User Guide

muRata
INNOVATOR IN ELECTRONICS

Wi-Fi™/Bluetooth® (NXP) for i.MX

Linux User Guide - Rev. 4.17

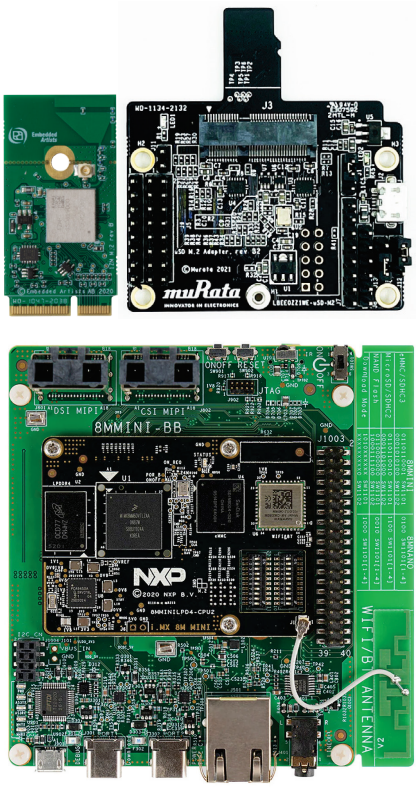


Table of Contents

Figures

Tables

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 6/8/9 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**: Murata  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.
`Console input/output or code snippet`	**Console I/O or Code Snippet** This text ***Style*** denotes console input/output or a code snippet.
`# Console I/O comment` `// Code snippet comment`	**Console I/O or Code Snippet Comment** This text ***Style*** denotes a console input/output or code snippet comment. • 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.

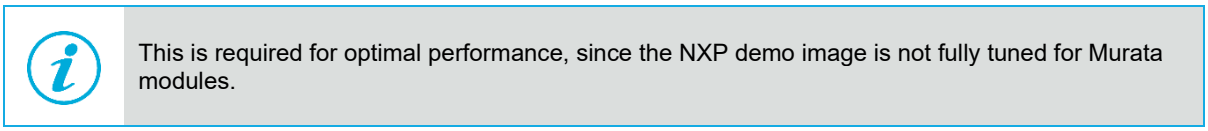
1 Introduction

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:

- 5.15.32_2.0.0
- 6.1.1_1.0.0
- 6.1.36_2.1.0
- 6.6.3_1.0.0
- 6.6.23_2.0.0

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 regulatory 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 2DS, 1XK, 2XK, 1ZM, 2DL and 2EL M.2 EVBs support WLAN-SDIO interface only. Type 1YM, 1XL and 2XS M.2 EVBs serve "double duty" in supporting both WLAN-PCIe (default strapping configuration) and WLAN-SDIO options.

Table 2: Current NXP i.MX Platforms Supported

NXP i.MX EVK Part number	NXP i.MX EVK	Murata Modules Supported	Interconnect
MCIMX93-EVK	i.MX 93 EVK	2DS, 1XK, 2XK, 1ZM, 2DL, 2EL, 1YM, 1XL, 2XS	M.2 (WLAN-SDIO)
MCIMX8QXP-CPU	i.MX 8QXP MEK	1YM, 1XL, 2XS	M.2 (WLAN-PCIe)
8MPLUSLPD4-EVK	i.MX 8MPLUS EVK	2DS, 1XK, 2XK, 1ZM, 2DL, 2EL, 1YM, 1XL, 2XS,	M.2: 2DS, 1XK, 2XK, 1ZM, 2DL, 2EL, 1YM (PCIe and SDIO), 1XL (PCIe and SDIO), 2XS (PCIe and SDIO)

NXP i.MX EVK Part number	NXP i.MX EVK	Murata Modules Supported	Interconnect
MCIMX8M-EVKB	i.MX 8MQuad EVK	2DS, 1XK, 2XK, 1ZM, 2DL, 2EL, 1YM, 1XL, 2XS	uSD-M.2 Adapter: 2DS (WLAN Only), 1XK, 2XK, 1ZM, 1YM, 2DL, 2EL, 1XL, 2XS M.2 (WLAN-PCIe): 1YM, 1XL, 2XS
8MMINILPD4-EVKB	i.MX 8M Mini EVK	2DS, 1XK, 2XK, 1ZM, 2DL, 2EL, 1YM, 1XL, 2XS	uSD-M.2 Adapter: 2DS, 1XK, 2XK, 1ZM, 2DL, 2EL, 1YM, 1XL, 2XS M.2 (WLAN-PCIe): 1YM – WLAN Only, 1XL – WLAN Only, 2XS – WLAN Only
8MNANOD4-EVK	i.MX 8M Nano EVK	2DS, 1XK, 2XK, 1ZM, 2DL, 2EL, 1YM, 1XL, 2XS	uSD-M.2 Adapter
MCIMX8ULP-EVK	i.MX 8ULP EVK	2DS, 1XK, 2XK, 1ZM, 2DL, 2EL, 1YM, 1XL, 2XS	M.2 (WLAN-SDIO)
MCIMX6UL-EVKB	i.MX 6UL EVK	2DS, 1XK, 2XK, 1ZM, 2DL, 2EL, 1YM, 1XL, 2XS	uSD-M.2 Adapter
MCIMX6ULL-EVK	i.MX 6ULL EVK	2DS, 1XK, 2XK, 1ZM, 2DL, 2EL, 1YM, 1XL, 2XS	uSD-M.2 Adapter

2DS Supports only WLAN. There is no Bluetooth.

For 2XK M.2 EVB, contact Murata via 2XK Community Forum page.

1.1 NXP i.MX 6 Platform Support

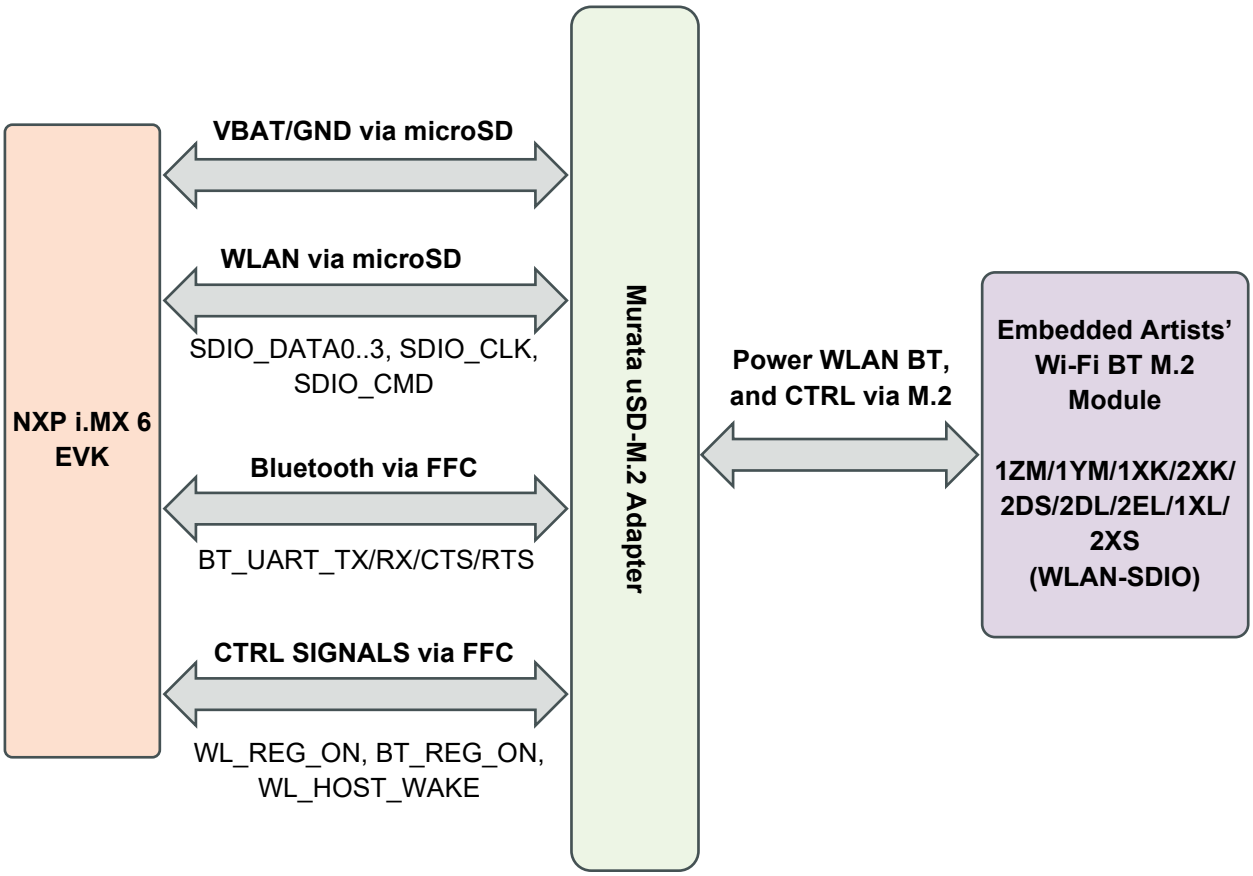
A high-level connection Diagram for the Murata uSD-M.2 Adapter (i.MX 6 platforms) is provided in **Figure 1**. All the Murata Wi-Fi/BT modules enabled by this release are shown. The wireless solution is arrived at by combining Murata's uSD-M.2 Adapter with Embedded Artists' Wi-Fi/BT M.2 EVB. Refer to Section 9 and Section 10 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, 1XL and 2XS M.2 modules (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 1XK, 2XK, 1ZM, 2DL and 2EL support 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".
- Type 1YM, 1XL and 2XS will support WLAN-PCIe/BT-UART and WLAN-SDIO/BT-UART only. Drivers for both the configurations are included in 5.15.32/6.1.1/ 6.1.36 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, 1XL and 2XS M.2 EVBs are 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

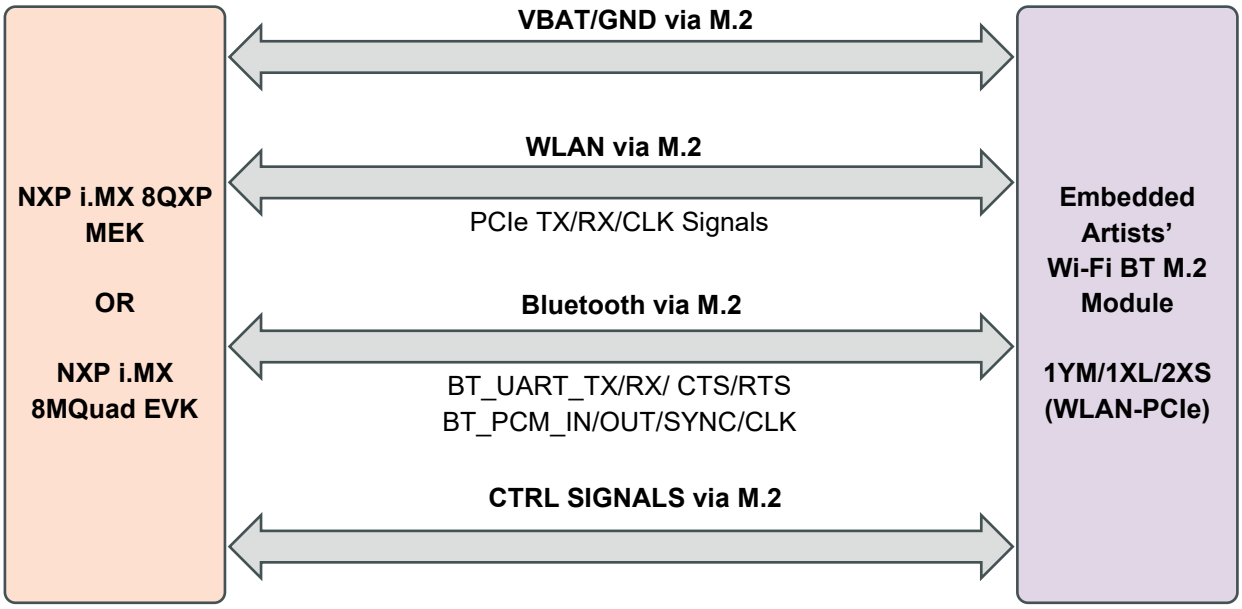


Figure 3 shows a simplified block diagram for the i.MX 8MQuad EVK Wi-Fi/BT SDIO interconnect. Type 2DS, 1XK, 2XK,1ZM, 2DL, 2EL, 1YM, 1XL and 2XS modules are supported via the uSD-M.2 adapter on i.MX 8MQuad EVK. Note that this EVK can use onboard eMMC flash. The eMMC flash is necessary as the default 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

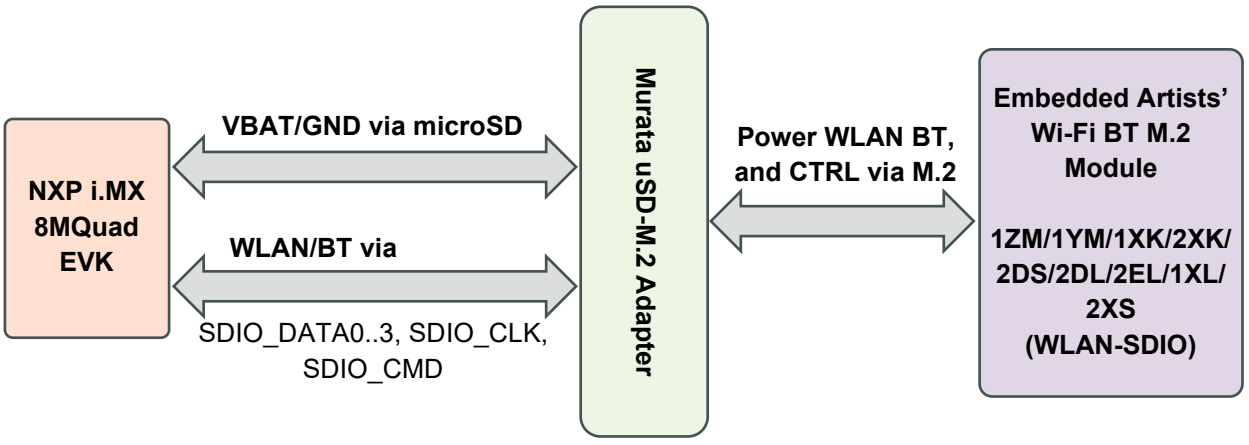


Figure 4 shows a simplified block diagram for the i.MX 8M Plus EVK Wi-Fi/BT interconnect (using M.2 connector). Currently Type 2DS, 1XK, 2XK, 1ZM, 2DL, 2EL (SDIO) and 1YM, 1XL, 2XS modules (PCIe and SDIO) are supported via the M.2 interface. Both WLAN-PCIe and WLAN-SDIO is supported via this interface.

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

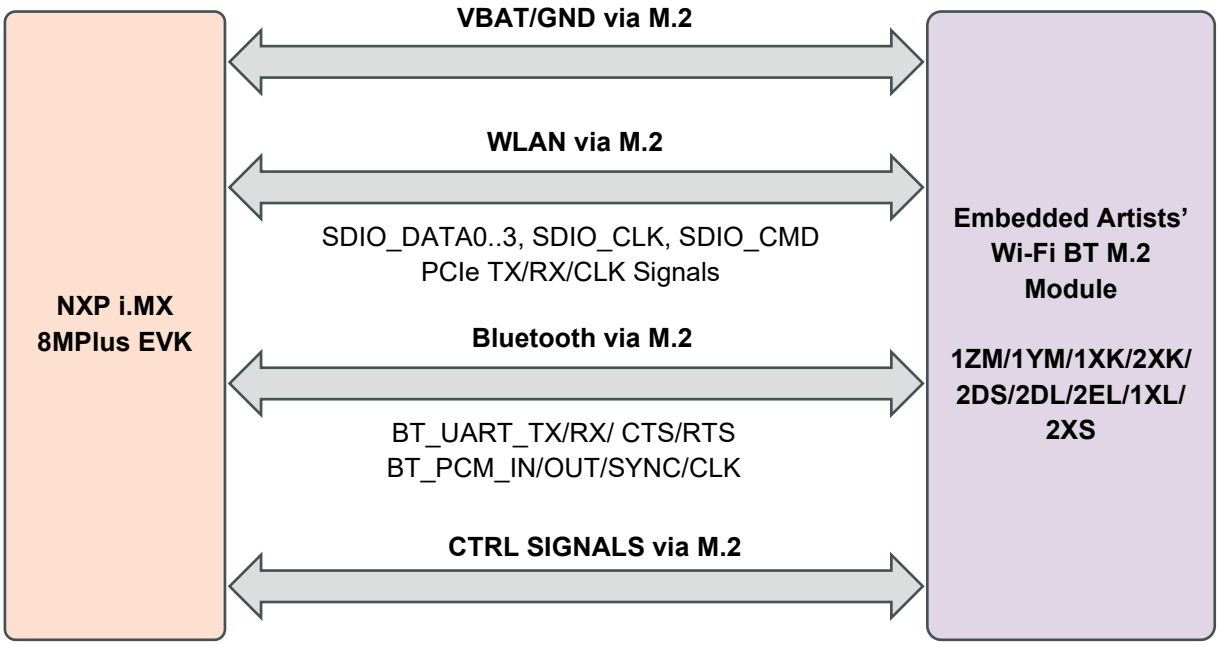
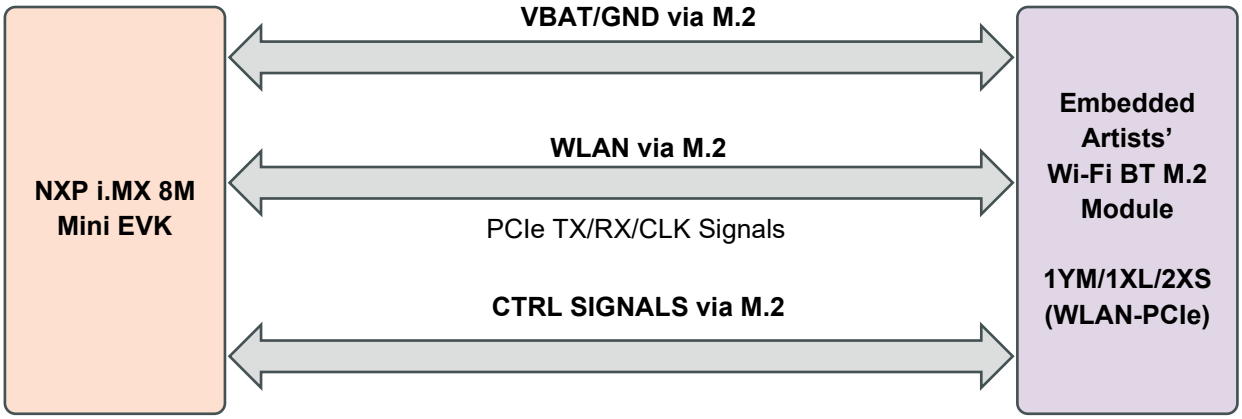


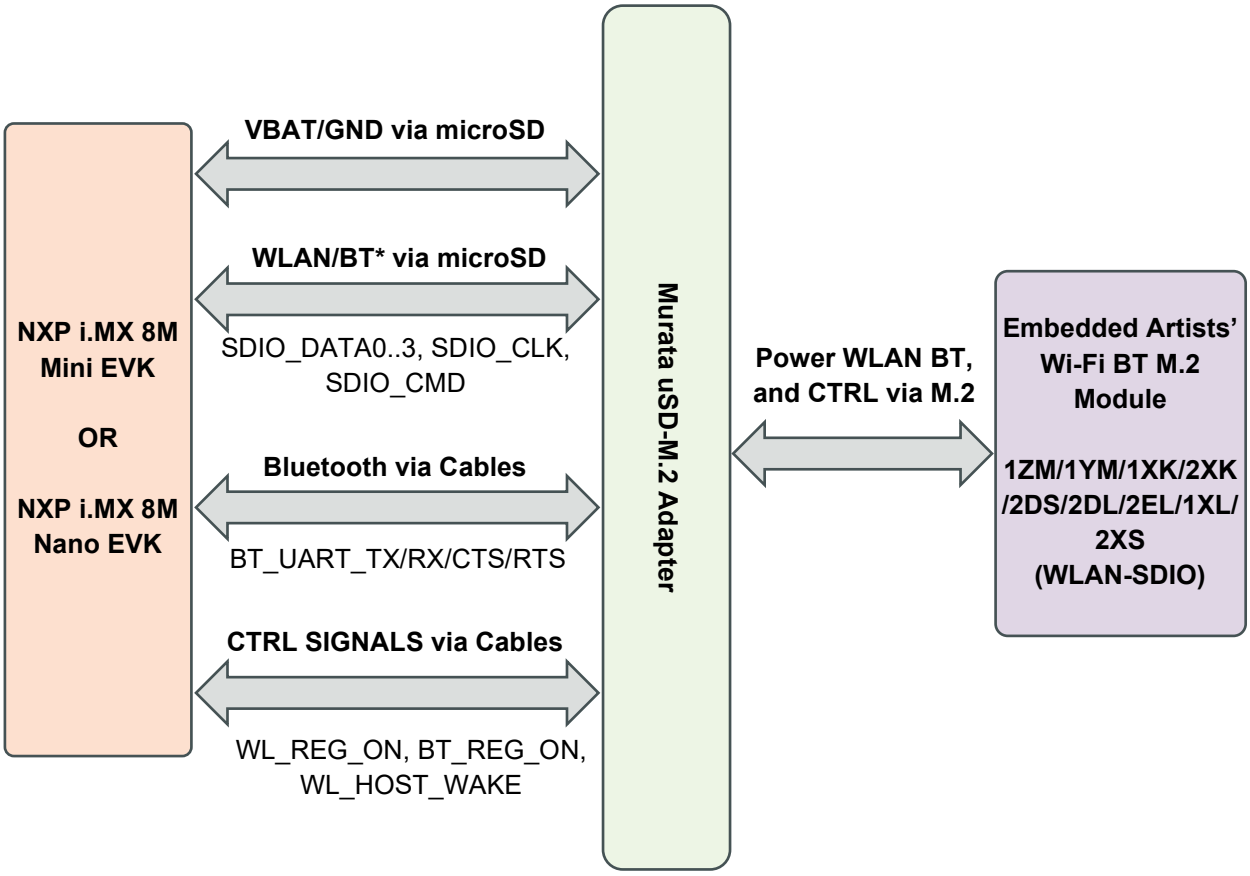
Figure 5 shows a simplified block diagram for the i.MX 8M Mini EVK Wi-Fi/BT interconnect (using M.2 connector). Currently Type 1YM, 1XL, 2XS module (WLAN Only – Bluetooth signals are not brought out) are supported via the WLAN-PCIe interface.

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



For WLAN-SDIO based module see **Figure 6**. Type 2DS, 1XK, 2XK, 1ZM, 2DL, 2EL, 1YM, 1XL and 2XS 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 Murata Community Forum

Murata has introduced the Murata Community Forum for technical support to our customers. The Forum can be accessed via this link.

The forum content is freely accessible to the public. However, users must log in to post questions or answers. Registration is free.

2.1 Registering

To participate in the forum, you must first register. Follow these steps to complete the registration process:

1. From the forum homepage, click on the **Login** button to navigate to the Login page.
2. Select the **Sign up** option to begin the registration process.
3. Scroll to the end of the Terms of Use, tick the checkboxes, and click on the **I Agree** button.
4. Provide your email and password, ensuring you use your company email address, then click on **Next**.
5. Click the **Send me an email** button to initiate email verification.
6. Verify your email by either clicking the verification link sent to your email or entering the confirmation code in the **Enter confirmation code** link and clicking **Verify**.
7. Fill in the required information, confirm the details, and click **Submit**.
8. Wait for the forum moderators to review and approve your application. You will receive an email notification once approved.

2.2 Logging In

Once registered, you can log in to the forum as follows:

1. From the forum homepage, click on the **Login** button to navigate to the Login page.
2. Enter your Username (the email address you used for registration) and your Password, then click **Verify**.
3. Agree to the Terms of Use by clicking on the **I Agree** button. (This step is required only during your first login)
4. After successful login, you will gain full access to the forum's features.

2.3 Using the Public Forum

To create a new post or ask a question, follow these steps:

5. Navigate to the appropriate category or subcategory that best matches the scope of your post using the forum menu or navigation options.
6. Click on the **Ask a Question** button to open the post composition page.
7. Provide the following details for your post:
8. Enter a clear and descriptive title in the **Question** field.
9. Add detailed information in the **Details** field.
10. Attach relevant files, if needed.

11. Add relevant tags (topics) for proper post classification and enhanced searchability. Suggestions will appear as you type.
12. Click the **Ask** button to publish your question. The button will only activate once you have filled in the **Question** field.

2.4 Using Private Group and Its Importance

Private forum groups play a vital role in ensuring confidentiality and secure collaboration for the following purposes:

- **Sharing sensitive information:** Enables restricted sharing of critical details within a controlled environment.
- **Schematics review:** Facilitates technical reviews of design schematics.
- **Logs and file sharing:** Supports sharing of files such as manufacturing firmware required for regulatory testing.

Private forum groups are designed for secure and restricted discussions. To access a private forum group:

13. Click on your Username on the top right to open the User menu, then click **My Profile** to go to your Profile page.
14. Locate the Groups block on the right and click on the name of the group you wish to access. If the group isn't visible, click **View All** to see the full list of groups you are a member of. Select the desired group. If you still cannot see your group, you are likely not a member yet. Please request access in any public post you have created, and a moderator will add you to the group (and create the private group if it does not exist).

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

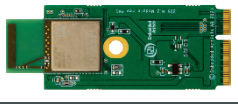
As already outlined in the Introduction, the Murata Wi-Fi/BT solution is primarily arrived at using Embedded Artists' Wi-Fi/BT M.2 EVBs in conjunction with Murata's uSD-M.2 Adapter. This section provides additional details on the hardware solution.

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

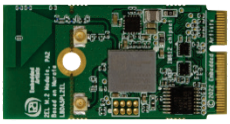
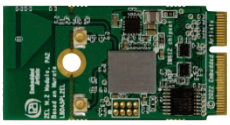
Embedded Artists designs, manufactures and distributes the Wi-Fi/BT M.2 EVBs based on Murata modules. These new M.2 EVBs are now Murata's official evaluation board for these modules in the Distribution Channel. Embedded Artists has excellent documentation support with a main landing page.

Table 3 shows the details of Embedded Artists' Wi-Fi/BT M.2 Modules. Note that Type 2DS (88W8801), Type 1XK/2XK (NXP IW416), Type 1ZM (NXP 88W8987), Type 2DL (NXP IW611) and Type 2EL (NXP IW612) support only WLAN-SDIO interface, whereas Type 1YM (NXP 88W8997), Type 1XL/2XS (NXP 88W9098) support both WLAN-PCIe and WLAN-SDIO interfaces. Also note that both Type 1ZM's and Type 1YM's WLAN-SDIO VIO interface only supports 1.8V[1]. To learn specifics on any of the M.2 EVBs, just click on Embedded Artists' M.2 Module Part Number (hyperlink included in table). To learn more specifics on the Murata module, just click Murata part number and you will be redirected to Murata's module landing page.

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

Murata Module	Chipset	Wi-Fi/BT support	Murata Part Number	EA M.2 Part Number	VIO	Interface	EVB Picture
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/2XK	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/2XS	NXP 88W9098	802.11a/b/g/n/ac/ax (2x2 MU-MIMO) BT/BLE 5.2	LBEE5ZZ1XL	EAR00387	1.8V/3.3V (WLAN SDIO, PCIe)	WLAN-PCIe WLAN-SDIO BT-UART	
2DS	NXP 88W8801	802.11b/g/n	LBWA0ZZ2DS	EAR00386	1.8V (WLAN SDIO)	WLAN-SDIO	

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

					3.3V (WLAN-SDIO)		
2DL	NXP IW611	802.11a/b/g/n/ac/ax BT/BLE 5.3	LBEE5PL2DL	EAR00422	1.8V (WLAN SDIO) 3.3V (WLAN-SDIO)	WLAN-SDIO	
2EL	NXP IW612	802.11a/b/g/n/ac/ax BT/BLE 5.3 IEEE 802.15.4	LBES5PL2EL	EAR00409	1.8V (WLAN SDIO) 3.3V (WLAN-SDIO)	WLAN-SDIO	

3.2 Murata's uSD-M.2 Adapter

Table 4: uSD-M.2 Adapter Kit (Rev B) Contents

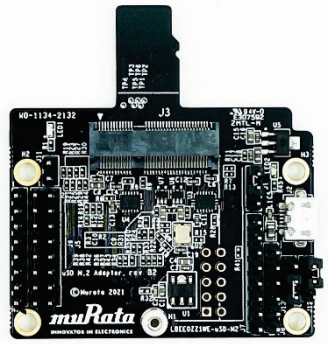
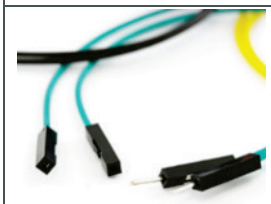
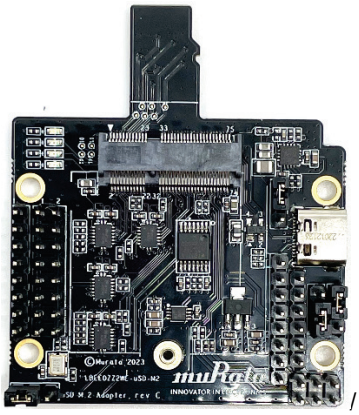
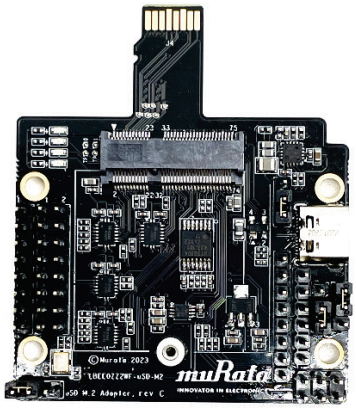
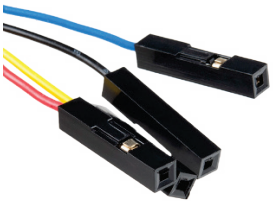
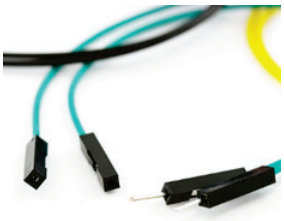
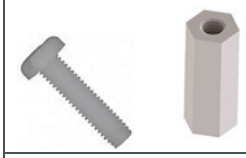
Picture of Contents	Description of Contents
	uSD-M.2 Adapter (Revision B2)
	M.2 screw for attaching Wi-Fi/Bluetooth M.2 Module
	75 mm 20-pos, 0.5 mm pitch flat/flex cable
	13 pieces 200 mm long male-to-female jumper cables (compatible with Arduino header)
	4 x 19 mm stand-offs in nylon and associated M3 screws
	microSD to SD Card Adapter

Table 5: uSD-M.2 Adapter Kit (Rev C) Contents

Picture of Contents	Description of Contents
 / 	uSD-M.2 Adapter (Version 2WE, Rev. C) or uSD-M.2 Adapter (Version 2WF, Rev. C)
	1 M.2 screw for attaching Wi-Fi/Bluetooth M.2 Evaluation Board (EVB)
	25 pieces 200 mm long female-to-female jumper cables (compatible with Arduino header).
	25 pieces 200 mm long male-to-female jumper cables (compatible with Arduino header).
	4 x 18 mm stand-off and screw sets in nylon.
	microSD to SD Card Adapter

For more information on the uSD-M.2 Adapter, refer to Section 9 or go to the Adapter landing page.

3.3 NXP i.MX versus Murata Module Interconnect

Table 6 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/C 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.2P**": Murata's uSD-M.2 Adapter (Rev B2/C) 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 1ZM 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 an 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, 1XL, 2XS) can be used.
- "**M.2W**": Only Embedded Artists' Wi-Fi/BT M.2 EVB is required. The NXP i.MX EVK has an 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 6**, 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 6: NXP i.MX/Murata Module Interconnect

NXP i.MX EVK Part Number	1ZM	1YM-SDIO	1YM-PCIe	1XL/2XS-SDIO	1XL/2XS-PCIe	1XK/2XK	2DS	2DL	2EL
	NXP 88W8987	NXP 88W8997	NXP 88W8997	NXP 88W9098	NXP 88W9098	NXP IW416	NXP 88W8801	NXP IW611	NXP IW612
MCIMX8QXP-CPU	NC[2]	NC[2]	M.2[3]	NC[2]	M.2[3]	NC[2]	NC[2]	NC[2]	NC[2]
8MPLUSLPD4-EVK	M.2[3]	M.2[3]	M.2[3]	M.2[3]	M.2[3]	M.2[3]	M.2[3]	M.2[3]	M.2[3]

[2] No Connection options available
[3] Wi-Fi/BT M.2 EVB plugs directly into M.2 connector

MCIMX8M-EVKB	uSD-M.2[4]	uSD-M.2[4]	M.2[3]	uSD-M.2[4]	M.2[3]	uSD-M.2[4]	uSD-M.2[4]	uSD-M.2[4]	uSD-M.2[4]
8MMINILPD4-EVKB	uSD-M.2P[5]	uSD-M.2P[5]	M.2W[6]	uSD-M.2P[5]	M.2W[6]	uSD-M.2P[5]	uSD-M.2P[5]	uSD-M.2P[5]	uSD-M.2P[5]
8MNANOD4-EVK	uSD-M.2[4]	uSD-M.2[4]	NC[2]	uSD-M.2[4]	NC[2]	uSD-M.2[4]	uSD-M.2[4]	uSD-M.2[4]	uSD-M.2[4]
MCIMX8ULP-EVK	M.2[3]	M.2[3]	NC[2]	M.2[3]	NC[2]	M.2[3]	M.2[3]	M.2[3]	M.2[3]
MCIMX93-EVK	M.2[3]	M.2[3]	NC[2]	M.2[3]	NC[2]	M.2[3]	M.2[3]	M.2[3]	M.2[3]
MCIMX6UL-EVKB	uSD-M.2[4]	uSD-M.2[4]	NC[2]	uSD-M.2[4]	NC[2]	uSD-M.2[4]	uSD-M.2[4]	uSD-M.2[4]	uSD-M.2[4]
MCIMX6ULL-EVK	uSD-M.2[4]	uSD-M.2[4]	NC[2]	uSD-M.2[4]	NC[2]	uSD-M.2[4]	uSD-M.2[4]	uSD-M.2[4]	uSD-M.2[4]

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

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

4.1 Murata's Customized NXP Wireless Driver Release

The NXP i.MX 5.15.32/6.1.1/6.1.36/6.6.3/6.6.23 BSP release for i.MX 6/8 integrates the NXP WLAN driver and has Bluetooth (BlueZ stack) support. Murata's Linux regulatory solution also provides the following enhancements/customizations:

- Comprehensive support for 2DS/1XK/2XK/1ZM/1YM/1XL/2XS/2DL/2EL on all possible i.MX targets - hardware interconnect is the only limiting factor: see **Table 8**.
- 1.8V SDIO VIO configuration for WLAN interface on i.MX 6UL(L) EVKs.

4.2 Specific i.MX Target Support Details

Murata's Linux regulatory solution supports the following NXP i.MX EVKs as outlined in **Table 7.** "MACHINE=target" is a direct reference to Yocto build. The "target" string is the keyword used to select hardware configuration for the build (important knowledge when running Murata build script). From this table, you can find a proper version of Linux Kernel for your target platform.

You can then refer to **Table 8** for the support of interrupt configuration, corresponding DTB (Device Tree Blob) files 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 (2DS/1XK/2XK/1ZM/1YM/2EL/2DL) supporting WLAN-SDIO interface only interface at 1.8V VIO. The i.MX 8M Mini/Nano/Quad EVKs with "uSD-M.2P" configuration require additional part(s) as documented in Section 7.3.

[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

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

NXP i.MX EVK Part #	NXP i.MX EVK	MACHINE= target	Kernel 5.15.32	Kernel 6.1.1	Kernel 6.1.36	Kernel 6.6.3	Kernel 6.6.23
MCIMX93-EVK	i.MX 93 EVK	imx93evk	X	✓	✓	✓	✓
MCIMX8QXP-CPU	i.MX 8QuadXPlus MEK	imx8qxpmek	✓	✓	✓	✓	✓
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	✓	✓	✓	✓	✓
MCIMX8ULP-EVK	i.MX 8ULP EVK	imx8ulp-lpddr4-evk	X	X	✓	✓	✓
MCIMX6UL-EVKB	i.MX 6UL EVK	imx6ulevk	✓	✓	✓	✓	✓
MCIMX6ULL-EVK	i.MX 6ULL EVK	imx6ull14x14evk	✓	✓	✓	✓	✓

Table 8: i.MX6/8/9 Targets supported by Murata

Target (MACHINE)	Hardware Config	i.MX DTB File	Interrupt Config
imx8qxpmek	M.2[7]	imx8qxp-mek.dtb	N/A
imx8mp-lpddr4-evk	M.2[7]	imx8mp-evk-usdhc1-m2.dtb	SDIO
imx8mp-lpddr4-evk	M.2[7]	imx8mp-evk.dtb	N/A
imx8mqevk	M.2[7]	imx8mq-evk-pcie1-m2.dtb	N/A
imx8mqevk	uSD-M.2[8]	imx8mq-evk.dtb	SDIO
imx8mmevk	M.2W[9]	imx8mm-evk.dtb	N/A
imx8mmevk	uSD-M.2P[10]	imx8mm-evk.dtb	SDIO
imx8mnddr4evk	uSD-M.2P[10]	imx8mn-evk.dtb	SDIO
imx8ulp-lpddr4-evk / imx8ulp-9x9-evk	M.2[7]	imx8ulp-evk.dtb	SDIO
imx93evk / imx93-11x11-lpddr4x-evk	M.2[7]	imx93-evk.dtb	SDIO
imx6ulevk	uSD-M.2[8]	imx6ul-14x14-evk-btwifi.dtb[11]	SDIO
imx6ull14x14evk	uSD-M.2[8]	imx6ull-14x14-evk-btwifi.dtb[11]	SDIO

[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

4.3 NXP's Default Demo Image

To easily enable NXP-based Wi-Fi/Bluetooth functionality, users must download default NXP's demo images of 5.15.32, 6.1.1, 6.1.36, 6.6.3 or 6.6.23 for i.MX platforms.

Table 9 below shows the locations of the various NXP demo images.

Table 9: NXP Default Demo Images

NXP i.MX EVK Part #	Kernel	Location
MCIMX93-EVK	6.6.23	LF_v6.6.23-2.0.0_images_IMX93EVK.zip
	6.6.3	LF_v6.6.3-1.0.0_images_IMX93EVK.zip
	6.1.36	LF_v6.1.36-2.1.0_images_IMX93EVK.zip
	6.1.1	LF_v6.1.1_1.0.0_images_IMX93EVK.zip
MCIMX8QXP-CPU	6.6.23	LF_v6.6.23-2.0.0_images_IMX8QXPC0MEK.zip
	6.6.3	LF_v6.6.3-1.0.0_images_IMX8QXPC0MEK.zip
	6.1.36	LF_v6.1.36-2.1.0_images_IMX8QXPC0MEK.zip
	6.1.1	LF_v6.1.1_1.0.0_images_IMX8QXPC0MEK.zip
	5.15.32	LF_v5.15.32-2.0.0_images_IMX8QXPC0MEK.zip
8MPLUSLPD4-EVK	6.6.23	LF_v6.6.23-2.0.0_images_IMX8MPEVK.zip
	6.6.3	LF_v6.6.3-1.0.0_images_IMX8MPEVK.zip
	6.1.36	LF_v6.1.36-2.1.0_images_IMX8MPEVK.zip
	6.1.1	LF_v6.1.1_1.0.0_images_IMX8MPEVK.zip
	5.15.32	LF_v5.15.32-2.0.0_images_IMX8MPEVK.zip
MCIMX8M-EVKB	6.6.23	LF_v6.6.23-2.0.0_images_IMX8MQEVK.zip
	6.6.3	LF_v6.6.3-1.0.0_images_IMX8MQEVK.zip
	6.1.36	LF_v6.1.36-2.1.0_images_IMX8MQEVK.zip
	6.1.1	LF_v6.1.1_1.0.0_images_IMX8MQEVK.zip
	5.15.32	LF_v5.15.32-2.0.0_images_IMX8MQEVK.zip
8MMINILPD4-EVKB	6.6.23	LF_v6.6.23-2.0.0_images_IMX8MMEVK.zip
	6.6.3	LF_v6.6.3-1.0.0_images_IMX8MMEVK.zip
	6.1.36	LF_v6.1.36-2.1.0_images_IMX8MMEVK.zip
	6.1.1	LF_v6.1.1_1.0.0_images_IMX8MMEVK.zip
	5.15.32	LF_v5.15.32-2.0.0_images_IMX8MMEVK.zip
8MNANOD4-EVK	6.6.23	LF_v6.6.23-2.0.0_images_IMX8MNEVK.zip
	6.6.3	LF_v6.6.3-1.0.0_images_IMX8MNEVK.zip
	6.1.36	LF_v6.1.36-2.1.0_images_IMX8MNEVK.zip
	6.1.1	LF_v6.1.1_1.0.0_images_IMX8MNEVK.zip
	5.15.32	LF_v5.15.32-2.0.0_images_IMX8MNEVK.zip
MCIMX8ULP-EVK	6.6.23	LF_v6.6.23-2.0.0_images_IMX8ULPEVK.zip

NXP i.MX EVK Part #	Kernel	Location
	6.1.36	LF_v6.1.36-2.1.0_images_IMX8ULPEVK.zip
MCIMX6UL-EVKB	6.6.23	LF_v6.6.23-2.0.0_images_IMX6UL7D.zip
	6.6.3	LF_v6.6.3-1.0.0_images_IMX6UL7D.zip
	6.1.36	LF_v6.1.36-2.1.0_images_IMX6UL7D.zip
	6.1.1	LF_v6.1.1_1.0.0_images_IMX6UL7D.zip
	5.15.32	LF_v5.15.32-2.0.0_images_IMX6UL7D.zip
MCIMX6ULL-EVK	6.6.23	LF_v6.6.23-2.0.0_images_IMX6UL7D.zip
	6.6.3	LF_v6.6.3-1.0.0_images_IMX6UL7D.zip
	6.1.36	LF_v6.1.36-2.1.0_images_IMX6UL7D.zip
	6.1.1	LF_v6.1.1_1.0.0_images_IMX6UL7D.zip
	5.15.32	LF_v5.15.32-2.0.0_images_IMX6UL7D.zip

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.

4.4 Additional Hardware/Software Considerations

4.4.1 1.8V Versus 3.3V VIO Signaling using Murata's uSD-M.2 Adapter

As shown in **Table 6** and **Table 8**, 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.

4.4.2 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.

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

As shown in **Table 6** and **Table 8**, 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.

4.4.4 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 8** 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 Regulatory Solution under the folder, "patch_files"

- **Load Drivers – Every time:**

  Enter the following command:

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

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

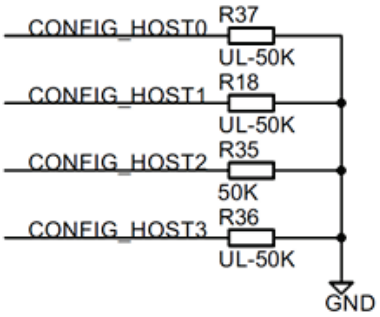
Type 1YM module supports more than one interface configuration. The Embedded Artists' Wi-Fi/BT M.2 EVB (EAR00370) is default configured (via resistor strapping options) for WLAN-PCIe/BT-UART. However, the following configuration options should be considered:

- 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 8** 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)

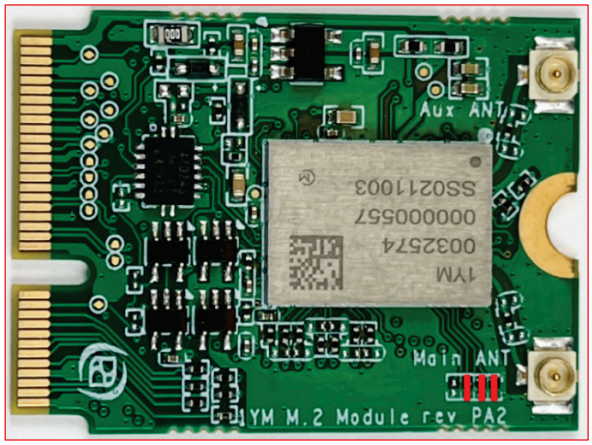
Figure 7: Type 1YM M.2 EVB Configuration Strapping Options



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)



4.4.6 Type 1XL/2XS M.2 EVB WLAN Bus Interface Configuration

Type 1XL/2XS modules support more than one interface configuration. The Embedded Artists' Wi-Fi/BT M.2 EVBs (EAR00387/EAR00411) are default configured (via resistor strapping options) for WLAN-PCIe/BT-UART. However, the following configuration options should be considered:

- WLAN-PCIe/BT-UART (**1XL/2XS** or **1XL/2X-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 (**1XL/2XS-SDIO**): This configuration is a supported option in NXP baseline BSP release. For existing interconnect, this mode is used when plugging in the M.2 EVB via the Murata uSD-M.2 adapter into the microSD connector. See **Figure 9** for resistor strapping option – two 50K Ohm resistors need to be added (to right of default one).

Regarding the actual configuration of the Embedded Artists Type 1XL/2XS 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.

Depending on the board version, the rework process is slightly different, see **Figure 9** and **Figure 10** below for rev A/PB2 and rev B, respectively.

For Rev A/PB2 modules, the "CONFIG_HOST" resistors are (in order left to right as pictured):

- R33 (CONF_HOST0) (install this resistor for WLAN-SDIO option)
- R18 (CONF_HOST1) (install this resistor in addition to R33 for WLAN-SDIO option)
- R35 (CONF_HOST2) (default resistor already installed for WLAN-PCIe. Remove this resistor for WLAN-SDIO option)

Figure 9: Type 1XL/2XS M.2 (Rev. A/PB2) Configured for WLAN-SDIO/BT-UART (1XL/2XS-SDIO)

Interface Control from top to bottom:

- R33 (CONF_HOST0)
- R18 (CONF_HOST1)
- R35 (CONF_HOST2)

PCIe: No resistors mounted on R33 and R18. Mount a 50 KΩ 0402 resistors on R35.

SDIO: Mount 50 KΩ 0402 resistors on all three positions: R33, R18 and R35.

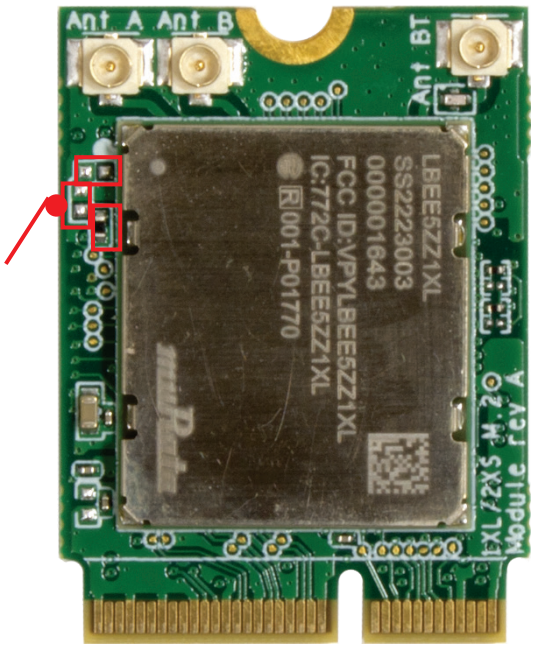
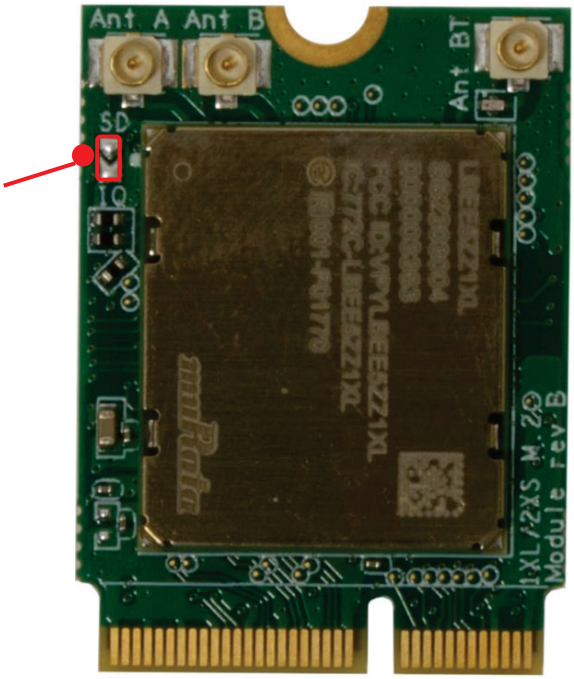


Figure 10: Type 1XL/2XS M.2 (Rev. B) Configured for WLAN-SDIO/BT-UART (1XL/2XS-SDIO)

Interface Control

PCIe: No resistors or solder bump mounted in this position.

SDIO: Mount a zero Ω 0402 resistors or solder bump in this position.



4.4.7 Type 1XL/2XS Production Process Number Changes

Murata has two types of Type 1XL/2XS modules with different Production Process numbers, printed on top of the module. There are no other visual differences between them. The initial baud rate of Bluetooth firmware is set to different values in the two types (one sets it to 3 Mbps, the other to 115200 bps). This may require a change in the DTS file based on the batch used. By default, the DTS file supports 115200 bps. The type can be identified from the second line of the module print, as shown in **Figure 11**.

Figure 11: Type 1XL/2XS Production Process Numbers

If the Production Process number starts with “SS”, the Bluetooth setting uses 115200 bps and no change in DTS file required. If the number starts with anything else (for example. “SA”), the baud rate used is 3 Mbps and requires the following property line to be added to the Bluetooth device node in the DTS file.

```
fw-init-baudrate = <3000000>
```

The **Table 10** below shows a few sample Production Process numbers and the initial Bluetooth baud rates.

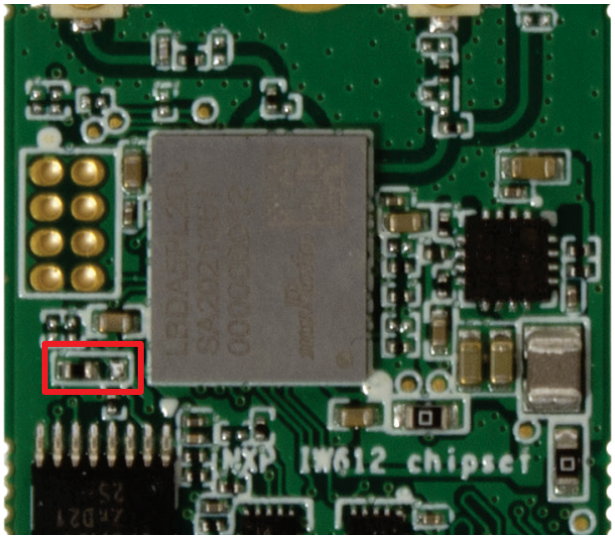
Table 10: Type 1XL/2XS Bluetooth Baud Rates

Module	Production Process Number	Baud Rate
1XL	SS2223003	115200 bps
	SS2303004	115200 bps
	SA2021047	3 Mbps
2XS	SS2117001	115200 bps
	SS2307002	115200 bps
	SA2021047	3 Mbps

4.4.8 2EL M.2 EVB Rework for NXP i.MX 8M Mini EVK

For connecting Type 2EL M.2 EVB's SPI interface to NXP i.MX 8M Mini, a minor rework is required on the 2EL EVB. This is because Type 2EL M.2 EVB uses I2C slave address 0x20, which is already used by i.MX 8M Mini EVK for other I2C slave devices. To overcome this limitation of NXP i.MX 8M Mini EVK, the I2C slave address of 2EL M.2 EVB has to be changed to 0x21, by changing the IO expander (U2) SJ1 resistor (shown in **Figure 12** below) from the 1-2 position to the 2-3 position. These changes are needed for evaluating 802.15.4. Also, for evaluating SPI interface, Murata's new uSD-M.2 adapter version (Rev C - 2WF) must be used.

Figure 12: Type 2EL M.2 EVB Rework for NXP i.MX 8M Mini EVK



4.4.9 Firmware Options

The parameters used during Wi-Fi module load are specified in the /lib/firmware/nxp/wifi_mod_para.conf file, which is passed during the modprobe command. This configuration file, by default, loads the combo firmware supporting both Wi-Fi and Bluetooth. However, there are other firmware files available in the /lib/firmware/nxp/ folder that can be used. For example, the following files are available for Type 2EL module (which also has an additional radio for 802.15.4):

- sd_w61x_v1.bin.se: Wi-Fi only firmware (SDIO interface)
- sduart_nw61x_v1.bin.se: Wi-Fi (SDIO) + Bluetooth (UART) combo firmware
- uartspi_n61x_v1.bin.se: Bluetooth (UART) + 802.15.4 (SPI) firmware
- uartuart_n61x_v1.bin.se: Bluetooth (UART) + 802.15.4 (UART) firmware

The prefix for these firmware file names specifies the interface(s) supported by the file.

4.5 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.

4.5.1 Install Ubuntu

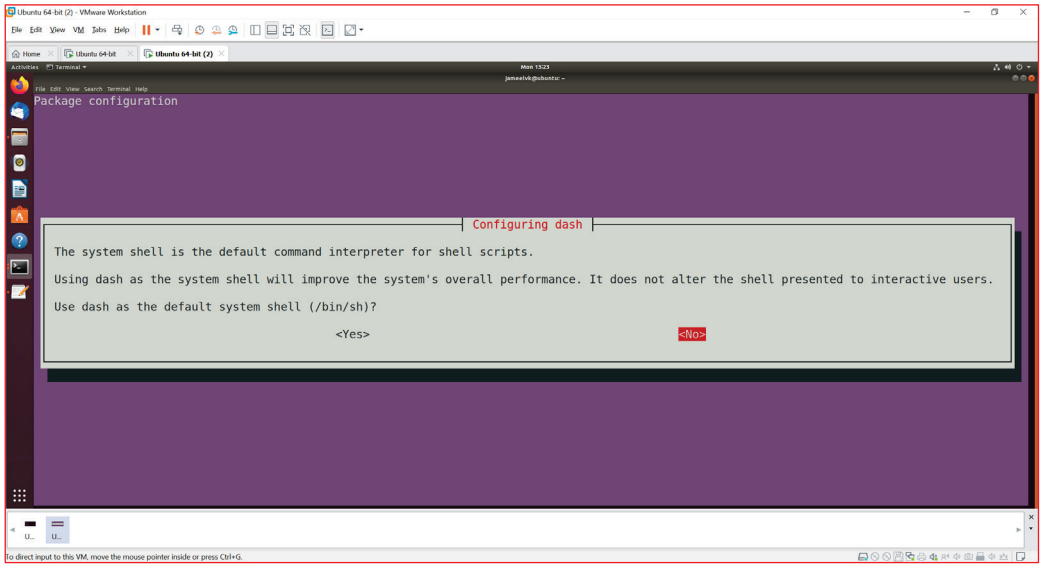
First step is to install Ubuntu 14.04, 16.04, 18.04, 20.04 or 22.04 (Murata's build is verified on Ubuntu 22.04 64-bit install) on the host - native PC or virtual environment like VMware. Host PC typically used requires Ubuntu 22.04/20.04/18.04/16.04/14.04 installed with 50 GB free disk space (80 GB 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 13**.

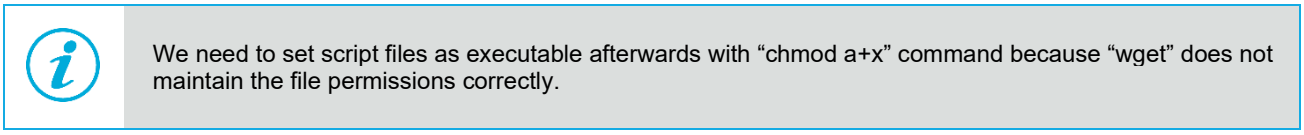
Figure 13: Configuring dash



4.5.2 Download Murata's Script Files

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

15. 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.
16. 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
```

4.5.3 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

4.5.4 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 22.04 (preferred), 20.04, 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 an i.MX BSP folder specific to the desired i.MX Yocto Release. The i.MX Yocto distribution cannot build different versions of Yocto (Linux kernel) in the same folder. Currently the following Yocto releases are supported:
  - 5.15.32_2.0.0
  - 6.1.1_1.0.0
  - 6.1.36_2.1.0
  - 6.6.3_1.0.0
  - 6.6.23_2.0.0
- 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 14.04/16.04/18.04/20.04/22.04).
- 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 Murata module. The regulatory solution provided by Murata is customized for each Murata module.
- Select i.MX Yocto release. As already pointed out the current i.MX BSP folder (from which script is being executed) can only support one i.MX Yocto release. If you need to test/evaluate different Yocto/kernel versions, then you must create additional folders.
- Select i.MX target: refer to **Table 7** and **Table 8** 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 Module
- 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.

5 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 11** below.

These two configurations include the i.MX 8M Mini EVK (8MMINILPD4-EVK variant) and i.MX 8M Nano EVK with the “uSD-M.2P” configuration. In both cases the NXP i.MX EVKs need to boot from eMMC in place of microSD card given that the WLAN-SDIO connection is over the uSD connector. Refer to Section 4.2 for detailed steps on flashing these platforms.

Table 11: Select Hardware Configurations which use eMMC Boot Configuration

Target (MACHINE)	Hardware Config	i.MX DTB File	Boot Config	Interrupt Config
imx8mmevk	M.2W[12]	imx8mm-evk.dtb	uSD	N/A
imx8mmevk	uSD-M.2P[13]	imx8mm-evk.dtb	eMMC	SDIO
imx8mnddr4evk	uSD-M.2P[13]	imx8mn-evk.dtb	eMMC	SDIO

5.1 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.

[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

5.1.1 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 14**. This Kingston device (MobileLite Plus microSD Reader) provides direct plug-ins for microSD and SD cards. It supports USB 3.2 and UHS-II microSD cards – allowing very fast transfer speeds. With the "right" (micro) SD Card Reader/Writer and UHS (micro) SD card, flashing a 1 GB i.MX image can be done in 10~20 seconds versus 1~2 minutes (or more).

Figure 14: 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
```

3. SD Card is now flashed with customized Murata wireless image.

The following table lists the commands for flashing 6UL/6ULL EVKs. The image names used here are those of NXP demo images. Replace as required.

Table 12: Linux Commands to Flash SD Card

Target (MACHINE)	Linux Command to Flash
imx6ulevk	sudo dd if=imx-image-full-imx6ul7d.wic of=/dev/sdx bs=1M && sync
imx6ull14x14evk	sudo dd if=imx-image-full-imx6ul7d.wic of=/dev/sdx bs=1M && sync sudo dd if=u-boot-imx6ull14x14evk_sd.imx of=/dev/sdx bs=1k seek=1 conv=fsync

5.1.2 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 utilities[15].
- Now click the "Write" button. A warning window will pop up that warns that the device being written to may be corrupted.
- Upon completion, a window with "Write Successful" should appear.
- Click "OK" on the "Write Successful" window.
- Now click "Exit" on "Win32 Disk Imager" window.
- To be safe, you may elect to "eject" the SD card removable memory device before removing it.

5.1.3 Steps to Load DTB Files

To load modified DTB files (such as those provided by the Murata Linux Regulatory 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.

5.2 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.

[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.

5.2.1 Software File Preparation

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

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

Table 13 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.

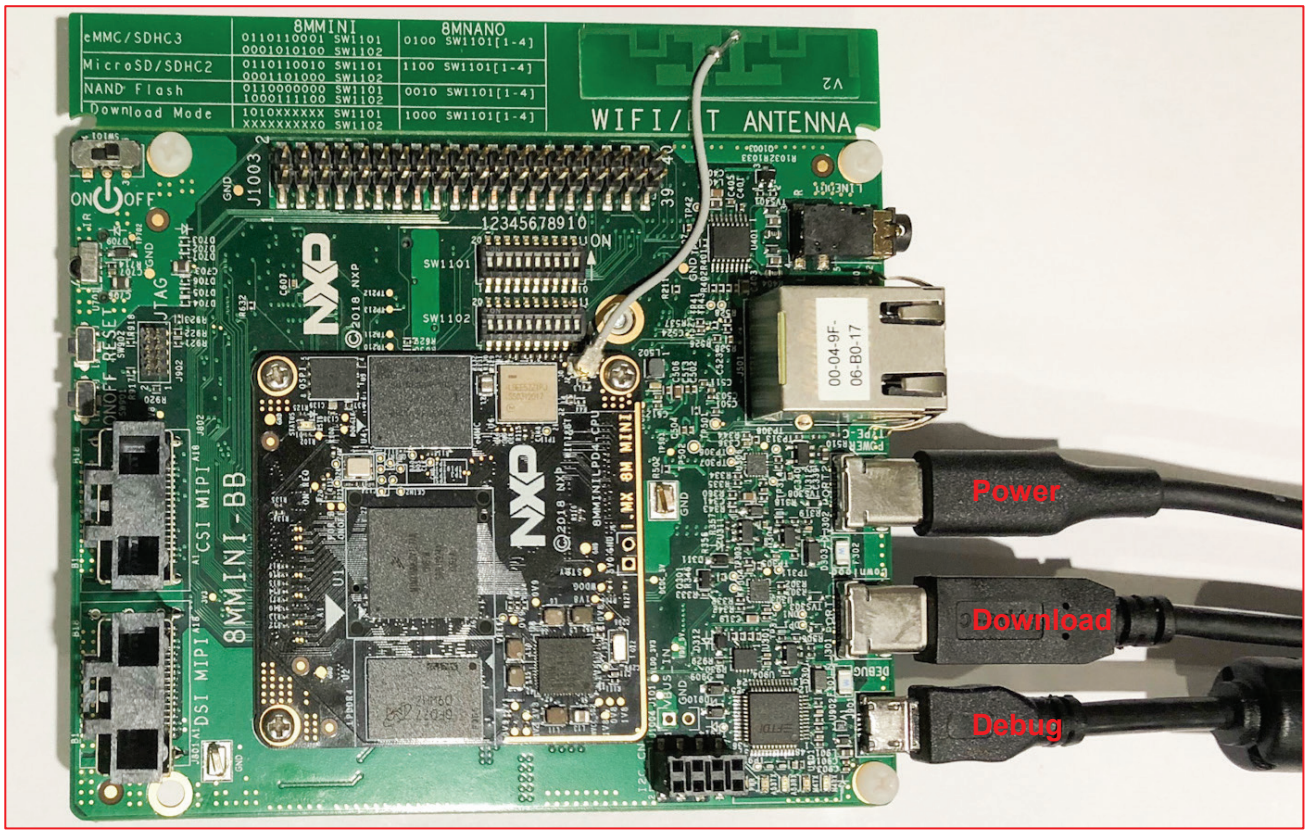
Table 13: Files to Flash i.MX 8M Mini & 8M Nano EVKs

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/

5.2.2 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 15** shows the necessary connections for power, download, and debug (serial console).

Figure 15: 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.

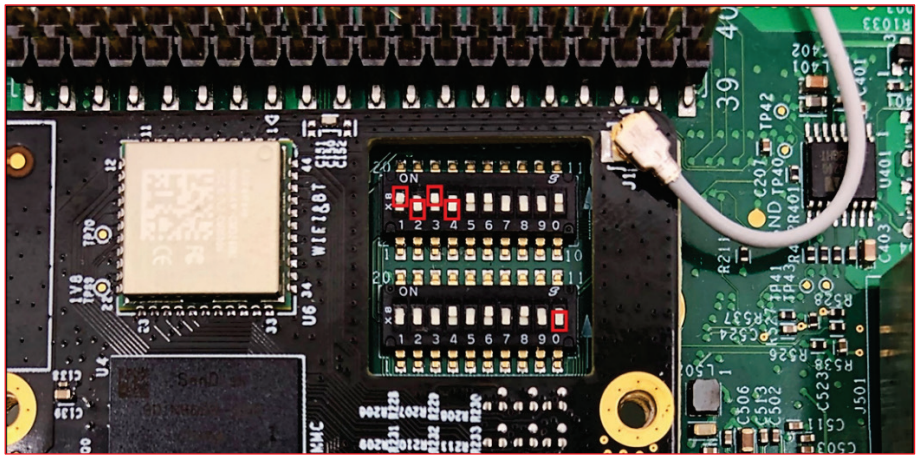
The DIP switch settings for i.MX 8M Mini EVK and 8M Nano EVK are different.

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 16**.

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

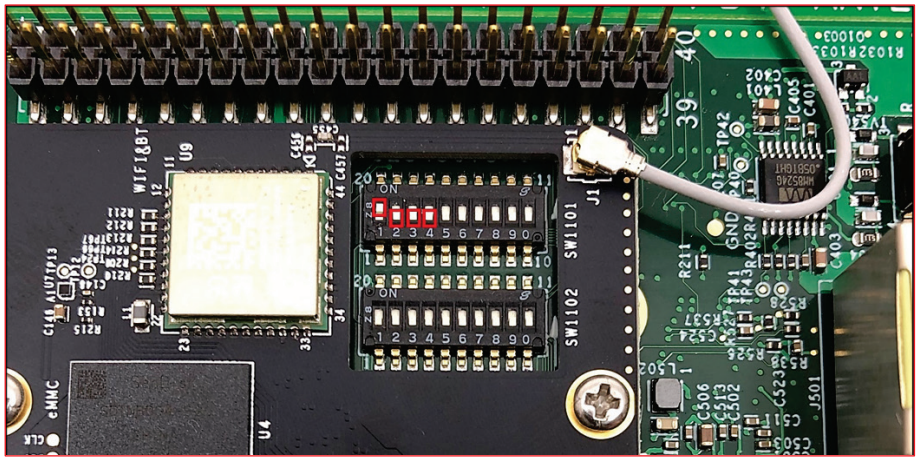


For i.MX 8M Nano EVK (8MNANOD4-EVK), this is accomplished by setting the DIP switch (SW1101) to the positions below. The DIP switch settings are also shown in **Figure 17**.

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

Where 1 – ON, 0 – OFF.

Figure 17: i.MX 8M Nano EVK DIP Switches Configured for Download



5.2.3 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 13**, 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
```

5.2.4 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.

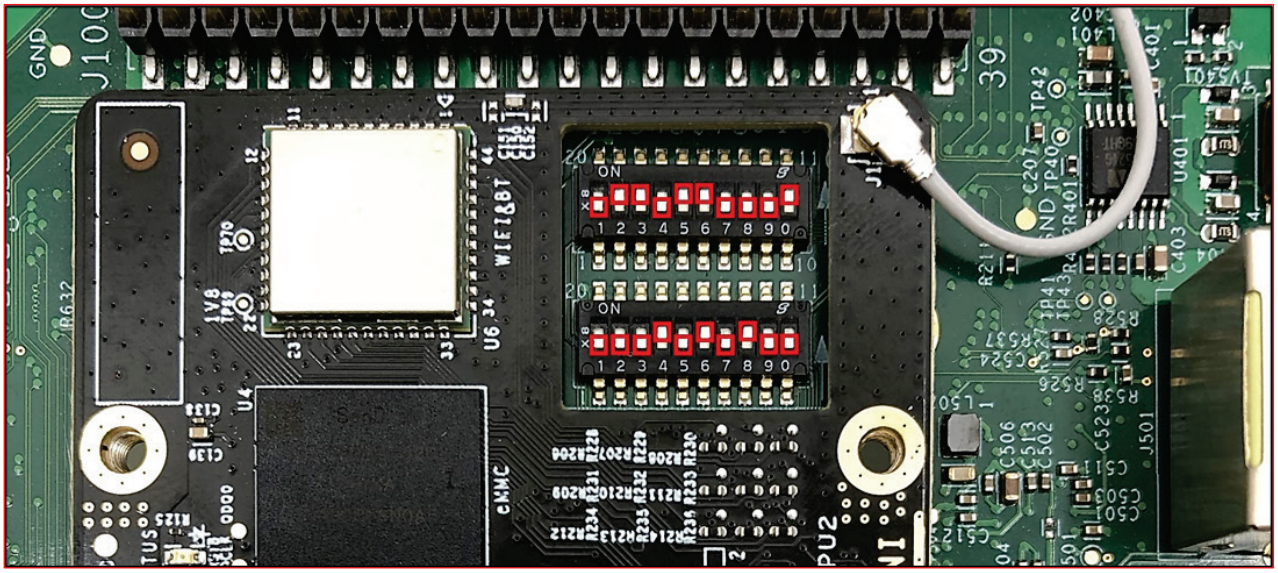
Now you *must* change the DIP switch settings to boot from eMMC. Note that the DIP switch settings for i.MX 8M Mini EVK and 8M Nano EVK are different.

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 18**.

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

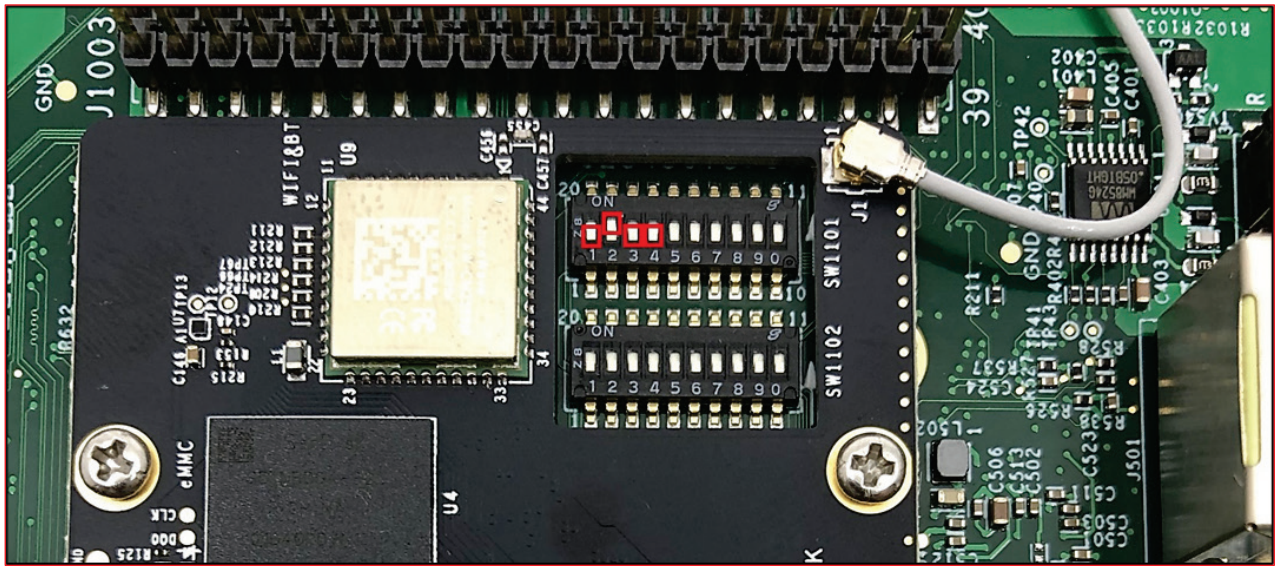


For i.MX 8M Nano EVK (8MNANOD4-EVK), this is accomplished by setting the DIP switch (SW1101) to the positions below. The DIP switch settings are also shown in **Figure 19.**

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

Where 1 – ON, 0 – OFF.

Figure 19: i.MX 8M Nano EVK DIP Switches Configured for eMMC Boot



6 Murata's Regulatory Solution for NXP Modules

This section describes Murata's regulatory solution for NXP Modules. Murata provides a simple but limited turnkey solution for quick verification. For a complete regulatory solution, a more involved manual procedure is required. This section describes both.

6.1 Murata Turnkey Solution

This regulatory solution provides a simple and easy mechanism to test Murata modules, however this does not provide the full support of all the regions.

6.1.1 Description of Regulatory Solution for WLAN/BT

WLAN: Murata's regulatory solution includes the regulatory binary files for various modules (2DS/1XK/1ZM/1YM/1XL/2XS/2EL/2DL) 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" and "mlanutl" application.

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

6.1.2 Tree view of the files list for Regulatory Solution

```
├── murata
│   ├── files
│   │       ├ 1XK
│   │       │   ├── db.txt
│   │       │   ├── db-murata.txt
│   │       │   ├── ed_mac.bin
│   │       │   ├── ed_mac.conf
│   │       │   ├── murata.hex
│   │       │   ├── regulatory.db
│   │       │   ├── regulatory.db.p7s
│   │       │   ├── txpower_CA.bin
│   │       │   ├── txpower_CA.conf
│   │       │   ├── txpower_EU.bin
│   │       │   ├── txpower_EU.conf
│   │       │   ├── txpower_JP.bin
│   │       │   ├── txpower_JP.conf
│   │       │   └── txpower_US.bin
│   │       │   ├── txpower_US.conf
```

```
│   │       ├ 1XL
│   │       │   ├── db.txt
│   │       │   ├── db-murata.txt
│   │       │   ├── ed_mac.bin
│   │       │   ├── ed_mac.conf
│   │       │   ├── murata.hex
│   │       │   ├── regulatory.db
│   │       │   ├── regulatory.db.p7s
│   │       │   ├── rutxpower_CA.bin
│   │       │   ├── rutxpower_CA.conf
│   │       │   ├── rutxpower_EU.bin
│   │       │   ├── rutxpower_EU.conf
│   │       │   ├── rutxpower_JP.bin
│   │       │   ├── rutxpower_JP.conf
│   │       │   └── rutxpower_US.bin
│   │       │   ├── rutxpower_US.conf
│   │       │   ├── txpower_CA.bin
│   │       │   ├── txpower_CA.conf
│   │       │   ├── txpower_EU.bin
│   │       │   ├── txpower_EU.conf
│   │       │   ├── txpower_JP.bin
│   │       │   ├── txpower_JP.conf
│   │       │   └── txpower_US.bin
│   │       │   ├── txpower_US.conf
│   │       ├ 1YM
│   │       │   ├── db.txt
│   │       │   ├── db-murata.txt
│   │       │   ├── ed_mac.bin
│   │       │   ├── ed_mac.conf
│   │       │   ├── murata.hex
│   │       │   ├── regulatory.db
│   │       │   ├── regulatory.db.p7s
```

```
│   │   │   ├── txpower_CA.bin
│   │   │   ├── txpower_CA.conf
│   │   │   ├── txpower_EU.bin
│   │   │   ├── txpower_EU.conf
│   │   │   ├── txpower_JP.bin
│   │   │   ├── txpower_JP.conf
│   │   │   └── txpower_US.bin
│   │   │   ├── txpower_US.conf
│   │   ├ 1ZM
│   │   │   ├── db.txt
│   │   │   ├── db-murata.txt
│   │   │   ├── ed_mac.bin
│   │   │   ├── ed_mac.conf
│   │   │   ├── murata.hex
│   │   │   ├── regulatory.db
│   │   │   ├── regulatory.db.p7s
│   │   │   ├── txpower_CA.bin
│   │   │   ├── txpower_CA.conf
│   │   │   ├── txpower_EU.bin
│   │   │   ├── txpower_EU.conf
│   │   │   ├── txpower_JP.bin
│   │   │   ├── txpower_JP.conf
│   │   │   └── txpower_US.bin
│   │   │   ├── txpower_US.conf
│   │   ├ 2DS
│   │   │   ├── db.txt
│   │   │   ├── db-murata.txt
│   │   │   ├── ed_mac.bin
│   │   │   ├── ed_mac.conf
│   │   │   ├── murata.hex
│   │   │   ├── regulatory.db
│   │   │   ├── regulatory.db.p7s
```

```
│   │   │   ├── txpower_CA.bin
│   │   │   ├── txpower_CA.conf
│   │   │   ├── txpower_EU.bin
│   │   │   ├── txpower_EU.conf
│   │   │   ├── txpower_JP.bin
│   │   │   ├── txpower_JP.conf
│   │   │   └── txpower_US.bin
│   │   │   ├── txpower_US.conf
│   │   ├ 2DL
│   │   │   ├── WlanCalData_ext_NXP_dANT.conf
│   │   │   ├── bt_power_config_EU.sh
│   │   │   ├── bt_power_config_JP.sh
│   │   │   ├── bt_power_config_US_CA.sh
│   │   │   ├── db.txt
│   │   │   ├── db-murata.txt
│   │   │   ├── ed_mac.bin
│   │   │   ├── ed_mac.conf
│   │   │   ├── murata.hex
│   │   │   ├── regulatory.bin
│   │   │   ├── regulatory.db
│   │   │   ├── regulatory.db.p7s
│   │   │   ├── rutxpower_CA.bin
│   │   │   ├── rutxpower_CA.conf
│   │   │   ├── rutxpower_EU.bin
│   │   │   ├── rutxpower_EU.conf
│   │   │   ├── rutxpower_JP.bin
│   │   │   ├── rutxpower_JP.conf
│   │   │   └── rutxpower_US.bin
│   │   │   ├── rutxpower_US.conf
│   │   │   ├── txpower_CA.bin
│   │   │   ├── txpower_CA.conf
│   │   │   ├── txpower_EU.bin
```

```
│   │   │   ├── txpower_EU.conf
│   │   │   ├── txpower_JP.bin
│   │   │   ├── txpower_JP.conf
│   │   │   └── txpower_US.bin
│   │   │   ├── txpower_US.conf
│   │   ├ 2EL
│   │   │   ├── WlanCalData_ext_NXP_dANT.conf
│   │   │   ├── bt_power_config_EU.sh
│   │   │   ├── bt_power_config_JP.sh
│   │   │   ├── bt_power_config_US_CA.sh
│   │   │   ├── db.txt
│   │   │   ├── db-murata.txt
│   │   │   ├── ed_mac.bin
│   │   │   ├── ed_mac.conf
│   │   │   ├── murata.hex
│   │   │   ├── regulatory.bin
│   │   │   ├── regulatory.db
│   │   │   ├── regulatory.db.p7s
│   │   │   ├── rutxpower_CA.bin
│   │   │   ├── rutxpower_CA.conf
│   │   │   ├── rutxpower_EU.bin
│   │   │   ├── rutxpower_EU.conf
│   │   │   ├── rutxpower_JP.bin
│   │   │   ├── rutxpower_JP.conf
│   │   │   └── rutxpower_US.bin
│   │   │   ├── rutxpower_US.conf
│   │   │   ├── txpower_CA.bin
│   │   │   ├── txpower_CA.conf
│   │   │   ├── txpower_EU.bin
│   │   │   ├── txpower_EU.conf
│   │   │   ├── txpower_JP.bin
│   │   │   ├── txpower_JP.conf
```

```
│   │       │   └── txpower_US.bin
│   │       │   ├── txpower_US.conf
│   │       │
│   │       ├ bt_power_config_1.sh
│   │       └ 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
```

6.1.2.1 Description of files

Table 14: Murata regulatory files

File Name	Description
db.txt	WLAN regulatory limitation configuration file (default).
db-murata.txt	WLAN regulatory limitation configuration file (Murata specific).
ed_mac.bin	WLAN Carrier Sense / Adaptivity threshold configuration file binary.
ed_mac.conf	WLAN Carrier Sense / Adaptivity threshold configuration file.
regulatory.db	File used by the Linux wireless subsystem to keep its regulatory database information.
regulatory.db.p7s	File used by the Linux wireless subsystem to keep its regulatory database information.
bt_power_config_1.sh	Bluetooth Tx power configuration file (except 2EL/2DL).
bt_power_config_EU.sh	Bluetooth Tx power configuration file for 2EL/2DL EU region.
bt_power_config_US_CA_JP.sh	Bluetooth Tx power configuration file for 2EL/2DL US/CA/JP regions.
wifi_mod_para_murata.conf	Default configuration file provided by NXP and used by Murata to modify the necessary parameters for various modules (2DS, 1XK, 1ZM, 1YM, 1XL/2XS, 2EL/2DL).
murata.hex	Key in hex file format
txpower_CA.bin	WLAN Tx power configuration files for Canada (IC) binary
txpower_CA.conf	WLAN Tx power configuration files for Canada (IC)
txpower_EU.bin	WLAN Tx power configuration files for EU (CE) binary
txpower_EU.conf	WLAN Tx power configuration files for EU (CE)
txpower_JP.bin	WLAN Tx power configuration files for Japan (TELEC) binary
txpower_JP.conf	WLAN Tx power configuration files for Japan (TELEC)
txpower_US.bin	WLAN Tx power configuration files for US (FCC) binary

txpower_US.conf	WLAN Tx power configuration files for US (FCC)
rutxpower_CA.bin	WLAN 11ax RU Tx power configuration files for Canada (IC) binary. Only applicable for 1XL/2XS/2EL/2DL
rutxpower_CA.conf	WLAN 11ax RU Tx power configuration files for Canada (IC). Only applicable for 1XL/2XS/2EL/2DL
rutxpower_EU.bin	WLAN 11ax RU Tx power configuration files for EU (CE) binary. Only applicable for 1XL/2XS/2DL/2EL
rutxpower_EU.conf	WLAN 11ax RU Tx power configuration files for EU (CE). Only applicable for 1XL/2XS/2DL/2EL
rutxpower_JP.bin	WLAN 11ax RU Tx power configuration files for Japan (TELEC) binary. Only applicable for 1XL/2XS/2EL/2DL
rutxpower_JP.conf	WLAN 11ax RU Tx power configuration files for Japan (TELEC). Only applicable for 1XL/2XS/2EL/2DL
rutxpower_US.bin	WLAN 11ax RU Tx power configuration files for US (FCC) binary. Only applicable for 1XL/2XS/2EL/2DL
rutxpower_US.conf	WLAN 11ax RU Tx power configuration files for US (FCC). Only applicable for 1XL/2XS/2EL/2DL
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

6.1.3 Process of Switching Region

The switch_regions.sh script and mlanutl application file allows the user to easily switch the regulatory region. To use the script, copy the folder, “murata” from the Murata Regulatory solution to “/lib/firmware/nxp” in the root file system. This is only required for the first time.

The Murata Build Script handles this process automatically.

Similarly, download the mlanutl application (either 32-bit or 64-bit) and place it in the home folder.

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

```
# switch_regions.sh <Module_name> <Country_code>
```

The following options are supported.

- Module name –
  - 2DS – For Embedded Artists’ 2DS M.2 EVB
  - 1XK – For Embedded Artists’ 1XK M.2 EVB
  - 1ZM – For Embedded Artists’ 1ZM M.2 EVB
  - 1YM – For Embedded Artists’ 1YM M.2 EVB (Both SDIO and PCIe)
  - 1XL – For Embedded Artists’ 1XL M.2 EVB
  - 2XS – For Embedded Artists’ 2XS M.2 EVB
  - 2DL – For Embedded Artists’ 2DL M.2 EVB

- 2EL – For Embedded Artists' 2EL M.2 EVB

- Country code
  - US - USA
  - DE - Europe
  - JP - Japan
  - CA – Canada

The country code for EU must be set to DE for the change to take effect.

This step sets up the driver parameters for correct configuration as per the settings. Once done, reboot the EVK for the changes to take effect.

The next step is loading the ed_mac.conf file and is required ONLY for EU region. Skip this step for other regions.

This is done using the downloaded mlanutl application. Issue the command below to load the configuration file.

```
# mlanutl <Interface_name> hostcmd /lib/firmware/nxp/ed_mac.conf ed_mac_ctrl_v2
HOSTCMD_RESP: CmdCode=0x130, Size=0x14, SeqNum=0x47, Result=0000
payload: len=12
01 00 05 00 01 00 0a 00 ff 00 1e 00
```

Now issue "iw reg set <Country_code>" to set the region.

```
$ iw reg set US
$ iw reg get
global
country US: DFS-FCC
        (902 - 904 @ 2), (N/A, 30), (N/A)
        (904 - 920 @ 16), (N/A, 30), (N/A)
        (920 - 928 @ 8), (N/A, 30), (N/A)
        (2400 - 2472 @ 40), (N/A, 30), (N/A)
        (5170 - 5250 @ 80), (N/A, 23), (N/A), AUTO-BW
        (5250 - 5330 @ 80), (N/A, 24), (0 ms), DFS, AUTO-BW
        (5490 - 5730 @ 80), (N/A, 24), (0 ms), DFS, AUTO-BW
        (5735 - 5835 @ 80), (N/A, 30), (N/A), AUTO-BW
        (5850 - 5895 @ 40), (N/A, 27), (N/A), NO-OUTDOOR, AUTO-BW, PASSIVE-SCAN
        (5925 - 7125 @ 320), (N/A, 12), (N/A), NO-OUTDOOR, PASSIVE-SCAN
        (57240 - 71000 @ 2160), (N/A, 40), (N/A)
```

6.1.3.1 Sample Description of Structure SD8987 from "wifi_mod_para.conf" configured for US with 1ZM module

```
SD8987 = {
          cfg80211_wext=0xf
          max_vir_bss=1
          cal_data_cfg=none
          ps_mode=1
          auto_ds=1
          host_mlme=1
```

```
            fw_name=nxp/sdiouart8987_combo_v0.bin
            txpwrlimit_cfg=nxp/txpower_US.bin
}
```

6.1.3.2 Sample Description of Structure PCIE8997 from "wifi_mod_para.conf" Configured for EU (DE) with 1YM Module

```
PCIE8997 = {
            cfg80211_wext=0xf
            max_vir_bss=1
            cal_data_cfg=none
            ps_mode=1
            auto_ds=1
            host_mlme=1
            fw_name=nxp/pcieuart8997_combo_v4.bin
            txpwrlimit_cfg=nxp/txpower_EU.bin
}
```

6.1.3.3 Sample Description of Structure SDIW612 from "wifi_mod_para.conf" Configured for US with 2EL Module

```
SDIW612 = {
        cfg80211_wext=0xf
        max_vir_bss=1
        cal_data_cfg=none
        ps_mode=1
        auto_ds=1
        host_mlme=1
        fw_name=nxp/sduart_nw61x_v1.bin.se
        init_hostcmd_cfg=nxp/rutxpower_US.bin
        txpwrlimit_cfg=nxp/txpower_US.bin
}
```

To specify dAnt (dedicated Antenna) for 2EL/2DL, please specify the following for 2EL/2DL:

- cal_data_cfg=nxp/WlanCalData_ext_NXP_dANT.conf

6.1.4 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)
```

6.2 Murata Full Regulatory Solution

Murata utilizes the kernel regdb mechanism to provide updated regulatory solution for Murata modules. However, this needs a manual procedure to customize it for a Murata module. The following steps describe this.

A new kernel image needs to be built for a particular Murata module. So, in case the user wants to use a different Murata module for evaluation, a new image needs to be built using the steps below.

There are three scenarios with regards to the regulatory support:

- Default regulatory support that comes with the kernel. This has no specific support for Murata modules. No additional steps are required in this case and the rest of this section can be skipped.
- Murata only regulatory support. This overwrites the default regulatory support of the Linux kernel. This is useful in case the user only wants to use Murata modules. The files required for this are hosted on Murata GitHub. For this, users can skip the Steps 1, 2 and 3 below, and proceed directly to Step 4.
- Merged regulatory support. This merges Murata module support with the kernel default regulatory support. This requires all the steps mentioned below to be followed.

Murata regulatory support (for both Murata only and merged options) supports the country codes for US, Japan, EU and Canada regions. In case user needs support for more country codes, the db.txt file(s) provided by Murata needs to be modified by the user and the same steps below need to be followed to generate the correct regulatory solution.

6.2.1 Overview

The steps required to rebuild the kernel modules (e.g. 6.6.23 Linux Kernel version) to enable full regulatory solution are given below. Refer to the below sections for more detailed steps.

1. Add the Murata hex file, “murata.hex”, to the kernel directory <linux-imx>/net/wireless/certs. No other changes are needed in the Linux kernel source code.
2. Build the Linux Kernel

```
$make -j8 zImage dtbs modules
```

3. Install the modules

```
$mkdir ../tempModules
$make modules_install INSTALL_MOD_PATH=../tempModules/
$cd ../tempModules/lib/modules
$tar -czvf 6.6.23-xyz.tar.gz 6.6.23-xyz
```

4. Build NXP Wi-Fi drivers (mlan.ko and moal.ko) against the Kernel (6.6.23) by following the NXP user manual UM11675.
5. Transfer the files to the target.
    - Flash kernel Image
    - Copy kernel modules (6.6.23-xyz.tar.gz) to /lib/modules and untar.
    - Copy mlan.ko and moal.ko to /home/root.
    - Copy Murata txpower*.bin files to /lib/firmware/nxp.

- Copy regulatory.db and regulatory.db.p7s to /lib/firmware.

6. Bring-up of WLAN

```
$insmod ./mlan.ko
$insmod ./moal.ko fw_name=nxp/sdiouartiw416_combo_v0.bin
cal_data_cfg=none host_mlme=1 cfg80211_wext=0xf ps_mode=2 auto_ds=2
txpwrlimit_cfg=nxp/txpower_US.bin
```

7. Change regulatory region as required

```
$iw reg set US
$iw reg get
```

The following sections detail the steps to customize the regulatory solution in more details.

6.2.2 Step 1: Set up wireless-regdb

1. Install required python package.

```
 $ sudo apt install python3-m2crypto
```

2. Clone the kernel wireless-regdb package.

```
$ git clone
https://kernel.googlesource.com/pub/scm/linux/kernel/git/sforshee/wireles
s-regdb

$ cd wireless-regdb
```

3. Edit the first line of db2bin.py file to set the correct python version.

```
#!/usr/bin/env python3
```

6.2.3 Step 2: Update db.txt File

This step is required if the user needs to modify the db.txt file provided by Murata to add/modify country codes.

1. Download the Murata db.txt file corresponding to the module from Murata GitHub.
   - Go to Murata GitHub page for regulatory solution.
   - Select the folder for the Murata module to be used.
   - To use Murata only module regulatory support, download the db-murata.txt file, and rename it to db.txt.
   - To use merged Murata module regulatory support, download the db.txt file.
2. Copy this db.txt file to the wireless-regdb folder to replace the default option. Users can also modify the content of the db.txt file to add/remove country code support.

> The Murata provided db.txt file contains the country codes US, DE, CP and JP that are specific to the module. The parameters within these country code sections should not be changed.

> For the rest of the steps, take note of your <machine/user_name> and replace appropriately in the commands shown below. In the examples provided, the user_name is set as “test”.

3. Perform a native make in wireless-regdb.

```
$ make
Generating regulatory.bin digitally signed by <user_name>...
./db2bin.py regulatory.bin db.txt ~/.wireless-regdb-
<user_name>.key.priv.pem
Generating certificate for <user_name>...
./gen-pubcert.sh ~/.wireless-regdb-<user_name>.key.priv.pem
<user_name>.x509.pem <user_name>
Signing regulatory.db (by <user_name>)...
$
```

4. Once the make is complete, the following files will be generated. These files will be used in Step 5.
   - regulatory.db
   - regulatory.db.p7s

6.2.4 Step 3: Convert Key

1. Invoke the following command to generate the key file.

```
$openssl asn1parse -out <user_name>.asn1 -inform PEM -in
<user_name>.x509.pem

# For example:
openssl asn1parse -out test.asn1 -inform PEM -in test.x509.pem
```

2. This command generates the asn1 file (e.g. test.asn1), as shown in **Figure 20** below.

Figure 20: Generating ASN1 File

```
test@ubuntu:~/projects/soln-for-nxp/wireless-regdb$ openssl asn1parse -out test.asn1 -inform PEM -in test.x509.pem
    0:d=0  hl=4 l= 679 cons: SEQUENCE
    4:d=1  hl=4 l= 399 cons: SEQUENCE
    8:d=2  hl=2 l=  20 prim: INTEGER           :2874033FF4DE17C17F26CAD1284DAAD1107A1588
   30:d=2  hl=2 l=  13 cons: SEQUENCE
   32:d=3  hl=2 l=   9 prim: OBJECT            :sha256WithRSAEncryption
   43:d=3  hl=2 l=   0 prim: NULL
   45:d=2  hl=2 l=  15 cons: SEQUENCE
   47:d=3  hl=2 l=  13 cons: SET
   49:d=4  hl=2 l=  11 cons: SEQUENCE
   51:d=5  hl=2 l=   3 prim: OBJECT            :commonName
   56:d=5  hl=2 l=   4 prim: UTF8STRING        :test
   62:d=2  hl=2 l=  32 cons: SEQUENCE
   64:d=3  hl=2 l=  13 prim: UTCTIME           :240702023943Z
   79:d=3  hl=2 l=  15 prim: GENERALIZEDTIME   :21240608023943Z
   96:d=2  hl=2 l=  15 cons: SEQUENCE
   98:d=3  hl=2 l=  13 cons: SET
  100:d=4  hl=2 l=  11 cons: SEQUENCE
  102:d=5  hl=2 l=   3 prim: OBJECT            :commonName
  107:d=5  hl=2 l=   4 prim: UTF8STRING        :test
  113:d=2  hl=4 l= 290 cons: SEQUENCE
  117:d=3  hl=2 l=  13 cons: SEQUENCE
  119:d=4  hl=2 l=   9 prim: OBJECT            :rsaEncryption
  130:d=4  hl=2 l=   0 prim: NULL
  132:d=3  hl=4 l= 271 prim: BIT STRING
  407:d=1  hl=2 l=  13 cons: SEQUENCE
  409:d=2  hl=2 l=   9 prim: OBJECT            :sha256WithRSAEncryption
  420:d=2  hl=2 l=   0 prim: NULL
  422:d=1  hl=4 l= 257 prim: BIT STRING
test@ubuntu:~/projects/soln-for-nxp/wireless-regdb$
```

3. Convert the binary to hex file by using the command below.

```
$ hexdump -v -e '"\n" 1/1 "  0x%02x," 6/1 " 0x%02x," 1/1 " 0x%02x,"'
./<user_name>.asn1 > murata.hex

# For example:
```

```
# hexdump -v -e '"\n" 1/1 "  0x%02x," 6/1 " 0x%02x," 1/1 " 0x%02x,"'
# ./test.asn1 > murata.hex
```

4. This command creates the hex file (e.g. murata.hex) from the binary key generated in the previous step (test.asn1), as shown in **Figure 21** below.

Figure 21: Generating HEX File

```
test@ubuntu:~/projects/tb9104-new-db-solution/6.6.3/wireless-regdb$ hexdump -v -e '"\n" 1/1 "  0x%02x," 6/1 " 0
x%02x," 1/1 " 0x%02x,"' ./test.asn1 > murata.hex
test@ubuntu:~/projects/tb9104-new-db-solution/6.6.3/wireless-regdb$ ls
CONTRIBUTING  debian-example   __pycache__        regulatory.db.5    test.x509.pem
db2bin.py     gen-pubcert.sh   README             regulatory.db.p7s  web
db2fw.py      LICENSE          regulatory.bin     sha1sum.txt        wens.key.pub.pem
dbparse.py    Makefile         regulatory.bin.5   test.asn1          wens.x509.pem
db.txt        murata.hex       regulatory.db      test.key.pub.pem   wireless-regdb.spec
test@ubuntu:~/projects/tb9104-new-db-solution/6.6.3/wireless-regdb$
```

5. Modify the .hex file to remove useless data around the last line, using any text editor (such as ‘vi’), as shown in **Figure 22** below.

Figure 22: Modifying HEX File

Original

```
0x48, 0xe0, 0xf6, 0x6b, 0xd4, 0x75, 0xbe, 0x33,
0xcf, 0xec, 0x45, 0x4a, 0xfd, 0x35, 0x35, 0x61,
0xbc, 0x9f, 0xb4, 0xe4, 0x84, 0xf7, 0x5c, 0xa2,
0x10, 0x33, 0x5f, 0x18, 0x54, 0x18, 0xc3, 0xea,
0x81, 0x90, 0xf3, 0xc0, 0x85, 0x5d, 0x33, 0xc9,
0x0a, 0x52, 0xc0, 0x5d, 0x2e, 0x39, 0x3d, 0xeb,
0xad, 0x18, 0x80, 0x4b, 0xa3, 0xb1, 0xbd, 0x4c,
0xb3, 0xc1, 0xf8, 0x41, 0x7e, 0xd8, 0x79, 0xcb,
0xcc, 0x54, 0xbf, 0x40, 0xcd, 0x67, 0xc7, 0x9c,
0x3c, 0xdf, 0x64, 0xb7, 0xae, 0x82, 0xb8, 0x8e,
0x1a, 0x42, 0x49, 0x9c, 0x9c, 0x49, 0x2b, 0xec,
0xd9, 0x72, 0xe5, 0xf2, 0x05, 0x8b, 0x1c, 0xff,
0xe6, 0x20, 0x39, 0x38, 0x07, 0x35, 0x4a, 0x36,
0x4c, 0x22, 0x6f, 0x   , 0x   , 0x   , 0x   , 0x   ,
```

Modified

```
0xcf, 0xec, 0x45, 0x4a, 0xfd, 0x35, 0x35, 0x61,
0xbc, 0x9f, 0xb4, 0xe4, 0x84, 0xf7, 0x5c, 0xa2,
0x10, 0x33, 0x5f, 0x18, 0x54, 0x18, 0xc3, 0xea,
0x81, 0x90, 0xf3, 0xc0, 0x85, 0x5d, 0x33, 0xc9,
0x0a, 0x52, 0xc0, 0x5d, 0x2e, 0x39, 0x3d, 0xeb,
0xad, 0x18, 0x80, 0x4b, 0xa3, 0xb1, 0xbd, 0x4c,
0xb3, 0xc1, 0xf8, 0x41, 0x7e, 0xd8, 0x79, 0xcb,
0xcc, 0x54, 0xbf, 0x40, 0xcd, 0x67, 0xc7, 0x9c,
0x3c, 0xdf, 0x64, 0xb7, 0xae, 0x82, 0xb8, 0x8e,
0x1a, 0x42, 0x49, 0x9c, 0x9c, 0x49, 0x2b, 0xec,
0xd9, 0x72, 0xe5, 0xf2, 0x05, 0x8b, 0x1c, 0xff,
0xe6, 0x20, 0x39, 0x38, 0x07, 0x35, 0x4a, 0x36,
0x4c, 0x22, 0x6f,
```

6. This final .hex file is used in Step 4 to build the Linux kernel.

6.2.5 Step 4: Build Linux Kernel

1. Clone the Linux kernel.

```
$git clone https://github.com/nxp-imx/linux-imx.git
$cd linux-imx
```

2. Set the environment variable KERNELDIR. This is required in subsequent steps.

```
$export KERNELDIR=`pwd`
```

3. Generate the kernel config file. Use the correct defconfig as per the target system.

```
$make imx_v7_defconfig
```

4. Build the kernel

```
$make -j8 zImage
```

This step will take time as this builds the full kernel. In case default regulatory support (without Murata module specific support) is needed, the build kernel can be used directly, and the following steps will not be required. However, to use Murata specific regulatory support, as well as user modified regulatory support, the rest of the steps needs to be followed to create the custom Linux kernel image.

5. Copy the created/downloaded HEX file to the kernel directory net/wireless/certs/.

```
$cp <WIRELESS_REGDB_DIR>/murata.hex <KERNEL_DIR>/net/wireless/certs/
```

In case the user plans to use unmodified Murata-only regulatory support and has skipped the previous steps, the HEX file can be downloaded from Murata GitHub.

- Go to Murata GitHub page for regulatory solution.
- Select the folder for the Murata module to be used.
- Download the murata.hex file.

6. Rebuild the kernel.

```
cd <KERNEL_DIR>
make -j8 zImage dtbs modules
```

7. This will update the affected kernel files and generate the customized kernel image, as shown in **Figure 23**.

Figure 23: Rebuilding Customized Kernel

```
test@ubuntu:~/projects/tb9104-new-db-solution/6.6.3/linux-imx$  make -j8 zImage
  CALL    scripts/checksyscalls.sh
  GEN     net/wireless/shipped-certs.c
  CC      net/wireless/shipped-certs.o
  AR      net/wireless/built-in.a
  AR      net/built-in.a
  AR      built-in.a
  AR      vmlinux.a
  LD      vmlinux.o
  OBJCOPY modules.builtin.modinfo
  GEN     modules.builtin
  GEN     .vmlinux.objs
  MODPOST vmlinux.symvers
  UPD     include/generated/utsversion.h
  CC      init/version-timestamp.o
  LD      .tmp_vmlinux.kallsyms1
  NM      .tmp_vmlinux.kallsyms1.syms
  KSYMS   .tmp_vmlinux.kallsyms1.S
  AS      .tmp_vmlinux.kallsyms1.S
  LD      .tmp_vmlinux.kallsyms2
  NM      .tmp_vmlinux.kallsyms2.syms
  KSYMS   .tmp_vmlinux.kallsyms2.S
  AS      .tmp_vmlinux.kallsyms2.S
  LD      vmlinux
  NM      System.map
  SORTTAB vmlinux
  OBJCOPY arch/arm/boot/Image
  Kernel: arch/arm/boot/Image is ready
  GZIP    arch/arm/boot/compressed/piggy_data
  AS      arch/arm/boot/compressed/piggy.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
```

8. Rebuild the affected modules.

```
$mkdir ../tempModules
$make modules_install INSTALL_MOD_PATH=../tempModules/
$cd ../tempModules/lib/modules

# Change the kernel module folder name in the below command as per the
setup
$tar -czvf 6.6.3-gccf0a99701a7.tar.gz 6.6.3-gccf0a99701a7
```

9. Rebuild the NXP Wireless driver against the new kernel, as shown in **Figure 24**.

```
$git clone http://github.com/nxp-imx/mwifiex.git
$cd mwifiex/mxm_wifiex/wlan_src
$make clean
$make build
```

Figure 24: Rebuilding NXP WLAN Drivers

6.2.6 Step 5: Install Regulatory Files

1. Flash the target board with the NXP demo image, following the procedures detailed in Section 4.
2. Copy the customized kernel to the target platform.
   - For 6UL/6ULL, copy the generated zImage to the /boot/ partition of the target board SD card.

     ```
     $ sudo cp zImage /media/<sd_card_name>/boot
     ```

   - For 8-series platforms, mount the boot partition and copy the generated zImage there.

     ```
     $ sudo scp Image /run/media/<bootm2clp1>
     ```

3. Copy the modified NXP WLAN drivers to a temporary folder on the target board.

   ```
   $ sudo cp mlan.ko /media/<sd_card_name>/root/home/root

   $ sudo cp moal.ko /media/<sd_card_name>/root/home/root
   ```

4. Copy the kernel modules archive to the /lib/modules/ folder on the target board.

   ```
   $ sudo cp *.tar.gz /media/<sd_card_name>/root/lib/modules
   ```

5. Download and copy the correct Murata module folder from Murata GitHub repo nxp-linux-calibration to the lib/firmware/nxp/ folder on the target board.

   ```
   sudo cp -Rfp murata/ /media/<sd_card_name>/root/lib/firmware/nxp
   ```

6. Copy the regulatory.db and regulatory.db.p7s files generated in Step 1 to this folder, overwriting the default ones.

   ```
   $scp <WIRELESS_REGDB_DIR>/regulatory.db
   /media/<sd_card_name>/root/lib/firmware/nxp/murata/

   $scp <WIRELESS_REGDB_DIR>/regulatory.db
   /media/<sd_card_name>/root/lib/firmware/nxp/murata/
   ```

7. Untar the module archive on the target.

   ```
   cd /media/<sd_card_name>/root/lib/modules
   sudo tar -xvf 6.6.3-gccf0a99701a7.tar.gz
   ```

6.2.7 Step 6: Test the Solution

1. Reboot the board and load the NXP WLAN drivers.

```
$ cd /home/root
$ insmod ./mlan.ko
$ insmod insmod ./moal.ko fw_name=nxp/<module_FW_name> cal_data_cfg=none
host_mlme=1 cfg80211_wext=0xf ps_mode=2 auto_ds=2
txpwrlimit_cfg=nxp/txpower_US.bin

# For example, command for loading 1XK module
# $ insmod ./mlan.ko
# $ insmod ./moal.ko fw_name=nxp/sdiouartiw416_combo_v0.bin
# cal_data_cfg=none host_mlme=1 cfg80211_wext=0xf ps_mode=2 auto_ds=2
# txpwrlimit_cfg=nxp/txpower_US.bin

# For example, command for loading 2DL module
# $ insmod ./mlan.ko
# insmod ./moal.ko fw_name=nxp/sduart_nw61x_v1.bin.se cal_data_cfg=none
# host_mlme=1 cfg80211_wext=0xf ps_mode=2 auto_ds=2
# init_hostcmd_cfg=nxp/rutxpower_US.bin txpwrlimit_cfg=nxp/txpower_US.bin


[  32.853955] wlan: Loading MWLAN driver
[  32.858084] wlan: Register to Bus Driver...
[  32.862534] vendor=0x02DF device=0x9159 class=0 function=1
[  32.868352] Attach moal handle ops, card interface type: 0x108
[  32.874377] rps set to 0 from module param
[  32.878499] No module param cfg file specified
[  32.882962] SDIO: max_segs=128 max_seg_size=65535
[  32.887681] rx_work=1 cpu_num=2
[  32.890958] Enable moal_recv_amsdu_packet
[  32.895027] Attach mlan adapter operations.card_type is 0x108.
[  32.901258] wlan: Enable TX SG mode
[  32.904749] wlan: Enable RX SG mode
[  32.909147] Request firmware: nxp/sdiouartiw416_combo_v0.bin
[  33.169202] Wlan: FW download over, firmwarelen=586856 downloaded
578116
[  34.569073] WLAN FW is active
[  34.572052] on_time is 34379061723
[  34.575474] Download txpwrlimit_cfg=nxp/txpower_US.bin
[  34.612655] VDLL image: len=8740
[  34.625754] FW country code WW does not match with US
[  34.631045] fw_cap_info=0x187ccf03, dev_cap_mask=0xffffffff
[  34.636643] max_p2p_conn = 8, max_sta_conn = 8
[  34.656489] Register NXP 802.11 Adapter mlan0
[  34.667097] Register NXP 802.11 Adapter uap0
[  34.683776] Register NXP 802.11 Adapter wfd0
[  34.688228] wlan: version = SDIW416---16.92.21.p119.11-MM6X16437.p21-
GPL-(FP92)
[  34.696949] wlan: Register to Bus Driver Done
[  34.706596] wlan: Driver loaded successfully
```

2. Check the current regulatory setting.

```
$ iw reg get
global
country 00: DFS-UNSET
        (755 - 928 @ 2), (N/A, 20), (N/A), PASSIVE-SCAN
        (2402 - 2472 @ 40), (N/A, 20), (N/A)
        (2457 - 2482 @ 20), (N/A, 20), (N/A), AUTO-BW, PASSIVE-SCAN
        (2474 - 2494 @ 20), (N/A, 20), (N/A), NO-OFDM, PASSIVE-SCAN
```

```
        (5170 - 5250 @ 80), (N/A, 20), (N/A), AUTO-BW, PASSIVE-SCAN
        (5250 - 5330 @ 80), (N/A, 20), (0 ms), DFS, AUTO-BW, PASSIVE-SCAN
        (5490 - 5730 @ 80), (N/A, 20), (0 ms), DFS, AUTO-BW, PASSIVE-SCAN
        (5735 - 5835 @ 80), (N/A, 20), (N/A), AUTO-BW, PASSIVE-SCAN
        (57240 - 63720 @ 2160), (N/A, 0), (N/A)
```

3. Change the regulatory setting

```
$ iw reg set DE
```

4. Check that the regulatory setting has been updated.

```
$ iw reg get
global
country DE: DFS-ETSI
        (2400 - 2483 @ 40), (N/A, 20), (N/A)
        (5170 - 5250 @ 80), (N/A, 23), (N/A), NO-OUTDOOR, AUTO-BW
        (5250 - 5330 @ 80), (N/A, 20), (0 ms), NO-OUTDOOR, DFS, AUTO-BW
        (5490 - 5710 @ 80), (N/A, 26), (0 ms), DFS, AUTO-BW
        (5725 - 5835 @ 80), (N/A, 13), (N/A), AUTO-BW
        (5945 - 6425 @ 160), (N/A, 23), (N/A), NO-OUTDOOR
        (57000 - 66000 @ 2160), (N/A, 40), (N/A)
```

5. Do similar tests for other supported country codes as required.

```
$ iw reg set JP
$ iw reg get
global
country JP: DFS-JP
        (2402 - 2482 @ 40), (N/A, 20), (N/A)
        (2474 - 2494 @ 20), (N/A, 20), (N/A), NO-OFDM
        (4910 - 4990 @ 40), (N/A, 23), (N/A)
        (5170 - 5250 @ 80), (N/A, 20), (N/A), NO-OUTDOOR, AUTO-BW
        (5250 - 5330 @ 80), (N/A, 20), (0 ms), NO-OUTDOOR, DFS, AUTO-BW
        (5490 - 5730 @ 80), (N/A, 23), (0 ms), DFS, AUTO-BW
        (5925 - 6425 @ 160), (N/A, 23), (N/A), NO-OUTDOOR
        (57000 - 66000 @ 2160), (N/A, 10), (N/A)

$ iw reg set US
$ iw reg get
global
country US: DFS-FCC
        (902 - 904 @ 2), (N/A, 30), (N/A)
        (904 - 920 @ 16), (N/A, 30), (N/A)
        (920 - 928 @ 8), (N/A, 30), (N/A)
        (2400 - 2472 @ 40), (N/A, 30), (N/A)
        (5170 - 5250 @ 80), (N/A, 23), (N/A), AUTO-BW
        (5250 - 5330 @ 80), (N/A, 24), (0 ms), DFS, AUTO-BW
        (5490 - 5730 @ 80), (N/A, 24), (0 ms), DFS, AUTO-BW
        (5735 - 5835 @ 80), (N/A, 30), (N/A), AUTO-BW
        (5850 - 5895 @ 40), (N/A, 27), (N/A), NO-OUTDOOR, AUTO-BW,
PASSIVE-SCAN
        (5925 - 7125 @ 320), (N/A, 12), (N/A), NO-OUTDOOR, PASSIVE-SCAN
        (57240 - 71000 @ 2160), (N/A, 40), (N/A)
```

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

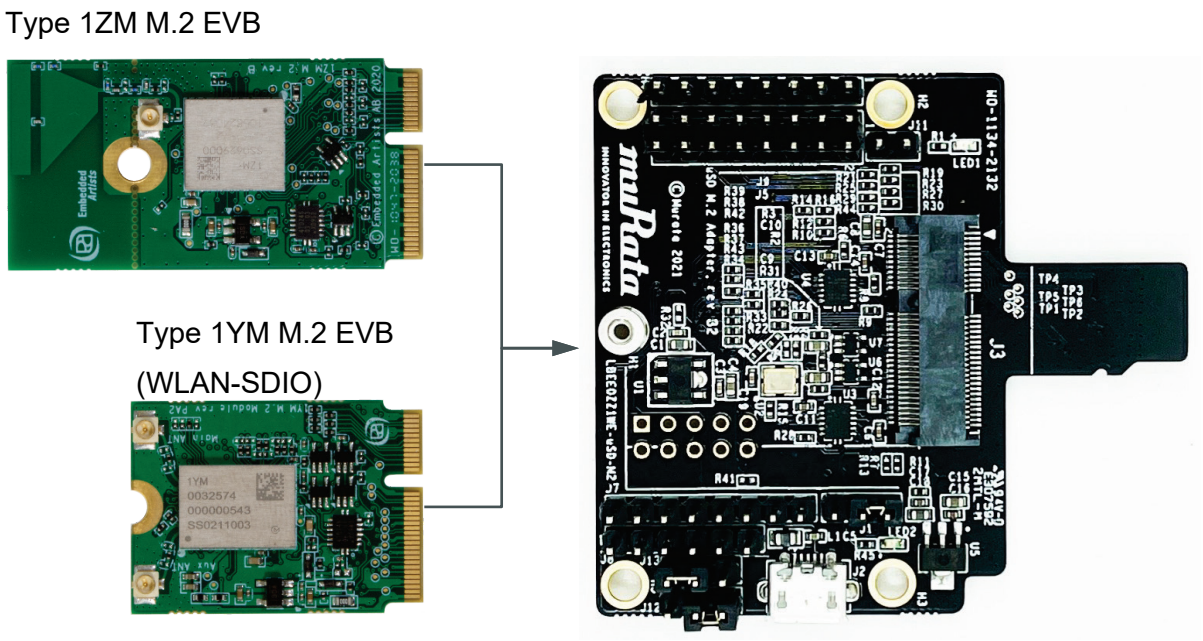
Embedded Artists' Wi-Fi/BT M.2 EVBs based on SDIO, listed in **Table 3** can be connected to the i.MX 6UL(L) EVKs through Murata's uSD-M.2 Adapter as shown in **Figure 25**. The following sub-sections details steps for bringing up Embedded Artists' Wi-Fi/BT EVBs on these EVKs.

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/C 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/C 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 25: uSD-M.2 Adapter with Type 1ZM and 1YM-SDIO* M.2 EVB Options



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

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/C 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 9.1. Connect the uSD-SD Card adapter and tape the uSD Adapter-SD Card per Section 9.3.
4. Connect ribbon cable at both ends before inserting Murata EVK into SD1 slot. Note the orientation as shown in **Figure 26**. 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 9.3.
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 4.1.3 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
```

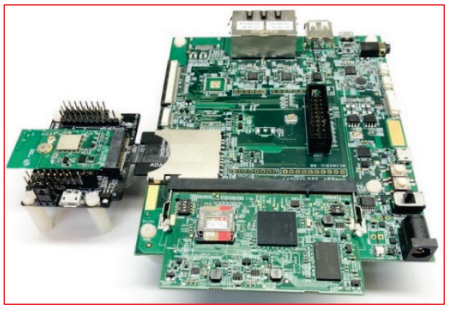
8. Load Drivers – Every time:

   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 8 to test/verify Wi-Fi and Bluetooth functionality.

Figure 26: i.MX 6ULL EVK with uSD-M.2 Adapter and Type 1ZM M.2 EVB



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

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

The NXP i.MX 8MQuad EVK (see **Figure 27**) 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 EVBs are Type 1YM, 1XL and 2XS (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 1YM/1XL/2XS M.2 EVB to M.2 connector as shown in **Figure 27**. 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 8**. 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
```

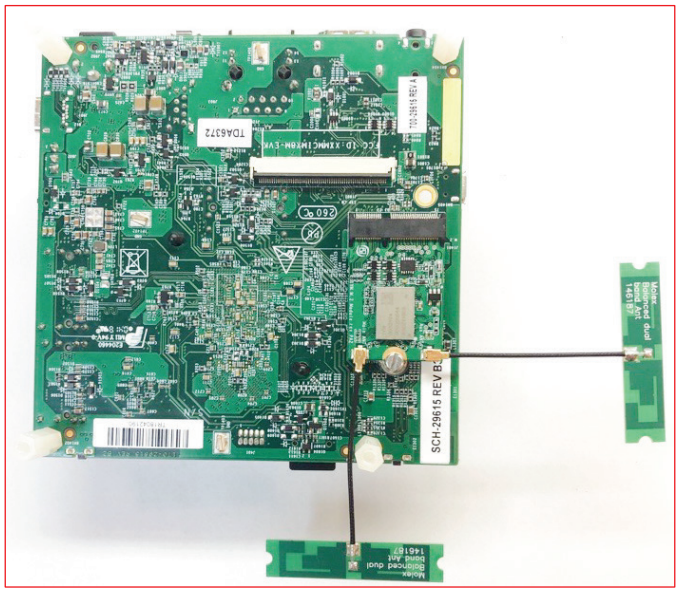
4. Load Drivers – Every time:

   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 8 to test/verify Wi-Fi and Bluetooth functionality.

Figure 27: i.MX 8MQuad with Type 1YM (Bottom View)



8.2 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 28** for upside-down EVK view showing M.2 connector (currently the only supported M.2 EVBs are Type 1YM, 1XL and 2XS). 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/1XL/2XS M.2 EVB to M.2 connector as shown in **Figure 28**. 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 8**. 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
```

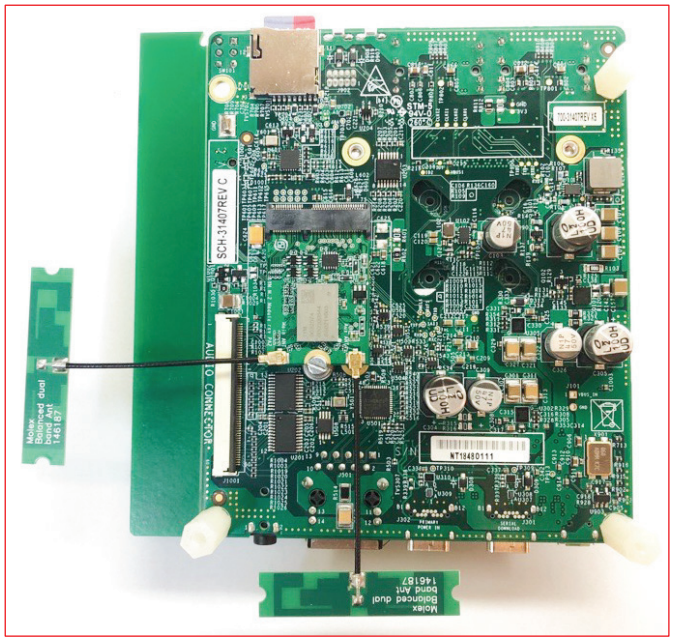
4. Load Drivers – Every time:

   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 8 to test/verify Wi-Fi and Bluetooth functionality.

Figure 28: i.MX 8M Mini with Type 1YM-PCIe (Bottom View)



8.3 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 29**)
- WLAN/Bluetooth (**Figure 30)**.

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

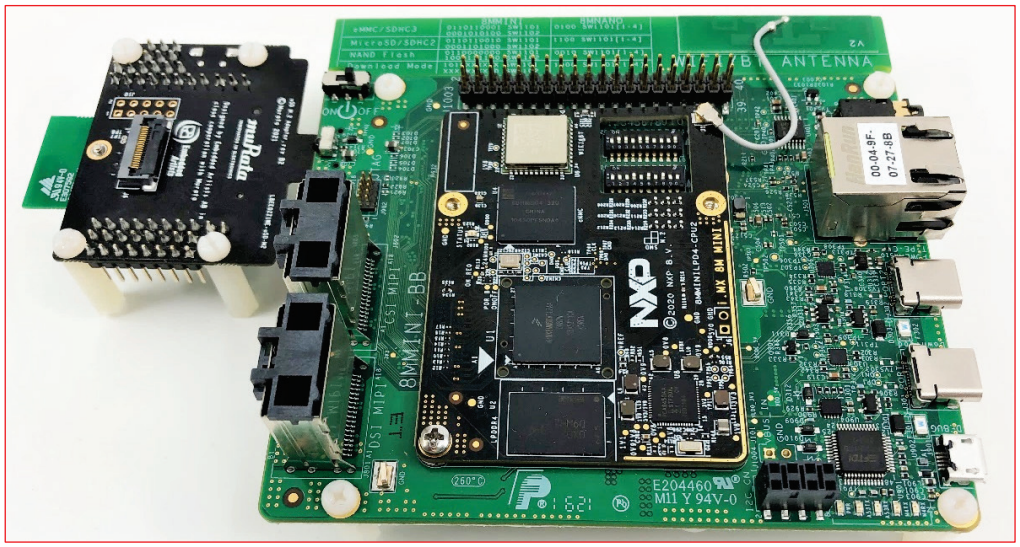
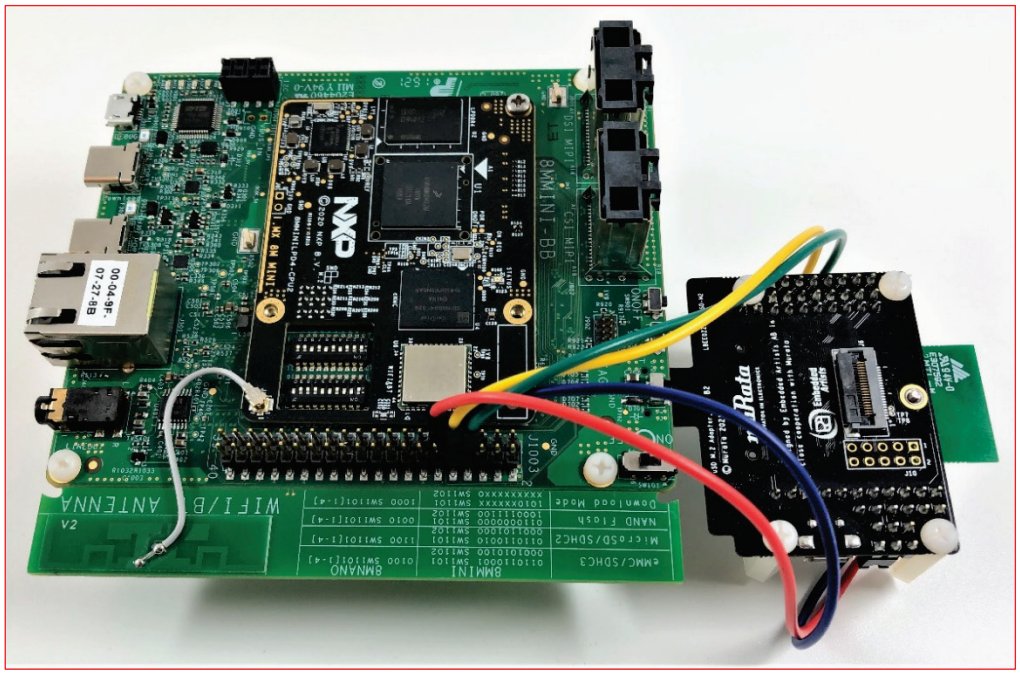


Figure 30: i.MX 8M Mini/Nano EVK with uSD-M.2 Adapter (WLAN/Bluetooth)



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/C 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, 1XK, 2XK, 2DS, 2DL, 2EL or 1YM, 1XL, 2XS (reworked for WLAN-SDIO operation as per Section 3.4.5) and Section 3.4.6) Wi-Fi/BT M.2 EVB to the M.2 connector on the uSD-M.2 Adapter per Section 9.1. Attach the uSD-M.2 Adapter/M.2 to EVK and tape the uSD Adapter-EVK connection per Section 9.3.
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 15**. Refer to **Figure 31**, **Figure 32**, **Figure 33** and **Figure 34** 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 35**, and **Figure 36**. 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 37**, 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 29**), 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 8**. 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 15: 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 8 to test/verify Wi-Fi and Bluetooth functionality.

Figure 31: Cable Connections on i.MX 8M Mini EVK (Even Number Connector View)

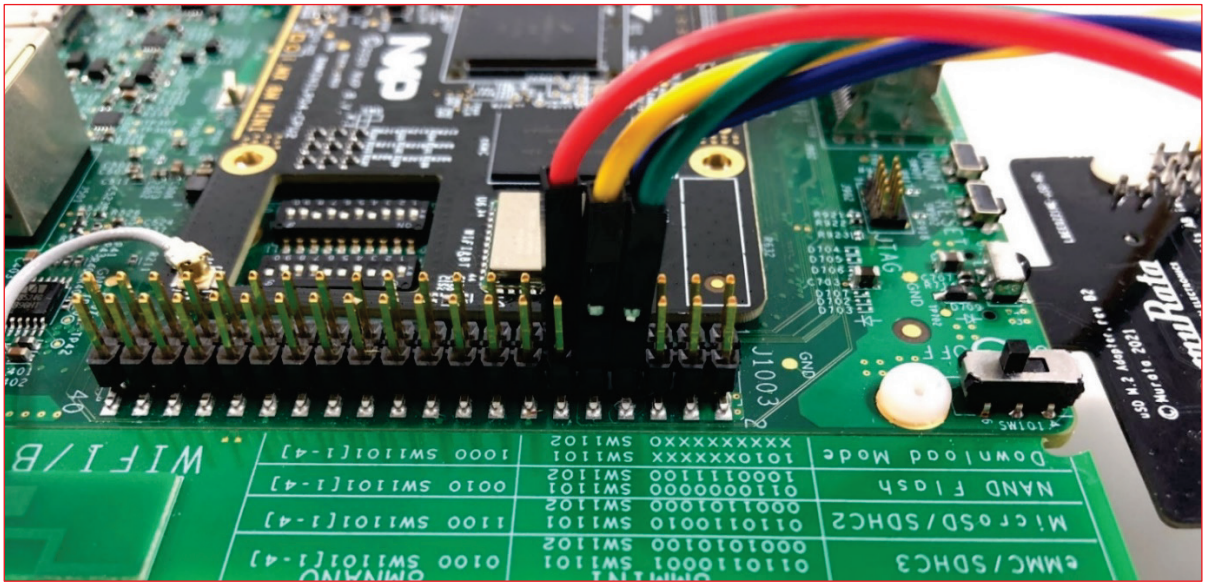


Figure 32: Cable Connections on i.MX 8M Mini EVK (Odd Number Connector View)

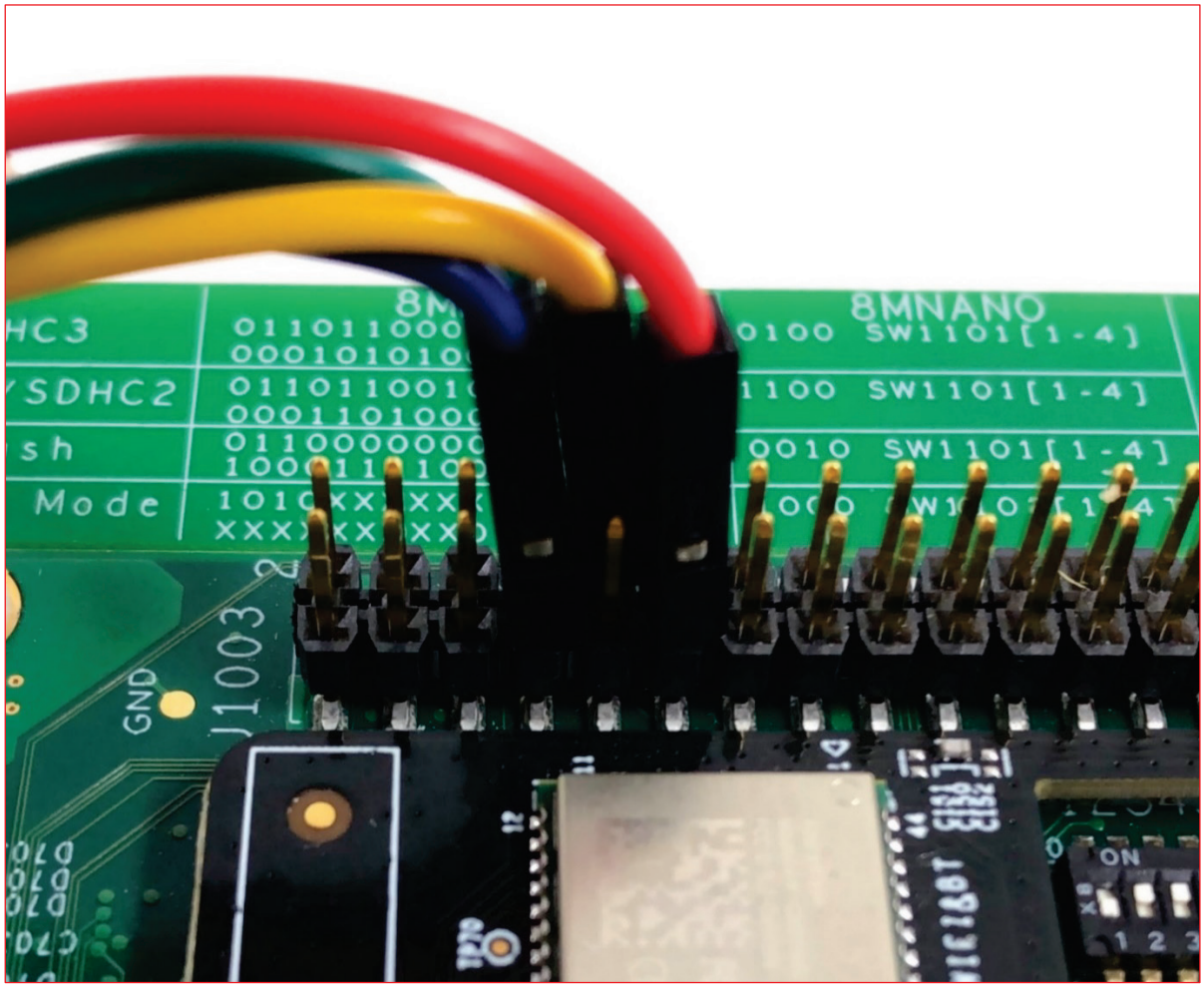


Figure 33: Cable Connections on uSD-M.2 Adapter (J9 Header)

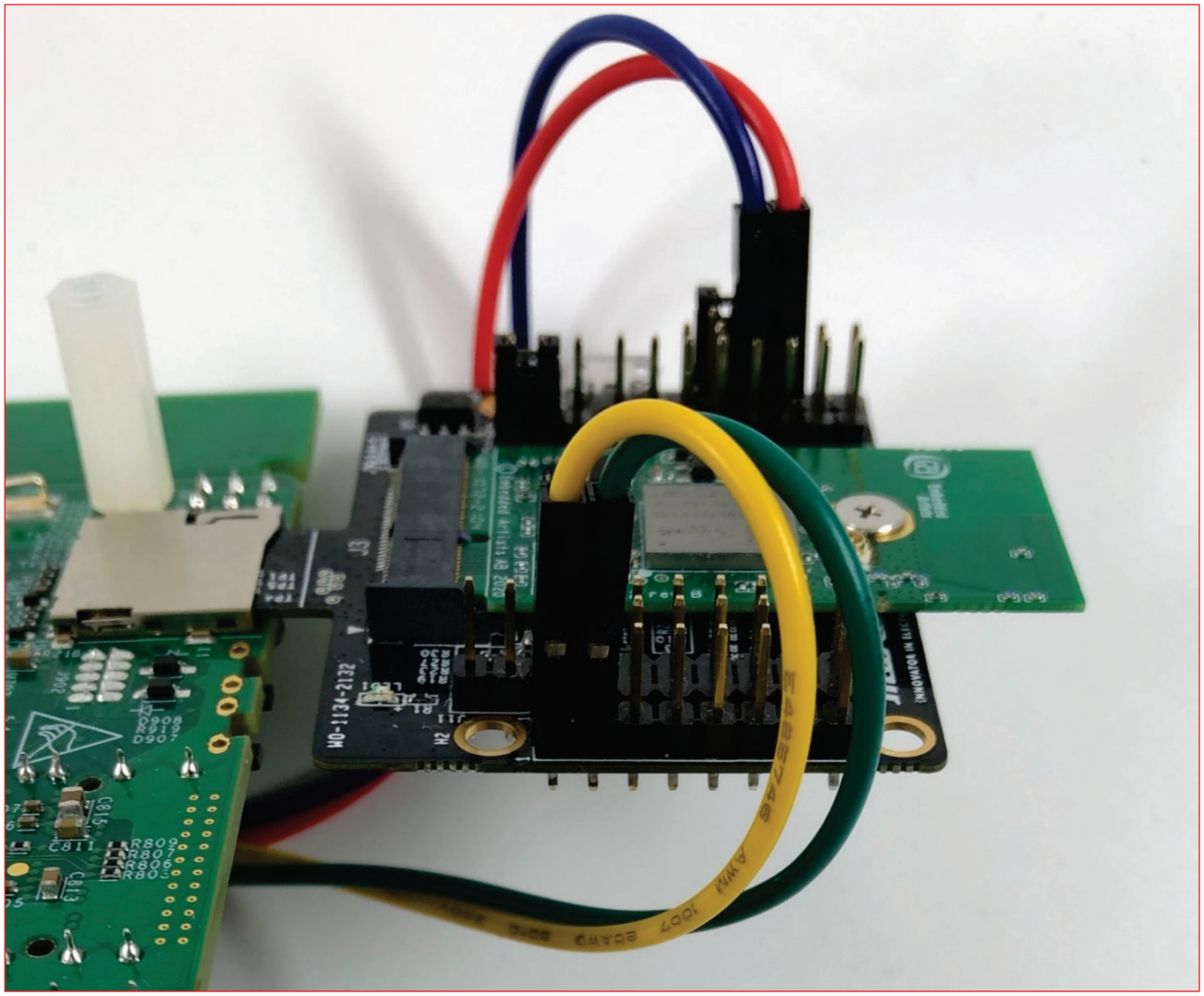


Figure 34: Cable connections on uSD-M.2 Adapter (J8 Header)

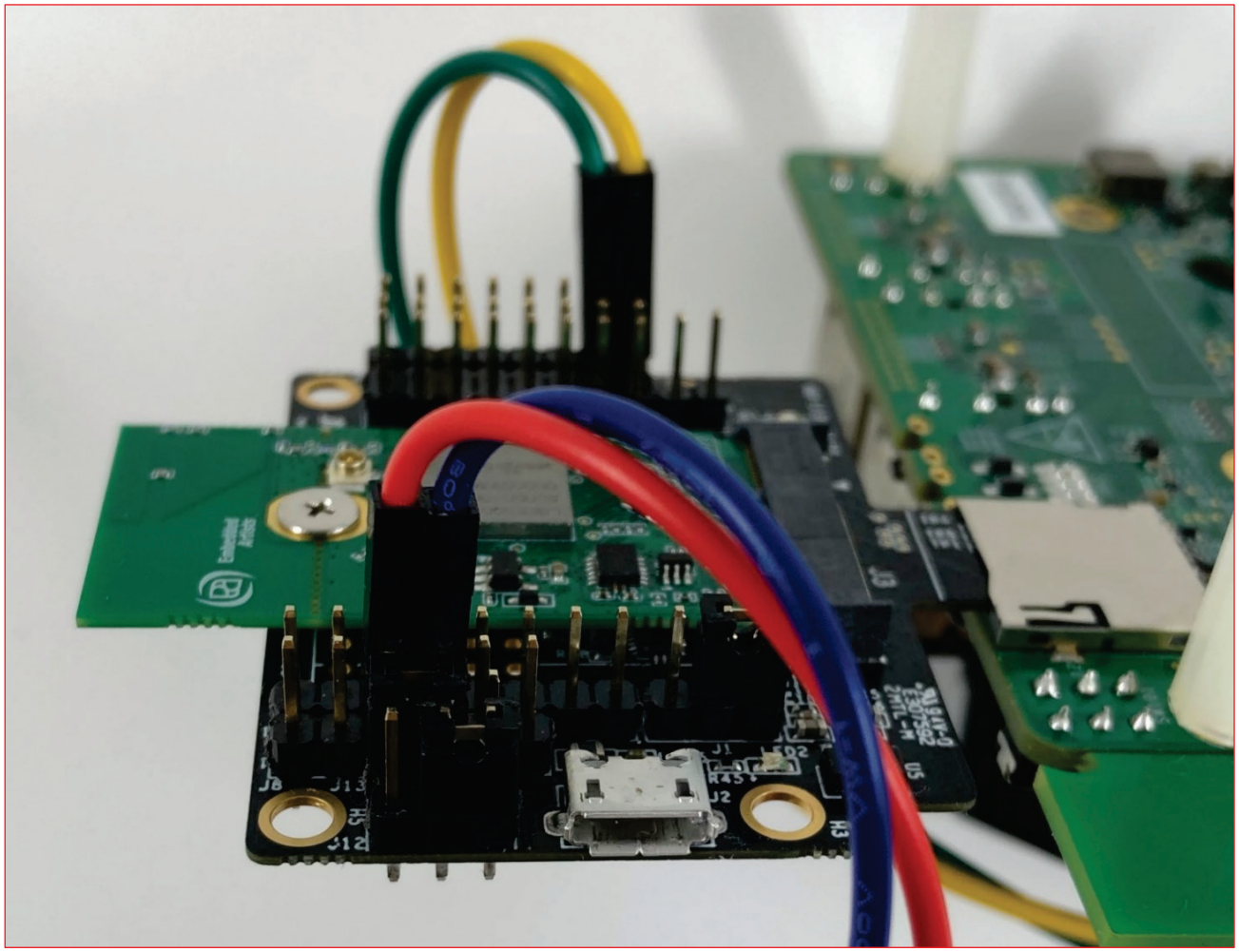


Figure 35: Low-Profile Jumper Wires (Digi-Key part number 1988-1178-ND)

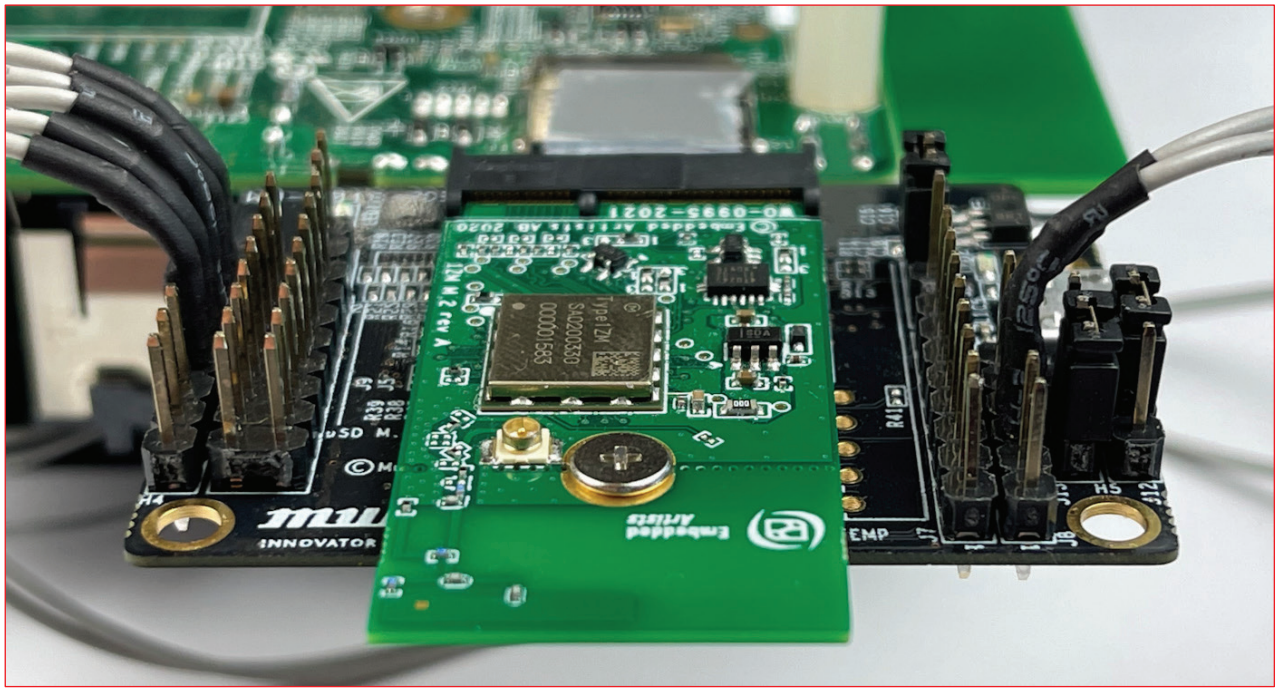


Figure 36: NXP i.MX EVK with Low-Profile Jumper Wires

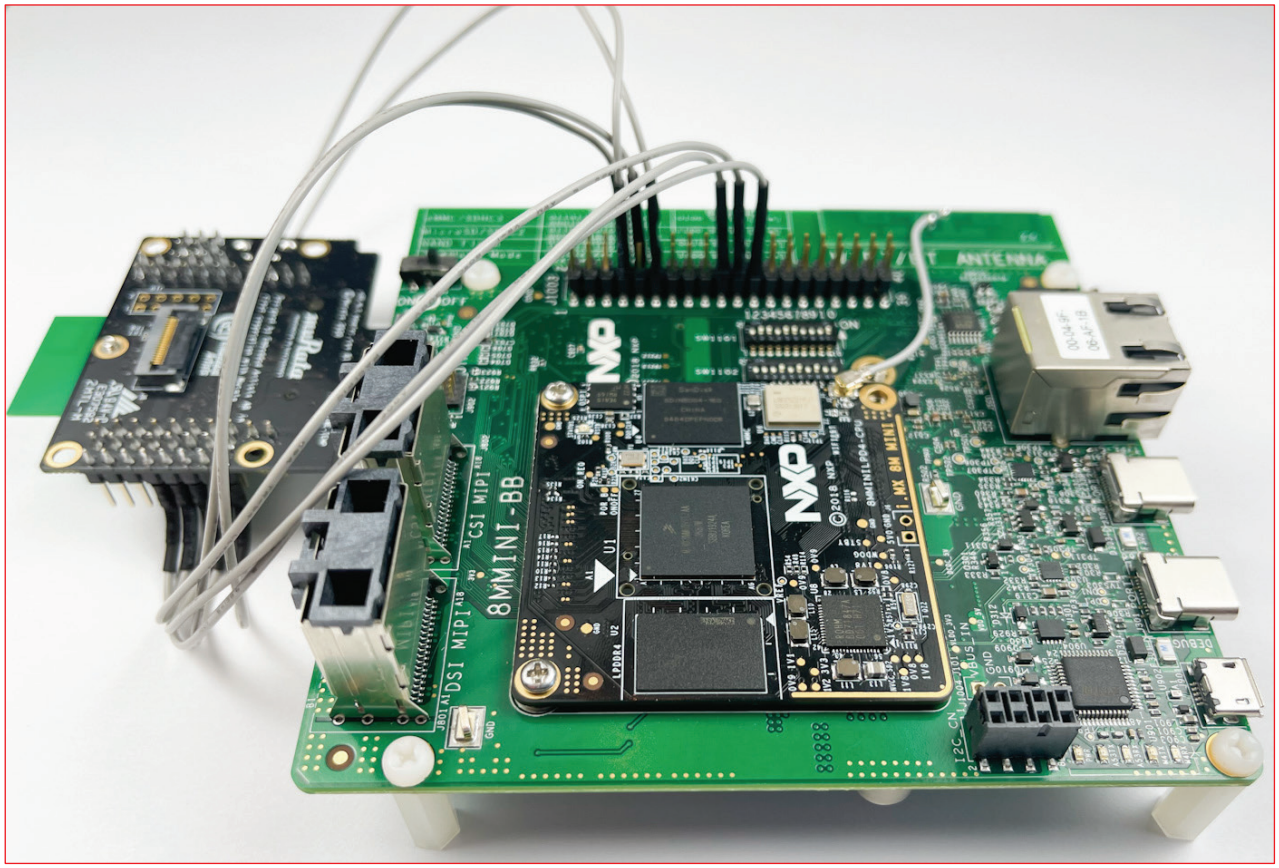
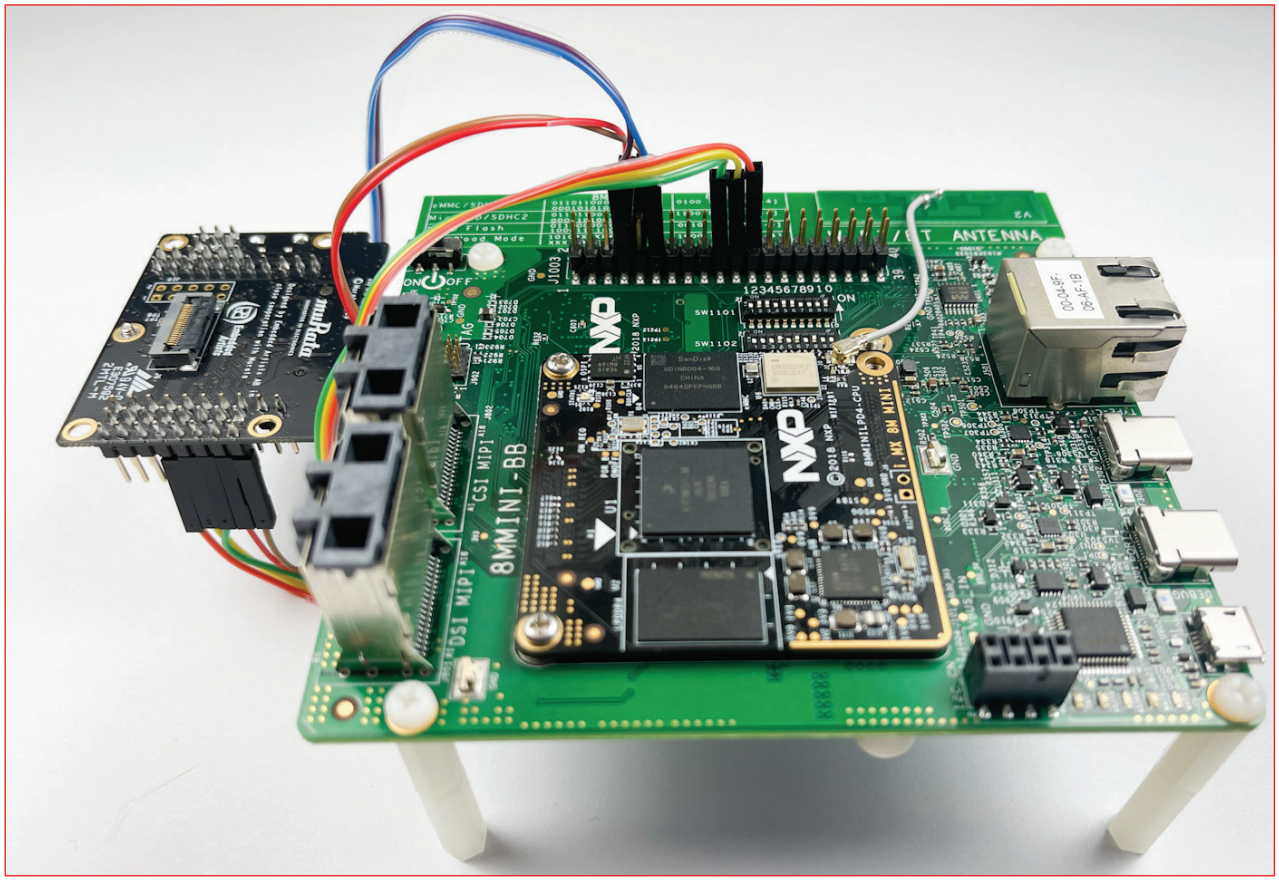


Figure 37: Additional Hex Standoff (Digi-Key Part Number RPC3570-ND)



9 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 16**.

Table 16: Embedded Wi-Fi/Bluetooth Files

Filename or Folder	Details
/usr/share/nxp_wireless/mlanutl	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.

9.1 Wi-Fi Interface Test/Verification

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

9.1.1 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
```

9.1.2 Bringing Up Wi-Fi Interface

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

9.1.2.1 For i.MX6UL(L)

1. Ensure that correct DTB file is set for 6UL(L) as mentioned in Section 6.1.
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/sdiouart8987_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/sdiouart8987_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.
- "sdiouart8987_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 mlan0 up" command to initialize the "mlan0" (WLAN) interface.

```
$ ifconfig mlan0 up
$ ifconfig mlan0

# “mlan0” interface is UP. WLAN MAC Address is shown
mlan0: 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 “mlan1” interface. That will be done later in Section 8.1.6.

9.1.2.2 For i.MX 8M-Mini/Nano

1. Ensure that correct DTB file is set for 8M-Mini/Nano as mentioned in Section 7.2/ Section 7.3.
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] mlan: 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 mlan 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 mlan1 up” command to initialize the “mlan1” (WLAN) interface.

```
$ ifconfig mlan1 up
$ ifconfig mlan1
# “mlan1” interface is UP. WLAN MAC Address 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
```

There are 2 mlan interfaces, “mlan0” and “mlan1” when you issue the command “ifconfig -a”. For Murata modules, “mlan1” interface must be used for Wi-Fi/BT bring-up. This is applicable only for i.MX 8M-Mini/8M-Nano.

```
$ ifconfig -a
mlan0: 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
```

9.1.3 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-ffb1-0ad1-63f3-9b91a979382c
                 * Manufacturer: NETGEAR, Inc.
                 * Model: R6300
                 * Model Number: R6300
                 * Serial Number: 4536
                 * Primary Device Type: 6-0050f204-1
                 * Device name: R6300
                 * Config methods: Display
                 * RF Bands: 0x3
                 * Unknown TLV (0x1049, 6 bytes): 00 37 2a 00 01 20
        WMM:     * Parameter version 1
                 * u-APSD
                 * BE: CW 15-1023, AIFSN 3
                 * BK: CW 15-1023, AIFSN 7
                 * VI: CW 7-15, AIFSN 2, TXOP 6016 usec
                 * VO: CW 3-7, AIFSN 2, TXOP 3264 usec
# … More SSID listings follow here.
```

9.1.4 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 8.1.6 where connectivity testing is performed.

To test:

1. Following example with “Murata_5G” SSID, now invoke “iw” connect command:

```
$ iw dev mlan0 connect Murata_5G
```

2. Check status of connection with “iw” link command:

```
$ iw dev mlan0 link
Connected to 60:38:e0:9a:a3:9e (on mlan0)
        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
```

9.1.5 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.

9.1.5.1 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.

9.1.5.2 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 8.1.5.1) 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 mlan0 -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): mlan0: link becomes ready
```

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

```
$ iw dev mlan0 link
Connected to 84:1b:5e:f6:a7:60 (on mlan0)
        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
```

9.1.6 STA/Client Mode: Basic WLAN Connectivity Testing

Prior to running connectivity tests, we need to assign an IP address to the “mlan0” interface. If the subnet address is known, one option is to use manual “ifconfig” command to assign an IP address to “mlan0”. Here is an example “ifconfig” command assuming subnet of 192.168.1.255:

```
$ ifconfig mlan0 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 mlan0
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.

9.1.7 AP Mode: Set Up Soft Access Point

In this section, the method for setting up a soft AP is presented. The uap0 interface is used with the hostapd tool to set up and start the soft AP.

Prior to running the “hostapd” tool, you might like to back up default/previous “/etc/hostapd.conf” file in case you overwrite it:

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

1. Create/modify the hostapd.conf file as per the AP requirement:

   ```
   $ vi /etc/hostapd.conf
   ```

   The hostapd.conf file contains the AP settings. Given below is an example conf file - change as per requirement. The ‘interface’ option must always be set to the uap interface on the system after driver load (uap0 in this case)

   ```
   # The interface must be set to the uap interface in the system
   interface=uap0

   # Set to ‘g’ for 2.4 GHz operation, set to ‘a’ for 5 GHz operation
   hw_mode=g

   # Channel number. 0 means not set, and the channel will be automatically
   set
   channel=0

   # Country code
   country_code=US

   # SSID name to be set up
   ssid=Test_SSID

   # Enable/disable 802.11n
   ieee80211n=1
   ```

   Save the file after modifications.

2. Create/modify the udhcpd.conf file to configure the DHCP server:

   ```
   $ vi /etc/udhcpd.conf
   ```

   The udhcpd.conf file contains the DHCP settings for the server. Given below is an example conf file - change as per requirement.

   ```
   start 192.168.1.10
   ```

```
end 192.168.1.20
interface uap0
```

3. Create udhcp client lease database file. The udhcp server uses this file to store all the assigned addresses. This file does not need to contain anything when created.

```
$ touch /var/lib/misc/udhcpd.leases
```

4. Clean up any existing instances.

```
$ killall wpa_supplicant
$ killall hostapd
$ ifconfig uap0 192.168.1.2 netmask 255.255.255.0 up
$ route add default gw 192.168.1.1
```

5. Start the DHCP server.

```
$ udhcpd /etc/udhcpd.conf
```

6. Start the Access Point.

```
$ hostapd /etc/hostapd.conf

rfkill: Cannot open RFKILL control device
[  599.640152] wlan: Starting AP
[  599.644153] fw doesn't support 11ax
[  599.658715] IPv6: ADDRCONF(NETDEV_CHANGE): uap0: link becomes ready
[  599.665565] wlan: AP started
[  599.669489] wlan: HostMlme uap0 send deauth/disassoc
[  599.674595] Set AC=3, txop=47 cwmin=3, cwmax=7 aifs=1
[  599.681875] Set AC=2, txop=94 cwmin=7, cwmax=15 aifs=1
[  599.689089] Set AC=0, txop=0 cwmin=15, cwmax=63 aifs=3
[  599.696295] Set AC=1, txop=0 cwmin=15, cwmax=1023 aifs=7
uap0: interface state UNINITIALIZED->ENABLED
uap0: AP-ENABLED
```

7. You can now connect to the SSID from another client and automatically receive an IP address. The console command output will display the interactions.

9.1.8 Wi-Fi Direct Testing

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

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 mlan0

# 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-mlan0-0: link is not ready
OK
<3>P2P-GROUP-STARTED p2p-mlan0-0IPv6: ADDRCONF(NETDEV_CHANGE): p2p-mlan0-
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.

9.2 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 (e.g., 1ZM or 1YM), you can verify the HCI UART connection by invoking “hciattach”, bringing up the interface with “hciconfig” and then invoking “hcitool scan” to see what Bluetooth devices are visible.

The Bluetooth test commands vary depending on which UART port is connected to Bluetooth. With the “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 on kernels 6.1.1 and below:

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

For kernel 6.1.36 and above, use the following command sequence:

```
modprobe btnxpuart
hciconfig hci0 up
hciconfig hci0 piscan
hciconfig hci0 noencrypt
hcitool scan
```

Table 17 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/ttymxc1 any 115200 flow
Setting TTY to N_HCI line discipline
Device setup complete
$ hciconfig hci0 up
$ hcitool -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/ttymxc1 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
$ hcitool scan
Scanning ...
        34:F3:9A:A6:00:53       SCOTTK-HPZBOOK
```

Table 17: 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.

9.3 802.15.4 Interface Test/Verification

Currently only Murata Type 2EL module provides SPI interface connectivity for 802.15.4. The steps below describe the process of bringing up the interface on NXP i.MX 8M Mini EVK, running kernel version 6.1.36.

Murata Rev C. version of uSD-M.2 adapter (2WE, 2WF) is required for 802.15.4 interface operation. Older revision (B1/B2/A) does not support 802.15.4 interface.

1. Download and flash the NXP Linux BSP release 6.1.36, as described in Section 4.
2. Download the required files from the Murata forum and unzip the archive.
3. Copy the extracted files ot-daemon, ot-ctl, fw_loader_imx_lnx and uartspi_n61x_v1.bin.se onto the i.MX platform, at location /home/root. Change the permission for ot-daemon, ot-ctl, and fw_loader_imx_lnx.

```
$cd /home/root/
$chmod a+x ot-daemon  ot-ctl fw_loader_imx_lnx
```

4. Copy the extracted dtb file, imx8mm-evk-iw612-evk.dtb onto the i.MX platform, at location "/run/media/ boot-mmcblk2p1/".
5. Reboot the EVK, interrupt boot and set the dtb file.

```
$setenv fdtfile imx8mm-evk-iw612-evk.dtb
$saveenv
$boot
```

6. Rework the 2EL M.2 EVB as per Section 3.4.8, if not done already. This is to enable the SPI interface to be detected properly by the NXP i.MX 8M Mini EVK.
7. Connect the Type 2EL M.2 EVB to the i.MX 8M Mini EVK using the Murata uSD-M.2 adapter (Rev C - 2WF). Install additional jumper connections between the uSD-M.2 adapter and the i.MX 8M Mini EVK as per **Table 18** below.
8. Turn on JP1 Pin 1 DIP switch, present in the back of the uSD-M.2 adapter, as shown in **Figure 38**. This sets up the muxed pins correctly.
9. Set the following jumpers in the uSD-M.2 adapter.
   - J13: Position 1-2. This sets the adapter to use target host VDD of 3.3V.
   - J12: Position 1-2. This sets the adapter to use M.2 module VDD of 1.8V.
   - J14: Position 1-2, as shown in **Figure 39**.
10. Issue the following command to download the firmware to the 2EL EVB.

```
$cd /home/root/
```

```
$./fw_loader_imx_lnx /dev/ttymxc2 115200 0 uartspi_n61x_v1.bin.se 3000000
```

11. To enable SPI buffers, write 1 to P0 of slave address 0x21. Start by issuing the following I2C detect command for detecting 0x21 slave address.

```
$i2cdetect 2
```

12. Write/Read Configuration register (P0) as output.

```
$i2cset 2 0x21 0x03 0xfe
$i2cget 2 0x21 0x03
```

13. Write/Read Output port register (P0) as high.

```
$i2cset 2 0x21 0x01 0x01
$i2cget 2 0x21 0x01
```

14. Run ot-daemon.

```
$./ot-daemon -v "spinel+spi:///dev/spidev1.0?gpio-reset-
device=/dev/gpiochip5&gpio-int-device=/dev/gpiochip5&gpio-int-
line=12&gpio-reset-line=13&spi-mode=0&spi-speed=1000000&spi-reset-
delay=500" -d 5  > ot.log 2>&1 &

sleep 2
```

15. Check the OpenThread version using the following command.

```
ot-ctl version
```

A sample console log output is provided in Appendix B.

Table 18: Murata uSD-M.2 Adapter Additional Connections for 2EL SPI Interfacing

uSD-M.2 Adapter Header/Pin	uSD-M.2 Adapter Pin Name	i.MX 8M Mini Header/Pin	i.MX 8M Mini Pin Name
J5 / Pin 2	I2C_SDA_HOST	40 Pin Header / Pin 3	I2C3_SDA_3V3
J5 / Pin 4	I2C_SCL_HOST	40 Pin Header / Pin 5	I2C3_SCL_3V3
J5 / Pin 6	SPI_INT	40 Pin Header / Pin 15	
J5 / Pin 8	SPI_SSEL	40 Pin Header / Pin 24	
J5 / Pin 10	SPI_TXD_HOST	40 Pin Header / Pin 19	IW61X_SPI_RXD [MOSI]
J5 / Pin 15	GND	40 Pin Header / Pin 39	
J7 / Pin 6	GND	J1003 / Pin 9	UART3_GND
J8 / Pin 3	UART_RTS_HOST	J1003 / Pin 11	UART3_RTS
J8 / Pin 4	BT_UART_CTS_HOST	J1003 / Pin 7	UART3_CTS
J9 / Pin 1	UART_TXD_HOST	J1003 / Pin 10	UART3_RXD
J9 / Pin 2	UART_RXD_HOST	J1003 / Pin 8	UART3_TXD
J9 / Pin 7	SPI_RXD_HOST [WL_WAKE_IN]	40 Pin Header / Pin 21	IW61X_SPI_TXD [MISO]
J9 / Pin 8	SPI_CLK [BT_WAKE_IN]	40 Pin Header / Pin 23	

Figure 38: Murata uSD-M.2 Adapter JP1 DIP Switch Setting

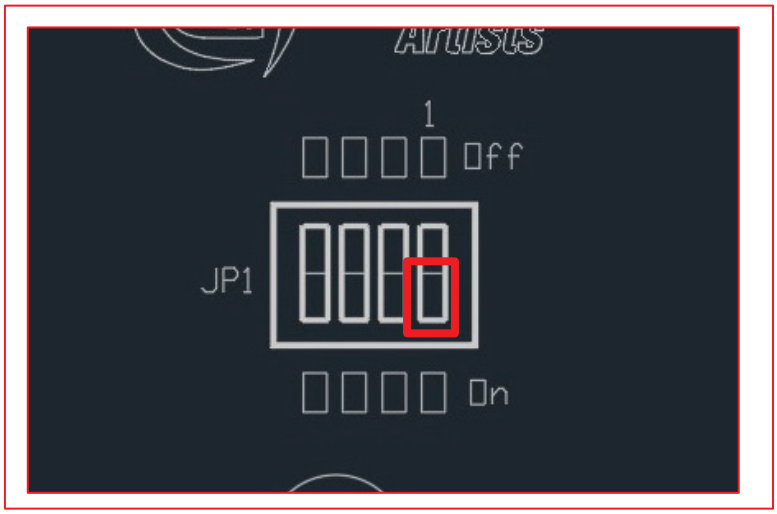
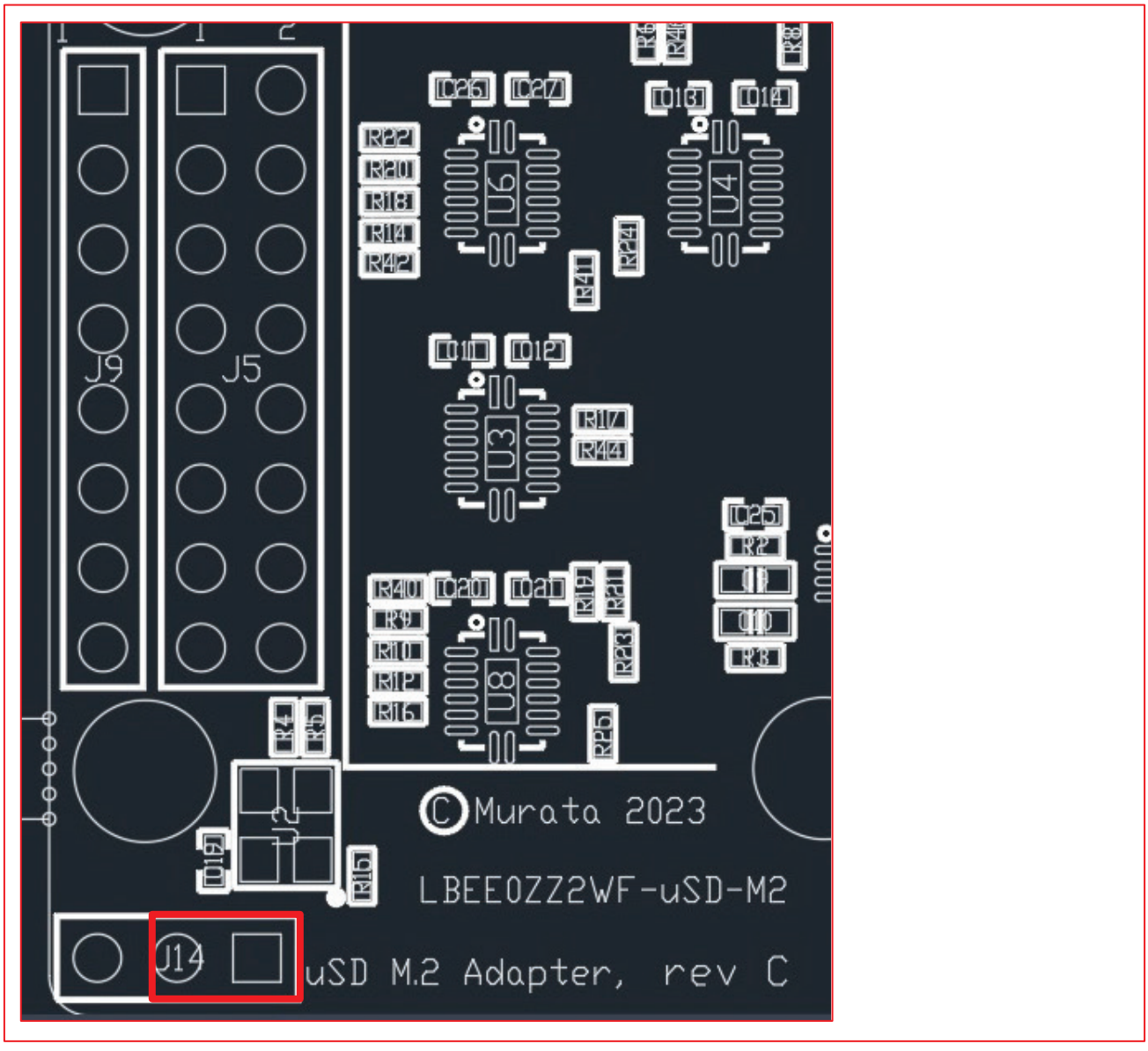


Figure 39: Murata uSD-M.2 Adapter J14 Jumper Setting for 2EL SPI Connection



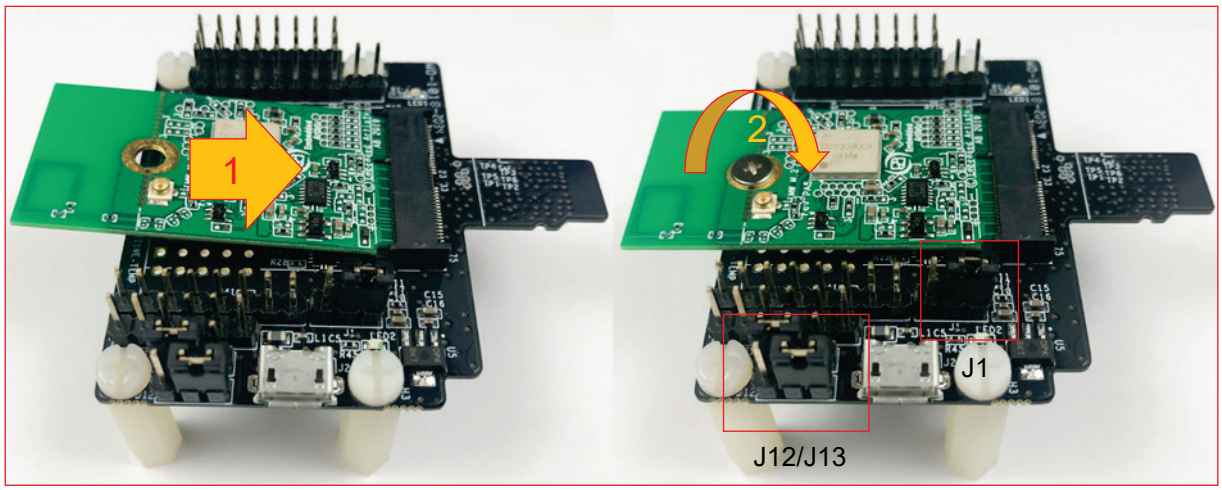
10 Murata's uSD-M.2 Adapter

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.

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

When connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter Rev B1/B2/C (**Figure 40**), 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 40: Connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter

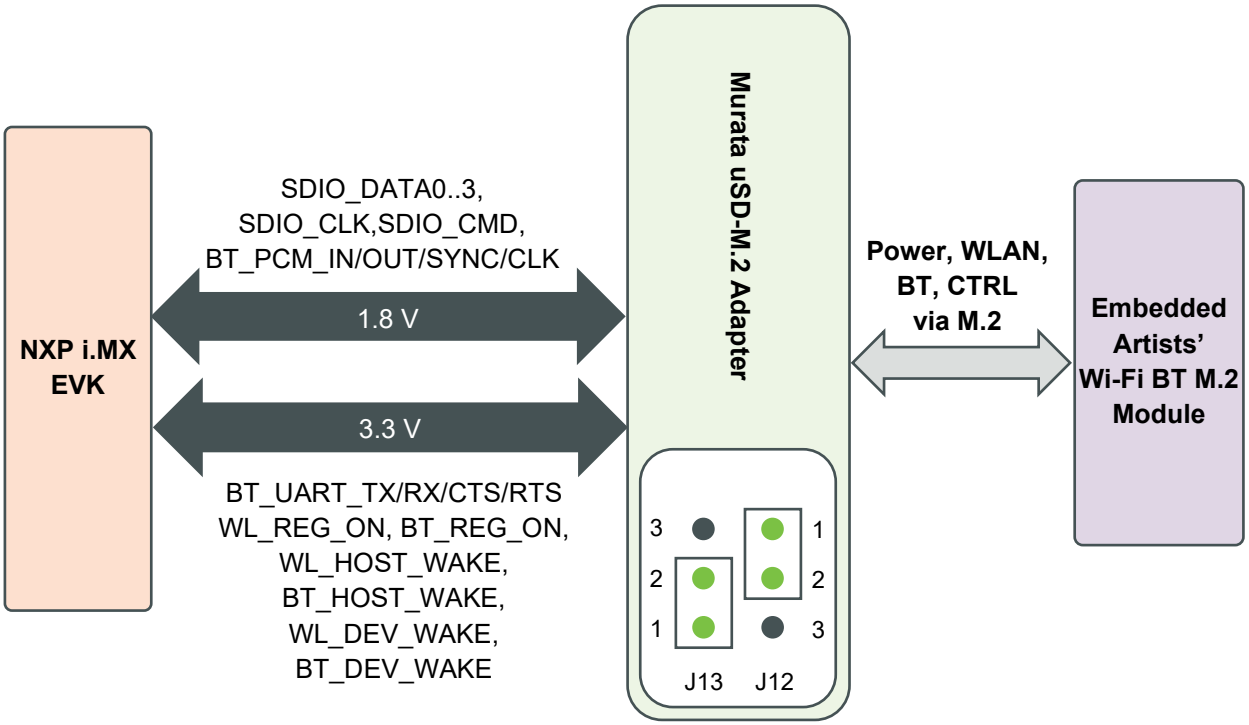


10.2 Configuring uSD-M.2 Adapter Jumpers for Correct VIO Signaling

Figure 41 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 41: Host/M.2 IO Voltage Level Shift Options on Rev B2 Adapter



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

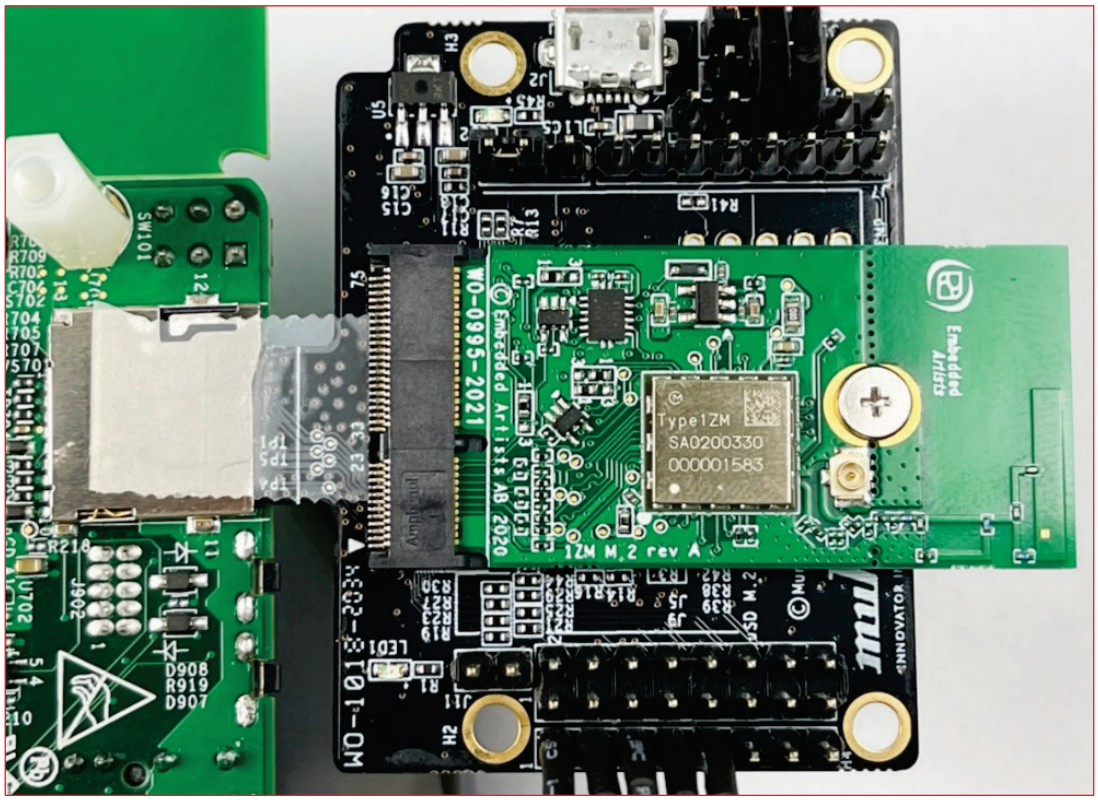
On both legacy NXP i.MX 6 EVKs and the newer i.MX 8 EVKs, a common issue that customers run into is an unreliable uSD/SD electrical connection when using Murata's uSD-M.2 Adapter. The poor interconnect is caused by two issues: push-push (micro) SD card connectors on NXP i.MX EVKs; and low friction interface 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 42**. 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 43**. 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 42: Securing uSD/SD Connection on i.MX 6 EVK

Figure 43: Securing uSD Connection on i.MX 8 EVK



10.4 uSD-M.2 Adapter High-Level Description

Figure 44 and **Figure 45** show the features on the uSD-M.2 Adapter Rev B; with text explanation in **Table 19**. **Figure 46**, **Figure 47**, **Figure 48** and **Figure 49** show the features on the uSD-M.2 Adapter Rev C; with text explanation in **Table 20**. 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 19: uSD-M.2 Adapter (Rev B) 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

Table 20: uSD-M.2 Adapter (Rev C) Features

Clear	Description
A	microSD connector provides Power (VBAT, GND) and WLAN-SDIO
B	SDIO bus test points (CLK, CMD, DAT0, DAT1, DAT2, DAT3)
C	**LED2** = 3.3V M.2 IO Voltage Indicator (Blue) – not illuminated in default configuration
D	Power LED Indicator (green): if not illuminated then no power applied to M.2 EVB
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	**J14** = M.2 VDDIO: J14 in 1-2 pos for M2 PIN64 M2 VDDIO (default); J14 in 2-3 pos for M2 VDDIO
H	External sleep clock input (32.768 kHz)
I	Threaded mount for M.2 screw: 30 mm distance from M.2 connector
J	Regulator to step down optional 5V VBAT from USB or Arduino header to 3.3V
K	**J7** = Optional Arduino Header Power Supply (8 pin header; 5V or 3.3V VBAT)
L	**J8** = BT UART RTS/CTS Signals (6 pin header)
M	**J13** = Host IO Voltage: J13 in 1-2 pos for 3.3V VDDIO (default); J13 in 2-3 pos for 1.8V
N	**J12** = M.2 IO Voltage: J12 in 1-2 pos for 1.8V VDDIO (default); J12 in 2-3 pos for 3.3V
O	**J2** = Optional 5V USB Power Supply via Micro-AB USB Connector
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 provide Power (VBAT, GND) and WLAN-SDIO
T	**JP1** = Slider Switch Setting **1** = DIR1_CTRL: OPEN = WL_WAKE_IN (input to M.2); CLOSED/ON = SPI_RXD_HOST (output to M.2) **2** = DIR2_CTRL: OPEN = PCM_SYNC is an input to M.2 (host is I2S master); CLOSED/ON = PCM_SYNC is an output to M.2 (host is I2S slave) **3** = DIR3_CTRL: OPEN = PCM_CLK is an input to M.2 (host is I2S master); CLOSED/ON = PCM_CLK is an output to M.2 (host is I2S slave) **4** = BT_IND_RST_HOST: OPEN = BT independent reset controlled by M.2 pin 54; CLOSED/ON = BT interface placed in reset, i.e., is disabled

Figure 44: uSD-M.2 Adapter (Rev B) Features (Top View)

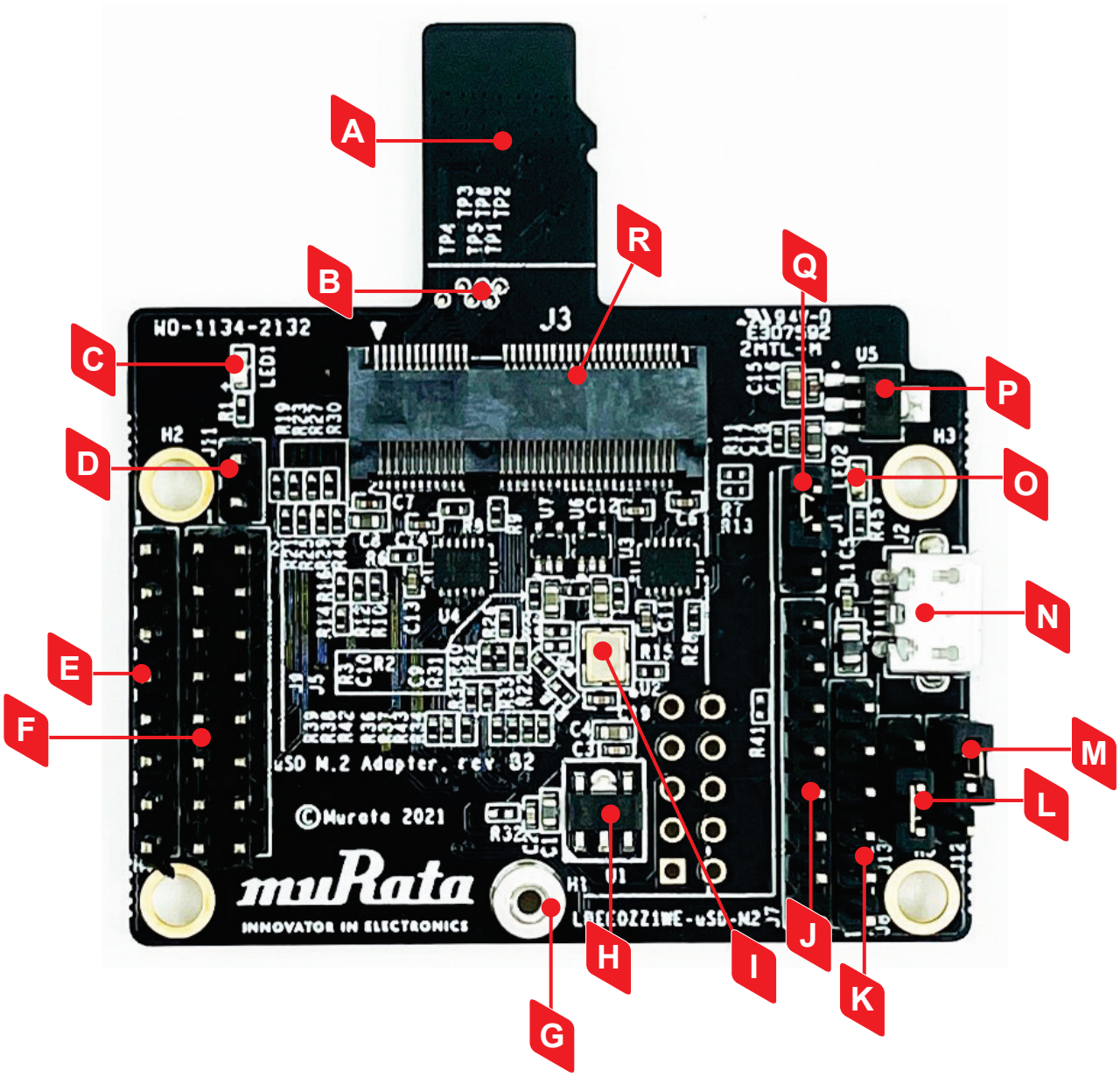


Figure 45: uSD-M.2 Adapter (Rev B) Features (Bottom View)

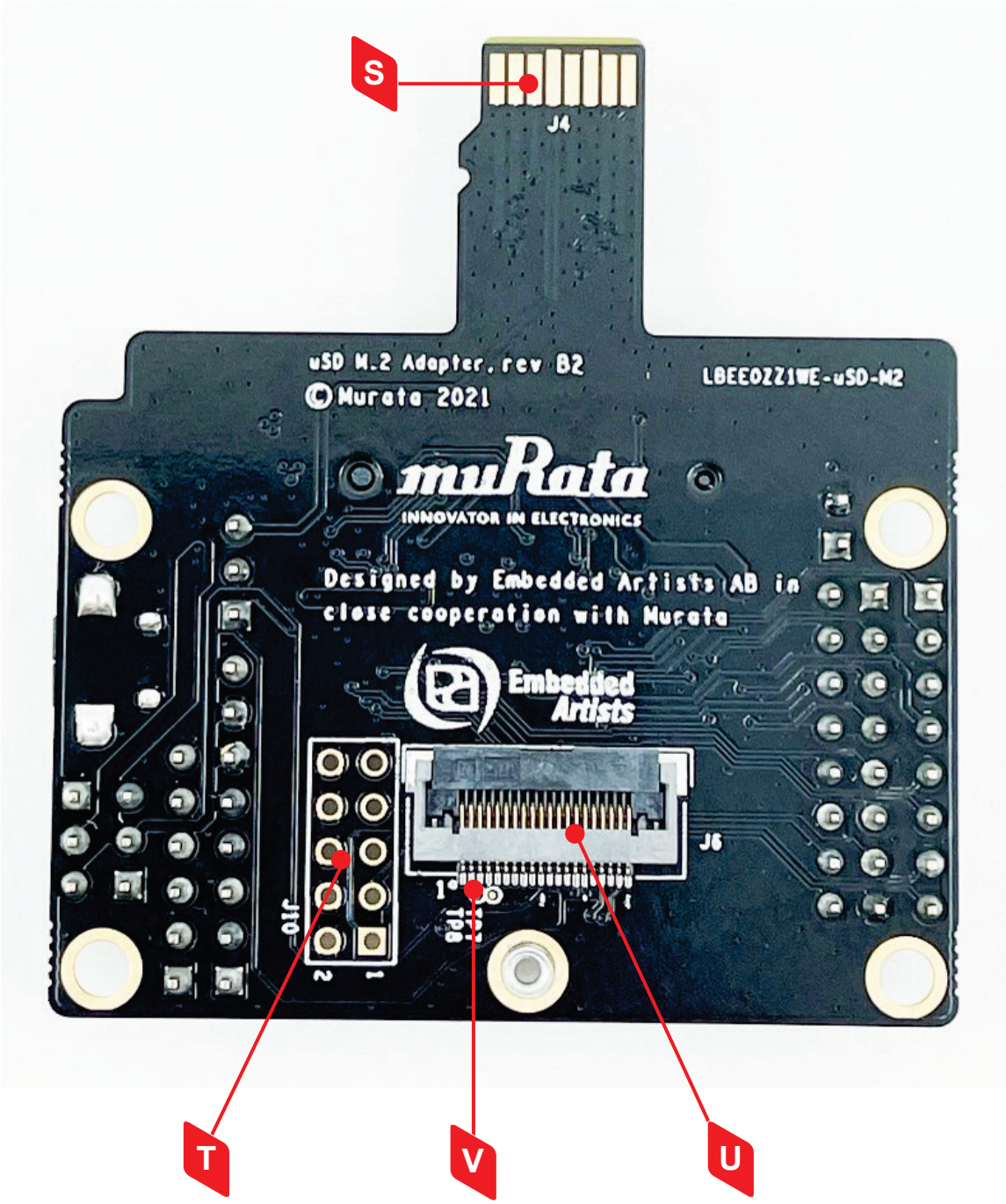


Figure 46: uSD-M.2 Adapter (Rev C - 2WE) Features (Top View)

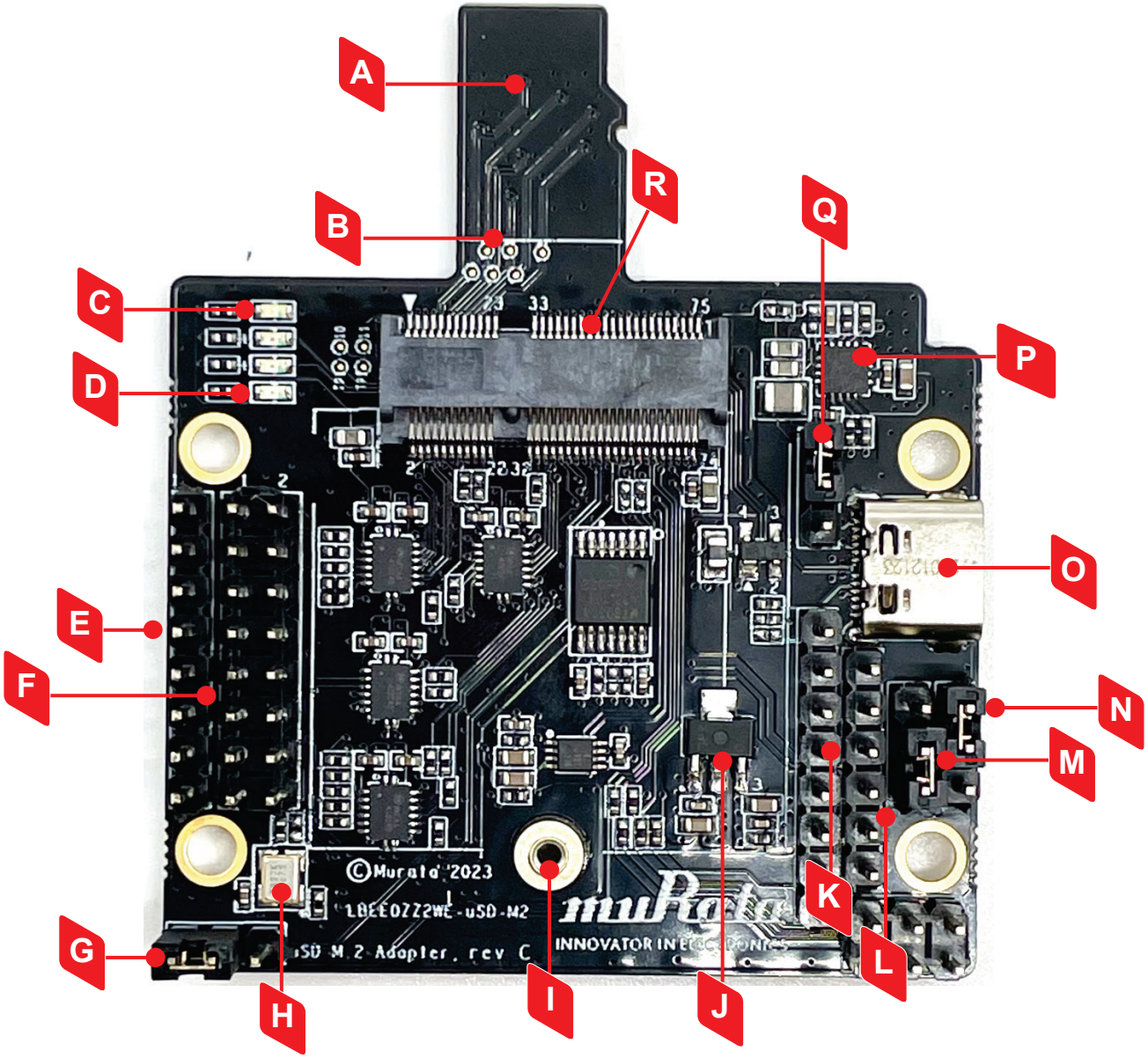


Figure 47: uSD-M.2 Adapter (Rev C - 2WE) Features (Bottom View)

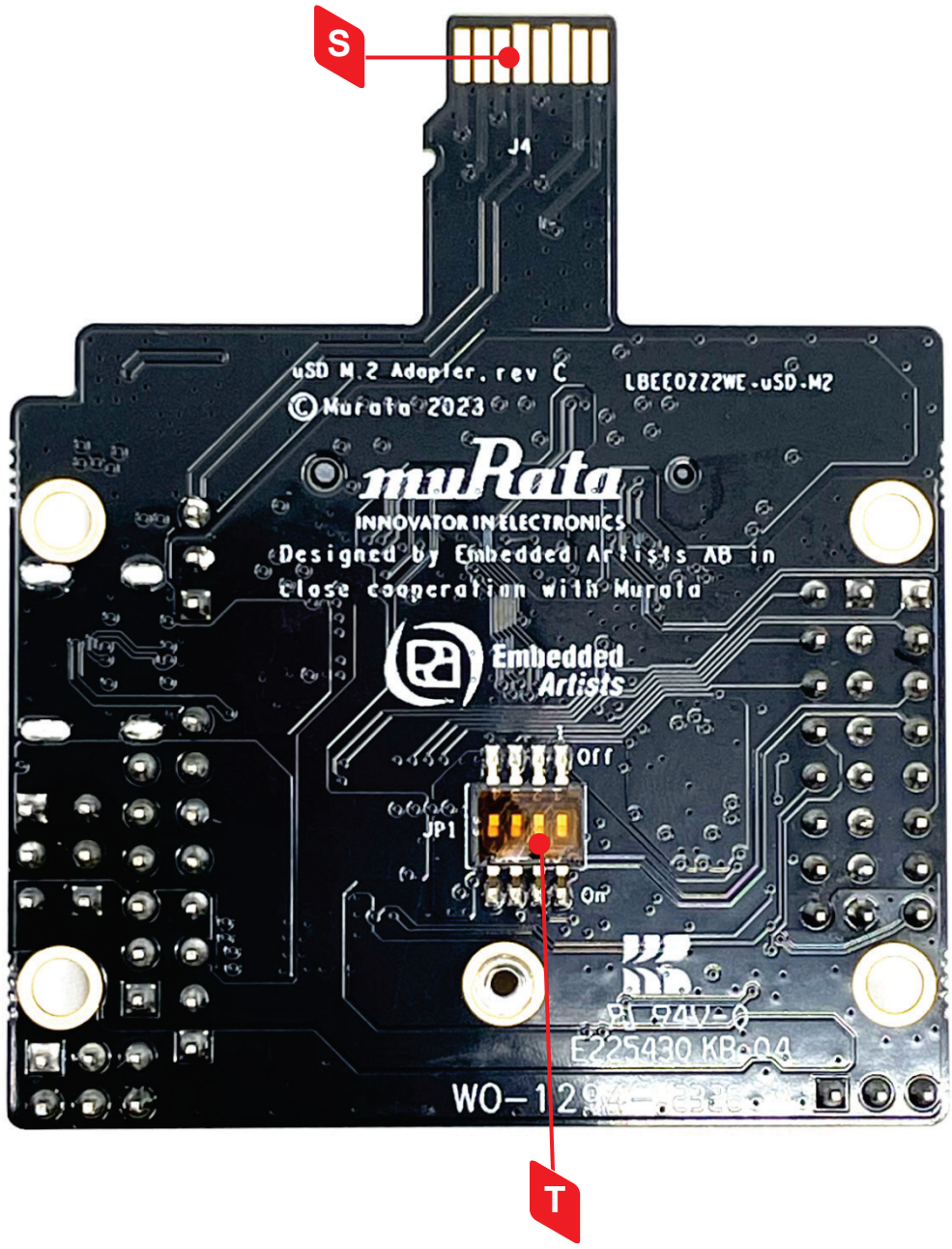


Figure 48: uSD-M.2 Adapter (Rev C - 2WF) Features (Top View)

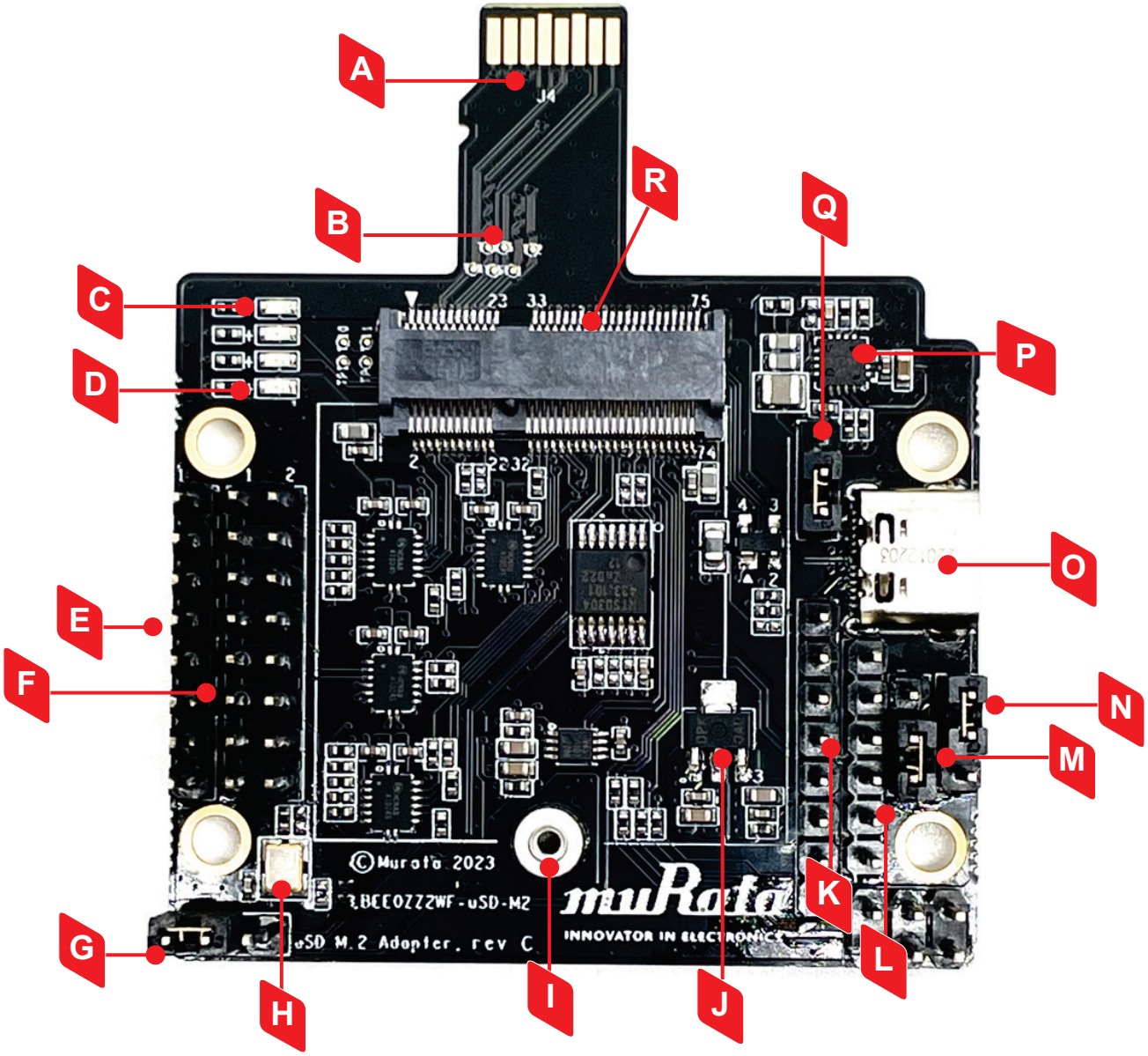
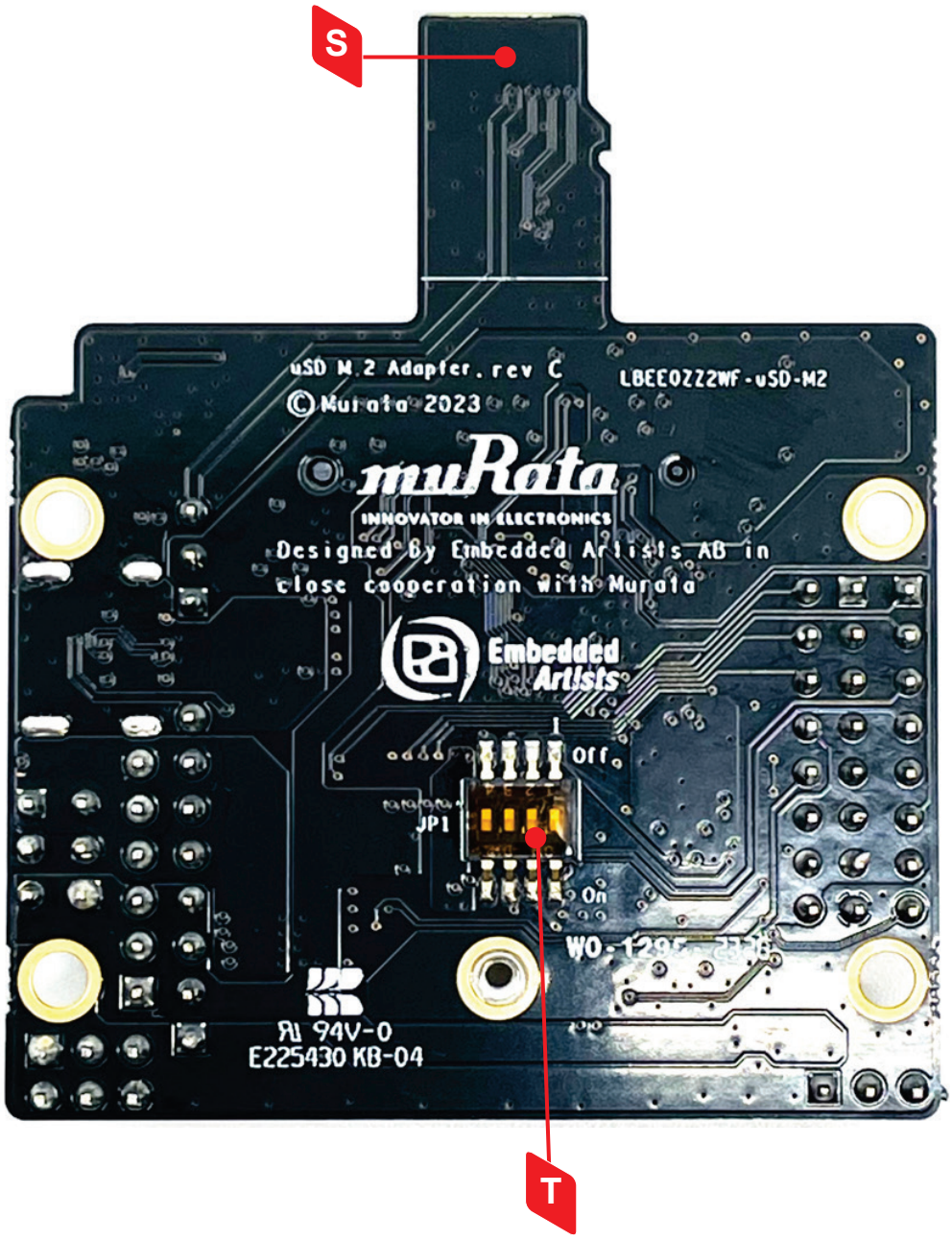


Figure 49: uSD-M.2 Adapter (Rev C - 2WF) Features (Bottom View)



11 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 2DS, 1XK, 2XK, 1ZM, 1YM, 1XL, 2XS, 2DL and 2EL 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.

12 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 50** shows that the carrier board can work with a variety of NXP i.MX6/7/8 (u)COM boards and eight different Murata-module based EVBs. With this platform, users can evaluate multiple i.MX processor and Wi-Fi/BT module permutations to find the best combination for their product. Also, Embedded Artists brings out all the test points you need for troubleshooting. With this platform, no adapter/interconnect is needed – either module-down solution or well-secured M.2 module. This is beneficial in two main areas: no limitation in Wi-Fi performance (WLAN-SDIO bus runs at full speed); and no mechanical issues (easy to secure M.2 module to EA Carrier Board). **Figure 50** shows how this platform works with (u)COM board and Wi-Fi/BT M.2 Modules. **Table 21** provides an Embedded Artists' i.MX (u)COM versus Murata module matrix. For comprehensive information on the Embedded Artists' solution refer to **Table 22**.

Figure 50: Combine i.MX COM with Wi-Fi/BT M.2 EVB

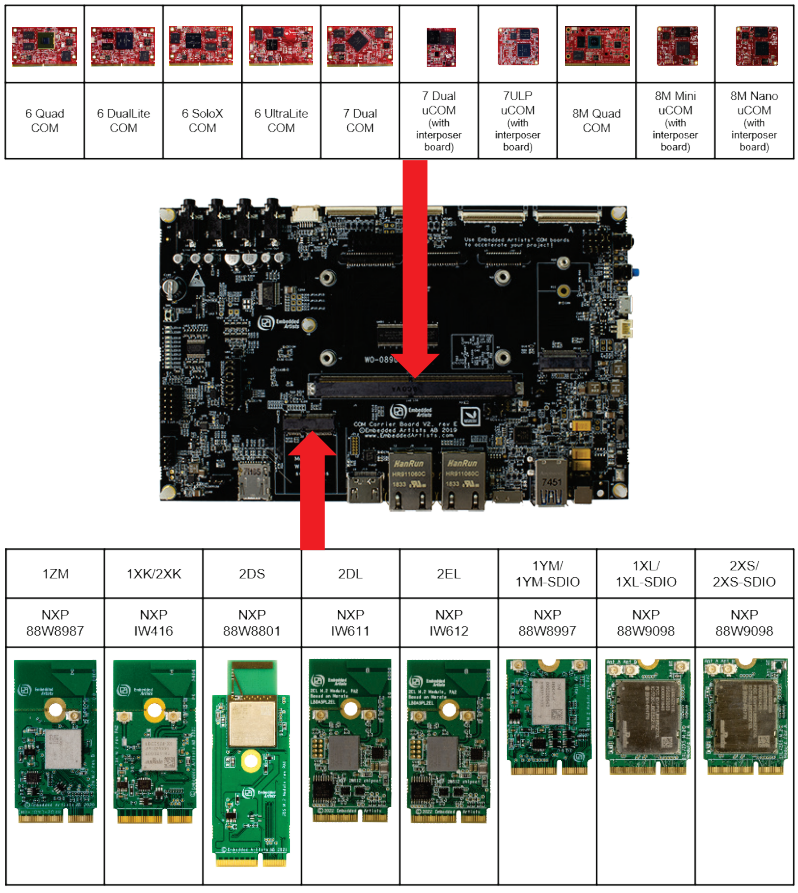


Table 21: Embedded Artists' i.MX Interconnect

EA i.MX (u)COM	1ZM	1YM	1XK/2XK	1XL/2XS	2DS	2DL	2EL
	NXP 88W8987	NXP 88W8997	NXP 88W8997	NXP 88W9098	NXP 88W8801	NXP IW611	NXP IW612
iMX93 uCOM	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]
iMX8M Quad COM	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]
iMX8M Mini uCOM	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]
iMX8M Nano uCOM	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]
iMX7 Dual COM	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]
iMX7 Dual uCOM	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]
iMX7ULP uCOM	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]
iMX6 Quad COM	NC[17]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]
iMX6 DualLite COM	NC[17]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]
iMX6 SoloX COM	NC[17]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]
iMX6 UltraLite COM	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]	M.2[16]

Table 22: 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.
i.MX 7/8/9 uCOM Carrier Board V3	Main baseboard which all the uCOM 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).
Getting Started with i.MX 8/9 Developer's Kit V3	How to bring up i.MX 8/9 Dev Kit (V3).
M.2 Module Family	Top level listing of M.2 EVBs.
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 23 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 23: 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.MX 6/7/8 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.

[16] Works with onboard M.2 slot
[17] No Connection option (due to 1ZM only supporting 1.8V SDIO VIO)

1XK M.2 Module Datasheet	Comprehensive details on 1XK Wi-Fi/BT M.2 Module.
1XL/2XS M.2 Module Datasheet	Comprehensive details on 1XL/2XS Wi-Fi/BT M.2 Module.
2DS M.2 Module Datasheet	Comprehensive details on 2DS Wi-Fi M.2 Module.
2DL/2EL M.2 Module Datasheet	Comprehensive details on 2DL/2EL Wi-Fi M.2 Module.

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

Table 24 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 24: 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.

13 Useful Links

Table 25 provides some useful links.

Table 25: 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.

14 Appendix A: Building Image Output

The following shows the output of building an image for NXP i.MX 8M Mini EVK, running 6.6.3 kernel release, for Murata module 1ZM, 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 Nanbield Yocto release running Linux 6.6.3.

```
Select which entry? 3
Selected : 6.6.3
```

5. Select 1ZM module.

```
Select your entry: 1
Selected module: 1ZM
```

6. Select i.MX 8M Mini EVK as target platform.

```
Select your entry: 4
Selected target: imx8mm-lpddr4-evk
```

7. Proceed with the default distro and image selections.

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

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

8. Enter build_8mm as build directory name.

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

9. Verify selection and start the build.

```
i.MX Yocto Release              : 6.6.3_1.0.0
Yocto branch                    : nanbield
Target                          : imx8mm-lpddr4-evk
Module                          : 1ZM
NXP i.MX EVK Part Number        : 8MMINILPD4-EVK
meta-murata-wireless Release Tag: imx-nanbield-6-6-3_r1.1
```

```
DISTRO                          : fsl-imx-wayland
Image                           : fsl-image-validation-imx
Build Directory                 : build_8mm

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

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

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

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

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

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

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

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

15 Appendix B: Type 2EL SPI Interface Testing Log

```
printenv fdtfile
fdtfile=imx8mm-evk-iw612-evk.dtb
u-boot=> boot
Working FDT set to 43000000
libfdt fdt_path_offset() returned FDT_ERR_NOTFOUND
starting USB...
Bus usb@32e40000: Port not available.
USB is stopped. Please issue 'usb start' first.
Card did not respond to voltage select! : -110
switch to partitions #0, OK
mmc2(part 0) is current device
Scanning mmc 2:1...
48008 bytes read in 2 ms (22.9 MiB/s)
Working FDT set to 43000000
Card did not respond to voltage select! : -110
Unable to open OP-TEE session (err=-5)
mm_communicate failed!
Error: Cannot initialize UEFI sub-system, r = 3
Running BSP bootcmd ...
switch to partitions #0, OK
mmc2(part 0) is current device
Failed to load 'boot.scr'
32262656 bytes read in 97 ms (317.2 MiB/s)
Booting from mmc ...
48008 bytes read in 17 ms (2.7 MiB/s)
## Flattened Device Tree blob at 43000000
   Booting using the fdt blob at 0x43000000
Working FDT set to 43000000
   Using Device Tree in place at 0000000043000000, end 000000004300eb87
Working FDT set to 43000000
adv7535_mipi2hdmi adv7535@3d: Can't find cec device id=0x3c
fail to probe panel device adv7535@3d
mxs_video lcdif@32e00000: failed to get any video link display timings
probe video device failed, ret -22

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x410fd034]
[    0.000000] Linux version 6.1.36+g04b05c5527e9 (oe-user@oe-host) (aarch64-
poky-linux-gcc (GCC) 12.3.0, GNU ld (GNU Binutils) 2.40.0.20230620) #1 SMP
PREEMPT Mon Sep  4 21:11:15 UTC 2023
[    0.000000] Machine model: FSL i.MX8MM EVK board
[    0.000000] efi: UEFI not found.
[    0.000000] Reserved memory: created CMA memory pool at 0x0000000096000000,
size 640 MiB
[    0.000000] OF: reserved mem: initialized node linux,cma, compatible id
shared-dma-pool
[    0.000000] NUMA: No NUMA configuration found
[    0.000000] NUMA: Faking a node at [mem 0x0000000040000000-
0x00000000bdffffff]
[    0.000000] NUMA: NODE_DATA [mem 0x95bcb700-0x95bcdfff]
[    0.000000] Zone ranges:
[    0.000000]   DMA      [mem 0x0000000040000000-0x00000000bdffffff]
[    0.000000]   DMA32    empty
[    0.000000]   Normal   empty
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000040000000-0x00000000bdffffff]
```

```
[    0.000000] Initmem setup node 0 [mem 0x0000000040000000-0x00000000bdffffff]
[    0.000000] On node 0, zone DMA: 8192 pages in unavailable ranges
[    0.000000] psci: probing for conduit method from DT.
[    0.000000] psci: PSCIv1.1 detected in firmware.
[    0.000000] psci: Using standard PSCI v0.2 function IDs
[    0.000000] psci: Trusted OS migration not required
[    0.000000] psci: SMC Calling Convention v1.2
[    0.000000] percpu: Embedded 20 pages/cpu s44520 r8192 d29208 u81920
[    0.000000] Detected VIPT I-cache on CPU0
[    0.000000] CPU features: detected: GIC system register CPU interface
[    0.000000] CPU features: kernel page table isolation forced ON by KASLR
[    0.000000] CPU features: detected: Kernel page table isolation (KPTI)
[    0.000000] CPU features: detected: ARM erratum 845719
[    0.000000] alternatives: applying boot alternatives
[    0.000000] Fallback order for Node 0: 0
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 508032
[    0.000000] Policy zone: DMA
[    0.000000] Kernel command line: console=ttymxc1,115200 root=/dev/mmcblk2p2
rootwait rw
[    0.000000] Dentry cache hash table entries: 262144 (order: 9, 2097152
bytes, linear)
[    0.000000] Inode-cache hash table entries: 131072 (order: 8, 1048576 bytes,
linear)
[    0.000000] mem auto-init: stack:all(zero), heap alloc:off, heap free:off
[    0.000000] Memory: 1336320K/2064384K available (19648K kernel code, 1618K
rwdata, 6740K rodata, 3328K init, 644K bss, 72704K reserved, 655360K cma-
reserved)
[    0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=4, Nodes=1
[    0.000000] rcu: Preemptible hierarchical RCU implementation.
[    0.000000] rcu:  RCU event tracing is enabled.
[    0.000000] rcu:  RCU restricting CPUs from NR_CPUS=256 to nr_cpu_ids=4.
[    0.000000]       Trampoline variant of Tasks RCU enabled.
[    0.000000]       Tracing variant of Tasks RCU enabled.
[    0.000000] rcu: RCU calculated value of scheduler-enlistment delay is 25
jiffies.
[    0.000000] rcu: Adjusting geometry for rcu_fanout_leaf=16, nr_cpu_ids=4
[    0.000000] NR_IRQS: 64, nr_irqs: 64, preallocated irqs: 0
[    0.000000] GICv3: GIC: Using split EOI/Deactivate mode
[    0.000000] GICv3: 128 SPIs implemented
[    0.000000] GICv3: 0 Extended SPIs implemented
[    0.000000] Root IRQ handler: gic_handle_irq
[    0.000000] GICv3: GICv3 features: 16 PPIs
[    0.000000] GICv3: CPU0: found redistributor 0 region 0:0x0000000038880000
[    0.000000] ITS: No ITS available, not enabling LPIs
[    0.000000] rcu: srcu_init: Setting srcu_struct sizes based on contention.
[    0.000000] arch_timer: cp15 timer(s) running at 8.00MHz (phys).
[    0.000000] clocksource: arch_sys_counter: mask: 0xffffffffffffff
max_cycles: 0x1d854df40, max_idle_ns: 440795202120 ns
[    0.000001] sched_clock: 56 bits at 8MHz, resolution 125ns, wraps every
2199023255500ns
[    0.000418] Console: colour dummy device 80x25
[    0.000483] Calibrating delay loop (skipped), value calculated using timer
frequency.. 16.00 BogoMIPS (lpj=32000)
[    0.000493] pid_max: default: 32768 minimum: 301
[    0.000550] LSM: Security Framework initializing
[    0.000637] Mount-cache hash table entries: 4096 (order: 3, 32768 bytes,
linear)
[    0.000649] Mountpoint-cache hash table entries: 4096 (order: 3, 32768
bytes, linear)
[    0.002034] cblist_init_generic: Setting adjustable number of callback
queues.
```

```
[    0.002045] cblist_init_generic: Setting shift to 2 and lim to 1.
[    0.002111] cblist_init_generic: Setting shift to 2 and lim to 1.
[    0.002262] rcu: Hierarchical SRCU implementation.
[    0.002265] rcu:  Max phase no-delay instances is 1000.
[    0.003436] EFI services will not be available.
[    0.003630] smp: Bringing up secondary CPUs ...
[    0.004134] Detected VIPT I-cache on CPU1
[    0.004220] GICv3: CPU1: found redistributor 1 region 0:0x00000000388a0000
[    0.004263] CPU1: Booted secondary processor 0x0000000001 [0x410fd034]
[    0.004758] Detected VIPT I-cache on CPU2
[    0.004819] GICv3: CPU2: found redistributor 2 region 0:0x00000000388c0000
[    0.004844] CPU2: Booted secondary processor 0x0000000002 [0x410fd034]
[    0.005275] Detected VIPT I-cache on CPU3
[    0.005338] GICv3: CPU3: found redistributor 3 region 0:0x00000000388e0000
[    0.005360] CPU3: Booted secondary processor 0x0000000003 [0x410fd034]
[    0.005415] smp: Brought up 1 node, 4 CPUs
[    0.005420] SMP: Total of 4 processors activated.
[    0.005424] CPU features: detected: 32-bit EL0 Support
[    0.005427] CPU features: detected: 32-bit EL1 Support
[    0.005431] CPU features: detected: CRC32 instructions
[    0.005489] CPU: All CPU(s) started at EL2
[    0.005508] alternatives: applying system-wide alternatives
[    0.007235] devtmpfs: initialized
[    0.014206] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff,
max_idle_ns: 7645041785100000 ns
[    0.014231] futex hash table entries: 1024 (order: 4, 65536 bytes, linear)
[    0.035158] pinctrl core: initialized pinctrl subsystem
[    0.037323] DMI not present or invalid.
[    0.037922] NET: Registered PF_NETLINK/PF_ROUTE protocol family
[    0.038963] DMA: preallocated 256 KiB GFP_KERNEL pool for atomic allocations
[    0.039162] DMA: preallocated 256 KiB GFP_KERNEL|GFP_DMA pool for atomic
allocations
[    0.039294] DMA: preallocated 256 KiB GFP_KERNEL|GFP_DMA32 pool for atomic
allocations
[    0.039351] audit: initializing netlink subsys (disabled)
[    0.039474] audit: type=2000 audit(0.036:1): state=initialized
audit_enabled=0 res=1
[    0.039960] thermal_sys: Registered thermal governor 'step_wise'
[    0.039964] thermal_sys: Registered thermal governor 'power_allocator'
[    0.039996] cpuidle: using governor menu
[    0.040183] hw-breakpoint: found 6 breakpoint and 4 watchpoint registers.
[    0.040265] ASID allocator initialised with 32768 entries
[    0.041163] Serial: AMBA PL011 UART driver
[    0.041224] imx mu driver is registered.
[    0.041248] imx rpmsg driver is registered.
[    0.050913] imx8mm-pinctrl 30330000.pinctrl: initialized IMX pinctrl driver
[    0.058349] platform 32e00000.lcdif: Fixed dependency cycle(s) with
/soc@0/bus@32c00000/mipi_dsi@32e10000
[    0.058634] platform 32e10000.mipi_dsi: Fixed dependency cycle(s) with
/soc@0/bus@30800000/i2c@30a30000/adv7535@3d
[    0.058874] platform 32e20000.csi1_bridge: Fixed dependency cycle(s) with
/soc@0/bus@32c00000/mipi_csi@32e30000
[    0.059200] platform 32e30000.mipi_csi: Fixed dependency cycle(s) with
/soc@0/bus@30800000/i2c@30a40000/ov5640_mipi@3c
[    0.059918] platform 32e40000.usb: Fixed dependency cycle(s) with
/soc@0/bus@30800000/i2c@30a30000/tcpc@50
[    0.065095] KASLR enabled
[    0.083370] HugeTLB: registered 1.00 GiB page size, pre-allocated 0 pages
[    0.083382] HugeTLB: 0 KiB vmemmap can be freed for a 1.00 GiB page
[    0.083386] HugeTLB: registered 32.0 MiB page size, pre-allocated 0 pages
[    0.083390] HugeTLB: 0 KiB vmemmap can be freed for a 32.0 MiB page
```

```
[    0.083394] HugeTLB: registered 2.00 MiB page size, pre-allocated 0 pages
[    0.083399] HugeTLB: 0 KiB vmemmap can be freed for a 2.00 MiB page
[    0.083403] HugeTLB: registered 64.0 KiB page size, pre-allocated 0 pages
[    0.083406] HugeTLB: 0 KiB vmemmap can be freed for a 64.0 KiB page
[    0.085230] ACPI: Interpreter disabled.
[    0.085901] iommu: Default domain type: Translated
[    0.085906] iommu: DMA domain TLB invalidation policy: strict mode
[    0.086179] SCSI subsystem initialized
[    0.086498] usbcore: registered new interface driver usbfs
[    0.086534] usbcore: registered new interface driver hub
[    0.086561] usbcore: registered new device driver usb
[    0.087643] mc: Linux media interface: v0.10
[    0.087676] videodev: Linux video capture interface: v2.00
[    0.087749] pps_core: LinuxPPS API ver. 1 registered
[    0.087753] pps_core: Software ver. 5.3.6 - Copyright 2005-2007 Rodolfo
Giometti <giometti@linux.it>
[    0.087765] PTP clock support registered
[    0.087907] EDAC MC: Ver: 3.0.0
[    0.088912] FPGA manager framework
[    0.088985] Advanced Linux Sound Architecture Driver Initialized.
[    0.089638] Bluetooth: Core ver 2.22
[    0.089667] NET: Registered PF_BLUETOOTH protocol family
[    0.089670] Bluetooth: HCI device and connection manager initialized
[    0.089678] Bluetooth: HCI socket layer initialized
[    0.089683] Bluetooth: L2CAP socket layer initialized
[    0.089695] Bluetooth: SCO socket layer initialized
[    0.090048] vgaarb: loaded
[    0.090544] clocksource: Switched to clocksource arch_sys_counter
[    0.090734] VFS: Disk quotas dquot_6.6.0
[    0.090774] VFS: Dquot-cache hash table entries: 512 (order 0, 4096 bytes)
[    0.090938] pnp: PnP ACPI: disabled
[    0.097685] NET: Registered PF_INET protocol family
[    0.097844] IP idents hash table entries: 32768 (order: 6, 262144 bytes,
linear)
[    0.099401] tcp_listen_portaddr_hash hash table entries: 1024 (order: 2,
16384 bytes, linear)
[    0.099458] Table-perturb hash table entries: 65536 (order: 6, 262144 bytes,
linear)
[    0.099472] TCP established hash table entries: 16384 (order: 5, 131072
bytes, linear)
[    0.099581] TCP bind hash table entries: 16384 (order: 7, 524288 bytes,
linear)
[    0.100042] TCP: Hash tables configured (established 16384 bind 16384)
[    0.100139] UDP hash table entries: 1024 (order: 3, 32768 bytes, linear)
[    0.100183] UDP-Lite hash table entries: 1024 (order: 3, 32768 bytes,
linear)
[    0.100344] NET: Registered PF_UNIX/PF_LOCAL protocol family
[    0.100717] RPC: Registered named UNIX socket transport module.
[    0.100722] RPC: Registered udp transport module.
[    0.100725] RPC: Registered tcp transport module.
[    0.100727] RPC: Registered tcp NFSv4.1 backchannel transport module.
[    0.101477] PCI: CLS 0 bytes, default 64
[    0.102113] hw perfevents: enabled with armv8_cortex_a53 PMU driver, 7
counters available
[    0.103027] kvm [1]: IPA Size Limit: 40 bits
[    0.104611] kvm [1]: GICv3: no GICV resource entry
[    0.104616] kvm [1]: disabling GICv2 emulation
[    0.104628] kvm [1]: GIC system register CPU interface enabled
[    0.104718] kvm [1]: vgic interrupt IRQ9
[    0.104804] kvm [1]: Hyp mode initialized successfully
[    0.105931] Initialise system trusted keyrings
```

```
[    0.106117] workingset: timestamp_bits=42 max_order=19 bucket_order=0
[    0.112478] squashfs: version 4.0 (2009/01/31) Phillip Lougher
[    0.113086] NFS: Registering the id_resolver key type
[    0.113149] Key type id_resolver registered
[    0.113152] Key type id_legacy registered
[    0.113222] nfs4filelayout_init: NFSv4 File Layout Driver Registering...
[    0.113227] nfs4flexfilelayout_init: NFSv4 Flexfile Layout Driver
Registering...
[    0.113248] jffs2: version 2.2. (NAND)   2001-2006 Red Hat, Inc.
[    0.113585] 9p: Installing v9fs 9p2000 file system support
[    0.149884] Key type asymmetric registered
[    0.149889] Asymmetric key parser 'x509' registered
[    0.149932] Block layer SCSI generic (bsg) driver version 0.4 loaded (major
243)
[    0.149938] io scheduler mq-deadline registered
[    0.149942] io scheduler kyber registered
[    0.156302] pwm-backlight backlight: supply power not found, using dummy
regulator
[    0.156953] EINJ: ACPI disabled.
[    0.167124] imx-sdma 302c0000.dma-controller: Direct firmware load for
imx/sdma/sdma-imx7d.bin failed with error -2
[    0.167140] imx-sdma 302c0000.dma-controller: Falling back to sysfs fallback
for: imx/sdma/sdma-imx7d.bin
[    0.172408] mxs-dma 33000000.dma-controller: initialized
[    0.177562] SoC: i.MX8MM revision 1.0
[    0.179990] Bus freq driver module loaded
[    0.185566] Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
[    0.188086] 30860000.serial: ttymxc0 at MMIO 0x30860000 (irq = 19, base_baud
= 5000000) is a IMX
[    0.188225] serial serial0: tty port ttymxc0 registered
[    0.188571] 30880000.serial: ttymxc2 at MMIO 0x30880000 (irq = 20, base_baud
= 5000000) is a IMX
[    0.189083] 30890000.serial: ttymxc1 at MMIO 0x30890000 (irq = 21, base_baud
= 1500000) is a IMX
[    1.345466] printk: console [ttymxc1] enabled
[    1.360351] imx-drm 32c00000.bus:display-subsystem: bound imx-lcdif-crtc.0
(ops lcdif_crtc_ops)
[    1.369227] imx_sec_dsim_drv 32e10000.mipi_dsi: version number is 0x1060200
[    1.376258] [drm:drm_bridge_attach] *ERROR* failed to attach bridge
/soc@0/bus@32c00000/mipi_dsi@32e10000 to encoder DSI-34: -517
[    1.387943] imx_sec_dsim_drv 32e10000.mipi_dsi: Failed to attach bridge:
32e10000.mipi_dsi
[    1.396216] imx_sec_dsim_drv 32e10000.mipi_dsi: failed to bind sec dsim
bridge: -517
[    1.410305] loop: module loaded
[    1.414835] of_reserved_mem_lookup() returned NULL
[    1.419876] megasas: 07.719.03.00-rc1
[    1.428888] spi-nor spi0.0: n25q256ax1 (32768 Kbytes)
[    1.438229] tun: Universal TUN/TAP device driver, 1.6
[    1.444260] thunder_xcv, ver 1.0
[    1.447546] thunder_bgx, ver 1.0
[    1.450812] nicpf, ver 1.0
[    1.455660] hns3: Hisilicon Ethernet Network Driver for Hip08 Family -
version
[    1.462894] hns3: Copyright (c) 2017 Huawei Corporation.
[    1.468259] hclge is initializing
[    1.471598] e1000: Intel(R) PRO/1000 Network Driver
[    1.476480] e1000: Copyright (c) 1999-2006 Intel Corporation.
[    1.482265] e1000e: Intel(R) PRO/1000 Network Driver
[    1.487233] e1000e: Copyright(c) 1999 - 2015 Intel Corporation.
[    1.493197] igb: Intel(R) Gigabit Ethernet Network Driver
```

```
[    1.498642] igb: Copyright (c) 2007-2014 Intel Corporation.
[    1.504256] igbvf: Intel(R) Gigabit Virtual Function Network Driver
[    1.510528] igbvf: Copyright (c) 2009 - 2012 Intel Corporation.
[    1.516650] sky2: driver version 1.30
[    1.520929] usbcore: registered new interface driver r8152
[    1.526707] VFIO - User Level meta-driver version: 0.3
[    1.535369] usbcore: registered new interface driver uas
[    1.540745] usbcore: registered new interface driver usb-storage
[    1.546838] usbcore: registered new interface driver usbserial_generic
[    1.553392] usbserial: USB Serial support registered for generic
[    1.559435] usbcore: registered new interface driver ftdi_sio
[    1.565202] usbserial: USB Serial support registered for FTDI USB Serial
Device
[    1.572544] usbcore: registered new interface driver usb_serial_simple
[    1.579095] usbserial: USB Serial support registered for carelink
[    1.585208] usbserial: USB Serial support registered for zio
[    1.590887] usbserial: USB Serial support registered for funsoft
[    1.596916] usbserial: USB Serial support registered for flashloader
[    1.603290] usbserial: USB Serial support registered for google
[    1.609227] usbserial: USB Serial support registered for libtransistor
[    1.615775] usbserial: USB Serial support registered for vivopay
[    1.621799] usbserial: USB Serial support registered for moto_modem
[    1.628089] usbserial: USB Serial support registered for motorola_tetra
[    1.634722] usbserial: USB Serial support registered for nokia
[    1.640576] usbserial: USB Serial support registered for novatel_gps
[    1.646954] usbserial: USB Serial support registered for hp4x
[    1.652723] usbserial: USB Serial support registered for suunto
[    1.658663] usbserial: USB Serial support registered for siemens_mpi
[    1.665051] usbcore: registered new interface driver usb_ehset_test
[    1.674478] input: 30370000.snvs:snvs-powerkey as
/devices/platform/soc@0/30000000.bus/30370000.snvs/30370000.snvs:snvs-
powerkey/input/input0
[    1.689092] snvs_rtc 30370000.snvs:snvs-rtc-lp: registered as rtc0
[    1.695316] snvs_rtc 30370000.snvs:snvs-rtc-lp: setting system clock to
2023-03-03T12:13:48 UTC (1677845628)
[    1.705292] i2c_dev: i2c /dev entries driver
[    1.711043] mx6s-csi 32e20000.csi1_bridge: initialising
[    1.717150] mxc_mipi-csi 32e30000.mipi_csi: supply mipi-phy not found, using
dummy regulator
[    1.725897] mxc_mipi-csi 32e30000.mipi_csi: mipi csi v4l2 device registered
[    1.732876] CSI: Registered sensor subdevice: mxc_mipi-csi.0
[    1.738554] mxc_mipi-csi 32e30000.mipi_csi: lanes: 2, hs_settle: 13,
clk_settle: 2, wclk: 1, freq: 333000000
[    1.751990] Bluetooth: HCI UART driver ver 2.3
[    1.756473] Bluetooth: HCI UART protocol H4 registered
[    1.761620] Bluetooth: HCI UART protocol BCSP registered
[    1.766962] Bluetooth: HCI UART protocol LL registered
[    1.772109] Bluetooth: HCI UART protocol ATH3K registered
[    1.777528] Bluetooth: HCI UART protocol Three-wire (H5) registered
[    1.783898] Bluetooth: HCI UART protocol Broadcom registered
[    1.789587] Bluetooth: HCI UART protocol QCA registered
[    1.796425] sdhci: Secure Digital Host Controller Interface driver
[    1.802631] sdhci: Copyright(c) Pierre Ossman
[    1.807710] Synopsys Designware Multimedia Card Interface Driver
[    1.814381] sdhci-pltfm: SDHCI platform and OF driver helper
[    1.822955] ledtrig-cpu: registered to indicate activity on CPUs
[    1.830052] SMCCC: SOC_ID: ARCH_SOC_ID not implemented, skipping ....
[    1.837026] usbcore: registered new interface driver usbhid
[    1.842614] usbhid: USB HID core driver
[    1.851383]  cs_system_cfg: CoreSight Configuration manager initialised
[    1.853277] mmc2: SDHCI controller on 30b60000.mmc [30b60000.mmc] using ADMA
```

```
[    1.859609] optee: probing for conduit method.
[    1.869552] optee: revision 3.21 (4e322819)
[    1.870634] optee: dynamic shared memory is enabled
[    1.880052] optee: initialized driver
[    1.886591] hantrodec 0 : module inserted. Major = 510
[    1.892282] hantrodec 1 : module inserted. Major = 510
[    1.898287] hx280enc: module inserted. Major <509>
[    1.908976] NET: Registered PF_LLC protocol family
[    1.914848] NET: Registered PF_INET6 protocol family
[    1.921004] Segment Routing with IPv6
[    1.924745] In-situ OAM (IOAM) with IPv6
[    1.928734] NET: Registered PF_PACKET protocol family
[    1.933833] bridge: filtering via arp/ip/ip6tables is no longer available by
default. Update your scripts to load br_netfilter if you need this.
[    1.947840] Bluetooth: RFCOMM TTY layer initialized
[    1.952747] Bluetooth: RFCOMM socket layer initialized
[    1.957922] Bluetooth: RFCOMM ver 1.11
[    1.961693] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
[    1.967018] Bluetooth: BNEP filters: protocol multicast
[    1.972266] Bluetooth: BNEP socket layer initialized
[    1.977265] Bluetooth: HIDP (Human Interface Emulation) ver 1.2
[    1.983204] Bluetooth: HIDP socket layer initialized
[    1.988942] 8021q: 802.1Q VLAN Support v1.8
[    1.993165] lib80211: common routines for IEEE802.11 drivers
[    1.998951] 9pnet: Installing 9P2000 support
[    2.003367] Key type dns_resolver registered
[    2.008069] mmc2: new HS400 Enhanced strobe MMC card at address 0001
[    2.008387] registered taskstats version 1
[    2.015596] mmcblk2: mmc2:0001 DG4032 29.1 GiB
[    2.018582] Loading compiled-in X.509 certificates
[    2.028246]  mmcblk2: p1 p2
[    2.032599] mmcblk2boot0: mmc2:0001 DG4032 4.00 MiB
[    2.039155] mmcblk2boot1: mmc2:0001 DG4032 4.00 MiB
[    2.045992] mmcblk2rpmb: mmc2:0001 DG4032 4.00 MiB, chardev (234:0)
[    2.056097] usb_phy_generic usbphynop1: supply vcc not found, using dummy
regulator
[    2.063887] usb_phy_generic usbphynop1: dummy supplies not allowed for
exclusive requests
[    2.072318] usb_phy_generic usbphynop2: supply vcc not found, using dummy
regulator
[    2.080063] usb_phy_generic usbphynop2: dummy supplies not allowed for
exclusive requests
[    2.148556] nxp-pca9450 0-0025: pca9450a probed.
[    2.153320] i2c i2c-0: IMX I2C adapter registered
[    2.159649] adv7511 1-003d: supply avdd not found, using dummy regulator
[    2.166486] adv7511 1-003d: supply dvdd not found, using dummy regulator
[    2.173250] adv7511 1-003d: supply pvdd not found, using dummy regulator
[    2.179995] adv7511 1-003d: supply a2vdd not found, using dummy regulator
[    2.186827] adv7511 1-003d: supply v3p3 not found, using dummy regulator
[    2.193569] adv7511 1-003d: supply v1p2 not found, using dummy regulator
[    2.201385] adv7511 1-003d: Probe failed. Remote port 'mipi_dsi@32e10000'
disabled
[    2.213866] i2c i2c-1: IMX I2C adapter registered
[    2.220142] pca953x 2-0020: using no AI
[    2.229545] ov5640_mipi 2-003c: No sensor reset pin available
[    2.235358] ov5640_mipi 2-003c: supply DOVDD not found, using dummy
regulator
[    2.242619] ov5640_mipi 2-003c: supply DVDD not found, using dummy regulator
[    2.249710] ov5640_mipi 2-003c: supply AVDD not found, using dummy regulator
[    2.267529] ov5640_mipi 2-003c: Read reg error: reg=300a
[    2.272856] ov5640_mipi 2-003c: Camera is not found
```

```
[    2.278011] i2c i2c-2: IMX I2C adapter registered
[    2.284687] pwm-backlight backlight: supply power not found, using dummy
regulator
[    2.285656] imx6q-pcie 33800000.pcie: host bridge /soc@0/pcie@33800000
ranges:
[    2.296331] imx-drm 32c00000.bus:display-subsystem: bound imx-lcdif-crtc.0
(ops lcdif_crtc_ops)
[    2.299623] imx6q-pcie 33800000.pcie:       IO 0x001ff80000..0x001ff8ffff ->
0x0000000000
[    2.308457] imx_sec_dsim_drv 32e10000.mipi_dsi: version number is 0x1060200
[    2.316491] imx6q-pcie 33800000.pcie:      MEM 0x0018000000..0x001fefffff ->
0x0018000000
[    2.323498] [drm:drm_bridge_attach] *ERROR* failed to attach bridge
/soc@0/bus@32c00000/mipi_dsi@32e10000 to encoder DSI-34: -19
[    2.343212] imx_sec_dsim_drv 32e10000.mipi_dsi: Failed to attach bridge:
32e10000.mipi_dsi
[    2.351488] imx_sec_dsim_drv 32e10000.mipi_dsi: failed to bind sec dsim
bridge: -19
[    2.359157] imx-drm 32c00000.bus:display-subsystem: bound 32e10000.mipi_dsi
(ops imx_sec_dsim_ops)
[    2.368890] [drm] Initialized imx-drm 1.0.0 20120507 for
32c00000.bus:display-subsystem on minor 0
[    2.383000] pps pps0: new PPS source ptp0
[    2.547101] imx6q-pcie 33800000.pcie: iATU: unroll T, 4 ob, 4 ib, align 64K,
limit 4G
[    2.583030] vddio: Bringing 1500000uV into 1800000-1800000uV
[    2.590011] fec 30be0000.ethernet eth0: registered PHC device 0
[    2.602407] imx-cpufreq-dt imx-cpufreq-dt: cpu speed grade 3 mkt segment 0
supported-hw 0x8 0x1
[    2.615640] galcore: clk_get vg clock failed, disable vg!
[    2.621528] Galcore version 6.4.11.p2.684571
[    2.649708] mmc1: SDHCI controller on 30b50000.mmc [30b50000.mmc] using ADMA
[    2.670366] [drm] Initialized vivante 1.0.0 20170808 for 38000000.gpu on
minor 1
[    2.758200] mmc1: new ultra high speed SDR104 SDIO card at address 0001
[    3.011461] OF: graph: no port node found in
/soc@0/bus@30800000/i2c@30a30000/tcpc@50/connector
[    3.020304] OF: graph: no port node found in
/soc@0/bus@30800000/i2c@30a30000/tcpc@50/connector
[    3.029049] OF: graph: no port node found in
/soc@0/bus@30800000/i2c@30a30000/tcpc@50/connector
[    3.060042] pwm-backlight backlight: supply power not found, using dummy
regulator
[    3.070917] pwm-backlight backlight: supply power not found, using dummy
regulator
[    3.081847] cfg80211: Loading compiled-in X.509 certificates for regulatory
database
[    3.092403] cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[    3.099025] platform regulatory.0: Direct firmware load for regulatory.db
failed with error -2
[    3.104843] ALSA device list:
[    3.107659] platform regulatory.0: Falling back to sysfs fallback for:
regulatory.db
[    3.110618]   No soundcards found.
[    3.552145] imx6q-pcie 33800000.pcie: Phy link never came up
[    4.563373] imx6q-pcie 33800000.pcie: Phy link never came up
[    4.569179] imx6q-pcie 33800000.pcie: PCI host bridge to bus 0000:00
[    4.575611] pci_bus 0000:00: root bus resource [bus 00-ff]
[    4.581124] pci_bus 0000:00: root bus resource [io  0x0000-0xffff]
[    4.587336] pci_bus 0000:00: root bus resource [mem 0x18000000-0x1fefffff]
[    4.594256] pci 0000:00:00.0: [16c3:abcd] type 01 class 0x060400
```

```
[    4.600480] pci 0000:00:00.0: reg 0x10: [mem 0x00000000-0x000fffff]
[    4.606781] pci 0000:00:00.0: reg 0x38: [mem 0x00000000-0x0000ffff pref]
[    4.613547] pci 0000:00:00.0: supports D1
[    4.617569] pci 0000:00:00.0: PME# supported from D0 D1 D3hot D3cold
[    4.625531] pci 0000:00:00.0: BAR 0: assigned [mem 0x18000000-0x180fffff]
[    4.632336] pci 0000:00:00.0: BAR 6: assigned [mem 0x18100000-0x1810ffff
pref]
[    4.639599] pci 0000:00:00.0: PCI bridge to [bus 01-ff]
[    4.645192] pcieport 0000:00:00.0: PME: Signaling with IRQ 220
[    4.651465] pwm-backlight backlight: supply power not found, using dummy
regulator
[    4.700661] EXT4-fs (mmcblk2p2): mounted filesystem with ordered data mode.
Quota mode: none.
[    4.709380] VFS: Mounted root (ext4 filesystem) on device 179:2.
[    4.715815] devtmpfs: mounted
[    4.719501] Freeing unused kernel memory: 3328K
[    4.724137] Run /sbin/init as init process
[    4.838082] systemd[1]: systemd 253.1^ running in system mode (+PAM -AUDIT -
SELINUX -APPARMOR +IMA -SMACK +SECCOMP -GCRYPT -GNUTLS -OPENSSL +ACL +BLKID -
CURL -ELFUTILS -FIDO2 -IDN2 -IDN -IPTC +KMOD -LIBCRYPTSETUP +LIBFDISK -PCRE2 -
PWQUALITY -P11KIT -QRENCODE -TPM2 -BZIP2 -LZ4 -XZ -ZLIB +ZSTD -BPF_FRAMEWORK
+XKBCOMMON +UTMP +SYSVINIT default-hierarchy=hybrid)
[    4.869873] systemd[1]: Detected architecture arm64.

Welcome to NXP i.MX Release Distro 6.1-mickledore (mickledore)!

[    4.944132] systemd[1]: Hostname set to <imx8mmevk>.
[    5.051361] systemd-sysv-generator[189]: SysV service
'/etc/init.d/umountnfs.sh' lacks a native systemd unit file. Automatically
generating a unit file for compatibility. Please update package to include a
native systemd unit file, in order to make it more safe and robust.
[    5.076932] systemd-sysv-generator[189]: SysV service '/etc/init.d/halt'
lacks a native systemd unit file. Automatically generating a unit file for
compatibility. Please update package to include a native systemd unit file, in
order to make it more safe and robust.
[    5.100724] systemd-sysv-generator[189]: SysV service '/etc/init.d/reboot'
lacks a native systemd unit file. Automatically generating a unit file for
compatibility. Please update package to include a native systemd unit file, in
order to make it more safe and robust.
[    5.127666] systemd-sysv-generator[189]: SysV service '/etc/init.d/rc.local'
lacks a native systemd unit file. Automatically generating a unit file for
compatibility. Please update package to include a native systemd unit file, in
order to make it more safe and robust.
[    5.153763] systemd-sysv-generator[189]: SysV service '/etc/init.d/fuse'
lacks a native systemd unit file. Automatically generating a unit file for
compatibility. Please update package to include a native systemd unit file, in
order to make it more safe and robust.
[    5.179297] systemd-sysv-generator[189]: SysV service '/etc/init.d/trousers'
lacks a native systemd unit file. Automatically generating a unit file for
compatibility. Please update package to include a native systemd unit file, in
order to make it more safe and robust.
[    5.203416] systemd-sysv-generator[189]: SysV service '/etc/init.d/single'
lacks a native systemd unit file. Automatically generating a unit file for
compatibility. Please update package to include a native systemd unit file, in
order to make it more safe and robust.
[    5.227397] systemd-sysv-generator[189]: SysV service '/etc/init.d/umountfs'
lacks a native systemd unit file. Automatically generating a unit file for
compatibility. Please update package to include a native systemd unit file, in
order to make it more safe and robust.
[    5.251651] systemd-sysv-generator[189]: SysV service '/etc/init.d/sshd'
lacks a native systemd unit file. Automatically generating a unit file for
```

```
compatibility. Please update package to include a native systemd unit file, in
order to make it more safe and robust.
[    5.279252] systemd-sysv-generator[189]: SysV service '/etc/init.d/sendsigs'
lacks a native systemd unit file. Automatically generating a unit file for
compatibility. Please update package to include a native systemd unit file, in
order to make it more safe and robust.
[    5.309647] systemd-sysv-generator[189]: SysV service
'/etc/init.d/psplash.sh' lacks a native systemd unit file. Automatically
generating a unit file for compatibility. Please update package to include a
native systemd unit file, in order to make it more safe and robust.
[    5.334352] systemd-sysv-generator[189]: SysV service '/etc/init.d/save-
rtc.sh' lacks a native systemd unit file. Automatically generating a unit file
for compatibility. Please update package to include a native systemd unit file,
in order to make it more safe and robust.
[    5.642278] systemd[1]: Queued start job for default target Graphical
Interface.
[    5.703317] systemd[1]: Created slice Slice /system/getty.
[  OK  ] Created slice Slice /system/getty.
[    5.729961] systemd[1]: Created slice Slice /system/modprobe.
[  OK  ] Created slice Slice /system/modprobe.
[    5.753475] systemd[1]: Created slice Slice /system/serial-getty.
[  OK  ] Created slice Slice /system/serial-getty.
[    5.776952] systemd[1]: Created slice User and Session Slice.
[  OK  ] Created slice User and Session Slice.
[    5.799023] systemd[1]: Started Dispatch Password Requests to Console
Directory Watch.
[  OK  ] Started Dispatch Password  ts to Console Directory Watch.
[    5.822977] systemd[1]: Started Forward Password Requests to Wall Directory
Watch.
[  OK  ] Started Forward Password R uests to Wall Directory Watch.
[    5.846968] systemd[1]: Reached target Host and Network Name Lookups.
[  OK  ] Reached target Host and Network Name Lookups.
[    5.871188] systemd[1]: Reached target Path Units.
[  OK  ] Reached target Path Units.
[    5.890704] systemd[1]: Reached target Remote File Systems.
[  OK  ] Reached target Remote File Systems.
[    5.914744] systemd[1]: Reached target Slice Units.
[  OK  ] Reached target Slice Units.
[    5.934738] systemd[1]: Reached target Swaps.
[  OK  ] Reached target Swaps.
[    5.993451] systemd[1]: Listening on RPCbind Server Activation Socket.
[  OK  ] Listening on RPCbind Server Activation Socket.
[    6.018844] systemd[1]: Reached target RPC Port Mapper.
[  OK  ] Reached target RPC Port Mapper.
[    6.039386] systemd[1]: Listening on Syslog Socket.
[  OK  ] Listening on Syslog Socket.
[    6.059208] systemd[1]: Listening on initctl Compatibility Named Pipe.
[  OK  ] Listening on initctl Compatibility Named Pipe.
[    6.087567] systemd[1]: Listening on Journal Audit Socket.
[  OK  ] Listening on Journal Audit Socket.
[    6.111852] systemd[1]: Listening on Journal Socket (/dev/log).
[  OK  ] Listening on Journal Socket (/dev/log).
[    6.135651] systemd[1]: Listening on Journal Socket.
[  OK  ] Listening on Journal Socket.
[    6.155383] systemd[1]: Listening on Network Service Netlink Socket.
[  OK  ] Listening on Network Service Netlink Socket.
[    6.180090] systemd[1]: Listening on udev Control Socket.
[  OK  ] Listening on udev Control Socket.
[    6.203808] systemd[1]: Listening on udev Kernel Socket.
[  OK  ] Listening on udev Kernel Socket.
[    6.227498] systemd[1]: Listening on User Database Manager Socket.
```

```
[  OK  ] Listening on User Database Manager Socket.
[    6.287063] systemd[1]: Mounting Huge Pages File System...
         Mounting Huge Pages File System...
[    6.311109] systemd[1]: Mounting POSIX Message Queue File System...
         Mounting POSIX Message Queue File System...
[    6.339704] systemd[1]: Mounting Kernel Debug File System...
         Mounting Kernel Debug File System...
[    6.359317] systemd[1]: Kernel Trace File System was skipped because of an
unmet condition check (ConditionPathExists=/sys/kernel/tracing).
[    6.377594] systemd[1]: Mounting Temporary Directory /tmp...
         Mounting Temporary Directory /tmp...
[    6.400594] systemd[1]: Starting Create List of Static Device Nodes...
         Starting Create List of Static Device Nodes...
[    6.427521] systemd[1]: Starting Load Kernel Module configfs...
         Starting Load Kernel Module configfs...
[    6.450627] systemd[1]: Starting Load Kernel Module drm...
         Starting Load Kernel Module drm...
[    6.475766] systemd[1]: Starting Load Kernel Module fuse...
         Starting Load Kernel Module fuse...
[    6.489536] fuse: init (API version 7.37)
[    6.504169] systemd[1]: Starting RPC Bind...
         Starting RPC Bind...
[    6.522898] systemd[1]: File System Check on Root Device was skipped because
of an unmet condition check (ConditionPathIsReadWrite=!/).
[    6.541045] systemd[1]: Starting Journal Service...
         Starting Journal Service...
[    6.569586] systemd[1]: Starting Load Kernel Modules...
         Starting Load Kernel Modules...
[    6.591085] systemd[1]: Starting Generate network units from Kernel command
line...
[    6.591873] systemd-journald[206]: Collecting audit messages is enabled.
         Starting Generate network  ts from Kernel command line...
[    6.627246] systemd[1]: Starting Remount Root and Kernel File Systems...
         Starting Remount Root and Kernel File Systems...
[    6.646366] EXT4-fs (mmcblk2p2): re-mounted. Quota mode: none.
[    6.687318] systemd[1]: Starting Coldplug All udev Devices...
         Starting Coldplug All udev Devices...
[    6.711066] systemd[1]: Starting Setup Virtual Console...
         Starting Setup Virtual Console...
[    6.737835] systemd[1]: Started RPC Bind.
[  OK  ] Started RPC Bind.
[    6.755312] systemd[1]: Started Journal Service.
[  OK  ] Started Journal Service.
[  OK  ] Mounted Huge Pages File System.
[  OK  ] Mounted POSIX Message Queue File System.
[  OK  ] Mounted Kernel Debug File System.
[  OK  ] Mounted Temporary Directory /tmp.
[  OK  ] Finished Create List of Static Device Nodes.
[  OK  ] Finished Load Kernel Module configfs.
[  OK  ] Finished Load Kernel Module drm.
[  OK  ] Finished Load Kernel Module fuse.
[  OK  ] Finished Load Kernel Modules.
[  OK  ] Finished Generate network units from Kernel command line.
[  OK  ] Finished Remount Root and Kernel File Systems.
[  OK  ] Finished Setup Virtual Console.
         Mounting FUSE Control File System...
         Mounting Kernel Configuration File System...
         Starting Flush Journal to Persistent Storage...
[    7.075145] systemd-journald[206]: Received client request to flush runtime
journal.
         Starting Apply Kernel Variables...
```

```
         Starting Create Static Device Nodes in /dev...
[  OK  ] Mounted FUSE Control File System.
[  OK  ] Mounted Kernel Configuration File System.
[  OK  ] Finished Flush Journal to Persistent Storage.
[  OK  ] Finished Apply Kernel Variables.
[  OK  ] Finished Create Static Device Nodes in /dev.
[  OK  ] Reached target Preparation for Local File Systems.
         Mounting /var/volatile...
[    7.313270] audit: type=1334 audit(1677845634.112:2): prog-id=5 op=LOAD
[    7.319962] audit: type=1334 audit(1677845634.120:3): prog-id=6 op=LOAD
         Starting Rule-based Manage for Device Events and Files...
[  OK  ] Mounted /var/volatile.
[  OK  ] Finished Coldplug All udev Devices.
         Starting Load/Save OS Random Seed...
[  OK  ] Reached target Local File Systems.
         Starting Create Volatile Files and Directories...
[  OK  ] Started Rule-based Manager for Device Events and Files.
[  OK  ] Finished Create Volatile Files and Directories.
         Starting Start Psplash Boot Screen...
         Starting Network Time Synchronization...
         Starting Record System Boot/Shutdown in UTMP...
[  OK  ] Started Start Psplash Boot Screen.
[  OK  ] Started Start psplash-syst progress communication helper.
[  OK  ] Finished Record System Boot/Shutdown in UTMP.
[  OK  ] Started Network Time Synchronization.
[  OK  ] Reached target System Initialization.
[  OK  ] Started Daily Cleanup of Temporary Directories.
[  OK  ] Reached target System Time Set.
[  OK  ] Started Daily rotation of log files.
[  OK  ] Reached target Timer Units.
[  OK  ] Listening on Avahi mDNS/DNS-SD Stack Activation Socket.
[  OK  ] Listening on D-Bus System Message Bus Socket.
         Starting Docker Socket for the API...
         Starting sshd.socket...
         Starting Weston socket...
         Starting Console System Startup Logging...
[  OK  ] Listening on Docker Socket for the API.
[  OK  ] Listening on sshd.socket.
[  OK  ] Listening on Weston socket.
[  OK  ] Finished Console System Startup Logging.
[  OK  ] Reached target Socket Units.
[  OK  ] Reached target Basic System.
[    8.328541] Registered IR keymap rc-empty
[    8.332692] rc rc0: gpio_ir_recv as /devices/platform/ir-receiver/rc/rc0
[  OK  ] Started     8.342291] input: gpio_ir_recv as /devices/platform/ir-
receiver/rc/rc0/input1
;39mJob spooling tools.
[    8.352442] pwm-backlight backlight: supply power not found, using dummy
regulator
[  OK  ] Started Periodic Command Scheduler.
         Starting D-Bus System Message Bus...
[  OK  ] Started Linux Firmware Loader Daemon.
[    8.426044] imx-sdma 30bd0000.dma-controller: firmware found.
[    8.426975] imx-sdma 302c0000.dma-controller: firmware found.
[    8.431994] imx-sdma 30bd0000.dma-controller: loaded firmware 4.6
[    8.437735] imx-sdma 302b0000.dma-controller: firmware found.
[    8.490089] caam 30900000.crypto: device ID = 0x0a16040100000000 (Era 9)
[    8.497153] caam 30900000.crypto: job rings = 1, qi = 0
[  OK  ] Started Configuration for i.MX GPU [    8.511045] pwm-backlight
backlight: supply power not found, using dummy regulator
(Former rc_gpu.S).
```

```
[    8.525489] pwm-backlight backlight: supply power not found, using dummy
regulator
[    8.542029] debugfs: File 'Playback' in directory 'dapm' already present!
[    8.549420] debugfs: File 'Capture' in directory 'dapm' already present!
[    8.568932] caam-snvs 30370000.caam-snvs: ipid matched - 0x3e
[    8.574814] caam-snvs 30370000.caam-snvs: violation handlers armed - non-
secure state
[    8.597999] pwm-backlight backlight: supply power not found, using dummy
regulator
         Starting IPv6 Packet Filtering Framework...
         Starting IPv4 Packet Filtering Framework...
         Starting Telephony service...
[  OK  ] Started Parsec Service.
[  OK  ] Started     8.694521] pwm-backlight backlight: supply power not found,
using dummy regulator
;39mUpdates psplash to basic.
         Starting LSB: Run /etc/rc.local if it exist...
[  OK  ] Started System Logging Service.
         Starting User Login Management...
[  OK  ] Started TEE Supplicant.
         Starting OpenSSH Key Generation...
[  OK  ] Finished IPv6 Packet Filtering Framework.
[  OK  ] Finished IPv4 Packet Filtering Framework.
[  OK  ] Started LSB: Run /etc/rc.local if it exist.
[  OK  ] Finished OpenSSH Key Generation.
[  OK  ] Reached target Preparation for Network.
[  OK  ] Reached target Hardware activated USB gadget.
         Starting Network Configuration...
[  OK  ] Created slice Slice /system/systemd-fsck.
[  OK  ] Found device /dev/mmcblk2p1.
         Starting File System Check on /dev/mmcblk2p1...
[  OK  ] Started User Login Management.
[  OK  ] Finished File System Check on /dev/mmcblk2p1.
         Mounting /run/media/boot-mmcblk2p1...
[  OK  ] Mounted /run/media/boot-mmcblk2p1.
[  OK  ] Started Network Configuration.
         Starting Wait for Network to be Configured...
         Starting Save/Restore Sound Card State...
[  OK  ] Finished Load/Save OS Random Seed.
[  OK  ] Started D-Bus System Message Bus.
[  OK  ] Finished Save/Restore Sound Card State.
[  OK  ] Reached target Sound Card.
         Starting Connection service...
[  OK  ] Started Telephony service.
[  OK  ] Started Connection service.
[  OK  ] Reached target Network.
         Starting Avahi mDNS/DNS-SD Stack...
         Starting containerd container runtime...
[  OK  ] Started NFS status monitor for NFSv2/3 locking..
         Starting Network Time Service...
[  OK  ] Started Update psplash to network.
         Starting Terminate Psplash Boot Screen...
         Starting /etc/rc.local Compatibility...
         Starting Permit User Sessions...
[  OK  ] Started Network Time Service.
[FAILED] Failed to start Terminate Psplash Boot Screen.
See 'systemctl status psplash-quit.service' for details.
[  OK  ] Started /etc/rc.local Compatibility.
[  OK  ] Finished Permit User Sessions.
[  OK  ] Started Avahi mDNS/DNS-SD Stack.
[  OK  ] Started Getty on tty1.
```

```
[  OK  ] Started Serial Getty on ttymxc1.
[  OK  ] Reached target Login Prompts.
[   10.411165] audit: type=1334 audit(1677845637.212:6): prog-id=9 op=LOAD
[   10.417919] audit: type=1334 audit(1677845637.216:7): prog-id=10 op=LOAD
         Starting Hostname Service...
         Starting Weston, a Wayland ositor, as a system service...
         Starting WPA supplicant...
[   10.516960] audit: type=1334 audit(1677845637.316:8): prog-id=11 op=LOAD
[   10.524714] audit: type=1334 audit(1677845637.324:9): prog-id=12 op=LOAD
         Starting User Database Manager...
[  OK  ] Started WPA supplicant.
[  OK  ] Started User Database Manager.
[  OK  ] Started Hostname Service.
[  OK  ] Created slice User Slice of UID 0.
         Starting User Runtime Directory /run/user/0...
[  OK  ] Started containerd container runtime.
[  OK  ] Finished User Runtime Directory /run/user/0.
         Starting User Manager for UID 0...
[   10.954486] audit: type=1006 audit(1677845637.752:10): pid=591 uid=0 old-
auid=4294967295 auid=0 tty=(none) old-ses=4294967295 ses=1 res=1
[   10.966972] audit: type=1300 audit(1677845637.752:10): arch=c00000b7
syscall=64 success=yes exit=1 a0=8 a1=ffffc1159540 a2=1 a3=1 items=0 ppid=1
pid=591 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0
tty=(none) ses=1 comm="(systemd)" exe="/lib/systemd/systemd" key=(null)
[  OK  ] Started User Manager for UID 0.
[  OK  ] Started Session c1 of User root.
[FAILED] Failed to start Weston, a  mpositor, as a system service.
See 'systemctl status weston.service' for details.
[   13.695563] fec 30be0000.ethernet eth0: Link is Up - 1Gbps/Full - flow
control rx/tx
[   13.703372] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready

NXP i.MX Release Distro 6.1-mickledore imx8mmevk ttymxc1

imx8mmevk login: root
root@imx8mmevk:~#
root@imx8mmevk:~#
root@imx8mmevk:~# ls
fw_loader_imx_lnx                          ot-ctl     spi-hdlc-adapter
uartspi_n61x_v1.bin.se
imx8mm-evk-iw612-evk-6-1-36-new-gen.dtb  ot-daemon  tmp
imx8mm-evk-iw612-evk.dtb                   ot.log1    uart
root@imx8mmevk:~#
root@imx8mmevk:~# ./fw_loader_imx_lnx /dev/ttymxc2 115200 0
uartspi_n61x_v1.bin.se 3000000
Protocol: NXP Proprietary
FW Loader Version: M322
ComPort : /dev/ttymxc2
BaudRate: 115200
FlowControl: 0
Filename: uartspi_n61x_v1.bin.se
Second BaudRate: 3000000

ChipID is : 7601, Version is : 0
File downloaded:        0:  252104File downloaded:       16:  252104File
downloaded:     2064:  252104File downloaded:     2080:  252104File downloaded:
4128:  252104
…
File downloaded:   249760:  252104File downloaded:   251808:  252104File
downloaded:   251824:  252104File downloaded:   252088:  252104File downloaded:
252104:  252104
```

```
Download Complete
time:1470
CTS is low
root@imx8mmevk:~#
root@imx8mmevk:~# i2cdetect -y 2
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: UU 21 -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
root@imx8mmevk:~# i2cset -y 2 0x21 0x03 0xfe
root@imx8mmevk:~# i2cget -y 2 0x21 0x03
0xfe
root@imx8mmevk:~# i2cset -y 2 0x21 0x01 0x01
root@imx8mmevk:~# i2cget -y 2 0x21 0x01
0x01
root@imx8mmevk:~# ./ot-daemon -v "spinel+spi:///dev/spidev1.0?gpio-reset-
device=/dev/gpiochip5&gpi o-int-device=/dev/gpiochip5&gpio-int-line=12&gpio-
reset-line=13&spi-mode=0&spi-speed=1000000&spi-reeset-delay=500" -d 5  >
ot.log1 2>&1 &
[1] 624
root@imx8mmevk:~# ./ot-ctl version
OPENTHREAD/0.01.00; POSIX; Jan 13 2023 01:31:45
Done
root@imx8mmevk:~#
```

16 Appendix C: 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

Acronym	Meaning
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

17 Technical Support Contact

Table 26 lists all the support resources available for the Murata Wi-Fi/BT solution.

Table 26: 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.

18 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.

18.1 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.

18.2 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.

18.3 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.

18.4 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.

18.5 Murata uSD-M.2 Adapter Datasheet (Rev C)

This datasheet documents the Rev C version of the Murata's latest uSD-M.2 adapter hardware and its interfacing options.

18.6 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.

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

This datasheet documents the Rev B1 version of the Murata' latest uSD-M.2 adapter hardware and its interfacing options.

18.8 Embedded Artists' Reference Documentation

Embedded Artists designed the 2DS/1XK/1ZM/1YM/1XL/2XS 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 27 lists some relevant documents published by Embedded Artists.

Table 27: 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/2XS M.2 Module Datasheet	Comprehensive details on 1XL/2XS Wi-Fi/BT M.2 Module.
2DS M.2 Module Datasheet	Comprehensive details on 2DS Wi-Fi M.2 Module.
2DL/2EL M.2 Module Datasheet	Comprehensive details on 2DL/2EL Wi-Fi M.2 Module.

18.9 Murata Linux Regulatory 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.

18.10 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.

18.11 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 28 provides the following information on all releases supported:

Table 28: NXP Reference Documentation Listing

Kernel release	NXP documentation link	Yocto name	Release information
5.15.32_2.0.0	Rev. L5.15.32_2.0.0_BSP	Kirkstone	imx-kirkstone-5-15-32_r1.1
6.1.1_1.0.0	Rev. L6.1.1_1.0.0_BSP	Langdale	imx-langdale-6-1-1_r1.1
6.1.36_2.1.0	Rev. L6.1.36_2.1.0_BSP	Mickledore	imx-mickledore-6-1-36_r1.1
6.6.3_1.0.0	Rev. L6.6.3_1.0.0_Docs	Nanbield	imx-nanbield-6-6-3_r1.1
6.6.23_2.0.0	Rev. L6.6.23_2.0.0_Docs	Scarthgap	imx-scarthgap-6-6-23_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	Updated to template 1.0. Added support for 2DS.
3.0	Oct 20, 2022	TF	Added support for 2XK, 1XL, 2XS.
4.0	Nov 23, 2022	TF	Updated to template 2.0.
4.1	Jan 24, 2023	TF	Moved 'Section 11: Murata's Regulatory Solution for NXP Modules' to before 'Section 6: Murata Wi-Fi/BT Bring-Up on i.MX 6 Platforms', and updated Section/Table references.
4.2	Feb 08, 2023	TF	Added support for Linux 5.15.32.
4.3	Feb 15, 2023	TF	Added Table 8 (NXP Default Demo Images) and Table 10 (Linux Commands to Flash SD Card)
4.4	Mar 31, 2023	TF	Updated Murata's Regulatory Solutions for NXP Modules as per latest changes.
4.5	Jul 20, 2023	TF	Added support for Linux 6.1.1. Added support for 2DL, 2EL, 1XL-SDIO, 2XS-SDIO.
4.6	Nov 01, 2023	TF	Added support for Linux 6.1.36.
4.7	Nov 16, 2023	TF	Added rework instructions for Type 2EL M.2 EVB for NXP i.MX 8M Mini EVK.
4.8	Nov 30, 2023	TF	Added support for 802.15.4 interface
4.9	Jan 31, 2024	TF	Added 1XL/2XS production process number differences.
4.10	May 16, 2024	TF	Added steps to set up soft AP. Updated EU country code.
4.11	Jul 03, 2024	TF	Updated Regulatory solution as per NXP changes
4.12	Jul 12, 2024	TF	Updated build script usage, 2EL dual antenna configuration driver load. Added support for Linux 6.6.3, new EVKs.
4.13	Aug 23, 2024	TF	Added support for Rev C uSD-M.2 Adapters.
4.14	Sep 19, 2024	TF	Removed legacy kernels' support.
4.15	Oct 30, 2024	TF	Updated regulatory solution sections
4.16	Jan 10, 2025	TF	Updated regulatory solution sections.
4.17	Feb 06, 2025	TF	Added Murata Community Forum section. Added Firmware Options section.