3V	WLAN: SDIO Bluetooth: UART	
1LV	CYW 43012	802.11 a/b/g/n/ac- friendly BT/BLE 5.0	LBEE59B1LV ↗	EAR00323 ↗	1.8V only	WLAN: SDIO Bluetooth: UART	
2AE	CYW 4373E	802.11 a/b/g/n/ac BT/BLE 5.2	LBEE5PK2AE ↗	EAR00388 ↗	1.8V/ 3.3V	WLAN: SDIO Bluetooth: UART	
2BC	CYW 4373	802.11 a/b/g/n/ac BT/BLE 5.2	LBEE5PK2BC ↗	EAR00436 ↗	1.8V/ 3.3V	WLAN: SDIO Bluetooth: UART	
2BZ	CYW 54590	802.11 a/b/g/n/ac (2x2 MIMO) BT/BLE 5.2	LBEE5XV2BZ ↗	EAR00414 ↗	1.8V/ 3.3V	WLAN: SDIO Bluetooth: UART	
1YN	CYW 43439	802.11 b/g/n BT/BLE 5.2	LBEE5KL1YN ↗	EAR00389 ↗	1.8V/ 3.3V	WLAN: SDIO Bluetooth: UART	
1XA	CYW 54591	802.11a/b/g/n/ac (2x2 MIMO; RSDB) BT/BLE 5.1	LBEE5XV1XA ↗	EAR00373 ↗	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 and some i.MX 8 Platforms requires the use of a Murata uSD-M.2 Adapter Kit in conjunction with Embedded Artists’ Wi-Fi/BT M.2 Module.

The [Murata uSD-M.2 Adapter Kit](#) (Part No: LBEE0ZZ1WE-uSD-M2) contents are shown in **Table 4**.



There are two versions of the uSD-M.2 Adapter: Rev A and Rev B1/B2. Currently only Rev B1/B2 is available through Distribution channel. It is backwards compatible to Rev A version; but has some important updates that include level shifting for Bluetooth UART and WLAN/Bluetooth control signals. This document differentiates important jumper settings on both the newest Rev B1/B2 and original Rev A Adapters.

Table 4: uSD-M.2 Adapter Kit Contents

Picture of Contents	Description of Contents
	uSD-M.2 Adapter (Revision B1/B2) Part Number: LBEE0ZZ1WE-uSD-M2
	M.2 screw for attaching Wi-Fi/Bluetooth M.2 Module
	75 mm 20-pos, 0.5 mm pitch flat/flex cable
	13 pieces 200 mm long male-to-female jumper cables (compatible with Arduino header)
	4x19 mm stand-offs in nylon and associated M3 screws
	microSD to SD Card Adapter

For more information on the uSD-M.2 Adapter, refer to [Section 8](#) or go to the [Adapter landing page](#).

2.3 NXP i.MX versus Murata Module Interconnect

Table 5 shows Murata module interconnect on various NXP i.MX Reference Platforms. Infineon chipset for each module is displayed. For a given i.MX platform and Murata module, you can quickly look up the compatibility. Details on the terminology used in the table are provided below:

- **“NC”** means “No Connect”. This is due to one or both of the following reasons:
 - VIO incompatible: Wi-Fi/BT M.2 Module requires VIO voltage level that the NXP i.MX Hardware cannot provide.
 - Physical bus (i.e., SDIO, PCIe, UART) and/or WLAN/Bluetooth control line interconnect is not available.
- **“MD”**: Module Down. The Murata module is not available on-board.
- **“uSD-M.2”**: Murata’s uSD-M.2 Adapter provides interconnect to the Embedded Artists’ Wi-Fi/BT M.2 Module. NXP i.MX Host WLAN SDIO is configured for 1.8V VIO default. The Bluetooth UART, and WLAN/BT control signals from i.MX Host are configured at 3.3V VIO (hardware limitation due to fixed voltage rails). Rev B1/B2 Adapter level shifts the BT UART and some of the WLAN/BT control signals. Although Rev A Adapter does not level shift BT UART (and some WLAN/BT control signals), it can still be used where shown for the i.MX 6UL and i.MX 6ULL EVKs.
- **“uSD-M.2W”**: Murata’s uSD-M.2 Adapter provides interconnect to the Embedded Artists’ Wi-Fi/BT M.2 Module. NXP i.MX Host WLAN SDIO is configured for 1.8V VIO default. The WLAN control signals from i.MX Host are configured at 3.3V VIO (hardware limitation due to fixed voltage rails). Rev B1/B2 Adapter level shifts some of the WLAN control signals. Although Rev A Adapter does not level shift WLAN control signals, it can still be used where shown for the i.MX 6UL and i.MX 6ULL EVKs. As such, there is no Bluetooth support – only WLAN.
- **“uSD-M.2P”**: Murata’s uSD-M.2 Adapter (Rev B1/B2) provides interconnect to the Embedded Artists’ Wi-Fi/BT M.2 Module. However, additional cabling to connect Bluetooth UART and WLAN/BT control signals is required for NXP i.MX 8M Mini EVK and 8M Nano EVK. The cable (Jumper Wire F/F 6”) is easily obtained through Distribution channel (example Digi-Key part numbers 1568-1644-ND or 1568-1513-ND).
- **“uSD-M.2-3.3V”**: Murata’s uSD-M.2 Adapter provides interconnect to the Embedded Artists’ Wi-Fi/BT M.2 Module. NXP i.MX Host WLAN SDIO is configured for (fixed at) 3.3V VIO. The Bluetooth UART, and WLAN/BT control signals from i.MX Host are configured at 3.3V VIO. Either Rev B1/B2 or Rev A uSD-M.2 Adapter can be used in this case – with correct jumper setting for 3.3V override mode.
- **“M.2”**: Only Embedded Artists’ Wi-Fi/BT M.2 EVB is required. Then NXP i.MX EVK has a M.2 connector. Currently only WLAN-PCIe, BT-UART, and WLAN/BT CTRL signals are brought out on the M.2 connectors for these specific NXP i.MX EVKs. As such, only Wi-Fi/BT M.2 EVBs which support WLAN-PCIe (i.e., 1XA) can be used.
- **“M.2W”**: Only Embedded Artists’ Wi-Fi/BT M.2 EVB is required. Then NXP i.MX EVK has a M.2 connector. Currently only WLAN-PCIe, and WLAN CTRL signals are brought out on the M.2 connectors for these specific NXP i.MX EVKs. As such, there is no Bluetooth support – only WLAN.



When using uSD-M.2 adapter, there are limitations on maximum SDIO clock frequency. 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 5: NXP i.MX/Murata Module Interconnect

NXP i.MX EVK Part Number	1DX	1MW	1LV	2AE	2BC	2BZ	1YN	1XA
	CYW 4343W	CYW 43455	CYW 43012	CYW 4373E	CYW 4373	CYW 54590	CYW 43439	CYW 54591
MCIMX8QXP-CPU	NC ⁵	NC ⁵	NC ⁵	NC ⁵	NC ⁵	NC ⁵	NC ⁵	M.2 ⁶
8MPLUSLPD4-EVK	M.2W ⁷	M.2W ⁷	M.2W ⁷	M.2W ⁷	M.2W ⁷	M.2W ⁷	M.2W ⁷	M.2W ⁷
MCIMX8M-EVKB	uSD-M.2W ⁸	uSD-M.2W ⁸	uSD-M.2W ⁸	uSD-M.2W ⁸	uSD-M.2W ⁸	uSD-M.2W ⁸	uSD-M.2W ⁸	M.2 ⁶
8MMINILPD4-EVKB	uSD-M.2P ⁹	uSD-M.2P ⁹	uSD-M.2P ⁹	uSD-M.2P ⁹	uSD-M.2P ⁹	uSD-M.2P ⁹	uSD-M.2P ⁹	M.2W ⁷
8MNANOD4-EVK	uSD-M.2P ⁹	MD ¹⁰ , uSD-M.2P ⁹	uSD-M.2P ⁹	uSD-M.2P ⁹	uSD-M.2P ⁹	uSD-M.2P ⁹	uSD-M.2P ⁹	NC ⁵
MCIMX6Q-SDB	uSD-M.2-3.3V ¹¹	uSD-M.2-3.3V ¹¹	NC ⁵	uSD-M.2-3.3V ¹¹	uSD-M.2-3.3V ¹¹	uSD-M.2-3.3V ¹¹	uSD-M.2-3.3V ¹¹	NC ⁵
MCIMX6UL-EVKB	uSD-M.2 ¹²	uSD-M.2 ¹²	uSD-M.2 ¹²	uSD-M.2 ¹²	uSD-M.2 ¹²	uSD-M.2 ¹²	uSD-M.2 ¹²	NC ⁵
MCIMX6ULL-EVK	uSD-M.2 ¹²	uSD-M.2 ¹²	uSD-M.2 ¹²	uSD-M.2 ¹²	uSD-M.2 ¹²	uSD-M.2 ¹²	uSD-M.2 ¹²	NC ⁵

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

3.1 “FMAC” Solution Overview

Previous NXP i.MX Kernels integrated the legacy “bcmhdh” WLAN driver. NXP later integrated the more relevant “FMAC” driver into their baseline BSP release. However, the NXP BSP-integrated “FMAC” driver never fully incorporated Infineon’s formal “FMAC” release.

There is also a distinct difference between the “FMAC” driver documented here; and 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 Infineon that is tested and verified. The Infineon “FMAC” release leverages the Linux Backports implementation to integrate the WLAN driver into the desired Linux kernel version.

Murata delivers this customer-friendly wireless driver release employing a customized Yocto layer “meta-murata-wireless”; which seamlessly disables any previous WLAN driver and pulls in the “FMAC” (officially supported) driver implementation. More specifically, it provides the following enhancements/customizations:

- Pull Infineon “FMAC” driver and run backports tool during Yocto build to generate necessary driver modules.
- Additional/necessary patches to Infineon “FMAC” driver for i.MX implementation.
- i.MX Linux kernel customizations to support “FMAC” driver with OOB IRQ interrupts.

⁵ No Connection options available

⁶ Wi-Fi/BT M.2 EVB plugs directly into M.2 connector

⁷ Wi-Fi/BT M.2 EVB plugs directly into M.2 connector; but only WLAN is functional

⁸ Works with uSD-M.2 Adapter (Rev B1/B2), but only WLAN is functional

⁹ Works with uSD-M.2 Adapter (Rev B1/B2) with additional cabling

¹⁰ Murata Module is soldered down

¹¹ Works with uSD-M.2 Adapter configured for 3.3V VIO override mode

¹² Works with uSD-M.2 Adapter configured for 1.8V VIO default

- Support 1.8V VIO signaling with NXP i.MX6UL(L) EVK.
- Support Wi-Fi/BT enablement on microSD slot of NXP i.MX 8M Mini/Nano EVKs.
- Fine tune DTS files so that optimal WLAN-SDIO throughput is achieved.
- WLAN production firmware files. For manufacturing test firmware (necessary for regulatory testing), please contact Murata directly. If no direct contact is available to you, then please post query to [Murata's Community Forum](#) .
- Murata NVRAM files for correctly configuring wireless module RF characteristics.
- Example Bluetooth patch files which allow customers to initial Bluetooth evaluation.
- WL tool binary necessary for interoperability and RF testing.
- Hostapd configuration (Infineon-specific version for a given "FMAC" release) with specific Infineon's patch release.
- Hostap-conf enablement.
- Hostap-utils enablement.
- WPA-supPLICANT configuration (Infineon-specific version for a given "FMAC" release) with Infineon's specific patch release.
- Wi-Fi Direct (P2P) enablement.

There are six versions of "FMAC" currently supported: "v5.4 spiga", "v5.4 baragon", "v5.10 cynder", "v5.10 drogon", "v5.15 ebrahim" and "v5.15 fafnir". Note that "spiga", "baragon", "cynder", "drogon", "ebirah" and "fafnir" are the Infineon codenames denoting "FMAC" release version. "v5.4", "v5.10" and "v5.15" are the latest kernel versions supported by the releases (can be backported to kernel version 3.0). To abbreviate references to specific versions of "FMAC", Murata uses just the Infineon codename – i.e., "spiga", "baragon", "cynder", "drogon", "ebirah" or "fafnir". It is strongly recommended to use the latest "FMAC" release version: currently this is "ebirah"/"fafnir".

3.2 Specific i.MX Target Support Details

Murata's customized Yocto layer ("meta-murata-wireless") supports the following NXP i.MX EVKs as outlined in **Table 6**. "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 EVKs (i.MX 8QXP, i.MX 8MQuad, i.MX 8M Mini, i.MX 8M Nano) 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 7** for the support of interrupt configuration, corresponding DTB (Device Tree Blob) files and hardware interconnect configuration (either module onboard, uSD-M.2 Adapter & M.2 EVB, or just M.2 EVB), and if the hardware configuration is limited to 3.3V VIO on WLAN SDIO interface (default WLAN SDIO VIO is 1.8V). The hardware 3.3V VIO WLAN SDIO interface limitation only applies to certain legacy i.MX 6 platforms (6Q, 6DL). The 3.3V VIO signaling requires the uSD-M.2 Adapter to be configured in a "3.3V VIO Override mode". Not all Wi-Fi/BT M.2 EVBs support 3.3V VIO on WLAN-SDIO interface.



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

NXP i.MX EVK Part #	NXP i.MX EVK	MACHINE=target	Kernel 4.14.98	Kernel 5.4.47	Kernel 5.10.52	Kernel 5.15.32
MCIMX8QXP-CPU	i.MX 8QuadXPlus MEK	imx8qxpmeek	✓	✓	✓	✓
8MPLUSLPD4-EVK	i.MX 8MPlus EVK	imx8mp-lpddr4-evk	✓	✓	✓	✓
MCIMX8M-EVKB	i.MX 8MQuad EVK	imx8mqevk	✓	✓	✓	✓
8MMINILPD4-EVKB	i.MX 8M Mini EVK	imx8mmevk	✓	✓	✓	✓
8MNANOD4-EVK	i.MX 8M Nano EVK	imx8mnddr4evk	✓	✓	✓	✓
MCIMX6Q-SDB	i.MX 6Quad SDB	imx6qsabresd	✓	✓	✓	✓
MCIMX6UL-EVKB	i.MX 6UL EVK	imx6ulevk	✓	✓	✓	✓
MCIMX6ULL-EVK	i.MX 6ULL EVK	imx6ull14x14evk	✓	✓	✓	✓

Table 7: i.MX6/8 Targets supported by Murata

Target (MACHINE)	Hardware Config	i.MX DTB File	Interrupt Config	3.3V VIO SDIO
imx8qxpmeek	M.2 ¹³	imx8qxp-mek.dtb	N/A	N/A
imx8mp-lpddr4-evk	M.2W ¹⁴	imx8mp-evk-usdhc1-m2.dtb	N/A	N/A
imx8mp-lpddr4-evk	M.2W ¹⁴	imx8mp-evk.dtb	N/A	N/A
imx8mqevk	MD ¹⁵ (1CX)	imx8mq-evk.dtb	N/A	N/A
imx8mqevk	M.2 ¹⁶	imx8mq-evk-pcie1-m2.dtb	N/A	N/A
imx8mqevk	uSD-M.2W ¹⁷	imx8mq-evk-usd-wifi.dtb	N/A	N/A
imx8mmevk	M.2W ¹⁸	imx8mm-evk.dtb	N/A	N/A
imx8mmevk	uSD-M.2P ¹⁹	imx8mm-evk-usd-m2-oob.dtb	OOB	N
imx8mmevk	uSD-M.2P ¹⁹	imx8mm-evk-usd-m2.dtb	SDIO	N
imx8mnddr4evk	MD ¹⁵ (1MW)	imx8mn-ddr4-evk.dtb	OOB	N
imx8mnddr4evk	uSD-M.2P ¹⁹	imx8mn-evk-usd-m2-oob.dtb	OOB	N
imx8mnddr4evk	uSD-M.2P ¹⁹	imx8mn-evk-usd-m2.dtb	SDIO	N
imx6qsabresd	uSD-M.2-3.3V ²⁰	imx6q-sabresd-btwifi-m2-oob.dtb	OOB	Y
imx6qsabresd	uSD-M.2-3.3V ²⁰	imx6q-sabresd-btwifi-m2.dtb	SDIO	Y
imx6ulevk	uSD-M.2 ²¹	imx6ul-14x14-evk-btwifi-m2-oob.dtb	OOB	N
imx6ulevk	uSD-M.2 ²¹	imx6ul-14x14-evk-btwifi-m2.dtb	SDIO	N
imx6ull14x14evk	uSD-M.2 ²¹	imx6ull-14x14-evk-btwifi-m2-oob.dtb	OOB	N
imx6ull14x14evk	uSD-M.2 ²¹	imx6ull-14x14-evk-btwifi-m2.dtb	SDIO	N

¹³ Wi-Fi/BT M.2 EVB plugs directly into M.2 connector

¹⁴ Wi-Fi/BT M.2 EVB plugs directly into M.2 connector; but only WLAN is functional

¹⁵ Murata Module is soldered down

¹⁶ Wi-Fi/BT M.2 EVB plugs directly into M.2 connector

¹⁷ Works with uSD-M.2 Adapter (Rev B1/B2), but only WLAN is functional

¹⁸ Wi-Fi/BT M.2 EVB plugs directly into M.2 connector; but only WLAN is functional

¹⁹ Works with uSD-M.2 Adapter (Rev B1/B2) with additional cabling

²⁰ Works with uSD-M.2 Adapter configured for 3.3V VIO override mode

²¹ Works with uSD-M.2 Adapter configured for 1.8V VIO default

3.3 Murata “FMAC” Customized i.MX Yocto Image Build

Murata’s solution to easily enable Infineon-based Wi-Fi/Bluetooth functionality requires a complete Linux image build (bootloader, kernel, DTB files, filesystem). To understand this requirement, we need to understand specifics about the NXP i.MX Linux image. The NXP i.MX image contains third party IP which is sub-licensed via a click-through EULA (when either downloading a NXP i.MX validation/demo image or building the image from source). As such Murata cannot make this image available directly to customers. As detailed by the [Murata Linux User Manual](#), Murata employs a wireless-enabling “meta-murata-wireless” layer to make this customized Yocto build simple and user-friendly. Nonetheless end users must still 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

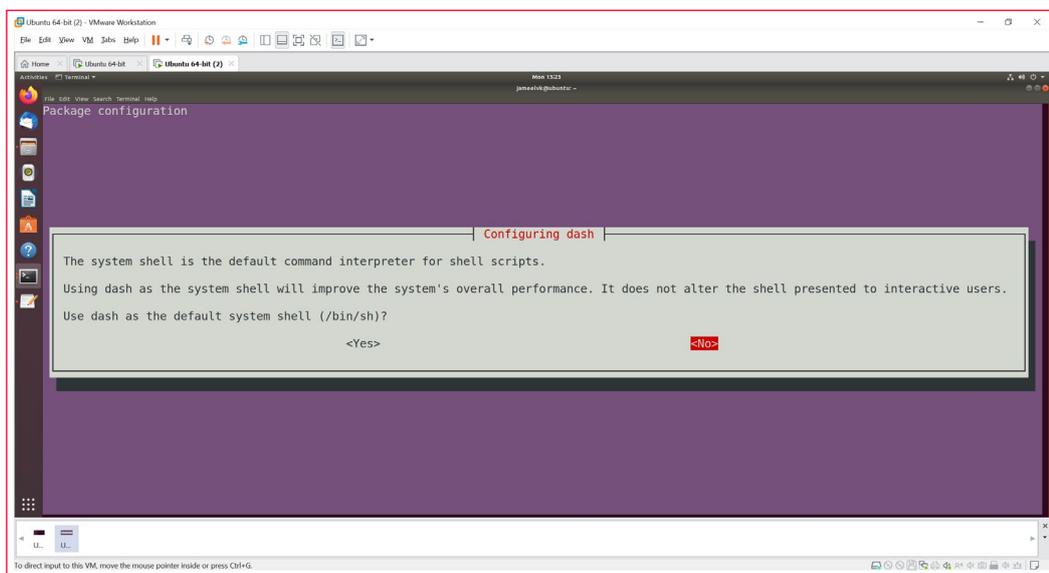
First step is to install Ubuntu 14.04, 16.04, 18.04 or 20.04 (Murata’s build is verified on Ubuntu 20.04 64-bit install) on the host - native PC or virtual environment like VMware. Host PC typically used requires Ubuntu 20.04/18.04/16.04/14.04 installed with 50 GB free disk space (80 GB needed for i.MX8 build). For more information on the Ubuntu download, please refer to [this link](#). The Ubuntu installation manual is provided [here](#). By default, Ubuntu sets the environment to use dash. It is mandatory that, User sets the default system shell to “No” when configuring dash. Follow the steps mentioned below for reconfiguring dash:

- Open “Terminal” App in Ubuntu 16.04 and enter the command, “sudo dpkg-reconfigure dash”

```
sudo dpkg-reconfigure dash
```

- Enter the password.
- Select “No” when “Configuring dash” screen appears as shown in **Figure 7**.

Figure 7: Configuring dash



3.3.2 Download Murata's Script Files

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

- Using “web browser” option to download “meta-murata-wireless” zip file and extract:
 - Click on “clone or download” button at [Murata GitHub](#).
 - 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/script-utils/latest” folder where the necessary README and script files are contained.
- Use “wget” command to pull specific files from Murata GitHub.



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/script-utils/latest/README.txt

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

wget --no-check-certificate --content-disposition
https://github.com/murata-wireless/meta-murata-
wireless/raw/master/script-
utils/latest/Murata_Wireless_Yocto_Build_CYW.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](#). 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 Documentation](#)).

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:

- Verifying Host Environment
- Verifying Host Script Version
- Installing Essential Yocto host packages

4. GIT Configuration: verifying Username and email ID

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_CYW.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](#) . 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 20.04 (preferred), 18.04, 16.04, or 14.04.
- 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:
 - 5.15.32_2.0.0 GA
 - 5.10.52_2.1.0
 - 5.4.47_2.2.0 GA
 - 4.14.98_2.3.0
- Creating the i.MX BSP folder is straightforward:

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

- 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.



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.

Murata's build script performs the following tasks:

- Verifies host environment (i.e., Ubuntu 14.04/16.04/18.04/20.04).
- Installs the ‘repo’ tool.
- 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.



Murata only runs “spot” tests before submitting fixes/enhancements to the branch. The “Developer” branch build may fail. In this case, the user is highly recommended to employ the “Stable” branch (formal tag release) and apply any necessary patches to it.

- 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 the “FMAC” release. The script displays both the “FMAC” codename and latest kernel version supported by that release. Currently, six “FMAC” releases are supported: “spiga”, “baragon”, “cynder”, “drogon”, “ebirah” and “fafnir” (most recent and up to date regarding fixes and enhancements).



Murata strongly recommends using “ebirah”/“fafnir” release.



For legacy builds (e.g., “mothra”, “manda”, “kong”, “zigra” etc., the flag LEGACY_SUPPORT can be enabled on the script. However, these legacy builds are not supported by Murata and can be broken.

- Select i.MX target: refer to **Table 6** and **Table 7** for more details.
- 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 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_CYW.sh
```

The script goes through the following stages:

- Verifying Host Environment
- Install the ‘repo’ tool.
- Verifying Script Version

- Select Release Type:
 - Stable: Murata tested/verified release tag. Stable is the recommended default.
 - Developer: Includes latest fixes on branch. May change at any time.
- Select “Linux Kernel”
- Select “FMAC” version
- Select Target
- Select DISTRO & Image
- Creation of Build directory
- Verify your selection
- Acceptance of End User License Agreement (EULA)
- 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 A](#). Once the Murata-customized i.MX image is built, it will be located at the following location:

```
<$BSP_DIR>/<build target folder - selected during script>
/tmp/dep/oy/images/<$target>/
```

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

```
~/murata-imx-bsp/imx6ulevk_build/tmp/dep/oy/images/imx6ulevk/
```

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

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

With the Linux image built and located, [Section 4](#) outlines flashing steps for (micro) SD card (on all platforms except 8MMINILPD4-EVK and 8MNANOD4-EVK) or eMMC on aforementioned i.MX 8M Mini/Nano EVKs.

3.4 Additional Hardware/Software Considerations

3.4.1 Out-Of-Band (OOB) IRQ Support on NXP i.MX EVKs

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-saving mechanisms are limited. Specifically, the host cannot shut down the SDIO bus when using SDIO in-band signaling for WLAN interrupt mechanism – having to continually poll for SDIO in-band interrupts instead. Also, WLAN throughput optimization is achieved with OOB IRQ configuration. By referring to **Table 7**, we can easily distinguish which legacy NXP i.MX 6 platforms require hardware rework to configure them for OOB IRQ signaling over WLAN bus. Any platform/target with “SDIO” as an option in the “SDIO Interrupt Config” column indicates that the default hardware configuration only supports SDIO in-band signaling. To be clear, we are flagging the following i.MX 6 Platforms which do not support WLAN OOB IRQ configuration out-of-box:

- **i.MX 6Q SDB/SDP**: This platform requires rework to enable Bluetooth and WLAN/Bluetooth control signals. Refer to the [Hardware User Manual](#) for specifics.
- **i.MX 6UL(L) EVKs**: these platforms require one resistor to be move to support OOB IRQ support.



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](#) .

In summary, SDIO in-band interrupts are recommended on these legacy i.MX 6 platforms unless necessary rework is done. Refer to the [Hardware User Manual](#)  for necessary modifications on all NXP i.MX 6 Platforms.



Murata does not support hardware rework – other than documenting it. The customer takes full responsibility when modifying their platform.

3.4.2 1.8V versus 3.3V VIO Signaling using Murata's uSD-M.2 Adapter

Some legacy i.MX 6 Platforms except i.MX 6UL(L) EVKs are limited to 3.3V VIO signaling over WLAN-SDIO when using the uSD-M.2 Adapter. When the Adapter is configured in “3.3V VIO Override Mode”, all signals connected to the Wi-Fi/BT M.2 EVB are at 3.3V VIO level. The uSD-M.2 Adapter must be jumpered specifically for this configuration; and only some of the Wi-Fi/BT M.2 EVBs support this non-default voltage signaling. Currently only Type 1DX, Type 1MW, Type 2AE Type 2BC, Type 2BZ and Type 1YN M.2 EVBs (WLAN-SDIO) support this 3.3V VIO override mode configuration. Note that Type 1LV M.2 EVB is 1.8V VIO only and thereby cannot run in this configuration.

3.4.3 UHS SDIO 3.0 operation on i.MX 6UL(L) EVKs with uSD-M.2 Adapter

When using Murata's uSD-M.2 adapter to interconnect the Wi-Fi/BT M.2 EVB to a NXP i.MX 6 platform, the maximum SDIO clock frequency is limited to 50 MHz for both 1.8V and 3.3V VIO. However, there is no such limitation with the NXP i.MX 8M Mini and 8M Nano EVKs which support a direct microSD connect – the i.MX 6 platforms require the microSD-to-SD Adapter. For UHS mode (and better overall hardware/software) support, Murata strongly recommends the [Embedded Artists' i.MX Developer Kits](#) . See [Embedded Artists' Solution section](#)  for more details.

3.4.4 WLAN/Bluetooth M.2 Direct Interconnect on NXP i.MX Platforms

As shown in **Table 5** and **Table 7**, NXP's i.MX 8 Family of Eve's does support direct M.2 interconnect. However, there are specific limitations on the existing platforms noted in this document:

- NXP i.MX 8QXP and i.MX 8MQuad EVKs support only WLAN-PCIe via the direct M.2 interconnect. No WLAN-SDIO interconnect is supported out-of-box.
- NXP i.MX 8MPlus EVK supports both WLAN-SDIO and WLAN-PCIe signals via the direct M.2 interconnect, but does not support Bluetooth-UART.
- NXP i.MX 8M Mini EVK only has WLAN-PCIe interconnect via the direct M.2 and does not support Bluetooth-UART.
- NXP i.MX 8M Nano EVK (although there is a M.2 connector on baseboard) does not support any M.2 WLAN/BT interconnect.



Embedded Artists' i.MX Developer Kits support full M.2 interconnect with additional debug signal support. This is one of the key reasons Murata recommends their hardware platforms.

3.4.5 Setting Correct Software Configuration before Testing Wi-Fi/BT Solution

There are two important steps to follow when configuring software when first booting Linux:

- Set DTB (Device Tree Blob) file correctly (“fdt_file” boot variable) when bootloader first comes up. If not set correctly, the kernel may not boot, or the Wi-Fi/BT may not function correctly. As already described, unless the platform is module down or the interconnect provides WLAN_HOST_WAKE connection, the default DTB selected uses in-band SDIO interrupts.



Starting from Linux Kernel Versions 5.10+, for i.MX6UL(L), the boot variable is “fdt_file”, whereas for i.MX8 Targets, the boot variable is “fdtfile”. Note the “_” (underscore) between the two variables.

- Invoke “set_module.sh 2AE|2BC” after Linux kernel boots and then reboot the platform if either of those modules are being used. This is necessary to configure the correct NVRAM, CLM Blob and Firmware. This step is not necessary for any of the other modules.

4 Preparing NXP i.MX Platforms to Boot Linux Image

In this section we document how to flash the (micro) SD card with the Linux image using both Linux and Windows PC. Most NXP i.MX EVKs use the (micro) SD card to boot the platforms. However, there are two special cases as documented in **Table 8** below. These two configurations include the i.MX 8M Mini EVK (8MMINILPD4-EVK variant) and i.MX 8M Nano EVK with the “uSD-M.2P” configuration. In both cases the NXP i.MX EVKs need to boot from eMMC in place of microSD card given that the WLAN-SDIO connection is over the uSD connector. Refer to [Section 4.2](#) for detailed steps on flashing these platforms.

Table 8: Select Hardware Configurations which use eMMC Boot Configuration

Target (MACHINE)	Hardware Config	i.MX DTB File	Boot Config	Interrupt Config	3.3V VIO SDIO
imx8mmevk	M.2W ²²	imx8mm-evk.dtb	uSD	N/A	N/A
imx8mmevk	uSD-M.2P ²³	imx8mm-evk-usd-m2-oob.dtb	eMMC	OOB	N
imx8mmevk	uSD-M.2P ²³	imx8mm-evk-usd-m2.dtb	eMMC	SDIO	N
imx8mnddr4evk	MD ²⁴ (1MW)	imx8mn-ddr4-evk.dtb	uSD	OOB	N
imx8mnddr4evk	uSD-M.2P ²³	imx8mn-evk-usd-m2-oob.dtb	eMMC	OOB	N
imx8mnddr4evk	uSD-M.2P ²³	imx8mn-evk-usd-m2.dtb	eMMC	SDIO	N

4.1 Flashing Murata-customized Linux Image to (micro) SD Card

This section presents two different platforms for flashing (micro) SD card. The primary support is on a Linux PC (preferably Ubuntu distro). In addition, steps are shown for Windows PC.

4.1.1 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 8**. This Kingston device ([MobileLite Plus microSD Reader](#)) provides direct plug-ins for microSD and SD cards. It supports USB 3.2 and UHS-II microSD 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 8: USB to SD Card Reader/Writer Adapter



²² Wi-Fi/BT M.2 EVB plugs directly into M.2 connector; but only WLAN is functional

²³ Works with uSD-M.2 Adapter (Rev B1/B2) with additional cabling

²⁴ Murata Module is soldered down

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”.



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.

Keep in mind that for Yocto release Zeus and later, the Linux image file has a “.wic” extension. Prior to Zeus, the filename extension is “.sdcad”.

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.wic 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](#) .

4.1.2 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”²⁵ 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, click on the folder icon/button and navigate to the location of the desired “.wic” file (for Yocto release Zeus and later).
- 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 utilities²⁶.
- 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.

²⁵ “Win32 Disk Imager” is an open source tool that can be downloaded from websites such as “sourceforge.net”.

²⁶ Unlike Linux environment, Windows PC does not require use of “USB to SD Card Reader/Writer” adapter.

- To be safe, you may elect to “eject” the SD card removable memory device before removing it.

4.2 Flashing Linux Image to NXP i.MX 8M Mini/Nano EVKs

Here is an overview of the procedure for flashing the i.MX 8M Mini/Nano EVK so it can support Murata’s uSD-M.2 Adapter with Embedded Artists’ Wi-Fi/BT M.2 EVB:

- Prepare necessary files to flash platform (“uuu.exe”, bootloader, and root file system).
- Configure i.MX hardware platform so eMMC can be flashed.
- Flash the platform (eMMC) with “uuu.exe” tool.
- Configure i.MX hardware platform so it will boot from eMMC.

Flashing steps (for i.MX 8M Mini and 8M Nano) are essentially identical with differences in filenames and switch settings. All differences are clearly noted. Note that flashing procedure is done using a Windows PC. However, the steps using Linux machine would be very similar. NXP provides “uuu” executables/binaries (on listed GitHub repository) for both Windows and Linux PC’s.

4.2.1 Software File Preparation

Before programming the eMMC, the user needs the following files:

- NXP’s programming utility (“uuu.exe”)
- i.MX 8M Mini/Nano Bootloader
- i.MX 8M Mini/Nano Root file system

Table 9 below lists the necessary files and their locations. “uuu.exe” is pulled from a GitHub repository. The bootloaders and images are from the user’s customized Murata Yocto build.

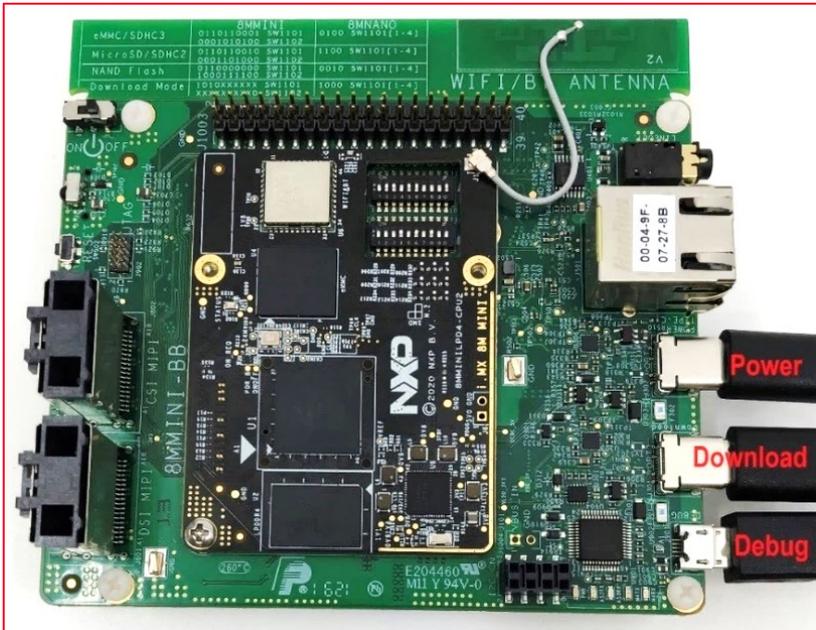
Table 9: Files to flash i.MX 8M Mini & 8M Nano EVKs

i.MX Host	Filename	Location
i.MX 8M Mini	uuu.exe	GitHub 
	imx-boot-imx8mmevk-sd.bin-flash_evk	\$BUILD_DIR/tmp/deploy/images/imx8mmevk/
	fsl-image-validation-imx-imx8mmevk.wic.bz2	\$BUILD_DIR/tmp/deploy/images/imx8mmevk/
i.MX 8M Nano	uuu.exe	GitHub 
	imx-boot-imx8mnddr4evk-sd.bin-flash_ddr4_evk	\$BUILD_DIR/tmp/deploy/images/imx8mnddr4evk/
	fsl-image-validation-imx-imx8mnddr4evk.wic.bz2	\$BUILD_DIR/tmp/deploy/images/imx8mnddr4evk/

4.2.2 i.MX 8M Mini or Nano EVK Hardware Configuration

This section describes steps to correctly configure i.MX hardware platform before using “uuu” executable to flash Linux image. **Figure 9** shows the necessary connections for power, download, and debug (serial console).

Figure 9: Power, Download, and Debug port connection to board



To download images using “uuu”, the board must be first put into download mode. The default DIP switch settings must be changed for either NXP i.MX 8 platform.



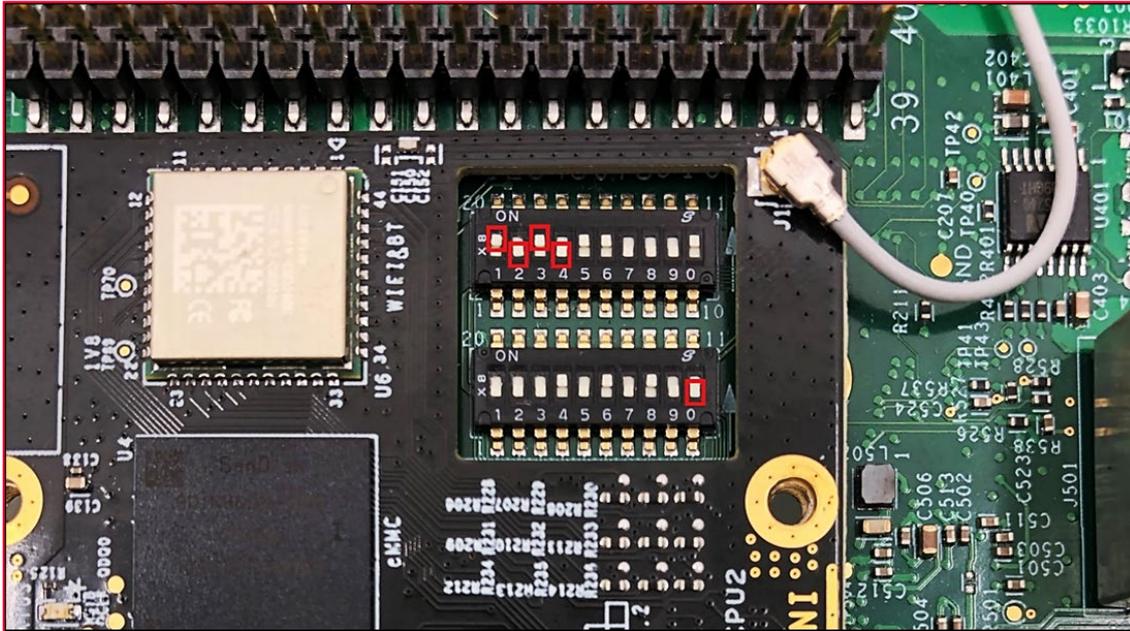
The DIP switch settings for i.MX 8M Mini EVK and 8M Nano EVK are different.

For i.MX 8M Mini EVK (8MMINILPD4-EVK), this is accomplished by setting the DIP switches (SW1101 and SW1102) to the positions below. The DIP switch settings are also shown in **Figure 10**.

Switch	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
SW1101	1	0	1	0	X	X	X	X	X	X
SW1102	X	X	X	X	X	X	X	X	X	0

Where 1 – ON, 0 – OFF and X – Do not care.

Figure 10: i.MX 8M Mini EVK DIP Switches configured for Download

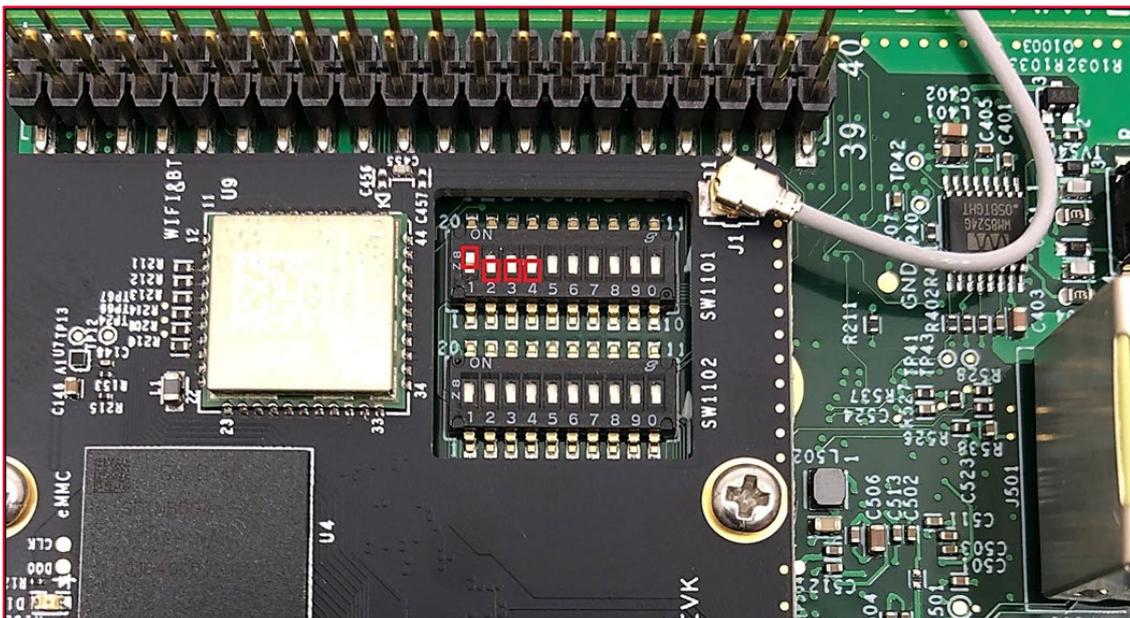


For i.MX 8M Nano EVK (8MNANOD4-EVK), this is accomplished by setting the DIP switch (SW1101) to the positions below. The DIP switch settings are also shown in **Figure 11**.

Switch	#1	#2	#3	#4
SW1101	1	0	0	0

Where 1 – ON, 0 – OFF.

Figure 11: i.MX 8M Nano EVK DIP Switches configured for Download



4.2.3 Flash Linux Image to eMMC on i.MX 8M Mini/Nano EVK

Now that the hardware is correctly configured, we can run “uuu.exe” executable to flash the platform.

- On Windows open a Command Prompt, navigate to the folder where the utility “uuu.exe” was downloaded along with the bootloader and root file system.
- Run the UUU tool.

Per **Table 9**, the filenames are specific to either i.MX 8M Mini or Nano EVK platform.

- For i.MX 8M Mini EVK, type the following:

```
C:\nxp-tool\uuu -b emmc_all imx-boot-imx8mmevk-sd.bin-flash_evk fsl-image-validation-imx-imx8mmevk.wic.bz2/*
```

- For i.MX 8M Nano EVK, type the following:

```
C:\nxp-tool\uuu -b emmc_all imx-boot-imx8mnddr4evk-sd.bin-flash_ddr4_evk fsl-image-validation-imx-imx8mnddr4evk.wic.bz2/*
```

- Upon successful programming, user will see the following messages.

```
uuu (universal update utility) for nxp imx chips - libuuu_1.3.191-0-f4fe24b9
Success 1          Failure 0
1:4                8/    8 [Done          ] FB: done
```

4.2.4 Configure i.MX 8M Mini/Nano EVK to boot from eMMC

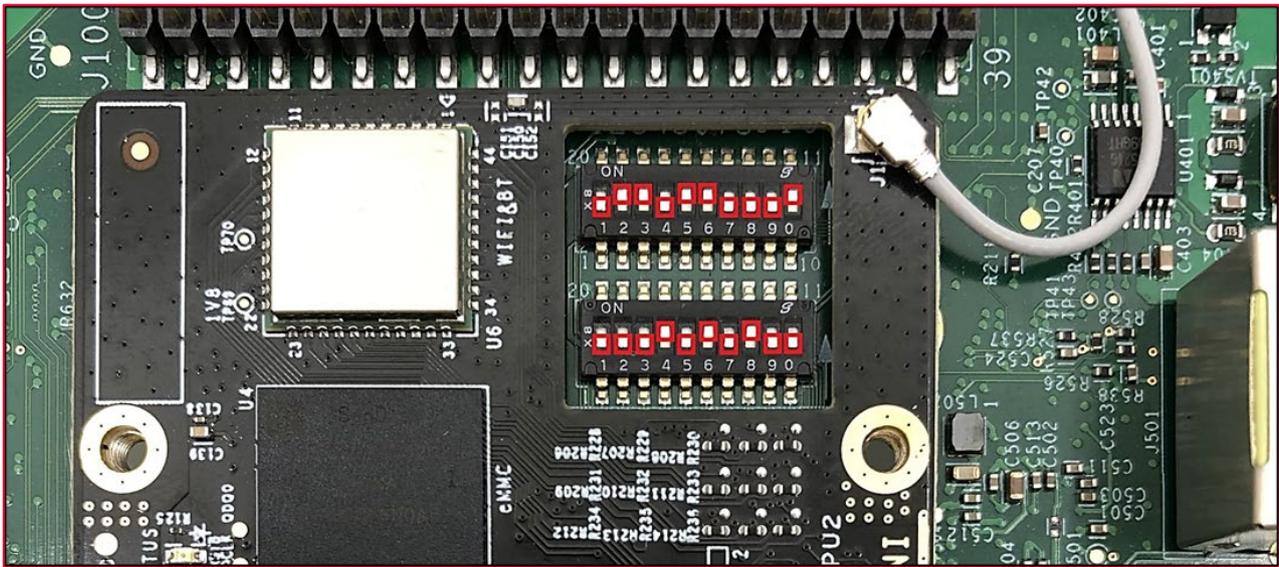
At this step, the i.MX 8M Mini/Nano EVK has been successfully flashed and is ready to boot. Now we must change the DIP switch settings to boot from eMMC. Note that the DIP switch settings for i.MX 8M Mini EVK and 8M Nano EVK are different.

For i.MX 8M Mini EVK (8MMINILPD4-EVK), this is accomplished by setting the DIP switches (SW1101 and SW1102) to the positions below. The DIP switch settings are also shown in **Figure 12**.

Switch	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
SW1101	0	1	1	0	1	1	0	0	0	1
SW1102	0	0	0	1	0	1	0	1	0	0

Where 1 – ON, 0 – OFF.

Figure 12: i.MX 8M Mini EVK DIP Switches configured for eMMC Boot

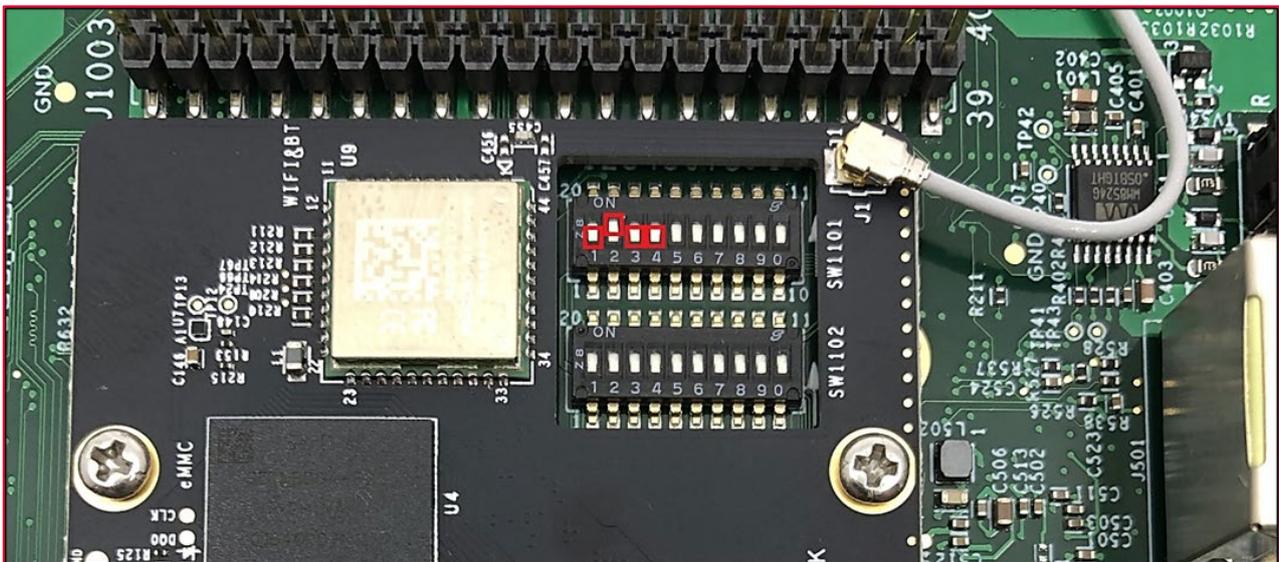


For i.MX 8M Nano EVK (8MNANOD4-EVK), this is accomplished by setting the DIP switch (SW1101) to the positions below. The DIP switch settings are also shown in **Figure 13**.

Switch	#1	#2	#3	#4
SW1101	1	0	0	0

Where 1 – ON, 0 – OFF.

Figure 13: i.MX 8M Nano EVK DIP Switches configured for eMMC Boot



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

Embedded Artists' Wi-Fi/BT M.2 EVBs based on SDIO, listed in **Table 3** (currently 1DX/1MW/1LV/2AE/2BC/2BZ/1YN) can be connected to the i.MX 6 Platforms through Murata's uSD-M.2 Adapter as shown in **Figure 14**. The following sub-sections details steps for bringing up Embedded Artists' Wi-Fi/BT EVBs on the i.MX 6 Platform supported: i.MX 6Q, i.MX 6UL(L).

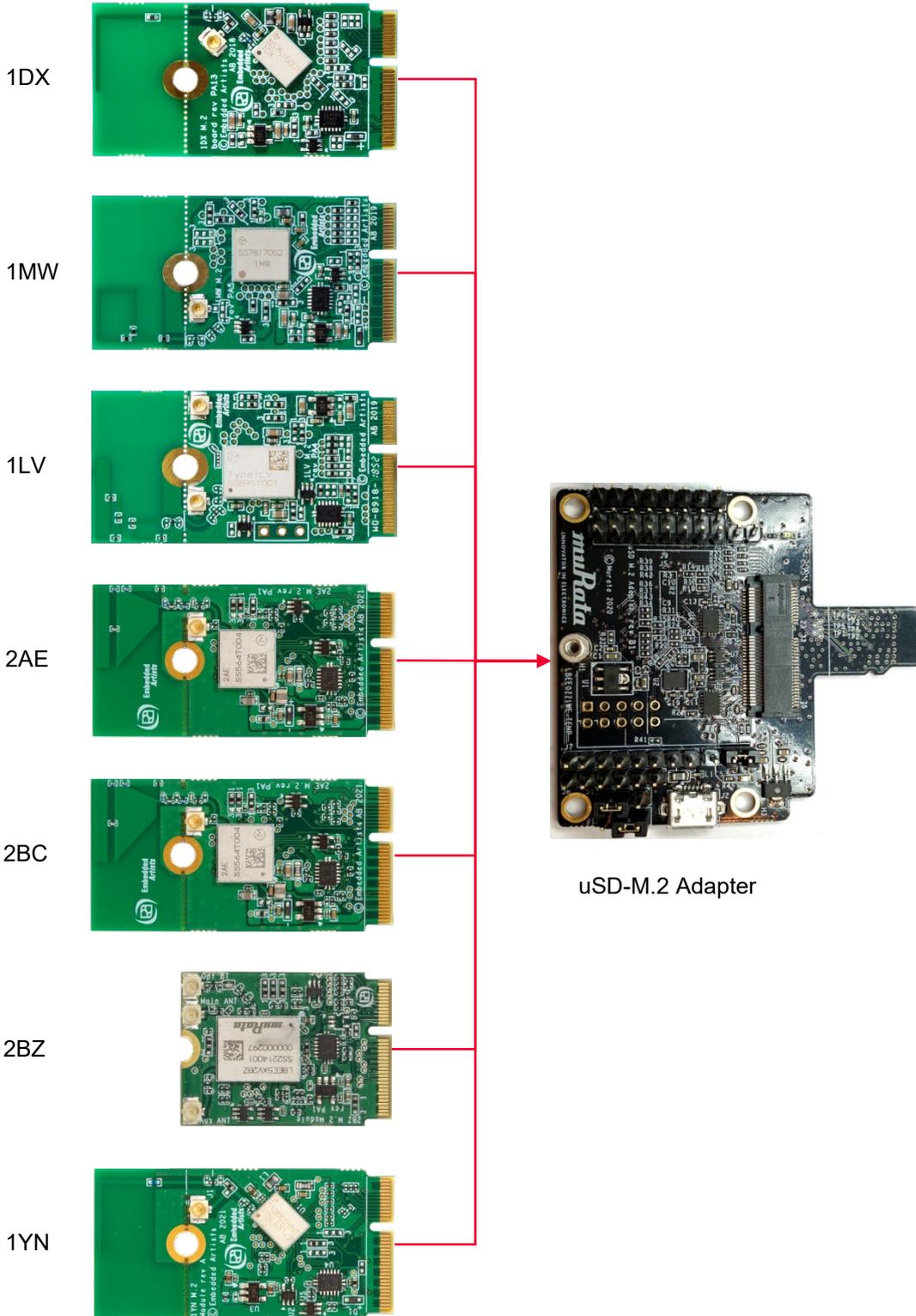


The 3.3V WLAN-SDIO VIO will only work with the Wi-Fi/BT M.2 EVBs that support it – currently Type 1DX, 1MW, 2AE, 2BC, 2BZ and 1YN. Type 1LV only supports 1.8V VIO – which is the default WLAN-SDIO VIO signaling for the M.2 specification. The 3.3V VIO override mode also configures BT-UART VIO at 3.3V – which is fine given that both host and target are signaling at that same voltage.

Both NXP i.MX 6UL and 6ULL EVKs are configured (via Murata's custom software solution) for the Wi-Fi/BT M.2 specification of 1.8V WLAN-SDIO VIO operation. Note that the M.2 specification also has BT UART VIO at 1.8V as well. The latest Rev B1/B2 uSD-M.2 Adapter incorporates level shifting to provide the necessary BT UART VIO. The legacy Rev A Adapter does not have level shifting – as such the BT UART signaling is mixed between host (3.3V) and target (1.8V). Although Rev A Adapter configuration still works, customers are recommended to use the latest Rev B1/B2 Adapter with correct voltage signaling on BT UART.

For more information on hardware configuration refer to the [Hardware User Manual](#) .

Figure 14: uSD-M.2 Adapter with M.2 EVB options



5.1 Connecting to i.MX 6Q(P)/DL SDB (3.3V WLAN-SDIO VIO Override)

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.

5.1.1 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 6Quad/DualLite 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.



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.

5.1.2 Wi-Fi/Bluetooth Bring-Up on i.MX 6Quad/DualLite SDB/SDP



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 only one drawback is Ethernet port access. To properly match Wi-Fi/BT EVK and i.MX6 platform heights, additional nylon standoffs are required.

- Ensure no power is applied to i.MX 6Quad(Plus)/DualLite SDB. 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.
- On uSD-M.2 Adapter, set Jumper J1 to position 2-3 (VBAT from microSD connector). Make sure Jumper J11 is removed (BT core enabled). Note Blue LED2 illuminates in 3.3V override mode.
 - For Rev B1/B2 adapter, install J13/J12 in 1-2/2-3 positions, respectively for 3.3V override mode.
 - For legacy Rev A adapter, short J12 for 3.3V VIO override mode.
- Connect the Wi-Fi/BT M.2 EVB to the M.2 connector on the uSD-M.2 Adapter per [Section 8.1](#). Connect the uSD-SD Card adapter and tape the uSD Adapter-SD Card connection per [Section 8.3](#).
- Connect the ribbon cable at both ends before inserting Wi-Fi/BT EVK (Wi-Fi/BT M.2 EVB/uSD-M.2 Adapter/uSD-SD Card) into SD2 slot. Note the orientation as shown in **Figure 15**.
- Prepare SD card to boot platform per [Section 4.1](#). Insert SD card, power on the platform and interrupt at u-boot. Set DTB configuration with “fdt_file” parameter (“oob” string configures OOB IRQ configuration; see [Section 3.4.1](#) for more details). You can check the available DTB files using the command “fatls mmc 1”. After setting DTB, save the u-boot configuration and boot the platform:

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

# Boot kernel
boot
```

- After the kernel boots, invoke “set_module.sh 2AE|2BC” if either of those modules are used and reboot:

```
set_module.sh 2AE|2BC
reboot
```

- Refer to [Section 7](#) to test/verify Wi-Fi and Bluetooth functionality.

Figure 15: i.MX 6Quad/DualLite SDB (Inverted) with uSD-M.2 Adapter and Type 1MW M.2 EVB



5.2 Connecting to i.MX 6UL EVK or i.MX 6ULL EVK (1.8V WLAN-SDIO VIO)

- Ensure no power is applied to i.MX 6UL(L) 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.
- On uSD-M.2 Adapter, set Jumper J1 to position 2-3 (VBAT from microSD connector). Make sure Jumper J11 is removed (BT core enabled). Note Blue LED2 is not illuminated for 1.8V VIO.
 - For Rev B1/B2 adapter, install J13/J12 in 1-2/1-2 positions, respectively for 1.8V VIO.
 - For legacy Rev A adapter, make sure J12 is not installed for 1.8V VIO.
- Connect the Wi-Fi/BT M.2 EVB to the M.2 connector on the uSD-M2 Adapter per [Section 8.1](#). Connect the uSD-SD Card adapter and tape the uSD Adapter-SD Card connection per [Section 8.3](#).
- Connect ribbon cable at both ends before inserting Murata EVK into SD1 slot. Note the orientation as shown in **Figure 16**. Make sure that the adapter clicks in correctly – the i.MX 6UL(L) EVKs have a Push-Push SD card connector. Tape the SD Card-EVK connection [Section 8.3](#).
- Prepare microSD card to boot platform per [Section 4.1](#). Insert microSD card, power on the platform and interrupt at u-boot. Set DTB configuration with “fdt_file” parameter (“oob” string configures OOB IRQ configuration; see [Section 3.4.1](#) for more details). You can check the available DTB files using the command “fatls mmc 1”. After setting DTB, save the u-boot configuration and boot the platform:

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

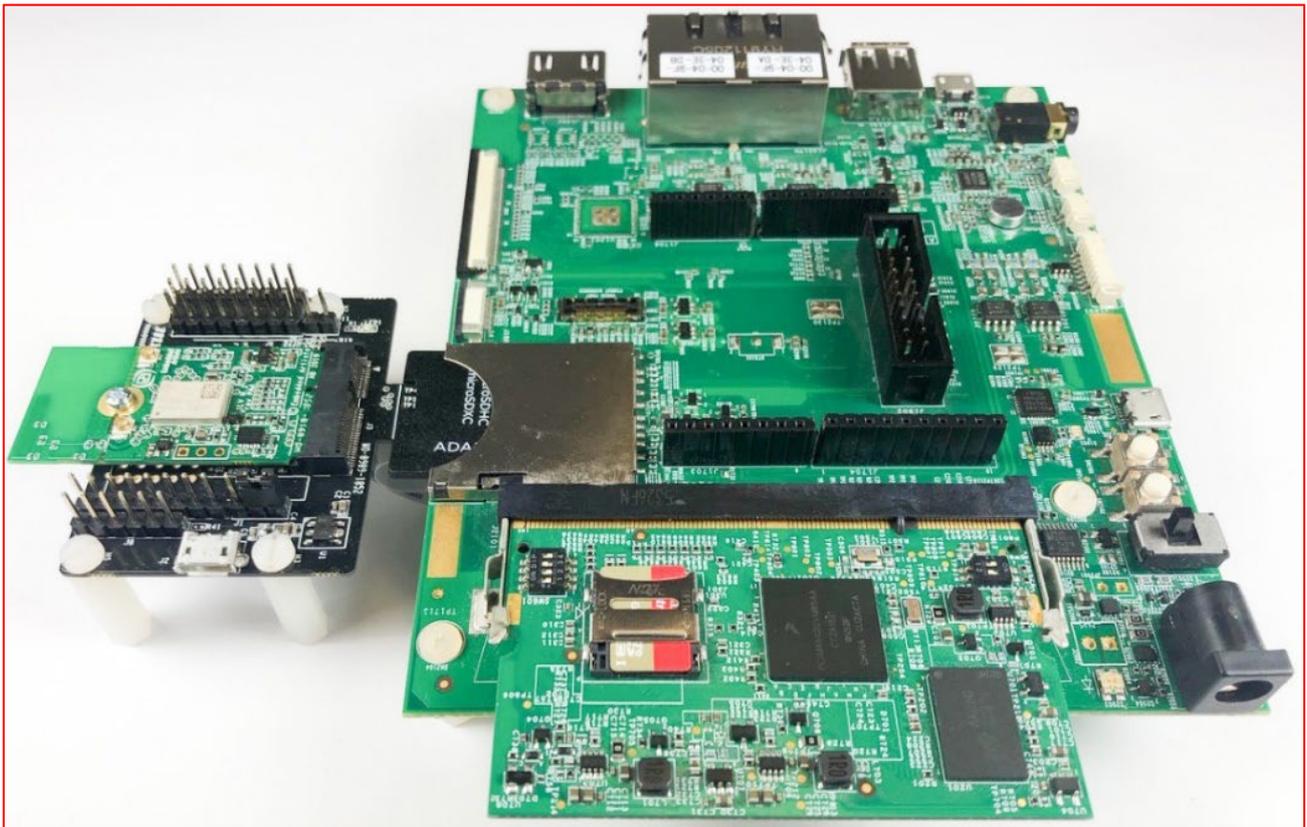
# Boot kernel
boot
```

- After the kernel boots, invoke “set_module.sh 2AE|2BC” if either of those modules are used and reboot :

```
set_module.sh 2AE|2BC
reboot
```

- Refer to [Section 7](#) to test/verify Wi-Fi and Bluetooth functionality.

Figure 16: i.MX 6UL 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

By selecting one of two DTB files, the user can configure for either the onboard Type 1CX module or use the M.2 connector. As shown in **Figure 17**, the i.MX 8MQuad EVK provides a secondary Wi-Fi/BT solution on the underside via a M.2 connector. The uSD-M.2 adapter provides WLAN-PCIe, BT-UART, control signals, and optionally BT-PCM. Currently the only supported M.2 EVB is Type 1XA (WLAN-PCIe).

- 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.
- If not using onboard Type 1CX module, connect Embedded Artists Type 1XA M.2 EVB to M.2 connector as shown in **Figure 17**. Attach two dual-band (2.4/5 GHz) 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.
- Prepare microSD card to boot platform per [Section 4.1](#). Insert microSD card, power on platform and interrupt at u-boot. Set DTB configuration with “fdt_file” parameter to either configure for Wi-Fi/BT M.2 EVB or onboard Type 1CX module. You can check the available DTB files using the command “fatls mmc 1”. Now save the u-boot configuration and boot the platform:

```
setenv fdt_file fsl-imx8mq-evk-pcie1-m2.dtb
  (OR fsl-imx8mq-evk.dtb)
saveenv

# Boot kernel
boot
```

- After the kernel boots, invoke “set_module.sh 2AE|2BC” if either of those modules are used and reboot:

```
set_module.sh 2AE|2BC
reboot
```

- Refer to [Section 7](#) to test/verify Wi-Fi and Bluetooth functionality.

Figure 17: i.MX 8MQuad with Type 1CX (bottom view)



6.2 Bringing up Wi-Fi/BT on i.MX 8M Mini/Nano EVKs (Onboard 1MW or M.2)

The 8M Mini EVK (8MMINID4-EVK) and 8M Nano EVK (8MNANOD4-EVK) both have Type 1MW module down. Both 8M Mini EVKs (8MMINILPD4-EVK and 8MMINID4-EVK) have a WLAN-PCIe M.2 connector on the baseboard – with no connection for Bluetooth-UART. See **Figure 18** for upside-down EVK view showing M.2 connector (current supported M.2 EVB is Type 1XA). The steps below detail how to bring up the onboard Type 1MW module or the Wi-Fi/BT M.2 EVB.

- 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.
- For Wi-Fi M.2 EVB option, connect Embedded Artists Type 1XA M.2 EVB to M.2 connector as shown in **Figure 18**. Attach two dual-band (2.4/5 GHz) 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.
- Prepare microSD card to boot platform per [Section 4.1](#). Insert microSD card, power on platform and interrupt at u-boot. Set DTB configuration with “fdt_file” parameter for correct platform per **Table 7**. Note that the 8MMINID4-EVK uses same DTB file for both onboard 1MW and M.2 EVB. Check the available DTB files by invoking “fatls mmc 1”. Now save u-boot configuration and boot the platform:

```
setenv fdt_file imx8mm-evk.dtb
(OR imx8mm-ddr4-evk.dtb OR imx8mn-ddr4-evk.dtb)
saveenv

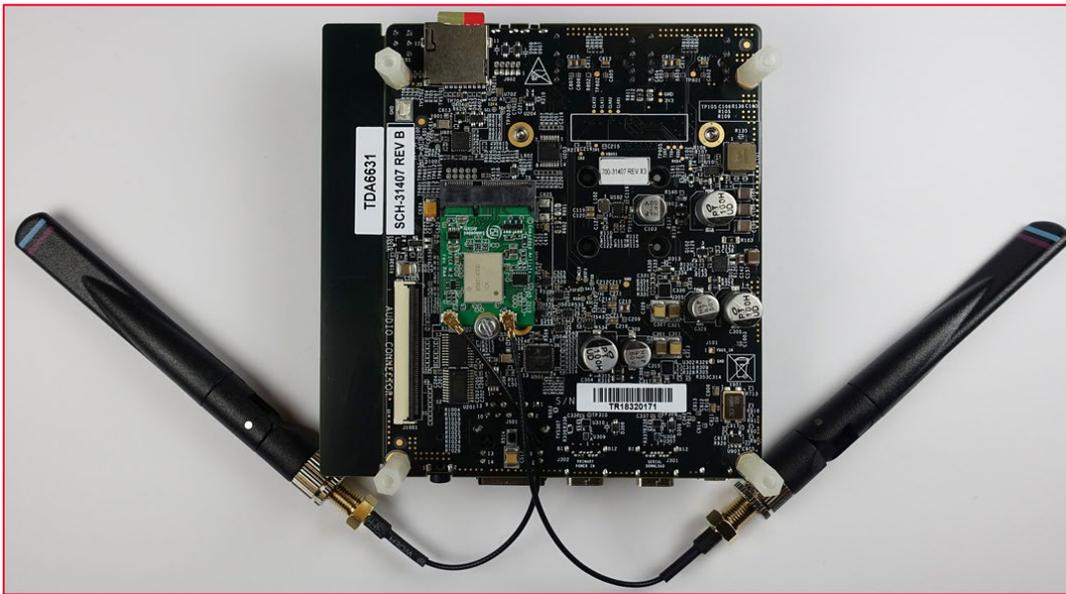
# Boot kernel
boot
```

- After the kernel boots, “set_module.sh 2AE|2BC” if either of those modules are used and reboot:

```
set_module.sh 2AE|2BC
reboot
```

- Refer to [Section 7](#) to test/verify Wi-Fi and Bluetooth functionality.

Figure 18: i.MX 8M Mini with Type 1XA (bottom view)



6.3 Bringing up Wi-Fi/BT on i.MX 8M Mini/Nano EVKs (uSD-M.2 Adapter)

There are two configurations for adding uSD-M.2 Adapter interconnect to i.MX 8M Mini/Nano EVK: WLAN/Bluetooth (**Figure 20**) and WLAN (**Figure 19**). The WLAN configuration is quite simple: just insert inverted uSD-M.2 Adapter with Wi-Fi/M.2 EVB attached into microSD slot. The WLAN/BT configuration requires additional jumper cables for Bluetooth-UART and WLAN/BT control signals.

Figure 19: i.MX 8M Mini/Nano EVK with uSD-M.2 Adapter (WLAN Only)

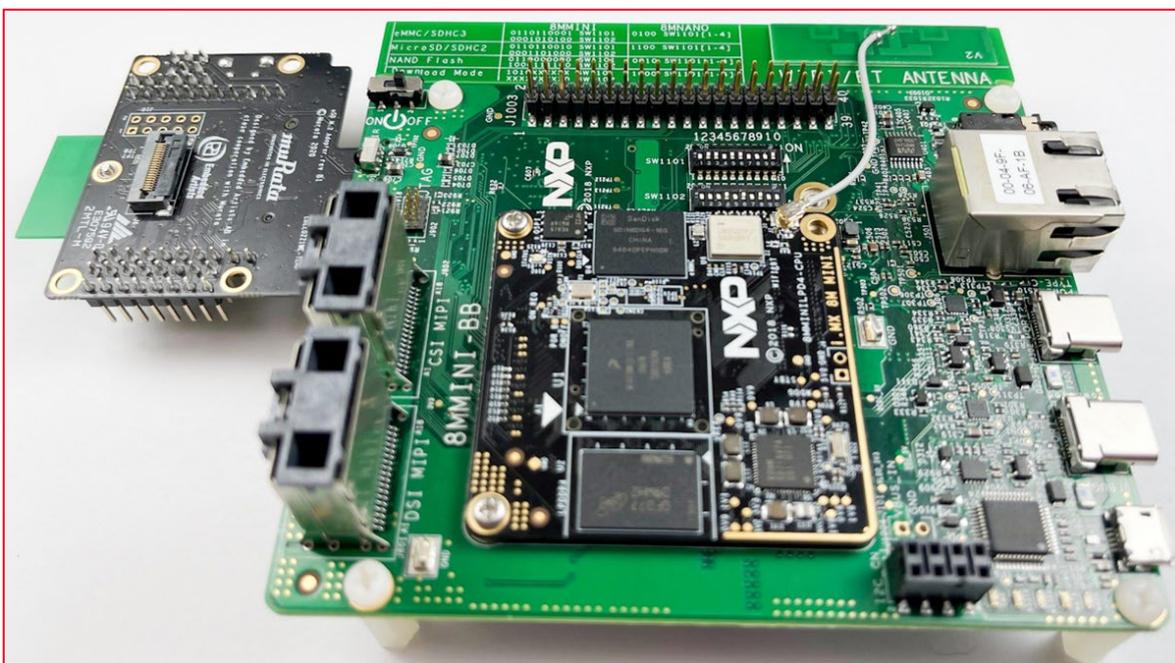
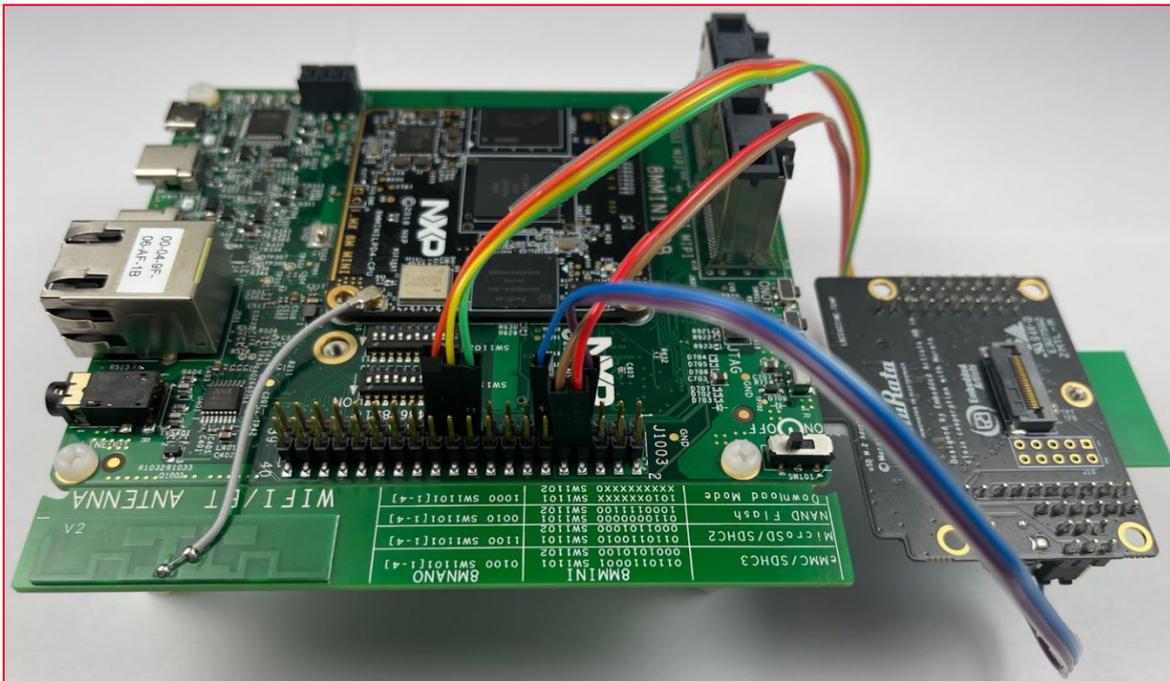


Figure 20: i.MX 8M Mini/Nano EVK with uSD-M.2 Adapter (WLAN/Bluetooth)



The only i.MX 8M Mini/Nano EVKs supported in this section have eMMC onboard: 8MMINILPD4-EVK and 8MNANOD4-EVK. Per [Section 4.2](#), the onboard eMMC is flashed so we can connect Murata's uSD-M.2 Adapter with Wi-Fi/BT M.2 EVB. For Bluetooth-UART and additional WLAN/BT control (WL_REG_ON, BT_REG_ON, WL_HOST_WAKE) signals interconnect, there are additional 6~7" F/F Jumper cables (with optional offsets) that are needed as referenced below. However, customers only needing WLAN-SDIO connectivity can insert uSD-M.2 Adapter (with Wi-Fi/BT M.2).

- 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.
- On uSD-M.2 Adapter, set Jumper J1 to position 2-3 (VBAT from microSD connector). Make sure Jumper J11 is removed (BT core enabled). Note Blue LED2 is not illuminated for 1.8V VIO.
 - For Rev B1/B2 adapter, install J13/J12 in 1-2/1-2 positions, respectively for 1.8V VIO.
 - For legacy Rev A adapter, make sure J12 is not installed for 1.8V VIO.
- Connect the Wi-Fi/BT M.2 EVB to the M.2 connector on the uSD-M2 Adapter per [Section 8.1](#). Attach the uSD-M.2 Adapter/M.2 to EVK and tape the uSD Adapter-EVK connection per [Section 8.3](#).
- For full Wi-Fi/BT configuration (Bluetooth-UART and WLAN/BT control lines), connect seven (7) jumper wires from the uSD-M.2 adapter to the i.MX 8M Mini/Nano EVK per [Table 10](#). Refer to [Figure 21](#), [Figure 22](#), [Figure 23](#) and [Figure 24](#) for additional details: colored wires are shown in these figures so users may more easily follow along. Note there is a clearance issue when attaching "normal" jumper wires. Murata recommends two different approaches:
 - Use low-profile jumper wires (like Digi-Key part number 1988-1178-ND) which can be bent at right-angles – see [Figure 25](#), and [Figure 26](#). The connectors on uSD-M.2 Adapter need to be bent at 45° angle so that there is no interference with default NXP i.MX standoffs.

OR:

- Use “normal” jumper wires (like Digi-Key part number 1568-1513-ND) with additional standoffs (like Digi-Key part number RPC3570-ND). Referring to **Figure 27**, the additional standoffs (which screw into existing NXP i.MX EVK standoffs) add necessary height to platform so there is no interference with jumper wire connectors.



For WLAN-only, no jumper cables need to be connected. For customers only needing to evaluate Wi-Fi (**Figure 19**), this provides a faster interconnect option. There is a Power-On-Reset (POR) circuit (on uSD-M.2 Adapter) for driving WL_REG_ON high – signal which enables WLAN core, thereby only requiring host interface to drive the WLAN-SDIO interface (SDIO in-band interrupts only).

- Per [Section 4.2](#), flash eMMC on i.MX 8M Mini/Nano EVK; and then configure DIP switch settings for eMMC boot.
- Power on the platform and interrupt at u-boot. Set DTB configuration with “fdt_file” parameter (“oob” string configures OOB IRQ configuration; see [Section 3.4.1](#) for more details). OOB interrupt configuration will only work if additional jumper wires are attached (WL_HOST_WAKE). You can check the available DTB files using the command “fatls mmc 2”. After setting DTB, save the u-boot configuration and boot the platform:

```
setenv fdt_file imx8mm-evk-usd-m2<-oob>.dtb
(OR imx8mn-evk-usd-m2<-oob>.dtb)
saveenv

# Boot kernel
boot
```

- After the kernel boots, invoke “set_module.sh 2AE|2BC” if either of those modules are used and reboot:

```
set_module.sh 2AE|2BC
reboot
```

- Refer to [Section 7](#) to test/verify Wi-Fi and Bluetooth functionality.

Table 10: i.MX 8M Mini/Nano EVK Jumper connections to uSD-M.2 Adapter

Signal Name	uSD-M.2 Adapter Header/Pin	i.MX 8M Mini/Nano EVK J1003 Pin	i.MX 8M Mini/Nano EVK Signal
BT_UART_TX	J9 / Pin 1	10	UART3_RXD
BT_UART_RX	J9 / Pin 2	8	UART3_TXD
WL_REG_ON	J9 / Pin 3	19	ECSPI2_MOSI
BT_REG_ON	J9 / Pin 4	21	ECSPI2_MISO
WL_HOST_WAKE	J9 / Pin 5	23	ECSPI2_SCLK
BT_UART_RTS	J8 / Pin 3	11	UART RTS line
BT_UART_CTS	J8 / Pin 4	7	UART CTS line

Figure 21: Cable connections on i.MX 8M Mini EVK (Even Number Connector View)

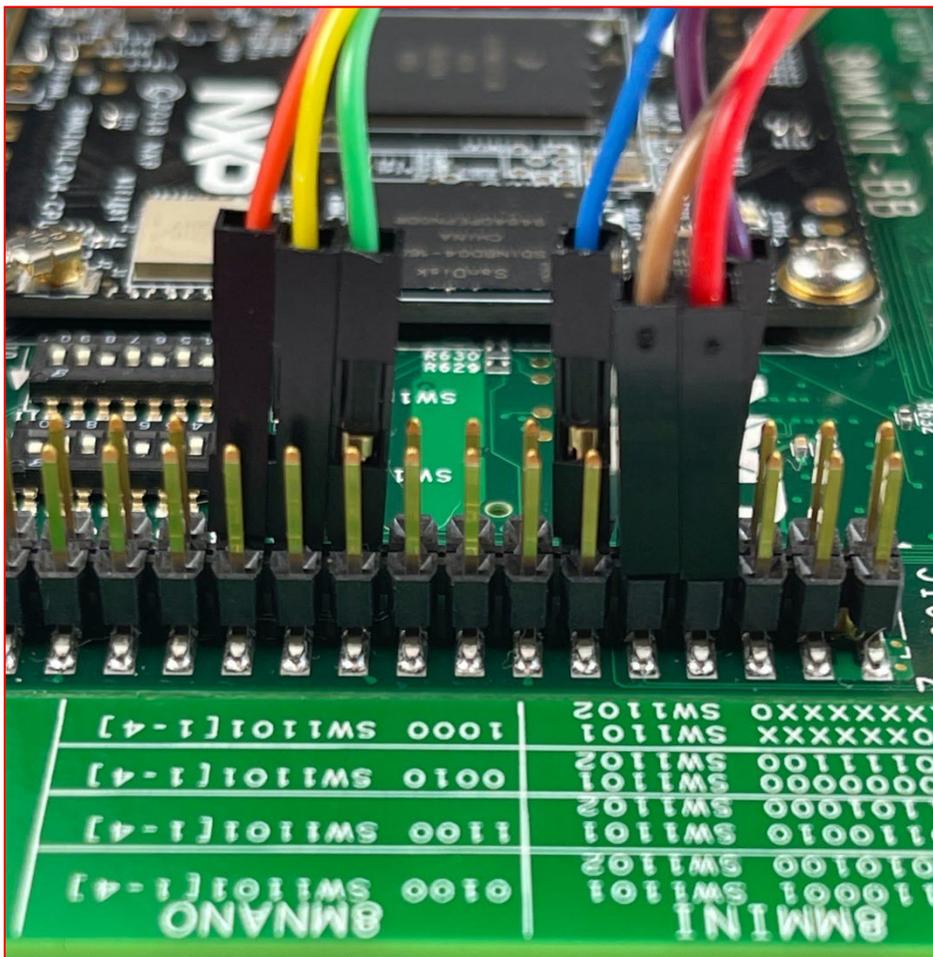


Figure 22: Cable connections on i.MX 8M Mini EVK (Odd Number Connector View)

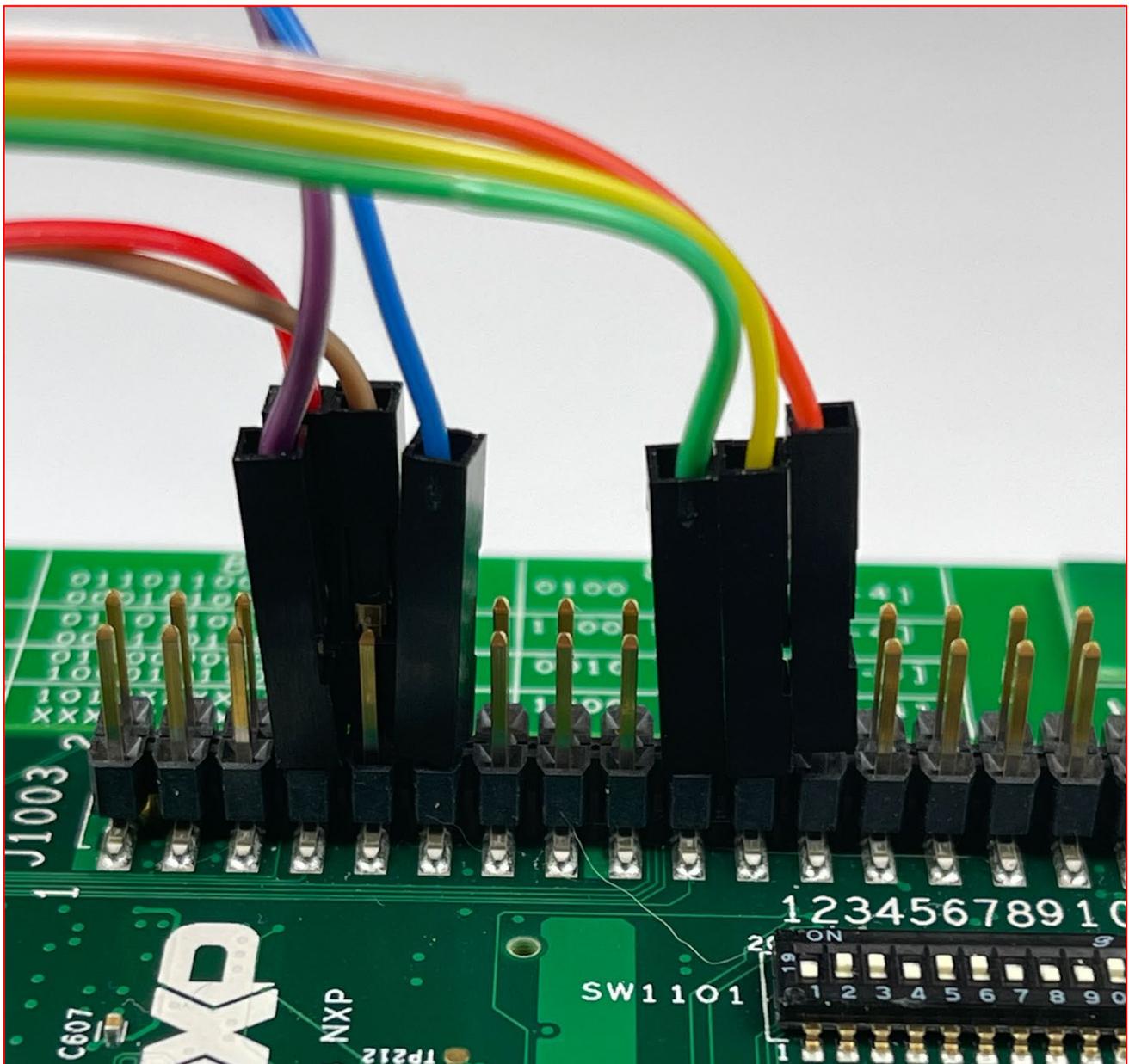


Figure 23: Cable connections on uSD-M.2 Adapter (J9 Header)

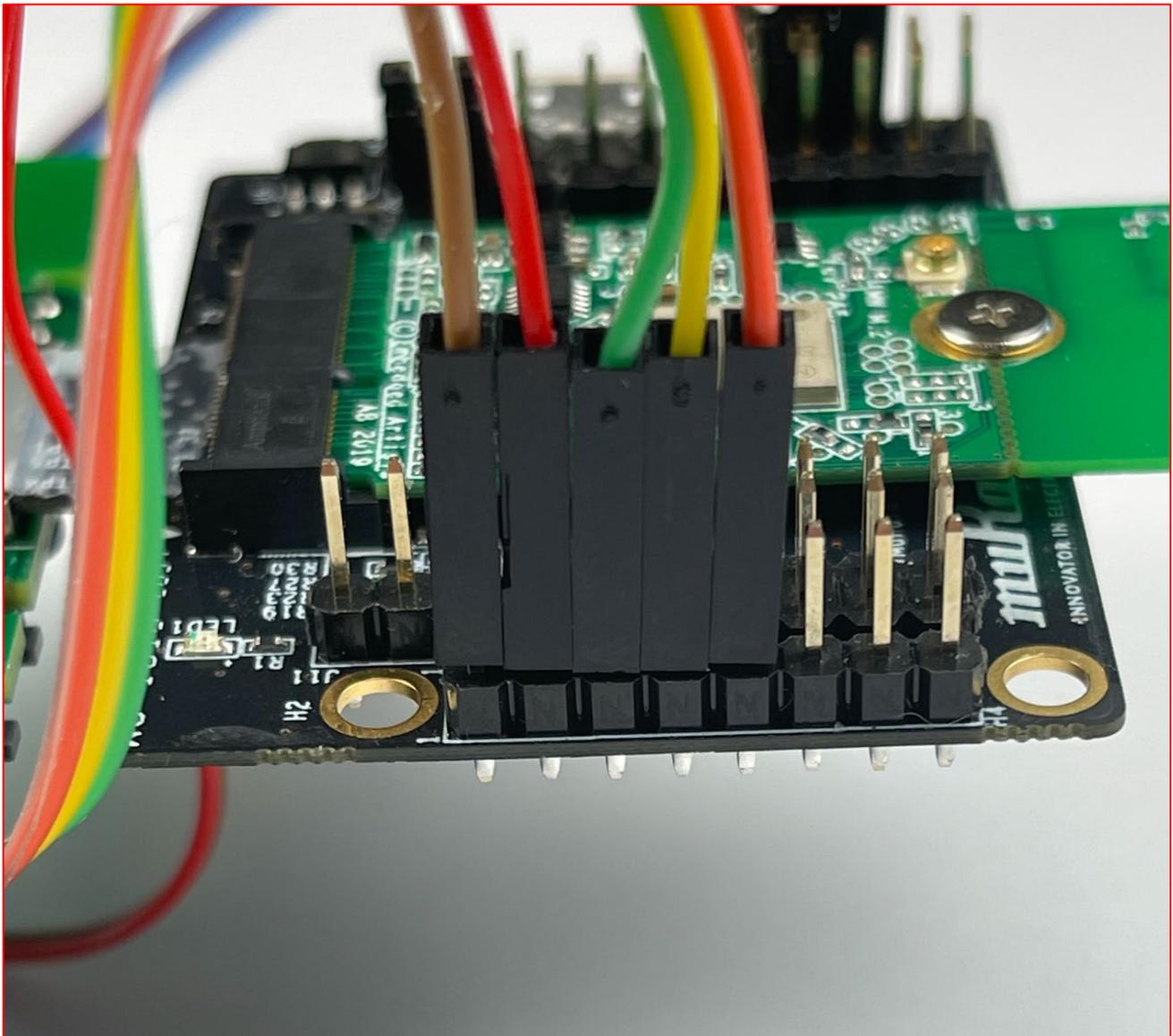


Figure 24: Cable connections on uSD-M.2 Adapter (J8 Header)

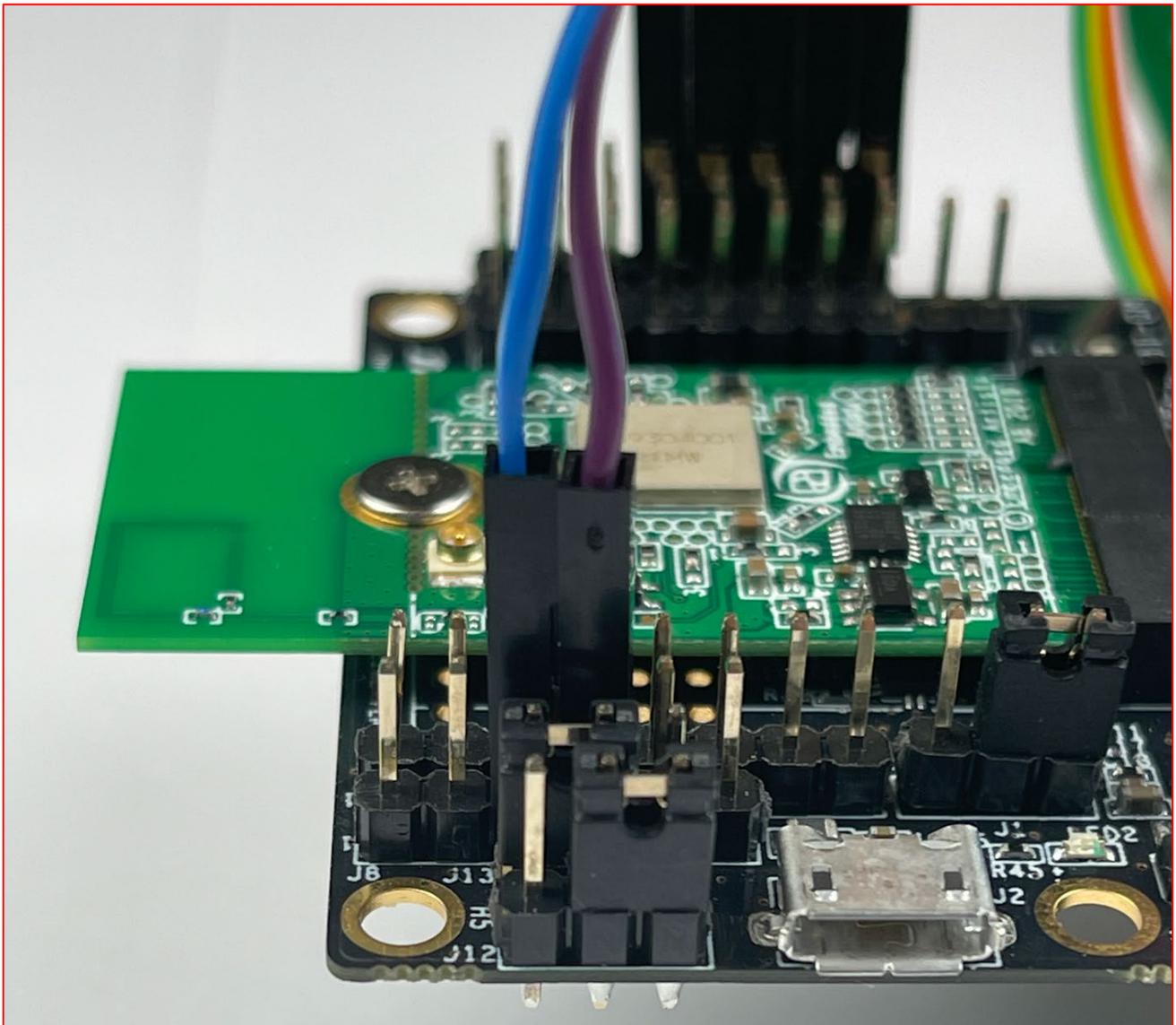


Figure 25: Low-Profile Jumper Wires (Digi-Key part number 1988-1178-ND)



Figure 26: NXP i.MX EVK with Low-Profile Jumper Wires

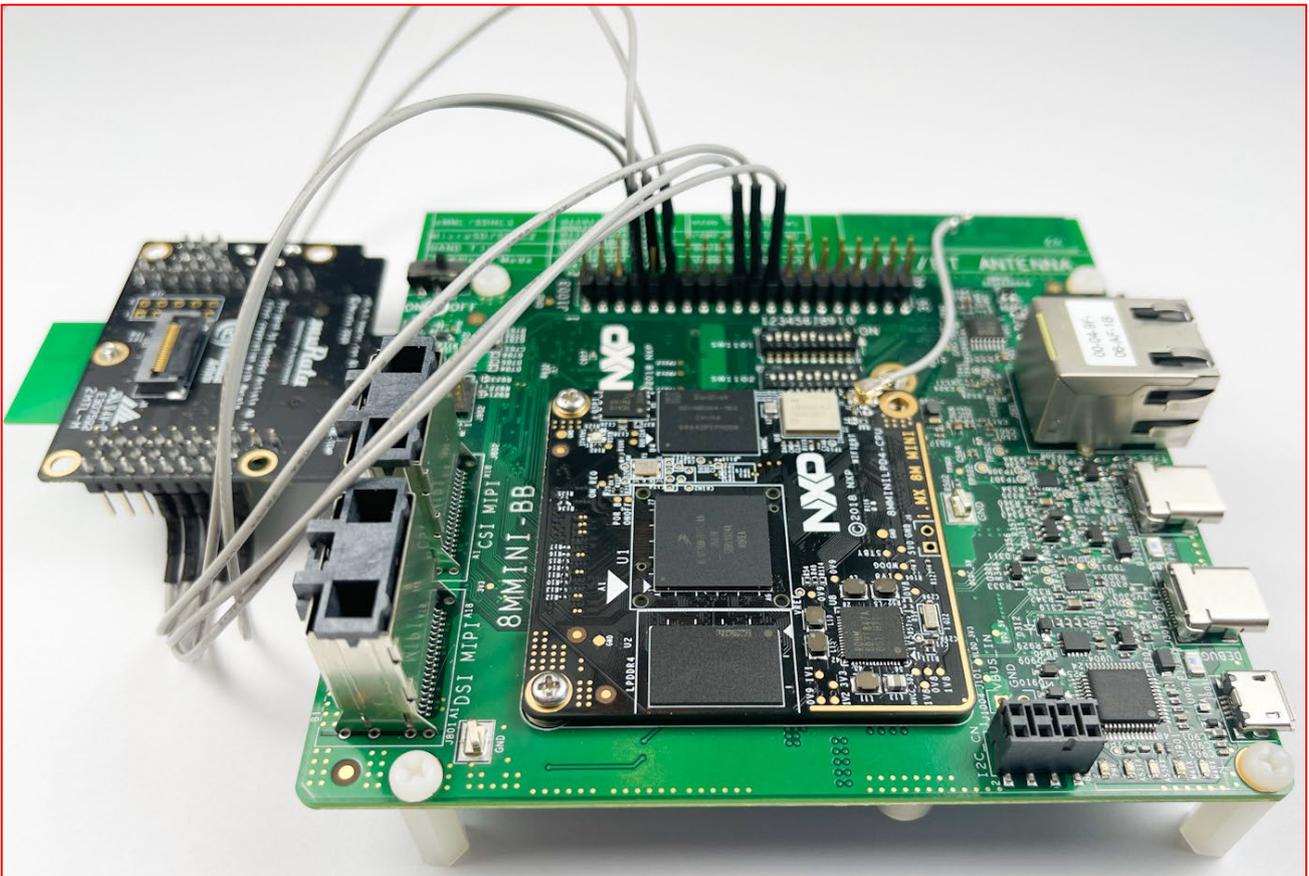
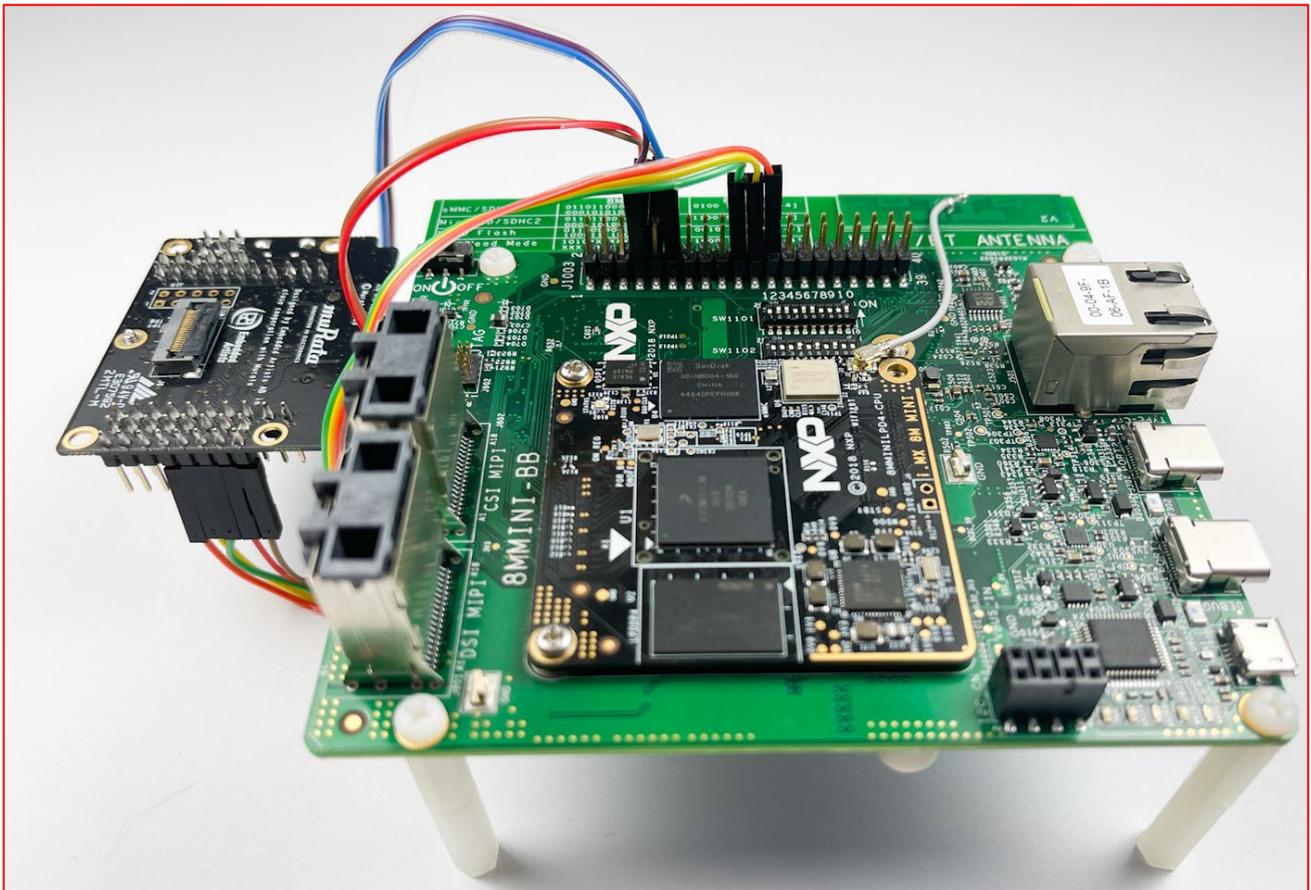


Figure 27: Additional Hex Standoff (Digi-Key part number RPC3570-ND)



7 Test/Verification of Wi-Fi and Bluetooth

Now the kernel should be booting correctly on the NXP i.MX platform with Murata module being correctly initialized (correct DTB configured, and optionally Infineon software configuration selected with “set_module.sh”). Next steps are to verify Wi-Fi and Bluetooth functionality. The Murata-customized i.MX images include all the necessary files to support Wi-Fi and Bluetooth bring-up/testing/verification (with exception of WLAN regulatory testing – see [Section 7.1.10](#)). The relevant folders and files are summarized in **Table 11**.

There is an “FMAC version” column in **Table 11**: this is used to differentiate any files that are specific to an Infineon “FMAC” driver release. In this table we can see that the file naming convention for WLAN firmware/NVRAM/CLM Blob depends on version of “FMAC”. For “FMAC” versions “spiga” and later, the folder is “/lib/firmware/cypress” with an initial filename string of “cyfmac”.

Table 11: Embedded Wi-Fi/Bluetooth Files

Folder / Filename	FMAC ver	Details
/lib/firmware/cypress/cyfmac<chipset>-sdio.bin	spiga+	“fmac” firmware file for WLAN SDIO chipset/module.
/lib/firmware/cypress/cyfmac<chipset>-sdio.txt	spiga+	“fmac” NVRAM file for WLAN SDIO chipset/module.
/lib/firmware/cypress/cyfmac<chipset>-sdio.clm_blob	spiga+	“fmac” regulatory binary for WLAN SDIO chipset/module.
/lib/firmware/cypress/cyfmac<chipset>-pcie.bin	spiga+	“fmac” firmware file for WLAN PCIe chipset/module.
/lib/firmware/cypress/cyfmac<chipset>-pcie.clm_blob	spiga+	“fmac” regulatory binary for WLAN PCIe chipset/module.
/lib/firmware/cypress/cyfmac<chipset>-pcie.txt	spiga+	“fmac” NVRAM file for WLAN PCIe chipset/module.
/etc/firmware/BCM43430A1_001.002.009.0159.0528.1DX.hcd	any	Type 1DX (CYW4343W) Bluetooth patch file.
/etc/firmware/BCM4345C0_003.001.025.0187.0366.1MW.hcd	any	Type 1MW (CYW43455) Bluetooth patch file.
/etc/firmware/BCM43012C0_003.001.015.0303.0267.1LV.sAnt.hcd	any	Type 1LV (CYW43012) Bluetooth patch file for shared antenna.
/etc/firmware/BCM43012C0_003.001.015.0300.0266.1LV.dAnt.hcd	any	Type 1LV (CYW43012) Bluetooth patch file for dedicated antenna.
/etc/firmware/BCM4373A0.2AE.hcd	any	Type 2AE (CYW4373E) Bluetooth patch file.
/etc/firmware/BCM4373A0.2BC.hcd	any	Type 2BC (CYW4373) Bluetooth patch file.
/etc/firmware/BCM4359D0_004.001.016.0241.0275.2BZ.sAnt.hcd	any	Type 2BZ (CYW54590) Bluetooth patch file for shared antenna.
/etc/firmware/BCM4359D0_004.001.016.0241.0274.2BZ.dAnt.hcd	any	Type 2BZ (CYW54590) Bluetooth patch file for dedicated antenna.
/etc/firmware/CYW4343A2_001.003.016.0031.0000.1YN.hcd	any	Type 1YN (CYW43439) Bluetooth patch file.
/etc/firmware/BCM4359D0_004.001.016.0241.0275.1XA.sAnt.hcd	any	Type 1XA (CYW54591) Bluetooth patch file for shared antenna.
/etc/firmware/BCM4359D0_004.001.016.0241.0274.1XA.dAnt.hcd	any	Type 1XA (CYW54591) Bluetooth patch file for dedicated antenna.
/usr/sbin/wl	any	“wl” tool used for RF testing, initial bring-up and debug.
/usr/sbin/iw	any	Linux “iw” executable.
/usr/sbin/wpa_supplicant	any	WPA supplicant executable.
/usr/sbin/wpa_cli	any	WPA CLI tool.
/usr/bin/wpa_passphrase	any	WPA Passphrase generator.
/etc/wpa_supplicant.conf	any	WPA supplicant configuration file.
/usr/sbin/hostapd	any	Hostapd – manages wireless link in Soft AP mode.
/usr/sbin/hostapd_cli	any	Hostapd CLI tool.

/etc/hostapd.conf	any	Hostapd configuration file.
/etc/udhcpd.conf	any	DHCP server configuration file.
/usr/bin/hciattach	any	“hciattach” binary – used for initializing Bluetooth.
/usr/bin/hciconfig	any	“hciconfig” binary – used for configuring Bluetooth.
/usr/bin/hcitool	any	“hcitool” binary – used for controlling Bluetooth interface.
/usr/bin/iperf3	any	“iperf” throughput test tool (iperf3 is latest version).

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:

```
# Set your favorite row and column width here
$ stty rows 80 cols 132

# Invoke this command after “stty”
$ export TERM=ansi
```

7.1.2 Set Module Script

Murata has included a script file in the customized Linux image which configures the wireless drivers for Type 2AE module. By default the Linux image is configured to support all Murata modules except the Type 2AE. This is because the Type 2BC and Type 2AE share the same chipset and the kernel automatically selects the Type 2BC support when the chipset is detected. In order to set up Type 2AE instead of Type 2BC, the `set_module` script can be used. The same script can be used to switch back to Type 2BC support. The usage is as shown below:

```
set_module.sh 2AE|2BC
```

The “`set_module.sh`” script file performs the following functions:

- Points to correct NVRAM, CLM Blob and Firmware for the module selected.

The “`set_module.sh 2AE|2BC`” step should have already been done in [Section 5](#)  or [Section 6](#) .



The script file only needs to be run once when switching the module to 2BC or 2AE – needs a reboot afterwards.

7.1.3 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/PCIe bus. As an example, we will bring up Wi-Fi when using following configuration:

- NXP i.MX 6ULL EVK
- Murata’s uSD-M.2 Adapter
- Embedded Artists’ Type 1MW M.2 Module (EVB)

- Murata i.MX image compiled for i.MX 6ULL

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
mmc0: SDHCI controller on 2190000.usdhc [2190000.usdhc] using ADMA
...
cfg80211: Loading compiled-in X.509 certificates for regulatory database
cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
mmc0: queuing unknown CIS tuple 0x80 (2 bytes)
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)
brcmfmac: brcmf_fw_alloc_request: using cypress/cyfmac43455-sdio for chip
BCM4345/6
usbcore: registered new interface driver brcmfmac
brcmfmac: brcmf_fw_alloc_request: using cypress/cyfmac43455-sdio for chip
BCM4345/6
brcmfmac: brcmf_c_preinit_dcmds: Murata Customized Version: imx-zeus-
zigma r1.0;
brcmfmac: brcmf_c_preinit_dcmds: Firmware: BCM4345/6 wl0: May 22 2020 21:24:34
version 7.45.214 (9c83742 CY) FWID 01-59feefd4
```

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

- “brcmfmac” string identifies “FMAC” driver log messages
- “cyfmac43455-sdio” 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” indicates specific version of firmware being loaded by “FMAC”. In this case the firmware version is 7.45.214.

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" interface is UP. WLAN MAC Address is shown
wlan0      Link encap:Ethernet  HWaddr 00:9D:6B:A6:EE:76
           UP BROADCAST MULTICAST  MTU:1500  Metric:1
           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)
```



No IP address is assigned yet to the “wlan0” interface. That will be done later in [Section 7.1.7](#).

7.1.4 STA/Client Mode: Scan for Visible Access Points

In this section, two different/simple methods for scanning are presented: one uses the Infineon “wl” tool (Infineon-specific tool reserved mainly for RF testing); the other uses the preferred Linux “iw” command. If you do not 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”, “iperf3”, 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.

7.1.4.1 Using “iw” Linux Command

“iw” is the default Linux command line tool for controlling/querying a WLAN interface. For more information on “iw” too, see [here](#) . In the following example of listing WLAN devices and performing a scan, there is one active AP with a SSID of “Murata_5G”. Here are the expected results:

```
# List available WLAN devices
$ iw dev
phy#0
    Interface wlan0
        ifindex 7
        wdev 0x1
        addr 00:9d:6b:a6:ee:76
        type managed
        channel 34 (5170 MHz), width: 20 MHz, center1: 5170 MHz
        txpower 31.00 dBm

# Perform scan on "wlan0" interface
$ iw dev wlan0 scan
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.
```

7.1.4.2 Using “wl” Tool

The Infineon “wl” tool is a powerful tool which allows the user to query/configure any number of radio characteristics. It is primarily used for RF testing/evaluation and regulatory certification. 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 document:

- [“WL Tool for Embedded 802.11 Systems”](#)  on [Infineon Linux Support Portal](#) .

Now that Wi-Fi interface is up and running (having loaded the default “cyfmac<chipset>-sdio.bin” – WLAN firmware), let us 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 this 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
# Broadcast SSID, with RSSI (received signal strength) one line below
SSID: "Murata_5G"
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)
  # Channel Number
  Primary channel: 36
  # 802.11ac bandwidth (40 MHz in this case)
HT Capabilities: 40Mhz SGI20 SGI40
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:dd7c0050f204104a0001101044000102103b000103104700101b52c4d5ffb10ad163f39b9
1a979382c1021000d4e4554474541522c20496e632e102300055236333030102400055236333030
10420004343533361054000800060050f2040001101100055236333030100800022008103c00010
31049000600372a000120
VS_IE:dd090010180200001c0000
VS_IE:dd180050f20201018c0003a4000027a400004243bc0062326600
# ..... More SSID listings follow here.
```



“oversize return buffer” log message (right after “wl scanresults” command) can be ignored.

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



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 7.1.7](#) where connectivity testing is performed.

7.1.5.1 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
  tx bitrate: 24.0 Mbit/s
  bss flags:
  dtim period: 2
  beacon int: 100
```

7.1.5.2 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>:xxxxxx] [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”:

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

Check status of connection with “wl assoc” command:

```
# Check association status
$ wl assoc
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:dd310050f204104a0001101044000102104700101b52c4d5ffb10ad163f39b91a979382c1
03c0001031049000600372a000120
VS_IE:dd090010180200001c0000
VS_IE:dd180050f20201018c0003a4000027a400004243bc0062326600
VS_IE:dd7c0050f204104a0001101044000102103b000103104700101b52c4d5ffb10ad163f39b9
1a979382c1021000d4e4554474541522c20496e632e102300055236333030102400055236333030
10420004343533361054000800060050f2040001101100055236333030100800022008103c00010
31049000600372a000120
  
```

7.1.6 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.



WPA supplicant must be up and running.

7.1.6.1 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:

```
# Tear down any existing network connections
> remove_network all
OK
<3>CTRL-EVENT-DISCONNECTED bssid=b0:00:b4:65:e0:60 reason=3 locally_generated=1

# Check status
> 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

# Initiate a scan
> scan
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>CTRL-EVENT-NETWORK-NOT-FOUND

# List results of scan
> scan_results
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 a network. This returns integer value which is then used for setting
parameters.
> add_network
0

# Set SSID to "Murata_5G"
> set_network 0 ssid "Murata_5G"
OK

# Set WPA passphrase
> set_network 0 psk "your_passphrase"
OK

# Enable network connection. If ssid and passphrase set correctly, connection
will be established.
> enable 0
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>Trying to associate with SSID 'Murata_5G'
# Connection established
<3>Associated with 84:1b:5e:f6:a7:60
```

```

<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

# Now verify that connection is established
> status
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 current configuration: this overwrites "/etc/wpa_supplicant.conf" file!
> save_config
OK

# Exit wpa_cli interactive mode
> quit

```

Now let us 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"
}

```



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.

7.1.6.2 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 7.1.6.1](#) ) 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
```

7.1.7 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):

```
# Command to invoke DHCP client and obtain IP address.
$ udhcpc -i wlan0
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/udhcpd.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
# Enter <CTRL-C> to terminate ping session
^C
--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 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](#) .

7.1.8 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. Note that 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

# Let's remove any network association
> remove_network all
OK
<3>CTRL-EVENT-DISCONNECTED bssid=84:1b:5e:f6:a7:60 reason=3 locally_generated=1

# Check status now
> > status
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

# Add P2P Group
> p2p_group_add
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 "wpa_cli" tool
> > quit
```

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
...
# New P2P interface
p2p-wlan0-0 Link encap:Ethernet HWaddr ba:d7:af:56:e1:fc
  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)

# Existing "wlan0" interface
wlan0 Link encap:Ethernet HWaddr b8:d7:af:56:61:fc
  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.

7.1.9 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 Soft AP 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 UNINITIALIZED: 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"
```

7.1.10 WLAN Manufacturing, RF or Regulatory 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 and included on Murata’s GitHub. To switch over to manufacturing test firmware, the user needs to contact Murata to obtain the manufacturing test firmware files. Here are the necessary steps:

- Obtain manufacturing test firmware tar ball. If you need assistance, please post on [Murata’s Community Support Forum](#) .
- After accessing root file system, “cd” to “lib/firmware/cypress” folder.
- Switch user to “root”.
- Create “mfgtest” and “production” sub-folders in “lib/firmware/cypress” folder.
- Backup existing “*.bin” and “*.clm_blob” files to “production” sub-folder.
- Copy manufacturing test firmware files (from “mfgtest” sub-folder) into “lib/firmware/cypress” folder.
- Reboot the platform: verify that “WLTEST” is logged when “FMAC” driver loads.

After accessing root file system, create sub-folders for production and manufacturing test firmware:

```
$ cd <path to mounted rootfs>/lib/firmware/cypress
$ 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/
```

Now boot i.MX platform and note firmware log message:

```
brcmfmac: brcmf_c_preinit_dcmts: 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/cypress”:

```
$ cd /lib/firmware/cypress
$ 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
```



The WLAN firmware version number 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/cypress
$ 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 “hcidtool 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 11** for more details. Here is the default command sequence to verify Bluetooth functionality:

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



If the platform is correctly configured, then the user should ONLY have to execute “hciattach” command followed by “hciconfig”, and “hcidtool”.

Table 12 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/ttymx1 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 12: GPIO and UART Settings for Bluetooth Tests

i.MX Platform	GPIO/UART Configuration	Notes
i.MX 8MQuad EVK	GPIO69; UART3	Type 1CX soldered down and M.2 EVB.
i.MX 8M Mini/Nano EVK (1MW)	GPIO37; UART1	Type 1MW soldered down.
i.MX 8M Mini/Nano EVK (uSD)	GPIO140; UART3	uSD-M.2 Adapter interconnect with M.2 EVB.
i.MX 6Q(P)/DL SDB	GPIO2; UART5	uSD-M.2 Adapter interconnect with M.2 EVB.
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(l) 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
# SKIP following step for i.MX 6UL/ULL EVK
echo out > /sys/class/gpio/gpio[GPIO #]/direction
echo 0 > /sys/class/gpio/gpio[GPIO #]/value
sleep 0.1
echo 1 > /sys/class/gpio/gpio[GPIO#]/value
hciattach /dev/ttymx[UART# -1] bcm43xx 3000000 flow -t 20
hciconfig hci0 up
hcitool scan
```

Regarding some more details on modifying DTS(l) 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 the [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;

    // Comment out "resets" line on BT UART descriptor
    // resets = <&modem_reset>;
    status = "okay";
};
```

8 Murata's uSD-M.2 Adapter

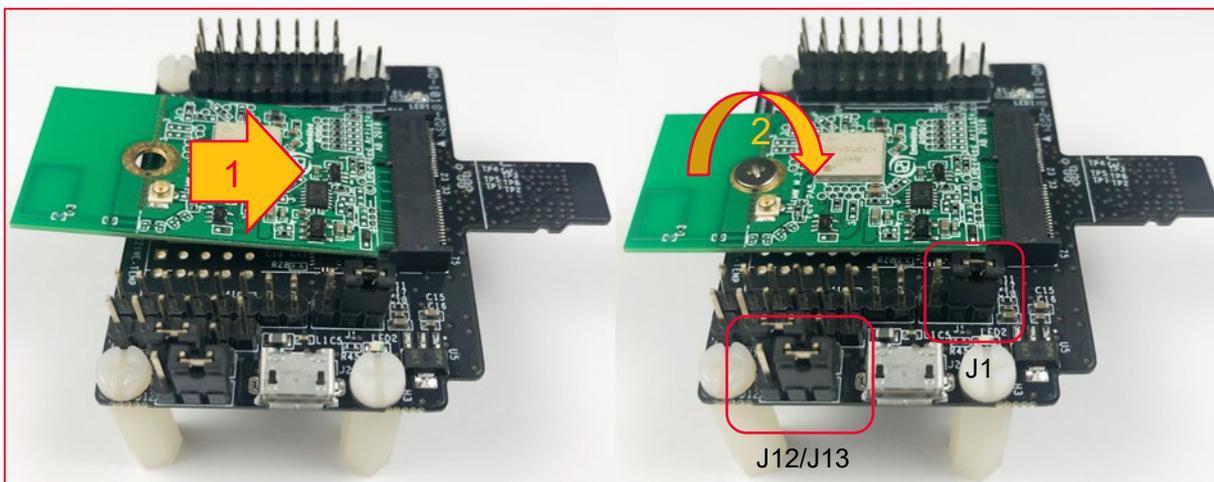
This section describes the following.

- Connection of Wi-Fi/BT M.2 EVB to uSD-M.2 adapter
- Configuration of uSD-M.2 adapter jumpers for correct VIO signaling
- Securing uSD-M.2 adapter to NXP i.MX EVK
- High-Level description of uSD-M.2 adapter

8.1 Connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter

When connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter Rev B1 (**Figure 28**), make sure to (#1) firmly insert it before using M.2 screw to (#2) secure it in place. Important Jumpers (J12, J13, and J1) are highlighted.

Figure 28: Connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter

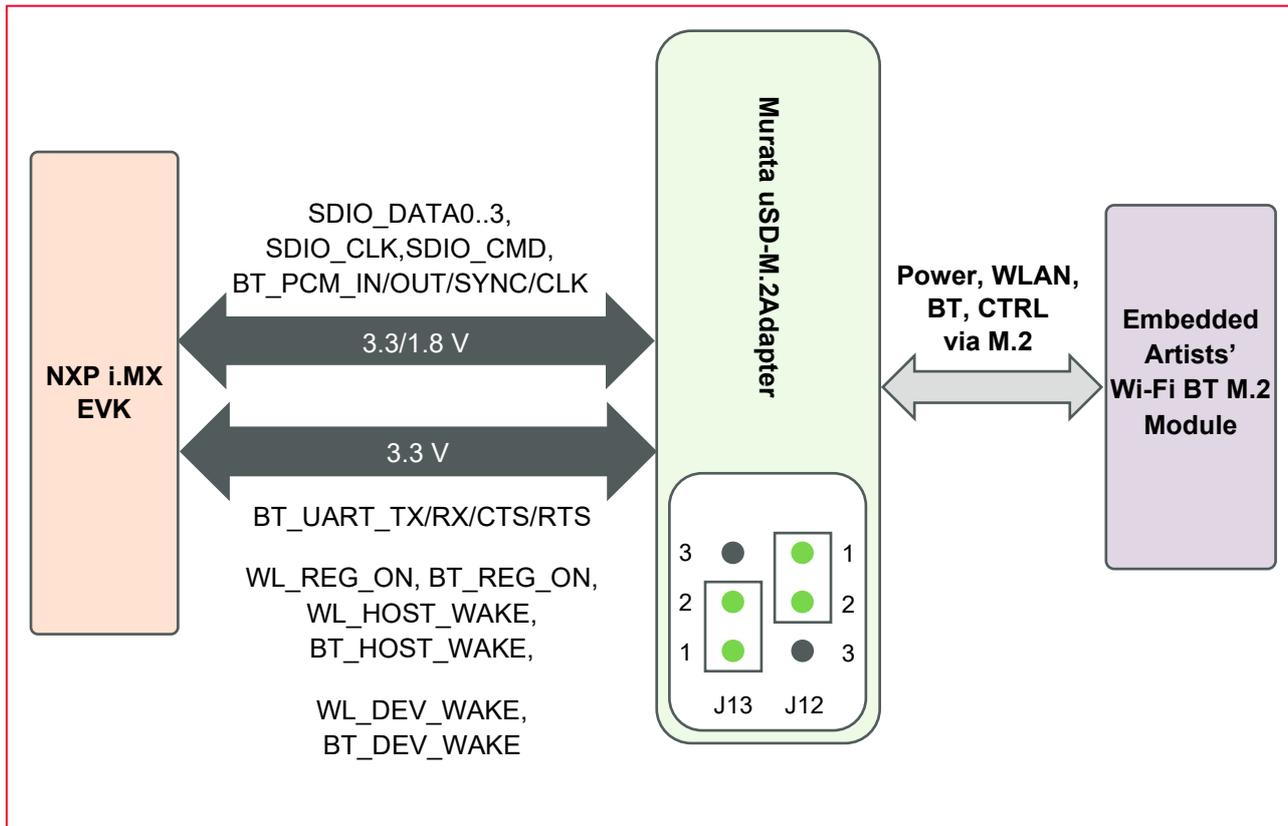


8.2 Configuring uSD-M.2 Adapter Jumpers for Correct VIO Signaling

Figure 29 shows a block diagram highlighting the Host (i.MX EVK) and Wi-Fi/BT M.2 EVB VIO signaling voltages. All i.MX EVKs (i.MX 8M Mini/Nano & i.MX 6UL(L) EVKs) have the Murata uSD-M.2 Adapters' J13/J12 jumpers set to 1-2/1-2 positions respectively for the default configuration:

- Host WLAN-SDIO VIO = 1.8V VIO
- Host BT-UART = 3.3V VIO
- Host WLAN/BT control signals = 3.3V VIO

Figure 29: Host/M.2 IO Voltage Level Shift Options on Rev B1 Adapter



8.3 Securing uSD-M.2 Adapter to NXP i.MX EVK

On both legacy NXP i.MX 6 EVKs and the newer i.MX 8 EVKs, a common issue that customers run into is an unreliable uSD/SD electrical connection when using Murata's uSD-M.2 Adapter. The poor interconnect is caused by two issues: push-push (micro) SD card connectors on NXP i.MX EVKs; and low friction interface between the uSD-M.2 Adapter and uSD-SD Adapter Card.

To properly secure the uSD-M.2 Adapter interconnect on the i.MX 6 EVKs, Murata **strongly recommends** to simply tape the uSD Adapter-SD Card connection and the SD Card-EVK connection as shown in **Figure 30**. Note that taping the SD Card-EVK connection makes the platform a little less flexible to work with. However, removing and re-applying clear tape is straightforward.

To properly secure the uSD-M.2 Adapter interconnect on the i.MX 8 EVKs, Murata **strongly recommends** to simply tape the uSD Adapter-EVK connection as shown in **Figure 31**. Note that taping the uSD Adapter-EVK connection makes the platform a little less flexible to work with. However, removing and re-applying clear tape is straightforward.

Figure 30: Securing uSD/SD Connection on i.MX 6 EVK

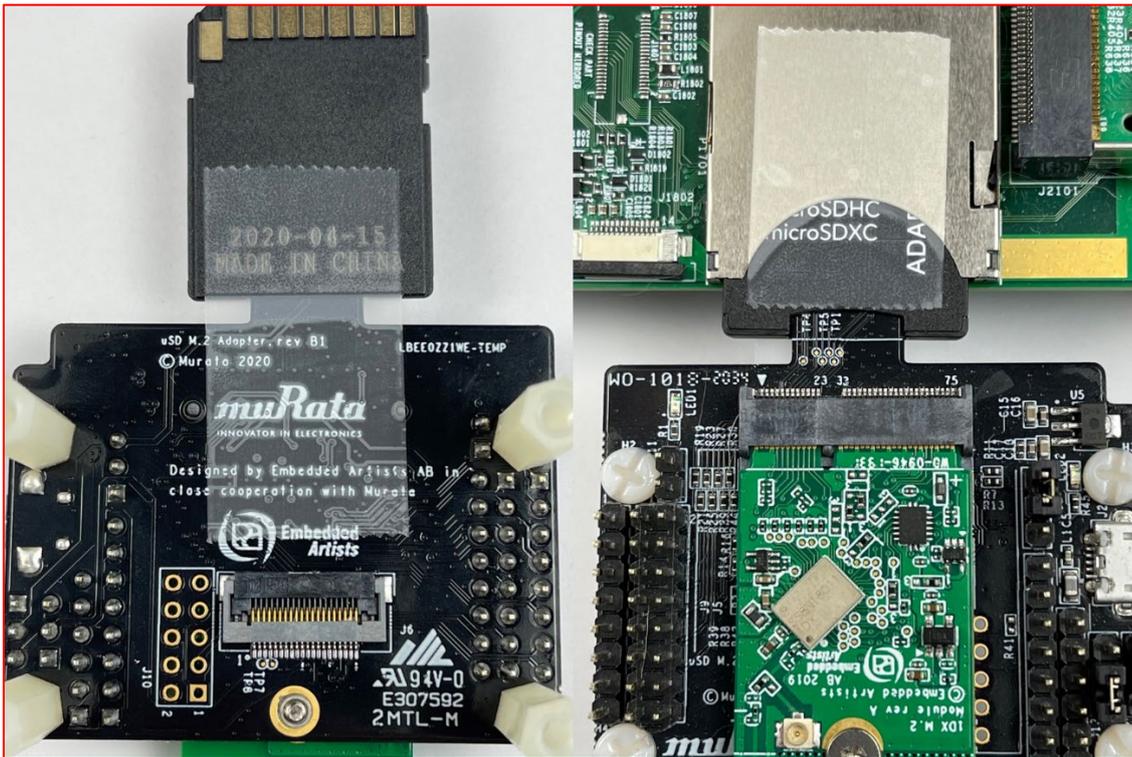
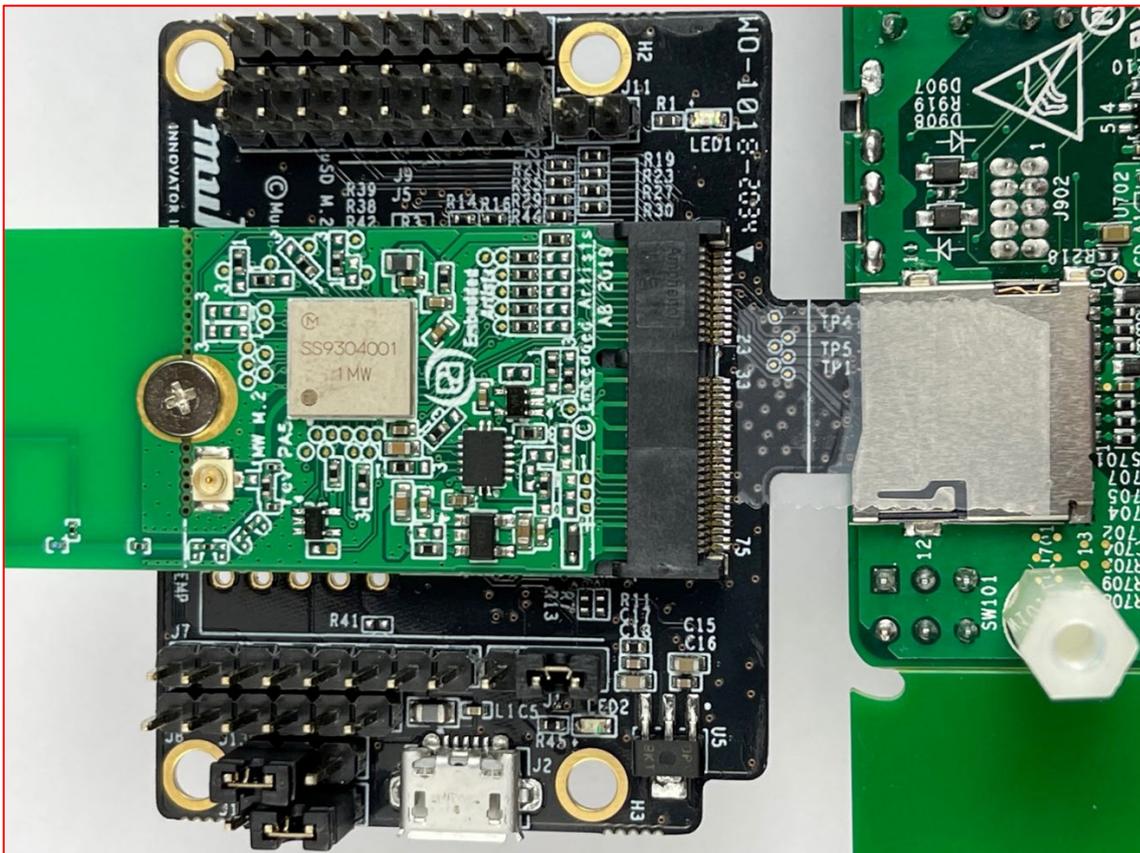


Figure 31: Securing uSD Connection on i.MX 8 EVK



8.4 uSD-M.2 Adapter High-Level Description

Figure 32 and **Figure 33** show the features on the uSD-M.2 Adapter; with text explanation in **Table 13**. The uSD-M.2 Adapter supports additional signals to WLAN-SDIO using either Arduino headers (J5, J8, and J9) or 20 pin FFC connector (J6). For more details on Murata's uSD-M.2 Adapter, refer to the [Adapter Datasheet](#) or [Hardware User Manual](#).

Table 13: uSD-M.2 Adapter Features

Char	Description
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)
E	J9 = BT UART TX/RX and WLAN/BT Control Signals (8 pin header)
F	J5 = Optional BT PCM and WLAN/BT Debug Signals (2x8 pin header)
G	Threaded mount for M.2 screw: 30 mm distance from M.2 connector
H	Regulator to step down optional 5V VBAT from USB or Arduino header to 3.3V
I	External sleep clock input (32.768 kHz)
J	J7 = Optional Arduino Header Power Supply (8 pin header; 5V or 3.3V VBAT)
K	J8 = BT UART RTS/CTS Signals (6 pin header)
L	J13 = Host IO Voltage: J13 in 1-2 pos for 3.3V VDDIO (default); J13 in 2-3 pos for 1.8V
M	J12 = M.2 IO Voltage: J12 in 1-2 pos for 1.8V VDDIO (default); J12 in 2-3 pos for 3.3V
N	J2 = Optional 5V USB Power Supply via Micro-AB USB Connector
O	LED2 = 3.3V M.2 IO Voltage Indicator (Blue) – not illuminated in default configuration
P	Regulator to provide optional 1.8V VIO to M.2 interface (M.2 EVBs have own 1.8V onboard)
Q	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
R	M.2 Connector: type 2230-xx-E
S	microSD connector pins: provides Power (VBAT, GND) and WLAN-SDIO
T	WLAN JTAG header (header pins not populated)
U	20 pin FFC connector (BT UART, BT PCM, WLAN/BT Control signals)
V	Additional test points from 20-pin flat/flex connector

Figure 32: uSD-M.2 Adapter Features (Top View)

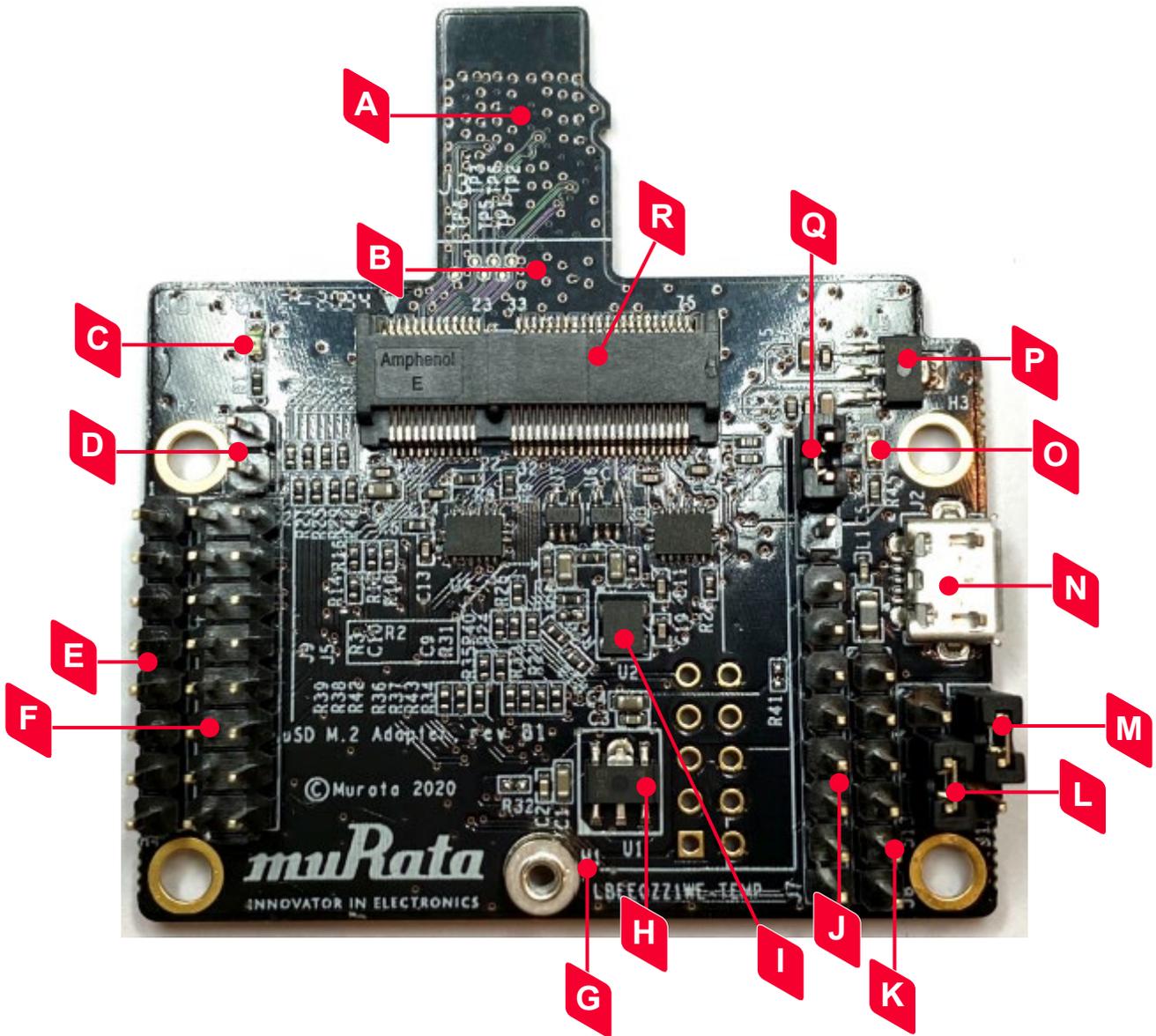
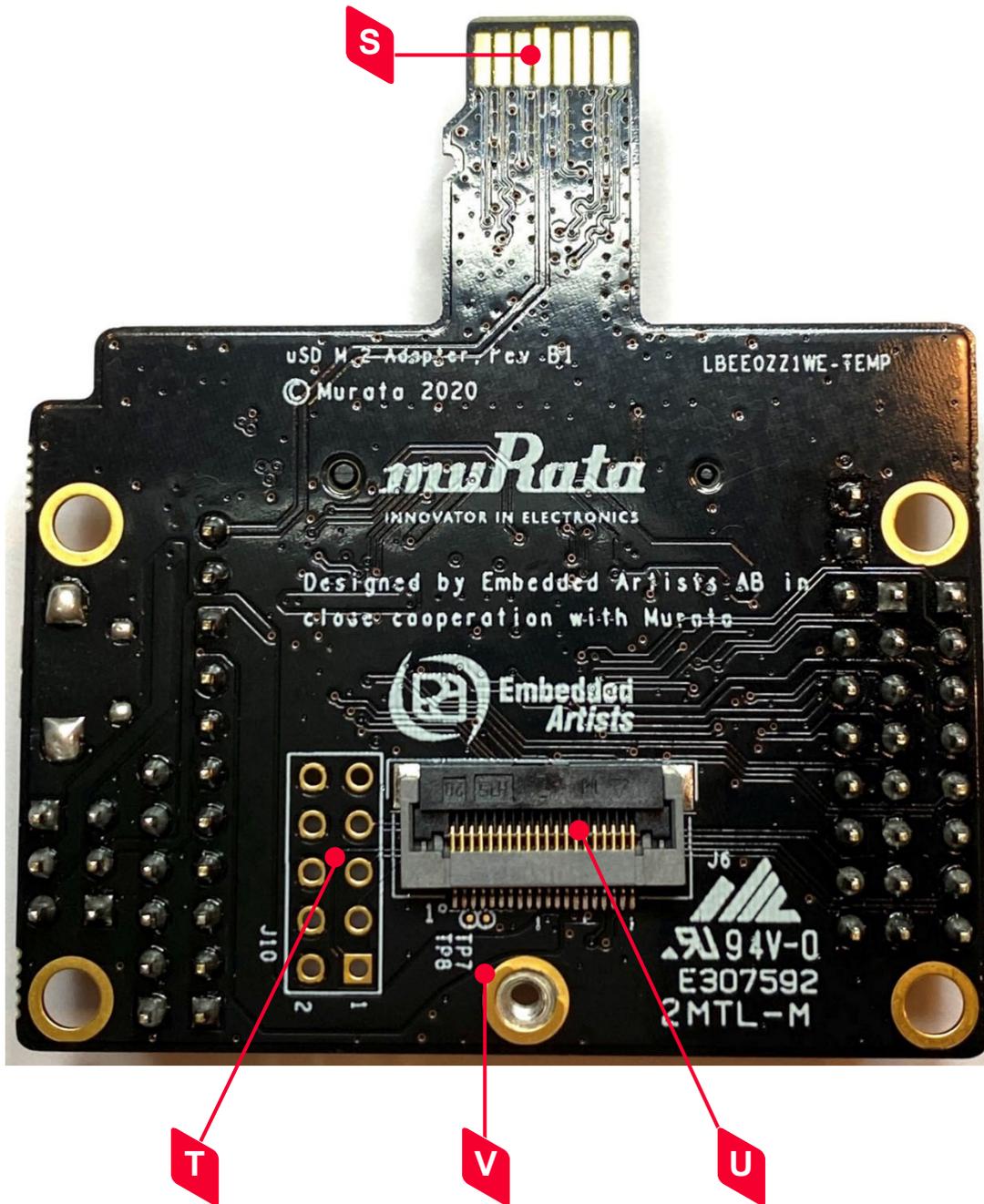


Figure 33: uSD-M.2 Adapter Features (Bottom View)



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

Embedded Artists designs, manufactures, and distributes all Wi-Fi/Bluetooth M.2 modules featuring Murata's mass-market modules (currently 1DX, 1MW, 1LV, 2AE, 2BC, 2BZ, 1YN, and 1XA for Infineon-based solutions). Murata partners very closely with Embedded Artists to test/validate all Wi-Fi/BT M.2 Modules. Customers can easily obtain these M.2 EVBs from Distributors (Mouser, Digi-Key, Future, and Arrow). For more information on the M.2 solution, please refer to [Embedded Artists website](#) .

10 Embedded Artists' i.MX + Wireless Solution

Murata has partnered with Embedded Artists to provide the ultimate solution for customers to evaluate Wi-Fi/Bluetooth modules easily and quickly. This solution is composed of three parts: Carrier board (baseboard), Computer on Module (COM) board, and Wi-Fi/BT M.2 Modules (EVBs). **Figure 34** shows that the carrier board can work with a variety of NXP i.MX6/7/8 (u)COM boards and eight different Murata-module based EVBs. With this platform, users can evaluate multiple i.MX processor and Wi-Fi/BT module permutations to find the best combination for their product. Also, Embedded Artists brings out all the test points you need for troubleshooting. With this platform, no adapter/interconnect is needed – either module-down solution or well-secured M.2 module. This is beneficial in two main areas: no limitation in Wi-Fi performance (WLAN-SDIO bus runs at full speed); and no mechanical issues (easy to secure M.2 module to EA Carrier Board). **Figure 34** shows how this platform works with (u)COM board and Wi-Fi/BT M.2 Modules. **Table 14** provides an Embedded Artists' i.MX (u)COM versus Murata module matrix. For comprehensive information on the Embedded Artists' solution refer to **Table 15**.

Figure 34: Combine i.MX COM with Wi-Fi/BT M.2 EVB

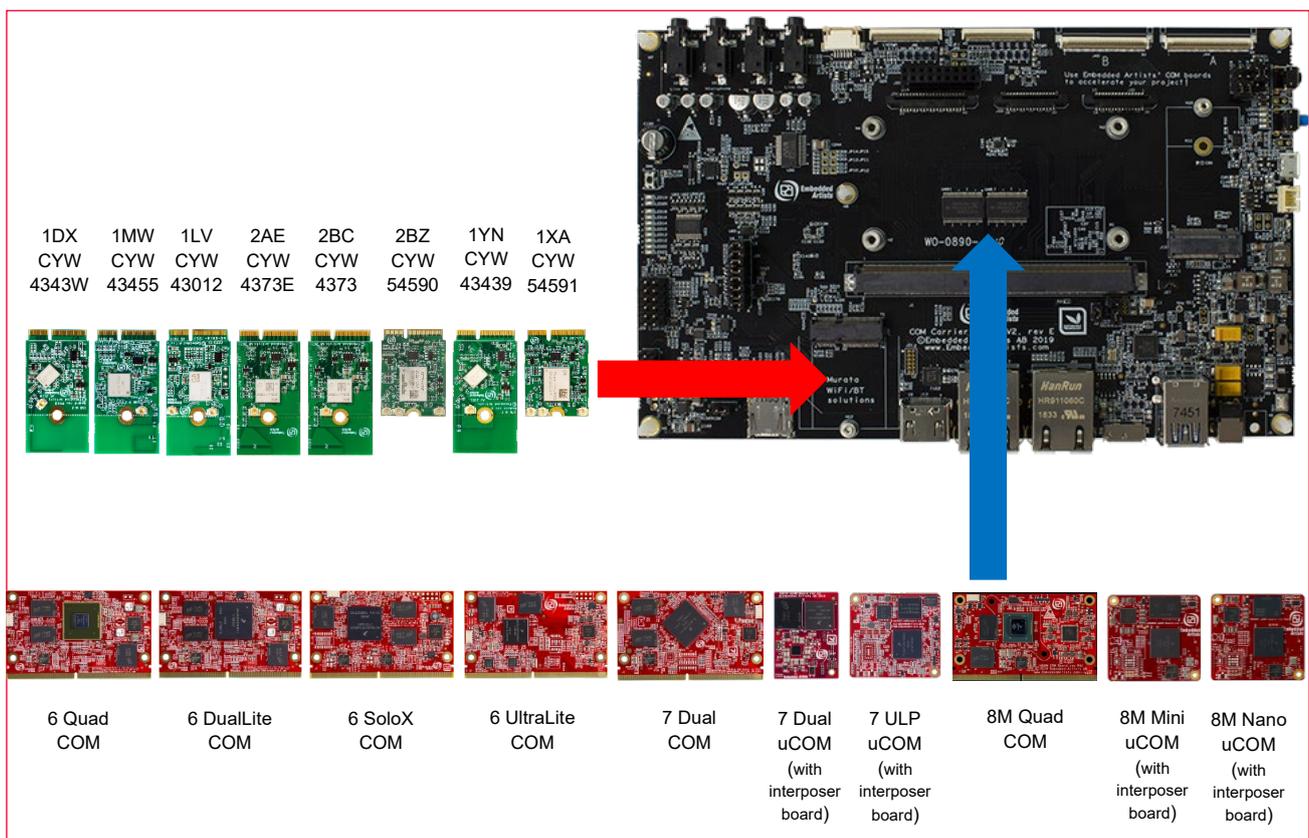


Table 14: Embedded Artists' i.MX Interconnect

EA i.MX (u)COM	1DX	1MW	1LV	2AE	2BC	2BZ	1YN	1XA
	CYW 4343W	CYW 43455	CYW 43012	CYW 4373E	CYW 4373	CYW 54590	CYW 43439	CYW 54591
iMX8M Quad COM	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷
iMX8M Mini uCOM	M.2 ²⁷	M.2 ²⁷ / MD ²⁸	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷
iMX8M Nano uCOM	M.2 ²⁷	M.2 ²⁷ / MD ²⁸	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	NC ²⁹
iMX7 Dual COM	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷
iMX7 Dual uCOM	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷
iMX7ULP uCOM	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷ / MD ²⁸	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	NC ²⁹
iMX6 Quad COM	M.2 ²⁷	M.2 ²⁷	M.2V ³⁰	M.2 ²⁷				
iMX6 DualLite COM	M.2 ²⁷	M.2 ²⁷	M.2V ³⁰	M.2 ²⁷				
iMX6 SoloX COM	M.2 ²⁷	M.2 ²⁷	M.2V ³⁰	M.2 ²⁷				
iMX6 UltraLite COM	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	M.2 ²⁷	NC ²⁹

Table 15: Embedded Artists' Landing Pages

Landing Pages	Notes
Embedded Artists' Website	The Art of Embedded Systems Development – made EASY™
i.MX 6/7/8 COM Boards	Listing of Computer-on-Module boards.
i.MX 6/7/8 COM Carrier Board V2	Main baseboard which all the COM boards plug into.
Getting Started with i.MX 6/7/8 Developer's Kit V2	How to bring up i.MX 6/7/8 Dev Kit (V2).
M.2 Module Family	Top level listing of 1DX, 1LV, 1MW, 2AE, 2BC, 2BZ, 1YN and 1XA M.2 EVB.
Application Development on an i.MX Developer's Kit	Description of C/C++, Python, Node.js, and Qt5 development.
Devices and Peripherals on an i.MX Kit	Description of how to work with peripherals and devices.

Table 16 includes links to Wi-Fi/BT M.2 Module datasheets, COM Carrier Board schematic and datasheet, reference WLAN-SDIO and WLAN-PCIe schematics, and (u)COM board specifications.

Table 16: Embedded Artists' Datasheets and Schematics

Datasheets and Schematics	Notes
i.MX 6/7/8 COM Carrier Board V2 Datasheet	Comprehensive definition of COM Carrier (baseboard).
i.MX6/7/8 COM Carrier Board V2 Schematics	Complete schematics including clear definition of uSD-M.2 Adapter.
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.

²⁷ Works with onboard M.2 slot.

²⁸ Module soldered down.

²⁹ No Connect.

³⁰ These platforms have fixed 3.3 V VIO for WLAN-SDIO. Although 1LV is 1.8V only, testing has yielded reliable results.

Datasheets and Schematics	Notes
EACOM Board Specification Guide 	Comprehensive definition of Embedded Artists' Computer-On-Module's.
1DX M.2 Module Datasheet 	Comprehensive details on 1DX Wi-Fi/BT M.2 Module.
1LV M.2 Module Datasheet 	Comprehensive details on 1LV Wi-Fi/BT M.2 Module.
1MW M.2 Module Datasheet 	Comprehensive details on 1MW Wi-Fi/BT M.2 Module.
2AE M.2 Module Datasheet 	Comprehensive details on 2AE Wi-Fi/BT M.2 Module.
2BC M.2 Module Datasheet 	Comprehensive details on 2BC Wi-Fi/BT M.2 Module.
2BZ M.2 Module Datasheet 	Comprehensive details on 2BZ Wi-Fi/BT M.2 Module.
1YN M.2 Module Datasheet 	Comprehensive details on 1YN Wi-Fi/BT M.2 Module.
1XA M.2 Module Datasheet 	Comprehensive details on 1XA Wi-Fi/BT M.2 Module.

Not only is the hardware solution much easier to work with, but the overall software solution makes things considerably more user-friendly. The Embedded Artists' i.MX Developer Kits are easy to flash, and their website provides ready-to-download Linux images which enable the wireless solution. This means that what may take customers 4 hours to accomplish with a NXP i.MX EVK (i.e., download Murata build script, build Yocto image, and flash platform), can be done in 10 minutes on the Embedded Artists' hardware (download Linux binary image, and flash image to platform).

Table 17 provides links to all the key software documentation and pre-built images. Embedded Artists maintains their own custom Linux release on GitHub. Their document "Working with Yocto to Build Linux" very much simplifies the Linux build process for customers.

Murata also supports any of the wireless solutions on Embedded Artists' Developer Kits on [Murata Community Forum](#) . Customers are welcome to register and post any questions they may have.

Table 17: Embedded Artists' User Manuals and Software

User Manuals and Software	Notes
Getting Started With M.2 Modules and i.MX 6/7/8 	Comprehensive document covering all major topics associated with using Wi-Fi/BT M.2 EVBs on EA's i.MX 6/7/8 Dev Kits.
i.MX Working with Yocto 	Comprehensive guide on building Linux images using Yocto framework.
Linux i.MX Images Download 	Pre-compiled images using "uuu" tool: allows users to easily flash i.MX platforms with latest Linux images with integrated Wi-Fi/BT support.
Wi-Fi/BT M.2 EVB Primer 	Introduction and drill-down on M.2 interface.

11 Useful Links

Table 18 provides some useful links.

Table 18: 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.

12 Appendix A

The following shows the output of building an image for [NXP i.MX 8MMini EVK](#), running [5.15.32](#) kernel release, using the Murata build script.

1. Start the build by running Murata's build script.

```
./Murata_Wireless_Yocto_Build_CYW.sh
```

2. Install the repo tool.

```
Do you want to continue? Y/n: Y
```

3. Select Stable build (this is Murata's tested release).

```
Select Stable ( 'n'=Developer )? Y/n: Y
Stable release selected
```

4. Select Hardknott Yocto release running Linux 5.15.32.

```
Select which entry? 0
Selected : 5.15.32
```

5. Select Fafnir FMAC.

```
Select which entry? 1
Selected : fafnir
```

6. Select i.MX 8MMini EVK as target platform.

```
Select your entry: 6
Selected target: imx8mmevk
```

7. Proceed with the default distro and image selections.

```
Murata default DISTRO & Image pre-selected are:
DISTRO: fsl-imx-wayland
Image: fsl-image-validation-imx

Proceed with this configuration? Y/n: Y
Proceeding with Murata defaults.
```

8. Enter build_8mm as build directory name.

```
Enter build directory name: build_8mm
```

9. Verify selection and start the build.

```
i.MX Yocto Release           : 5.15.32_2.0.0 GA
Yocto branch                 : kirkstone
fmac version                 : fafnir
Target                       : imx8mmevk
NXP i.MX EVK Part Number    : 8MMINILPD4-EVK
meta-murata-wireless Release Tag: imx-kirkstone-fafnir_r1.0
DISTRO                       : fsl-imx-wayland
Image                       : fsl-image-validation-imx
Build Directory              : build_8mm

Please verify your selection
Do you accept selected configurations ? Y/n: Y
```

10. Accept the End User License Agreement (EULA).

```
Do you want to continue? Y/n: Y
```

11. Accept the third-party EULA (press 'space' to read next page, 'q' to quit reading).

```
Do you accept the EULA you just read? (y/n) y  
EULA has been accepted.
```

12. Start the build. Typically, this takes 2~4 hours to complete. Ensure that there is a minimum of 50 GB free disk space.

```
Do you want to start the build ? Y/n: Y
```

13. Once the build is complete, the image will be available in ~/linux-imx/build_8mm/tmp/deploy/images/imx8mmevk/ folder. Look for the file with ".wic.bz2" extension.

13 Acronyms

Acronym	Meaning
AP	Access Point
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BSP	Board Support Package
BT	Bluetooth
CE	Conformité Européenne
CLI	Command Line Interface
CLK	Clock
CMD	Command
COM	Computer on Module
CPU	Central Processing Unit
CRDA	Central Regulatory Domain Agent
CTRL	Control
CTS	Clear to Send
CYW	Cypress, now Infineon
DHCP	Dynamic Host Configuration Protocol
DIP	Dual In-line Package
DTB	Device Tree Blob: Kernel reads in at boot time for configuration.
DTS	Device Tree Source
EA	Embedded Artists designs, manufactures and distributes current Wi-Fi/BT M.2 EVBs . EA also have enhanced i.MX developer kits which provide comprehensive support for Murata modules .
eMMC	Embedded Multi-Media Controller: integrated flash memory and controller on single die.
EULA	End User License Agreement
EVB	Evaluation Board (Embedded Artists' Wi-Fi/BT module)
EVK	Evaluation Kit (includes EVB + Adapter)
EVKB	Evaluation Kit Board
FCC	Federal Communications Commission
FFC	Flat Flex Cable
FW	Firmware
GIT	Global Information Tracker
GND	Ground
GPIO	General Purpose Input/Output
HCI	Host Controller Interface
I2S	Inter-IC Sound
IC	Industry Canada
IFX	Infineon
IRQ	Interrupt Request Line
LED	Light Emitting Diode
MAC	Medium Access Control
MEK	Multisensory Enablement Kit
MIMO	Multiple Input Multiple Output
NVRAM	Non-Volatile Random-Access Memory
OOB	Out of Band
O/S	Operation System
P2P	Peer-to-Peer
PC	Personal Computer
PCB	Printed Circuit Board
PCIe	PCI Express
PCM	Pulse Code Modulation
PSK	Pre-Shared Key
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
RTS	Request to Send
RX	Receive

Acronym	Meaning
SABRE	Smart Application Blueprint for Rapid Engineering
SD	Secure Digital
SDIO	Secure Digital Input Output
SSID	Service Set Identifier
STA	Station
SW	Software
SYNC	Synchronization
TELEC	Telecom Engineering Center
TX	Transmit
UART	Universal Asynchronous Receiver/Transmitter
UHS	Ultra-High Speed
UI	User Interface
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
VMware	Virtual Machine Software
Wi-Fi	Wireless LAN: "Wi-Fi" is a registered trademark of Wi-Fi Alliance
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access

14 Technical Support Contact

Table 19 lists all the support resources available for the Murata Wi-Fi/BT solution.

Table 19: List of Support Resources

Support Site	Notes
Murata Community Forum 	Primary support point for technical queries. This is an open forum for all customers. Registration is required.
Murata i.MX Landing Page 	No login credentials required. Murata documentation covering hardware, software, testing, etc. is provided here.
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.
Murata Module Landing Page 	No login credentials required. Murata documentation covering all Infineon-based Wi-Fi/BT modules is provided here.

15 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.

15.1 Murata Wi-Fi/Bluetooth for i.MX Linux User Manual for CYW-based Module

The [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.

15.2 Murata Wi-Fi/Bluetooth for i.MX Linux Quick Start Guide for CYW-based Module

The [Quick Start Guide](#)  provides quick steps to get started with Murata Wi-Fi/BT Infineon chipset-based solution with the help of an example.

15.3 Murata Wi-Fi/BT Solution for i.MX Hardware User Manual

The [Hardware User Manual](#)  describes the Murata uSD-M.2 Adapter hardware. All interface signals to the NXP i.MX RT, 6, 7, and 8 EVKs are described. Specifics on interfacing each i.MX EVK to Murata uSD-M.2 Adapter are provided.

15.4 Murata’s Community Forum Support

Murata’s Community provides online support for the Murata Wi-Fi/Bluetooth modules on various i.MX platforms. Refer to [this link](#)  for the Forum’s main Wi-Fi/Bluetooth landing page.

15.5 Murata uSD-M.2 Adapter Datasheet (Rev B2)

[This datasheet](#)  documents the Rev B2 version of the Murata’s latest uSD-M.2 adapter hardware and its interfacing options. This adapter is equivalent to the Rev B1, with a slightly modified sleep clock.

15.6 Murata uSD-M.2 Adapter Datasheet (Rev B1)

[This datasheet](#)  documents the Rev B1 version of the Murata’ latest uSD-M.2 adapter hardware and its interfacing options.

15.7 Murata uSD-M.2 Adapter Datasheet (legacy Rev A)

[This datasheet](#)  documents the legacy version of the Murata’ uSD-M.2 adapter hardware and its interfacing options. This adapter version is no longer manufactured.

15.8 Embedded Artists’ Reference Documentation

Embedded Artists designed the 1DX/1MW/1LV/2AE/2BC/2BZ/1YN/1XA M.2 EVBs in close collaboration with Murata. Refer to [this main landing page](#)  for more information.



Embedded Artists manufactures and distributes the Wi-Fi/BT M.2 EVBs.

Table 20 lists some relevant documents published by Embedded Artists.

Table 20: 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.
2AE M.2 Module Datasheet 	Comprehensive details on 2AE Wi-Fi/BT M.2 Module.
2BC M.2 Module Datasheet 	Comprehensive details on 2BC Wi-Fi/BT M.2 Module.
2BZ M.2 Module Datasheet 	Comprehensive details on 2BZ Wi-Fi/BT M.2 Module.
1YN M.2 Module Datasheet 	Comprehensive details on 1YN Wi-Fi M.2 Module.
1XA M.2 Module Datasheet 	Comprehensive details on 1XA Wi-Fi M.2 Module.

15.9 Murata's i.MX Wireless Solutions Landing Page

[This website landing page](#)  provides latest/comprehensive information on Murata's i.MX Wireless solutions which use the uSD-M.2 Adapter as a key enabler so customers can easily evaluate Murata's modules on i.MX processors.

15.10 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 21 provides the following information on all releases supported:

Table 21: NXP Reference Documentation Listing

Kernel release	NXP documentation link	Yocto name	FMAC code name	Release information
5.15.32_2.0.0	Rev. L5.15.32_2.0.0_BSP 	Kirkstone	Fafnir Ebirah	imx-kirkstone-fafnir_r1.0 imx-kirkstone-ebirah_r1.0
5.10.52_2.1.0	Rev. L5.10.52_2.1.0_BSP 	Hardknott	Cynder Drogon	imx-hardknott-cynder_r1.0 imx-hardknott-drogon_r1.0
5.4.47_2.2.0	Rev. L5.4.47_2.2.0_BSP 	Zeus	Baragon Spiga	imx-zeus-baragon_r1.0 imx-zeus-spiga_r1.0
4.14.98_2.3.0	Rev. L4.14.98_2.3.0_BSP 	Sumo	Baragon Spiga	imx-sumo-baragon_r1.0 imx-sumo-spiga_r1.1

Each archive downloadable (NXP documentation link) contains the following:

- i.MX_Yocto_Project_User's_Guide.pdf / IMXLXOCTOUG
- i.MX_Linux_User's_Guide.pdf / IMXLUG
- i.MX_Linux_Reference_Manual.pdf / IMXLXRM
- i.MX_Linux_Release_Notes.pdf / IMXLXRN

Revision History

Revision	Date	Author	Change Description
1.0	Nov 17, 2020	TF	Initial Release. Added support for i.MX Linux Kernel version 4.14.98, 5.4.47, and Type 1XA. Explained Embedded Artists' hardware/software solution for evaluating processors and EVBs. NOTE: Material moved from previous Quick Start Guide.
1.1	Jan 28, 2021	TF	Minor corrections, updated links and some photos.
2.0	May 12, 2022	TF	Migrated to new format. Updated for latest software support.
3.0	Nov 23, 2022	TF	Migrated to template 2.0.
3.1	Feb 02, 2023	TF	Added support for kernel 5.15.32, FMAC "ebirah" and "fafnir". Removed support for kernel 4.9.123. Added modules 2BC and 2BZ. Replaced 'switch_module' script usage with 'set_module'.



Copyright © Murata Manufacturing Co., Ltd. All rights reserved. The information and content in this document are provided “as-is” with no warranties of any kind and are for informational purpose only. Data and information have been carefully checked and are believed to be accurate; however, no liability or responsibility for any errors, omissions, or inaccuracies is assumed.

Wi-Fi® is a registered trademark of Wi-Fi Alliance. The Bluetooth® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. Other brand and product names are trademarks or registered trademarks of their respective owners.

Specifications are subject to change without notice.