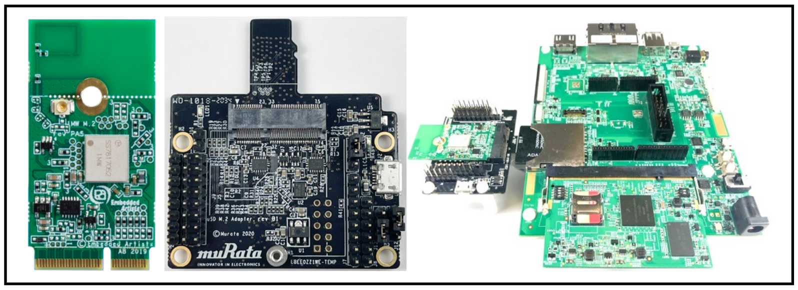
E2B-07-1325

Murata Wi-Fi/BT (CYW) Solution for i.MX

Linux User Guide



Revision History

Revision	Date	Author	Change Description
1.0	Nov 17, 2020	TF	Initial Release. Added support for i.MX Linux Kernel version 4.14.98, 5.4.47, and Type 1XA. Explained Embedded Artists' hardware/software solution for evaluating processors and EVBs. NOTE: Material moved from previous Quick Start Guide.
1.1	Jan 28, 2021	TF	Minor corrections, updated links and some photos.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

This page left intentionally blank.

1 Introduction

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

- 5.4.47_2.2.0
- 4.14.98_2.3.0
- 4.9.123_2.3.0
- 4.1.15_2.0.0

Note that for each kernel version, multiple Cypress "fmac" WLAN driver versions are supported. This document describes the following steps:

- How to build Murata's customized image and flash it to various NXP i.MX EVK's. This image enables Cypress "fmac" WLAN driver (while disabling previous legacy drivers). It also enables/configures associated components such as WPA supplicant, Hostapd, WLAN firmware/regulatory/NVRAM files, Bluetooth patch files, etc.
- Connect and/or configure applicable Wi-Fi/BT solution for a given NXP i.MX EVK & power up.
- Initialize & configure WLAN and Bluetooth interfaces.
- Exercise WLAN and Bluetooth functionality.

The NXP Platforms currently supported are based on i.MX 8, i.MX 7 and i.MX 6 with some Cypress-based Murata modules soldered down. However, in most cases the wireless solution is provided by connecting Embedded Artists' Wi-Fi/BT M.2 EVB directly to the NXP i.MX EVK; or using Murata's uSD-M.2 Adapter as an interconnect solution. Refer to **Table 1: Current NXP i.MX Platforms Supported** for more details.

Note that some of the NXP i.MX Reference platforms have restricted hardware interfaces (i.e. i.MX 7Dual SDB) which do not permit additional Wi-Fi/BT interconnect options. Also, all the NXP i.MX 6 family EVK's (except i.MX 6UL and i.MX 6ULL) only provide 3.3V VIO signaling option. This limits the interfacing of specific Wi-Fi/BT M.2 EVB's which operate at the non-standard 3.3V VIO signaling level (i.e. currently only 1DX/1MW M.2 EVB's support 3.3V VIO on WLAN-SDIO interface).

Table 1: Current NXP i.MX Platforms Supported

NXP i.MX EVK Part number	NXP i.MX EVK	Murata modules supported	Inter-Connect
MCIMX8QXP-CPU	i.MX 8QuadXPlus MEK	1CX, 1XA	M.2
MCIMX8M-EVKB	i.MX 8MQuad EVK	1CX, 1XA	1CX Soldered down; 1XA via M.2
8MMINILPD4-EVK	i.MX 8M Mini EVK	1DX, 1MW, 1LV, 1CX, 1XA	1DX, 1MW, 1LV via uSD-M.2 Adapter; 1CX, 1XA via M.2 (WLAN Only)
8MMINID4-EVK[1]	i.MX 8M Mini EVK	1MW, 1CX, 1XA	1MW Soldered down; 1CX or 1XA via M.2 (WLAN Only)
8MNANOD4-EVK	i.MX 8M Nano EVK	1DX, 1MW, 1LV	1MW Soldered down; 1DX, 1MW, 1LV via uSD-M.2 Adapter
MCIMX7SABRE	i.MX 7Dual SDB	ZP[2]	ZP Soldered down
MCIMX7ULP-EVK	i.MX 7ULP EVK	1DX	1DX Soldered down
MCIMX6QP-SDB	i.MX 6QuadPlus SDB	1DX, 1MW	uSD-M.2 Adapter
MCIMX6Q-SDB	i.MX 6Quad SDB	1DX, 1MW	uSD-M.2 Adapter
MCIMX6SX-SDB	i.MX 6SX SDB	1DX, 1MW	uSD-M.2 Adapter
MCIMX6UL-EVKB	i.MX 6UL EVK	1DX, 1MW, 1LV	uSD-M.2 Adapter
MCIMX6ULL-EVK	i.MX 6ULL EVK[3]	1DX, 1MW, 1LV	uSD-M.2 Adapter

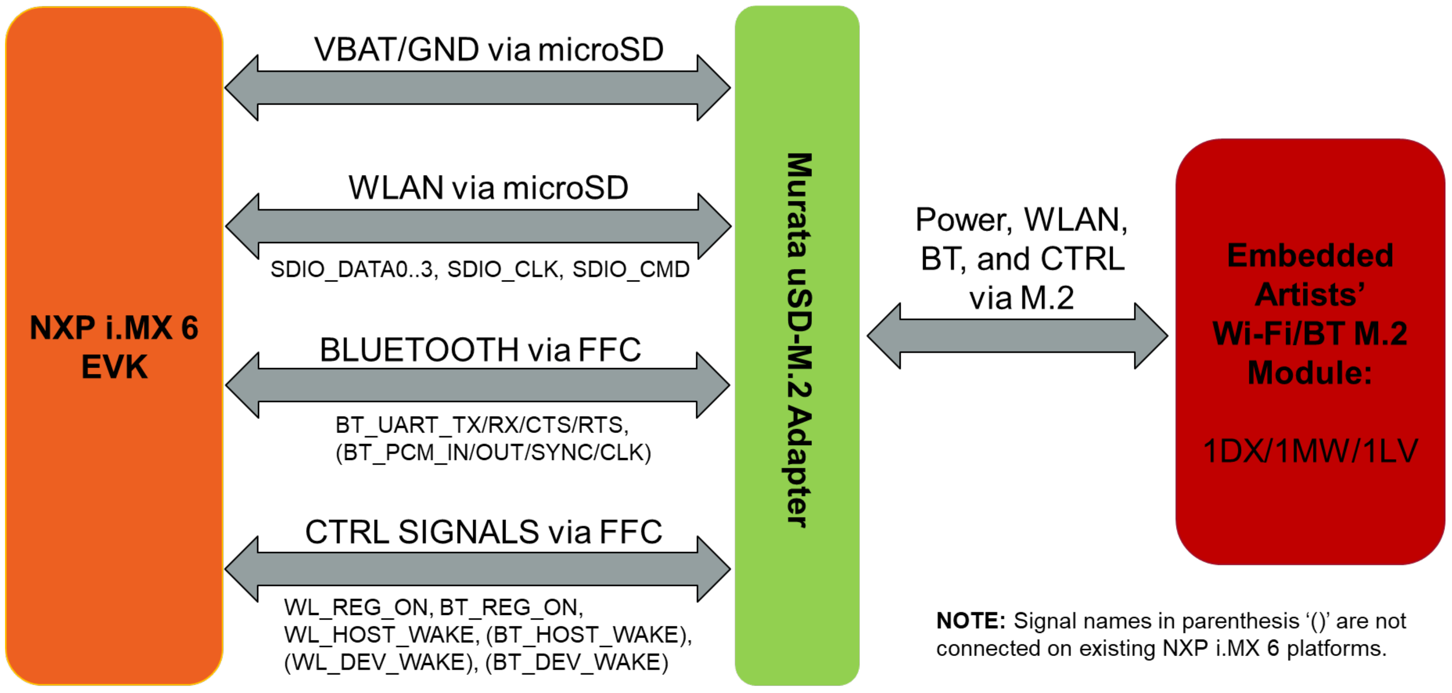
[1] Note that the 8MMINID4-EVK variant has NAND flash (not eMMC like 8MMINILPD4-EVK). Given limited NAND flash support, Murata does not support the Wi-Fi/BT uSD-M.2 interconnect option on this platform. However, the 8MMINID4-EVK does have Type 1MW module soldered down. Lastly both i.MX 8M Mini EVK's only have WLAN-PCIe interconnect on the M.2 connector – no Bluetooth-UART interconnect is available.
[2] This is a legacy module and no longer promoted. Use Type 1MW as suggested replacement.
[3] i.MX 6ULL is used for evaluation of i.MX 6ULZ.

1.1 NXP i.MX 6 Platform Support

A high-level connection Diagram for the Murata uSD-M.2 Adapter (i.MX 6 platforms) is provided in **Figure 1.** All the Murata Wi-Fi/BT modules enabled by this release are shown. The wireless solution is arrived at by combining Murata's uSD-M.2 Adapter with Embedded Artists' Wi-Fi/BT M.2 EVB. Refer to **Section 8** and **Section 9** for more details on the uSD-M.2 Adapter and Wi-Fi/BT M.2 Modules. Note that Murata ***no longer supports*** the legacy i.MX V1/V2 Interconnect Kit which used 60-pin Samtec connectors. Murata has collaborated very closely with Embedded Artists to arrive at the new Wi-Fi/BT M.2 EVB (Module) solution. As evident from the Embedded Artists' documentation, the M.2 EVB is optimized for evaluation with the following features:

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

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



NXP i.MX 6 EVK
VBAT/GND via microSD
WLAN via microSD
SDIO_DATA0..3, SDIO_CLK, SDIO_CMD
BLUETOOTH via FFC
BT_UART_TX/RX/CTS/RTS, (BT_PCM_IN/OUT/SYNC/CLK)
CTRL SIGNALS via FFC
WL_REG_ON, BT_REG_ON, WL_HOST_WAKE, (BT_HOST_WAKE), (WL_DEV_WAKE), (BT_DEV_WAKE)
Murata uSD-M.2 Adapter
Power, WLAN, BT, and CTRL via M.2
Embedded Artists' Wi-Fi/BT M.2 Module:
1DX/1MW/1LV
NOTE: Signal names in parenthesis '()' are not connected on existing NXP i.MX 6 platforms.

1.2 NXP i.MX 8 Platform Support

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

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

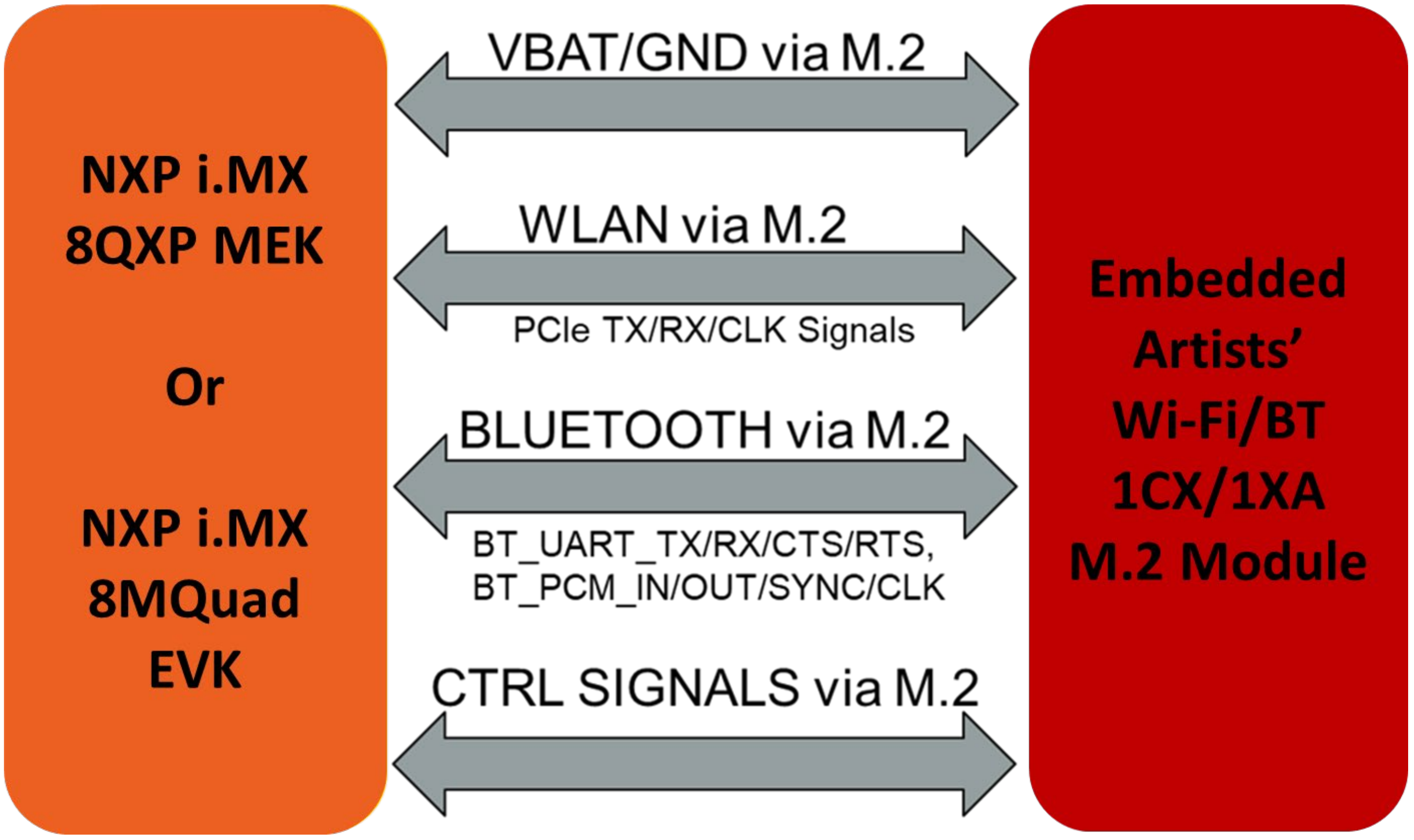


Figure 3 shows a simplified block diagram for the i.MX 8M Mini EVK Wi-Fi/BT interconnect with onboard M.2 EVB option. Currently, Type 1CX and 1XA M.2 Modules (WLAN Only) are supported with 2x2 802.11ac MIMO and WLAN-PCIe interface. Note that the NXP i.MX 8M Mini EVK **does not bring out** the Bluetooth signals to the M.2 connector – **WLAN only**.

Figure 4 shows a simplified block diagram for the i.MX 8M Mini/Nano EVK interconnect with Murata's uSD-M.2 Adapter option (with Embedded Artists' Wi-Fi/BT M.2 EVB). Only WLAN-SDIO based modules are supported in this configuration: 1MW, 1DX and 1LV. To properly support this (WLAN-SDIO VIO @1.8V; BT-UART VIO @3.3V) interconnect, Rev B1 uSD-M.2 Adapter needs to be used given that it correctly level shifts the Bluetooth and WLAN/BT control signals between the M.2 EVB and the NXP i.MX 8M Mini EVK.

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

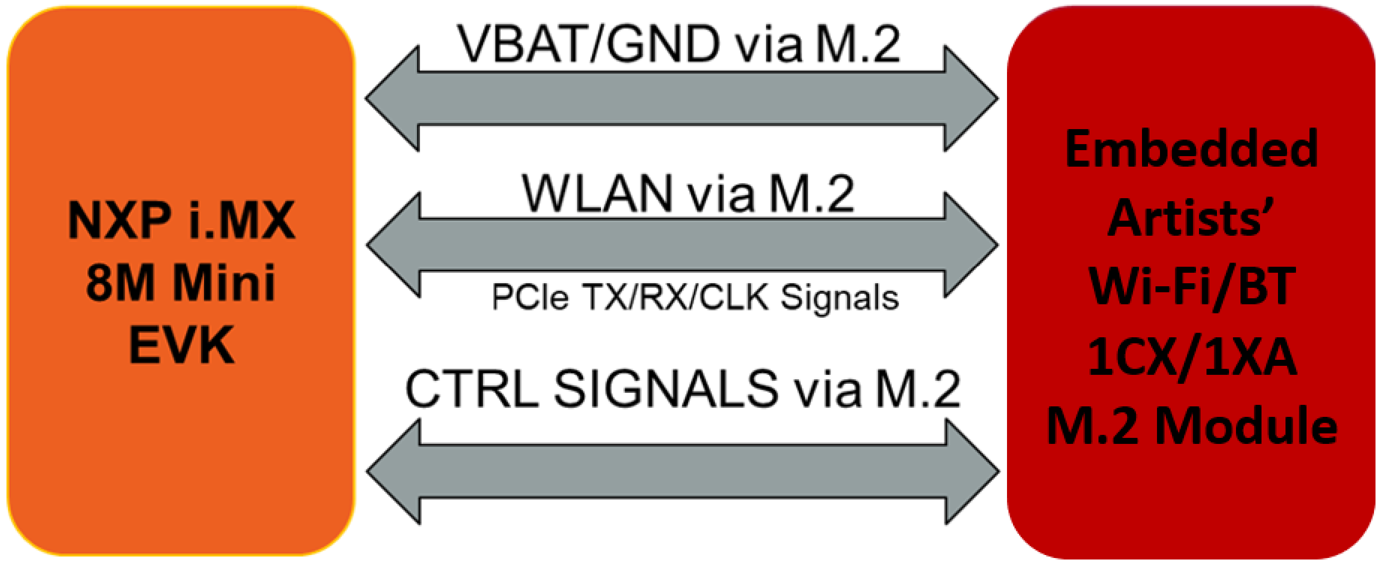
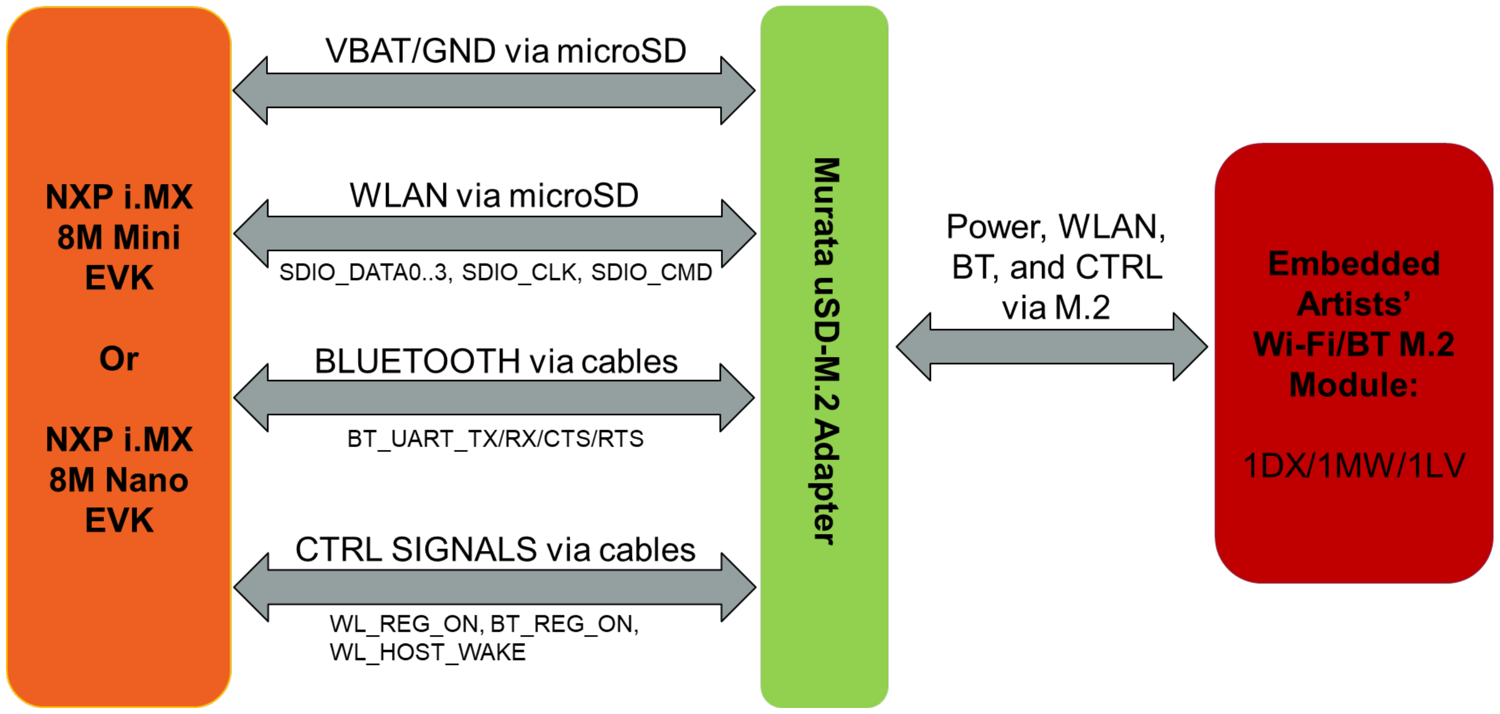


Figure 4: i.MX 8M Mini/Nano EVK Wi-Fi/BT SDIO Interconnect Block Diagram



1.3 Acronyms

Table 2: Acronyms used in User Guide

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

1.4 References

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

1.4.1 Murata Wi-Fi/BT (CYW) Solution for i.MX Linux User Manual

This manual describes all steps necessary to build the file system, kernel, DTB files, and WLAN "fmac" driver necessary for supporting NXP i.MX Platforms and the Murata Wi-Fi/BT EVK.

1.4.2 Murata Wi-Fi/BT (CYW) Solution for i.MX Linux Quick Start Guide

This Quick Start Guide provides quick steps to get started with Murata Wi-Fi/BT Cypress chipset-based solution with the help of an example.

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

This manual describes the Murata uSD-M.2 Adapter hardware. All interface signals to the NXP i.MX RT, 6, 7, and 8 EVK's are described. Specifics on interfacing each i.MX EVK to Murata uSD-M.2 Adapter are provided.

1.4.4 Murata's Community Forum Support

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

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

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

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

This datasheet documents the current version of the Murata's legacy uSD-M.2 adapter hardware and its interfacing options. This adapter version is no longer manufactured.

1.4.7 Embedded Artists' Reference Documentation

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

1.4.8 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.

Table 3: Embedded Artists Documentation Listing

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

1.4.9 NXP Reference Documentation

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

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

Table 4 provides the following information on all releases supported:

- Kernel version
- Documentation link(s)
- Corresponding Yocto release name
- Supported "fmac" release name(s)

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

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

Table 4: NXP Reference Documentation Listing

Kernel release	NXP documentation link	Yocto code name	“fmac” code name	Release information
5.4.47_2.2.0	Rev. L5.4.47_2.2.0_BSP	Zeus	Zigra	imx-zeus-zigra_r1.0
4.14.98_2.3.0	Rev. L4.14.98_2.3.0_BSP	Sumo	Zigra Kong Manda	imx-sumo-zigra_r1.0 imx-sumo-kong_r1.1 imx-sumo-manda_r1.2
4.9.123_2.3.0	Rev. L4.9.123_2.3.0_8MMini_GA	Rocko-Mini	Zigra Kong Manda	imx-rocko-mini-zigra_r1.0 imx-rocko-mini-kong_r1.0 imx-rocko-mini-manda_r2.2
4.1.15_2.0.0	Rev. L4.1.15_2.0.0_BSP	Krogoth	Zigra Kong Manda	imx-krogoth-zigra_r1.0 imx-krogoth-kong_r1.0 imx-krogoth-manda_r2.2

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

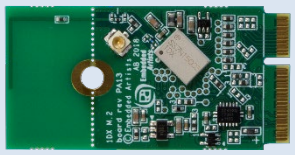
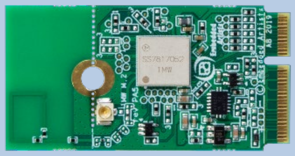
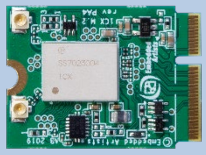
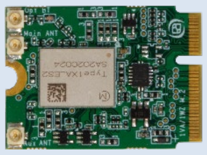
As already outlined in the **Introduction,** the Murata Wi-Fi/BT solution is primarily arrived at using Embedded Artists’ Wi-Fi/BT M.2 EVB’s in conjunction with Murata’s uSD-M.2 Adapter. This section provides additional details on the hardware solution.

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

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

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

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

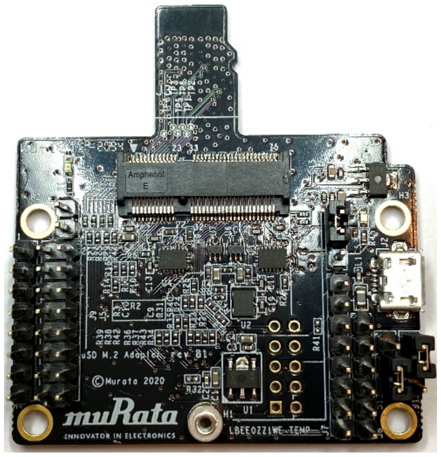
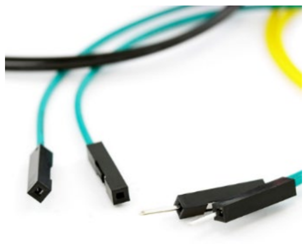
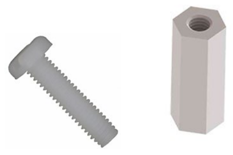
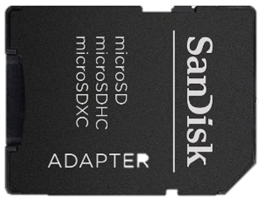
Murata Module	Chipset	Wi-Fi/BT Support	Murata Part Number	EA M.2 Module Part Number	VIO	Interface	EVB Picture
1DX	CYW 4343W	802.11b/g/n BT/BLE 5.1	LBEE5KL1DX	EAR00318	1.8V/ 3.3V	WLAN: SDIO Bluetooth: UART	
1MW	CYW 43455	802.11a/b/g/n/ac (1x1 SISO) BT/BLE 5.0	LBEE5HY1MW	EAR00315	1.8V/ 3.3V	WLAN: SDIO Bluetooth: UART	
1LV	CYW 43012	802.11 a/b/g/n/ac-friendly BT/BLE 5.0	LBEE59B1LV	EAR00323	1.8V only	WLAN: SDIO Bluetooth: UART	
1CX	CYW 4356	802.11a/b/g/n/ac (2x2 MIMO) BT/BLE 5.0	LBEH5UL1CX	EAR00321	1.8V/ 3.3V	WLAN: PCIe Bluetooth: UART	
1XA	CYW 54591	802.11a/b/g/n/ac (2x2 MIMO; RSDB) BT/BLE 5.1	LBEE5XV1XA	EAR00373	1.8V/ 3.3V	WLAN: PCIe Bluetooth: UART	

2.2 Murata's uSD-M.2 Adapter

The Wi-Fi/BT solution for NXP i.MX 6 and some i.MX 8 Platforms requires the use of a Murata uSD-M.2 Adapter Kit in conjunction with Embedded Artists' Wi-Fi/BT M.2 Module.

The Murata uSD-M.2 Adapter Kit (Part No: **LBEE0ZZ1WE-TEMP**) contents are shown in **Table 6**. Note that there are two versions of the uSD-M.2 Adapter: Rev A and Rev B1. Currently only Rev B1 is available through Distribution channel. It is backwards compatible to Rev A version; but has some important updates that include level shifting for Bluetooth UART and WLAN/Bluetooth control signals. This document differentiates important jumper settings on both the newest Rev B1 and original Rev A Adapters.

Table 6: uSD-M.2 Adapter Kit Contents

Picture of Contents	Description of Contents
	Murata uSD-M.2 Adapter (Revision B1) Part Number: **LBEE0ZZ1WE-TEMP**
	M.2 screw for attaching Wi-Fi/Bluetooth M.2 Module
	75mm 20-pos, 0.5mm pitch flat/flex cable
	13 pieces 200mm long male-to-female jumper cables (compatible with Arduino header)
	4 x 19mm stand-offs in nylon and associated M3 screws
	microSD to SD Card Adapter

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

2.3 NXP i.MX versus Murata Module Interconnect

Table 7 shows Murata module interconnect on various NXP i.MX Reference Platforms. Cypress 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 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 EVK's.
- "**uSD-M.2^{+}**": Murata's uSD-M.2 Adapter (Rev B1) provides interconnect to the Embedded Artists' Wi-Fi/BT M.2 Module. However, additional cabling to connect Bluetooth UART and WLAN/BT control signals is required for NXP i.MX 8M Mini EVK and 8M Nano EVK. The cable (Jumper Wire F/F 6") is easily obtained through Distribution channel (example Digi-Key part numbers 1568-1644-ND or 1568-1513-ND).
- "**uSD-M.2-3.3V**": Murata's uSD-M.2 Adapter provides interconnect to the Embedded Artists' Wi-Fi/BT M.2 Module. NXP i.MX Host WLAN SDIO is configured for (fixed at) 3.3V VIO. The Bluetooth UART, and WLAN/BT control signals from i.MX Host are configured at 3.3V VIO. Either Rev B1 or Rev A uSD-M.2 Adapter can be used in this case – with correct jumper setting for 3.3V override mode.
- "**M.2**": Only Embedded Artists' Wi-Fi/BT M.2 EVB is required. The NXP i.MX EVK has a M.2 connector. Currently only WLAN-PCIe, BT-UART, and WLAN/BT CTRL signals are brought out on the M.2 connectors for these specific NXP i.MX EVK's. As such, ***only*** Wi-Fi/BT M.2 EVB's which support WLAN-PCIe (i.e. 1CX and 1XA) can be used.
- "**M.2^{W}**": Only Embedded Artists' Wi-Fi/BT M.2 EVB is required. The NXP i.MX EVK has a M.2 connector. Currently only WLAN-PCIe, and WLAN CTRL signals are brought out on the M.2 connectors for these specific NXP i.MX EVK's. As such, there is no Bluetooth support – only WLAN.

NOTE: When using uSD-M2 adapter, there are limitations on maximum SDIO clock frequency. For UHS mode support (i.e. MAX SDIO clock is 200 MHz for Type 1MW) and for comprehensive signal support, ***Murata recommends*** the Embedded Artists' i.MX Developer Kits.

Table 7: NXP i.MX/Murata Module Interconnect

NXP i.MX EVK Part Number	1DX	1MW	1LV	1CX	1XA
	CYW4343W	**CYW43455**	**CYW43012**	**CYW4356**	**CYW54591**
MCIMX8QXP-CPU	NC	NC	NC	M.2	M.2
MCIMX8M-EVKB	NC	NC	NC	MD, M.2	M.2
8MMINILPD4-EVK	uSD-M.2$^+$	uSD-M.2$^+$	uSD-M.2$^+$	M.2^W	M.2^W
8MMINID4-EVK[4]	NC	MD	NC	M.2^W	M.2^W
8MNANOD4-EVK	uSD-M.2$^+$	MD, uSD-M.2$^+$	uSD-M.2$^+$	NC	NC
MCIMX7ULP-EVK	MD	NC	NC	NC	NC
MCIMX6QP-SDB	uSD-M.2-3.3V	uSD-M.2-3.3V	NC	NC	NC
MCIMX6Q-SDB	uSD-M.2-3.3V	uSD-M.2-3.3V	NC	NC	NC
MCIMX6SX-SDB	uSD-M.2-3.3V	uSD-M.2-3.3V	NC	NC	NC
MCIMX6UL-EVKB	uSD-M.2	uSD-M.2	uSD-M.2	NC	NC
MCIMX6ULL-EVK	uSD-M.2	uSD-M.2	uSD-M.2	NC	NC
MCIMX7SABRE	ZP soldered down. No other modules supported.				

uSD-M.2 = works with uSD-M.2 Adapter configured for 1.8V VIO default

uSD-M.2$^+$ = works with uSD-M.2 Adapter (Rev B1) with additional cabling

uSD-M.2-3.3V = works with uSD-M.2 Adapter configured for 3.3V VIO override mode

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

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

MD = Murata Module is soldered down

NC = No Connection options available

[4] Note that the 8MMINID4-EVK variant has NAND flash (not eMMC like 8MMINILPD4-EVK). Given limited NAND flash support, Murata does not support the Wi-Fi/BT uSD-M.2 interconnect option on this platform.

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

3.1 “fmac” Solution Overview

Previous NXP i.MX Kernels integrated the legacy “bcmdhd” WLAN driver. NXP later integrated the more relevant “***fmac***” driver into their baseline BSP release. However, the NXP BSP-integrated “***fmac***” driver never fully incorporated Cypress’ formal “***fmac***” release.

There is also a ***distinct difference*** between the “***fmac***” driver documented here; and the ***“brcmfmac”*** open-source community drivers integrated into kernel.org Linux releases. The ***“fmac”*** driver (as customized by Murata) is an official open-source release from Cypress that is tested and verified. The Cypress ***“fmac”*** release leverages the Linux Backports implementation to integrate the WLAN driver into the desired Linux kernel version.

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

- Pull Cypress ***“fmac”*** driver and run backports tool during Yocto build to generate necessary driver modules.
- Additional/necessary patches to Cypress ***“fmac”*** driver for i.MX implementation.
- i.MX Linux kernel customizations to support ***“fmac”*** driver with OOB IRQ interrupts.
- Support 1.8V VIO signaling with NXP i.MX6UL(L) EVK.
- Support Wi-Fi/BT enablement on microSD slot of NXP i.MX 8M Mini/Nano EVK’s.
- Fine tune DTS files so that optimal WLAN-SDIO throughput is achieved.
- WLAN production firmware files. For manufacturing test firmware (necessary for regulatory testing), please contact Murata directly. If no direct contact is available to you, then please post query to Murata’s Community Forum at https://community.murata.com.
- Murata NVRAM files for correctly configuring wireless module RF characteristics.
- Example Bluetooth patch files which allow customers to initial Bluetooth evaluation.
- WL tool binary necessary for interoperability and RF testing.
- Hostapd configuration (Cypress-specific version for a given “***fmac***” release) with specific Cypress’ patch release.
- Hostap-conf enablement.
- Hostap-utils enablement.
- WPA-supplicant configuration (Cypress-specific version for a given “fmac” release[5]) with Cypress’ specific patch release.
- Wi-Fi Direct (P2P) enablement.

There are three versions of ***“fmac”*** currently supported: ***“v4.14 manda”***, ***“v4.14 kong”*** and ***“v5.4 zigra”***. Note that ***“manda”***, ***“kong”*** and “***zigra***” are the Cypress codenames denoting ***“fmac”***

[5] Cypress “zigra” and “kong” WLAN driver releases use WPA supplicant and Hostapd version 2.9; whereas “manda” uses version 2.6.

release version. ***"v4.12"***, ***"v4.14"*** and ***"v5.4"*** are the latest kernel versions supported by the releases (can be backported to kernel version 3.0). To abbreviate references to specific versions of ***"fmac"***, Murata uses ***just*** the Cypress codename – i.e. ***"manda"***, ***"kong"*** or ***"zigra"***. It is strongly recommended to use the latest ***"fmac"*** release version: currently this is ***"zigra"***.

3.2 Specific i.MX Target Support Details

Murata's customized Yocto layer ("***meta-murata-wireless***") supports the following NXP i.MX EVK's as outlined in **Table 8**. ***"MACHINE=target"*** is a direct reference to Yocto build. The ***"target"*** string is the keyword used to select hardware configuration for the build (important knowledge when running Murata build script). With the newer EVK's (i.MX 8QXP, i.MX 8MQuad, i.MX 8M Mini, i.MX 8M Nano, and i.MX 7ULP) only certain kernel versions are supported. From this table, you can find a proper version of Linux Kernel for your target platform. You can then refer to **Table 9** for the support of interrupt configuration, corresponding DTB (Device Tree Blob) files, hardware interconnect configuration (either module onboard, uSD-M.2 Adapter & M.2 EVB, or just M.2 EVB), and if the hardware configuration is limited to 3.3V VIO on WLAN SDIO interface (default WLAN SDIO VIO is 1.8V). The hardware 3.3V VIO WLAN SDIO interface limitation only applies to certain legacy i.MX 6 platforms (6QP, 6Q, 6DL, 6SX). The 3.3V VIO signaling requires the uSD-M.2 Adapter to be configured in a "3.3V VIO Override mode". Not all Wi-Fi/BT M.2 EVB's support 3.3V VIO on WLAN-SDIO interface. **NOTE:** please refer to **Section 3.4.1** and/or the Hardware User Manual to ensure that i.MX EVK hardware configuration supports OOB IRQ signaling if that is your desired WLAN interrupt mode.

Table 8: NXP i.MX EVK / Yocto (MACHINE) target / Kernel Version Matrix

NXP i.MX EVK Part Number	NXP i.MX EVK	MACHINE=target	Kernel 4.1.15	Kernel 4.9.123	Kernel 4.14.98	Kernel 5.4.47
MCIMX8QXP-CPU	i.MX 8QuadXPlus MEK	imx8qxpmek	N	Y	Y	Y
MCIMX8M-EVKB	i.MX 8MQuad EVK	imx8mqevk	N	Y	Y	Y
8MMINILPD4-EVK	i.MX 8M Mini EVK	imx8mmevk	N	Y	Y	Y
8MMINID4-EVK	i.MX 8M Mini EVK	imx8mmddr4evk	N	Y	Y	Y
8MNANOD4-EVK	i.MX 8M Nano EVK	imx8mnddr4evk	N	N	Y	Y
MCIMX7SABRE	i.MX 7Dual SDB	imx7dsabresd	Y	Y	Y	Y
MCIMX7ULP-EVK	i.MX 7ULP EVK	imx7ulpevk	N	Y	Y	Y
MCIMX6QP-SDB	i.MX 6QuadPlus SDB	imx6qpsabresd	Y	Y	Y	Y
MCIMX6Q-SDB	i.MX 6Quad SDB	imx6qsabresd	Y	Y	Y	Y
	i.MX 6DualLite SDB	imx6dlsabresd	Y	Y	Y	Y
MCIMX6SX-SDB	i.MX 6SX SDB	imx6sxsabresd	Y	Y	Y	Y
MCIMX6UL-EVKB	i.MX 6UL EVK	imx6ulevk	Y	Y	Y	Y
MCIMX6ULL-EVK	i.MX 6ULL EVK	imx6ull14x14evk	Y	Y	Y	Y

Table 9: i.MX6/7/8 Targets supported by Murata

Target (MACHINE)	Hardware Config	i.MX DTB File	Interrupt Config	3.3V VIO SDIO
imx8qxpmek	M.2	fsl-imx8qxp-mek.dtb	N/A	N/A
imx8mqevk	MD (1CX)	fsl-imx8mq-evk.dtb	N/A	N/A
imx8mqevk	M.2	fsl-imx8mq-evk-pcie1-m2.dtb	N/A	N/A
imx8mmevk	M.2^{w}	imx8mm-evk.dtb	N/A	N/A
imx8mmevk	uSD-M.2^{+}	imx8mm-evk-usd-m2-oob.dtb	OOB	N
imx8mmevk	uSD-M.2^{+}	imx8mm-evk-usd-m2.dtb	SDIO	N
imx8mmddr4evk	MD (1MW)	imx8mm-ddr4-evk.dtb	OOB	N
imx8mmddr4evk	M.2^{w}	imx8mm-ddr4-evk.dtb	N/A	N/A
imx8mnddr4evk	MD (1MW)	imx8mn-ddr4-evk.dtb	OOB	N
imx8mnddr4evk	uSD-M.2^{+}	imx8mn-evk-usd-m2-oob.dtb	OOB	N
imx8mnddr4evk	uSD-M.2^{+}	imx8mn-evk-usd-m2.dtb	SDIO	N
imx7dsabresd	MD (ZP)	imx7d-sdb.dtb	OOB	N
imx7ulpevk	MD (1DX)	imx7ulp-evk.dtb	OOB	N
imx6qpsabresd	uSD-M.2-3.3V	imx6qp-sabresd-btwifi-m2-oob.dtb	OOB	Y
imx6qpsabresd	uSD-M.2-3.3V	imx6qp-sabresd-btwifi-m2.dtb	SDIO	Y
imx6qsabresd	uSD-M.2-3.3V	imx6q-sabresd-btwifi-m2-oob.dtb	OOB	Y
imx6qsabresd	uSD-M.2-3.3V	imx6q-sabresd-btwifi-m2.dtb	SDIO	Y
imx6dlsabresd	uSD-M.2-3.3V	imx6dl-sabresd-btwifi-m2-oob.dtb	OOB	Y
imx6dlsabresd	uSD-M.2-3.3V	imx6dl-sabresd-btwifi-m2.dtb	SDIO	Y
imx6sxsabresd	uSD-M.2-3.3V	imx6sx-sdb-btwifi-m2-oob.dtb	OOB	Y
imx6sxsabresd	uSD-M.2-3.3V	imx6sx-sdb-btwifi-m2.dtb	SDIO	Y
imx6ulevk	uSD-M.2	imx6ul-14x14-evk-btwifi-m2-oob.dtb	OOB	N
imx6ulevk	uSD-M.2	imx6ul-14x14-evk-btwifi-m2.dtb	SDIO	N
imx6ull14x14evk	uSD-M.2	imx6ull-14x14-evk-btwifi-m2-oob.dtb	OOB	N
Imx6ull14x14evk	uSD-M.2	imx6ull-14x14-evk-btwifi-m2.dtb	SDIO	N

uSD-M.2 = works with uSD-M.2 Adapter configured for 1.8V VIO default
uSD-M.2^{+} = works with uSD-M.2 Adapter (Rev B1) with additional cabling
uSD-M.2-3.3V = works with uSD-M.2 Adapter configured for 3.3V VIO override mode
M.2 = Wi-Fi/BT M.2 EVB plugs directly into M.2 connector
M.2^{w} = Wi-Fi/BT M.2 EVB plugs directly into M.2 connector; but only WLAN is functional
MD = Murata Module is soldered down

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

Murata's solution to easily enable Cypress-based Wi-Fi/Bluetooth functionality requires a complete Linux image build (bootloader, kernel, DTB files, filesystem). To understand this requirement, we need to understand specifics about the NXP i.MX Linux image. The NXP i.MX image contains third party IP which is sub-licensed via a click-through EULA (when either downloading a NXP i.MX validation/demo image or building the image from source). As such Murata cannot make this image available directly to customers. As detailed by the Murata Linux User Manual, Murata employs a wireless-enabling "meta-murata-wireless" layer to make this customized Yocto build simple and user-friendly. Nonetheless end users must still configure a Linux build environment and follow specific steps to arrive at the desired image for a given i.MX target and Murata WLAN/BT configuration. Murata has greatly simplified the build requirement by providing scripts for Ubuntu host setup and customized Yocto build – scripts easily downloadable from Murata's GitHub. Steps for downloading, configuring, and invoking these scripts are detailed here.

3.3.1 Install Ubuntu

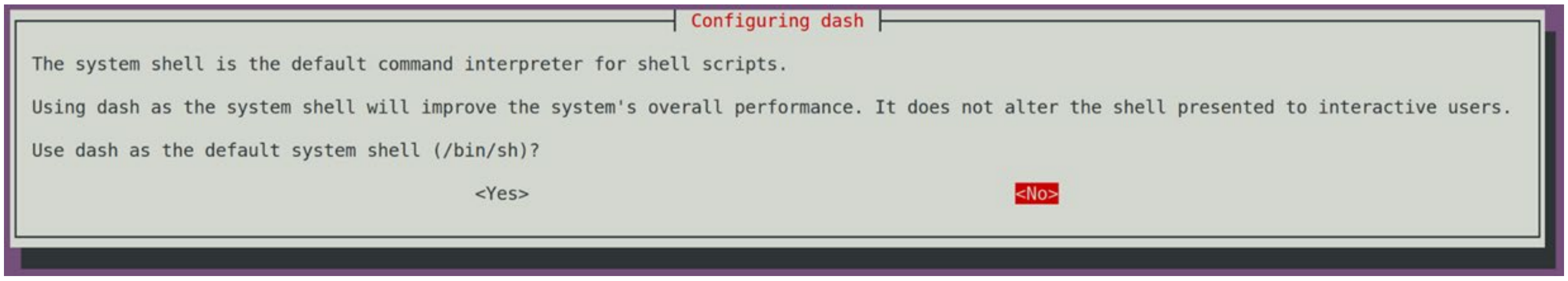
First step is to install Ubuntu 14.04, 16.04 or 18.04 (Murata's build is verified on Ubuntu 16.04 64-bit install) on the host - native PC or virtual environment like VMware. Host PC typically used requires Ubuntu 18.04/16.04/14.04 installed with 50 GB free disk space (80 GB needed for i.MX8 build). For more information on the Ubuntu download, please refer to this link: https://www.ubuntu.com/download/desktop. 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 5**.

Figure 5: Configuring dash



```
Configuring dash

The system shell is the default command interpreter for shell scripts.

Using dash as the system shell will improve the system's overall performance. It does not alter the shell presented to interactive users.

Use dash as the default system shell (/bin/sh)?

                    <Yes>                    <No>
```

3.3.2 Download Murata's Script Files

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

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

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

OR:

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

```
wget --no-check-certificate --content-disposition https://github.com/murata-wireless/meta-murata-
wireless/raw/master/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.sh

chmod a+x *.sh
```

3.3.3 Configure Ubuntu for i.MX Yocto Build

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

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

```
./Host_Setup_for_Yocto.sh
```

The script goes through the following stages:

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

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

3.3.4 Murata's i.MX Yocto Build Script

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

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

- Installed 64-bit version of Ubuntu 16.04 (preferred), 14.04, or 18.04.
- Ran Murata's host setup script in **Section 3.3.3** to add necessary packages for Yocto build and configure GIT.
- Created a i.MX BSP folder ***specific*** to the desired i.MX Yocto Release. The i.MX Yocto distribution **cannot** build different versions of Yocto (Linux kernel) in the same folder. Currently the following Yocto releases are supported:
    - 5.4.47_2.2.0 GA
    - 4.14.98_2.3.0
    - 4.9.123_2.3.0 GA
    - 4.1.15_2.0.0 GA

- Once the build script successfully completes, the i.MX BSP folder will contain:
    - Yocto "***sources***" and "***downloads***" folder.
    - "***meta-murata-wireless***" folder – is a sub-folder of "***sources***".
    - One or more i.MX build folders.

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

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

Murata's build script performs the following tasks:

- Verifies host environment (i.e. Ubuntu 14.04/16.04/18.04).
- Check to make sure script being run is the latest version.
- Prompts the user to select release type:
    - "**Stable**" corresponds to "***meta-murata-wireless***" release/tag (rather than a branch). Murata tests wireless functionality on i.MX platforms for each release/tag. This release type is recommended for baseline image builds or initial bring-up testing.
    - "**Developer**" corresponds to a branch which can be a "moving target". When performing the automated build, the script file pulls the latest branch contents – as opposed to a specific GIT commit on that branch. If the user wants the latest fixes, then this is the best option to go with. **NOTE**: Murata only runs "spot" tests before submitting fixes/enhancements to the branch. The "**Developer**" branch build may fail. In this case, the user is highly recommended to employ the "**Stable**" branch (formal tag release) and apply any necessary patches to it.
- Select i.MX Yocto release. As already pointed out the current i.MX BSP folder (from which script is being executed) can only support **one** i.MX Yocto release. If you need to test/evaluate different Yocto/kernel versions, then **you must** create additional folders.
- Select the wireless solution. For newer kernel releases, Murata provides Wi-Fi support for both Cypress and NXP chipset-based modules. For older kernels, this option is not provided as only Cypress based chipsets are supported.
- Select the "fmac" release. The script displays both the "fmac" codename and latest kernel version supported by that release. Currently, three "fmac" releases are supported: "***manda***", "***kong***" and "***zigra***" (most recent and up to date regarding fixes and enhancements). Murata strongly recommends using "***zigra***" release.
- Select i.MX target: refer to **Table 8** and **Table 9** 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/Freescale End User's License Agreement (EULA). There is 3rd party IP included in the i.MX Yocto build. This step addresses the sub-licensing issue. During this step, the user must review a fair bit of legal documentation (by repeatedly entering space bar) or if already familiar with the EULA language, enter 'q' to bypass displaying the complete agreement. The final step of this EULA step prompts the user to enter "y" to accept.
- Last and final step is to confirm that user wants to kick off the final build process (invoke "***bitbake*** **<image>**" command).

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

```
./Murata_Wireless_Yocto_Build.sh
```

The script goes through the following stages:

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

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

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

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

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

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

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

With the Linux image built and located, **Section 4.1** outlines flashing steps for (micro) SD card (on all platforms except 8MMINILPD4-EVK and 8MNANOD4-EVK). Steps for flashing eMMC on the i.MX 8M Mini/Nano EVK's are detailed in **Section 4.2**.

3.4 Additional Hardware/Software Considerations

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

The preferred interrupt configuration is OOB IRQ. Murata recommends that the user runs OOB IRQ (if possible) on all i.MX reference platforms. Otherwise, power-saving mechanisms are limited. Specifically, the host cannot shut down the SDIO bus when using SDIO in-band signaling for WLAN interrupt mechanism – having to continually poll for SDIO in-band interrupts instead. Also, WLAN throughput optimization is achieved with OOB IRQ configuration. By referring to **Table 9**, we can easily distinguish which legacy NXP i.MX 6 platforms require hardware rework to configure them for OOB IRQ signaling over WLAN bus. Any platform/target with "**SDIO**" as an option in the "**SDIO Interrupt Config**" column indicates that the default hardware configuration only supports SDIO in-band signaling. To be clear, we are flagging the following i.MX 6 Platforms which do not support WLAN OOB IRQ configuration out-of-box:

- **i.MX 6Q/DL SDB/SDP or i.MX 6 QP SDB**: all these platforms require rework to enable Bluetooth and WLAN/Bluetooth control signals. Refer to the Hardware User Manual for specifics.
- **i.MX 6SoloX SDB**: this platform will work with SDIO in-band signaling. However, to provide correct WL_REG_ON (WLAN core enable/disable) **OR** OOB IRQ support, rework must be done. Refer to the Hardware User Manual for specifics.
- **i.MX 6UL(L) EVK's**: these platforms require one resistor to be move to support OOB IRQ support. **NOTE:** For OOB IRQ configuration on i.MX6UL/ULL EVK, the second (of two) Ethernet ports is disabled due to hardware conflict (documented in Hardware User Manual).

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

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

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

As shown in **Table 7** and **Table 9**, specific NXP's i.MX 6 Platforms are limited to 3.3V VIO signaling over WLAN-SDIO when using the uSD-M.2 Adapter. This is due to NXP platform hardware limitations – fixed voltage rails. All i.MX 6 Platforms ***except*** i.MX 6UL(L) EVK's have this limitation. When the Adapter is configured in "3.3V VIO Override Mode", all signals connected to the Wi-Fi/BT M.2 EVB are at 3.3V VIO level. The uSD-M.2 Adapter must be jumpered specifically for this configuration; and only some of the Wi-Fi/BT M.2 EVB's support this non-default voltage signaling. Currently only Type 1DX and Type 1MW M.2 EVB's (WLAN-SDIO) support this 3.3V VIO override mode configuration. Note that Type 1LV M.2 EVB is 1.8V VIO only and thereby cannot run in this configuration.

3.4.3 UHS SDIO 3.0 operation on i.MX 6 Platforms with uSD-M.2 Adapter

When using Murata's uSD-M2 adapter to interconnect the Wi-Fi/BT M.2 EVB to a NXP i.MX 6 platform, the maximum SDIO clock frequency is limited to 50MHz for both 1.8V and 3.3V VIO. However, there is no such limitation with the NXP i.MX 8M Mini and 8M Nano EVK's which support a direct microSD connect – the i.MX 6 platforms require the microSD-to-SD Adapter. For UHS mode (and better overall hardware/software) support, ***Murata strongly recommends*** the Embedded Artists' i.MX Developer Kits. See Embedded Artists' Solution section for more details.

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

As shown in **Table 7** and **Table 9**, NXP's i.MX 8 Family of EVK's 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 EVK's 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.

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

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

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

- Set DTB (Device Tree Blob) file correctly ("***fdt_file***" boot variable) when bootloader first comes up. If not set correctly, the kernel may not boot, or the Wi-Fi/BT may not function correctly. As already described, unless the platform is module down or the interconnect provides WLAN_HOST_WAKE connection, the default DTB selected uses in-band SDIO interrupts.
- Invoke "***switch_module.sh cyw***" after Linux kernel boots and then ***reboot the platform***. This is necessary to configure "cfg80211" library and WPA supplicant. The Murata-customized image supports the embedded NXP wireless solution which uses a different "cfg80211" library and WPA supplicant to what Cypress-based "fmac" driver requires.

4 Preparing NXP i.MX Platforms to Boot Linux Image

In this section we document how to flash the (micro) SD card with the Linux image using both Linux and Windows PC. Most NXP i.MX EVK's use the (micro) SD card to boot the platforms. However, there are ***two special cases*** as documented in **Table 10** below. These two configurations include the i.MX 8M Mini EVK (8MMINILPD4-EVK variant) and i.MX 8M Nano EVK with the "uSD-M.2$^+$" configuration. In both cases the NXP i.MX EVK's 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 10: Select Hardware Configurations which use eMMC Boot Configuration

Target (MACHINE)	Hardware Config	i.MX DTB File	Boot Config	Interrupt Config	3.3V VIO SDIO
imx8mmevk	M.2^w	imx8mm-evk.dtb	uSD	N/A	N/A
imx8mmevk	uSD-M.2$^+$	imx8mm-evk-usd-m2-oob.dtb	**eMMC**	OOB	N
imx8mmevk	uSD-M.2$^+$	imx8mm-evk-usd-m2.dtb	**eMMC**	SDIO	N
imx8mnddr4evk	MD (1MW)	imx8mn-ddr4-evk.dtb	uSD	OOB	N
imx8mnddr4evk	uSD-M.2$^+$	imx8mn-evk-usd-m2-oob.dtb	**eMMC**	OOB	N
imx8mnddr4evk	uSD-M.2$^+$	imx8mn-evk-usd-m2.dtb	**eMMC**	SDIO	N
uSD-M.2$^+$ = works with uSD-M.2 Adapter (Rev B1) with additional cabling M.2^w = Wi-Fi/BT M.2 EVB plugs directly into M.2 connector; but only WLAN is functional MD = Murata Module is soldered down					

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

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

4.1.1 Linux PC Steps to Flash SD Card

Now that the SD card image is built, we can now flash the (micro) SD card used for booting the i.MX platform. Insert the (micro) SD card into a host machine (PC). It is imperative that the (micro) SD card comes up as "/dev/sdx" device. If it does not, then you may require a USB to (micro) SD card adapter as shown in **Figure 6**. 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 6: USB to SD Card Reader/Writer Adapter

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

```
dmesg
```

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

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

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

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

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

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

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

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

4.1.2 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" 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[6], click on the folder icon/button and navigate to the location of the desired "*.sdcard" file (or "*.wic" file, for Yocto release Zeus and later). You need to change "*.img" file type to "*.*" to select/open the "*.sdcard" file.
- Select the "Device" button and select the drive letter corresponding to the (micro) SD card: formatting SD card may be necessary beforehand with Windows low-level utilities[7].
- 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.

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

4.2 Flashing Murata-customized Linux Image to NXP i.MX 8M Mini/Nano EVK's

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

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

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

4.2.1 Software File Preparation

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

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

Table 11 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.

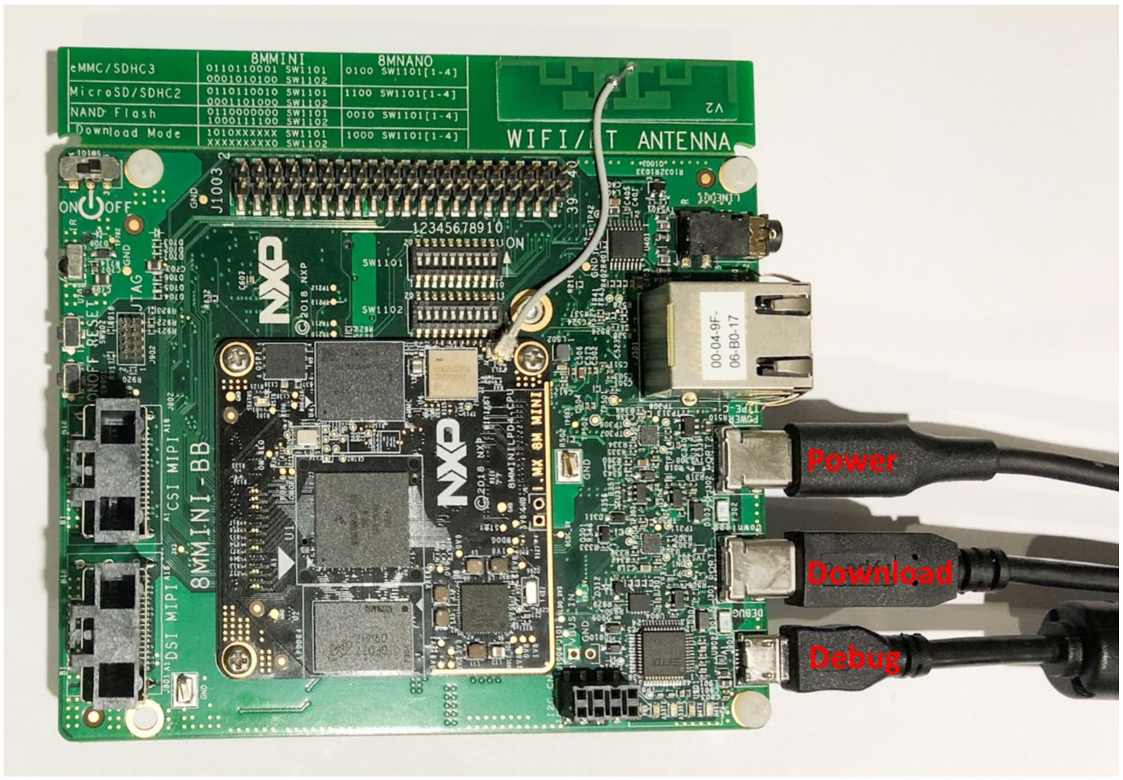
Table 11: Files to flash i.MX 8M Mini & 8M Nano EVK's

i.MX Host	Filename	Location
i.MX 8M Mini	uuu.exe	https://github.com/NXPmicro/mfgtools/releases/tag/uuu_1.3.191
	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	https://github.com/NXPmicro/mfgtools/releases/tag/uuu_1.3.191
	imx-boot-imx8mnddr4evk-sd.bin-flash_ddr4_evk	$BUILD_DIR/tmp/deploy/images/imx8mnddr4evk/
	fsl-image-validation-imx-imx8mnddr4evk.wic.bz2	$BUILD_DIR/tmp/deploy/images/imx8mnddr4evk/

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

This section describes steps to correctly configure i.MX hardware platform before using “uuu” executable to flash Linux image. **Figure 7** shows the necessary connections for power, download, and debug (serial console).

Figure 7: 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. 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 8**.

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.

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 9** .

SW1101	1	0	0	0

Where: 1 – ON, 0 – OFF

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

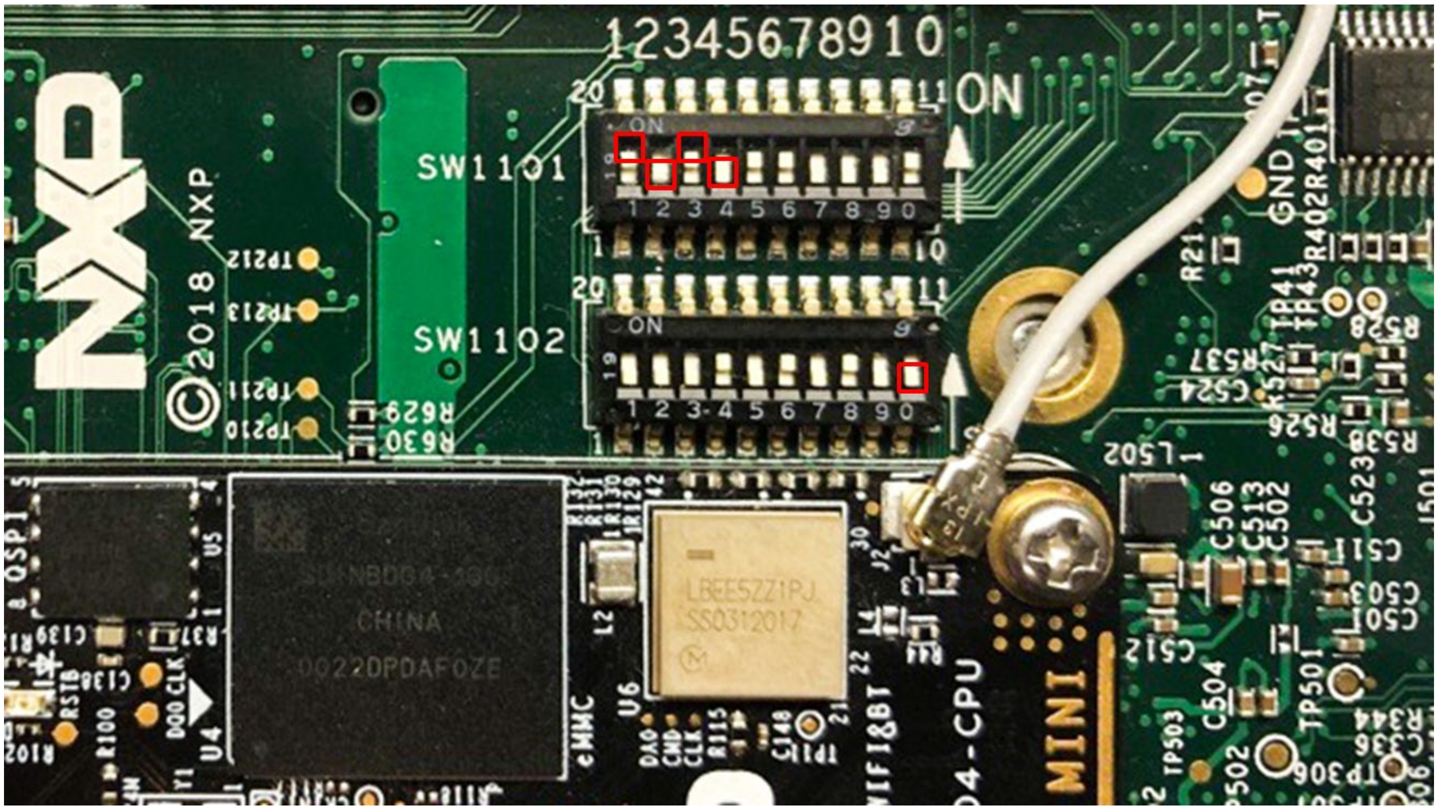
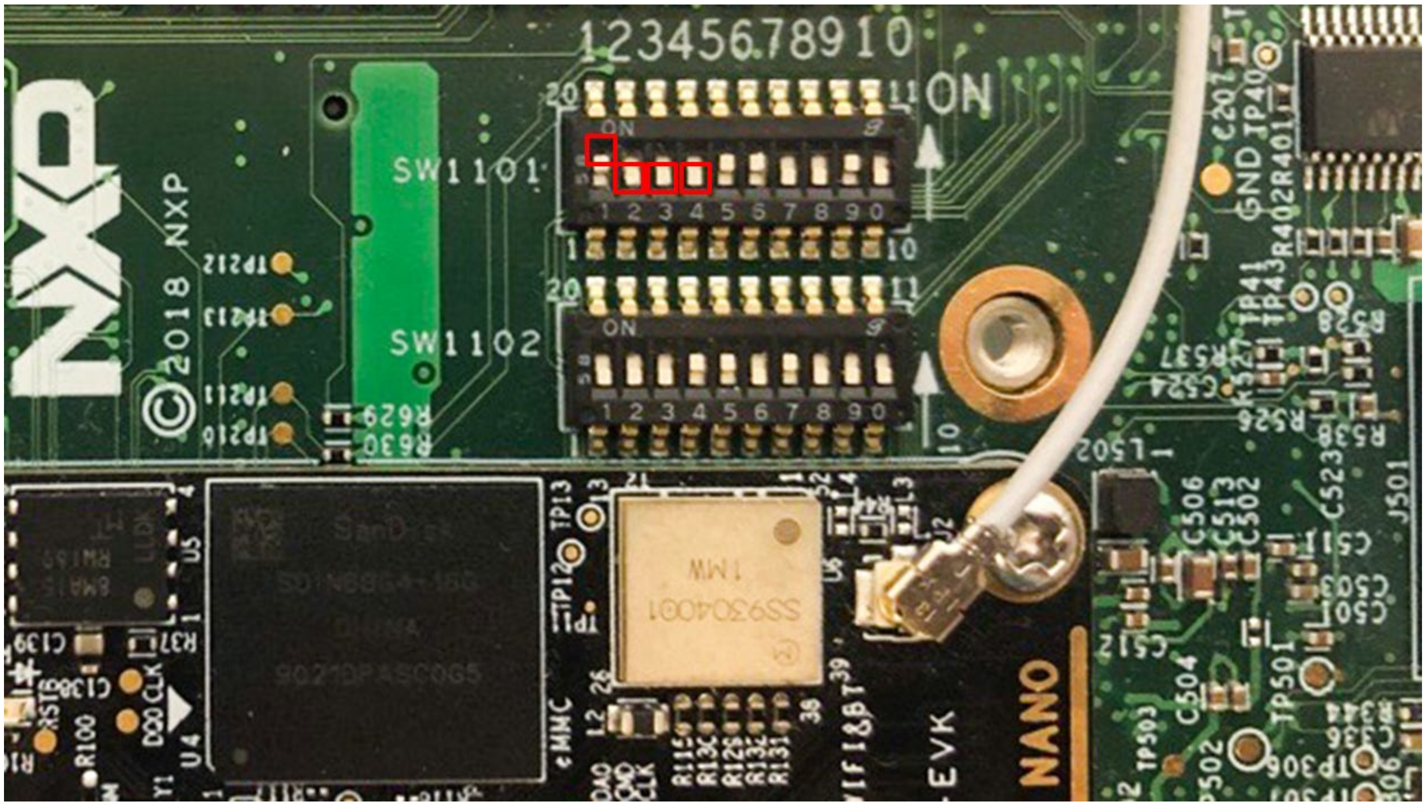


Figure 9: i.MX 8M Nano EVK DIP Switches configured for Download



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

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

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

Per **Table 11**, the filenames are specific to either i.MX 8M Mini or Nano EVK platform.

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

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

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

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

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

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

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

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

For **i.MX 8M Mini EVK** (8MMINILPD4-EVK), this is accomplished by setting the DIP switches (SW1101 and SW1102) to the positions below. The DIP switch settings are also shown in **Figure 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 and X – Do not care.

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

SW1101	0	1	0	0

Where: 1 – ON, 0 – OFF

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

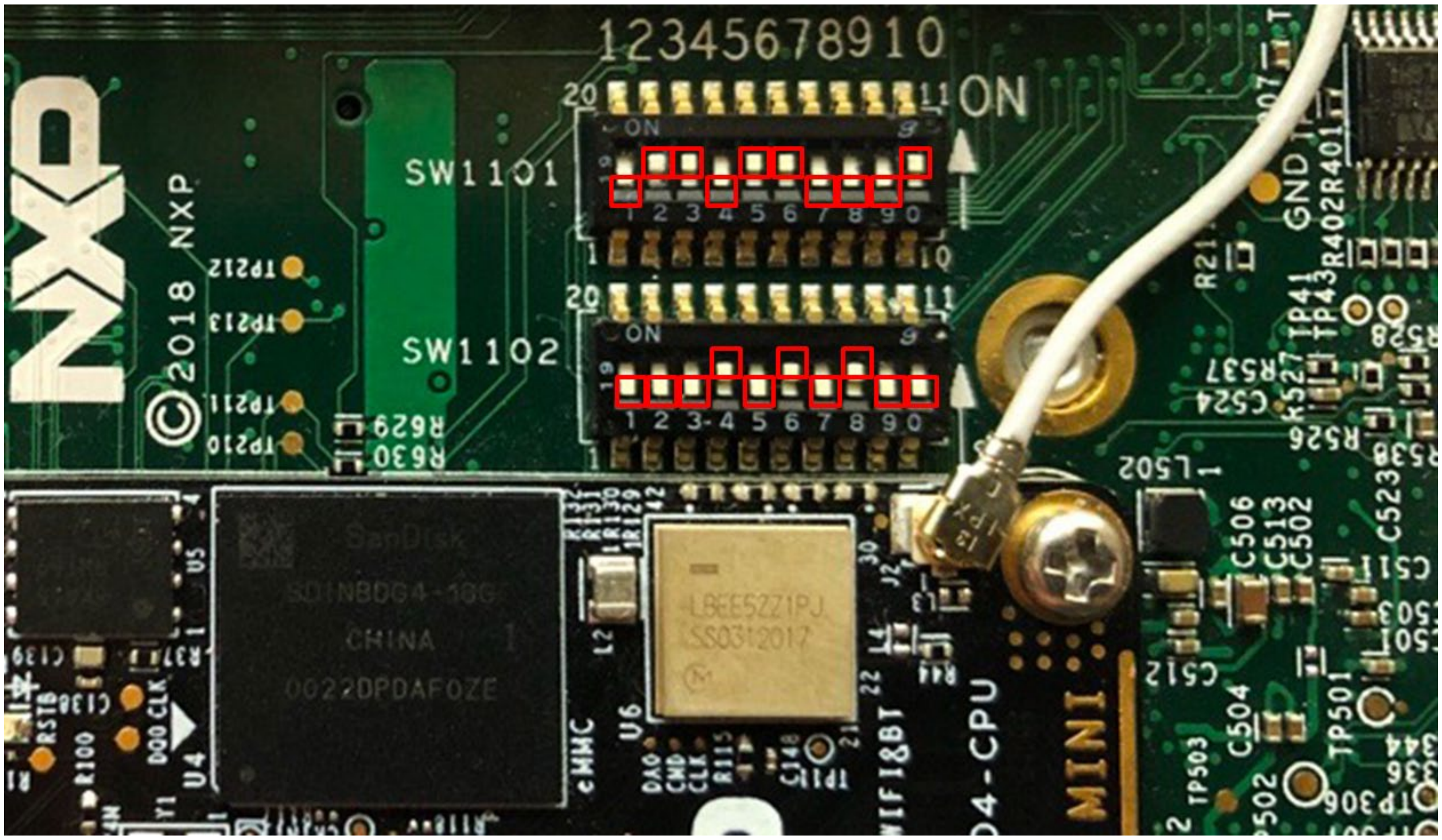
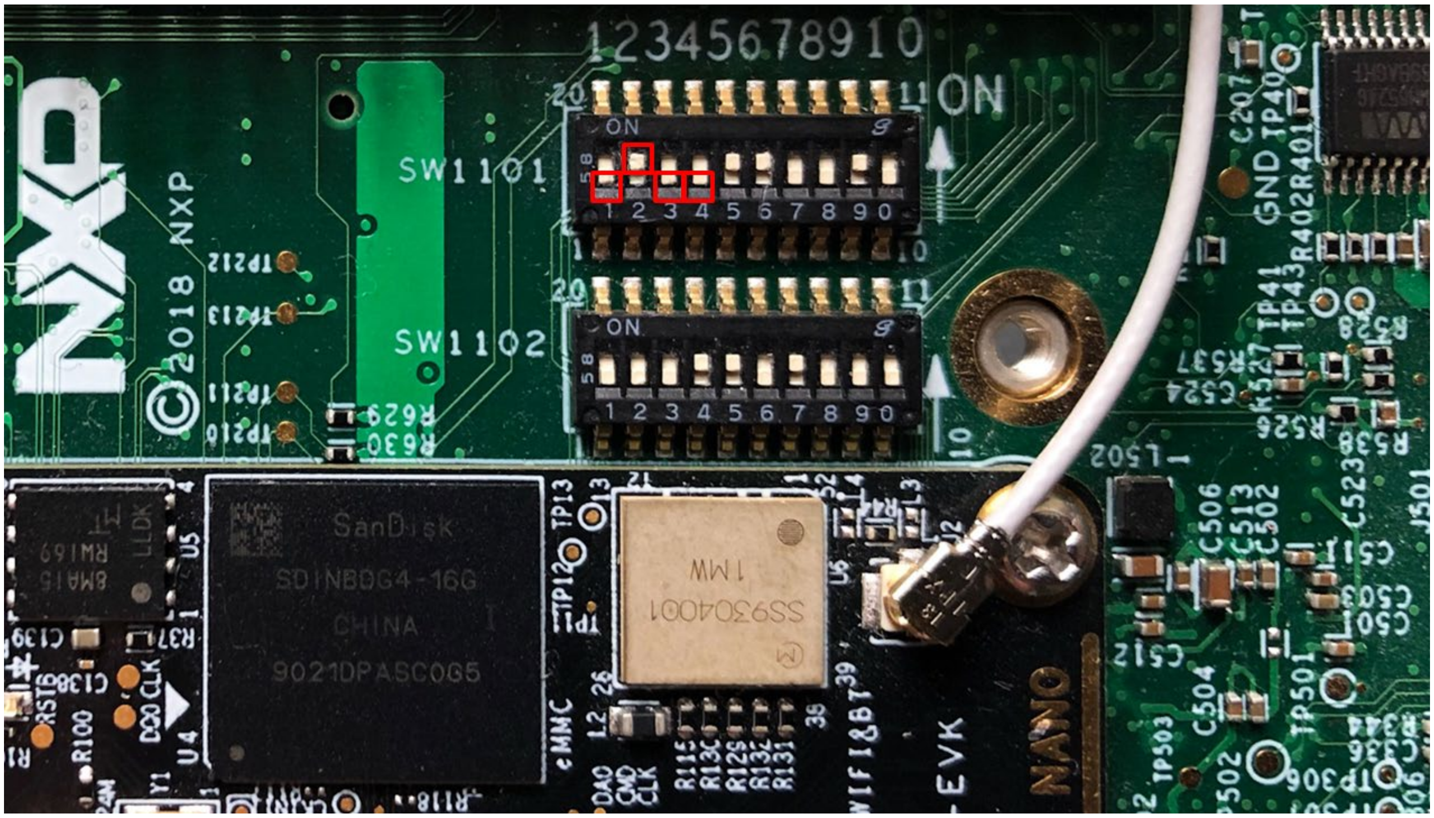


Figure 11: i.MX 8M Nano EVK DIP Switches configured for eMMC Boot



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

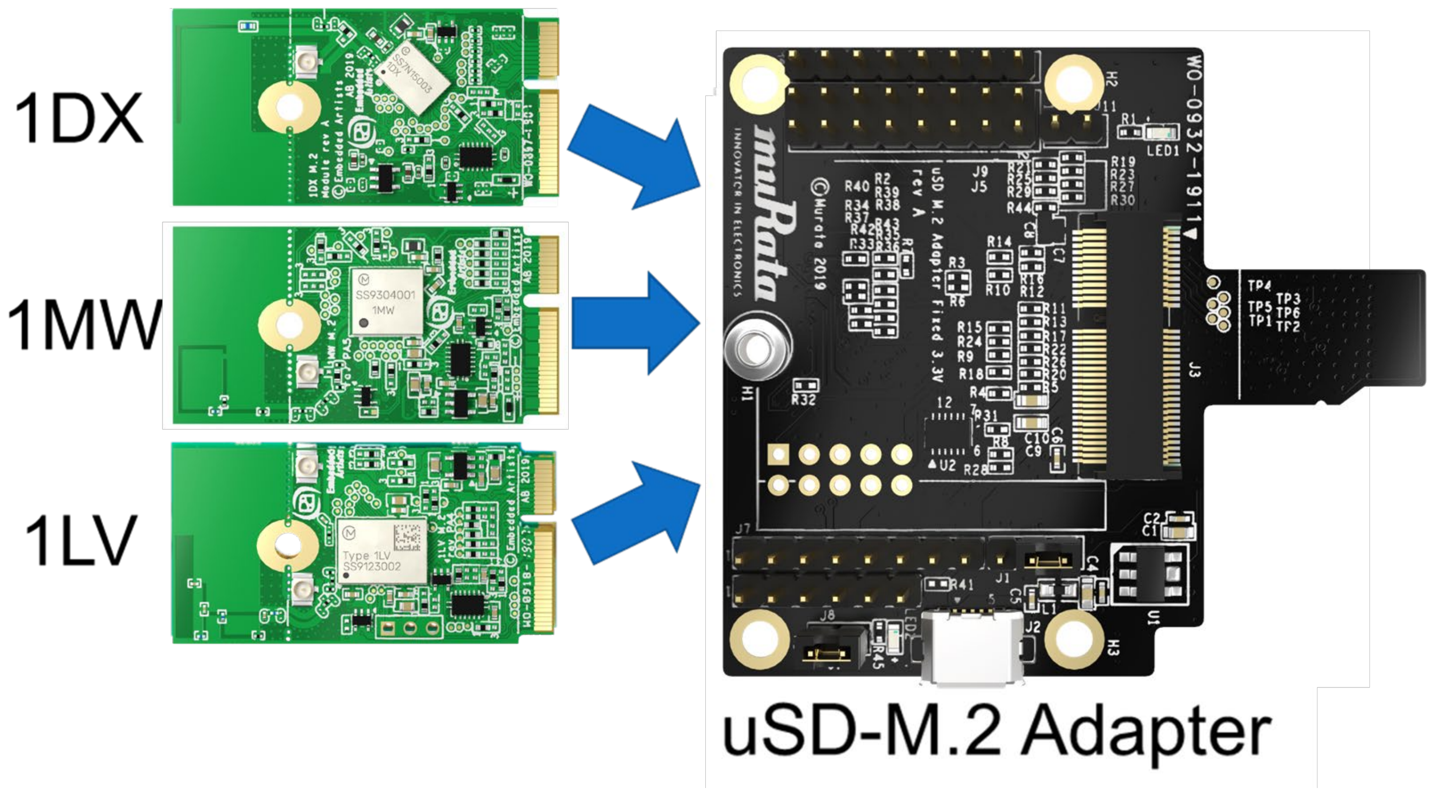
Embedded Artists' Wi-Fi/BT M.2 EVB's based on SDIO, listed in **Table 5** (currently 1DX/1MW/1LV) can be connected to the i.MX 6 Platforms through Murata's uSD-M.2 Adapter as shown in **Figure 12**. The following sub-sections details steps for bringing up Embedded Artists' Wi-Fi/BT EVB's on the three major variants of the i.MX 6 Platform supported: i.MX 6Q(P)/DL, i.MX 6SX, and i.MX 6UL(L).

As previously described, all legacy NXP i.MX 6 Platforms with exception of i.MX 6UL(L) EVK's require a 3.3V WLAN-SDIO VIO override mode configuration due to a host limitation. The 3.3V WLAN-SDIO VIO will only work with the Wi-Fi/BT M.2 EVB's that support it – ***currently Type 1DX and 1MW.*** Note that Type ***1LV only supports 1.8V VIO*** – which is the default WLAN-SDIO VIO signaling for the M.2 specification. Note that the 3.3V VIO override mode also configures BT-UART VIO at 3.3V – which is fine given that both host and target are signaling at that same voltage.

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

For more information on hardware configuration refer to the Hardware User Manual.

Figure 12: uSD-M.2 Adapter with type 1DX/1MW/1LV M.2 EVB options



5.1 Connecting to i.MX 6SX SDB (3.3V WLAN-SDIO VIO Override)

The Murata Wi-Fi/BT EVB is connected via the SD2 slot: see **Figure 13** below.

[1] Ensure no power is applied to i.MX 6SoloX SDB. Connect J16 micro-USB port to PC and start terminal emulator: “Minicom” on Linux or “Tera Term” on Windows. Set port to 115200-N-8-1.

[2] On uSD-M.2 Adapter, set Jumper J1 to position 2-3 (VBAT from microSD connector). Make sure Jumper J11 is removed (BT core enabled). Note Blue LED2 ***illuminates*** in 3.3V override mode.
  a. For Rev B1 adapter, install J13/J12 in 1-2/2-3 positions, respectively for 3.3V override mode.
  b. For legacy Rev A adapter, short J12 for 3.3V VIO override mode.

[3] Secure the Wi-Fi/BT M.2 EVB to the M.2 connector on the uSD-M2 Adapter per **Section 8.1**. Connect the uSD-SD Card adapter and tape the uSD Adapter-SD Card connection per **Section 8.3**.

[4] Connect the ribbon cable at both ends before inserting Wi-Fi/BT EVK (Wi-Fi/BT M.2 EVB/uSD-M.2 Adapter/uSD-SD Card) into SD2 slot. Note the orientation as shown in **Figure 13**.

[5] Prepare SD card to boot platform per **Section 4.1.** Insert SD card, power on the platform and interrupt at u-boot. Set DTB configuration with “***fdt_file***” parameter (“***oob***” string configures OOB IRQ configuration – if i.MX Platform has been configured correctly; see **Section 3.4.1** for more details). You can check the available dtb files using the command “***fatls mmc 1***”. After setting DTB, save the u-boot configuration and boot the platform:

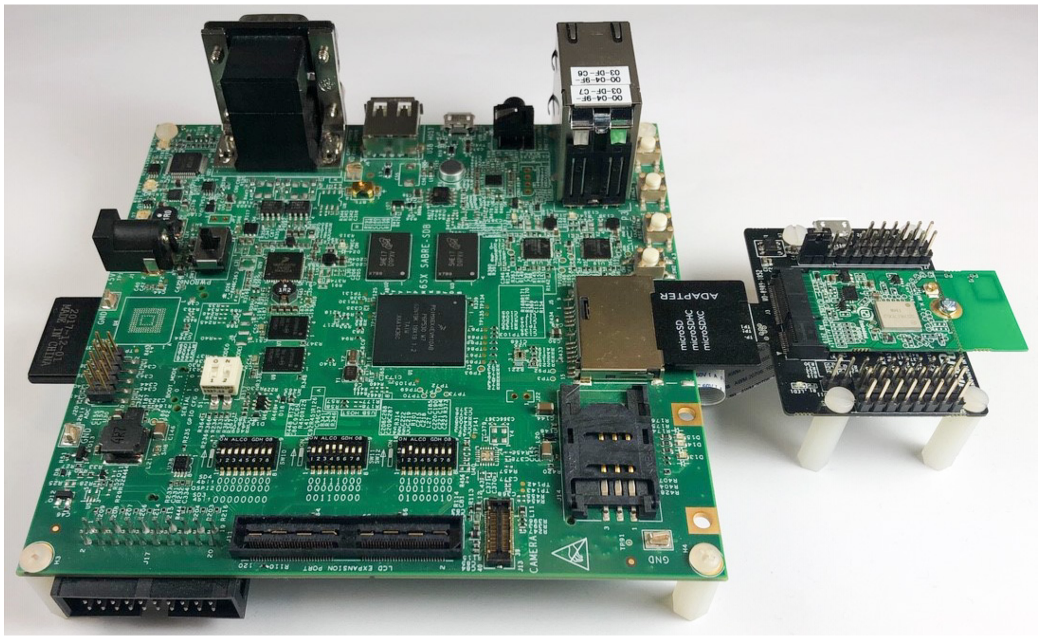
```
setenv fdt_file imx6sx-sdb-btwifi-m2<-oob>.dtb
saveenv
boot    ← causes platform to boot kernel
```

[6] After the kernel boots, invoke “***switch_module.sh cyw***” for Cypress-based solution and reboot:

```
switch_module.sh cyw
reboot
```

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

Figure 13: i.MX 6SoloX SDB with uSD-M.2 Adapter and Type 1MW M.2 EVB



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

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

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

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

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

5.2.2 Specific Hardware Considerations for i.MX 6QuadPlus SDB

Depending on the revision, rework **may be required** for the i.MX 6QuadPlus SDB. This rework connects the Bluetooth UART and Wi-Fi/BT control signals (WL_REG_ON, BT_REG_ON, and WL_HOST_WAKE). Revision B of i.MX 6QuadPlus SDB populates the necessary resistors for connecting BT UART and Wi-Fi/BT control signals. If your board has revision earlier than B (i.e. A2) then you will have to populate the necessary resistors. Refer to the Hardware User Manual for necessary rework. For the Rev A2 board, page 15 of the NXP schematic (SPF-28857_A2.pdf) correctly captures the necessary rework to be done. Note the much simpler rework on the i.MX 6QuadPlus SDB given that the no special "crossing" of TX and RX resistor pads is necessary. Those schematic notes are repeated below. **NOTE:** To use J13, populate resistors R209 - R213 and depopulate the SPI NOR FLASH U14.

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

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

[1] Ensure no power is applied to i.MX 6Quad(Plus)/DualLite SDB. Connect J509 micro-USB port to PC and start emulator: "Minicom" on Linux or "Tera Term" on Windows. Set port to 115200-N-8-1.
[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 ***illuminates*** in 3.3V override mode.
   a. For Rev B1 adapter, install J13/J12 in 1-2/2-3 positions, respectively for 3.3V override mode.
   b. For legacy Rev A adapter, short J12 for 3.3V VIO override mode.

[3] Connect the Wi-Fi/BT M.2 EVB to the M.2 connector on the uSD-M2 Adapter per **Section 8.1**. Connect the uSD-SD Card adapter and tape the uSD Adapter-SD Card connection per **Section 8.3**.
[4] Connect the ribbon cable at both ends before inserting Wi-Fi/BT EVK (Wi-Fi/BT M.2 EVB/uSD-M.2 Adapter/uSD-SD Card) into SD2 slot. Note the orientation as shown in **Figure 14**.
[5] Prepare SD card to boot platform per **Section 4.1**. Insert SD card, power on the platform and interrupt at u-boot. Set DTB configuration with "***fdt_file***" parameter ("***oob***" string configures OOB IRQ configuration; see **Section 3.4.1** for more details). You can check the available dtb files using the command "***fatls mmc 1***". After setting DTB, save the u-boot configuration and boot the platform:

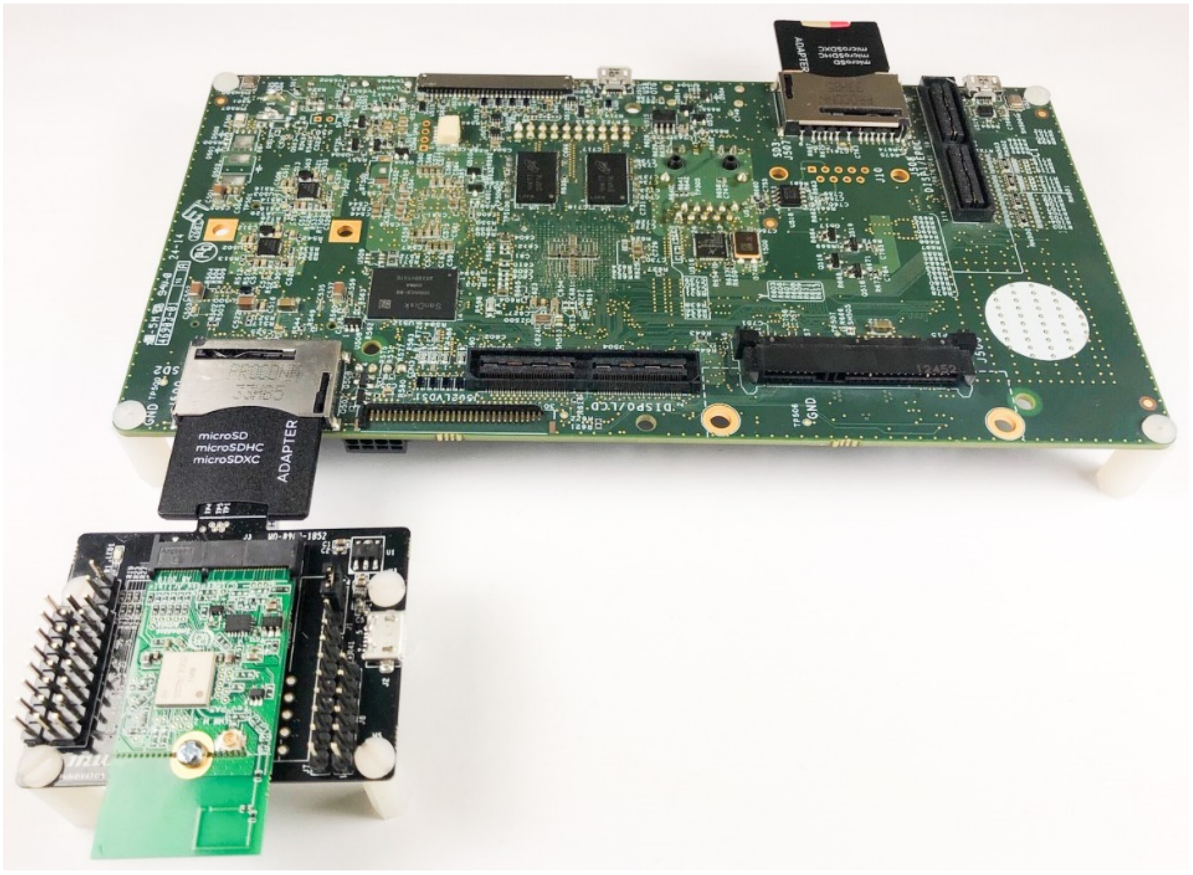
```
setenv fdt_file imx6q-sabresd-btwifi-m2<-oob>.dtb
                  (OR imx6dl-sabresd-btwifi-m2<-oob>.dtb,  imx6qp-sabresd-btwifi-m2<-oob>.dtb)
saveenv
boot  🡐 causes platform to boot kernel
```

[6] After the kernel boots, invoke "***switch_module.sh cyw***" for Cypress-based solution and reboot:

```
switch_module.sh cyw
reboot
```

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

Figure 14: i.MX 6Quad/DualLite SDB (Inverted) with uSD-M.2 Adapter and Type 1MW M.2 EVB



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

[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.
  a. For Rev B1 adapter, install J13/J12 in 1-2/1-2 positions, respectively for 1.8V VIO.
  b. For legacy Rev A adapter, make sure J12 is not installed for 1.8V VIO.

[3] Connect the Wi-Fi/BT M.2 EVB to the M.2 connector on the uSD-M2 Adapter per **Section 8.1**. Connect the uSD-SD Card adapter and tape the uSD Adapter-SD Card connection per **Section 8.3**.

[4] Connect ribbon cable at both ends before inserting Murata EVK into SD1 slot. Note the orientation as shown in **Figure 15**. ***Make sure that the adapter clicks in correctly*** – the i.MX 6UL(L) EVK's have a Push-Push SD card connector. Tape the SD Card-EVK connection per **Section 8.3**.

[5] Prepare microSD card to boot platform per **Section 4.1**. Insert microSD card, power on the platform and interrupt at u-boot. Set DTB configuration with "***fdt_file***" parameter ("***oob***" string configures OOB IRQ configuration; see **Section 3.4.1** for more details). You can check the available dtb files using the command "***fatls mmc 1***". After setting DTB, save the u-boot configuration and boot the platform:

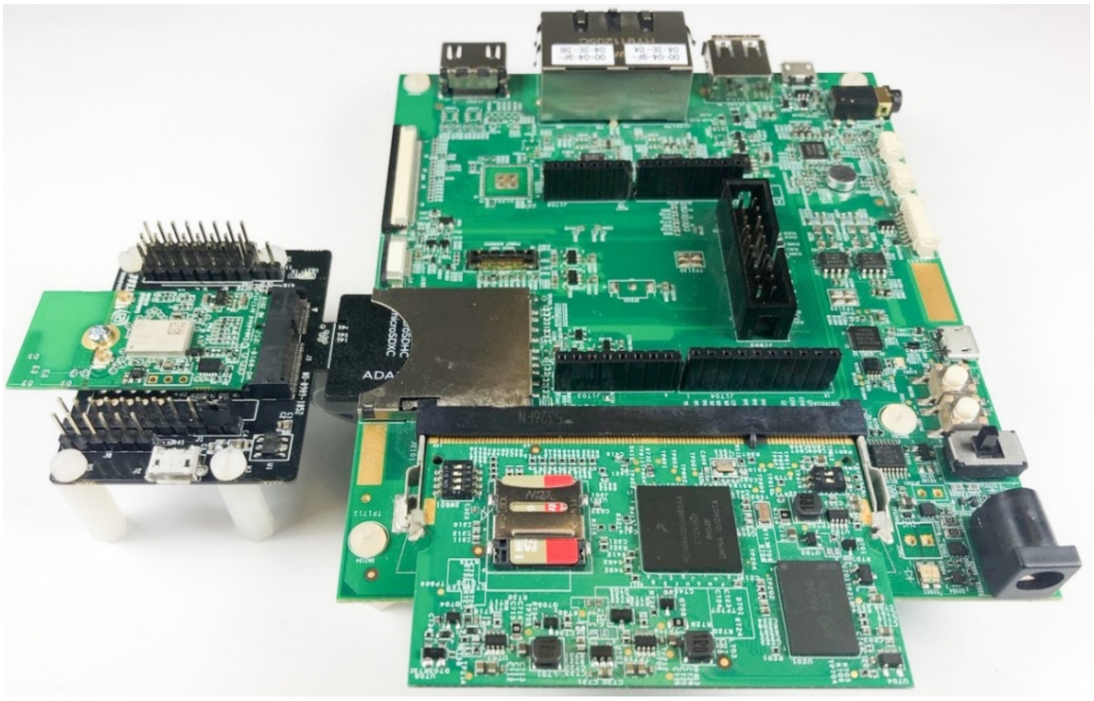
```
setenv fdt_file imx6ul-14x14-evk-btwifi-m2<-oob>.dtb  (OR imx6ull-14x14-evk-btwifi-m2<-oob>.dtb)
saveenv
boot  ← causes platform to boot kernel
```

[6] After the kernel boots, invoke "***switch_module.sh cyw***" for Cypress-based solution and reboot:

```
switch_module.sh cyw
reboot
```

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

Figure 15: i.MX 6UL EVK with uSD-M.2 Adapter and Type 1LV M.2 EVB



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

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

By selecting one of two DTB files, the user can configure for either the onboard Type 1CX module or use the M.2 connector. As shown in **Figure 16**, the i.MX 8MQuad EVK provides a secondary Wi-Fi/BT solution on the underside via a M.2 connector. The uSD-M.2 adapter provides WLAN-PCIe, BT-UART, control signals, and optionally BT-PCM. Currently the only supported M.2 EVB's are Type 1CX and Type 1XA (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] If not using onboard Type 1CX module, connect Embedded Artists Type 1XA or 1CX M.2 EVB to M.2 connector as shown in **Figure 16**. Attach two dual-band (2.4/5GHz) antennas with U.FL. connector as shown – otherwise there is severe signal attenuation. Now you can flip platform right-side-up. Try and keep antennas at right angles for optimum WLAN throughput performance.
[3] Prepare microSD card to boot platform per **Section 4.1**. Insert microSD card, power on platform and interrupt at u-boot. Set DTB configuration with "***fdt_file***" parameter to either configure for Wi-Fi/BT M.2 EVB or onboard Type 1CX module. You can check the available dtb files using the command "***fatls mmc 1***". Now save the u-boot configuration and boot the platform:

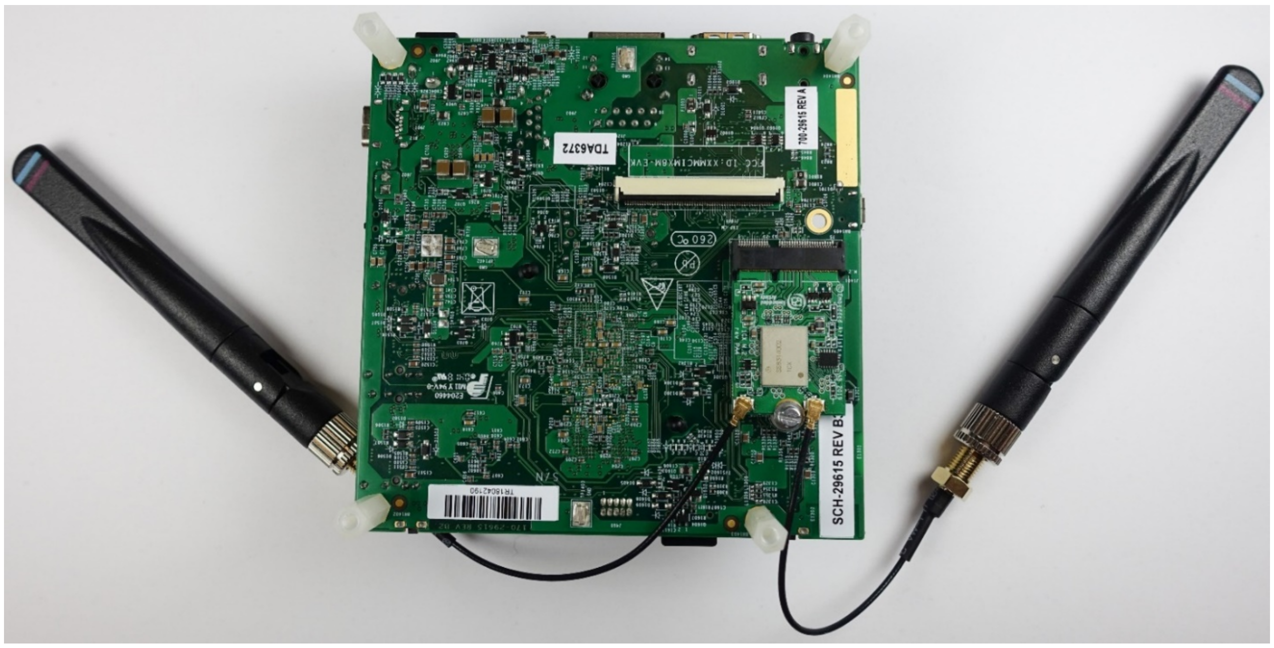
```
setenv fdt_file fsl-imx8mq-evk-pcie1-m2.dtb    (OR fsl-imx8mq-evk.dtb)
saveenv
boot  ← causes platform to boot kernel
```

[4] After the kernel boots, invoke "***switch_module.sh cyw***" for Cypress-based solution and reboot:

```
switch_module.sh cyw
reboot
```

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

Figure 16: i.MX 8MQuad with Type 1CX (bottom view)



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

The 8M Mini EVK (8MMINID4-EVK) and 8M Nano EVK (8MNANOD4-EVK) both have Type 1MW module down. Both 8M Mini EVK's (8MMINILPD4-EVK and 8MMINID4-EVK) have a WLAN-PCIe M.2 connector on the baseboard – with no connection for Bluetooth-UART. See **Figure 17** for upside-down EVK view showing M.2 connector (current supported M.2 EVB's are Type 1CX and 1XA). The steps below detail how to bring up the onboard Type 1MW module or 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] For Wi-Fi M.2 EVB option, connect Embedded Artists Type 1CX or 1XA M.2 EVB to M.2 connector as shown in **Figure 17**. Attach two dual-band (2.4/5GHz) antennas with U.FL. connector as shown – otherwise there is severe signal attenuation. Now you can flip platform right-side-up. Try and keep antennas at right angles for optimum WLAN throughput performance.
[3] Prepare 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 9**. Note that the 8MMINID4-EVK uses same DTB file for both onboard 1MW and M.2 EVB. Check the available dtb files by invoking "***fatls mmc 1***". Now save u-boot configuration and boot the platform:

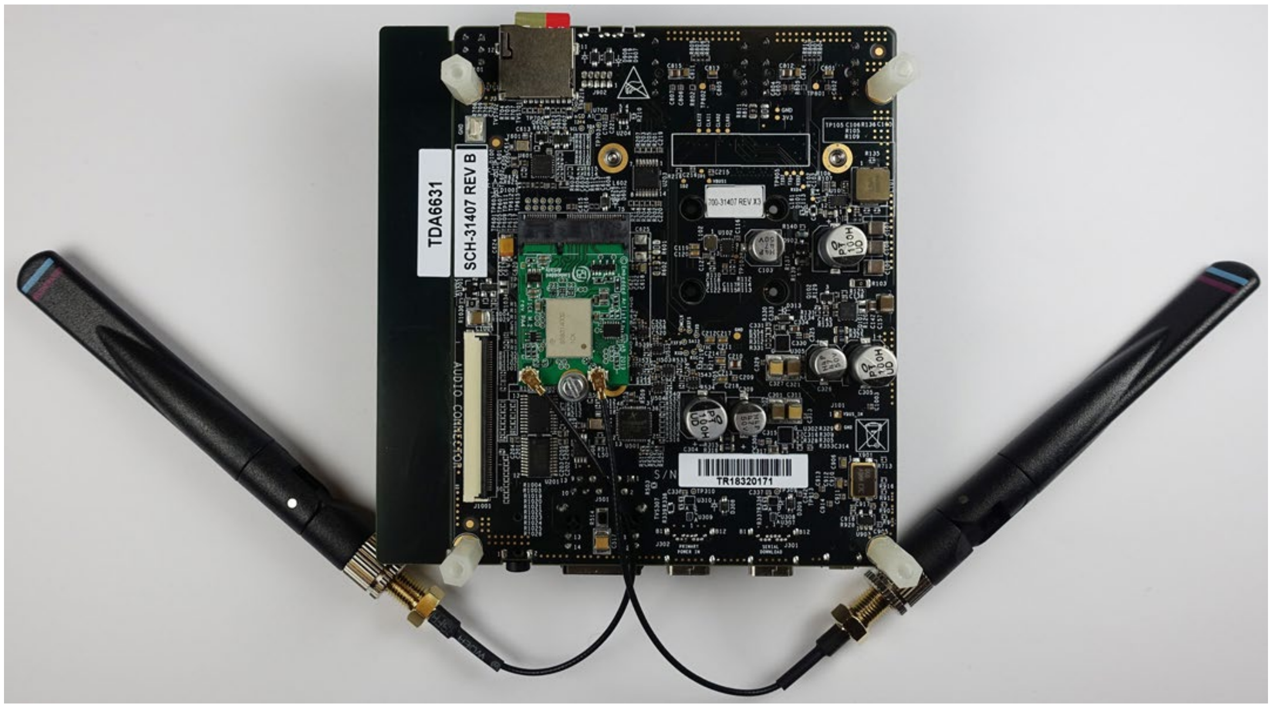
```
setenv fdt_file imx8mm-evk.dtb   (OR  imx8mm-ddr4-evk.dtb OR imx8mn-ddr4-evk.dtb)
saveenv
boot  ← causes platform to boot kernel
```

[4] After the kernel boots, invoke "***switch_module.sh cyw***" for Cypress-based solution and reboot:

```
switch_module.sh cyw
reboot
```

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

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



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

There are two configurations for adding uSD-M.2 Adapter interconnect to i.MX 8M Mini/Nano EVK: WLAN/Bluetooth (**Figure 19**) and WLAN (**Figure 18**). ***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 18: i.MX 8M Mini/Nano EVK with uSD-M.2 Adapter (WLAN Only)

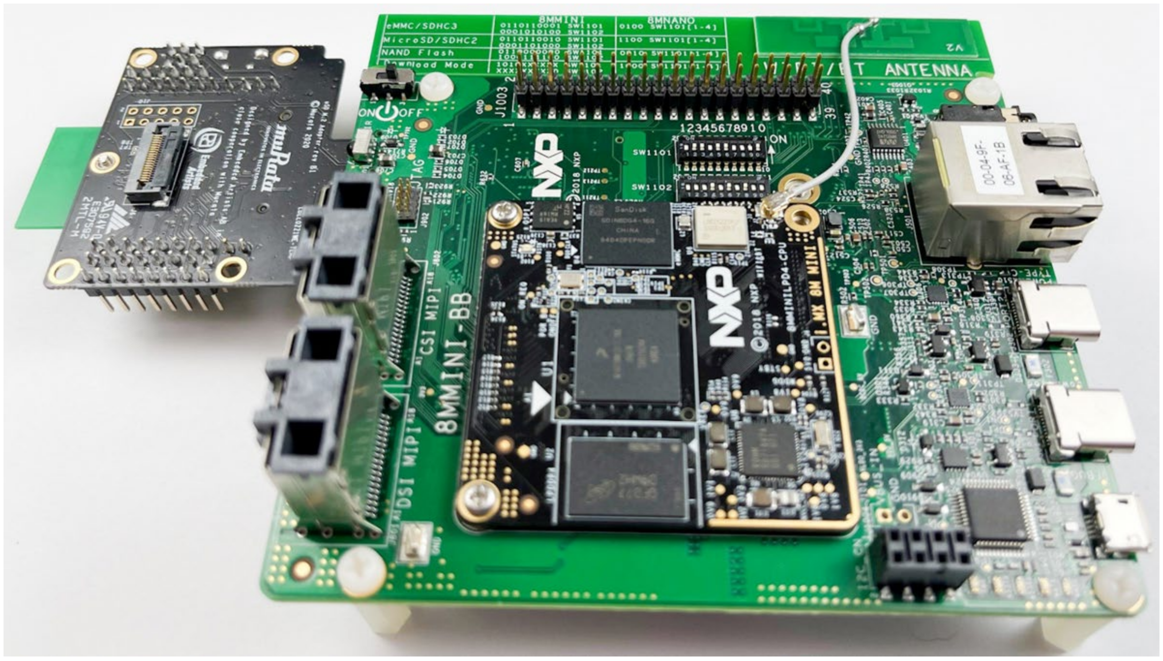
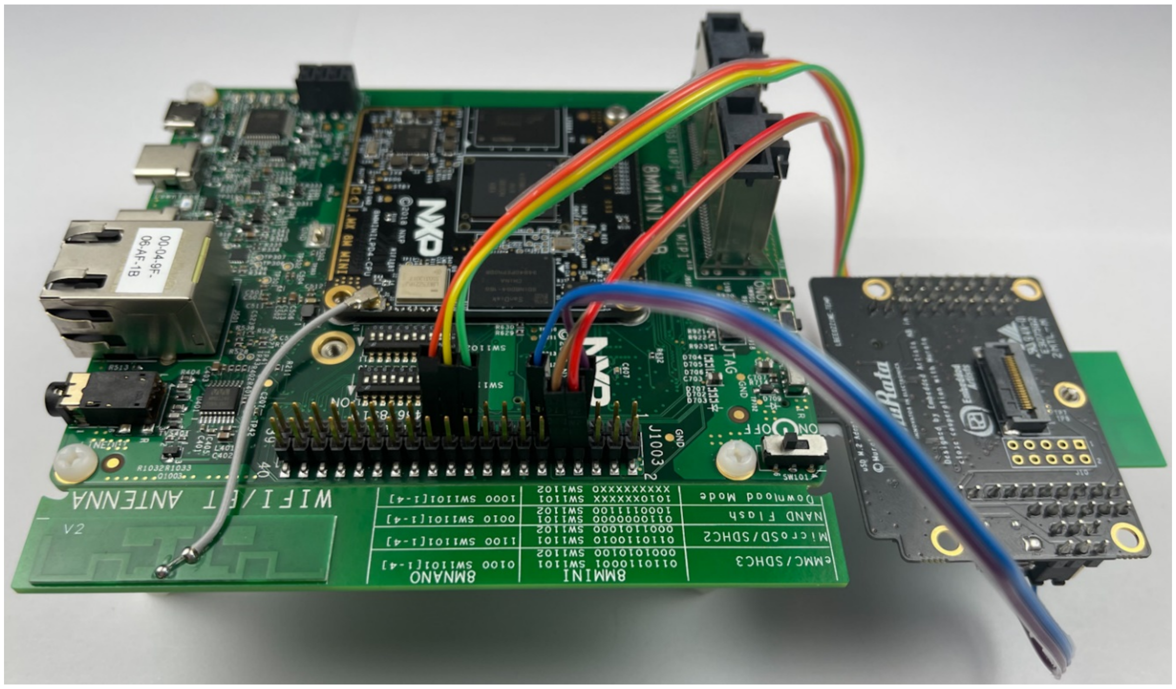


Figure 19: i.MX 8M Mini/Nano EVK with uSD-M.2 Adapter (WLAN/Bluetooth)



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

[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.
  a. For Rev B1 adapter, install J13/J12 in 1-2/1-2 positions, respectively for 1.8V VIO.
  b. For legacy Rev A adapter, make sure J12 is not installed for 1.8V VIO.

[3] Connect the Wi-Fi/BT M.2 EVB to the M.2 connector on the uSD-M2 Adapter per **Section 8.1**. Attach the uSD-M.2 Adapter/M.2 to EVK and tape the uSD Adapter-EVK connection per **Section 8.3**.

[4] For full Wi-Fi/BT configuration (Bluetooth-UART and WLAN/BT control lines), connect seven (7) jumper wires from the uSD-M.2 adapter to the i.MX 8M Mini/Nano EVK per **Table 12**. Refer to **Figure 20**, **Figure 21, Figure 22,** and **Figure 23** 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:

  a. Use low-profile jumper wires (like Digi-Key part number 1988-1178-ND) which can be bent at right-angles – see **Figure 24**, and **Figure 25**. 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:

  b. 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 26**, 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.

NOTE: for ***WLAN-only, no jumper cables need to be connected.*** For customers only needing to evaluate Wi-Fi (**Figure 18**), 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 "***fdt_file***" parameter ("***oob***" string configures OOB IRQ configuration; see **Section 3.4.1** for more details). OOB interrupt configuration will only work if additional jumper wires are attached (WL_HOST_WAKE). You can check the available dtb files using the command "***fatls mmc 2***". After setting DTB, save the u-boot configuration and boot the platform:

```
setenv fdt_file imx8mm-evk-usd-m2<-oob>.dtb  (OR imx8mn-evk-usd-m2<-oob>.dtb)
saveenv
boot  ← causes platform to boot kernel
```

[7] After the kernel boots, invoke "***switch_module.sh cyw***" for Cypress-based solution and reboot:

```
switch_module.sh cyw
reboot
```

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

Table 12: i.MX 8M Mini/Nano EVK Jumper connections to uSD-M.2 Adapter

Signal Name	uSD-M.2 Adapter Header/Pin	i.MX 8M Mini/Nano EVK J1003 Pin	i.MX 8M Mini/Nano EVK Signal
BT_UART_TX	J9 / Pin 1	10	UART3_RXD
BT_UART_RX	J9 / Pin 2	8	UART3_TXD
WL_REG_ON	J9 / Pin 3	19	ECSPI2_MOSI
BT_REG_ON	J9 / Pin 4	21	ECSPI2_MISO
WL_HOST_WAKE	J9 / Pin 5	23	ECSPI2_SCLK
BT_UART_RTS	J8 / Pin 3	11	UART3_RTS
BT_UART_CTS	J8 / Pin 4	7	UART3_CTS

Figure 20: Cable connections on i.MX 8M Mini EVK (Even Number Connector View)

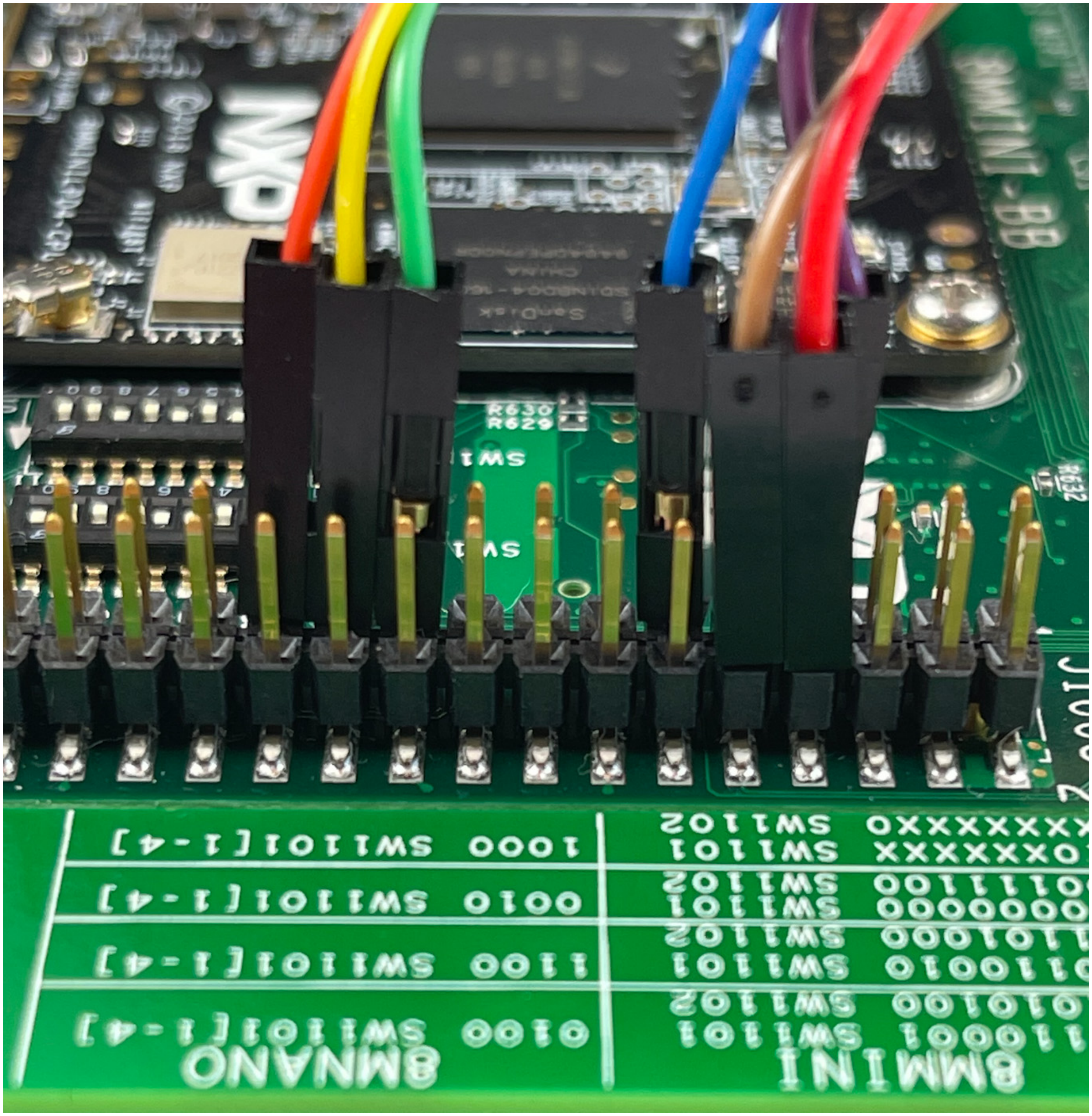


Figure 21: Cable connections on i.MX 8M Mini EVK (Odd Number Connector View)

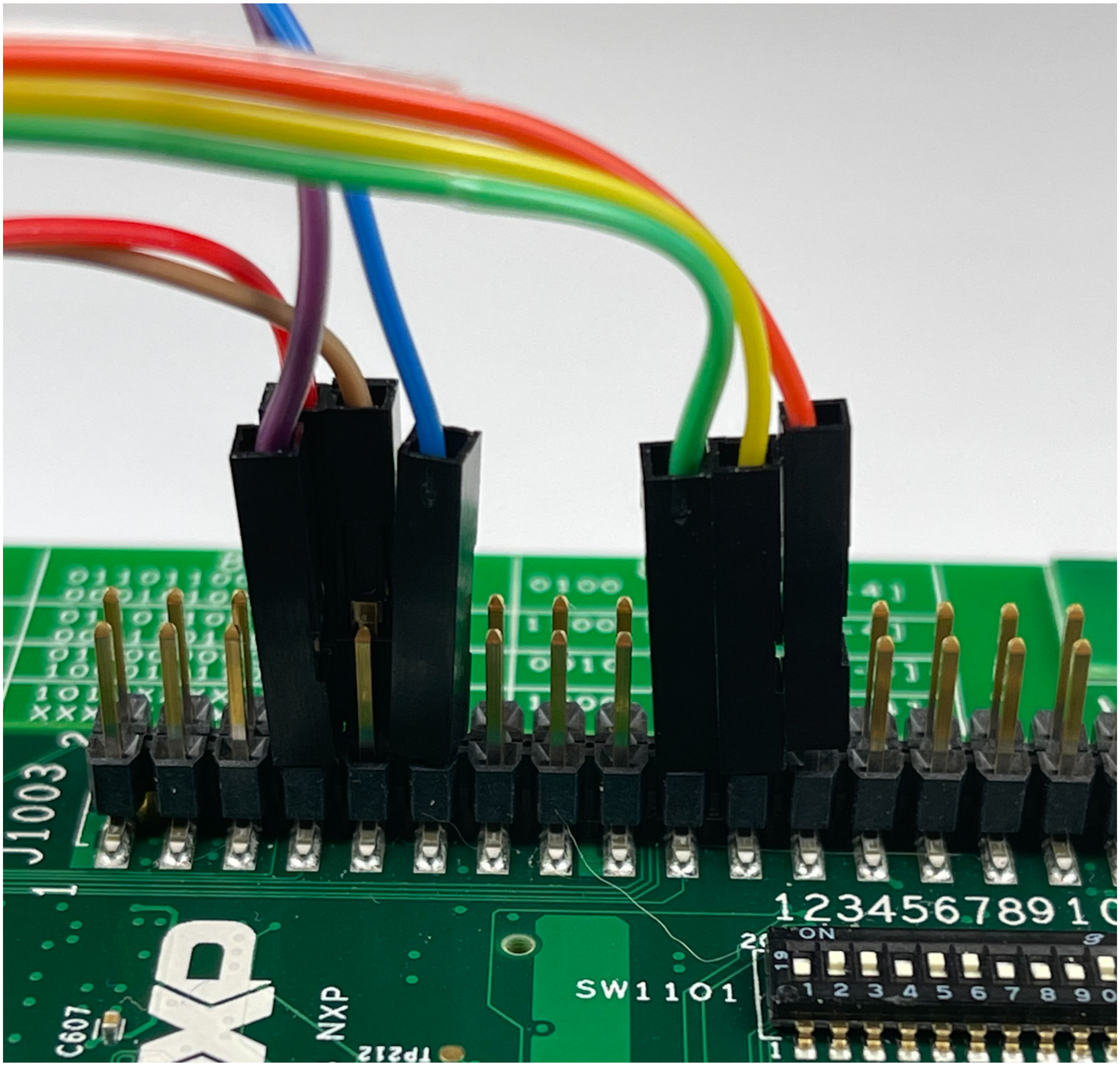


Figure 22: Cable connections on uSD-M.2 Adapter (J9 Header)

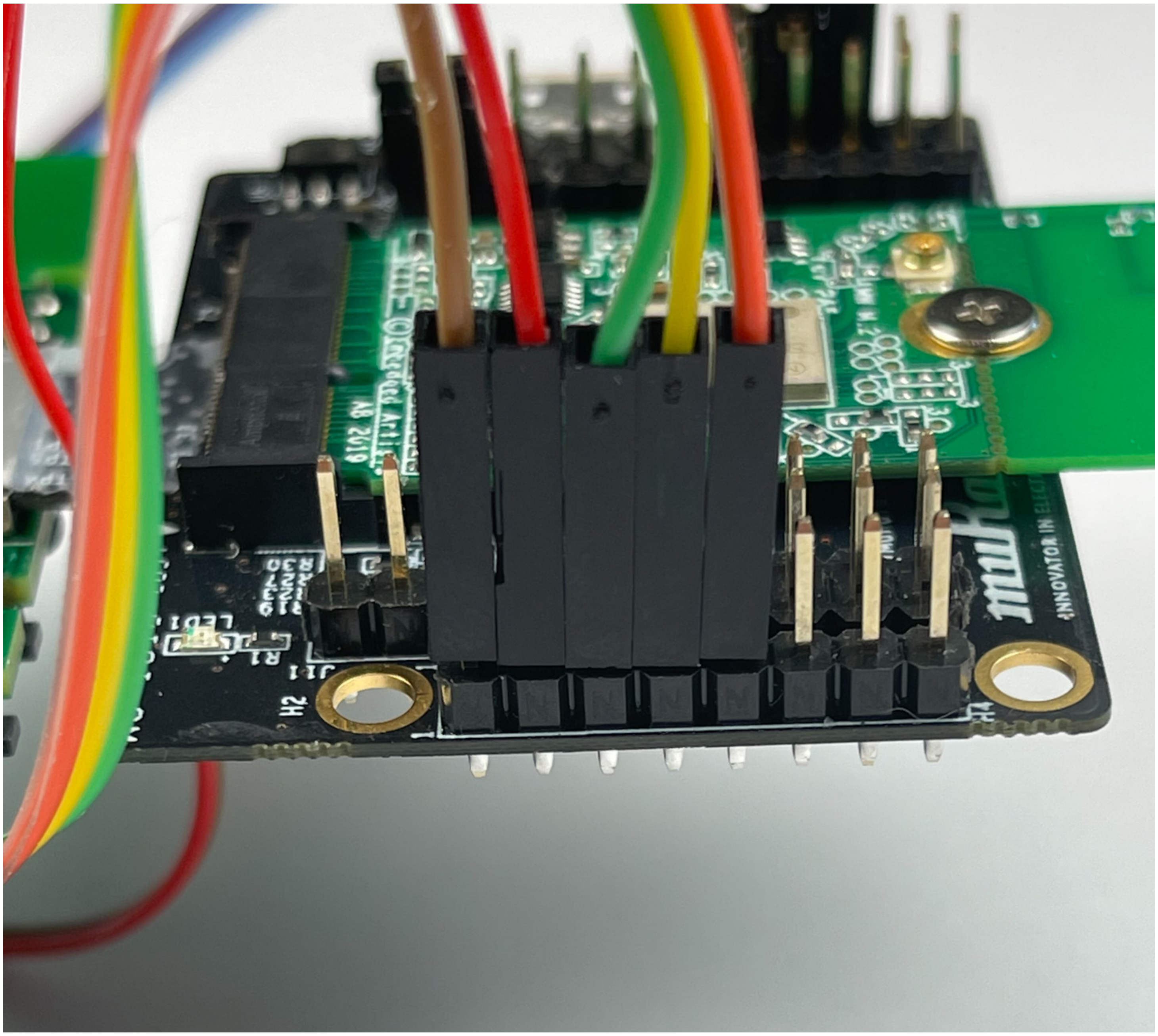


Figure 23: Cable connections on uSD-M.2 Adapter (J8 Header)

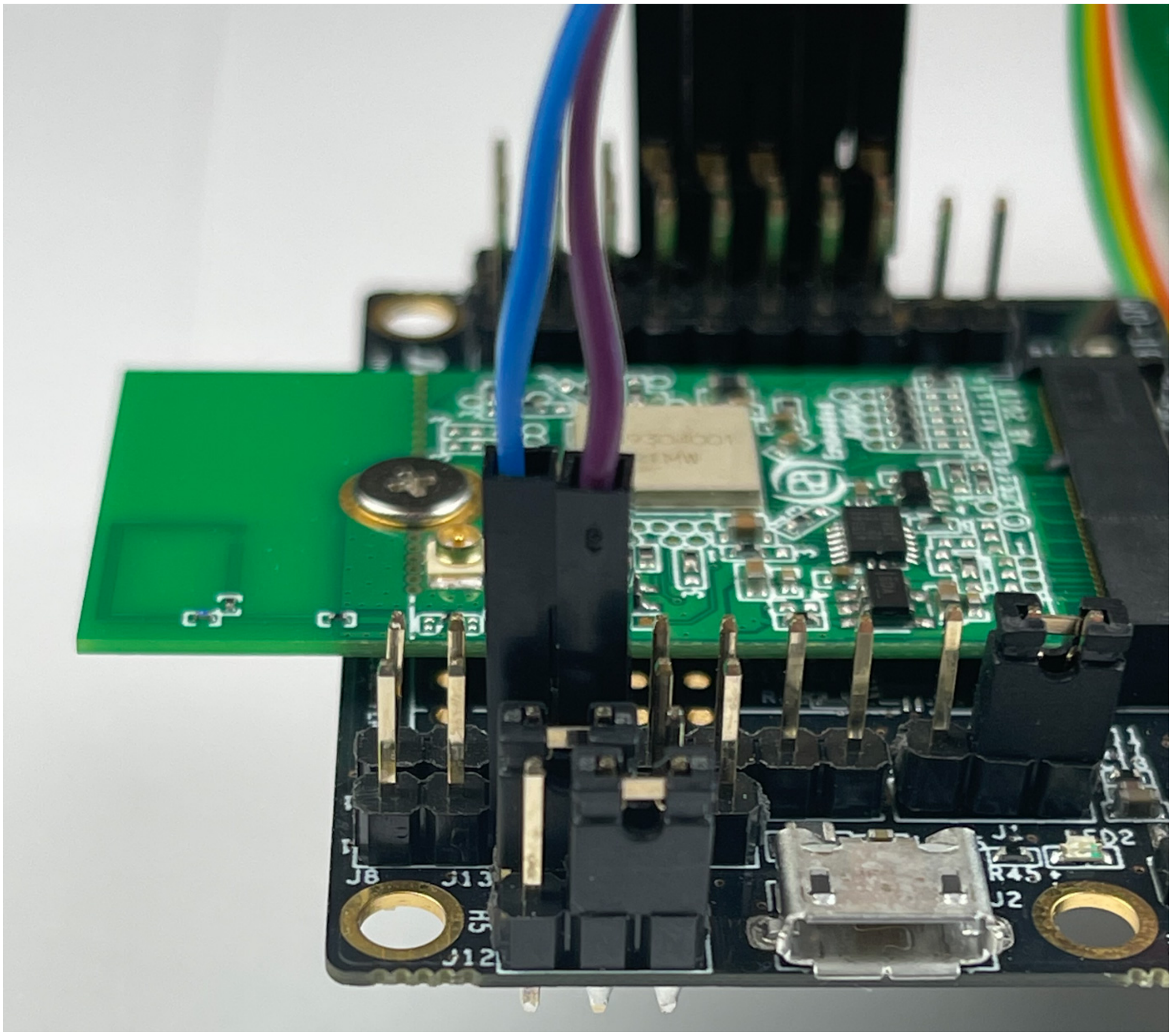


Figure 24: Low-Profile Jumper Wires (Digi-Key part number 1988-1178-ND)

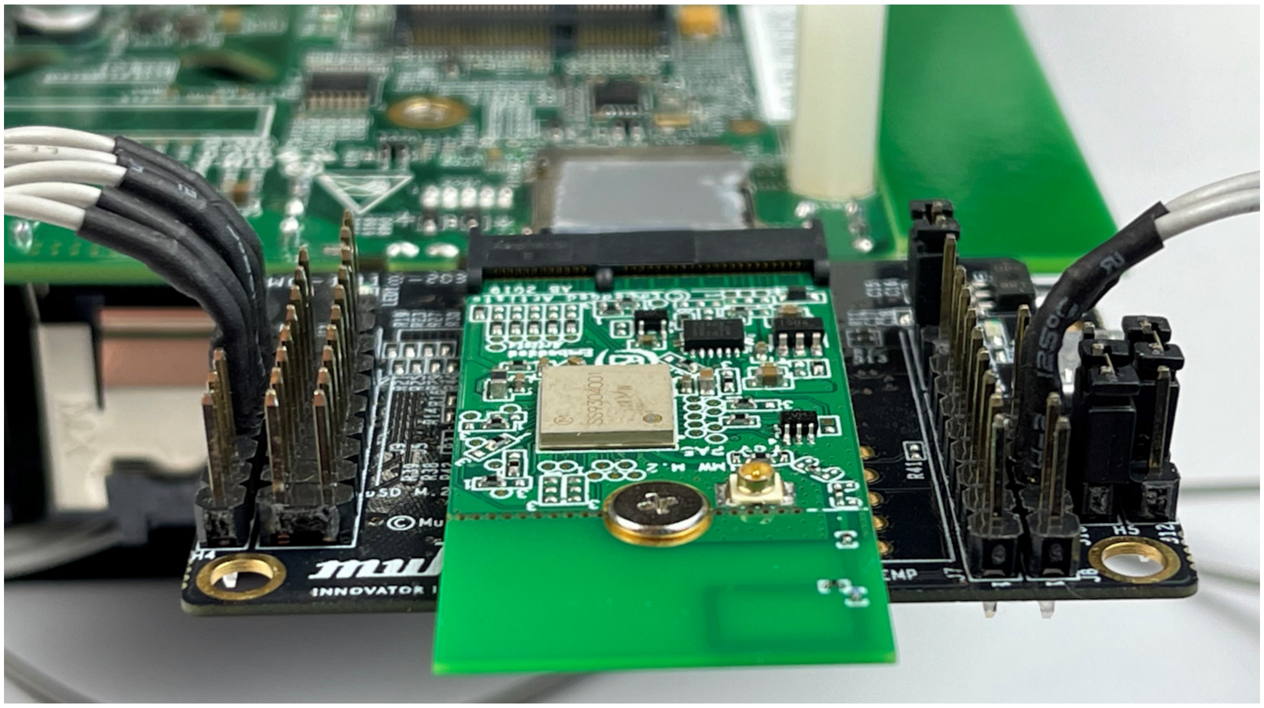


Figure 25: NXP i.MX EVK with Low-Profile Jumper Wires

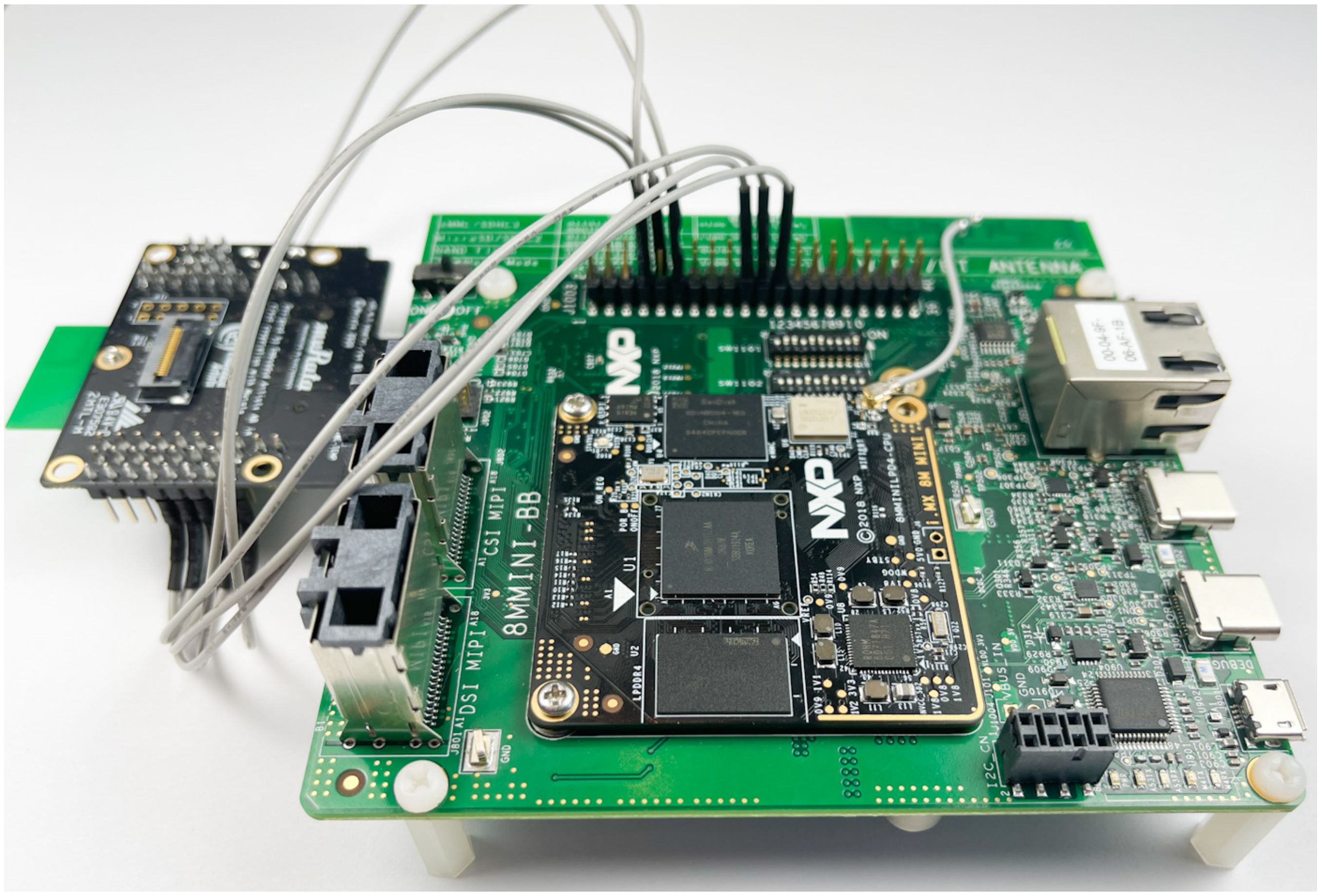
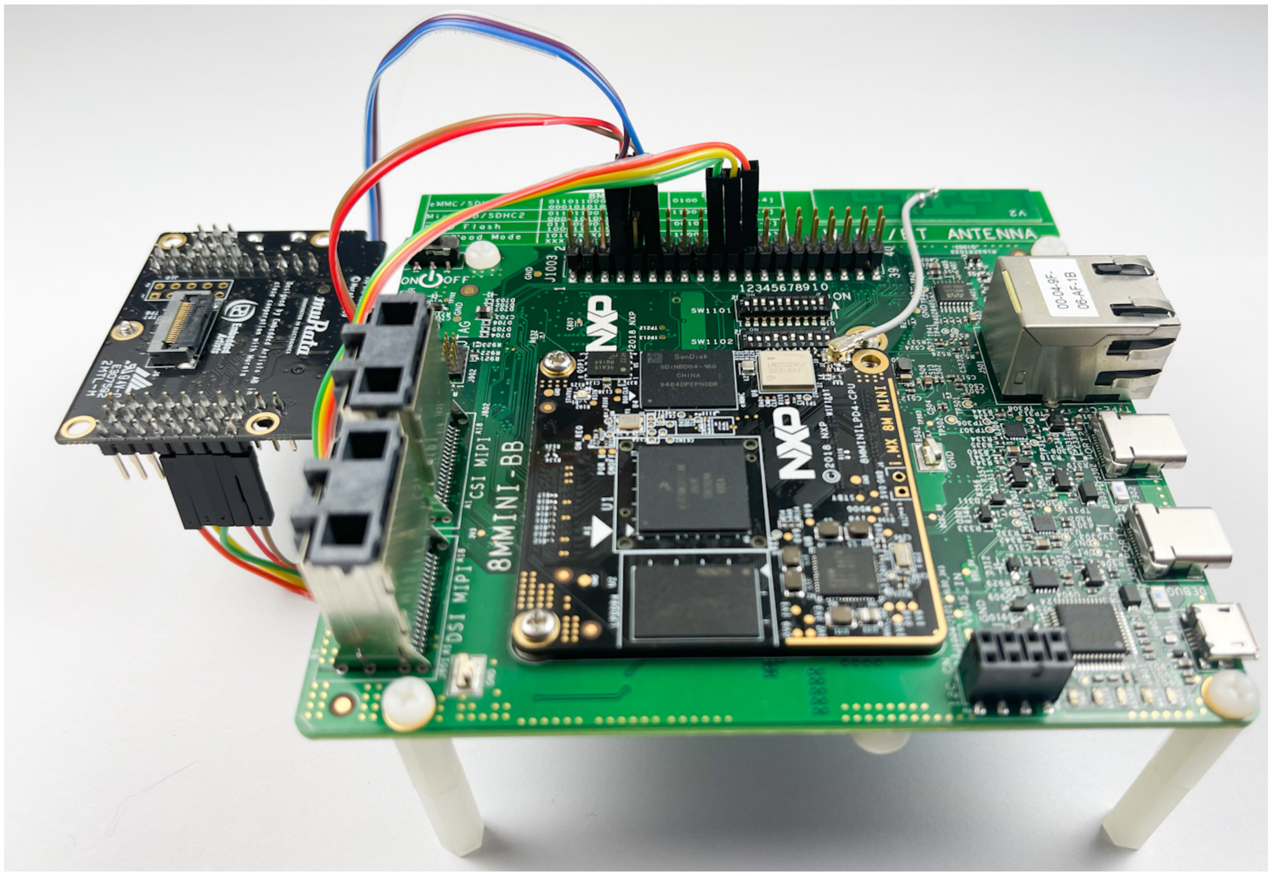


Figure 26: Additional Hex Standoff (Digi-Key part number RPC3570-ND)



7 Test/Verification of Wi-Fi and Bluetooth

Now the kernel should be booting correctly on the NXP i.MX platform with Murata module being correctly initialized (correct DTB configured and Cypress software configuration selected with "switch_module.sh"). Next steps are to verify Wi-Fi and Bluetooth functionality. The Murata-customized i.MX images include all the necessary files to support Wi-Fi and Bluetooth bring-up/testing/verification (with exception of WLAN regulatory testing – see **Section 7.1.10**). The relevant folders and files are summarized in **Table 13**.

There is an "fmac version" column in **Table 13:** this is used to differentiate any files that are specific to a Cypress "fmac" driver release. In this table we can see that the file naming convention for WLAN firmware/nvram/clm_blob depends on version of "fmac". For "fmac" driver versions "kong" and earlier, the folder is "/lib/firmware/brcm" with an initial filename string of "brcmfmac". For "fmac" versions "zigra" and later, the folder is "/lib/firmware/cypress" with an initial filename string of "cyfmac".

Table 13: Embedded Wi-Fi/Bluetooth Files

Folder / Filename	fmac ver	Details
/lib/firmware/cypress/cyfmac<chipset>-sdio.bin	zigra+	"fmac" firmware file for WLAN SDIO chipset/module.
/lib/firmware/cypress/cyfmac<chipset>-sdio.txt	zigra+	"fmac" NVRAM file for WLAN SDIO chipset/module.
/lib/firmware/cypress/cyfmac<chipset>-sdio.clm_blob	zigra+	"fmac" regulatory binary for WLAN SDIO chipset/module.
/lib/firmware/cypress/cyfmac<chipset>-pcie.bin	zigra+	"fmac" firmware file for WLAN PCIe chipset/module.
/lib/firmware/cypress/cyfmac<chipset>-pcie.clm_blob	zigra+	"fmac" regulatory binary for WLAN PCIe chipset/module.
/lib/firmware/cypress/cyfmac<chipset>-pcie.txt	zigra+	"fmac" NVRAM file for WLAN PCIe chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-sdio.bin	kong-	"fmac" firmware file for WLAN SDIO chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-sdio.txt	kong-	"fmac" NVRAM file for WLAN SDIO chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-sdio.clm_blob	kong-	"fmac" regulatory binary for WLAN SDIO chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-pcie.bin	kong-	"fmac" firmware file for WLAN PCIe chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-pcie.clm_blob	kong-	"fmac" regulatory binary for WLAN PCIe chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-pcie.txt	kong-	"fmac" NVRAM file for WLAN PCIe chipset/module.
/etc/firmware/BCM4343A1_001.002.009.0093.0395.1DX.hcd	any	Type 1DX (CYW4343W) Bluetooth patchfile.
/etc/firmware/BCM4345C0_003.001.025.0144.0266.1MW.hcd	any	Type 1MW (CYW43455) Bluetooth patchfile.
/etc/firmware/BCM43012C0_003.001.015.0102.0141.1LV.hcd	any	Type 1LV (CYW43012) Bluetooth patchfile.
/etc/firmware/BCM4356A2_001.003.015.0106.0403.1CX.hcd	any	Type 1CX (CYW4356) Bluetooth patchfile.
/etc/firmware/BCM4359D0.004.001.016.0200.1XA.hcd	any	Type 1XA (CYW54591) Bluetooth patchfile.
/usr/sbin/wl	any	"wl" tool used for RF testing, initial bring-up and debug.
/usr/sbin/iw	any	Linux "iw" executable.
/usr/sbin/wpa_supplicant	any	WPA supplicant executable.
/usr/sbin/wpa_cli	any	WPA CLI tool.
/usr/bin/wpa_passphrase	any	WPA Passphrase generator.
/etc/wpa_supplicant.conf	any	WPA supplicant configuration file.
/usr/sbin/hostapd	any	Hostapd – manages wireless link in Soft AP mode.
/usr/sbin/hostapd_cli	any	Hostapd CLI tool.
/etc/hostapd.conf	any	Hostapd configuration file.
/etc/udhcpd.conf	any	DHCP server configuration file.
/etc/network/interfaces	any	Modify this file to automatically bring up "wlan0" interface. Only applies to Kernel 4.1.15.
/usr/bin/hciattach	any	"hciattach" binary – used for initializing Bluetooth.
/usr/bin/hciconfig	any	"hciconfig" binary – used for configuring Bluetooth.
/usr/bin/hcitool	any	"hcitool" binary – used for controlling Bluetooth interface.
/usr/bin/iperf3	any	"iperf" throughput test tool (iperf3 is latest version).

7.1 Wi-Fi Interface Test/Verification

7.1.1 Useful Environment Setup on NXP Linux

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

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

7.1.2 Switch Module Script

Murata has included a script file in the customized Linux image which configures the wireless drivers for Cypress solution. The usage is as shown below:

```
switch_module.sh cyw
```

The "***switch_module.sh***" script file performs the following functions:

- Correctly loads the necessary CFG80211 module based on the module selected.
- Points to correct WPA supplicant for the selected module selected.

The "switch_module.sh cyw" step should have already been done in **Section 5** or **Section 6**. Note that the script file only needs to be run once – first time kernel is booted.

7.1.3 Bringing Up Wi-Fi Interface

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

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

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

```
sdhci: Secure Digital Host Controller Interface driver
sdhci: Copyright(c) Pierre Ossman
sdhci-pltfm: SDHCI platform and OF driver helper
sdhci-esdhc-imx 2190000.usdhc: allocated mmc-pwrseq
mmc0: SDHCI controller on 2190000.usdhc [2190000.usdhc] using ADMA
…
cfg80211: Loading compiled-in X.509 certificates for regulatory database
cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
mmc0: queuing unknown CIS tuple 0x80 (2 bytes)
mmc0: queuing unknown CIS tuple 0x80 (3 bytes)
```

```
mmc0: queuing unknown CIS tuple 0x80 (3 bytes)
mmc0: queuing unknown CIS tuple 0x80 (7 bytes)
mmc0: queuing unknown CIS tuple 0x80 (6 bytes)
brcmfmac: brcmf_fw_alloc_request: using cypress/cyfmac43455-sdio for chip BCM4345/6
usbcore: registered new interface driver brcmfmac
brcmfmac: brcmf_fw_alloc_request: using cypress/cyfmac43455-sdio for chip BCM4345/6
brcmfmac: brcmf_c_preinit_dcmds: Murata Customized Version: imx-zeus-zigra_r1.0;
brcmfmac: brcmf_c_preinit_dcmds: Firmware: BCM4345/6 wl0: May 22 2020 21:24:34 version 7.45.214 (9c83742 CY)
FWID 01-59feefd4
```

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

- "***brcmfmac***" string identifies "fmac" driver log messages
- "***cyfmac43455-sdio***" indicates the WLAN firmware file being loaded.
- "***Murata Customized Version***" indicates that "***meta-murata-wireless***" was used to generate this image. The branch or release/tag information is included.
- "***Firmware***" indicates specific version of firmware being loaded by "fmac". In this case the firmware version is 7.45.214.

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

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

$ ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 00:9D:6B:A6:EE:76        ← WLAN MAC Address
          UP BROADCAST MULTICAST  MTU:1500  Metric:1           ← "wlan0" interface is UP
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

NOTE: no IP address is assigned yet to the "wlan0" interface. That will be done later **Section 7.1.7**.

7.1.4 STA/Client Mode: Scan for Visible Access Points

In this section, two different/simple methods for scanning are presented: one uses the Cypress "***wl***" tool (Cypress-specific tool reserved mainly for RF testing); the other uses the preferred Linux "***iw***" command. If you do not see a list of SSID's and there are broadcasting Access Points (Wireless Routers) in range, then something is wrong. Please check antenna connection, setup, etc. Also note that the strength of received signals is important to do connectivity testing (i.e. "***ping***", "***iperf3***", etc.). Very attenuated signals will be in the high 80's or 90's (see "***RSSI***" value). If close to an Access Point, the returned "***RSSI***" value should be between -30 and -50 dBm for a properly configured setup.

7.1.4.1 Using "iw" Linux Command

"***iw***" is the default Linux command line tool for controlling/querying a WLAN interface. For more information on "***iw***" tool: https://wireless.wiki.kernel.org/en/users/documentation/iw. In the following example of listing WLAN devices and performing a scan, there is one active AP with a SSID of "***Murata_5G***". Here are the expected results:

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


$iw dev wlan0 scan   🡐  perform scan on "wlan0" interface

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

```
        HT TX MCS rate indexes are undefined
    HT operation:
         * primary channel: 36
         * secondary channel offset: above
         * STA channel width: any
         * RIFS: 1
         * HT protection: no
         * non-GF present: 0
         * OBSS non-GF present: 0
         * dual beacon: 0
         * dual CTS protection: 0
         * STBC beacon: 0
         * L-SIG TXOP Prot: 0
         * PCO active: 0
         * PCO phase: 0
    Extended capabilities: BSS Transition, 6
    VHT capabilities:
        VHT Capabilities (0x0f825932):
            Max MPDU length: 11454
            Supported Channel Width: neither 160 nor 80+80
            RX LDPC
            short GI (80 MHz)
            SU Beamformer
            SU Beamformee
        VHT RX MCS set:
            1 streams: MCS 0-9
            2 streams: MCS 0-9
            3 streams: MCS 0-9
            4 streams: not supported
            5 streams: not supported
            6 streams: not supported
            7 streams: not supported
            8 streams: not supported
        VHT RX highest supported: 0 Mbps
        VHT TX MCS set:
            1 streams: MCS 0-9
            2 streams: MCS 0-9
            3 streams: MCS 0-9
            4 streams: not supported
            5 streams: not supported
            6 streams: not supported
            7 streams: not supported
            8 streams: not supported
        VHT TX highest supported: 0 Mbps
    VHT operation:
         * channel width: 1 (80 MHz)
```

```
          * center freq segment 1: 42
          * center freq segment 2: 0
          * VHT basic MCS set: 0x0000
   WPS:     * Version: 1.0
          * Wi-Fi Protected Setup State: 2 (Configured)
          * Response Type: 3 (AP)
          * UUID: 1b52c4d5-ffb1-0ad1-63f3-9b91a979382c
          * Manufacturer: NETGEAR, Inc.
          * Model: R6300
          * Model Number: R6300
          * Serial Number: 4536
          * Primary Device Type: 6-0050f204-1
          * Device name: R6300
          * Config methods: Display
          * RF Bands: 0x3
          * Unknown TLV (0x1049, 6 bytes): 00 37 2a 00 01 20
   WMM:     * Parameter version 1
          * u-APSD
          * BE: CW 15-1023, AIFSN 3
          * BK: CW 15-1023, AIFSN 7
          * VI: CW 7-15, AIFSN 2, TXOP 6016 usec
          * VO: CW 3-7, AIFSN 2, TXOP 3264 usec…… etc.  🡐 More SSID listings follow here.
```

7.1.4.2 Using "wl" Tool

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

- "WL Tool for Embedded 802.11 Systems" on Cypress Linux Support Portal.

Now that Wi-Fi interface is up and running (having loaded the default "***cyfmac<chipset>-sdio.bin***" – WLAN firmware), let us do some basic testing to verify functionality. The "***wl scan***" command will initiate an active probe of all visible SSID's. The "***wl scanresults***" will return a list of visible SSID's. In this example, one active AP has SSID of "***Murata_5G***". Here are expected results:

```
$ wl scan
$ wl scanresults
brcmfmac: brcmf_cfg80211_vndr_cmds_dcmd_handler: oversize return buffer 130048
SSID: "Murata_5G"   🡐 Broadcast SSID, with RSSI (received signal strength) one line below
Mode: Managed   RSSI: -38 dBm   SNR: 0 dB      noise: 0 dBm    Flags: RSSI on-channel  Channel: 36/80
BSSID: 84:1B:5E:F6:A7:60       Capability: ESS
Supported Rates: [ 6(b) 9 12(b) 18 24(b) 36 48 54 ]
Extended Capabilities: BSS_Transition
```

```
VHT Capable:
    Chanspec: 5GHz channel 42 80MHz (0xe02a)
    Primary channel: 36    ← Channel Number
    HT Capabilities: 40Mhz SGI20 SGI40  ← 802.11ac bandwidth (40 MHz in this case)
    Supported HT MCS : 0-23
    Supported VHT MCS:
        NSS1 Tx: 0-9   Rx: 0-9
        NSS2 Tx: 0-9   Rx: 0-9
        NSS3 Tx: 0-9   Rx: 0-9
WPS: V2.0 Configured
VS_IE:dd7c0050f204104a0001101044000102103b000103104700101b52c4d5ffb10ad163f39b91a979382c102
1000d4e4554474541522c20496e632e1023000552363330301024000552363330301042000434353336105400
00800060050f20400011011000552363330301008000220081 03c000103104900060 0372a000120
VS_IE:dd090010180200001c0000
VS_IE:dd180050f20201018c0003a4000027a400004243bc0062326600

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

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

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

In the following test sequences of using "***wl***" tool or "***iw***" command, the WLAN interface ***is not assigned*** an IP address. That is done later in **Section 7.1.7** where connectivity testing is performed.

7.1.5.1 Using "iw" Linux Command

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

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

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

```
$ iw dev wlan0 link
Connected to 84:1b:5e:f6:a7:60 (on wlan0)
    SSID: Murata_5G
    freq: 5180
    RX: 1944 bytes (8 packets)
    TX: 0 bytes (0 packets)
    signal: -44 dBm
    tx bitrate: 24.0 Mbit/s
    bss flags:
    dtim period:    2
    beacon int:    100
```

7.1.5.2 Using “wl” Tool

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

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

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

```
$ wl join Murata_5G  🡰 connect to “Murata_5G” Access Point
IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
```

Check status of connection with “***wl assoc***” command:

```
$ wl assoc  🡰 check association status
SSID: "Murata_5G"
Mode: Managed   RSSI: -44 dBm   SNR: 0 dB       noise: -91 dBm  Flags: RSSI on-channel  Channel: 36/80
BSSID: 84:1B:5E:F6:A7:60        Capability: ESS
Supported Rates: [ 6(b) 9 12(b) 18 24(b) 36 48 54 ]
Extended Capabilities: BSS_Transition
VHT Capable:
        Chanspec: 5GHz channel 42 80MHz (0xe02a)
        Primary channel: 36
        HT Capabilities: 40Mhz SGI20 SGI40
        Supported HT MCS : 0-7
        Supported VHT MCS:
                NSS1 Tx: 0-9   Rx: 0-9
                NSS2 Tx: 0-9   Rx: 0-9
                NSS3 Tx: 0-9   Rx: 0-9
WPS: V2.0 Configured
VS_IE:dd310050f204104a0001101044000102104700101b52c4d5ffb10ad163f39b91a979382c103c000103104
9000600372a000120
VS_IE:dd090010180200001c0000
VS_IE:dd180050f20201018c0003a4000027a400004243bc0062326600
VS_IE:dd7c0050f204104a0001101044000102103b000103104700101b52c4d5ffb10ad163f39b91a979382c102
1000d4e4554474541522c20496e632e1023000552363330301024000552363330301042000434353336105400
0800060050f2040001101100055236333030100800022008103c0001031049000600372a000120
```

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

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

7.1.6.1 Using "wpa_cli" Command

"***wpa_cli***" can only be invoked once the "***wlan0***" interface is configured and the WPA supplicant is running. The user can follow this command sequence to establish a secure client connection to an Access Point with WPA2-PSK authentication. Prior to running "***wpa_cli***", you might like to back up default/previous "***/etc/wpa_supplicant.conf***" file in case you overwrite it:

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

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

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

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

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

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

```
> remove_network all  ← tear down any existing network connections
OK
<3>CTRL-EVENT-DISCONNECTED bssid=b0:00:b4:65:e0:60 reason=3 locally_generated=1
> status              ← check status
wpa_state=INACTIVE
p2p_device_address=ba:d7:af:56:61:fc
address=b8:d7:af:56:61:fc
uuid=a3178764-c106-57ee-8ec2-6bf2e0607166
> scan                   ← initiate a scan
OK
<3>CTRL-EVENT-SCAN-STARTED
```

```
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>CTRL-EVENT-NETWORK-NOT-FOUND
> scan_results                  🡐 list results of scan
bssid / frequency / signal level / flags / ssid
84:1b:5e:f6:a7:60      5180   -38    [WPA2-PSK-CCMP][WPS][ESS]      Murata_5G
84:1b:5e:f6:a7:61      2412   -36    [WPA2-PSK-CCMP][WPS][ESS]      Murata_2G
…
> add_network   🡐 add a network. This returns integer value which is then used for setting parameters.
0
> set_network 0 ssid "Murata_5G"                 🡐 set SSID to "Murata_5G"
OK
> set_network 0 psk "your_passphrase"            🡐 set WPA passphrase
OK
> enable 0   🡐 enable network connection. If ssid and passphrase set correctly, connection will be established.
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>Trying to associate with SSID 'Murata_5G'
<3>Associated with 84:1b:5e:f6:a7:60             🡐 connection established
<3>CTRL-EVENT-CONNECTED - Connection to 84:1b:5e:f6:a7:60 completed [id=0 id_str=]
<3>CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
 > status                       🡐 now verify that connection is established
bssid=84:1b:5e:f6:a7:60
freq=5180
ssid=Murata_5G
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
p2p_device_address=ba:d7:af:56:61:fc
address=b8:d7:af:56:61:fc
uuid=a3178764-c106-57ee-8ec2-6bf2e0607166
> save_config      🡐 save current configuration: this overwrites "/etc/wpa_supplicant.conf" file!
OK
> quit             🡐 exit wpa_cli interactive mode
```

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

```
$ more /etc/wpa_supplicant.conf

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

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

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

7.1.6.2 Using "wpa_supplicant.conf" file

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

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

network={
      key_mgmt=NONE
}
```

With the default configuration, the WPA supplicant will establish a connection with any random Access Point that has no authentication scheme enabled (i.e. "open"). Using "***Murata_5G***" SSID example, the relevant/modified contents of the "/***etc/wpa_supplicant.conf***" file (already shown in **Section 7.1.6.1)** is:

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

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

To establish a secured WPA2-PSK connection by only modifying "***/etc/wpa_supplicant.conf***" file:

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

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

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

Verify that connection is re-established with Access Point:

```
$ iw dev wlan0 link
Connected to 84:1b:5e:f6:a7:60 (on wlan0)
      SSID: Murata_5G
      freq: 5180
      RX: 1659 bytes (7 packets)
      TX: 264 bytes (2 packets)
      signal: -45 dBm
      tx bitrate: 24.0 MBit/s
      bss flags:
      dtim period:    2
      beacon int:     100
```

7.1.7 STA/Client Mode: Basic WLAN Connectivity Testing

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

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

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

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

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

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

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

7.1.8 Wi-Fi Direct Testing

In this section we use the "***wpa_cli***" tool to configure the i.MX6/Murata Wi-Fi platform as a P2P Group Owner (P2P GO). Another device (i.e. another i.MX platform or smartphone) is required to act as the P2P Client. Together the P2P GO and Client devices establish a P2P Group. Note that WPA supplicant must be up and running. Prior to running "***wpa_cli***", you might like to back up default/previous "***/etc/wpa_supplicant.conf***" file in case you overwrite it:

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

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

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

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

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

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

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

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

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

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

7.1.9 Soft AP or Wi-Fi Hot Spot Testing

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

<u>Important Dependencies:</u>

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

First off, we need to kill the WPA supplicant:

```
$ killall wpa_supplicant
```

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

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

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

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

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

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

7.1.10 WLAN Manufacturing, RF or Regulatory Testing

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

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

NOTE: For ***"fmac"*** release ***"kong"*** and older, the folder location is ***"/lib/firmware/brcm"***

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

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

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

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

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

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

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

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

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

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

```
$ killall wpa_supplicant
```

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

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

7.2 Bluetooth Interface Test/Verification

For Murata modules supporting Bluetooth, we can verify the HCI UART connection by invoking "***hciattach***", bringing up the interface with "***hciconfig***" and then invoking "***hcitool scan***" to see what Bluetooth devices are visible. The Bluetooth test commands vary depending on which UART port is connected to Bluetooth. With the new "***modem_reset***" construct in DTS(I) files, the Bluetooth core should come up in the correct state to be initialized (i.e. as kernel boots, the Bluetooth core is reset and is taken out of reset). When the BlueZ "***hciattach***" call is invoked, a Bluetooth patch file is pulled from the "/etc/firmware/" folder – refer to **Table 13** for more details. Here is the default command sequence to verify Bluetooth functionality:

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

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

Table 14 lists the BT_REG_ON GPIO and UART ports for the various i.MX platforms. Here is example output using i.MX 6ULL EVK with Type 1MW module (no BT_REG_ON toggle):

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

Table 14: GPIO and UART Settings for Bluetooth Tests

i.MX6 Platform	GPIO/UART Configuration	Notes
i.MX 8MQuad EVK	GPIO69; UART3	Type 1CX soldered down and M.2 EVB.
i.MX 8M Mini/Nano EVK (1MW)	GPIO37; UART1	Type 1MW soldered down.
i.MX 8M Mini/Nano EVK (uSD)	GPIO140; UART3	uSD-M.2 Adapter interconnect with M.2 EVB.
i.MX 7ULP EVK	LPUART6	Type 1DX soldered down.
i.MX 6Q(P)/DL SDB	GPIO2; UART5	uSD-M.2 Adapter interconnect with M.2 EVB.
i.MX 6SX SDB	GPIO171; UART5	uSD-M.2 Adapter interconnect with M.2 EVB.
i.MX 6UL/ULL EVK	GPIO508; UART2	GPIO508 does not allow its direction to be set. Always output.

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

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

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

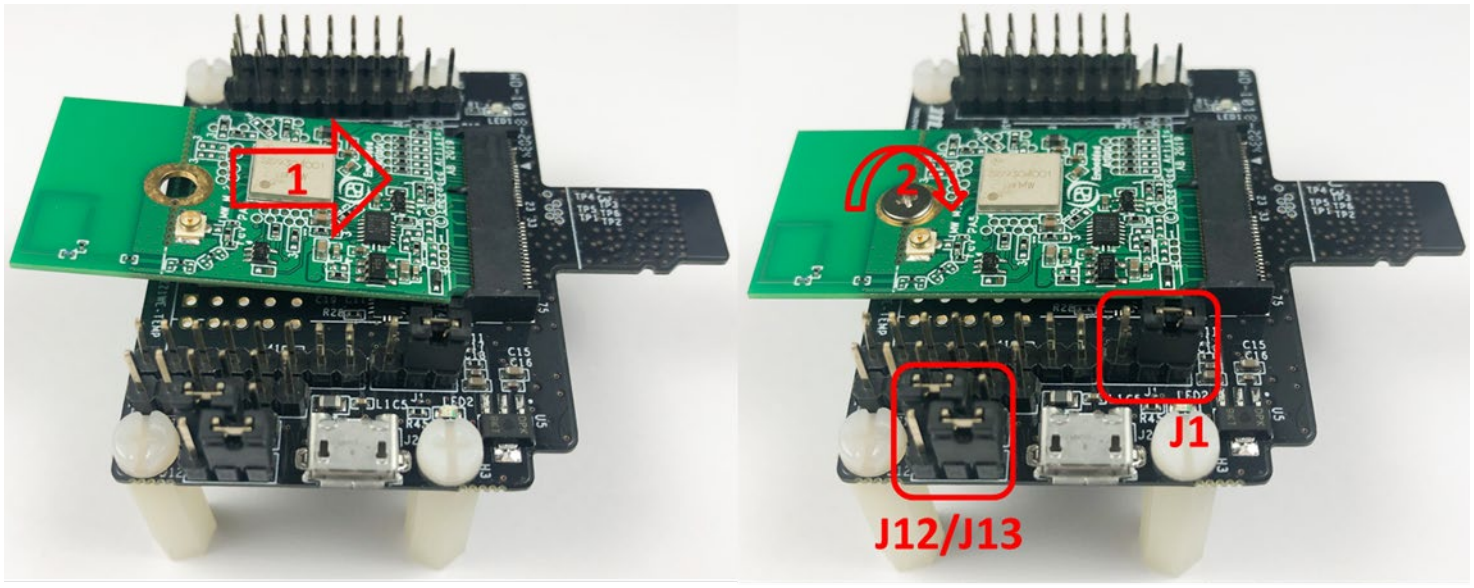
```
&uart3 { /* BT */
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_uart3>;
    assigned-clocks = <&clk IMX8MQ_CLK_UART3_SRC>;
    assigned-clock-parents = <&clk IMX8MQ_SYS1_PLL_80M>;
    fsl,uart-has-rtscts;
//  resets = <&modem_reset>;   🡐 Comment out “resets” line on BT UART descriptor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

8 Murata's uSD-M.2 Adapter

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

When connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter Rev B1 (**Figure 27**), 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 27: Connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter

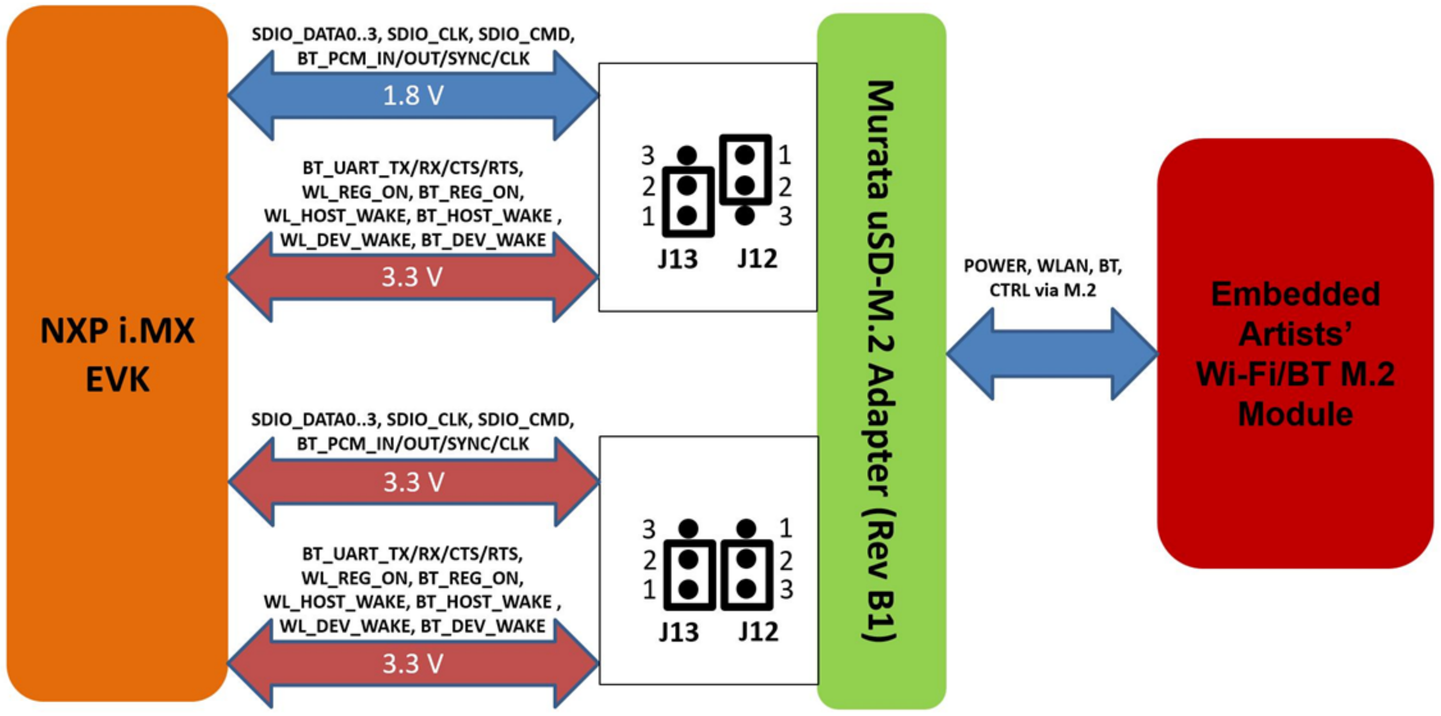


8.2 Configuring uSD-M.2 Adapter Jumpers for Correct VIO Signaling

Figure 28 shows a block diagram highlighting the Host (i.MX EVK) and Wi-Fi/BT M.2 EVB VIO signaling voltages. The legacy i.MX 6 EVK's (excluding the i.MX 6UL(L) EVK's) have J13/J12 set to 1-2/2-3 positions respectively for the 3.3V VIO override mode setting (WLAN-SDIO, BT-UART, and WLAN/BT control signals all at 3.3V VIO).

All other i.MX EVK's (i.MX 8M Mini/Nano & i.MX 6UL(L) EVK's) have J13/J12 set to 1-2/1-2 positions respectively for the 1.8V VIO default configuration (WLAN-SDIO VIO at 1.8V VIO; BT-UART and WLAN/BT control signals at 3.3V VIO).

Figure 28: Host/M.2 IO Voltage Level Shift Options on Rev B1 Adapter


NXP i.MX EVK
SDIO_DATA0..3, SDIO_CLK, SDIO_CMD, BT_PCM_IN/OUT/SYNC/CLK
1.8 V
BT_UART_TX/RX/CTS/RTS, WL_REG_ON, BT_REG_ON, WL_HOST_WAKE, BT_HOST_WAKE , WL_DEV_WAKE, BT_DEV_WAKE
3.3 V
SDIO_DATA0..3, SDIO_CLK, SDIO_CMD, BT_PCM_IN/OUT/SYNC/CLK
3.3 V
BT_UART_TX/RX/CTS/RTS, WL_REG_ON, BT_REG_ON, WL_HOST_WAKE, BT_HOST_WAKE , WL_DEV_WAKE, BT_DEV_WAKE
3.3 V
3 2 1
1 2 3
J13
J12
3 2 1
1 2 3
J13
J12
Murata uSD-M.2 Adapter (Rev B1)
POWER, WLAN, BT, CTRL via M.2
Embedded Artists' Wi-Fi/BT M.2 Module

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

On both legacy NXP i.MX 6 EVK's and the newer i.MX 8 EVK's, 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 EVK's; 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 EVK's, Murata ***strongly recommends*** to simply tape the uSD Adapter-SD Card connection and the SD Card-EVK connection as shown in **Figure 29**. 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 EVK's, Murata ***strongly recommends*** to simply tape the uSD Adapter-EVK connection as shown in **Figure 30**. 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 29: Securing uSD/SD Connection on i.MX 6 EVK

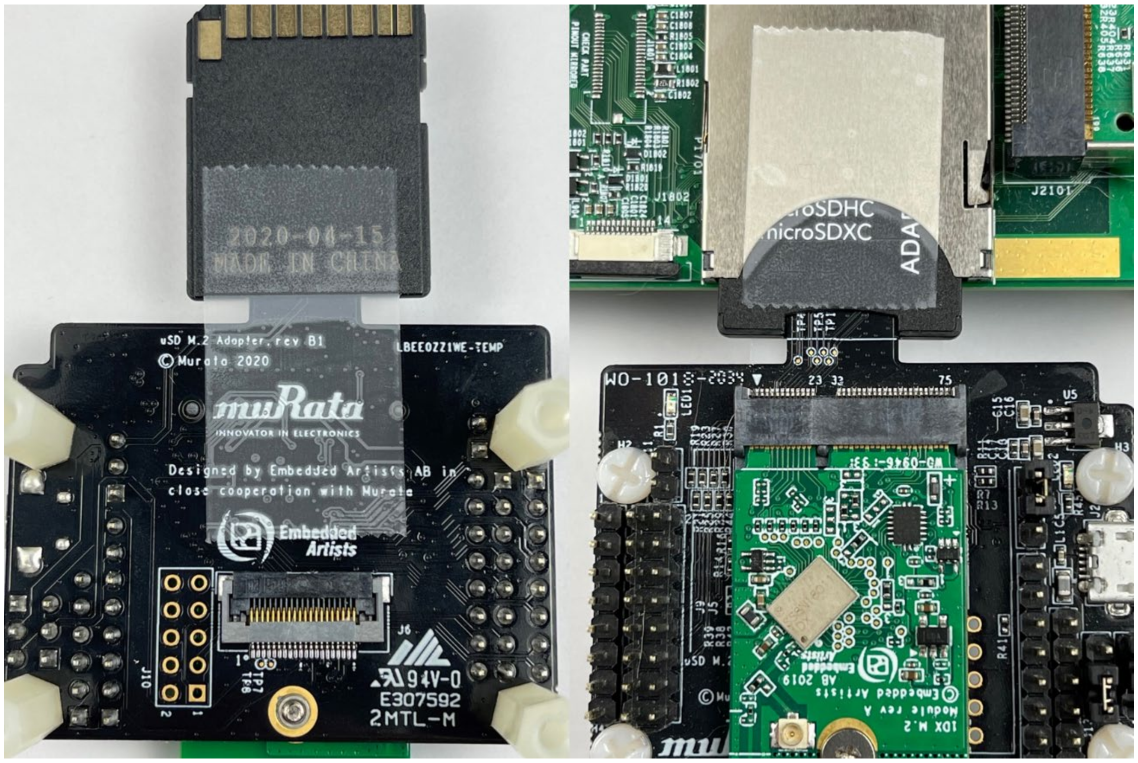
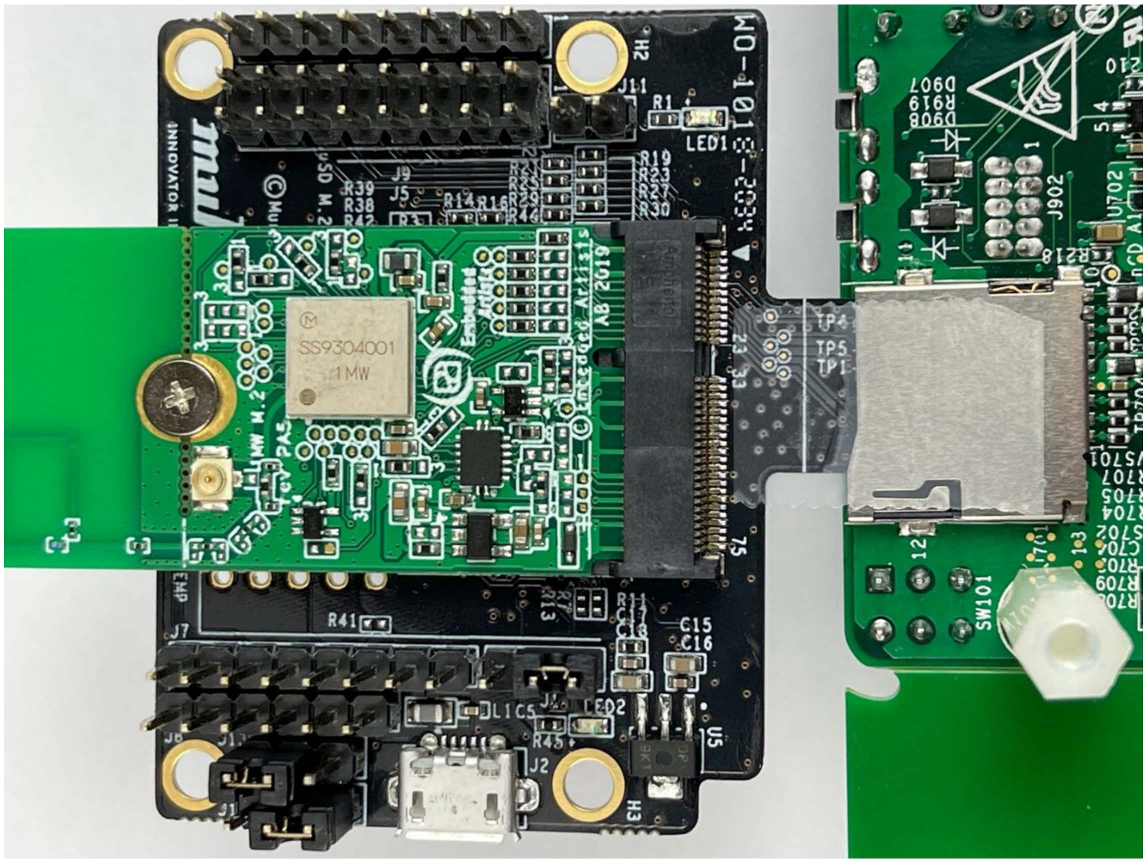


Figure 30: Securing uSD Connection on i.MX 8 EVK



8.4 uSD-M.2 Adapter High-Level Description

Figure 31 and **Figure 32** show the features on the uSD-M.2 Adapter; with text explanation in **Table 15.** 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 15: uSD-M.2 Adapter Features

Char	Description
A	microSD connector provides Power (VBAT, GND) and WLAN-SDIO
B	SDIO bus test points (CLK, CMD, DAT0, DAT1, DAT2, DAT3)
C	Power LED Indicator (green): if not illuminated then no power applied to M.2 EVB
D	J11 = Optional BT Disable Jumper for WLAN-Only Mode (close this jumper to drive BT_REG_ON low and disable Bluetooth Core; thereby optimizing power consumption)
E	J9 = BT UART TX/RX and WLAN/BT Control Signals (8 pin header)
F	J5 = Optional BT PCM and WLAN/BT Debug Signals (2x8 pin header)
G	Threaded mount for M.2 screw: 30mm 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 (32.768kHz)
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 EVB's have own 1.8V onboard)
Q	J1 = Power Supply Selector Jumper must be installed to power Adapter (unless J5 Arduino Header Pins #15/16 are connected to external GND/3.3V VBAT). **Position 1-2:** 5V/3.3V VBAT supply from micro-USB (J2); or Arduino (J7) **Position 2-3:** VBAT supply (typical 3.1~3.3V) from microSD connector
R	M.2 Connector: type 2230-xx-E
S	microSD connector pins: provides Power (VBAT, GND) and WLAN-SDIO
T	WLAN JTAG header (header pins not populated)
U	20 pin FFC connector (BT UART, BT PCM, WLAN/BT Control signals)
V	Additional test points from 20pin flat/flex connector

Figure 31: uSD-M.2 Adapter Features (Top View)

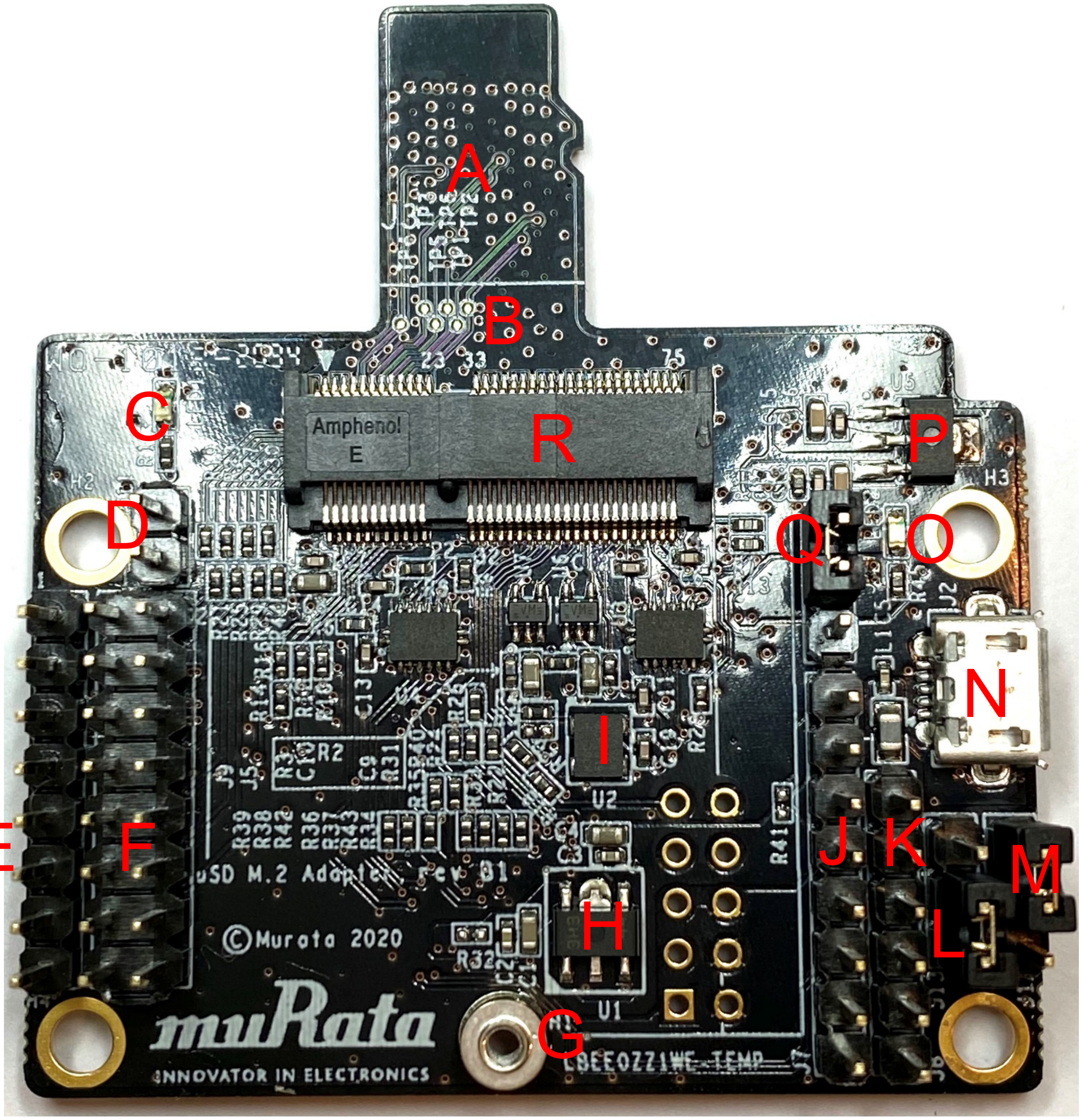
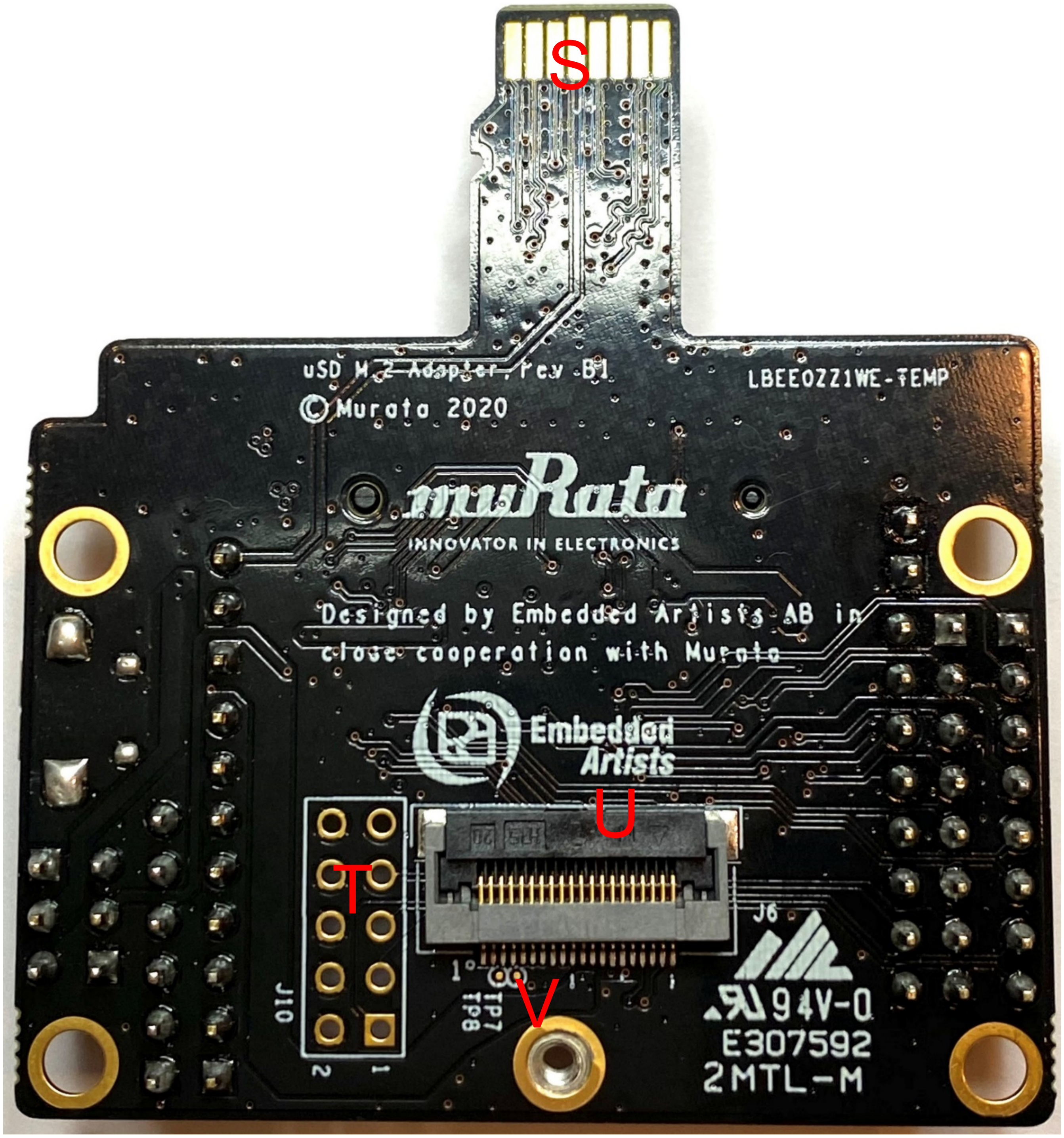


Figure 32: uSD-M.2 Adapter Features (Bottom View)



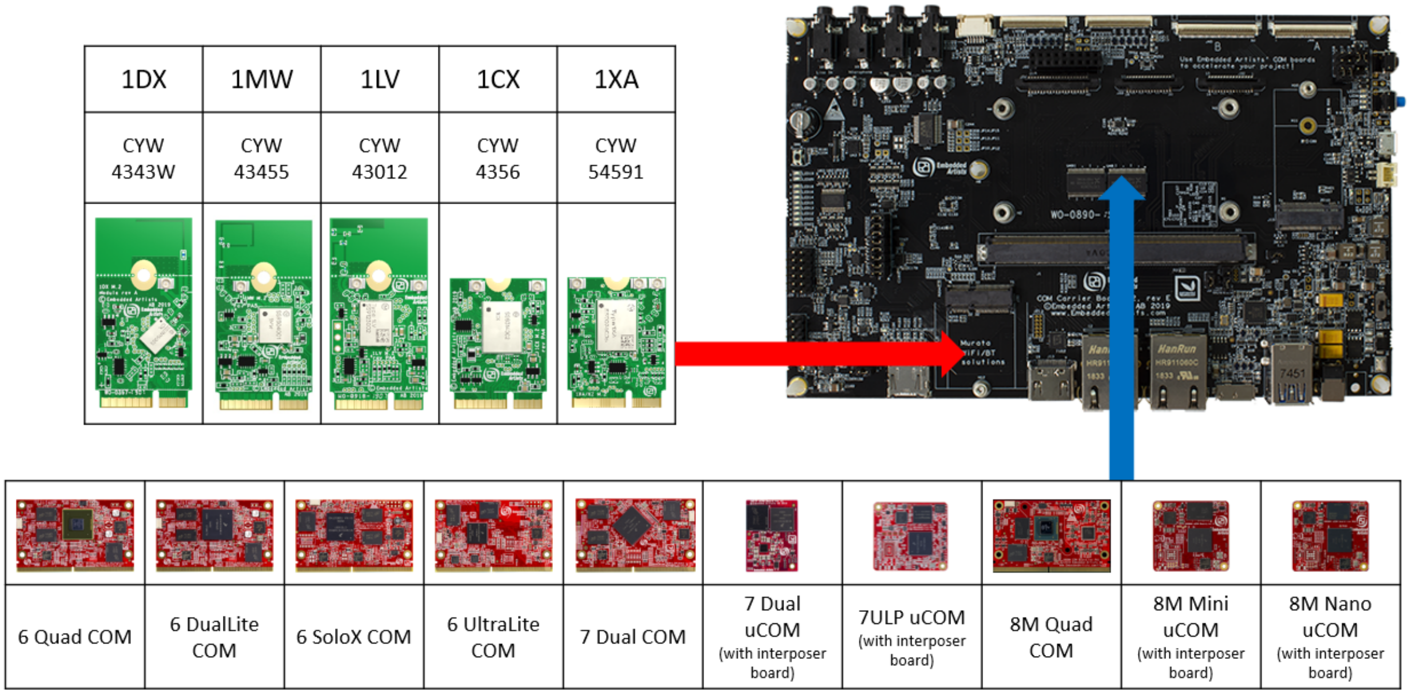
9 Embedded Artists' Wi-Fi/BT M.2 Modules

Embedded Artists designs, manufactures, and distributes all Wi-Fi/Bluetooth M.2 modules featuring Murata's mass-market modules (currently 1DX, 1MW, 1LV, 1CX, and 1XA for Cypress-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 EVB's from Distributors (Mouser, Digi-Key, Future, and Arrow). For more information on the M.2 solution, please refer to www.embeddedartists.com/m2/.

10 Embedded Artists' i.MX + Wireless Solution

Murata has partnered with Embedded Artists to provide the ultimate solution for customers to evaluate Wi-Fi/Bluetooth modules easily and quickly. This solution is composed of three parts: Carrier board (baseboard), Computer on Module (COM) board, and Wi-Fi/BT M.2 Modules (EVB's). **Figure 33** shows that the carrier board can work with a variety of NXP i.MX6/7/8 (u)COM boards and five different Murata-module based EVBs. With this platform, users can evaluate multiple i.MX processor and Wi-Fi/BT module permutations to find the best combination for their product. Also, Embedded Artists brings out all the test points you need for troubleshooting. With this platform, no adapter/interconnect is needed – either module-down solution or well-secured M.2 module. This is beneficial in two main areas: no limitation in Wi-Fi performance (WLAN-SDIO bus runs at full speed); and no mechanical issues (easy to secure M.2 module to EA Carrier Board). **Figure 33** shows how this platform works with (u)COM board and Wi-Fi/BT M.2 Modules. **Table 16** provides an Embedded Artists' i.MX (u)COM versus Murata module matrix. For comprehensive information on the Embedded Artists' solution refer to **Table 17**.

Figure 33: Combine i.MX COM with Wi-Fi/BT M.2 EVB



1DX	1MW	1LV	1CX	1XA
CYW 4343W	CYW 43455	CYW 43012	CYW 4356	CYW 54591

6 Quad COM	6 DualLite COM	6 SoloX COM	6 UltraLite COM	7 Dual COM	7 Dual uCOM (with interposer board)	7ULP uCOM (with interposer board)	8M Quad COM	8M Mini uCOM (with interposer board)	8M Nano uCOM (with interposer board)

Table 16: Embedded Artists' i.MX Interconnect

EA i.MX (u)COM	1DX	1MW	1LV	1CX	1XA
	CYW4343W	**CYW43455**	**CYW43012**	**CYW4356**	**CYW54591**
iMX8M Quad COM	M.2	M.2	M.2	M.2	M.2
iMX8M Mini uCOM	M.2	M.2 / MD	M.2	M.2	M.2
iMX8M Nano uCOM	M.2	M.2 / MD	M.2	NC	NC
iMX7 Dual COM	M.2	M.2	M.2	M.2	M.2
iMX7 Dual uCOM	M.2	M.2	M.2	M.2	M.2
iMX7ULP uCOM	M.2	M.2	M.2 / MD	NC	NC
iMX6 Quad COM	M.2	M.2	M.2^{V}	M.2	M.2
iMX6 DualLite COM	M.2	M.2	M.2^{V}	M.2	M.2
iMX6 SoloX COM	M.2	M.2	M.2^{V}	M.2	M.2
iMX6 UltraLite COM	M.2	M.2	M.2	NC	NC

M.2 = Works with onboard M.2 slot.
MD = Module soldered down.
NC = No Connect.
M.2^{V} = These platforms have fixed 3.3 V VIO for WLAN-SDIO. Although 1LV is 1.8V only, testing has yielded reliable results.

Table 17: Embedded Artists' Landing Pages

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

Table 18 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 18: Embedded Artists' Datasheets and Schematics

Datasheets and Schematics	Notes
i.MX 6/7/8 COM Carrier Board V2 Datasheet	Comprehensive definition of COM Carrier (baseboard).
i.MX6/7/8 COM Carrier Board V2 Schematics	Complete schematics including clear definition of uSD-M.2 Adapter.
M.2 SDIO Interface Schematic	Reference schematic for customers designing in WLAN-SDIO M.2 EVB.
M.2 PCIe Interface Schematic	Reference schematic for customers designing in WLAN-PCIe M.2 EVB.
EACOM Board Specification Guide	Comprehensive definition of Embedded Artists' Computer-On-Module's.
1DX M.2 Module Datasheet	Comprehensive details on 1DX Wi-Fi/BT M.2 Module.
1LV M.2 Module Datasheet	Comprehensive details on 1LV Wi-Fi/BT M.2 Module.
1MW M.2 Module Datasheet	Comprehensive details on 1MW Wi-Fi/BT M.2 Module.
1CX M.2 Module Datasheet	Comprehensive details on 1CX Wi-Fi/BT M.2 Module.

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

Table 19 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 our Community Forum: https://community.murata.com. Customers are welcome to register and post any questions they may have.

Table 19: 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 EVB's on EA's i.MX 6/7/8 Dev Kits.
i.MX Working with Yocto	Comprehensive guide on building Linux images using Yocto framework.
Linux i.MX Images Download	Pre-compiled images using "uuu" tool: allows users to easily flash i.MX platforms with latest Linux images with integrated Wi-Fi/BT support.
Wi-Fi/BT M.2 EVB Primer	Introduction and drill-down on M.2 interface.

11 Technical Support Contact

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

Table 20: List of Support Resources

Support Site	Notes
Murata Community Forum	**Primary support point for technical queries.** This is an open forum for all customers. Registration is required.
Murata i.MX Landing Page	**No** login credentials required. Murata documentation covering hardware, software, testing, etc. is provided here.
Murata uSD-M.2 Adapter Landing Page	Landing page for uSD-M.2 Adapter. In conjunction with Murata i.MX Landing Page, this should provide the user with comprehensive getting started documentation.
Murata Module Landing Page	**No** login credentials required. Murata documentation covering all Cypress-based Wi-Fi/BT modules is provided here.

12 Additional Useful Links

In addition to **Table 20** listings of support resources, **Table 21** provides some useful links.

Table 21: Additional Useful Links

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