

Wi-Fi®/Bluetooth® (NXP) for i.MX

Linux User Guide - Rev. 2.0

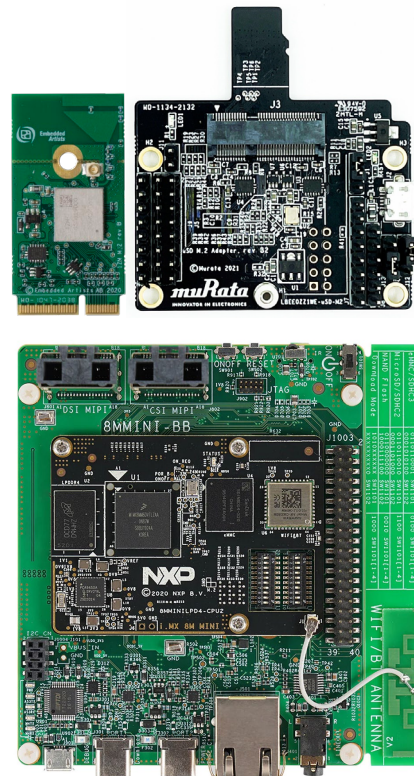


Table of Contents

1 Introduction.....	6
1.1 NXP i.MX 6 Platform Support.....	7
1.2 NXP i.MX 8 Platform Support.....	8
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.....	12
2.3 NXP i.MX versus Murata Module Interconnect.....	13
3 Wi-Fi/BT Software Solution for i.MX.....	14
3.1 Murata's Customized NXP Wireless Driver Release	14
3.2 Specific i.MX Target Support Details	14
3.3 NXP's Default Demo Image	15
3.4 Additional Hardware/Software Considerations	16
3.4.1 1.8V Versus 3.3V VIO Signaling using Murata's uSD-M.2 Adapter.....	16
3.4.2 UHS SDIO 3.0 Operation on i.MX 6UL(L) EVKs with uSD-M.2 Adapter.....	16
3.4.3 WLAN/Bluetooth M.2 Direct Interconnect on NXP i.MX Platforms	16
3.4.4 Setting Correct Software Configuration Before Testing Wi-Fi/BT Solution	17
3.4.5 Type 1YM M.2 EVB WLAN/Bluetooth Bus Interface Configuration	17
3.5 Murata Customized i.MX Yocto Image Build	18
3.5.1 Install Ubuntu	19
3.5.2 Download Murata's Script Files	19
3.5.3 Configure Ubuntu for i.MX Yocto Build	20
3.5.4 Murata's i.MX Yocto Build Script.....	20
4 Preparing NXP i.MX Platforms to Boot Linux Image.....	23
4.1 Flashing Linux Image to (Micro) SD Card.....	23
4.1.1 Linux PC Steps to Flash SD Card.....	23
4.1.2 Windows PC Steps to Flash SD Card.....	24
4.1.3 Steps to Load DTB Files.....	25
4.2 Flashing Linux Image to NXP i.MX 8M Mini/Nano EVKs	25
4.2.1 Software File Preparation	25
4.2.2 i.MX 8M Mini or Nano EVK Hardware Configuration.....	26
4.2.3 Flash Linux Image to eMMC on i.MX 8M Mini/Nano EVK.....	28
4.2.4 Configure i.MX 8M Mini/Nano EVK to Boot from eMMC	28
5 Murata Wi-Fi/BT Bring-Up on i.MX 6 Platforms.....	30
5.1 Connecting to i.MX 6UL EVK or i.MX 6ULL EVK.....	31
6 Murata Wi-Fi/BT Bring-Up on i.MX 8 Platforms.....	32

6.1 Bringing up Wi-Fi/BT on i.MX 8MQuad EVK.....	32
6.2 Bringing up Wi-Fi/BT on i.MX 8M Mini/Nano EVKs (M.2)	33
6.3 Bringing up Wi-Fi/BT on i.MX 8M Mini/Nano EVKs (uSD-M.2 Adapter).....	34
7 Test/Verification of Wi-Fi and Bluetooth	42
7.1 Wi-Fi Interface Test/Verification	42
7.1.1 Useful Environment Setup on NXP Linux	42
7.1.2 Bringing Up Wi-Fi Interface.....	42
7.1.3 STA/Client Mode: Scan for Visible Access Points.....	46
7.1.4 STA/Client Mode: Connecting to Unsecured Access Point or Wireless Router	48
7.1.5 STA/Client Mode: Connecting to Secured Access Point or Wireless Router (WPA2-PSK)	49
7.1.6 STA/Client Mode: Basic WLAN Connectivity Testing.....	52
7.1.7 Wi-Fi Direct Testing	53
7.2 Bluetooth Interface Test/Verification.....	55
8 Murata's uSD-M.2 Adapter.....	56
8.1 Connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter	56
8.2 Configuring uSD-M.2 Adapter Jumpers for Correct VIO Signaling.....	56
8.3 Securing uSD-M.2 Adapter to NXP i.MX EVK	57
8.4 uSD-M.2 Adapter High-Level Description	59
9 Embedded Artists' Wi-Fi/BT M.2 Modules.....	62
10 Embedded Artists' i.MX + Wireless Solution	62
11 Murata's Regulatory Solution for NXP Modules	64
11.1 Description of Regulatory Solution for WLAN/BT.....	64
11.2 Tree view of the files list for Linux Patched Solution	65
11.2.1 Description of files	67
11.3 Purpose of switch_regions.sh.....	67
11.3.1 Sample Description of Structure SD8987 from "wifi_mod_para.conf" configured for US with 1ZM module.....	68
11.3.2 Sample Description of Structure PCIE8997 from "wifi_mod_para.conf" Configured for EU with 1YM Module.....	68
11.4 Sample log of switch_regions.sh: (Ex: 1YM-SDIO@ 8M-MINI).....	68
12 Useful Links	69
13 Appendix A: Building Image Output	69
14 Appendix B: Acronyms.....	71
15 Technical Support Contact.....	72
16 References	73
16.1 Wi-Fi/Bluetooth for i.MX Linux Quick Start Guide for NXP-based Module.....	73
16.2 Murata Wi-Fi/BT Solution for i.MX Hardware User Manual.....	73

16.3 Murata's Community Forum Support.....	73
16.4 Murata's FCC Regulatory Test Guide.....	73
16.5 Murata uSD-M.2 Adapter Datasheet (Rev B2).....	73
16.6 Murata uSD-M.2 Adapter Datasheet (Rev B1).....	73
16.7 Embedded Artists' Reference Documentation	73
16.8 Murata Linux Patched Solution.....	74
16.9 Murata's i.MX Wireless Solutions Landing Page	74
16.10 NXP Reference Documentation	74
Revision History.....	75

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: Type 1YM M.2 EVB Configuration Strapping Options	18
Figure 8: Type 1YM M.2 Configured for WLAN-SDIO/BT-UART (1YM-SDIO).....	18
Figure 9: Configuring dash.....	19
Figure 10: USB to SD Card Reader/Writer Adapter	24
Figure 11: Power, Download, and Debug port Connection to Board	26
Figure 12: i.MX 8M Mini EVK DIP Switches Configured for Download.....	27
Figure 13: i.MX 8M Nano EVK DIP Switches Configured for Download.....	27
Figure 14: i.MX 8M Mini EVK DIP Switches Configured for eMMC Boot.....	29
Figure 15: i.MX 8M Nano EVK DIP Switches Configured for eMMC Boot	29
Figure 16: uSD-M.2 Adapter with Type 1ZM and 1YM-SDIO* M.2 EVB Options.....	30
Figure 17: i.MX 6ULL EVK with uSD-M.2 Adapter and Type 1ZM M.2 EVB.....	32
Figure 18: i.MX 8MQuad with Type 1YM (Bottom View)	33
Figure 19: i.MX 8M Mini with Type 1YM-PCIe (Bottom View).....	34
Figure 20: i.MX 8M Mini/Nano EVK with uSD-M.2 Adapter (WLAN Only)	34
Figure 21: i.MX 8M Mini/Nano EVK with uSD-M.2 Adapter (WLAN/Bluetooth).....	35
Figure 22: Cable Connections on i.MX 8M Mini EVK (Even Number Connector View).....	37
Figure 23: Cable Connections on i.MX 8M Mini EVK (Odd Number Connector View).....	37
Figure 24: Cable Connections on uSD-M.2 Adapter (J9 Header).....	38
Figure 25: Cable connections on uSD-M.2 Adapter (J8 Header).....	39
Figure 26: Low-Profile Jumper Wires (Digi-Key part number 1988-1178-ND)	39
Figure 27: NXP i.MX EVK with Low-Profile Jumper Wires	40

Figure 28: Additional Hex Standoff (Digi-Key Part Number RPC3570-ND)	41
Figure 29: Connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter	56
Figure 30: Host/M.2 IO Voltage Level Shift Options on Rev B2 Adapter	57
Figure 31: Securing uSD/SD Connection on i.MX 6 EVK	58
Figure 32: Securing uSD Connection on i.MX 8 EVK	58
Figure 33: uSD-M.2 Adapter Features (Top View)	60
Figure 34: uSD-M.2 Adapter Features (Bottom View)	61
Figure 35: Combine i.MX COM with Wi-Fi/BT M.2 EVB	62

Tables

Table 1: Document Conventions	5
Table 2: Current NXP i.MX Platforms Supported	6
Table 3: Embedded Artists' Wi-Fi/BT M.2 Modules Supported	12
Table 4: uSD-M.2 Adapter Kit Contents	12
Table 5: NXP i.MX/Murata Module Interconnect	14
Table 6: NXP i.MX EVK / Yocto (MACHINE) target / Kernel Version Matrix	15
Table 7: i.MX6/8 Targets supported by Murata	15
Table 8: Select Hardware Configurations which use eMMC Boot Configuration	23
Table 9: Files to Flash i.MX 8M Mini & 8M Nano EVKs	25
Table 10: i.MX 8M Mini/Nano EVK Jumper Connections to uSD-M.2 Adapter	36
Table 11: Embedded Wi-Fi/Bluetooth Files	42
Table 12: GPIO and UART Settings for Bluetooth Tests	55
Table 13: uSD-M.2 Adapter Features	59
Table 14: Embedded Artists' i.MX Interconnect	63
Table 15: Embedded Artists' Landing Pages	63
Table 16: Embedded Artists' Datasheets and Schematics	63
Table 17: Embedded Artists' User Manuals and Software	64
Table 18: Murata regulatory files	67
Table 19: Useful Links	69
Table 20: List of Support Resources	72
Table 21: Embedded Artists Documentation Listing	74
Table 22: NXP Reference Documentation Listing	74

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.](#) 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:

- 5.10.52_2.1.0
- 5.10.72_2.2.0

Following steps are described in this document:

- How to use NXP's demo images and flash it to various NXP i.MX EVKs, thereby enabling the NXP WLAN driver and associated components (WPA supplicant, WLAN firmware, calibration files, regulatory DB file, 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.
- How to use Murata's patched solution to correctly configure regulatory settings.



This is required for optimal performance, since the NXP demo image is not fully tuned for Murata modules.

The NXP Platforms currently supported are based on i.MX 8 and i.MX 6 series. The wireless solution for the following platforms 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**. Note that the uSD-M.2 Adapter only supports WLAN-SDIO configuration. Some instances of M.2 connect currently support WLAN-PCIe and/or WLAN-SDIO due to NXP i.MX reference platform configuration. Type 1ZM, 1XK, and 2DS M.2 EVB supports WLAN-SDIO interface only. Type 1YM M.2 EVB serves "double duty" in supporting both WLAN-PCIe (default strapping configuration) and WLAN-SDIO options. Type 1XL currently supports WLAN-PCIe only.

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	1YM, 1XL	M.2 (WLAN-PCIe)
8MPLUSLPD4-EVK	i.MX 8MPLUS EVK	1ZM, 1YM, 1XK, 1XL, 2DS	M.2: 1ZM, 1YM (PCIe and SDIO), 1XK, 1XL (PCIe), 2DS
MCIMX8M-EVKB	i.MX 8MQuad EVK	1ZM, 1YM, 1XK, 1XL, 2DS	uSD-M.2 Adapter: 1ZM, 1YM, 1XK, 2DS (WLAN Only) M.2 (WLAN-PCIe): 1YM, 1XL
8MMINILPD4-EVKB	i.MX 8M Mini EVK	1ZM, 1YM, 1XK, 1XL, 2DS	uSD-M.2 Adapter: 1ZM, 1YM, 1XK, 2DS M.2 (WLAN-PCIe): 1YM – WLAN Only, 1XL – WLAN Only
8MNANOD4-EVK	i.MX 8M Nano EVK	1ZM, 1YM, 1XK, 2DS	uSD-M.2 Adapter

MCIMX6UL-EVKB	i.MX 6UL EVK	1ZM, 1YM, 1XK, 2DS	uSD-M.2 Adapter
MCIMX6ULL-EVK	i.MX 6ULL EVK	1ZM, 1YM, 1XK, 2DS	uSD-M.2 Adapter



2DS Supports only WLAN. There is no Bluetooth.

1.1 NXP i.MX 6 Platform Support

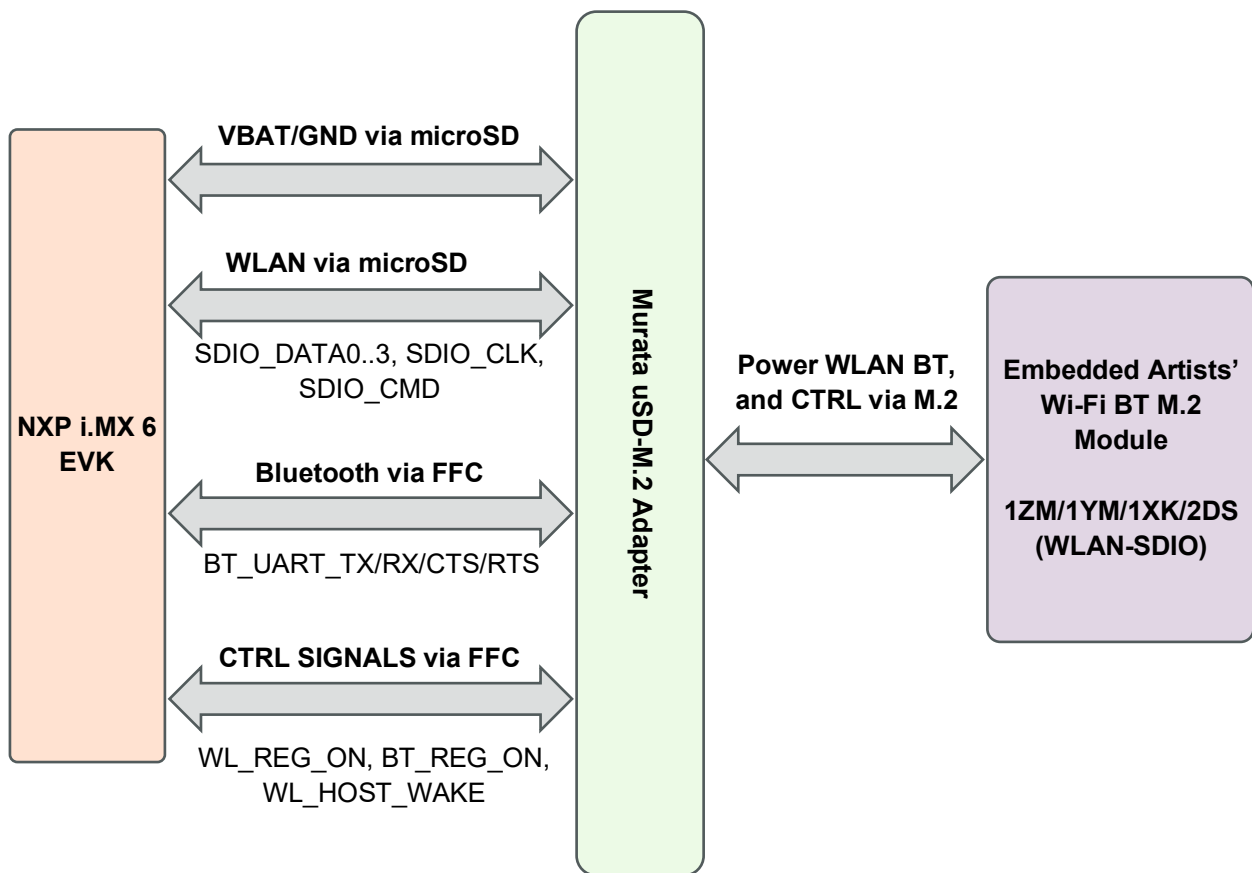
A high-level connection Diagram for the Murata uSD-M.2 Adapter (i.MX 6 platforms) is provided in **Figure 1**. All the Murata Wi-Fi/BT modules enabled by this release are shown. The wireless solution is arrived at by combining [Murata's uSD-M.2 Adapter](#) with [Embedded Artists' Wi-Fi/BT M.2 EVB](#). 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.
- ❖ Comprehensive test points (including SDIO DATA, CLK, and CMD lines).
- ❖ Strapping options on specific Wi-Fi/BT M.2 EVBs that support multiple bus interface configurations. Currently this is limited to Type 1YM M.2 Module (supporting WLAN-PCIe and WLAN-SDIO configuration options).



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



Regarding the Murata modules currently supported:

- Type 1ZM and 1XK supports WLAN-SDIO and BT-UART interfaces only.
- Type 1YM supports WLAN-PCIe/BT-UART (default). It can be configured (via strapping options) to support additional evaluation mode of WLAN-SDIO/BT-UART. This strapping configuration 1YM M.2 EVB (WLAN-SDIO/BT-UART) is denoted as "1YM-SDIO".
- 1YM will support WLAN-PCIe/BT-UART and WLAN-SDIO/BT-UART only. Drivers for both the configurations are included in Linux 5.10.52/5.10.72 BSP releases. For more information on Type 1YM M.2 EVB configuration refer to [Section 3.4.5](#).
- Type 2DS supports only WLAN-SDIO.

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 PCIe interconnect. Type 1YM M.2 EVB is supported over the WLAN-PCIe/BT-UART interfaces. The i.MX 8QXP MEK is supported by this same configuration as well.



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

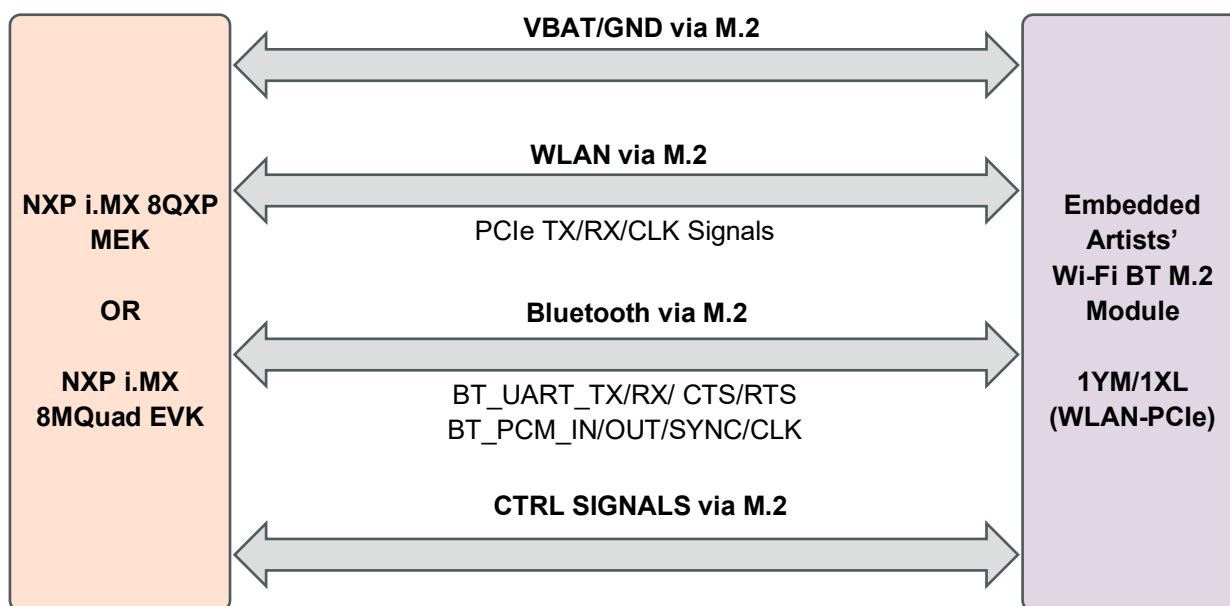


Figure 3 shows a simplified block diagram for the i.MX 8MQuad EVK Wi-Fi/BT SDIO interconnect. Type 1ZM, 1YM, 1XK and 2DS modules are supported via the uSD-M.2 adapter on i.MX 8MQuad EVK. Note that this EVK can use onboard eMMC flash. The eMMC flash is necessary as the default uSD card slot is no longer available for booting the Linux image. As pictured, the Type 1YM M.2 EVB strapping ([Section 3.4.5](#)) needs to be configured for WLAN-SDIO/BT-UART operation. NXP baseline release supports WLAN-PCIe/BT-UART and WLAN-SDIO/BT-UART. Of course, the default for 1YM M.2 EVB (part EAR00370 as shipped) is WLAN-PCIe/BT-UART.

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

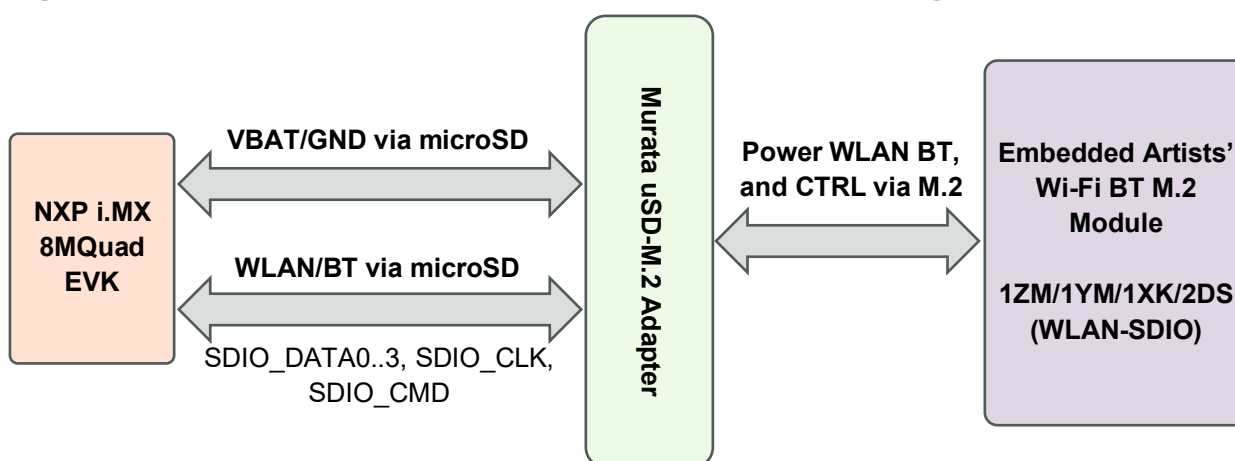


Figure 4 shows a simplified block diagram for the i.MX 8M Plus EVK Wi-Fi/BT interconnect (using M.2 connector). Currently Type 1ZM, 1YM module (PCIe and SDIO), 1XK and 2DS is supported via the M.2 interface. Both WLAN-PCIe and WLAN-SDIO is supported via this interface.

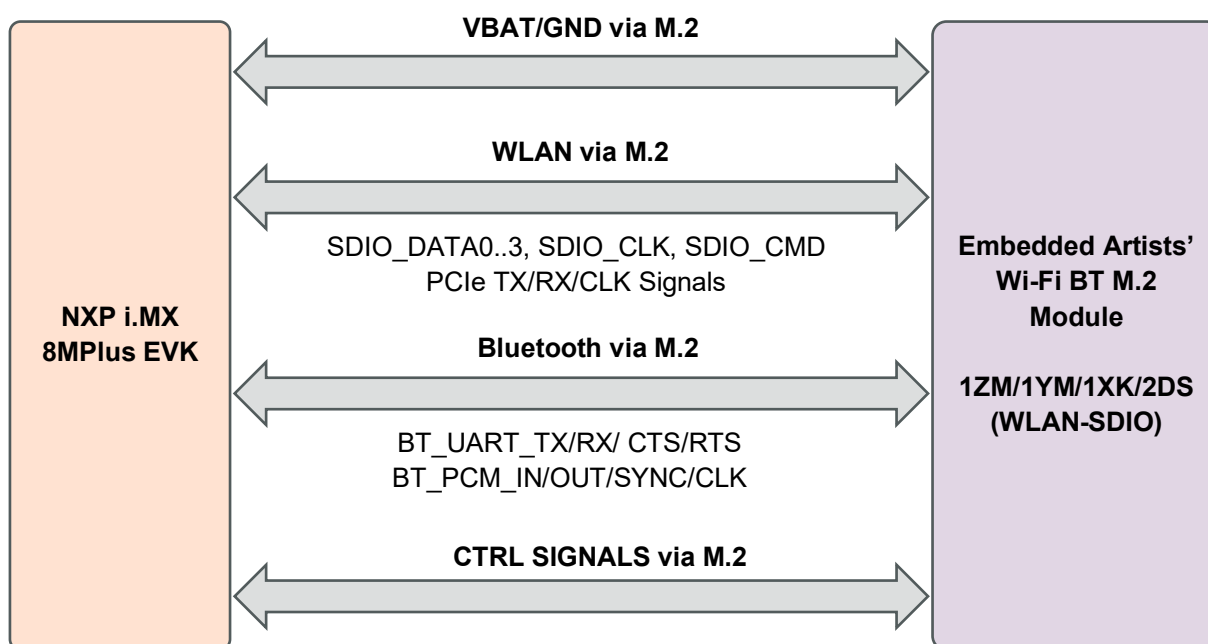
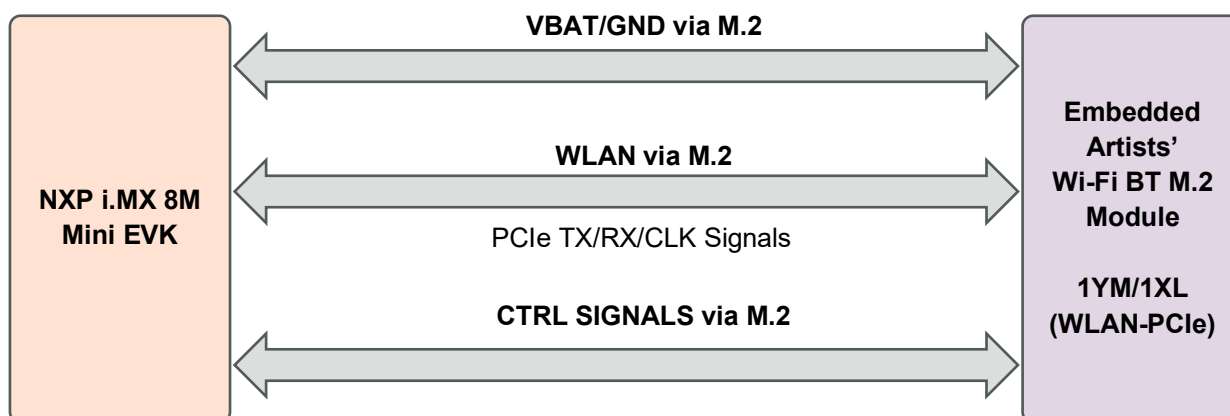
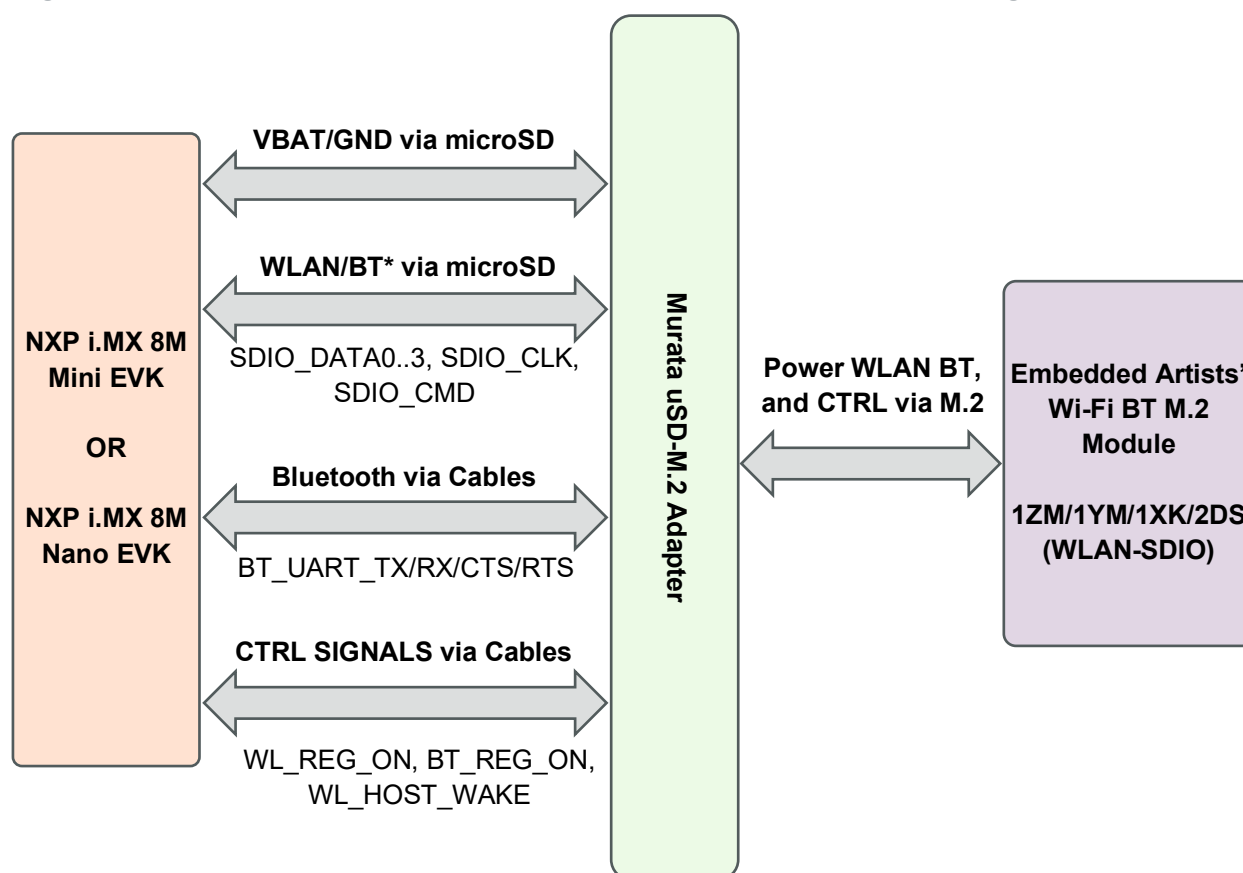
Figure 4: i.MX 8MPlus EVK Wi-Fi/BT Interconnect Block Diagram

Figure 5 shows a simplified block diagram for the i.MX 8M Mini EVK Wi-Fi/BT interconnect (using M.2 connector). Currently Type 1YM, 1XL module (WLAN Only – Bluetooth signals are not brought out) are supported via the WLAN-PCIe interface.

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

For WLAN-SDIO based module see **Figure 6**. Type 1ZM, 1YM, 1XK and 2DS modules are supported via the uSD-M.2 adapter on both the i.MX 8M Mini and i.MX 8M Nano EVKs. Note that both variants of these EVKs use onboard eMMC flash. The eMMC flash is necessary as the default uSD card slot is no longer available for booting the Linux image. As pictured, the Type 1YM M.2 EVB strapping ([Section 3.4.5](#)) needs to be configured for WLAN-SDIO/BT-UART operation (see “WLAN/BT* via microSD” in **Figure 6**). NXP baseline release supports WLAN-PCIe/BT-UART and WLAN-SDIO/BT-UART.

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 1ZM (NXP 88W8987), Type 1XK (NXP IW416) and Type 2DS (88W8801) supports only WLAN-SDIO interface, whereas Type 1YM (NXP 88W8997) supports both WLAN-PCIe and WLAN-SDIO interfaces. Also note that both Type 1ZM's and Type 1YM's WLAN-SDIO VIO interface only supports 1.8V¹. Type 1XL (NXP 88W9098) supports WLAN-PCIe. 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.

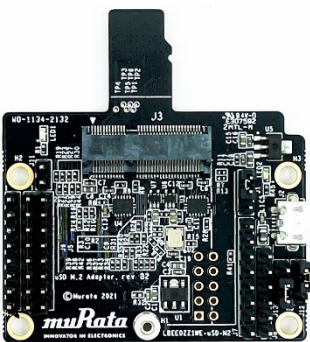

¹ The M.2 standard specifies 1.8V VIO voltage for SDIO, UART, and PCM interfaces.





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
1ZM	NXP 88W8987	802.11b/g/n/ac BT/BLE 5.1	LBEE5QD1ZM	EAR00364	1.8V Only	WLAN-SDIO BT-UART	
1YM	NXP 88W8997	802.11a/b/g/n/ac (2x2 MIMO) BT/BLE 5.2	LBEE5XV1YM	EAR00370	1.8V (WLAN SDIO, PCIe) 3.3V (WLAN-PCIe Only)	WLAN-PCIe WLAN-SDIO BT-UART BT-SDIO	
1XK	NXP IW416	802.11a/b/g/n BT/BLE 5.2	LBEE5CJ1XK	EAR00385	1.8V (WLAN SDIO) 3.3V (WLAN-SDIO)	WLAN-SDIO BT-UART	
1XL	NXP 88W9098	802.11a/b/g/n/ac/ax (2x2 MU-MIMO) BT/BLE 5.2	LBEE5ZZ1XL	EAR00387	1.8V/3.3V (WLAN-PCIe Only)	WLAN-PCIe BT-UART	
2DS	NXP 88W8801	802.11b/g/n	LBWA0ZZ2DS	EAR00386	1.8V (WLAN SDIO) 3.3V (WLAN-SDIO)	WLAN-SDIO	

2.2 Murata's uSD-M.2 Adapter

Table 4: uSD-M.2 Adapter Kit Contents

Picture of Contents	Description of Contents
	uSD-M.2 Adapter (Revision B2)
	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)
	4 x 19 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. NXP 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.
- **“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.2+”**: Murata’s uSD-M.2 Adapter (Rev 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. For this configuration, Type 12M M.2 EVB requires BT-UART and WLAN/BT control signal connection whereas Type 1YM M.2 EVB only requires WLAN/BT control signal connection (in current 1YM-SDIO* configuration).
- **“M.2”**: Only Embedded Artists’ Wi-Fi/BT M.2 EVB is required. The 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. 1YM) can be used.

- **“M.2^W”**: Only Embedded Artists’ Wi-Fi/BT M.2 EVB is required. The 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.
- Again, the default hardware strapping option for EA’s Type 1YM M.2 EVB (EAR00370) is WLAN-PCIe/BT-UART. In **Table 5**, we refer to this default configuration as **“1YM-PCIe”**. It is otherwise referred to as 1YM. The 1YM (WLAN-SDIO/BT-UART) M.2 configuration will be referenced as **“1YM-SDIO”**.

Table 5: NXP i.MX/Murata Module Interconnect

NXP i.MX EVK Part Number	1ZM	1YM-SDIO	1YM-PCIe	1XL	1XK	2DS
	NXP 88W8987	NXP 88W8997	NXP 88W8997	NXP 88W9098	NXP IW416	NXP 88W8801
MCIMX8QXP-CPU	NC ^[2]	NC	M.2 ^[3]	M.2	NC	NC
8MPLUSLPD4-EVK	M.2	M.2	M.2	M.2	M.2	M.2
MCIMX8M-EVKB	uSD-M.2 ^[4]	uSD-M.2	M.2	M.2	uSD-M.2	uSD-M.2
8MMINILPD4-EVKB	uSD-M.2 ^{+[5]}	uSD-M.2 ⁺	M.2 ^{W[6]}	M.2 ^W	uSD-M.2 ⁺	uSD-M.2 ⁺
8MNANOD4-EVK	uSD-M.2	uSD-M.2	NC	NC	uSD-M.2	uSD-M.2
MCIMX6UL-EVKB	uSD-M.2	uSD-M.2	NC	NC	uSD-M.2	uSD-M.2
MCIMX6ULL-EVK	uSD-M.2	uSD-M.2	NC	NC	uSD-M.2	uSD-M.2

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

This section describes different aspects of Murata’s Wi-fi/BT software solution for i.MX.

3.1 Murata’s Customized NXP Wireless Driver Release

The NXP i.MX Linux 5.10.52/5.10.72 BSP release for i.MX 6/8 integrates the NXP WLAN driver and has Bluetooth (BlueZ stack) support. Murata’s Linux patched solution also provides the following enhancements/customizations:

- Comprehensive support for 1ZM/1YM/1XK/1XL/2DS on all possible i.MX targets - hardware interconnect is the only limiting factor: see **Table 7**.
- 1.8V SDIO VIO configuration for WLAN interface on i.MX 6UL(L) EVKs.

3.2 Specific i.MX Target Support Details

Murata’s Linux patched solution 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). From this table, you can find a proper version of Linux Kernel for your target platform.

² No Connection options available

³ Wi-Fi/BT M.2 EVB plugs directly into M.2 connector

⁴ Works with uSD-M.2 Adapter configured for 1.8V default

⁵ Works with uSD-M.2 Adapter (Rev B1/B2) with additional cabling

⁶ Wi-Fi/BT M.2 EVB plugs directly into M.2 connector; but only WLAN is functional

You can then refer to **Table 7** for the support of interrupt configuration, corresponding DTB (Device Tree Blob) files and hardware interconnect configuration (either uSD-M.2 Adapter & M.2 EVB, or just M.2 EVB). Note that the current Wi-Fi/BT M.2 EVBs (1ZM/1YM/1XK/2DS) supporting WLAN-SDIO interface only interface at 1.8V VIO. The i.MX 8M Mini/Nano/Quad EVKs with “uSD-M.2+” configuration require additional part(s) as documented in [Section 6.3](#).

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

NXP i.MX EVK Part #	NXP i.MX EVK	MACHINE=target	Kernel 5.10.52	Kernel 5.10.72
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	✓	✓
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
imx8qxpmeek	M.2 ^[7]	imx8qxp-mek.dtb	N/A
imx8mp-lpddr4-evk	M.2	imx8mp-evk-usdhc1-m2.dtb	SDIO
imx8mp-lpddr4-evk	M.2	imx8mp-evk.dtb	N/A
imx8mqevk	M.2	imx8mq-evk-pcie1-m2.dtb	N/A
imx8mqevk	uSD-M.2 ^[8]	imx8mq-evk.dtb	SDIO
imx8mmevk	M.2 ^[9]	imx8mm-evk.dtb	N/A
imx8mmevk	uSD-M.2 ^[10]	imx8mm-evk.dtb	SDIO
imx8mnddr4evk	uSD-M.2+	imx8mn-evk.dtb	SDIO
imx6ulevk	uSD-M.2	imx6ul-14x14-evk-btwifi.dtb ^[11]	SDIO
imx6ull14x14evk	uSD-M.2	imx6ull-14x14-evk-btwifi.dtb ^[11]	SDIO

3.3 NXP's Default Demo Image

To easily enable NXP-based Wi-Fi/Bluetooth functionality, users must download [default NXP's demo images](#) of 5.10.52 or 5.10.72 for i.MX platforms.

With the Linux demo images, [Section 4.1](#) outlines flashing steps for (micro) SD card - on all platforms except 8MMINILPD4-EVKB and 8MNANOD4-EVK. Steps for flashing eMMC on the i.MX 8M Mini/Nano EVKs is detailed in [Section 4.2](#).



Only 5.10.72 supports 2DS.

⁷ Wi-Fi/BT M.2 EVB plugs directly into M.2 connector

⁸ Works with uSD-M.2 Adapter configured for 1.8V VIO default

⁹ Wi-Fi/BT M.2 EVB plugs directly into M.2 connector; but only WLAN is functional

¹⁰ Works with uSD-M.2 Adapter (Rev B2) with additional cabling

¹¹ Custom DTB file for 6UL(L) configured for 1.8V

3.4 Additional Hardware/Software Considerations

3.4.1 1.8V Versus 3.3V VIO Signaling using Murata's uSD-M.2 Adapter

As shown in **Table 5** and **Table 7**, the only NXP i.MX 6 Platforms supported include i.MX 6UL(L) EVKs. This is due to a WLAN-SDIO VIO limitation on both Type 1ZM and 1YM. Both modules only support a WLAN-SDIO VIO of 1.8V. The NXP i.MX 6UL(L) EVKs are the only i.MX 6 Platforms (with appropriate interconnect) which can support this required 1.8V signaling over WLAN-SDIO bus.




The Murata uSD-M.2 Adapter should never be configured in "3.3V VIO Override Mode" when connected to Type 1ZM or 1YM M.2 EVBs.



The BT-UART and some WLAN/BT control signals are level-shifted on the uSD-M.2 Adapter (default configuration) from 3.3V VIO (Host) to 1.8V VIO (M.2).

Refer to the [Murata Wi-Fi/BT Solution for i.MX Hardware User Manual](#)  for more details.

3.4.2 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 6UL(L) EVK, the maximum SDIO clock frequency is limited to 50 MHz. 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 6UL(L) EVKs require the additional 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.

3.4.3 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 EVKs does support direct M.2 interconnect. However, there are specific limitations on the existing platforms noted in this document:

- Only WLAN-PCIe is supported. No WLAN-SDIO interconnect is supported out-of-box.



Note that the i.MX 8MQuad EVK can be reworked to connect WLAN-SDIO signals but that is not supported currently by Murata.

- Both NXP i.MX 8M Mini EVKs only have WLAN-PCIe interconnect and neither platform supports 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.

3.4.4 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 file – one time only:**

Set DTB (Device Tree Blob) file correctly (“fdt_file”/“fdtfile” 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. Refer to **Table 7** for more details regarding correct DTB file selection. Refer to [Section 4.1.3](#) for instructions on loading DTB files on flashed SD card.



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. It is necessary that users flash the DTB file for 6UL(L) present in the [Murata Linux Patched Solution](#) under the folder, patch_files” .

- **Load Drivers – Every time:**

Enter the following command:

```
modprobe moal mod_para=nxp/wifi_mod_para.conf
```

3.4.5 Type 1YM M.2 EVB WLAN/Bluetooth Bus Interface Configuration

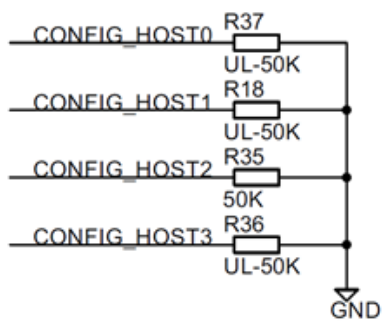
Type 1YM module supports more than one interface configuration. The Embedded Artists' Wi-Fi/BT M.2 EVB (EAR00370) is default configured (via resistor strapping options) for WLAN-PCIe/BT-UART. However, the following configuration options should be considered:

- **WLAN-PCIe/BT-UART (1YM or 1YM-PCIe):** default configuration and supported in software release. For existing interconnect, this mode is used when plugging the M.2 EVB directly into NXP M.2 connector.
- **WLAN-SDIO/BT-UART (1YM-SDIO):** currently not supported in software release, this configuration will be a supported option in NXP baseline BSP release. See **Figure 7** for resistor strapping option – two 50K Ohm resistors need to be added (to right of default one).
- **WLAN-SDIO/BT-SDIO (1YM-SDIO*):** currently supported in software release, this configuration is for customer evaluation purposes only. It is not intended for final shipping product. The customer must have special software release access on NXP website to build the Linux image with this support option. The Murata build script provides necessary details on NXP software package name. See **Figure 7** for resistor strapping option – one 50K Ohm resistor needs to be added (to right of default one).

Regarding the actual configuration of the Embedded Artists Type 1YM M.2 Module (EVB), the customer needs to perform some minor rework to modify the resistor strapping options – if not going with WLAN-PCIe/BT-UART default. The necessary rework (addition of 50K Ohm 0201 resistors) is easily illustrated in **Figure 7** which shows both relevant schematic capture and configuration-strapping table. The “CONFIG_HOST” resistors are (in order left to right as pictured):

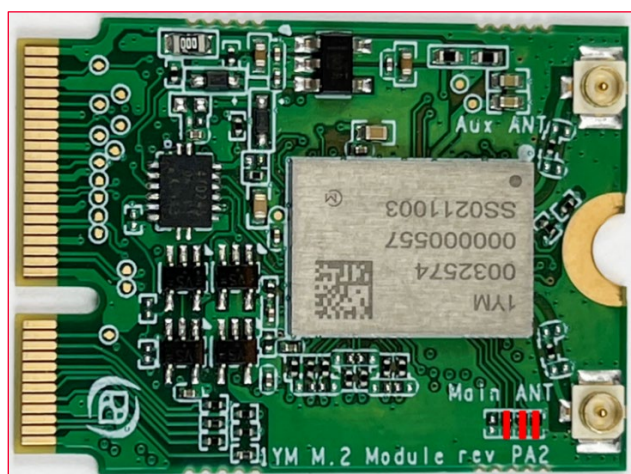
- CONFIG_HOST3
- CONFIG_HOST2 (default resistor already installed for WLAN-PCIe/BT-UART)
- CONFIG_HOST1 (install this resistor only for WLAN-SDIO/BT-SDIO option)
- CONFIG_HOST0 (install this resistor in addition to #1 for WLAN-SDIO/BT-UART option)

Figure 7: Type 1YM M.2 EVB Configuration Strapping Options

	CONFIG_HOST[2:0]	WLAN	Bluetooth	
	0 0 0	SDIO	UART	
	0 0 1	SDIO	SDIO	
	0 1 0	PCIe	USB 2.0	
	0 1 1	PCIe	UART	Default
	1 0 0	USB 3.0/2.0	UART	
	1 0 1	USB 2.0	USB 2.0	
	1 1 0	USB 3.0/2.0	USB 3.0/2.0	
	1 1 1	USB 3.0	USB 3.0	

0 = strap resistor mounted
1 = strap resistor not mounted

Figure 8: Type 1YM M.2 Configured for WLAN-SDIO/BT-UART (1YM-SDIO)




3.5 Murata Customized i.MX Yocto Image Build

Murata's solution to easily enable NXP-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. 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.

3.5.1 Install Ubuntu

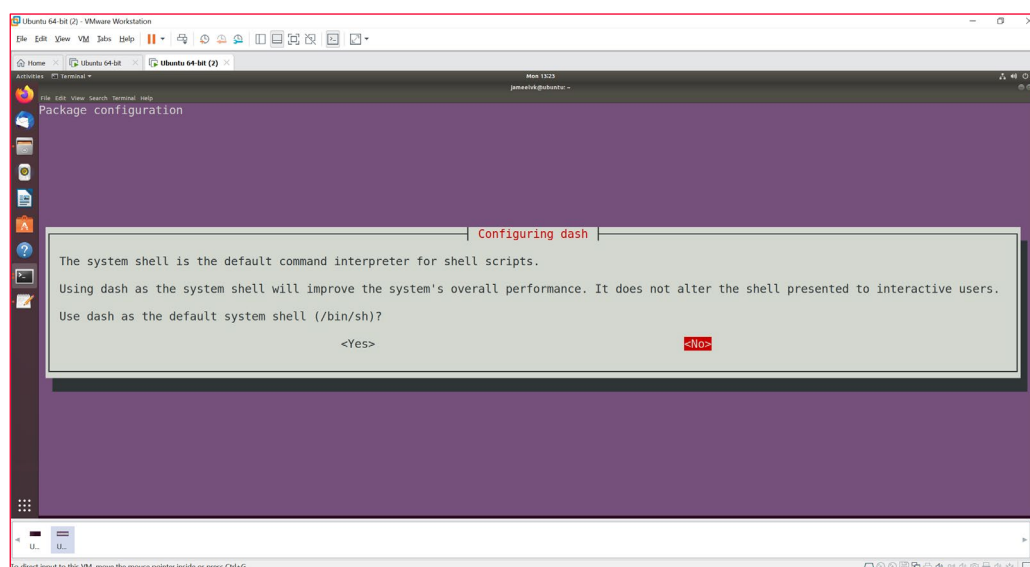
First step is to install Ubuntu 14.04, 16.04, 18.04 or 20.04 (Murata’s build is verified on Ubuntu 20.04 64-bit install) on the host - native PC or virtual environment like VMware. Host PC typically used requires Ubuntu 20.04/18.04/16.04/14.04 installed with 50 GB free disk space (80 GB needed for i.MX8 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 9**.

Figure 9: Configuring dash



3.5.2 Download Murata’s Script Files

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

1. 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.
2. 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_NXP.sh

chmod a+x *.sh
```

3.5.3 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-NXP.txt 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:

1. Verifying Host Environment
2. Verifying Host Script Version
3. Installing Essential Yocto host packages
4. GIT Configuration: verifying Username and email ID

3.5.4 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_NXP.sh](#) . For plain ASCII text version, you can go to [this link](#)  or just hit the “Raw” button. For more information (README-NXP.txt 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 3.5.3](#)  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:
 - 5.10.52_2.1.0
 - 5.10.72_2.2.0
- Creating the i.MX BSP folder is straightforward:

```
cd ~  
mkdir murata-imx-bsp  
cd murata-imx-bsp  
cp <Script Path>/Murata_Wireless_Yocto_Build_NXP.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 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_NXP.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 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:


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

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

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

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

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

With 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.

4 Preparing NXP i.MX Platforms to Boot Linux Image

This section describes 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.2+” 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
imx8mmevk	M.2 ^[12]	imx8mm-evk.dtb	uSD	N/A
imx8mmevk	uSD-M.2 ^[13]	imx8mm-evk.dtb	eMMC	SDIO
imx8mnddr4evk	uSD-M.2 ⁺	imx8mn-evk.dtb	eMMC	SDIO

4.1 Flashing Linux Image to (Micro) SD Card

This section presents supports 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 you can now flash the (micro) SD card used for booting the i.MX platform.

1. 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 10**. 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).

¹² Wi-Fi/BT M.2 EVB plugs directly into M.2 connector; but only WLAN is functional

¹³ Works with uSD-M.2 Adapter (Rev B2) with additional cabling

Figure 10: USB to SD Card Reader/Writer Adapter

2. 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.

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.

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”¹⁴ 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 utilities¹⁵.

¹⁴ “Win32 Disk Imager” is an open-source tool that can be downloaded from websites such as “sourceforge.net”.

¹⁵ Unlike Linux environment, Windows PC does not require use of “USB to SD Card Reader/Writer” adapter.

- Now click the “Write” button. A warning window will pop up that warns that the device being written to may be corrupted.
- Upon completion, a window with “Write Successful” should appear.
- Click “OK” on the “Write Successful” window.
- Now click “Exit” on “Win32 Disk Imager” window.
- To be safe, you may elect to “eject” the SD card removable memory device before removing it.

4.1.3 Steps to Load DTB Files

To load modified DTB files (such as those provided by the [Murata Linux Patched Solution](#)), please plug in the flashed SD card to the host PC. The boot partition containing the existing DTB files will be detected. Copy the new DTB files in this partition, unmount the SD card and plug out.

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 8 M 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.

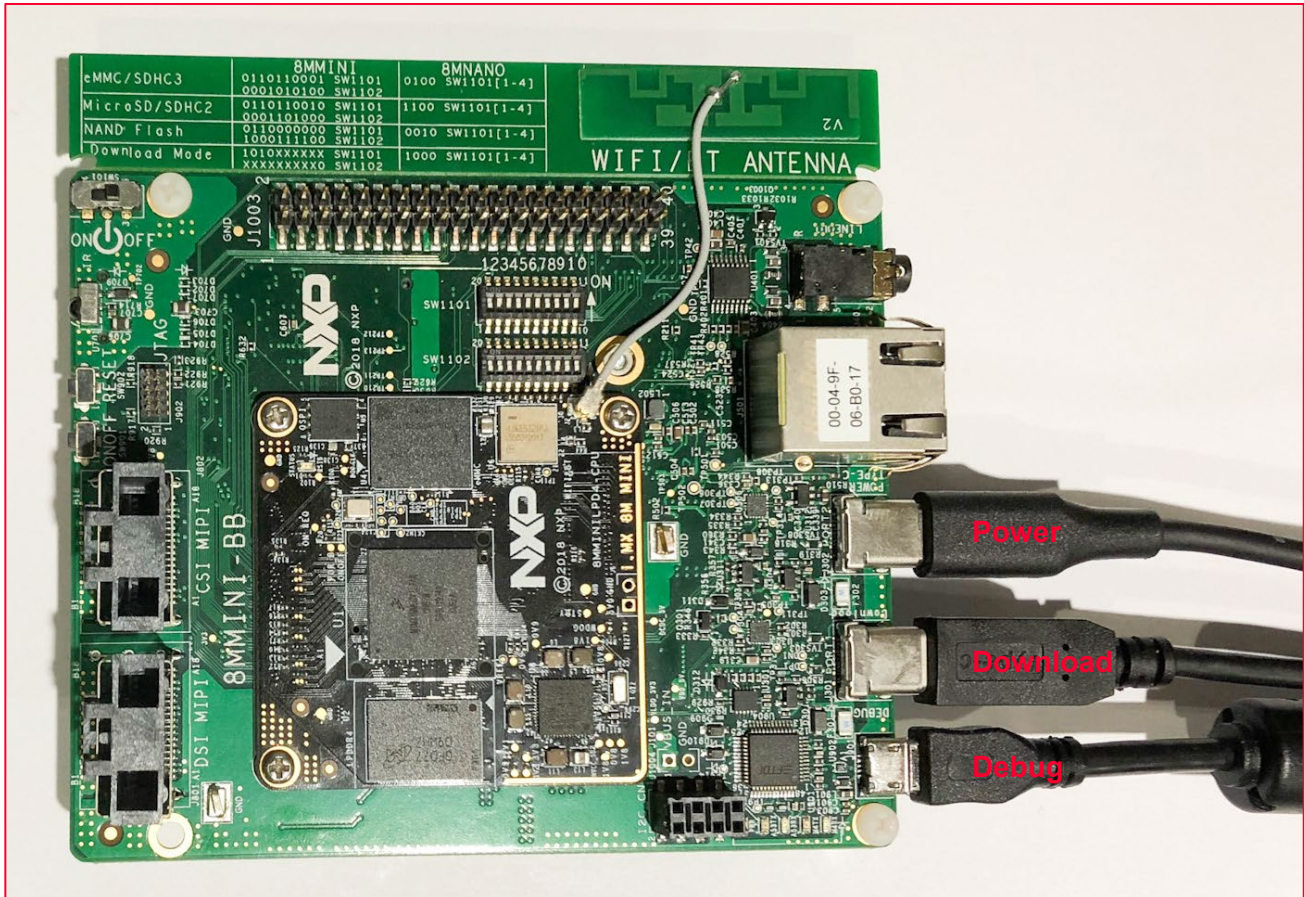
Table 9: Files to Flash i.MX 8M Mini & 8M Nano EVKs

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 11** shows the necessary connections for power, download, and debug (serial console).

Figure 11: 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.



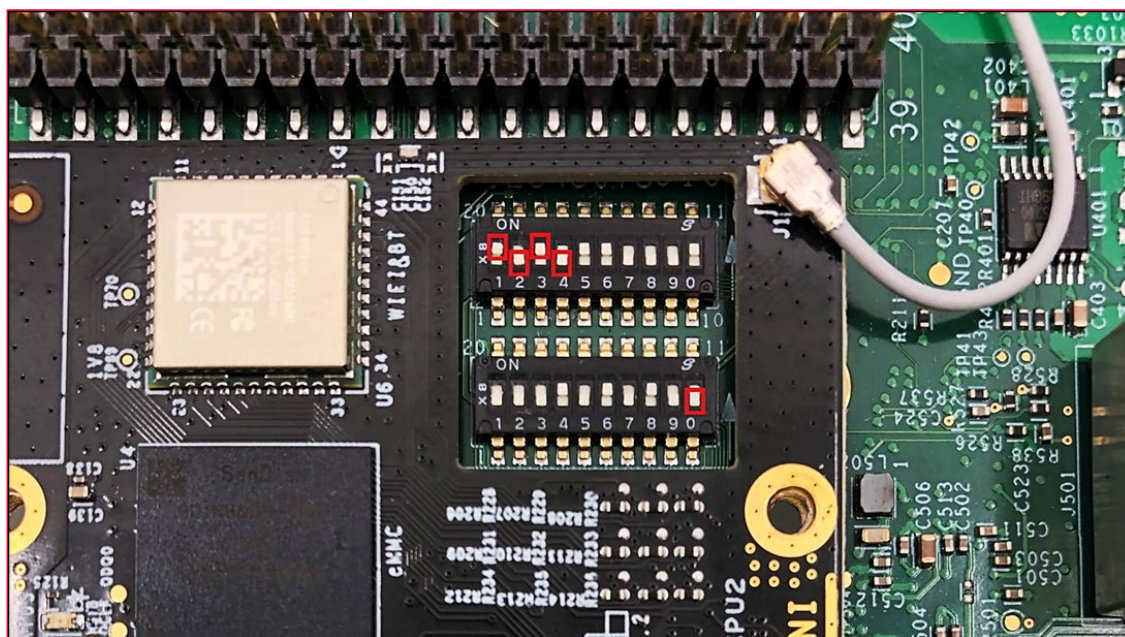
The DIP switch settings for i.MX 8M Mini EVK and 8M Nano EVK are different.

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 12**.

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 12: i.MX 8M Mini EVK DIP Switches Configured for Download

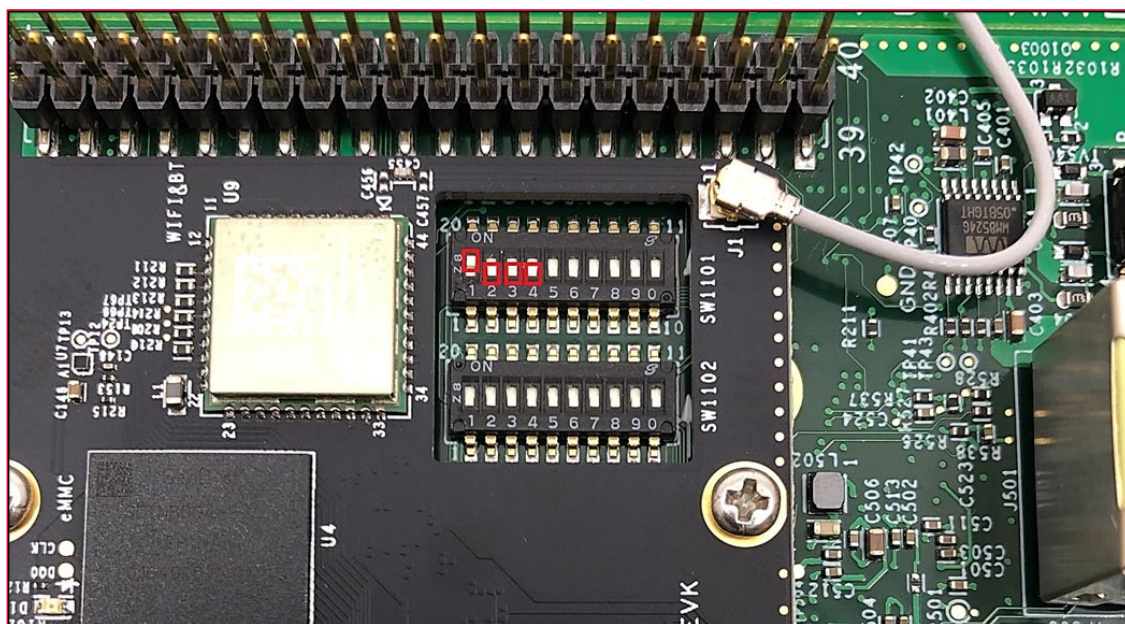


For i.MX 8M Nano EVK (8MNANOD4-EVK), this is accomplished by setting the DIP switch (SW1101) to the positions below. The DIP switch settings are also shown in **Figure 13**.

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 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.

1. 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.
2. 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/*
```

3. 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.

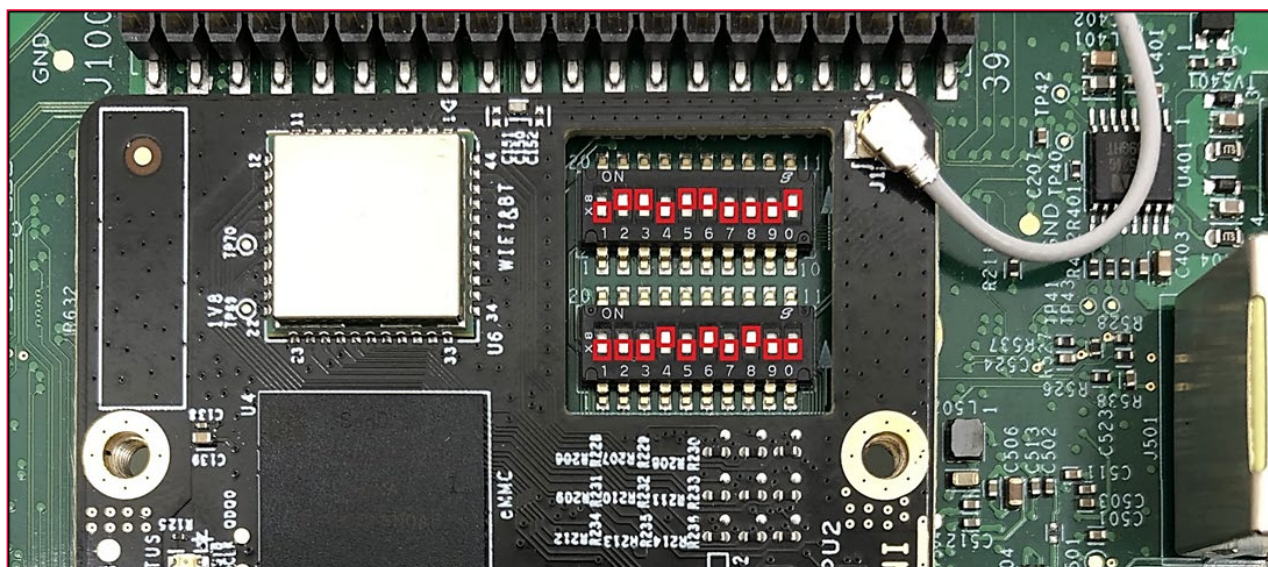
Now you *must* change the DIP switch settings to boot from eMMC. Note that the DIP switch settings for i.MX 8M Mini EVK and 8M Nano EVK are different.

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 14**.

	#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 14: i.MX 8M Mini EVK DIP Switches Configured for eMMC Boot

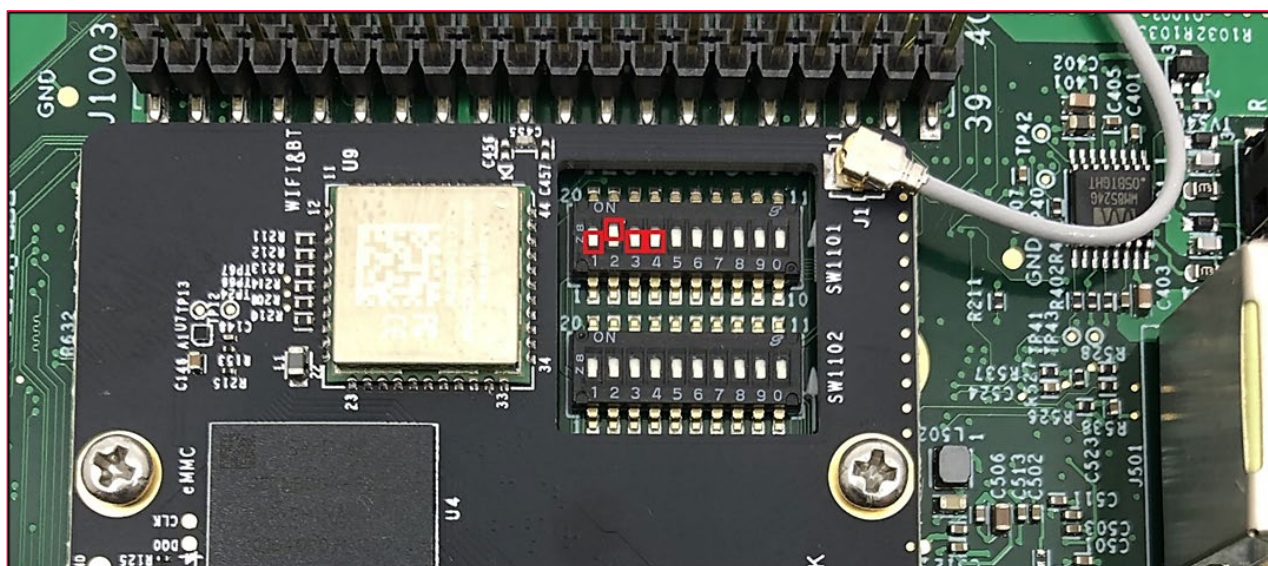


For i.MX 8M Nano EVK (8MNANOD4-EVK), this is accomplished by setting the DIP switch (SW1101) to the positions below. The DIP switch settings are also shown in **Figure 15**.

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

Where 1 – ON, 0 – OFF.

Figure 15: 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 1ZM and 1YM-SDIO*) can be connected to the i.MX 6UL(L) EVKs through Murata's uSD-M.2 Adapter as shown in **Figure 16**. The following sub-sections details steps for bringing up Embedded Artists' Wi-Fi/BT EVBs on these EVKs.



Type 1ZM and Type 1YM-SDIO M.2 EVBs (and modules) only support WLAN-SDIO VIO of 1.8V. This rules out interfacing these 1ZM/1YM M.2 EVBs to the NXP i.MX 6Q(P)/DL/SoloX SDBs which only support 3.3V VIO over WLAN-SDIO interface.

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 B2 uSD-M.2 Adapter is equivalent to Rev B1, with the only component change being the sleep clock (32 KHz). Both Rev B2 and B1 incorporates level shifting to provide the necessary BT-UART 1.8V VIO (M.2) and WLAN/BT control signals. 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). Customers are recommended to use the latest Rev B2 Adapter with correct voltage signaling on BT-UART.


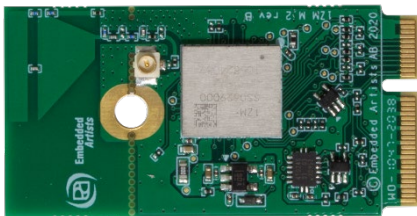
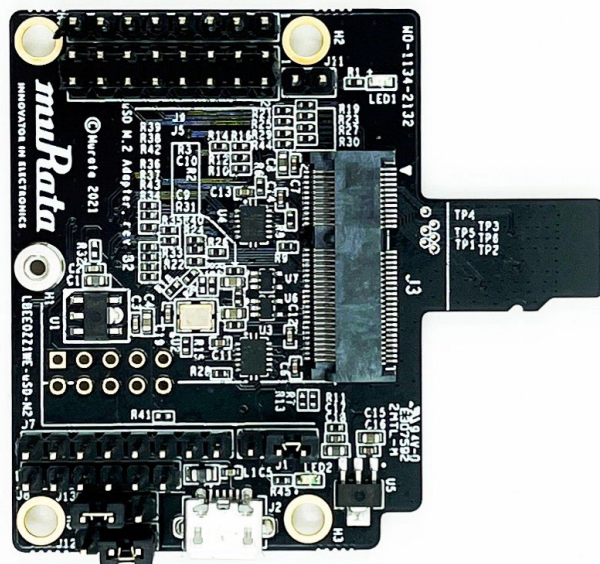
For more information on hardware configuration refer to the [Murata Wi-Fi/BT Solution for i.MX Hardware User Manual](#) .

Figure 16: uSD-M.2 Adapter with Type 1ZM and 1YM-SDIO* M.2 EVB Options

Type 1ZM M.2 EVB










Type 1YM M.2 EVB
(WLAN-SDIO)



5.1 Connecting to i.MX 6UL EVK or i.MX 6ULL EVK

To connect

1. 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.
2. 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.
3. Connect the 1ZM or 1YM (reworked for WLAN-SDIO operation per [Section 3.4.5](#) ) 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 per [Section 8.3](#) .
4. Connect ribbon cable at both ends before inserting Murata EVK into SD1 slot. Note the orientation as shown in **Figure 17**. 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 per [Section 8.3](#) .
5. Prepare microSD card to boot platform per [Section 4.1](#) .
6. Fetch DTB file of 6UL(L) configured for 1.8V VIO from [here](#)  and copy it to “boot” folder in microSD card (Refer to [Section 8.1.3](#)  for details). Actual DTS file for arriving at the new DTB file of 6UL(L) is also provided for reference.
7. Insert microSD card, power on the platform and interrupt at u-boot. Set DTB configuration with “fdt_file” parameter. You can check available DTB files using command “fatls mmc 1”. With DTB set, save the u-boot configuration and boot platform:

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

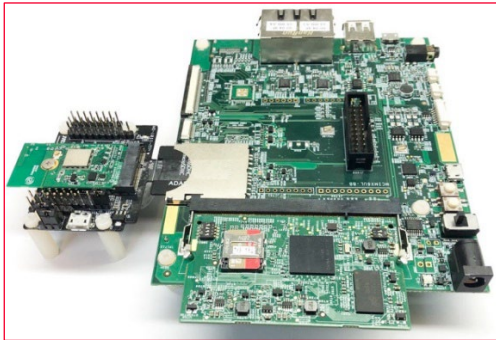
# Boot kernel
boot
```

8. Load Drivers – Every time:

After the kernel boots, Enter the command, “modprobe moal mod_para=nxp/wifi_mod_para.conf” for loading WLAN driver

```
modprobe moal mod_para=nxp/wifi_mod_para.conf
```

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

Figure 17: i.MX 6ULL EVK with uSD-M.2 Adapter and Type 1ZM M.2 EVB

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

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

The NXP i.MX 8MQuad EVK (see **Figure 18**) 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 1YM (WLAN-PCIe).

1. Connect J1701 micro-USB port to PC and start terminal emulator: “Minicom” on Linux or “Tera Term” on Windows. Set port to 115200-N-8-1.
2. Connect Embedded Artists the 1YM 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.
3. 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**. 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
saveenv

# Boot kernel
boot
```

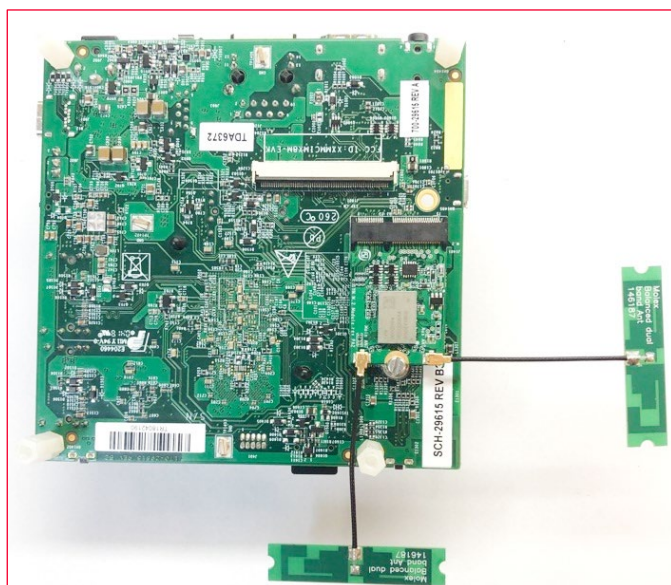
4. Load Drivers – Every time:

After the kernel boots, Enter the command, “modprobe moal mod_para=nxp/wifi_mod_para.conf” for loading WLAN driver

```
modprobe moal mod_para=nxp/wifi_mod_para.conf
```

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

Figure 18: i.MX 8MQuad with Type 1YM (Bottom View)



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

Both NXP i.MX 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 19** for upside-down EVK view showing M.2 connector (currently the only supported M.2 EVB is Type 1YM). The steps below detail how to bring up the Wi-Fi/BT M.2 EVB.

1. Connect J901 micro-USB port to PC and start terminal emulator: “Minicom” on Linux or “Tera Term” on Windows. Set port to 115200-N-8-1.
2. Connect Embedded Artists Type 1YM M.2 EVB to M.2 connector as shown in **Figure 19**. 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.
3. Prepare the platform to boot per [Section 4.2](#). Power on the platform and interrupt at u-boot. Set DTB configuration with “fdt_file” parameter for correct platform per **Table 7**. Check the available DTB files by invoking “fatls mmc 1”. Now save u-boot configuration and boot the platform:

```
setenv fdtfile imx8mm-evk.dtb
saveenv

# Boot kernel
boot
```

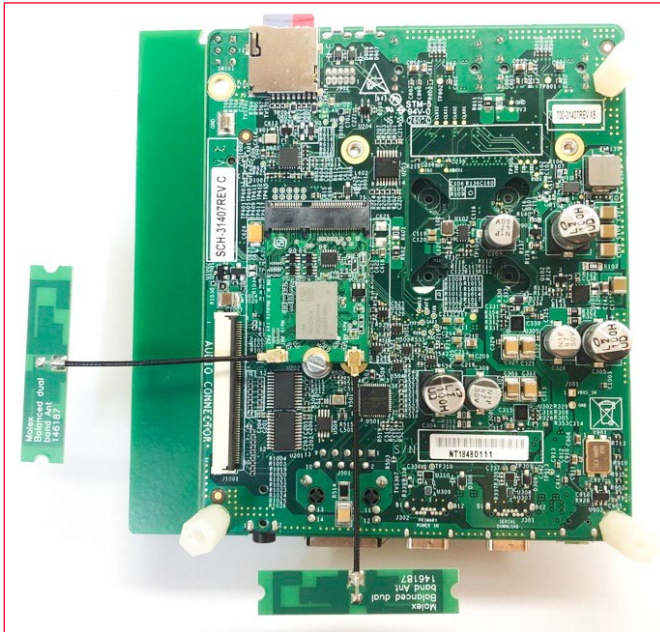
4. Load Drivers – Every time:

After the kernel boots, Enter the command, “modprobe moal mod_para=nxp/wifi_mod_para.conf” for loading WLAN driver

```
modprobe moal mod_para=nxp/wifi_mod_para.conf
```

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

Figure 19: i.MX 8M Mini with Type 1YM-PCle (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 (Figure 20)
- WLAN/Bluetooth (Figure 21).

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

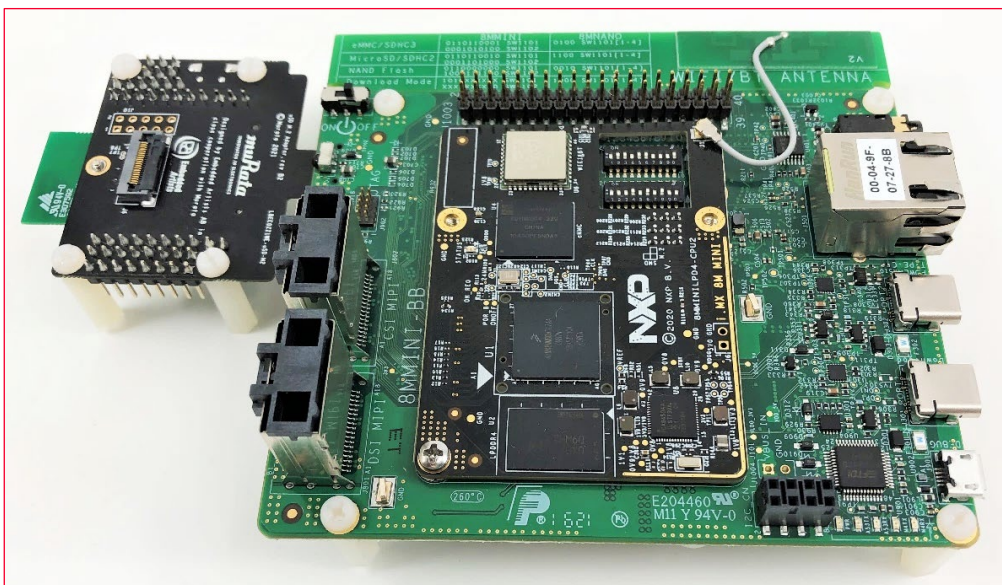
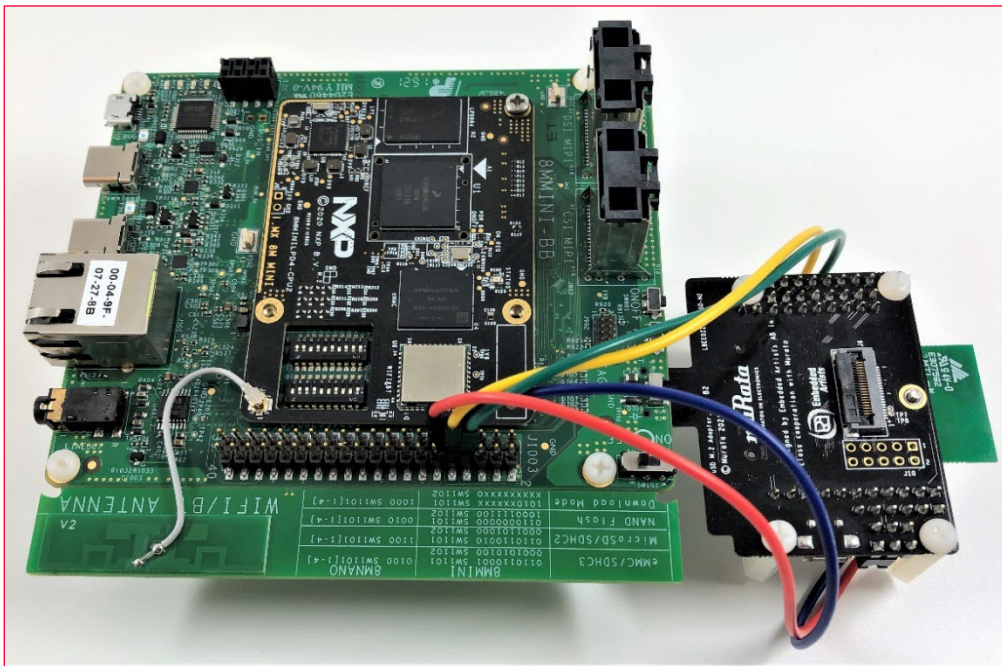


Figure 21: i.MX 8M Mini/Nano EVK with uSD-M.2 Adapter (WLAN/Bluetooth)


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 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).

Here are the steps:

1. Connect J901 micro-USB port to PC and start terminal emulator: "Minicom" on Linux or "Tera Term" on Windows. Set port to 115200-N-8-1.
2. 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.
3. Connect the 1ZM or 1YM (reworked for WLAN-SDIO operation as per [Section 3.4.5](#)) Wi-Fi/BT M.2 EVB to the M.2 connector on the uSD-M.2 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](#).
4. For full Wi-Fi/BT configuration (Bluetooth-UART and WLAN/BT control lines), connect four jumper wires from the uSD-M.2 adapter to the i.MX 8M Mini/Nano EVK per [Table 10](#). Refer to [Figure 22](#), [Figure 23](#), [Figure 24](#) and [Figure 25](#) 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 26](#), and [Figure 27](#). 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 28**, 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 20**), 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).

5. Per [Section 4.2](#), flash eMMC on i.MX 8M Mini/Nano EVK; and then configure DIP switch settings for eMMC boot.
6. Power on the platform and interrupt at u-boot. Set DTB configuration with “fdtfile” parameter for correct platform per **Table 7**. You can check available DTB files using the command “fatls mmc 2”. After setting DTB, save the u-boot configuration and boot the platform:

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

# Boot kernel
boot
```

7. After the kernel boots, enter the following command for loading WLAN driver.

```
modprobe moal mod_para=nxp/wifi_mod_para.conf
```

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	Notes
BT_UART_TX	J9 / Pin 1	10	UART Tx line
BT_UART_RX	J9 / Pin 2	8	UART Rx line
BT_UART_RTS	J8 / Pin 3	11	UART RTS line
BT_UART_CTS	J8 / Pin 4	7	UART CTS line

The figures below for visual examples of the steps described above. Refer to [Section 7](#) to test/verify Wi-Fi and Bluetooth functionality.

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

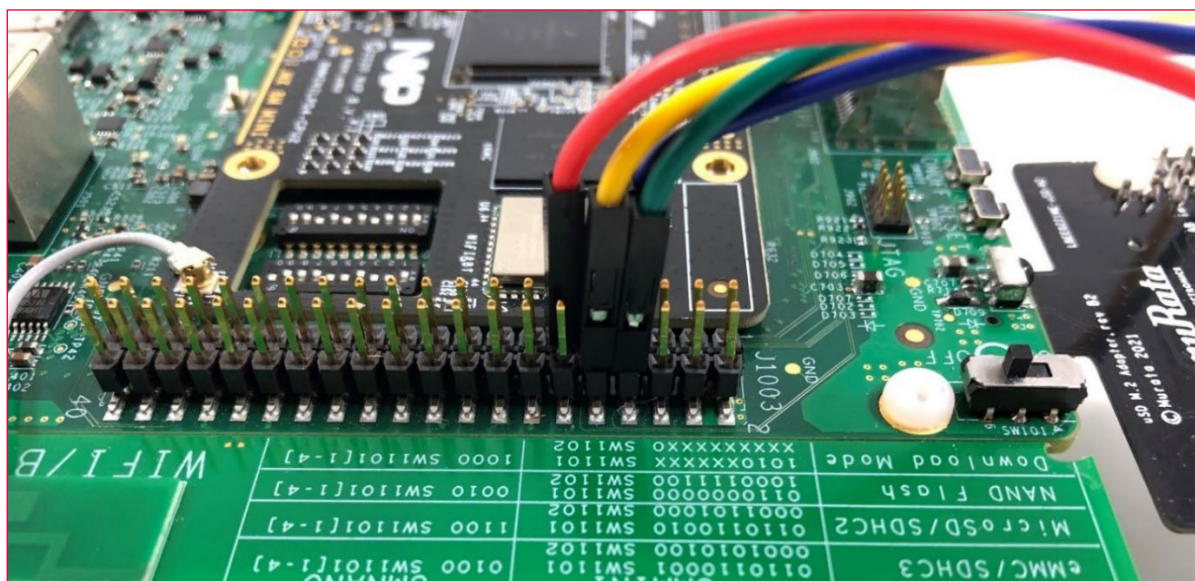


Figure 23: Cable Connections on i.MX 8M Mini EVK (Odd Number Connector View)

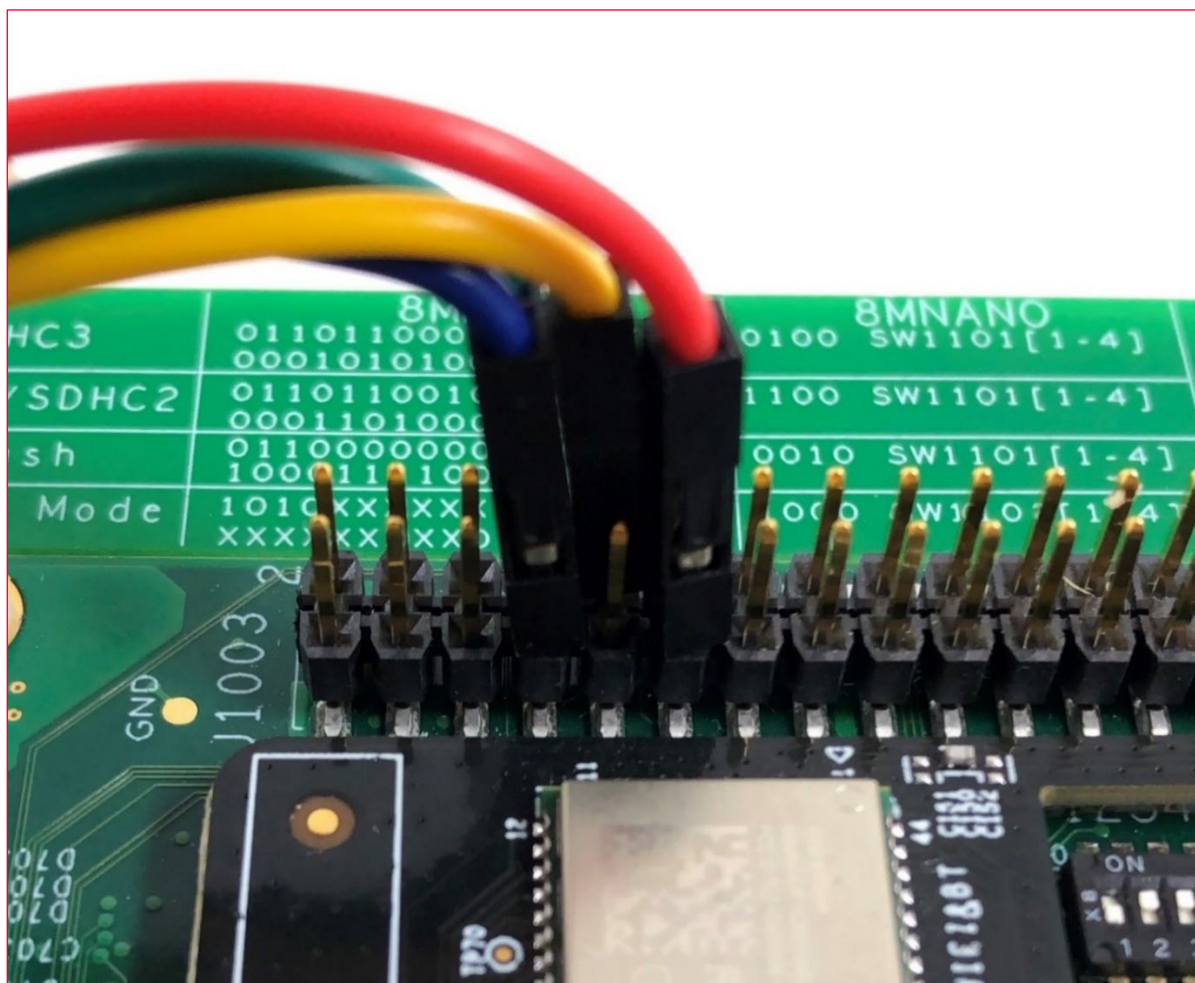


Figure 24: Cable Connections on uSD-M.2 Adapter (J9 Header)

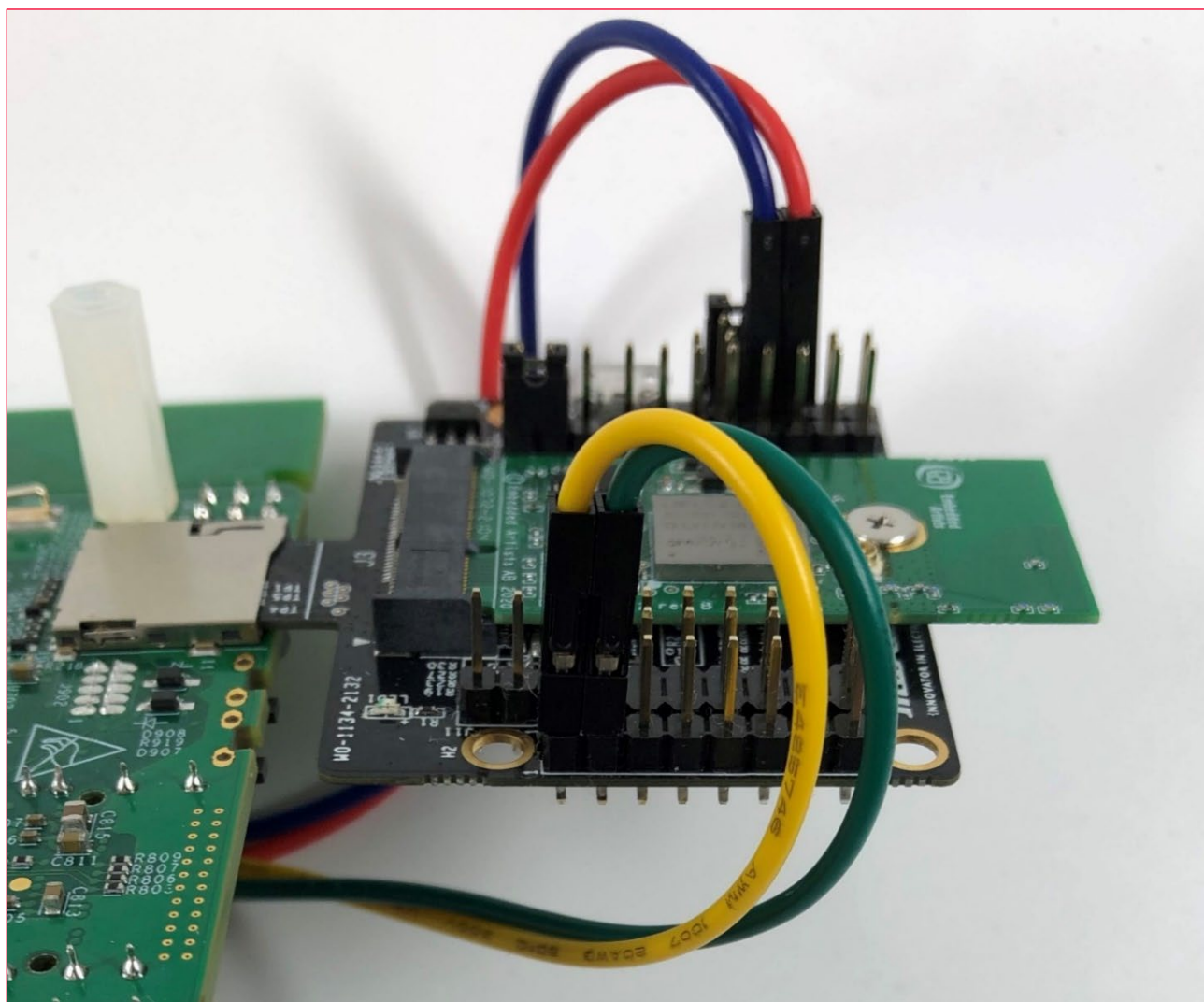


Figure 25: Cable connections on uSD-M.2 Adapter (J8 Header)

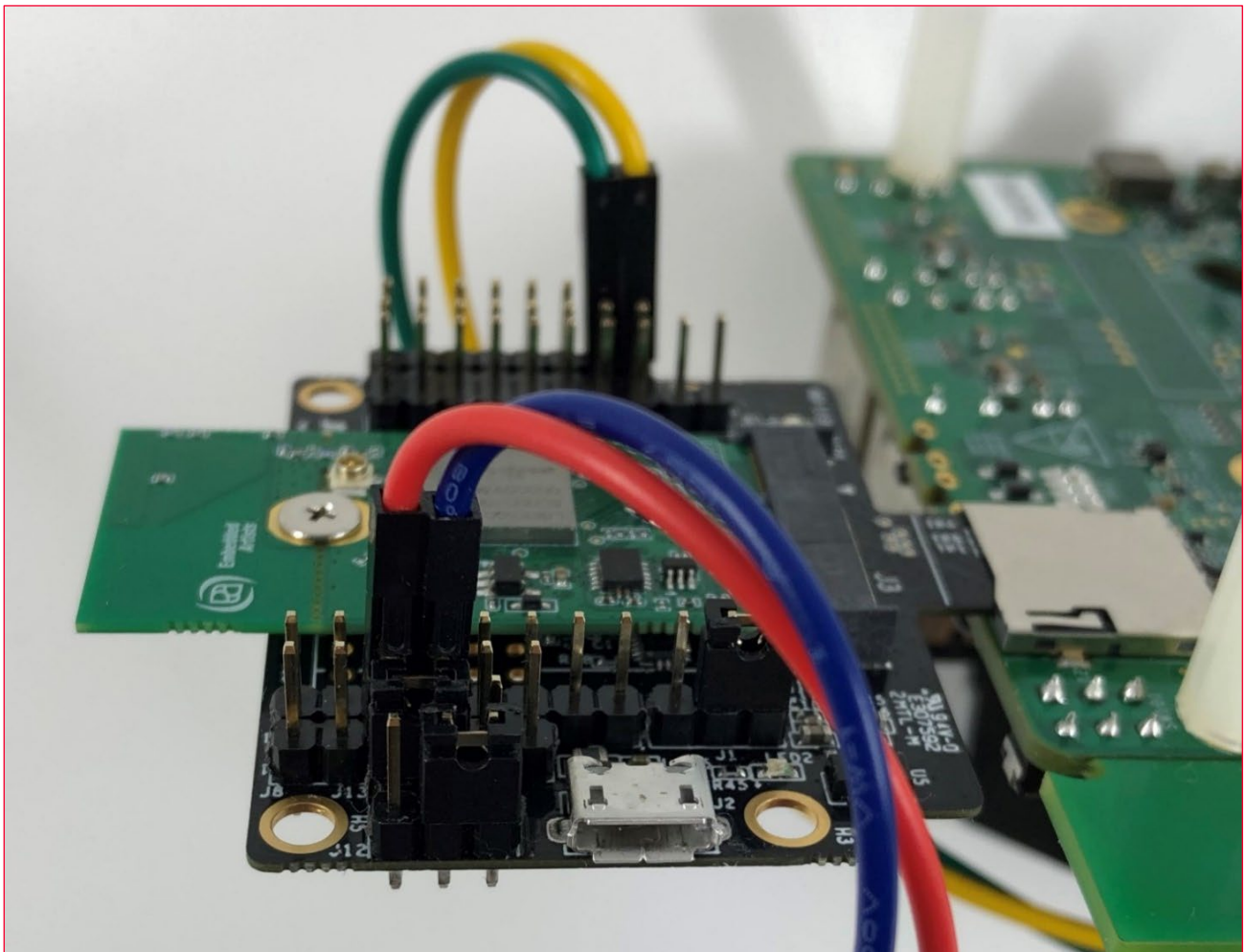


Figure 26: Low-Profile Jumper Wires (Digi-Key part number 1988-1178-ND)

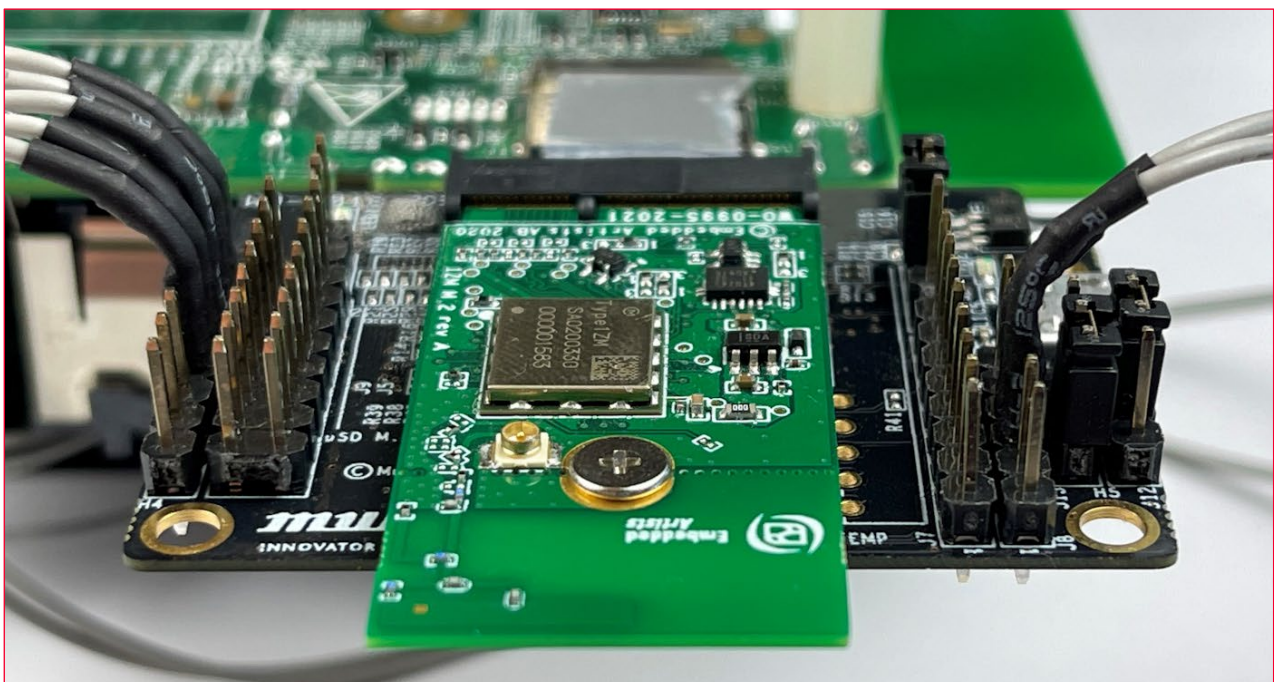
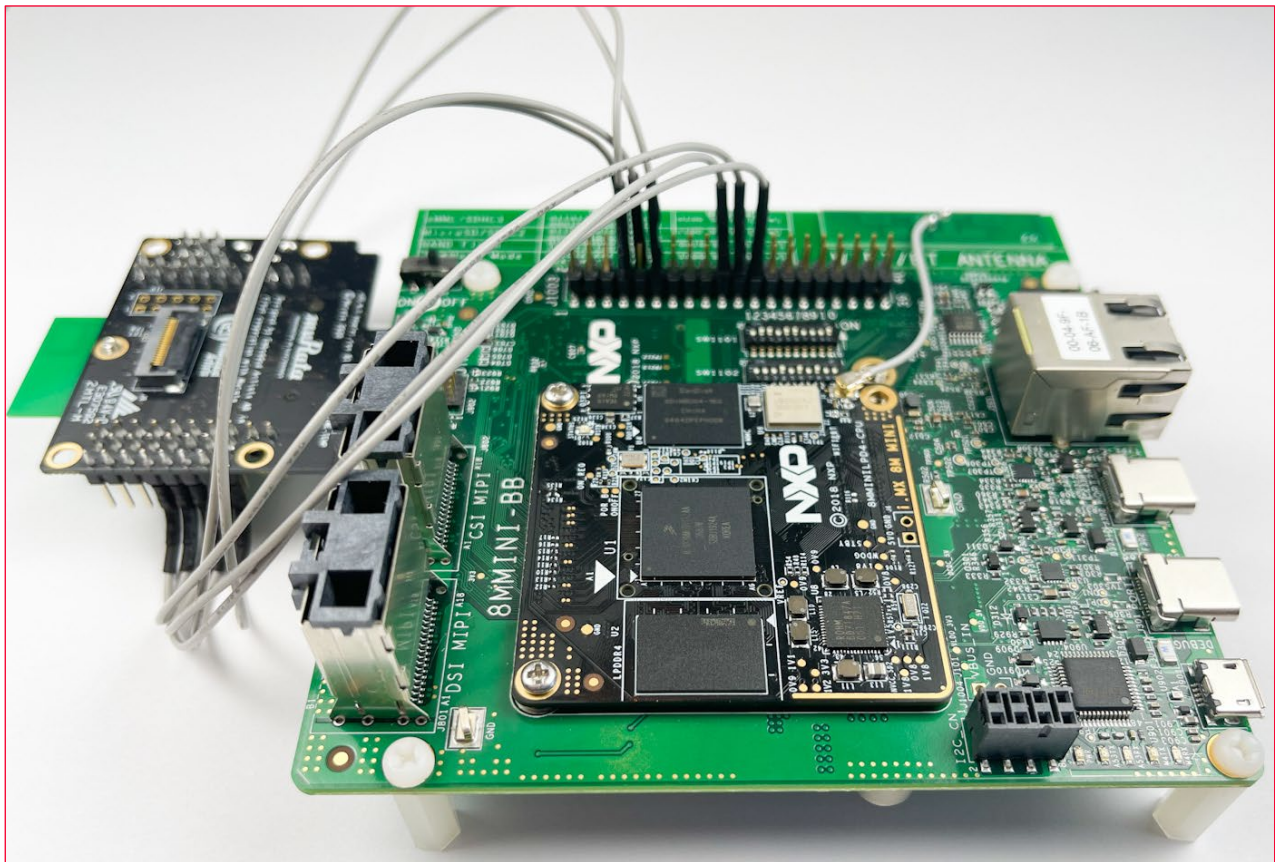


Figure 27: NXP i.MX EVK with Low-Profile Jumper Wires



7 Test/Verification of Wi-Fi and Bluetooth

The kernel should be booting correctly on the NXP i.MX platform with Murata module being correctly initialized (correct DTB configured and NXP driver loaded using modprobe command). 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. The relevant folders and files are summarized in **Table 11**.

Table 11: Embedded Wi-Fi/Bluetooth Files

Filename or Folder	Details
/usr/share/nxp_wireless/mlanutil	NXP utility application for controlling WLAN STA interface and RF testing
/usr/sbin/iw	Linux “iw” executable.
/usr/sbin/wpa_supplicant	WPA supplicant executable.
/usr/sbin/wpa_cli	WPA CLI tool.
/usr/bin/wpa_passphrase	WPA Passphrase generator.
/etc/wpa_supplicant.conf	WPA supplicant configuration file.
/usr/sbin/hostapd	Hostapd executable – manages wireless link in Soft AP mode.
/usr/sbin/hostapd_cli	Hostapd CLI tool.
/etc/hostapd.conf	Hostapd configuration file.
/usr/bin/hciattach	“hciattach” binary – used for initializing Bluetooth UART connection.
/usr/bin/hciconfig	“hciconfig” binary – used for configuring Bluetooth interface.
/usr/bin/hcitool	“hcitool” binary – used for controlling Bluetooth interface.
/usr/bin/iperf3	iPerf throughput test tool.

7.1 Wi-Fi Interface Test/Verification

This section describes the Wi-Fi interface test/verification.

7.1.1 Useful Environment Setup on NXP Linux

Once the kernel has booted and you have 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 Bringing Up Wi-Fi Interface

This section describes how to bring up the Wi-Fi interface.

7.1.2.1 For i.MX6UL(L)

1. Ensure that correct DTB file is set for 6UL(L) as mentioned in [Section 5.1](#) .
2. After the kernel boots, enter the command, “modprobe moal mod_para=nxp/wifi_mod_para.conf” for loading WLAN driver.

```
modprobe moal mod_para=nxp/wifi_mod_para.conf
```



This step needs to be performed every time user powers up the target.

3. As part of driver loading sequence (in this example), the WLAN device is probed over the SDIO 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 1ZM M.2 Module (EVB)
- NXP's demo image for i.MX 6ULL

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

```
root@imx6ul7d:~# modprobe moal mod_para=nxp/wifi_mod_para.conf
[ 39.989170] mlan: loading out-of-tree module taints kernel.
[ 40.135334] wlan: Loading MWLAN driver
[ 40.230733] vendor=0x02DF device=0x9149 class=0 function=1
[ 40.236506] Attach moal handle ops, card interface type: 0x105
[ 40.256268] SD8987: init module param from usr cfg
[ 40.261322] card type: SD8987, config block: 0
[ 40.265791] cfg80211_wext=0xf
[ 40.270982] wfd_name=p2p
[ 40.273550] max_vir_bss=1
[ 40.276189] cal_data_cfg=none
[ 40.281229] drv_mode = 7
[ 40.283795] ps_mode = 2
[ 40.286253] auto_ds = 2
[ 40.291149] fw_name=nxp/sdjouart8987_combo_v0.bin
[ 40.296203] SDIO: max_segs=128 max_seg_size=65535
[ 40.302342] rx_work=0 cpu_num=1
[ 40.305552] Attach mlan adapter operations.card_type is 0x105.
[ 40.319323] wlan: Enable TX SG mode
[ 40.322844] wlan: Enable RX SG mode
[ 40.330051] Request firmware: nxp/sdjouart8987_combo_v0.bin
[ 40.890551] Wlan: FW download over, firmwarelen=530240 downloaded 530240
[ 41.849051] WLAN FW is active
[ 41.852105] on_time is 41833538837
[ 41.876768] fw_cap_info=0x181c3f03, dev_cap_mask=0xffffffff
[ 41.882541] max_p2p_conn = 8, max_sta_conn = 8
[ 41.971990] wlan: version = SD8987---16.92.10.p210.1-MM5X16266.p4-GPL-
(FP92)
[ 42.043020] wlan: Driver loaded successfully
```

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

- “wlan” string identifies driver log messages
 - “card_type: SD8987” string identifies the chipset being 88W8987.
 - “sdjouart8987_combo_v0.bin” indicates the WLAN/Bluetooth firmware file being loaded.
 - “version” indicates specific version of firmware being loaded by the driver.
4. Now invoke “ifconfig wlan0 up” command to initialize the “wlan0” (WLAN) interface.

```
$ ifconfig wlan0 up
$ ifconfig wlan0

# "wlan0" interface is UP. WLAN MAC Address is shown
wlan0: flags=4099<UP, BROADCAST,MULTICAST> mtu 1500
    ether d4:53:83:c1:b9:d6 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```



No IP address is assigned yet to the “wlan1” interface. That will be done later in [Section 7.1.6](#).

7.1.2.2 For i.MX 8M-Mini/Nano

1. Ensure that correct DTB file is set for 8M-Mini/Nano as mentioned in [Section 6.2](#) / [Section 6.3](#).
2. After the kernel boots, enter the command, “modprobe moal mod_para=nxp/wifi_mod_para.conf” for loading WLAN driver. This step needs to be performed every time user powers up the target.

```
modprobe moal mod_para=nxp/wifi_mod_para.conf
```

As part of driver loading sequence (in this example), the WLAN device is probed over the SDIO bus. As an example, we will bring up Wi-Fi when using following configuration:

- NXP i.MX 8M-Mini EVKB
- Murata’s uSD-M.2 Adapter
- Embedded Artists’ Type 1ZM M.2 Module (EVB)
- NXP’s demo image for i.MX 8M-Mini

Expected output after loading the driver for 1ZM:

```
root@imx8mmevk: ~# modprobe moal mod_para=nxp/wifi_mod_para.conf
[ 28.854770] wlan: loading out-of-tree module taints kernel.
[ 28.879436] wlan: Loading MWLAN driver
[ 28.884290] vendor=0x02DF device=0x9149 class=0 function=1
[ 28.889861] Attach moal handle ops, card interface type: 0x105
[ 28.896355] SD8987: init module param from usr cfg
[ 28.901187] card type: SD8987, config block: 0
[ 28.905650] cfg80211_wext=0xf
[ 28.908629] wfd_name=p2p
[ 28.911176] max_vir_bss=1
[ 28.913798] cal_data_cfg=none
[ 28.916778] drv_mode = 7
[ 28.919326] ps_mode = 2
[ 28.921770] auto_ds = 2
[ 28.924227] fw_name=nxp/sdiouart8987_combo_v0.bin
[ 28.928973] SDIO: max_segs=128 max_seg_size=65535
[ 28.933697] rx_work=1 cpu_num=4
[ 28.936870] Attach wlan adapter operations.card_type is 0x105.
[ 28.943051] wlan: Enable TX SG mode
```

```
[ 28.946541] wlan: Enable RX SG mode
[ 28.954950] Request firmware: nxp/sdiouart8987_combo_v0.bin
[ 29.396714] Wlan: FW download over, firmwarelen=530240 downloaded
530240
[ 30.376759] WLAN FW is active
[ 30.379748] on_time is 30376924000
[ 30.404244] fw_cap_info=0x181c3f03, dev_cap_mask=0xffffffff
[ 30.409849] max_p2p_conn = 8, max_sta_conn = 8
[ 30.440578] wlan: version = SD8987----16.92.10. p210.1-MM5X16266.p4-
GPL- (FP92)
[ 30.452804] vendor=0x02DF device=0x9149 class=0 function=1
[ 30.458649] Attach moal handle ops, card interface type: 0x105
[ 30.464838] SD8987: init module param from usr cfg
[ 30.469968] Configuration block, fallback processing
[ 30.475020] Configuration fallback to, card_type: 0x105, blk_id: 0x0
[ 30.481492] SDIO: max_segs=128 max_seg_size=65535
[ 30.486316] rx_work=1 cpu_num=4
[ 30.489602] Attach mlan adapter operations.card_type is 0x105.
[ 30.495977] wlan: Enable TX SG mode
[ 30.499627] wlan: Enable RX SG mode
[ 30.519195] Request firmware: nxp/sdiouart8987 combo v0.bin
[ 30.951996] Wlan: FW download over, firmwarelen=530240 downloaded
530240
[ 31.928487] WLAN FW is active
[ 31.931477] on_time is 31928653125
[ 31.956280] fw_cap_info=0x181c3f03, dev_cap_mask=0xffffffff
[ 31.961885] max_p2p_conn = 8, max_sta_conn = 8
[ 31.988952] wlan: version = SD8987----16.92.10.p210.1-MM5X16266.p4-
GPL- (FP92)
[ 31.998716] wlan: Driver loaded successfully
```

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

- “wlan” string identifies driver log messages
- “card_type: SD8987” string identifies the chipset being 88W8987.
- “sdiouart8987_combo_v0.bin” indicates the WLAN/Bluetooth firmware file being loaded.
- “version” indicates specific version of firmware being loaded by the driver.

3. Now invoke “ifconfig wlan1 up” command to initialize the “wlan1” (WLAN) interface.

```
$ ifconfig wlan1 up
$ ifconfig wlan1
# "wlan1" interface is UP. WLAN MAC Address is shown.
wlan1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether d4:53:83:c1:b9:d6 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```



There are 2 wlan interfaces, “wlan0” and “wlan1” when you issue the command “ifconfig -a”. For Murata modules, “wlan1” interface must be used for Wi-Fi/BT bring-up. This is applicable only for i.MX 8M-Mini/8M-Nano.

```
$ ifconfig -a
wlan0: flags=4098<BROADCAST,MULTICAST> mtu 1500
```

```

ether ec:2e:98:f7:04:a9 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

# "mLAN1" interface is UP for Murata. WLAN MAC Address for Murata is shown
mLAN1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
ether d4:53:83:c1:b9:d6 txqueuelen 1000 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

7.1.3 STA/Client Mode: Scan for Visible Access Points

In this section, a simple method for scanning using the Linux “iw” command is presented.

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.



Note that the strength of received signals is important to do connectivity testing (i.e. “ping”, “iPerf”, etc.). Very attenuated signals will be in the high 80’s or 90’s (see “RSSI” value). If close to an Access Point, the returned “RSSI” value should be between -30 and -50 dBm for a properly configured setup.

“iw” is the default Linux command line tool for controlling a WLAN interface. One useful link to learn more about “iw” is on the [Linux Wireless wiki](#). In the following example of listing WLAN devices and performing a scan, one active AP has SSID of “Murata_5G”. Here are expected results:

```

# List available WLAN devices
$ iw dev
phy#0
    Interface p2p1
        ifindex 8
        wdev 0x100000003
        addr d6:53:83:c1:be:ec
        type managed
        txpower 24.00 dBm
    Interface uap1
        ifindex 7
        wdev 0x100000002
        addr d4:53:83:c1:bf:ec
        type AP
        txpower 24.00 dBm
    Interface mlan0
        ifindex 6
        wdev 0x100000001
        addr d4:53:83:c1:be:ec
        type managed
        txpower 24.00 dBm

# Perform scan on "mLAN0" interface
$ iw dev mLAN0 scan
wlan: mLAN0 START SCAN
wlan: SCAN COMPLETED: scanned AP count=13

BSS 84:1b:5e:f6:a7:60 (on mLAN0)

```

```

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-ffbf1-0ad1-63f3-9b91a979382c
    * 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

# ... More SSID listings follow here.

```

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



In the following test sequences of using “iw” command, the WLAN interface is not assigned an IP address. That is done later in [Section 7.1.6](#) where connectivity testing is performed.

To test:

- 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 60:38:e0:9a:a3:9e (on wlan0)
    SSID: Murata_Test_5G
    freq: 5180
    RX: 0 bytes (0 packets)
    TX: 2437 bytes (19 packets)
    signal: -36 dBm
    tx bitrate: 433.3 MBit/s VHT-MCS 9 80MHz short GI VHT-NSS 1

    bss flags:
    dtim period: 1

```

```
beacon int: 100
```

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

This section covers 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.

7.1.5.1 Using “wpa_cli” Command

“wpa_cli” can only be invoked once the “mlan0” 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
```

Here are the steps:

1. 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 mlan0 -D nl80211 -c /etc/wpa_supplicant.conf -B
Successfully initialized wpa_supplicant
rfkill: Cannot open RFKILL control device
rfkill: Cannot get wiphy information
```

2. Now invoke “wpa_cli” which brings up the tool in interactive mode:

```
$ wpa_cli -i mlan0
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
```

3. Following previous example, you can 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

```

4. 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.

7.1.5.2 Using “wpa_supplicant.conf” file


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

```

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

network={
    key_mgmt=NONE
}

```

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

```

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

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

```

To establish a secured WPA2-PSK connection by only modifying “/etc/wpa_supplicant.conf” file, follow these steps:

1. Modify “/etc/wpa_supplicant.conf” file to configure desired connection.
2. 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 is:

```
$ 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
```

3. Verify that connection is re-established with Access Point:

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

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

7.1.6 STA/Client Mode: Basic WLAN Connectivity Testing

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

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

Alternatively, you 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/udhcpc.d/50default: Adding DNS 192.168.1.1
```

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

```
$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
```



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

If you 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](#) .

7.1.7 Wi-Fi Direct Testing

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

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
```

1. Now you 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
```

2. 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.
3. To verify new virtual P2P interface, just invoke “ifconfig” command:

```
$ ifconfig
...
# New P2P interface
p2p-mlan0-0 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 "mlan0" interface
mlan0      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)
```

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

```
$ ifconfig p2p-mlan0-0 192.168.2.1 netmask 255.255.255.0
```

5. 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.

7.2 Bluetooth Interface Test/Verification

Before initializing the Bluetooth interface, the WLAN interface (mLAN0) must be first brought up. For the standard/default configuration of BT-UART interconnect (i.e., 1ZM or 1YM), you 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 “modem_reset” construct in DTS file, 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).

Here is the default command sequence to verify Bluetooth functionality:

```
hciattach /dev/ttymx[UART# -1] any 115200 flow
hcidtool -i hci0 cmd 0x3f 0x0009 0xc0 0xc6 0x2d 0x00
killall hciattach
hciattach /dev/ttymx[UART# -1] any -s 3000000 3000000 flow
hciconfig hci0 up
hciconfig hci0 piscan
hciconfig hci0 noencrypt
hcidtool scan
```

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 1ZM module:

```
$ hciattach /dev/ttymx1 any 115200 flow
Setting TTY to N_HCI line discipline
Device setup complete
$ hciconfig hci0 up
$ hcidtool -i hci0 cmd 0x3f 0x0009 0xc0 0xc6 0x2d 0x00
< HCI Command: ogf 0x3f, ocf 0x0009, plen 4
  C0 C6 2D 00
> HCI Event: 0x0e plen 4
  01 09 FC 00
$ killall hciattach
# Ignore the error in next line
[ 1669.277042] Bluetooth: hci0: sending frame failed (-49)
$ hciattach /dev/ttymx1 any -s 3000000 3000000 flow
Setting TTY to N_HCI line discipline
Device setup complete
$ hciconfig hci0 up
$ hciconfig hci0 piscan
$ hciconfig hci0 noencrypt
$ hciconfig -a
$ hcidtool 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	ttymxc Configuration	Notes
i.MX 8MQuad/8MPlus EVK	GPIO69; UART3	ttymxc2	
i.MX 8M Mini/Nano EVK (uSD)	GPIO140; UART3	ttymxc2	uSD-M.2 Adapter interconnect with M.2 EVB.
i.MX 6UL/ULL EVK	GPIO508; UART2	ttymxc1	GPIO508 does not allow its direction to be set. Always output.

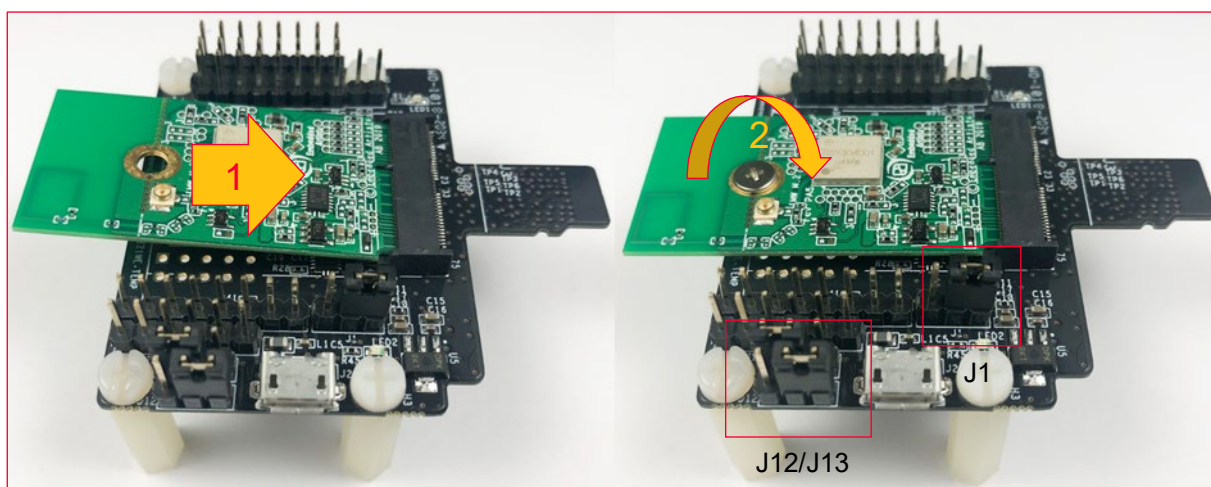
8 Murata's uSD-M.2 Adapter

This section describes the process of connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter and Configuring uSD-M.2 adapter jumpers for correct VIO signaling. (I added this, could these be internal links)

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 B2 (**Figure 29**), 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 29: Connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter

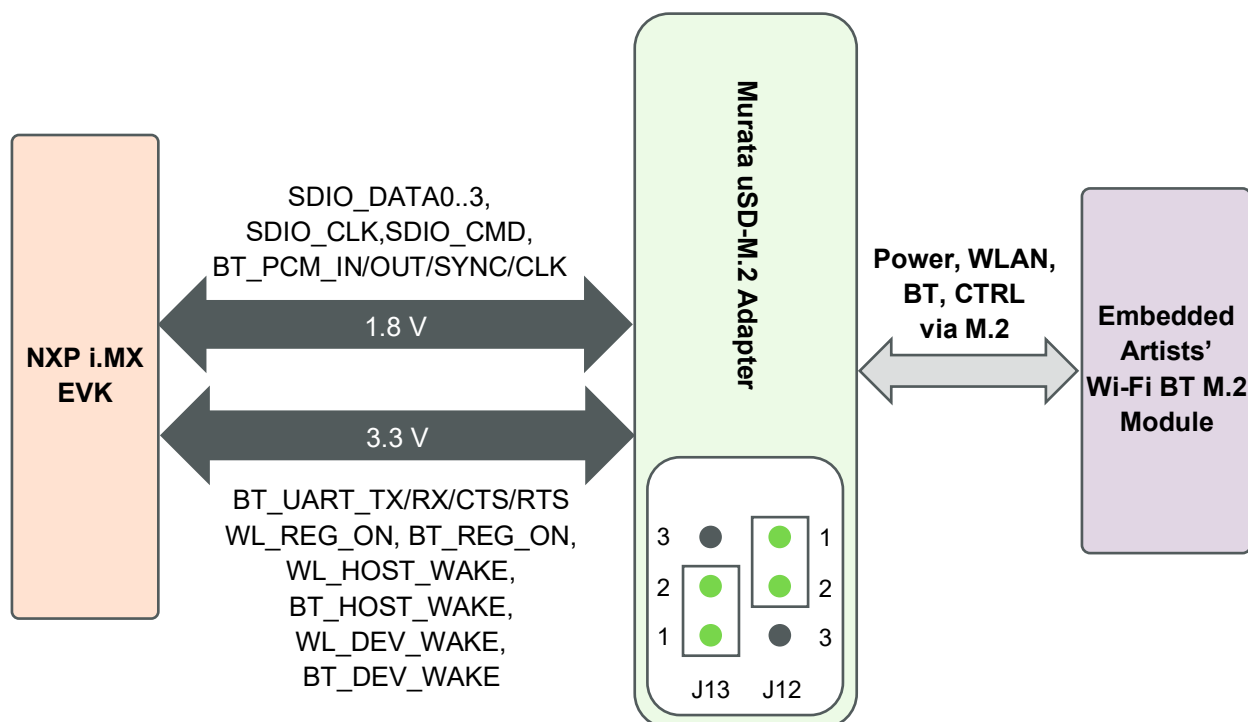


8.2 Configuring uSD-M.2 Adapter Jumpers for Correct VIO Signaling

Figure 30 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 30: Host/M.2 IO Voltage Level Shift Options on Rev B2 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 31**. 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 32**. 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 31: Securing uSD/SD Connection on i.MX 6 EVK

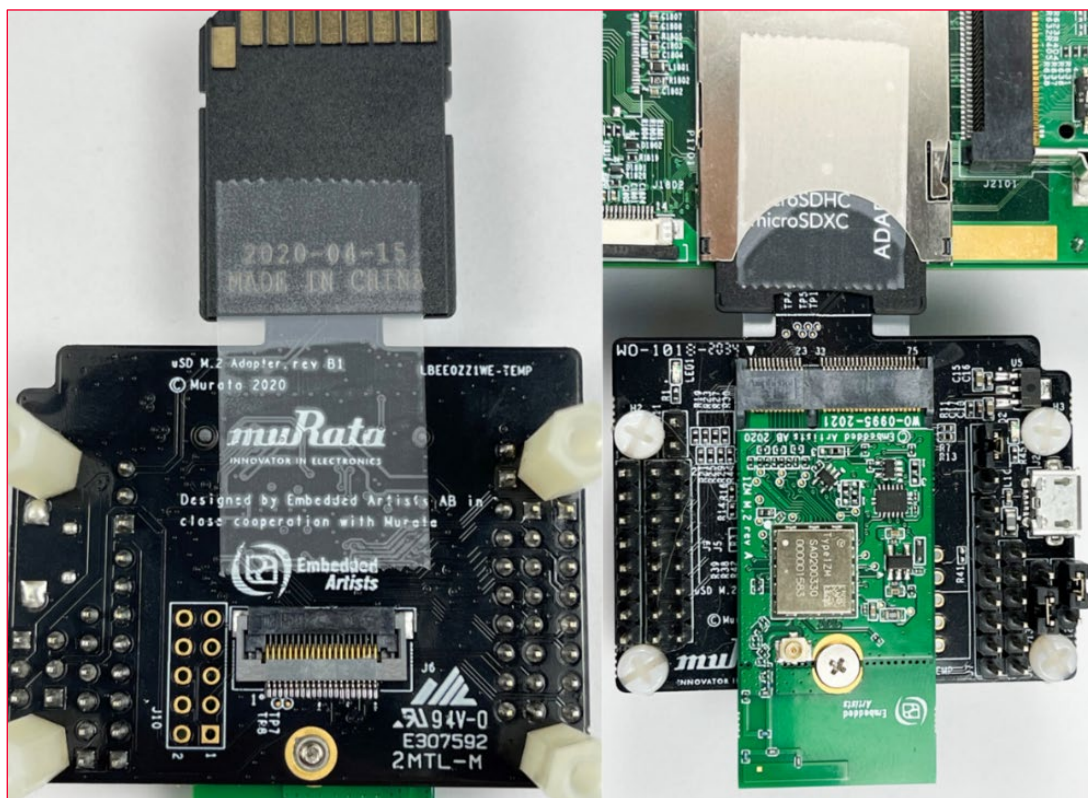
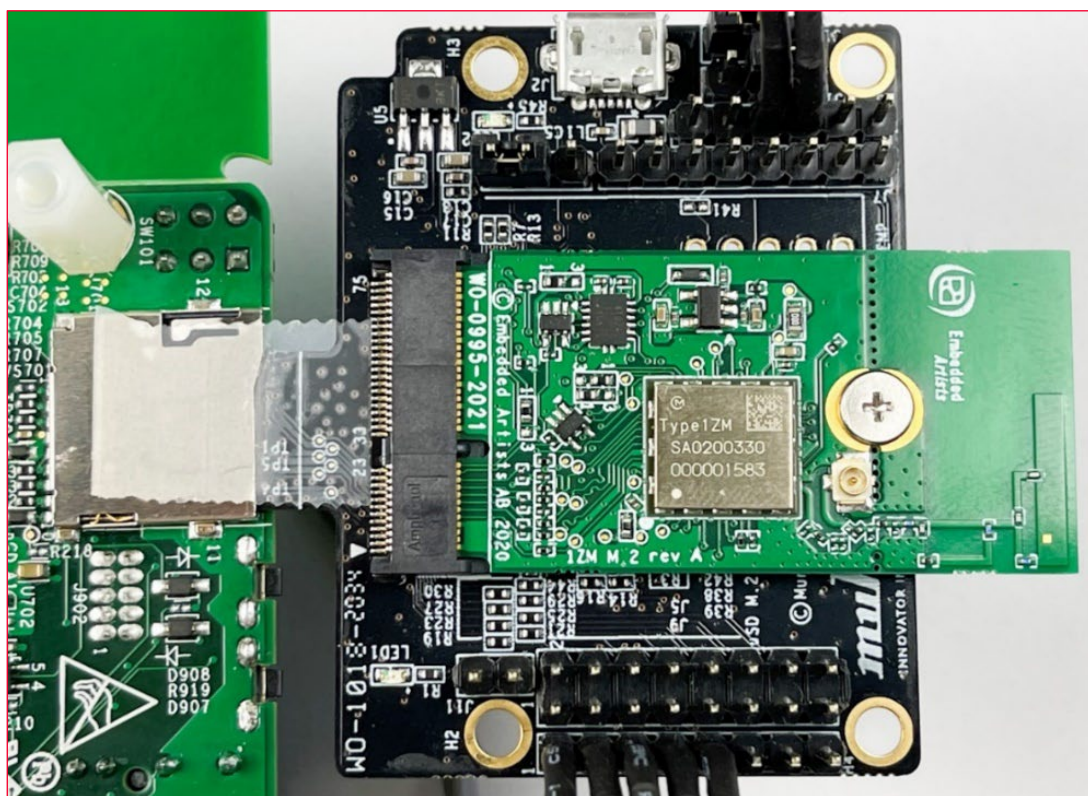


Figure 32: Securing uSD Connection on i.MX 8 EVK



8.4 uSD-M.2 Adapter High-Level Description

Figure 33 and **Figure 34** 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 that provide 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 33: uSD-M.2 Adapter Features (Top View)

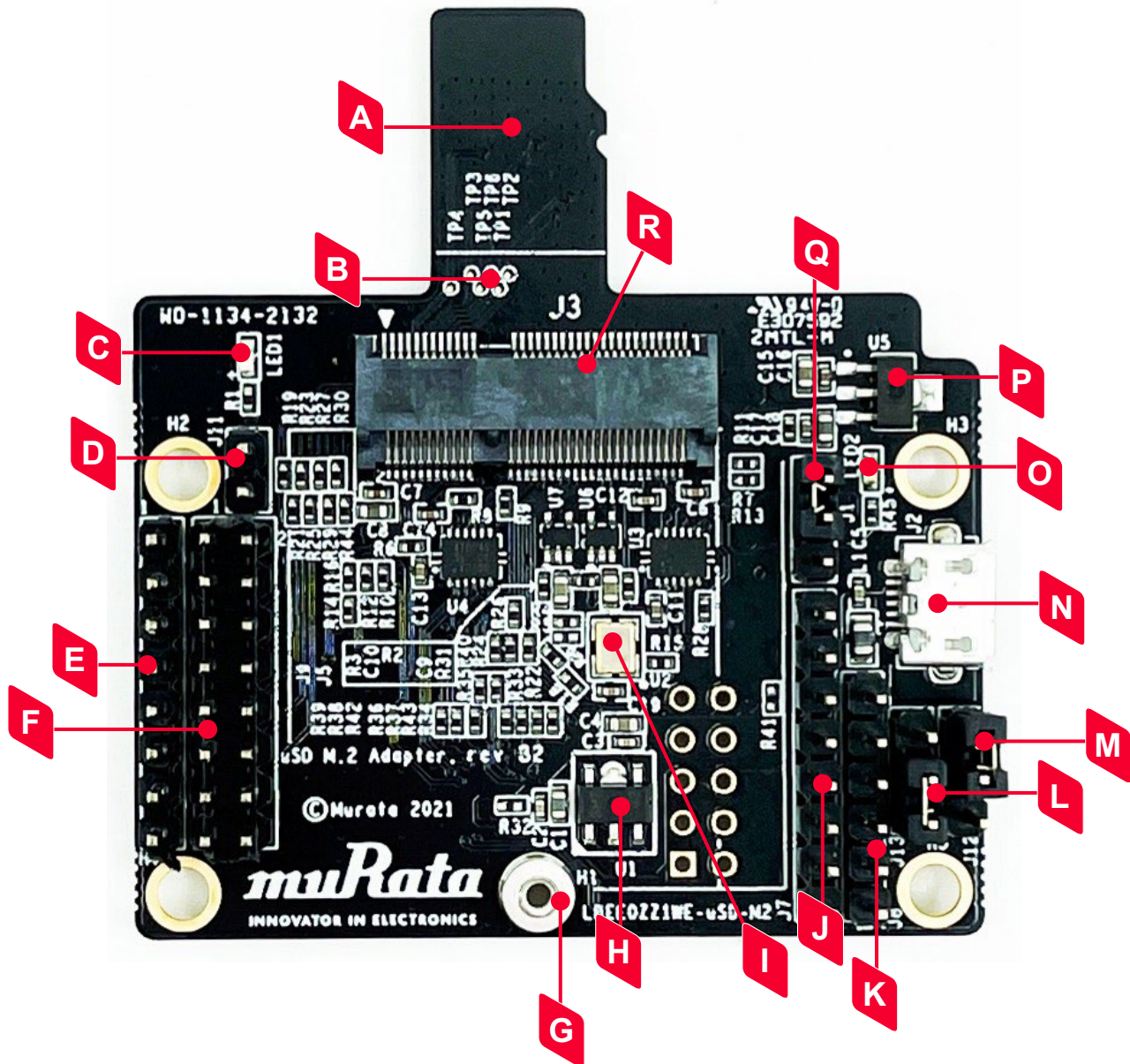
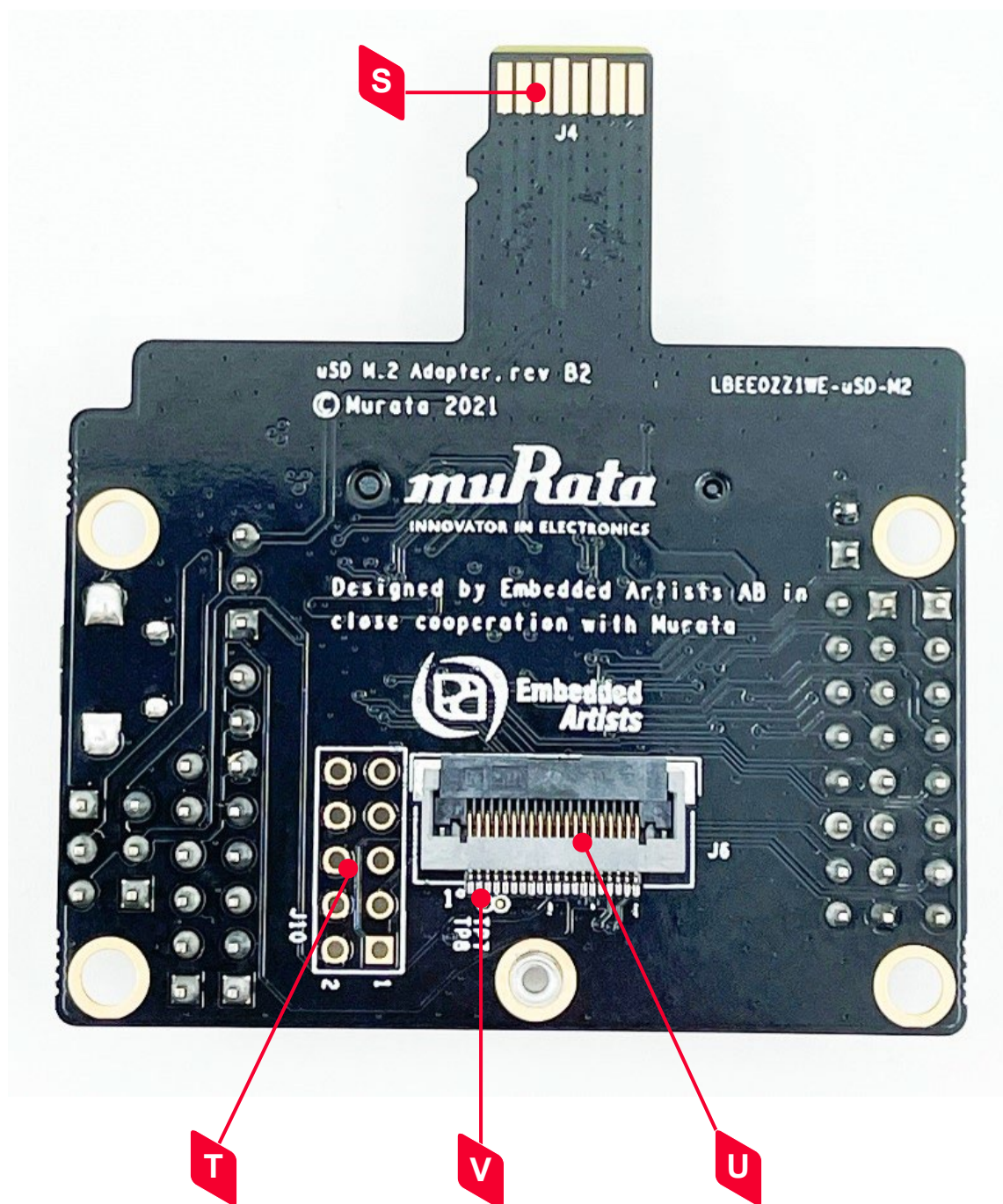



Figure 34: uSD-M.2 Adapter Features (Bottom View)



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 1ZM, 1YM, 1XK, 1XL and 2DS for NXP-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 35** shows that the carrier board can work with a variety of NXP i.MX6/7/8 (u)COM boards and five 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 35** 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 35: Combine i.MX COM with Wi-Fi/BT M.2 EVB

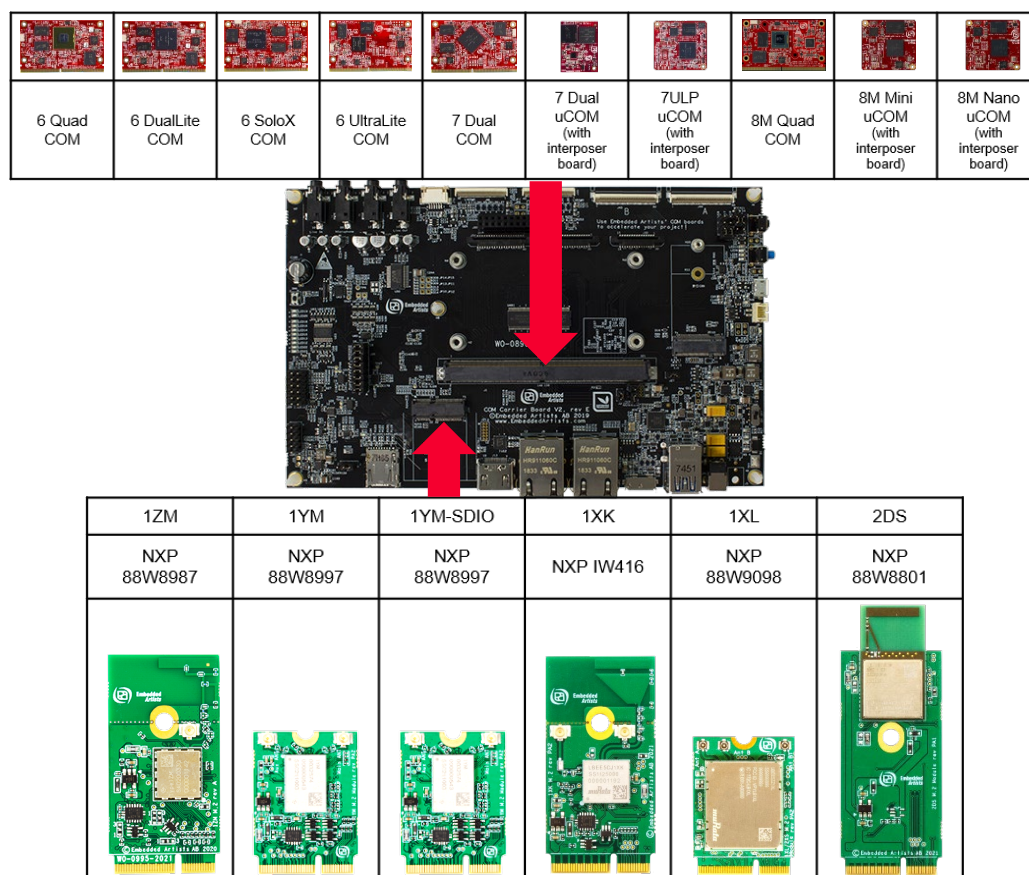


Table 14: Embedded Artists' i.MX Interconnect

EA i.MX (u)COM	1ZM	1YM	1XK	1XL	2DS
	NXP 88W8987	NXP 88W8997	NXP 88W8997	NXP 88W9098	NXP 88W8801
iMX8M Quad COM ↗	M.2 ^[16]	M.2	M.2	M.2	M.2
iMX8M Mini uCOM ↗	M.2	M.2	M.2	M.2	M.2
iMX8M Nano uCOM ↗	M.2	M.2	M.2	M.2	M.2
iMX7 Dual COM ↗	M.2	M.2	M.2	M.2	M.2
iMX7 Dual uCOM ↗	M.2	M.2	M.2	M.2	M.2
iMX7ULP uCOM ↗	M.2	M.2	M.2	M.2	M.2
iMX6 Quad COM ↗	NC ^[17]	M.2	M.2	M.2	M.2
iMX6 DualLite COM ↗	NC	M.2	M.2	M.2	M.2
iMX6 SoloX COM ↗	NC	M.2	M.2	M.2	M.2
iMX6 UltraLite COM ↗	M.2	M.2	M.2	M.2	M.2

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 1ZM and 1YM 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.
EACOM Board Specification Guide ↗	Comprehensive definition of Embedded Artists' Computer-On-Module's.
1ZM M.2 Module Datasheet ↗	Comprehensive details on 1ZM Wi-Fi/BT M.2 Module.
1YM M.2 Module Datasheet ↗	Comprehensive details on 1YM Wi-Fi/BT M.2 Module.
1XK M.2 Module Datasheet ↗	Comprehensive details on 1XK Wi-Fi/BT M.2 Module.
1XL M.2 Module Datasheet ↗	Comprehensive details on 1XL Wi-Fi/BT M.2 Module.
2DS M.2 Module Datasheet ↗	Comprehensive details on 2DS Wi-Fi M.2 Module.

¹⁶ Works with onboard M.2 slot

¹⁷ No Connection option (due to 1ZM only supporting 1.8V SDIO VIO)

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 of 5.10.35 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 Murata's Regulatory Solution for NXP Modules

This section describes Murata's regulatory solution for NXP Modules.

11.1 Description of Regulatory Solution for WLAN/BT

NXP IC based module shall use CRDA mechanism which is provided by Linux-wireless. Compile the new regulatory.bin file from "db.txt" which is provided by Murata with following manner of [wireless-regdb](#).

WLAN: Murata's patched solution automatically generates "regulatory.bin" for various modules (1XK/1ZM/1YM/2DS) and for various countries (US/EU/CA/JP). Setting appropriate Tx power regulatory binary files for various countries using "iw" command for the specified module is taken care by the script file, "switch_regions.sh".

BT: Bluetooth Tx power is configured by using hcitool / "bt_power_config_1.sh" script file.

11.2 Tree view of the files list for Linux Patched Solution

NXP_Linux_Patched_Solution_for_5.10.***/

```

|
|—— murata
|   |—— files
|       |—— 1XK
|           |—— db.txt
|           |—— ed_mac.bin
|           |—— regulatory.bin
|           |—— txpower_CA.bin
|           |—— txpower_EU.bin
|           |—— txpower_JP.bin
|           |—— txpower_US.bin
|       |—— 1YM
|           |—— db.txt
|           |—— ed_mac.bin
|           |—— regulatory.bin
|           |—— txpower_CA.bin
|           |—— txpower_EU.bin
|           |—— txpower_JP.bin
|           |—— txpower_US.bin
|       |—— 1ZM
|           |—— db.txt
|           |—— ed_mac.bin
|           |—— regulatory.bin
|           |—— txpower_CA.bin
|           |—— txpower_EU.bin
|           |—— txpower_JP.bin
|           |—— txpower_US.bin
|       |—— 2DS
|           |—— db.txt
|           |—— ed_mac.bin
  
```

```

| | | | regulatory.bin
| | | | txpower_CA.bin
| | | | txpower_EU.bin
| | | | txpower_JP.bin
| | | | txpower_US.bin
| | | | 32_bit
| | | | | crda
| | | | | libreg.so
| | | | | regdbdump
| | | | 64_bit
| | | | | crda
| | | | | libreg.so
| | | | | regdbdump
| | | |
| | | | bt_power_config_1.sh
| | | | regulatory.rules
| | | | wifi_mod_para_murata.conf
| | |
| | | README.txt
| | | switch_regions.sh
| |
| patch_files
| | imx6ul-14x14-evk-btwifi.dtb
| | imx6ul-14x14-evk-btwifi.dts
| | imx6ull-14x14-evk-btwifi.dtb
| | imx6ull-14x14-evk-btwifi.dts

```

11.2.1 Description of files

Table 18: Murata regulatory files

File Name	Description
db.txt	WLAN regulatory limitation configuration file.
ed_mac.bin	WLAN Carrier Sense / Adaptivity threshold configuration file.
regulatory.bin	File used by the Linux wireless subsystem to keep its regulatory database information. It is read by CRDA upon the Linux Kernel's request for regulatory information for a specific country code.
bt_power_config_1.sh	Bluetooth Tx power configuration file.
regulatory.rules	udev rule put in place to trigger CRDA to send the respective regulatory domain.
wifi_mod_para_murata.conf	Default configuration file provided by NXP and used by Murata to modify the necessary parameters for various modules (1XK, 1ZM, 1YM, 2DS).
txpower_CA.bin	WLAN Tx power configuration files for Canada (IC).
txpower_EU.bin	WLAN Tx power configuration files for EU (CE).
txpower_JP.bin	WLAN Tx power configuration files for Japan (TELEC).
txpower_US.bin	WLAN Tx power configuration files for US (FCC).
CRDA	Central Regulatory Data Base
libreg.so	Library file
regdbdump	Used to parse the regulatory.bin file
README.txt	Murata regulatory solution readme
switch_regions.sh	Script to switch regulatory region
imx6ul-14x14-evk-btwifi.dtb	Modified DTB file for i.MX 6UL EVK
imx6ul-14x14-evk-btwifi.dts	Modified DTS file for i.MX 6UL EVK
imx6ull-14x14-evk-btwifi.dtb	Modified DTB file for i.MX 6ULL EVK
imx6ull-14x14-evk-btwifi.dts	Modified DTS file for i.MX 6ULL EVK

11.3 Purpose of switch_regions.sh

The switch_regions.sh script file allows the user to easily switch the regulatory region. To use the script, copy the folder, "NXP_Linux_Patched_Solution_for_5.10.***/murata" from the [Murata patched solution](#) to "/lib/firmware/nxp" in the root file system. This is only required for the first time.

Execute the script file, "switch_regions.sh" for deploying regulatory solution.

Ex: switch_regions.sh <Module_name> <Country_code>

The following options are supported.

- Module name –
 - 1ZM – For Embedded Artists' 1ZM M.2 EVB
 - 1XK – For Embedded Artists' 1XK M.2 EVB
 - 1YM – For Embedded Artists' 1YM M.2 EVB (Both SDIO and PCIe)
 - 2DS – For Embedded Artists' 2DS M.2 EVB
- Country code
 - US - USA
 - EU - Europe
 - JP - Japan
 - CA – Canada

11.3.1 Sample Description of Structure SD8987 from “wifi_mod_para.conf” configured for US with 1ZM module

```
SD8987 = {
    cfg80211_wext=0xf
    wfd_name=p2p
    max_vir_bss=1
    cal_data_cfg=none
    drv_mode=7
    ps_mode=2
    auto_ds=2
    fw_name=nxp/sdiouart8987_combo_v0.bin
    txpwrlimit_cfg=nxp/txpower_US.bin
}
```

11.3.2 Sample Description of Structure PCIE8997 from “wifi_mod_para.conf” Configured for EU with 1YM Module

```
PCIE8997 = {
    cfg80211_wext=0xf
    wfd_name=p2p
    max_vir_bss=1
    cal_data_cfg=none
    drv_mode=7
    ps_mode=2
    auto_ds=2
    fw_name=nxp/pcieuart8997_combo_v4.bin
    txpwrlimit_cfg=nxp/txpower_EU.bin
    init_hostcmd_cfg=nxp/ed_mac.bin
}
```



ed_mac.bin is only applicable for EU country code.

11.4 Sample log of switch_regions.sh: (Ex: 1YM-SDIO@8M-MINI)

For country code CA (Canada):

```
root@imx8mmevk:/lib/firmware/nxp/murata# ./switch_regions.sh 1YM CA

Setting up for 1YM (64 bit):
-----
Setup complete.

global
country CA: DFS-FCC
    (2402 - 2472 @ 40), (N/A, 30), (N/A)
    (5150 - 5250 @ 80), (N/A, 23), (N/A), NO-OUTDOOR, AUTO-BW
    (5250 - 5350 @ 80), (N/A, 24), (0 ms), DFS, AUTO-BW
    (5470 - 5600 @ 80), (N/A, 24), (0 ms), DFS
    (5650 - 5730 @ 80), (N/A, 24), (0 ms), DFS
    (5735 - 5835 @ 80), (N/A, 30), (N/A)
```

For country code US (United States):

```
root@imx8mmevk:/lib/firmware/nxp/murata# ./switch_regions.sh 1YM US

Setting up for 1YM (64 bit):
-----
Setup complete.

global
country US: DFS-FCC
    (2400 - 2472 @ 40), (N/A, 30), (N/A)
    (5150 - 5250 @ 80), (N/A, 23), (N/A), AUTO-BW
    (5250 - 5350 @ 80), (N/A, 23), (0 ms), DFS, AUTO-BW
    (5470 - 5730 @ 160), (N/A, 23), (0 ms), DFS
    (5730 - 5850 @ 80), (N/A, 30), (N/A)
    (57240 - 71000 @ 2160), (N/A, 40), (N/A)
```

12 Useful Links

Table 19 provides some useful links.

Table 19: 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.

13 Appendix A: Building Image Output

The following shows the output of building an image for [NXP i.MX 8MMini EVK](#), running [5.10.52](#) kernel release, using the Murata build script.

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

```
./Murata_Wireless_Yocto_Build_NXP.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 5.10.52.

```
Select which entry? 0
Selected : 5.10.52
```

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

```
Select your entry: 4
Selected target: imx8mmevk
```

6. 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.
```

7. Enter build_8mm as build directory name.

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

8. Verify selection and start the build.

```
i.MX Yocto Release           : 5.10.52_2.1.0
Yocto branch                 : hardknott
Target                       : imx8mmevk
NXP i.MX EVK Part Number    : 8MMINILPD4-EVK
meta-murata-wireless Release Tag: imx-hardknott-5-10-52_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
```

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

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

10. 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.
```

11. 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
```

12. 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.

14 Appendix B: Acronyms





Acronym	Meaning
1YM	AKA "1YM-PCIe", indicates that M.2 EVB is strapped for WLAN-PCIe/BT-UART - refer to Section 3.4.5 
1YM-SDIO	Type 1YM M.2 EVB strapped for WLAN-SDIO/BT-UART - refer to Section 3.4.5 
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
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
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
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
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

15 Technical Support Contact

Table 20 lists all the support resources available for the Murata Wi-Fi/BT solution.


Table 20: 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.


16 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.


16.1 Wi-Fi/Bluetooth for i.MX Linux Quick Start Guide for NXP-based Module

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


16.2 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.


16.3 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.


16.4 Murata's FCC Regulatory Test Guide

[This document](#)  provides all the steps necessary to run the FCC regulatory certification test for Murata Wi-Fi/Bluetooth modules based on NXP chipsets using Murata's regulatory test tool.


16.5 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.

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

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

16.7 Embedded Artists' Reference Documentation

Embedded Artists designed the 1ZM/1YM/1XK/2DS 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 21 lists some relevant documents published by Embedded Artists.

Table 21: 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.
1ZM M.2 Module Datasheet	Comprehensive details on 1ZM Wi-Fi/BT M.2 Module.
1YM M.2 Module Datasheet	Comprehensive details on 1YM Wi-Fi/BT M.2 Module.
1XK M.2 Module Datasheet	Comprehensive details on 1XK Wi-Fi/BT M.2 Module.
1XL M.2 Module Datasheet	Comprehensive details on 1XL Wi-Fi/BT M.2 Module.
2DS M.2 Module Datasheet	Comprehensive details on 2DS Wi-Fi M.2 Module.

16.8 Murata Linux Patched Solution

This [archive file](#) contains the files required for correctly configuring the regulatory region settings, as well as script files to easily deploy the changes.

16.9 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.

16.10 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 22 provides the following information on all releases supported:

Table 22: NXP Reference Documentation Listing

Kernel release	NXP documentation link	Yocto name	Release information
5.10.52_2.1.0	Rev. L5.10.52_2.1.0_BSP	Hardknott	imx-hardknott-5-10-52_r1.0
5.10.72_2.2.0	Rev. L5.10.72_2.2.0_BSP	Hardknott	imx-hardknott-5-10-72_r1.0

Revision History

Revision	Date	Author	Change Description
1.0	Nov 17, 2020	TF	Initial Release.
1.1	Jan 28, 2021	TF	Removed unsupported hardware and software configurations.
1.2	Jan 13, 2022	TF	Added support for Linux 5.10.52, 5.10.72 and build script. Removed 5.4.47.
2.0	May 27, 2022	TF	Migrated to new format. Added support for 2DS.



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.