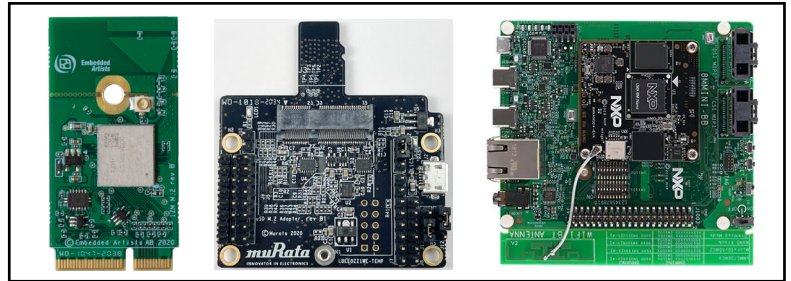


**Murata Wi-Fi/BT
(NXP) Solution for i.MX**

Linux User Guide



Revision History

Revision	Date	Author	Change Description
1.0	Nov 17, 2020	TF	Initial Release.
1.1	Jan 28, 2021	TF	Removed unsupported hardware and software configurations.

Table of Contents

REVISION HISTORY	1
TABLE OF CONTENTS	2
1 INTRODUCTION	6
1.1 NXP i.MX 6 Platform Support.....	7
1.2 NXP i.MX 8 Platform Support.....	8
1.3 Acronyms	10
1.4 References.....	11
1.4.1 Murata Wi-Fi/BT (NXP) Solution for i.MX Linux User Manual.....	11
1.4.2 Murata Wi-Fi/BT (NXP) Solution for i.MX Linux Quick Start Guide.....	11
1.4.3 Murata Wi-Fi/BT Solution for i.MX Hardware User Manual.....	11
1.4.4 Murata’s Community Forum Support	11
1.4.5 Murata uSD-M.2 Adapter Datasheet (Rev B1).....	11
1.4.6 Murata uSD-M.2 Adapter Datasheet (legacy Rev A)	11
1.4.7 Embedded Artists’ Reference Documentation	11
1.4.8 Murata’s i.MX Wireless Solutions Landing Page	11
1.4.9 NXP Reference Documentation	12
2 WI-FI/BT HARDWARE SOLUTION FOR I.MX	13
2.1 Embedded Artists’ Wi-Fi/BT M.2 EVB’s	13
2.2 Murata’s uSD-M.2 Adapter.....	13
2.3 NXP i.MX versus Murata Module Interconnect.....	15
3 WI-FI/BT SOFTWARE SOLUTION FOR I.MX	16
3.1 Murata’s Customized NXP Wireless Driver Release	16
3.2 Specific i.MX Target Support Details	17
3.3 Murata Customized i.MX Yocto Image Build	18
3.3.1 Install Ubuntu.....	18
3.3.2 Download Murata’s Script Files.....	19
3.3.3 Configure Ubuntu for i.MX Yocto Build	19
3.3.4 Murata’s i.MX Yocto Build Script.....	20
3.4 Additional Hardware/Software Considerations	22
3.4.1 1.8V versus 3.3V VIO Signaling using Murata’s uSD-M.2 Adapter	22
3.4.2 UHS SDIO 3.0 operation on i.MX 6UL(L) EVK’s with uSD-M.2 Adapter	22
3.4.3 WLAN/Bluetooth M.2 Direct Interconnect on NXP i.MX Platforms	22
3.4.4 Setting Correct Software Configuration before testing Wi-Fi/BT Solution.....	23
3.4.5 Type 1YM M.2 EVB WLAN/Bluetooth Bus Interface Configuration	24
4 PREPARING NXP I.MX PLATFORMS TO BOOT LINUX IMAGE	26
4.1 Flashing Murata-customized Linux Image to (micro) SD Card	26
4.1.1 Linux PC Steps to Flash SD Card.....	26
4.1.2 Windows PC Steps to Flash SD Card.....	27
4.2 Flashing Murata-customized Linux Image to NXP i.MX 8M Mini/Nano EVK’s	28
4.2.1 Software File Preparation.....	28
4.2.2 i.MX 8M Mini or Nano EVK Hardware Configuration	29
4.2.3 Flash Linux Image to eMMC on i.MX 8M Mini/Nano EVK.....	31
4.2.4 Configure i.MX 8M Mini/Nano EVK to boot from eMMC	31
5 MURATA WI-FI/BT BRING-UP ON I.MX 6 PLATFORMS	33
5.1 Connecting to i.MX 6UL EVK or i.MX 6ULL EVK.....	34
6 MURATA WI-FI/BT BRING-UP ON I.MX 8 PLATFORMS	35
6.1 Bringing up Wi-Fi/BT on i.MX 8MQuad EVK.....	35
6.2 Bringing up Wi-Fi/BT on i.MX 8M Mini/Nano EVKs (M.2).....	36

6.3	Bringing up Wi-Fi/BT on i.MX 8M Mini/Nano EVK's (uSD-M.2 Adapter)	37
7	TEST/VERIFICATION OF WI-FI AND BLUETOOTH	46
7.1	Wi-Fi Interface Test/Verification	47
7.1.1	Useful Environment Setup on NXP Linux	47
7.1.2	Switch module script	47
7.1.3	Bringing Up Wi-Fi Interface	47
7.1.4	STA/Client Mode: Scan for Visible Access Points	49
7.1.5	STA/Client Mode: Connecting to Unsecured Access Point or Wireless Router	52
7.1.6	STA/Client Mode: Connecting to Secured Access Point or Wireless Router (WPA2-PSK)	52
7.1.7	STA/Client Mode: Basic WLAN Connectivity Testing	56
7.1.8	Wi-Fi Direct Testing	57
7.1.9	Soft AP or Wi-Fi Hot Spot Testing	58
7.2	Bluetooth Interface Test/Verification	59
7.2.1	Bringing Up 1YM-SDIO* Bluetooth Interface	60
8	MURATA'S USD-M.2 ADAPTER	61
8.1	Connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter	61
8.2	Configuring uSD-M.2 Adapter Jumpers for Correct VIO Signaling	61
8.3	Securing uSD-M.2 Adapter to NXP i.MX EVK	62
8.4	uSD-M.2 Adapter High-Level Description	64
9	EMBEDDED ARTISTS' WI-FI/BT M.2 MODULES	67
10	EMBEDDED ARTISTS' I.MX + WIRELESS SOLUTION	67
11	TECHNICAL SUPPORT CONTACT	70
12	ADDITIONAL USEFUL LINKS	70

LIST OF TABLES

Table 1:	Current NXP i.MX Platforms Supported	6
Table 2:	Acronyms used in User Guide	10
Table 3:	Embedded Artists Documentation Listing	12
Table 4:	NXP Reference Documentation Listing	12
Table 5:	Embedded Artists' Wi-Fi/BT M.2 Modules Supported	13
Table 6:	uSD-M.2 Adapter Kit Contents	14
Table 7:	NXP i.MX/Murata Module Interconnect	16
Table 8:	NXP i.MX EVK / Yocto (MACHINE) target / Kernel Version Matrix	17
Table 9:	i.MX6/8 Targets supported by Murata	17
Table 10:	Select Hardware Configurations which use eMMC Boot Configuration	26
Table 11:	Files to flash i.MX 8M Mini & 8M Nano EVK's	28
Table 12:	i.MX 8M Mini/Nano EVK Jumper connections to uSD-M.2 Adapter	39
Table 13:	Embedded Wi-Fi/Bluetooth Files	46
Table 14:	GPIO and UART Settings for Bluetooth Tests	60
Table 15:	uSD-M.2 Adapter Features	64
Table 16:	Embedded Artists' i.MX Interconnect	68
Table 17:	Embedded Artists' Landing Pages	68
Table 18:	Embedded Artists' Datasheets and Schematics	69
Table 19:	Embedded Artists' User Manuals and Software	70
Table 20:	List of Support Resources	70
Table 21:	Additional Useful Links	70

LIST OF FIGURES

Figure 1: i.MX 6 EVK Wi-Fi/BT Interconnect Block Diagram	7
Figure 2: i.MX 8QXP MEK or 8MQuad EVK Wi-Fi/BT Interconnect Block Diagram	8
Figure 3: i.MX 8M Mini EVK Wi-Fi/BT PCIe Interconnect Block Diagram	9
Figure 4: i.MX 8M Mini/Nano EVK Wi-Fi/BT SDIO Interconnect Block Diagram	9
Figure 5: Configuring dash	18
Figure 6: Type 1YM M.2 EVB Configuration Strapping Options	24
Figure 7: Type 1YM M.2 Configured for WLAN-SDIO/BT-SDIO (1YM-SDIO*)	25
Figure 8: Type 1YM M.2 Configured for WLAN-SDIO/BT-UART (1YM-SDIO)	25
Figure 9: USB to SD Card Reader/Writer Adapter	26
Figure 10: Power, Download, and Debug port connection to board	29
Figure 11: i.MX 8M Mini EVK DIP Switches configured for Download	30
Figure 12: i.MX 8M Nano EVK DIP Switches configured for Download	30
Figure 13: i.MX 8M Mini EVK DIP Switches configured for eMMC Boot	32
Figure 14: i.MX 8M Nano EVK DIP Switches configured for eMMC Boot	32
Figure 15: uSD-M.2 Adapter with Type 1ZM and 1YM-SDIO* M.2 EVB options	33
Figure 16: i.MX 6ULL EVK with uSD-M.2 Adapter and Type 1ZM M.2 EVB	34
Figure 17: i.MX 8MQuad with Type 1YM (bottom view)	35
Figure 18: i.MX 8M Mini with Type 1YM-PCIe (bottom view)	36
Figure 19: i.MX 8M Mini/Nano EVK with uSD-M.2 Adapter (WLAN Only)	37
Figure 20: i.MX 8M Mini/Nano EVK with uSD-M.2 Adapter (WLAN/Bluetooth)	37
Figure 21: Cable connections on i.MX 8M Mini EVK (Even Number Connector View)	40
Figure 22: Cable connections on i.MX 8M Mini EVK (Odd Number Connector View)	41
Figure 23: Cable connections on uSD-M.2 Adapter (J9 Header)	42
Figure 24: Cable connections on uSD-M.2 Adapter (J8 Header)	43
Figure 25: Low-Profile Jumper Wires (Digi-Key part number 1988-1178-ND)	44
Figure 26: NXP i.MX EVK with Low-Profile Jumper Wires	44
Figure 27: Additional Hex Standoff (Digi-Key part number RPC3570-ND)	45
Figure 28: Connecting the Wi-Fi/BT M.2 EVB to uSD-M.2 Adapter	61
Figure 29: Host/M.2 IO Voltage Level Shift Options on Rev B1 Adapter	62
Figure 30: Securing uSD/SD Connection on i.MX 6 EVK	63
Figure 31: Securing uSD Connection on i.MX 8 EVK	63
Figure 32: uSD-M.2 Adapter Features (Top View)	65
Figure 33: uSD-M.2 Adapter Features (Bottom View)	66
Figure 34: Combine i.MX COM with Wi-Fi/BT M.2 EVB	67

This page left intentionally blank.

1 Introduction

Murata has partnered with [NXP Semiconductors N.V.](#) and [Embedded Artists AB](#) to offer a complete Wi-Fi and Bluetooth connectivity environment for building world class Internet-connected products on NXP's family of i.MX Processors. The Murata Connectivity Modules enable developers to minimize the development time and effort for connectivity function implementation. This document details enabling Murata Wi-Fi/BT solutions on reference NXP i.MX platforms. The current Linux i.MX release supports NXP's [L5.4.47 2.2.0 BSP](#). Following steps are described in this document:

- How to build Murata's customized image and flash it to various NXP i.MX EVK's, thereby enabling the NXP WLAN driver and associated components (WPA supplicant, WLAN firmware, calibration files, regulatory DB file, etc.).
- Connect and/or configure applicable Wi-Fi/BT solution for a given NXP i.MX EVK & power up.
- Initialize & configure WLAN and Bluetooth interfaces.
- Exercise WLAN and Bluetooth functionality.

The NXP Platforms currently supported are based on i.MX 8 and i.MX 6 series. The wireless solution for the following platforms is provided by connecting [Embedded Artists' Wi-Fi/BT M.2 EVB](#) directly to the NXP i.MX EVK; or using [Murata's uSD-M.2 Adapter](#) as an interconnect solution. Refer to **Table 1**. Note that the uSD-M.2 Adapter only supports WLAN-SDIO configuration, whereas all instances of M.2 connect currently only support WLAN-PCIe due to NXP i.MX reference platform configuration. Type 1ZM M.2 EVB supports WLAN-SDIO interface only. Type 1YM M.2 EVB serves "double duty" in supporting both WLAN-PCIe (default strapping configuration) and WLAN-SDIO options.

Table 1: Current NXP i.MX Platforms Supported

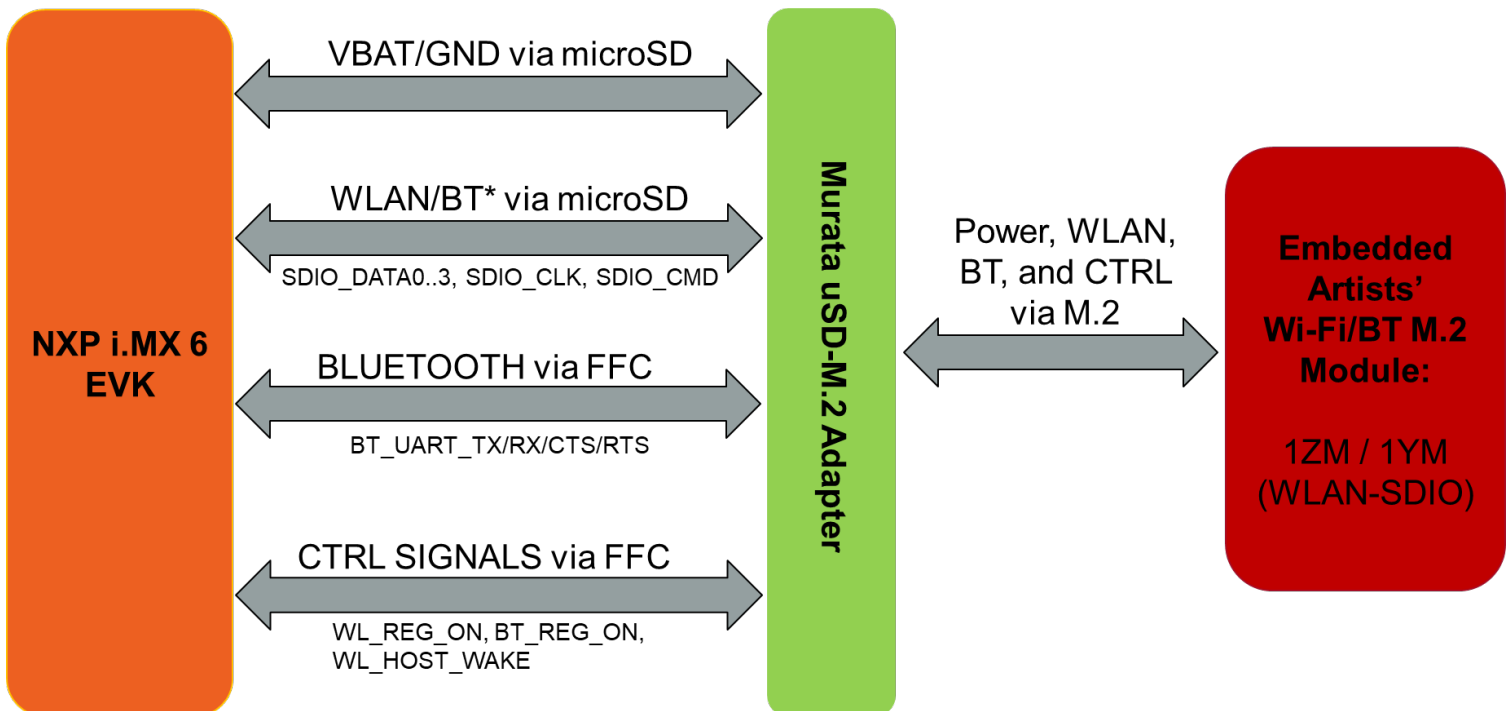
NXP i.MX EVK Part number	NXP i.MX EVK	Murata Modules Supported	Inter-Connect
MCIMX8QXP-CPU	i.MX 8QXP MEK	1YM	M.2 (WLAN-PCIe)
MCIMX8M-EVKB	i.MX 8MQuad EVK	1YM	M.2 (WLAN-PCIe)
8MMINILPD4-EVK	i.MX 8M Mini EVK	1ZM,1YM	uSD-M.2 Adapter: 1ZM, 1YM M.2 (WLAN-PCIe): 1YM – WLAN Only
8MMINID4-EVK	i.MX 8M Mini EVK	1YM	M.2 (WLAN-PCIe) – WLAN Only
8MNANOD4-EVK	i.MX 8M Nano EVK	1ZM, 1YM	uSD-M.2 Adapter: 1ZM, 1YM M.2 (WLAN-PCIe): 1YM – WLAN Only
MCIMX6UL-EVKB	i.MX 6UL EVK	1ZM, 1YM	uSD-M.2 Adapter
MCIMX6ULL-EVK	i.MX 6ULL EVK	1ZM, 1YM	uSD-M.2 Adapter

1.1 NXP i.MX 6 Platform Support

A high-level connection Diagram for the Murata uSD-M.2 Adapter (i.MX 6 platforms) is provided in **Figure 1**. All the Murata Wi-Fi/BT modules enabled by this release are shown. The wireless solution is arrived at by combining [Murata's uSD-M.2 Adapter](#) with [Embedded Artists' Wi-Fi/BT M.2 EVB](#). Refer to **Section 8** and **Section 9** for more details on the uSD-M.2 Adapter and Wi-Fi/BT M.2 Modules. 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).
- Strapping options on specific Wi-Fi/BT M.2 EVB's that support multiple bus interface configurations. Currently this is limited to Type 1YM M.2 Module (supporting WLAN-PCIe and WLAN-SDIO configuration options).

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



Regarding the Murata modules currently supported:

- Type 1ZM supports WLAN-SDIO and BT-UART interfaces only.
- Type 1YM supports WLAN-PCIe/BT-UART (default). It can be configured (via strapping options) to support additional evaluation mode (currently supported in Murata customized software) of WLAN-SDIO/BT-SDIO. This strapping configuration is denoted as “1YM-SDIO*”.
- Eventual long-term software support for **1YM will support WLAN-PCIe/BT-UART and WLAN-SDIO/BT-UART only**. The 1YM M.2 EVB (WLAN-SDIO/BT-UART) is denoted as “1YM-SDIO”. For more information on Type 1YM M.2 EVB configuration refer to **Section 3.4.5**.

1.2 NXP i.MX 8 Platform Support

Figure 2 shows a simplified block diagram for the i.MX 8MQuad EVK Wi-Fi/BT interconnect. Type 1YM M.2 EVB is supported over the WLAN-PCIe/BT-UART interfaces. Note that **no** uSD-M.2 Adapter is used for i.MX 8MQuad EVK – just the Wi-Fi/BT M.2 EVB (Module). The i.MX 8QXP MEK is supported by this same configuration as well.

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

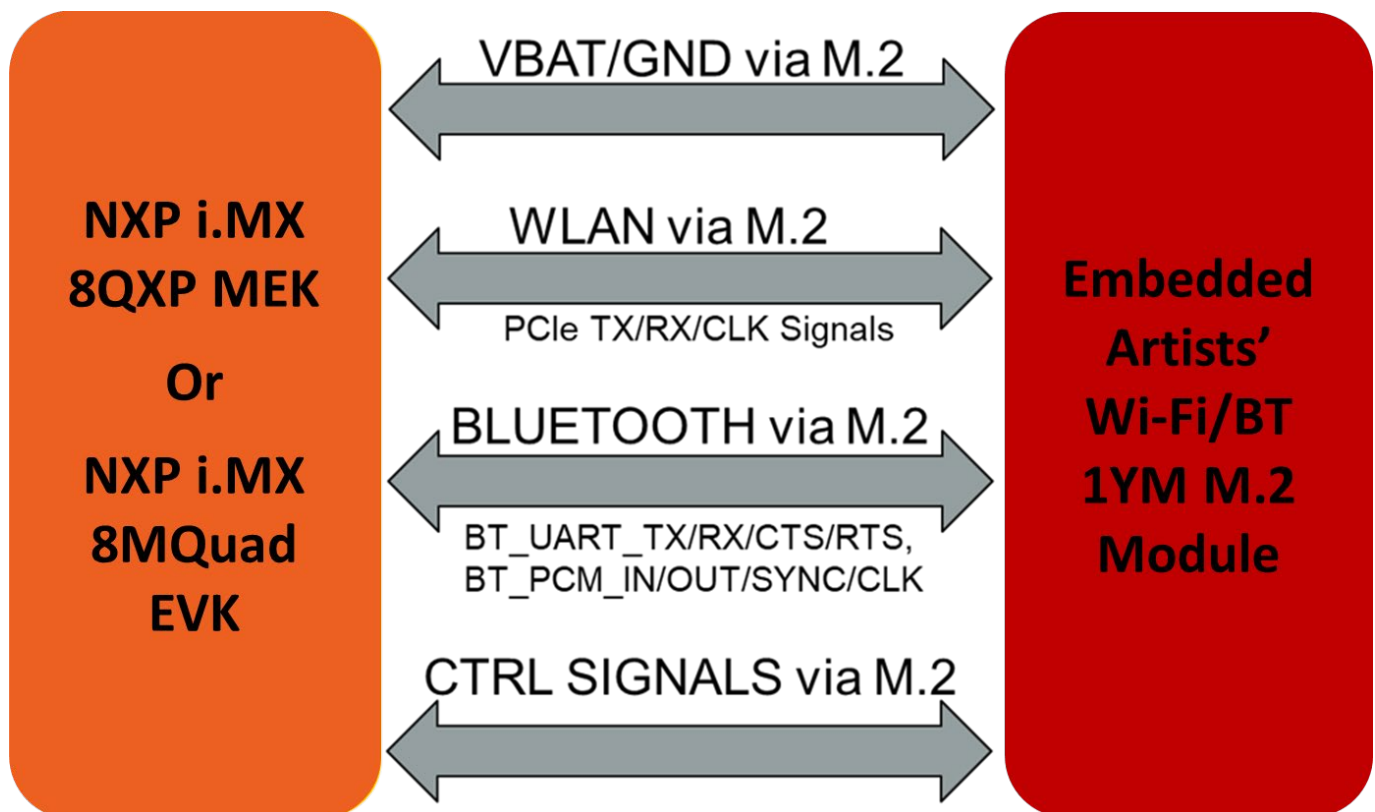
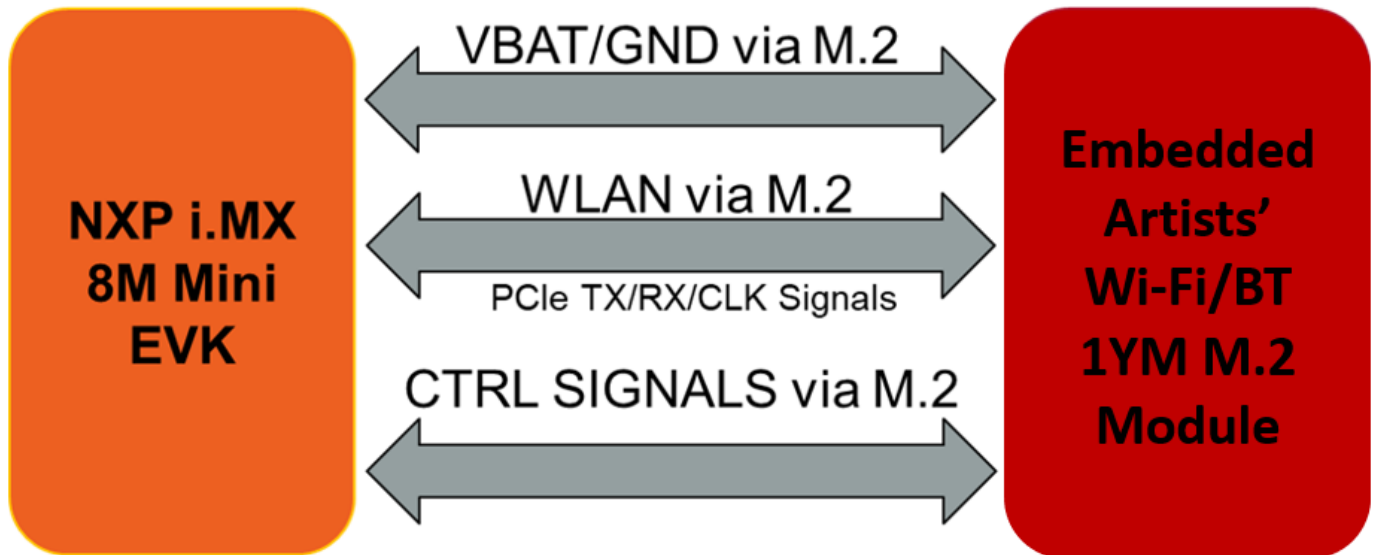


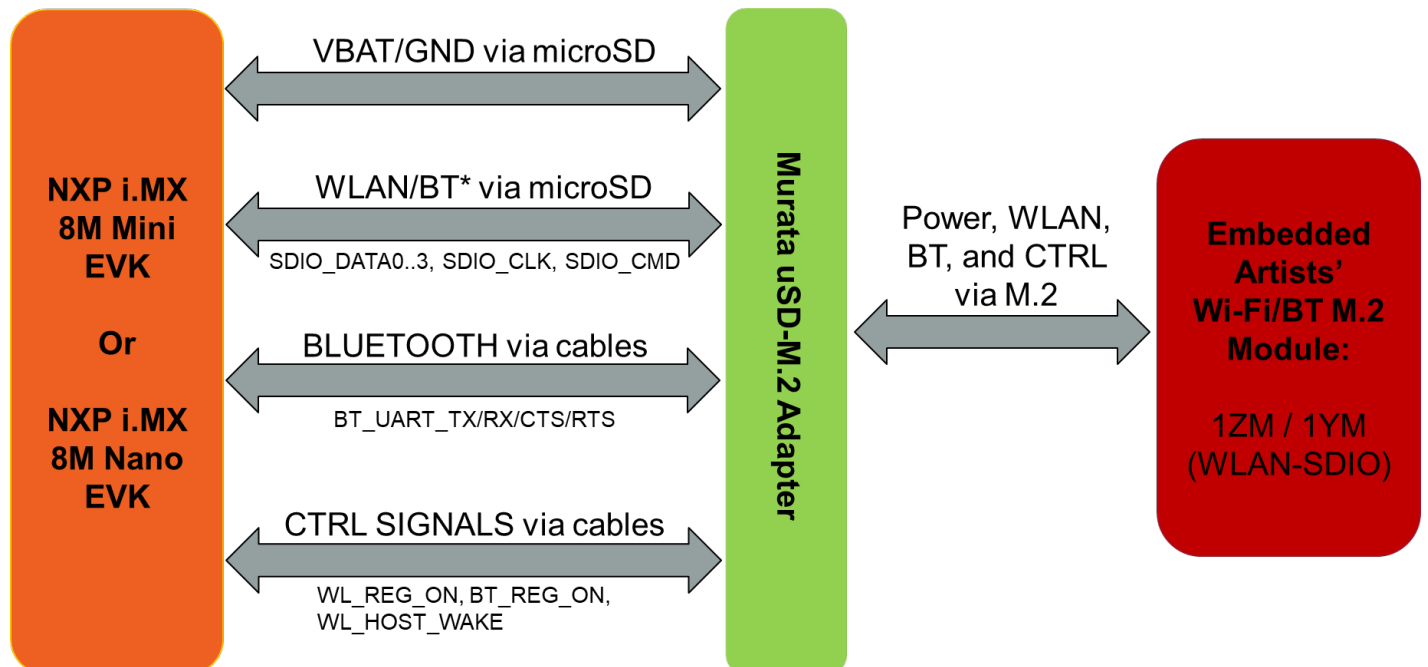
Figure 3 shows a simplified block diagram for the i.MX 8M Mini EVK Wi-Fi/BT interconnect (using M.2 connector). Currently Type 1YM module (WLAN Only – Bluetooth signals are not brought out) is supported via the WLAN-PCIe interface.

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



For WLAN-SDIO based module see **Figure 4**: Type 1ZM and 1YM modules are supported via the uSD-M.2 adapter on both the i.MX 8M Mini and i.MX 8M Nano EVK's. Note that both variants of these EVK's use onboard eMMC flash. The eMMC flash is necessary as the default uSD card slot is no longer available for booting the Linux image. As pictured, the Type 1YM M.2 EVB strapping (**Section 3.4.5**) needs to be configured for WLAN-SDIO/BT-SDIO operation (see "WLAN/BT* via microSD" in **Figure 4**). Murata's customized build script provides software build option for "1YM-SDIO*". For eventual NXP baseline release, it will support WLAN-PCIe/BT-UART and WLAN-SDIO/BT-UART. Of course, the default for 1YM M.2 EVB (part EAR00370 as shipped) is WLAN-PCIe/BT-UART.

Figure 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
1YM	AKA “1YM-PCIe”, indicates that M.2 EVB is strapped for WLAN-PCIe/BT-UART - refer to Section 3.4.5
1YM-SDIO	Type 1YM M.2 EVB strapped for WLAN-SDIO/BT-UART - refer to Section 3.4.5
1YM-SDIO*	Type 1YM M.2 EVB strapped for WLAN-SDIO/BT-SDIO - refer to Section 3.4.5
AP	Access Point
API	Application Programming Interface
BSP	Board Support Package
BT	Bluetooth
CTRL	Control
CTS	Clear to Send
DHCP	Dynamic Host Configuration Protocol
DTB	Device Tree Blob: Kernel reads in at boot time for configuration.
EA	Embedded Artists designs, manufactures and distributes current Wi-Fi/BT M.2 EVB’s (link here). EA also have enhanced i.MX developer kits which provide comprehensive support for Murata modules (link here).
eMMC	Embedded Multi-Media Controller: integrated flash memory and controller on single die.
EULA	End User License Agreement
EVB	Evaluation Board (Embedded Artists’ Wi-Fi/BT module)
EVK	Evaluation Kit (includes EVB + Adapter)
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
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
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 (NXP) Solution for i.MX Linux User Manual

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

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

This [Quick Start Guide](#) provides quick steps to get started with Murata Wi-Fi/BT NXP 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 1ZM/1YM 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.
1ZM M.2 Module Datasheet	Comprehensive details on 1ZM Wi-Fi/BT M.2 Module.
1YM M.2 Module Datasheet	Comprehensive details on 1YM Wi-Fi/BT M.2 Module.

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

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

- [i.MX_Yocto_Project_User's_Guide.pdf](#) / IMXLXOCTOUG
- [i.MX_Linux_User's_Guide.pdf](#) / IMXLUG
- [i.MX_Linux_Reference_Manual.pdf](#) / IMXLXRM
- [i.MX_Linux_Release_Notes.pdf](#) / IMXLXRN

Table 4: NXP Reference Documentation Listing

Kernel release	NXP documentation link	Yocto code name
5.4.47_2.2.0	Rev. L5.4.47 2.2.0 BSP	Zeus

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 1ZM (NXP 88W8987) supports only WLAN-SDIO interface, whereas Type 1YM (NXP 88W8997) supports both WLAN-PCIe and WLAN-SDIO interfaces. Also note that both **Type 1ZM's and Type 1YM's WLAN-SDIO VIO interface only supports 1.8V**¹. 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 Part #	VIO	Interface	EVB Picture
1ZM	NXP 88W8987	802.11b/g/n/ac BT/BLE 5.1	LBEE5QD1ZM	EAR00364	1.8V Only	WLAN-SDIO BT-UART	
1YM	NXP 88W8997	802.11a/b/g/n/ac (2x2 MIMO) BT/BLE 5.2	LBEE5XV1YM	EAR00370	1.8V (WLAN SDIO, PCIe) 3.3V (WLAN-PCIe Only)	WLAN-PCIe WLAN-SDIO BT-UART BT-SDIO	

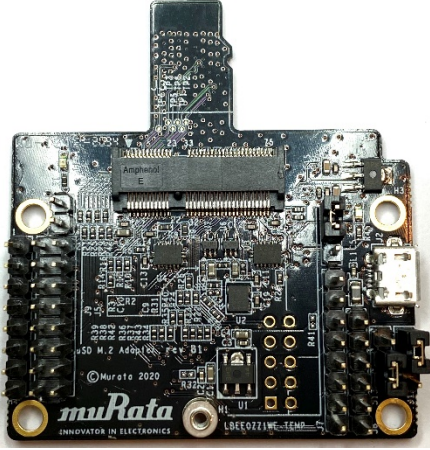

2.2 Murata's uSD-M.2 Adapter

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

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

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
	<p>uSD-M.2 Adapter (Revision B1)</p>
	<p>M.2 screw for attaching Wi-Fi/Bluetooth M.2 Module</p>
	<p>75mm 20-pos, 0.5mm pitch flat/flex cable</p>
	<p>13 pieces 200mm long male-to-female jumper cables (compatible with Arduino header)</p>
	<p>4 x 19mm stand-offs in nylon and associated M3 screws</p>
	<p>microSD to SD Card Adapter</p>

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

- **“NC”** means “No Connect”. This is due to one or both of the following reasons:
 - VIO incompatible: Wi-Fi/BT M.2 Module requires VIO voltage level that the NXP i.MX Hardware cannot provide.
 - Physical bus (i.e. SDIO, PCIe, UART) and/or WLAN/Bluetooth control line interconnect is not available.
- **“uSD-M.2”**: Murata’s uSD-M.2 Adapter provides interconnect to the Embedded Artists’ Wi-Fi/BT M.2 Module. NXP i.MX Host WLAN SDIO is configured for 1.8V VIO default. The Bluetooth UART, and WLAN/BT control signals from i.MX Host are configured at 3.3V VIO (hardware limitation due to fixed voltage rails). Rev B1 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. For this configuration, Type 1ZM M.2 EVB requires BT-UART and WLAN/BT control signal connection whereas Type 1YM M.2 EVB only requires WLAN/BT control signal connection (in current 1YM-SDIO* configuration).
- **“M.2”**: Only Embedded Artists’ Wi-Fi/BT M.2 EVB is required. The NXP i.MX EVK has 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. 1YM) can be used.
- **“M.2^W”**: Only Embedded Artists’ Wi-Fi/BT M.2 EVB is required. Then NXP i.MX EVK has a M.2 connector. Currently only WLAN-PCIe, and WLAN CTRL signals are brought out on the M.2 connectors for these specific NXP i.MX EVK’s. As such, there is no Bluetooth support – only WLAN.
- Again, the default hardware strapping option for EA’s Type 1YM M.2 EVB (EAR00370) is WLAN-PCIe/BT-UART. In **Table 7**, we refer to this default configuration as **“1YM-PCIe”**. It is otherwise referred to as 1YM. The non-default (WLAN-SDIO/BT-SDIO) strapping option is referred to as **“1YM-SDIO*”**. Eventually (when software is available) the 1YM (WLAN-SDIO/BT-UART) M.2 configuration will be referenced as **“1YM-SDIO”**.

Table 7: NXP i.MX/Murata Module Interconnect

NXP i.MX EVK Part Number	<u>1ZM</u>	<u>1YM-SDIO*</u>	<u>1YM-PCIe</u>
	NXP 88W8987	NXP 88W8997	NXP 88W8997
MCIMX8QXP-CPU	NC	NC	M.2
MCIMX8M-EVKB	NC	NC	M.2
8MMINILPD4-EVK	uSD-M.2 ⁺	uSD-M.2 ⁺	M.2 ^W
8MMINID4-EVK²	NC	NC	M.2 ^W
8MNANOD4-EVK	uSD-M.2 ⁺	uSD-M.2 ⁺	NC
MCIMX6UL-EVKB	uSD-M.2	uSD-M.2	NC
MCIMX6ULL-EVK	uSD-M.2	uSD-M.2	NC

uSD-M.2 = works with uSD-M.2 Adapter configured for 1.8V default
uSD-M.2⁺ = works with uSD-M.2 Adapter (Rev B1) with additional cabling
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
NC = No Connection options available

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

3.1 Murata’s Customized NXP Wireless Driver Release

The NXP i.MX Linux 5.4.47 for i.MX 6/8 integrates the NXP WLAN driver and has Bluetooth (BlueZ stack) support. Murata’s customized Yocto layer “*meta-murata-wireless*” **provides the following enhancements/customizations:**

- Comprehensive support for 1ZM/1YM on all possible i.MX targets - hardware interconnect is the only limiting factor: see **Table 9**.
- Customized (evaluation-only) solution for Type 1YM configured in WLAN-SDIO/BT-SDIO mode. The eventual NXP release will integrate WLAN-SDIO/BT-UART support for Type 1YM.
- Patches for DTB and Kernel for enabling Murata modules.
- Correct WPA-suplicant linking and configuration.
- 1.8V SDIO VIO configuration for WLAN interface on i.MX 6UL(L) EVK’s.

² 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.

3.2 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). 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 and hardware interconnect configuration (either uSD-M.2 Adapter & M.2 EVB, or just M.2 EVB). Note that the current Wi-Fi/BT M.2 EVB’s (1ZM/1YM) supporting WLAN-SDIO interface **only interface at 1.8V VIO**. The i.MX 8M Mini/Nano EVK’s with “*uSD-M.2+*” configuration require additional part(s) as documented in **Section 6.3**.

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

NXP i.MX EVK Part #	NXP i.MX EVK	MACHINE=target	Kernel 5.4.47
MCIMX8QXP-CPU	i.MX 8QuadXPlus MEK	imx8qxpmek	Y
MCIMX8M-EVKB	i.MX 8MQuad EVK	imx8mqevk	Y
8MMINILPD4-EVK	i.MX 8M Mini EVK	imx8mmevk	Y
8MMINID4-EVK	i.MX 8M Mini EVK	imx8mmdr4evk	Y
8MNANOD4-EVK	i.MX 8M Nano EVK	imx8mnddr4evk	Y
MCIMX6UL-EVKB	i.MX 6UL EVK	imx6ulevk	Y
MCIMX6ULL-EVK	i.MX 6ULL EVK	imx6ull14x14evk	Y

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

Target (MACHINE)	Hardware Config	i.MX DTB File	Interrupt Config
imx8qxpmek	M.2	fsl-imx8qxp-mek.dtb	N/A
imx8mqevk	M.2	fsl-imx8mq-evk-pcie1-m2.dtb	N/A
imx8mmevk	M.2 ^w	imx8mm-evk.dtb	N/A
imx8mmevk	uSD-M.2 ⁺	imx8mm-evk-usd-m2.dtb	SDIO
imx8mmdr4evk	M.2 ^w	imx8mm-ddr4-evk.dtb	N/A
imx8mnddr4evk	uSD-M.2 ⁺	imx8mn-evk-usd-m2.dtb	SDIO
imx6ulevk	uSD-M.2	imx6ul-14x14-evk-btwifi-m2.dtb	SDIO
imx6ull14x14evk	uSD-M.2	imx6ull-14x14-evk-btwifi-m2.dtb	SDIO

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
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

3.3 Murata Customized i.MX Yocto Image Build

Murata’s solution to easily enable NXP-based Wi-Fi/Bluetooth functionality requires a complete Linux image build (bootloader, kernel, DTB files, filesystem). To understand this requirement, we need to understand specifics about the NXP i.MX Linux image. The NXP i.MX image contains third party IP which is sub-licensed via a click-through EULA (when either downloading an 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 still must configure a Linux build environment and follow specific steps to arrive at the desired image for a given i.MX target and Murata WLAN/BT configuration. Murata has greatly simplified the build requirement by providing scripts for Ubuntu host setup and customized Yocto build – scripts easily downloadable from Murata’s GitHub. Steps for downloading, configuring, and invoking these scripts are detailed here.

3.3.1 Install Ubuntu

First step is to install Ubuntu 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](#). For more information on the Ubuntu download, please refer to this link:

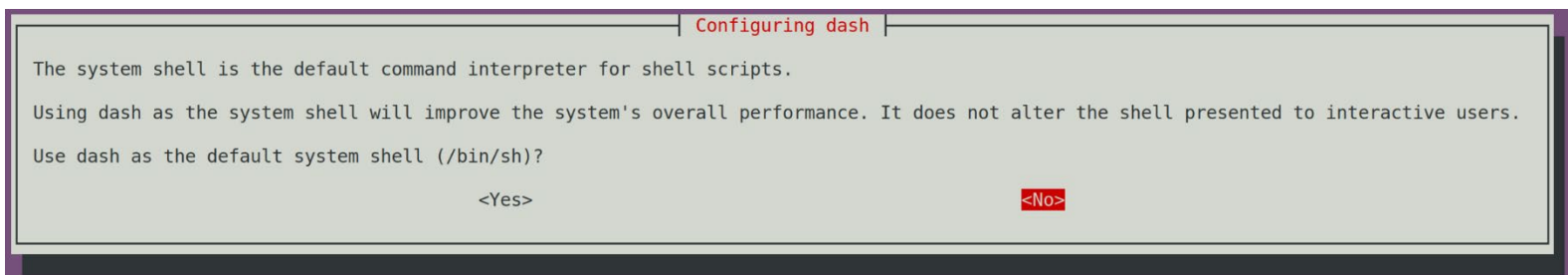
<https://www.ubuntu.com/download/desktop>. The Ubuntu installation manual is provided [here](#). By default, Ubuntu sets the environment to use dash. It is mandatory that, **User sets the default system shell to “No” when configuring dash**. Follow the steps mentioned below **for reconfiguring dash**:

- 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



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 release is supported**:
 - 5.4.47_2.2.0 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. Murata provides Wi-Fi support for both NXP and Cypress chipset-based modules.
- 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
- 6) Optional Step for NXP “1YM-SDIO”
- 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.wic
```

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 is detailed in **Section 4.2**.

3.4 Additional Hardware/Software Considerations

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

As shown in **Table 7** and **Table 9**, the only NXP i.MX 6 Platforms supported include i.MX 6UL(L) EVK's. This is due to a WLAN-SDIO VIO limitation on both Type 1ZM and 1YM. Both modules only support a WLAN-SDIO VIO of 1.8V. The NXP i.MX 6UL(L) EVK's are the only i.MX 6 Platforms (with appropriate interconnect) which can support this required 1.8V signaling over WLAN-SDIO bus. As such, the Murata uSD-M.2 Adapter **should never be** configured in “**3.3V VIO Override Mode**” when connected to Type 1ZM or 1YM M.2 EVB's. Note that the BT-UART and some WLAN/BT control signals are level-shifted on the uSD-M.2 Adapter (default configuration) from 3.3V VIO (Host) to 1.8V VIO (M.2). Refer to the [Hardware User Manual](#) for more details.

3.4.2 UHS SDIO 3.0 operation on i.MX 6UL(L) EVK's 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 6UL(L) EVK's, **the maximum SDIO clock frequency is limited to 50MHz**. 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 6UL(L) EVK's require the additional microSD-to-SD Adapter. For UHS mode (and better overall hardware/software) support, **Murata strongly recommends** the [Embedded Artists' i.MX Developer Kits](#). See [Embedded Artists' Solution](#) section for more details.

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

As shown in **Table 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.

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

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

- **Set DTB (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. Refer to **Table 9: i.MX6/8 Targets supported by Murata** for more details regarding correct DTB file selection.
- Invoke **“switch_module.sh <1zm | 1ym-sdio | 1ym-pcie>”** 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.

Providing more details on “switch_module” script, Murata has included this script file in its images for customers to easily set up and switch between EVBs. **The usage is as shown below:**

```
switch_module.sh <module>
```

Where <module> can be:

- **1zm:** Sets up the EVK for 1ZM module (WLAN-SDIO is only interface supported)
- **1ym-pcie:** Sets up the EVK for 1YM (WLAN-PCIe)
- **1ym-sdio:** Sets up the EVK for 1YM (WLAN-SDIO)

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.

Once the user has run the **“switch_module.sh”** script, it is necessary to enter **“reboot”** command:

```
reboot
```

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

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

- WLAN-PCIe/BT-UART (**1YM** or **1YM-PCIe**): default configuration and supported in software release. For existing interconnect, this mode is used when plugging the M.2 EVB directly into NXP M.2 connector.
- WLAN-SDIO/BT-UART (**1YM-SDIO**): currently not supported in software release, this configuration **will be** a supported option in NXP baseline BSP release. See **Figure 8** for resistor strapping option – two 50K Ohm resistors need to be added (to right of default one).
- WLAN-SDIO/BT-SDIO (**1YM-SDIO***): currently supported in software release, this configuration is for customer evaluation purposes only. It is not intended for final shipping product. The customer must have special software release access on NXP website to build the Linux image with this support option. The Murata build script provides necessary details on NXP software package name. Also refer to [Linux User Manual](#) for more details. See **Figure 7** for resistor strapping option – one 50K Ohm resistor needs to be added (to right of default one).

Regarding the actual configuration of the Embedded Artists Type 1YM M.2 Module (EVB), the customer needs to perform some minor rework to modify the resistor strapping options – if not going with WLAN-PCIe/BT-UART default. The necessary rework (addition of 50K Ohm 0201 resistors) is easily illustrated in **Figure 6** which shows both relevant schematic capture and configuration-strapping table. **The “CONFIG HOST” resistors are (in order left to right as pictured):**

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

Figure 6: Type 1YM M.2 EVB Configuration Strapping Options

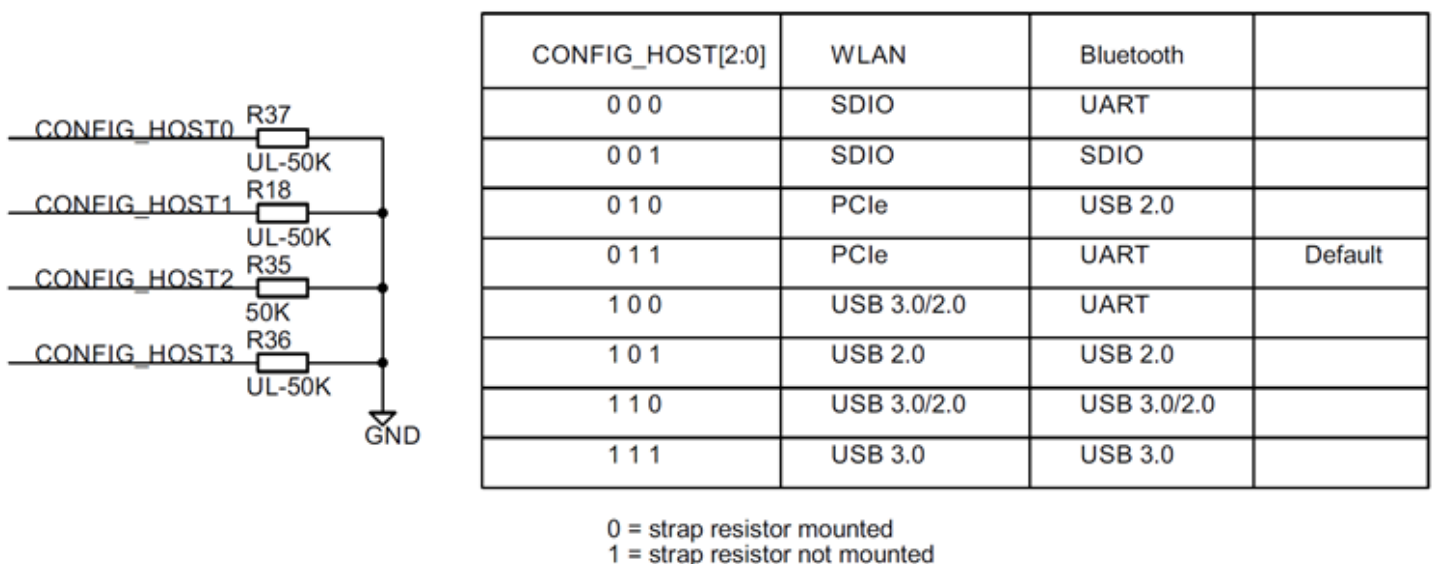


Figure 7: Type 1YM M.2 Configured for WLAN-SDIO/BT-SDIO (1YM-SDIO*)

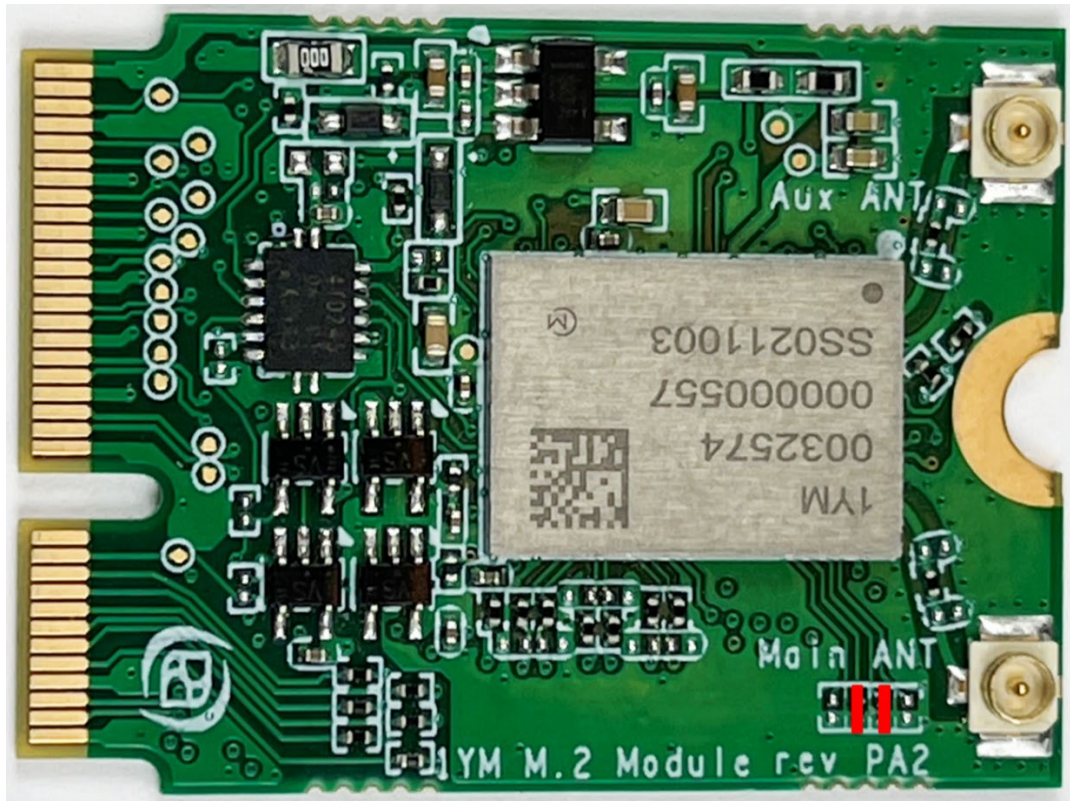
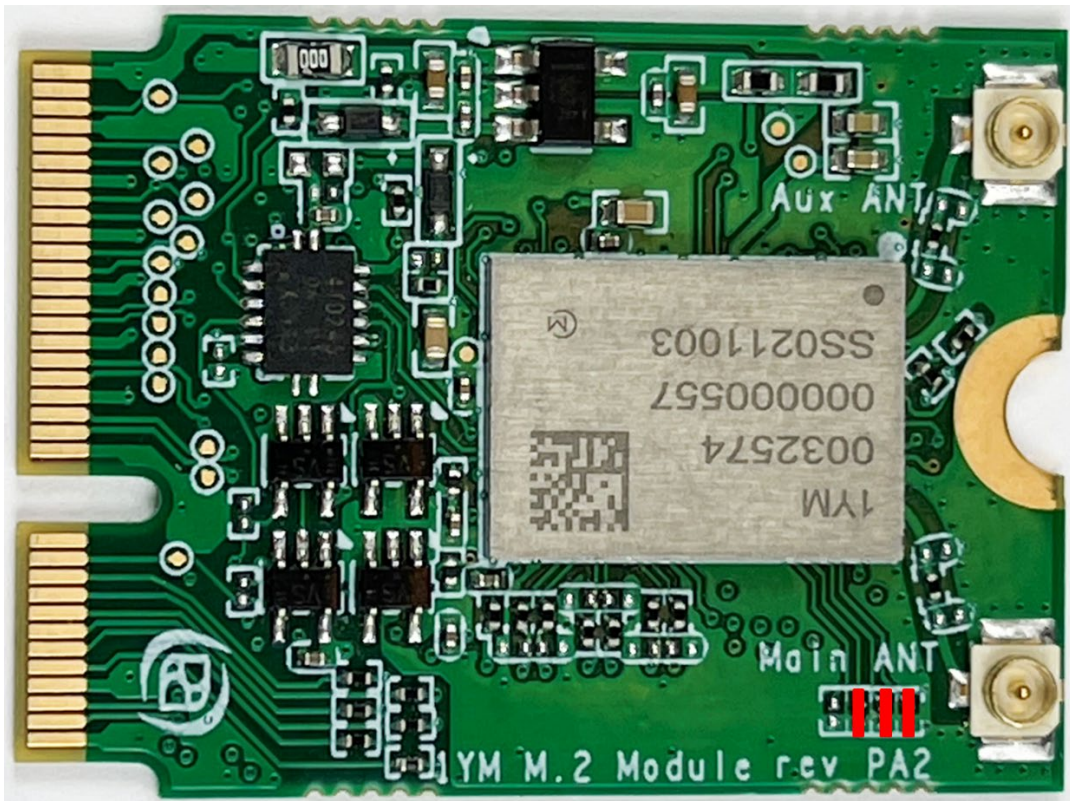


Figure 8: Type 1YM M.2 Configured for WLAN-SDIO/BT-UART (1YM-SDIO)



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
imx8mmevk	M.2 ^w	imx8mm-evk.dtb	uSD	N/A
imx8mmevk	uSD-M.2 ⁺	imx8mm-evk-usd-m2.dtb	eMMC	SDIO
imx8mnddr4evk	uSD-M.2 ⁺	imx8mn-evk-usd-m2.dtb	eMMC	SDIO

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

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 9**. 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 9: USB to SD Card Reader/Writer Adapter



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

```
dmesg
```

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

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

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

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

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

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

⇒ **SD Card is now flashed with customized Murata wireless image.**

4.1.2 Windows PC Steps to Flash SD Card

In case you need to later flash the same (micro) SD card image using a Windows PC, the following steps have been included. Windows utilities such as “Win32 Disk Imager” 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” program³, click on the folder icon/button and navigate to the location of the desired “*.wic” file (for Yocto release Zeus and later).
- Select the “Device” button and select the drive letter corresponding to the (micro) SD card: formatting SD card may be necessary beforehand with Windows low-level utilities⁴.
- 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.

³ “Win32 Disk Imager” is an open source tool that can be downloaded from websites such as “sourceforge.net”.

⁴ 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.

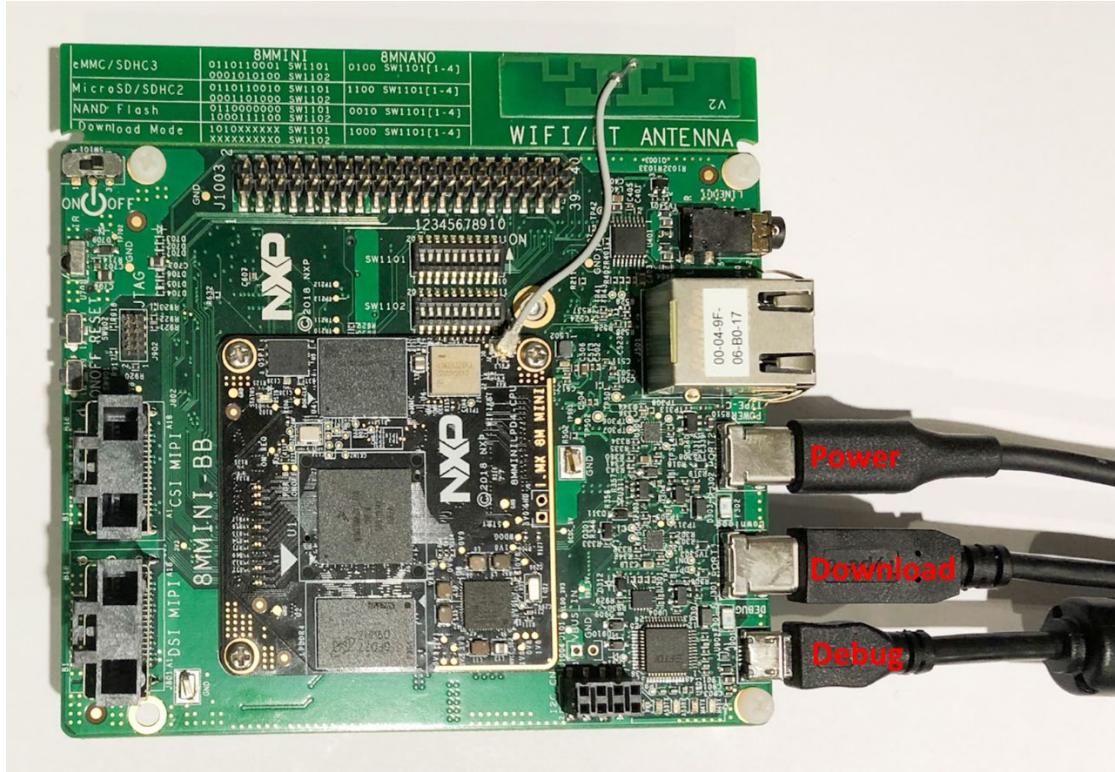
Table 11: Files to flash i.MX 8M Mini & 8M Nano EVK's

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

Figure 10: 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 11**.

SW1101	1	0	1	0	X	X	X	X	X
SW1102	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 12**.

SW1101	1	0	0	0
---------------	---	---	---	---

Where: 1 – ON, 0 – OFF

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

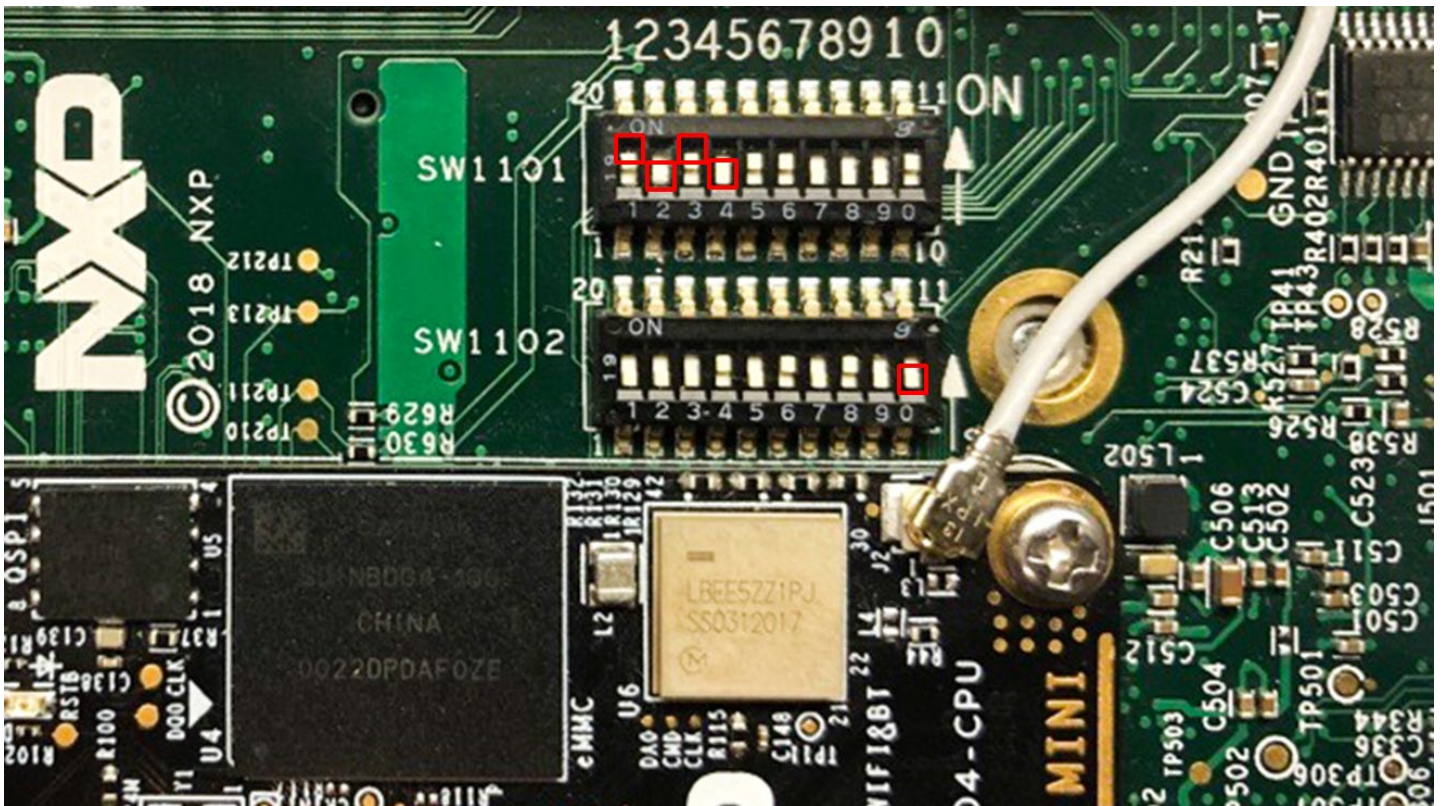
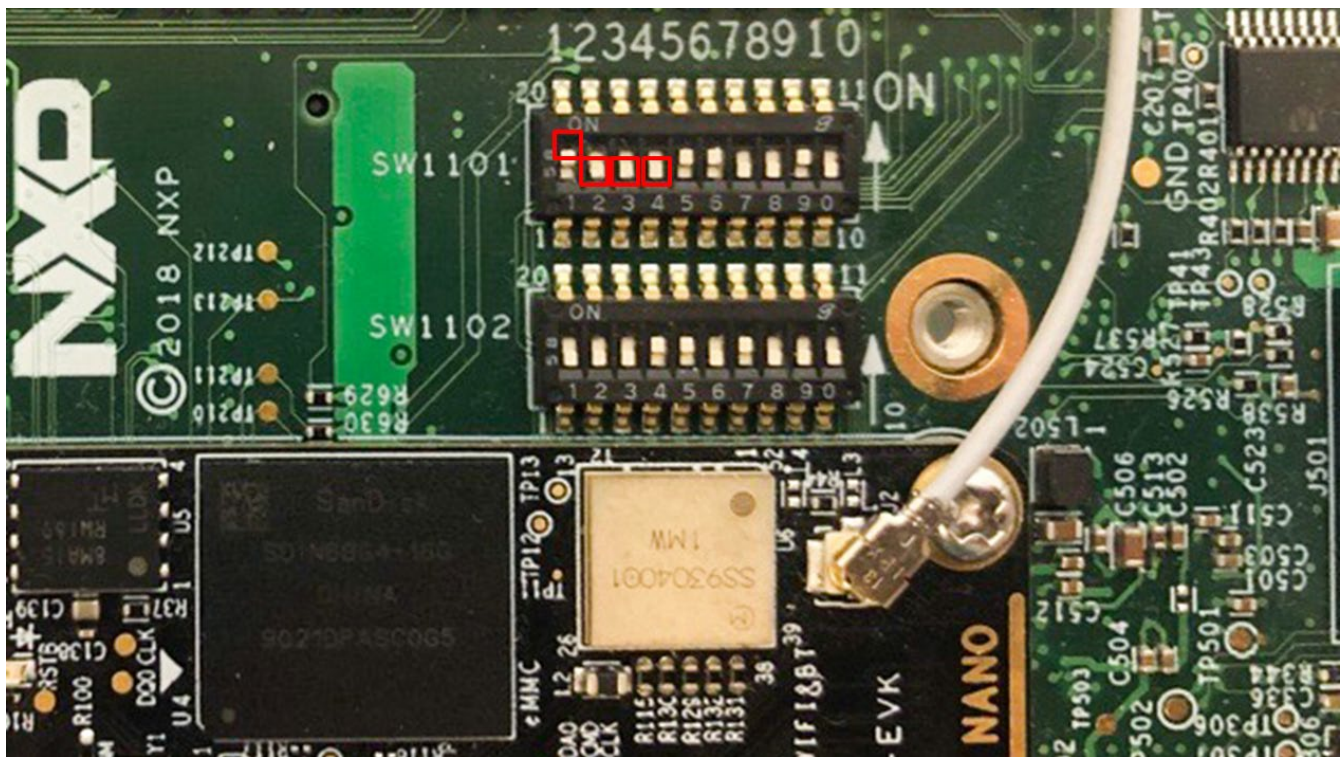


Figure 12: i.MX 8M Nano EVK DIP Switches configured for Download



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 13**.

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 14**.

SW1101	0	1	0	0
---------------	---	---	---	---

Where: 1 – ON, 0 – OFF

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

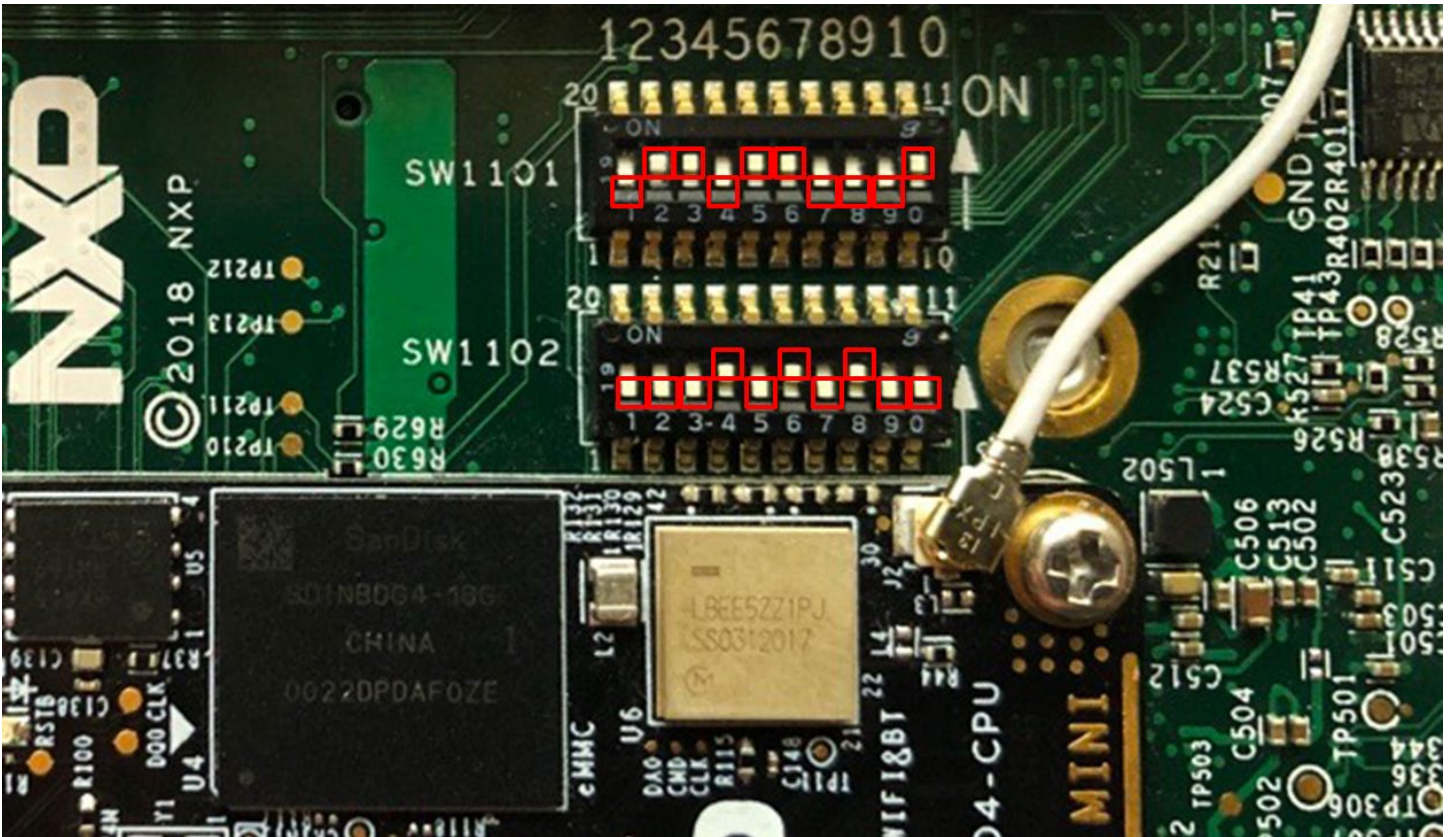
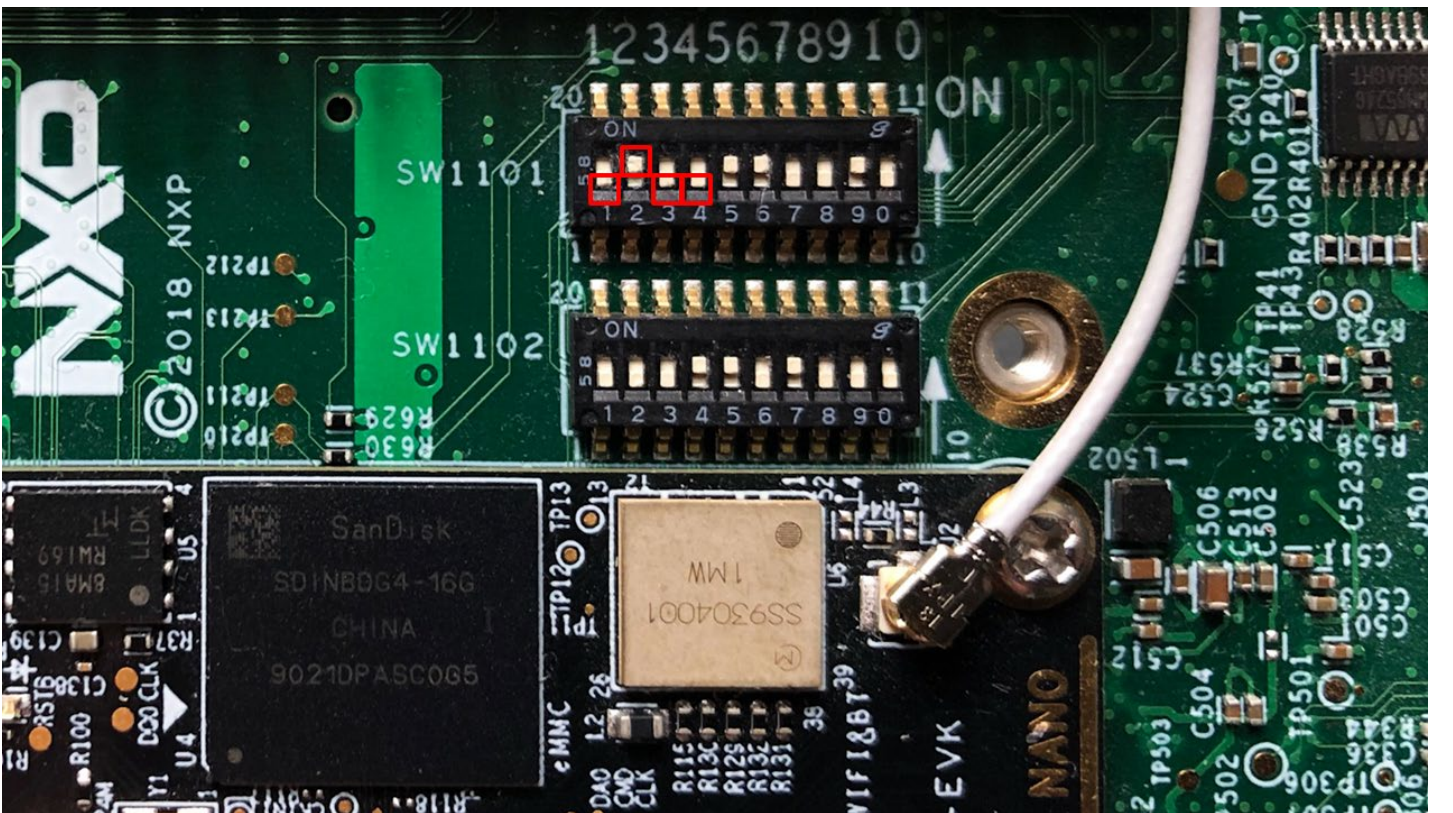


Figure 14: i.MX 8M Nano EVK DIP Switches configured for eMMC Boot



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

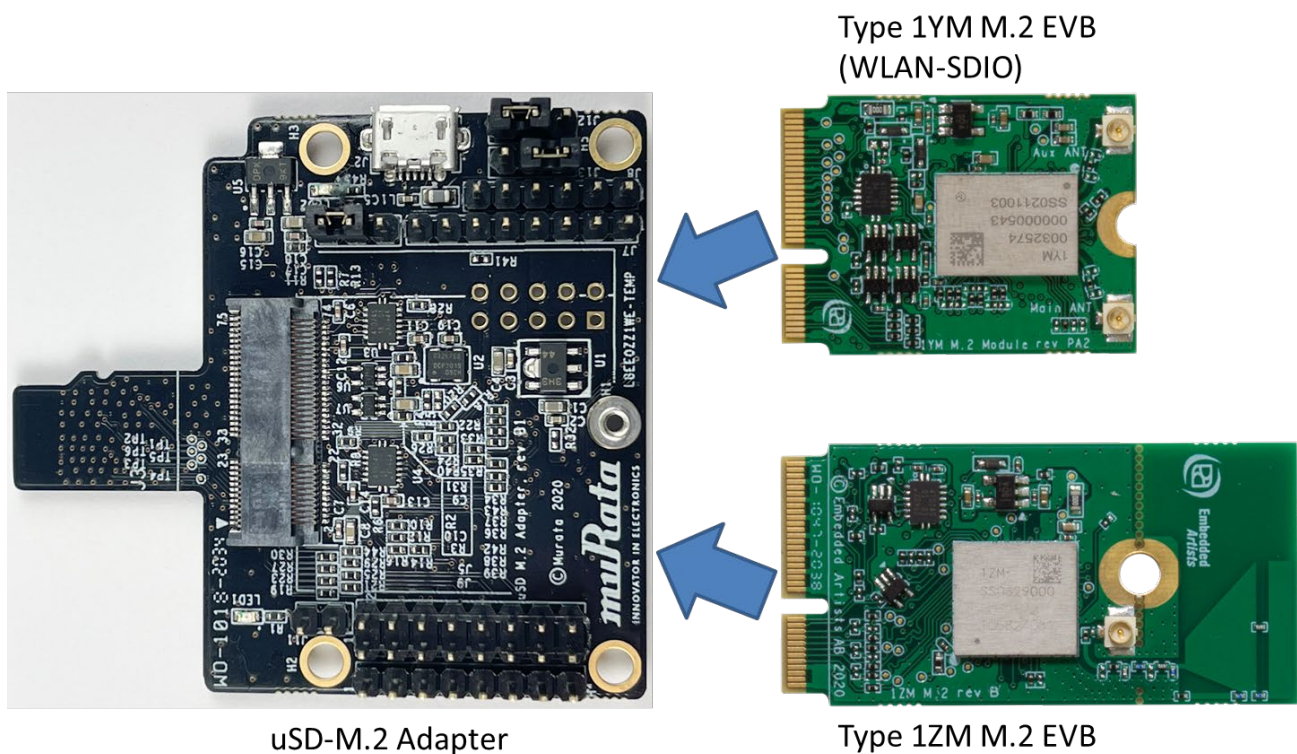
Embedded Artists' Wi-Fi/BT M.2 EVB's based on SDIO, listed in **Table 5** (currently 1ZM and 1YM-SDIO*) can be connected to the i.MX 6UL(L) EVK's through Murata's uSD-M.2 Adapter as shown in **Figure 15**. The following sub-sections details steps for bringing up Embedded Artists' Wi-Fi/BT EVB's on these EVK's.

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

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 1.8V VIO (M.2). The Rev B1 Adapter also level shifts some WLAN/BT control signals. The legacy Rev A Adapter does not have level shifting – as such the BT-UART signaling is mixed between host (3.3V) and target (1.8V). Customers **are recommended to use the latest Rev B1 Adapter** with correct voltage signaling on BT-UART.

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

Figure 15: uSD-M.2 Adapter with Type 1ZM and 1YM-SDIO* M.2 EVB options



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

- [1] Ensure no power is applied to i.MX 6UL(L) EVK. Connect J1101 micro-USB port to PC and start terminal emulator: “Minicom” on Linux or “Tera Term” on Windows. Set port to 115200-N-8-1.
- [2] On uSD-M.2 Adapter, set Jumper J1 to position 2-3 (VBAT from microSD connector). Make sure Jumper J11 is removed (BT core enabled). Note Blue LED2 **is not illuminated** for 1.8V VIO.
 - 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 **1ZM or 1YM** (reworked for WLAN-SDIO operation per **Section 3.4.5**) **Wi-Fi/BT M.2 EVB** to the M.2 connector on the uSD-M2 Adapter per **Section 8.1**. Connect the uSD-SD Card adapter and **tape** the uSD Adapter-SD Card per **Section 8.3**.
- [4] Connect ribbon cable at both ends before inserting Murata EVK into SD1 slot. Note the orientation as shown in **Figure 16**. **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. You can check available dtb files using command “**fatls mmc 1**”. With DTB set, save the u-boot configuration and boot platform:

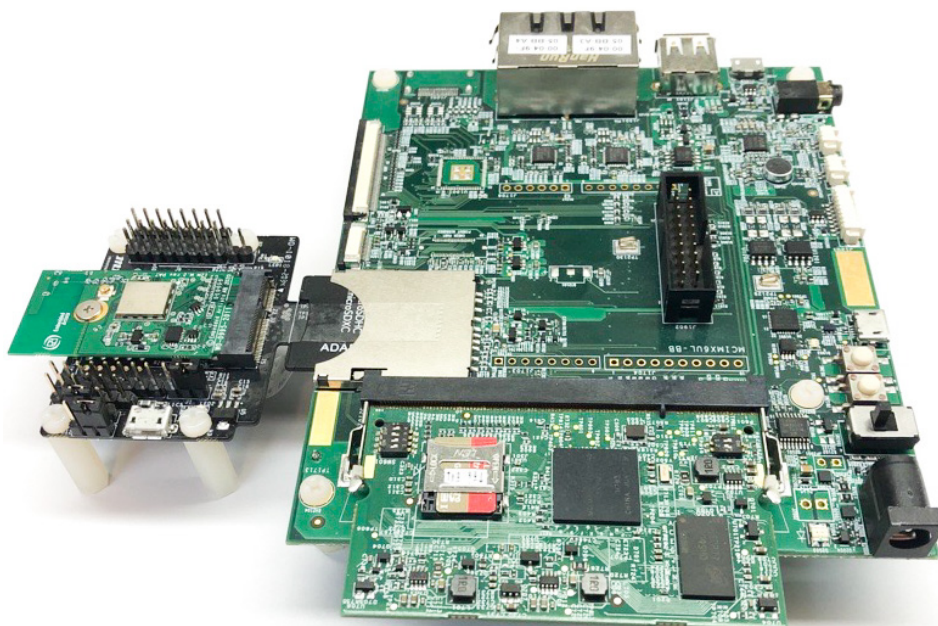
```
setenv fdt_file imx6ul-14x14-evk-btwifi-m2.dtb (OR imx6ull-14x14-evk-btwifi-m2.dtb)
saveenv
boot ← causes platform to boot kernel
```

- [6] After the kernel boots, invoke “**switch_module.sh 1zm**” or “**switch_module.sh 1ym-sdio**” for NXP-based solutions and reboot:

```
switch_module.sh 1zm (OR 1ym-sdio)
reboot
```

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

Figure 16: i.MX 6ULL EVK with uSD-M.2 Adapter and Type 1ZM M.2 EVB



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

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

The NXP i.MX 8MQuad EVK (see **Figure 17**) provides a secondary Wi-Fi/BT solution on the underside via a M.2 connector. The uSD-M.2 adapter provides WLAN PCIe, BT UART, control signals, and optionally BT PCM. Currently the only supported M.2 EVB is Type 1YM (WLAN-PCIe).

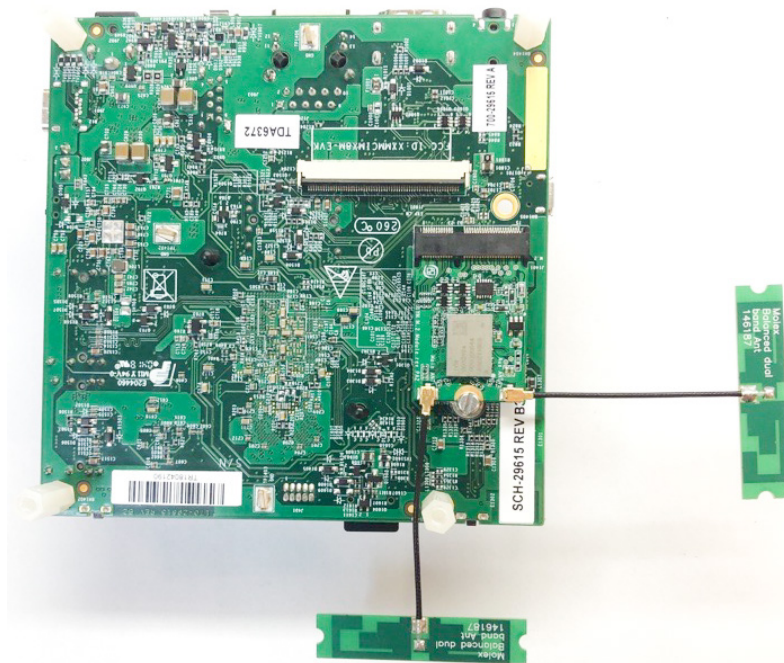
- [1] Connect J1701 micro-USB port to PC and start terminal emulator: “Minicom” on Linux or “Tera Term” on Windows. Set port to 115200-N-8-1.
- [2] Connect Embedded Artists the **1YM M.2 EVB** to M.2 connector as shown in **Figure 17**. Attach two dual-band (2.4/5GHz) antennas with U.FL connector as 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**. You can check the available dtb files using the command “**fatls mmc 1**”. Now save the u-boot configuration and boot the platform:

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

- [4] After kernel boots, invoke “**switch_module.sh 1ym-pcie**” for NXP-based solution and reboot:
switch_module.sh 1ym-pcie
reboot

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

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



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

Both NXP i.MX 8M Mini EVKs (8MMINILPD4-EVK and 8MMINID4-EVK) have a WLAN-PCIe M.2 connector on the baseboard – **with no connection for Bluetooth-UART**. See **Figure 18** for upside-down EVK view showing M.2 connector (currently the only supported M.2 EVB is Type 1YM). The steps below detail how to bring up the Wi-Fi/BT M.2 EVB.

- [1] Connect J901 micro-USB port to PC and start terminal emulator: “Minicom” on Linux or “Tera Term” on Windows. Set port to 115200-N-8-1.
- [2] Connect Embedded Artists Type 1YM M.2 EVB to M.2 connector as shown in **Figure 18**. 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 the platform to boot per **Section 4.2**. Power on the platform and interrupt at u-boot. Set DTB configuration with “*fdt_file*” parameter for correct platform per **Table 9**. Check the available dtb files by invoking “*fatls mmc 1*”. Now save u-boot configuration and boot the platform:

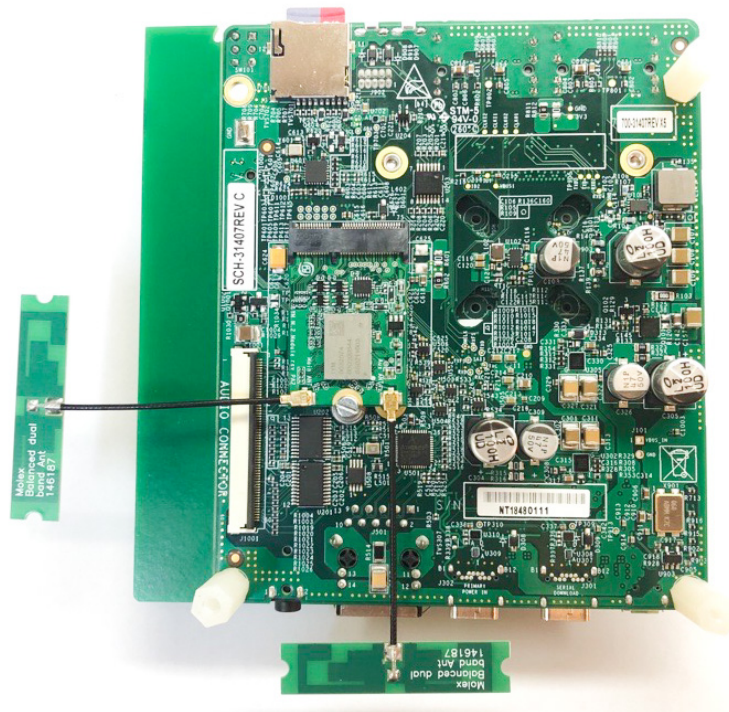
```
setenv fdt_file imx8mm-evk.dtb
saveenv
boot ← causes platform to boot kernel
```

- [4] After the kernel boots, invoke “*switch_module.sh 1ym-pcie*” for NXP-based solution and reboot:

```
switch_module.sh 1ym-pcie
reboot
```

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

Figure 18: i.MX 8M Mini with Type 1YM-PCIe (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 20) and WLAN (Figure 19). **The WLAN configuration is quite simple:** just insert inverted uSD-M.2 Adapter with Wi-Fi/M.2 EVB attached into microSD slot. The WLAN/BT configuration requires additional jumper cables for Bluetooth-UART and WLAN/BT control signals.

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

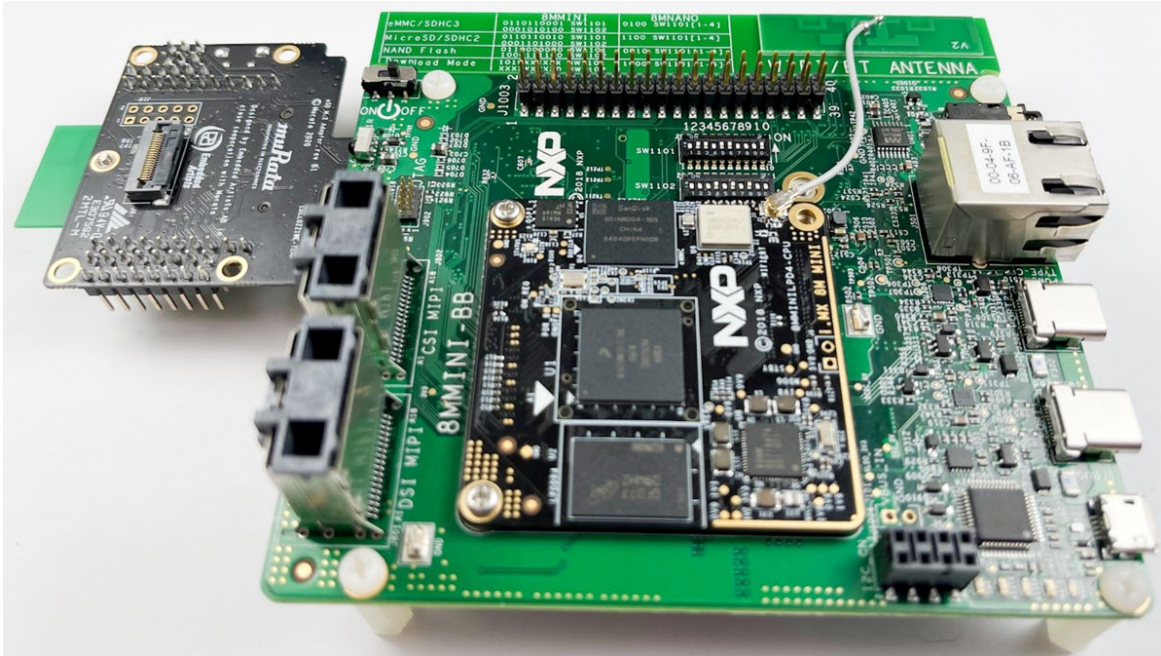
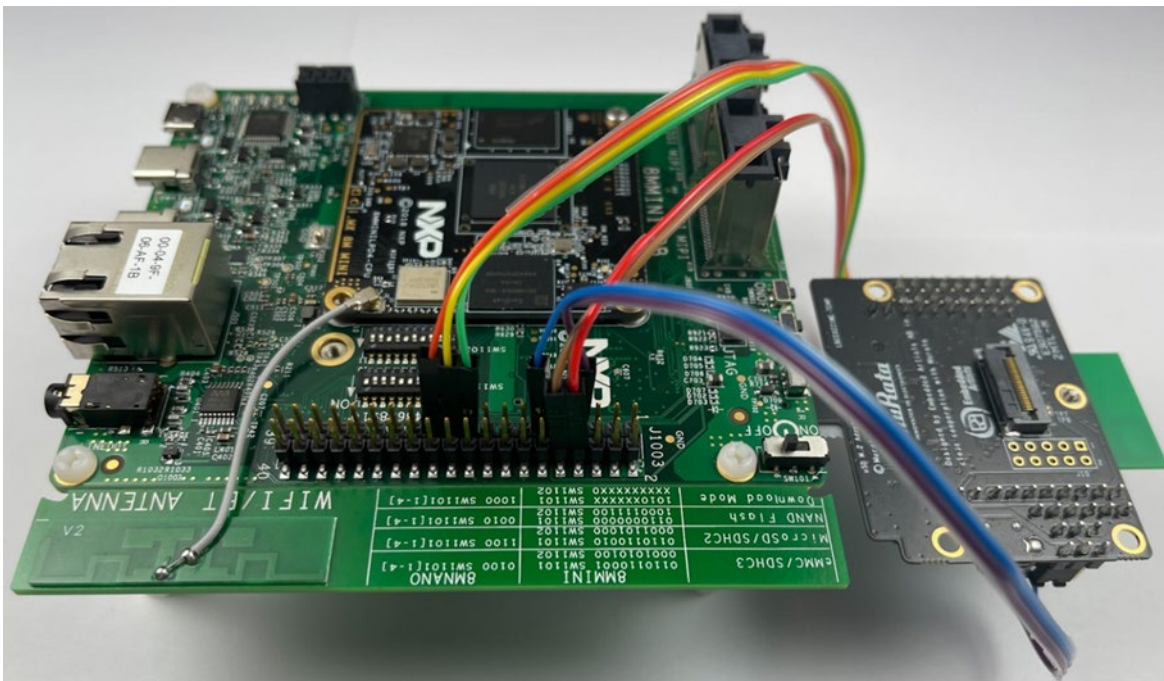


Figure 20: i.MX 8M Mini/Nano EVK with uSD-M.2 Adapter (WLAN/Bluetooth)



The only i.MX 8M Mini/Nano 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 (optional)) 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 **1ZM or 1YM** (reworked for WLAN-SDIO operation per **Section 3.4.5) Wi-Fi/BT M.2 EVB** to the M.2 connector on the uSD-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 (only six are required, skipping the WL_HOST_WAKE line) jumper wires from the uSD-M.2 adapter to the i.MX 8M Mini/Nano EVK per **Table 12**. Refer to **Figure 21**, **Figure 22**, **Figure 23** and **Figure 24** for additional details: colored wires are shown in these figures so users may more easily follow along. **Note there is a clearance issue** when attaching "normal" jumper wires. **Murata recommends two different approaches:**

- a. Use low-profile jumper wires (like **Digi-Key part number 1988-1178-ND**) which can be bent at right-angles – see **Figure 25**, and **Figure 26**. The connectors on uSD-M.2 Adapter need to be bent at 45° angle so that there is no interference with default NXP i.MX standoffs.

OR:

- 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 27**, the additional standoffs (which screw into existing NXP i.MX EVK standoffs) add necessary height to platform so there is no interference with jumper wire connectors.

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

- [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 for correct platform per **Table 9**. You can check available dtb files using the command “**fatls mmc 2**”. After setting DTB, save the u-boot configuration and boot the platform:

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

[7] After the kernel boots, invoke “**switch_module.sh 1zm**” or “**switch_module.sh 1ym-sdio**” for NXP-based solution and reboot:

```
switch_module.sh 1zm (OR 1ym-sdio)
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	Notes
BT_UART_TX	J9 / Pin 1	10	
BT_UART_RX	J9 / Pin 2	8	
WL_REG_ON	J9 / Pin 3	19	
BT_REG_ON	J9 / Pin 4	21	
WL_HOST_WAKE	J9 / Pin 5	23	Optional connection.
BT_UART_RTS	J8 / Pin 3	11	
BT_UART_CTS	J8 / Pin 4	7	

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

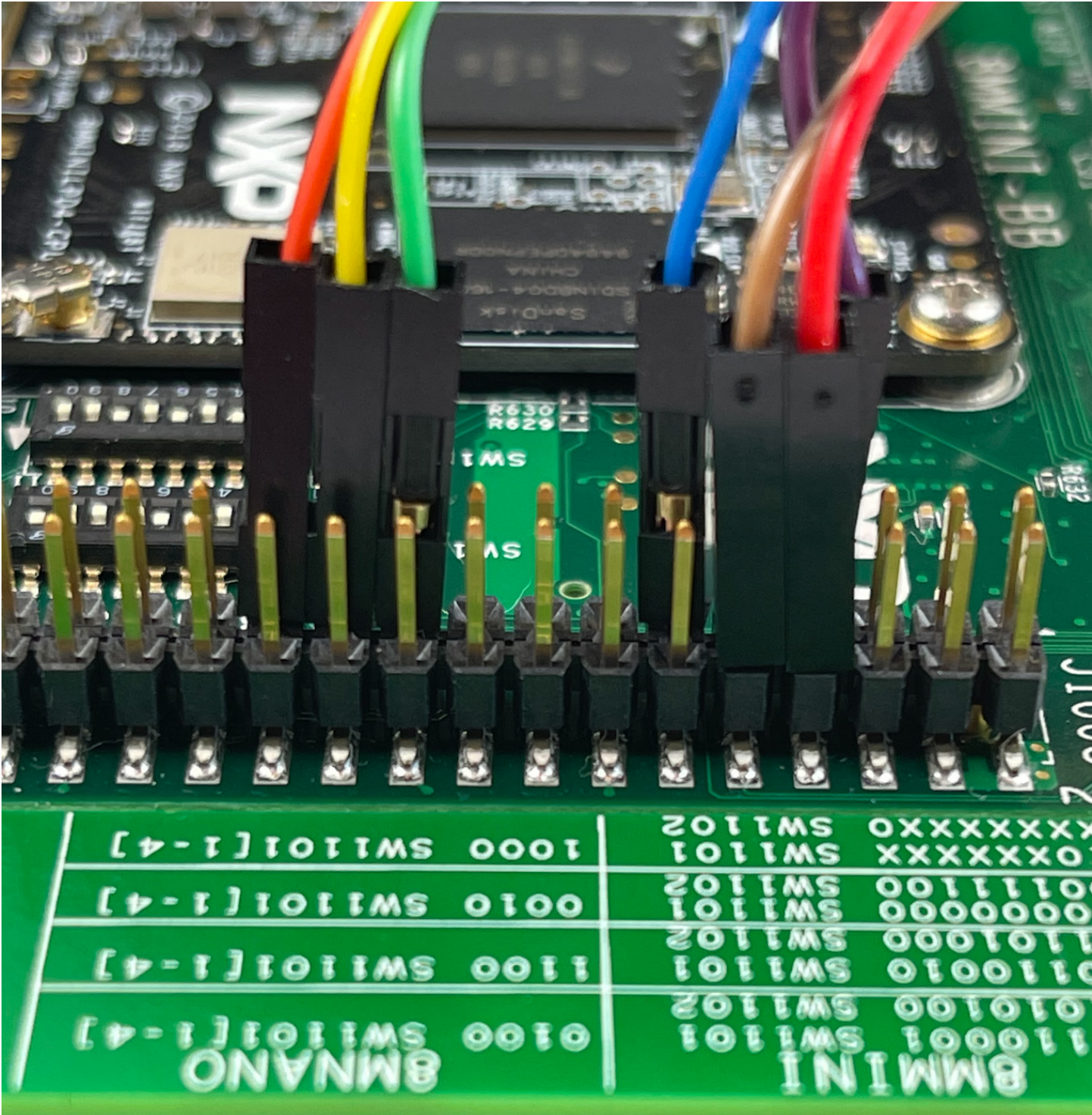


Figure 22: Cable connections on i.MX 8M Mini EVK (Odd Number Connector View)

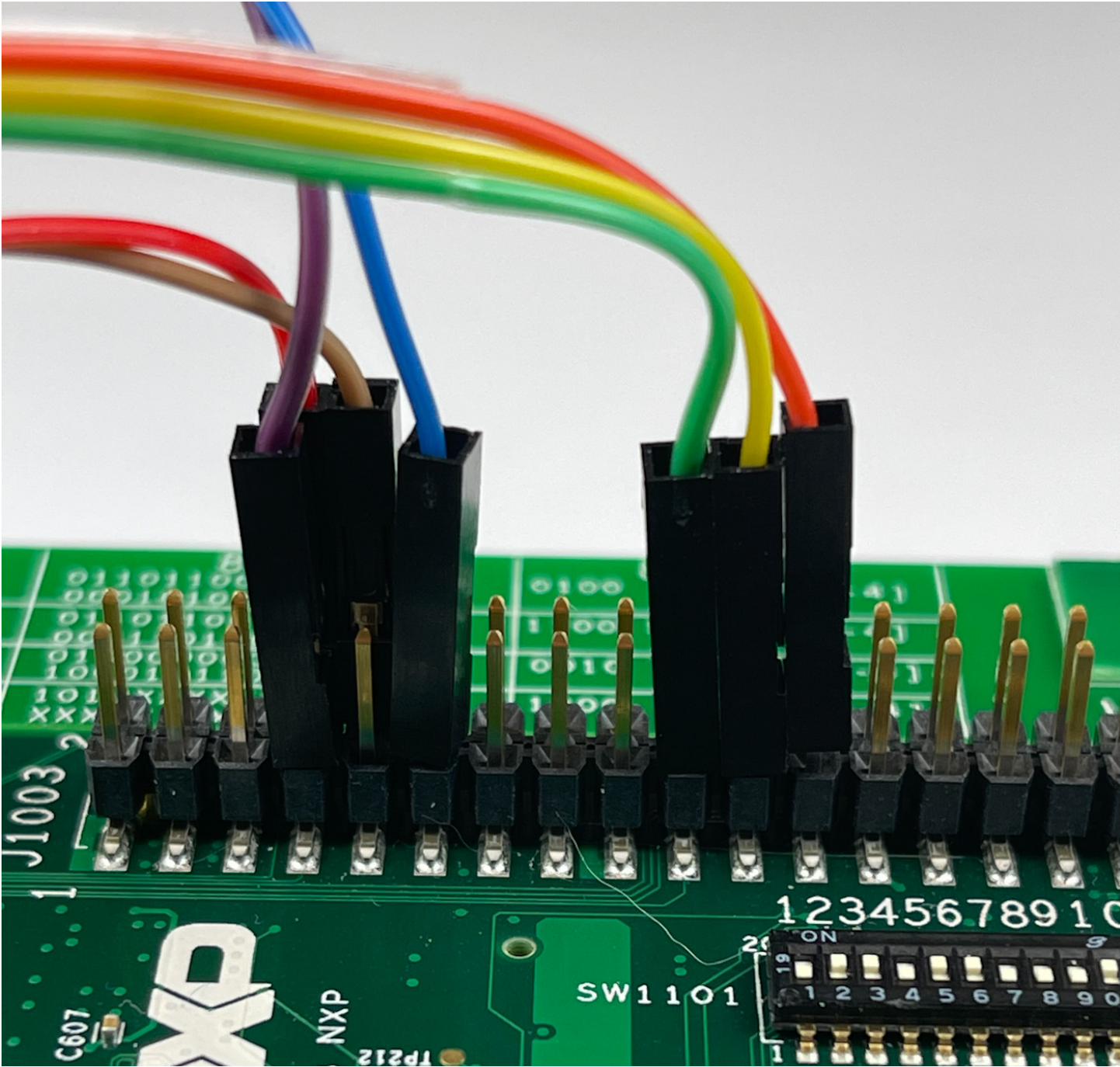


Figure 23: Cable connections on uSD-M.2 Adapter (J9 Header)

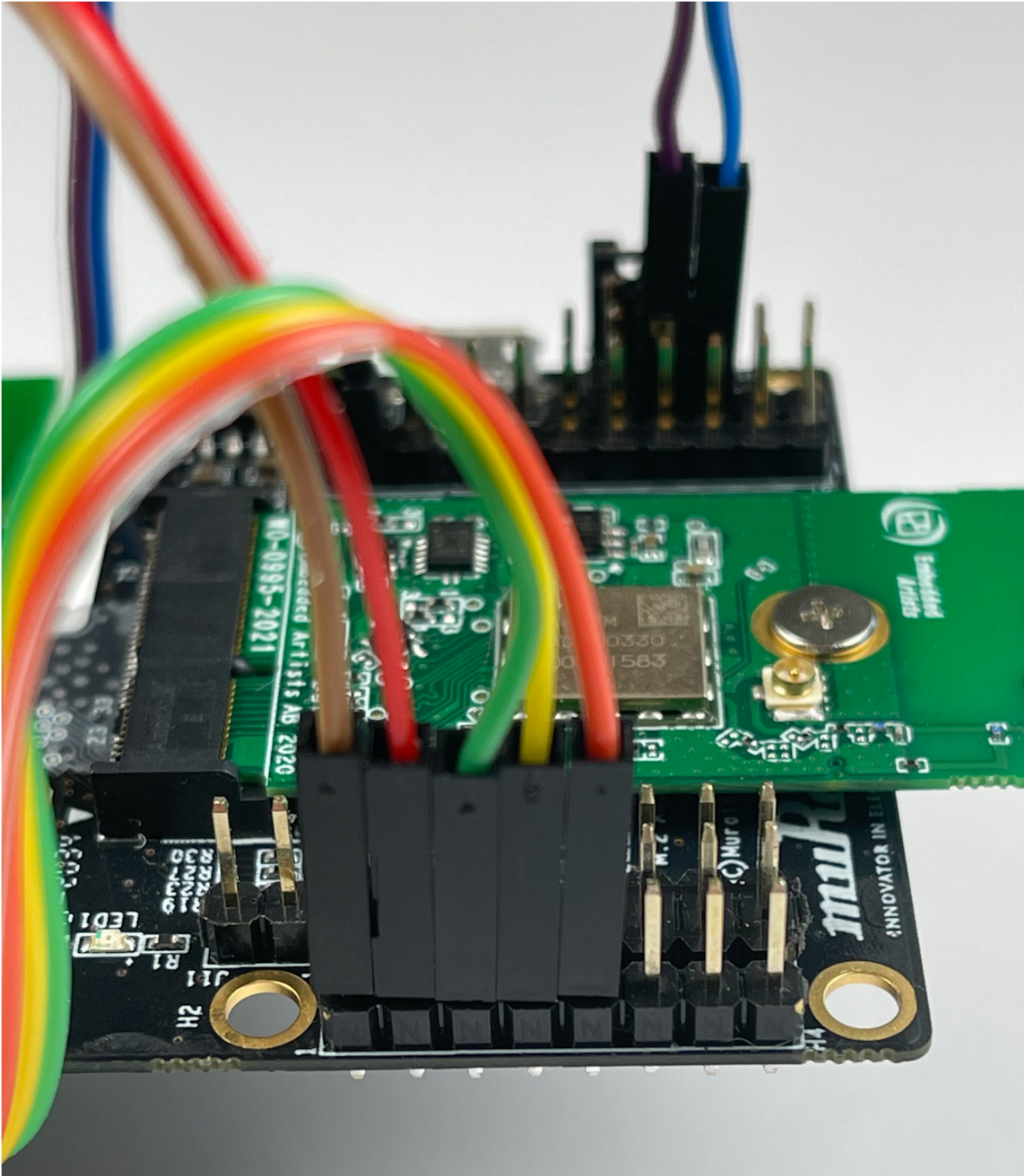


Figure 24: Cable connections on uSD-M.2 Adapter (J8 Header)

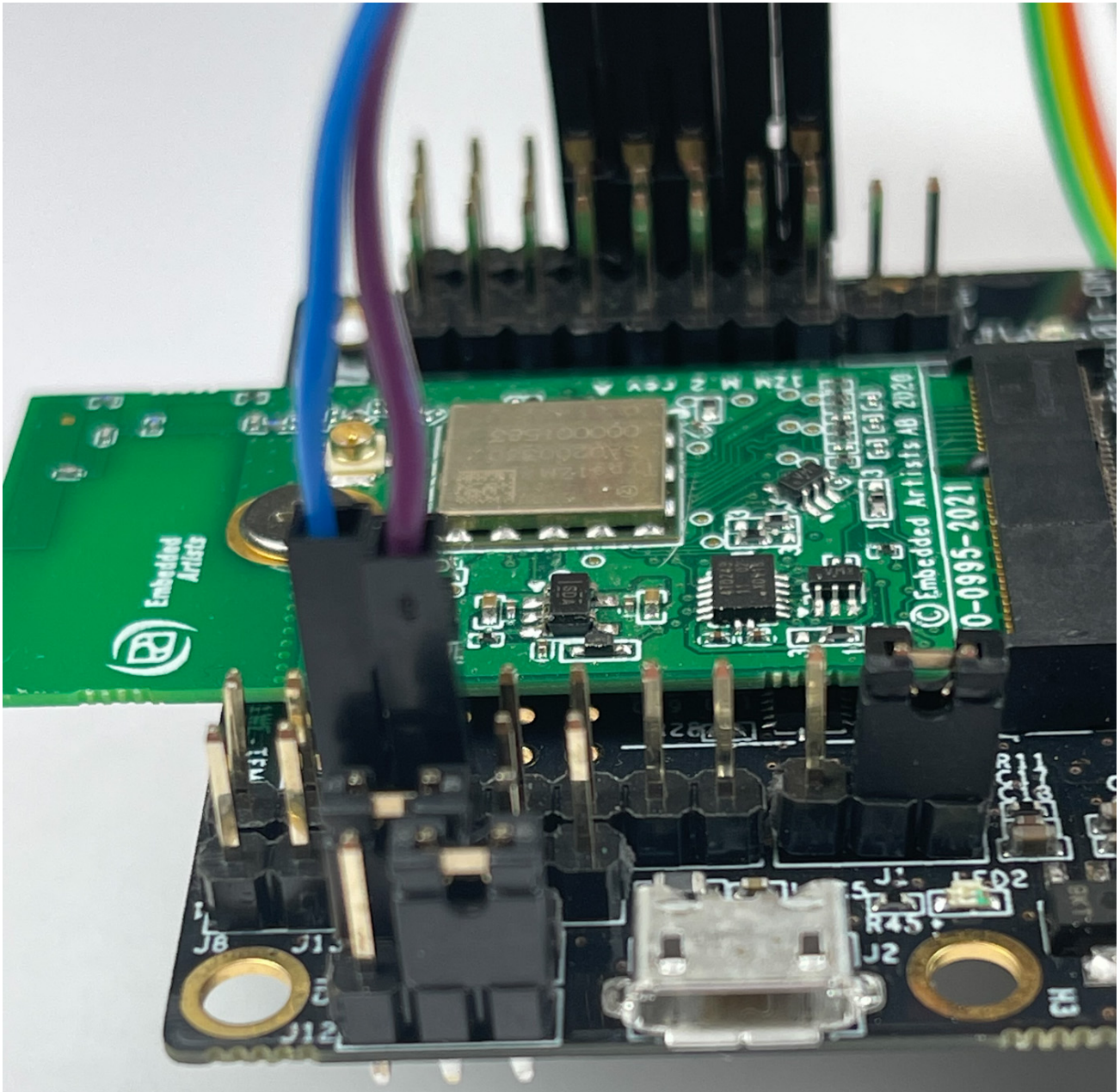


Figure 25: Low-Profile Jumper Wires (Digi-Key part number 1988-1178-ND)

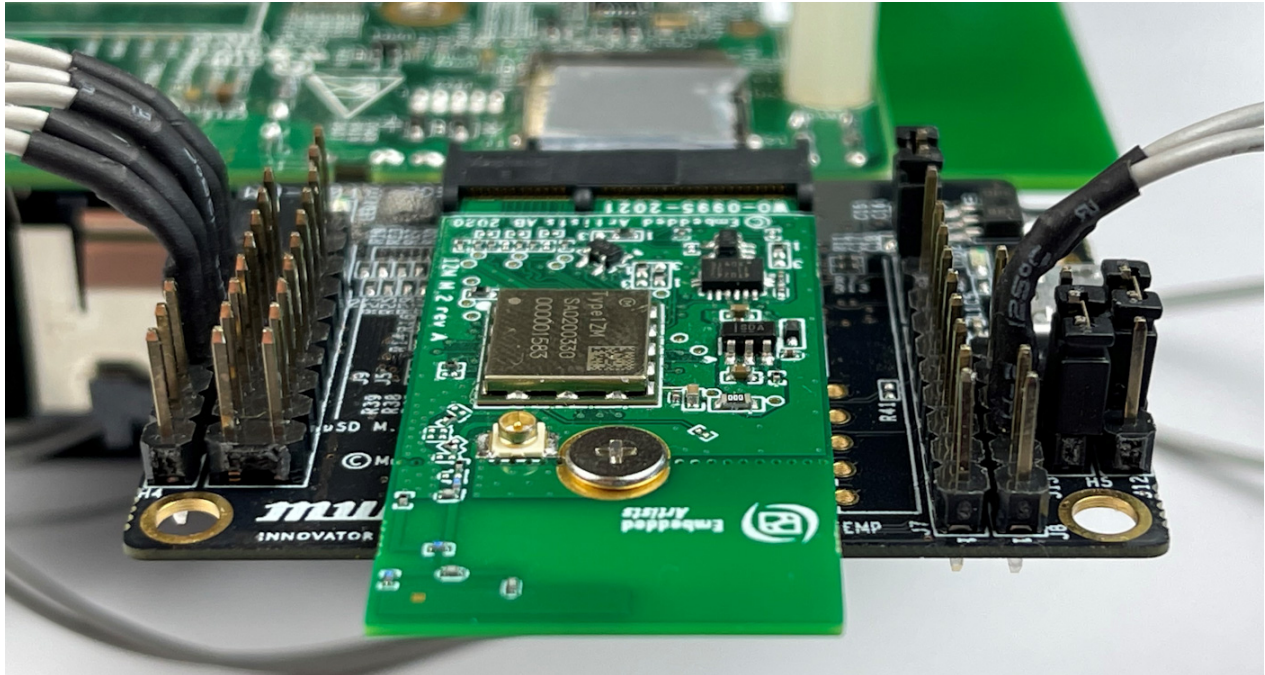


Figure 26: NXP i.MX EVK with Low-Profile Jumper Wires

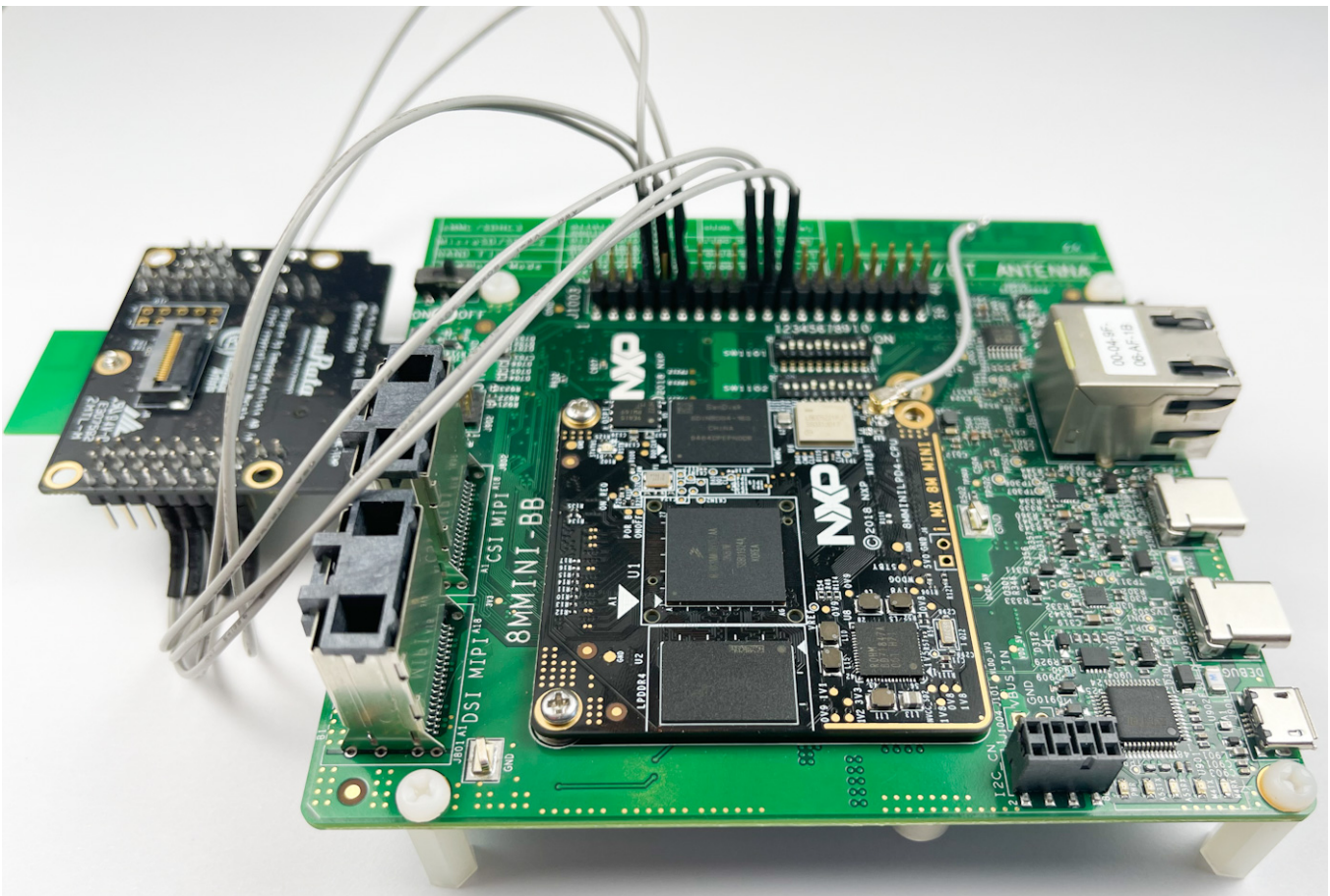
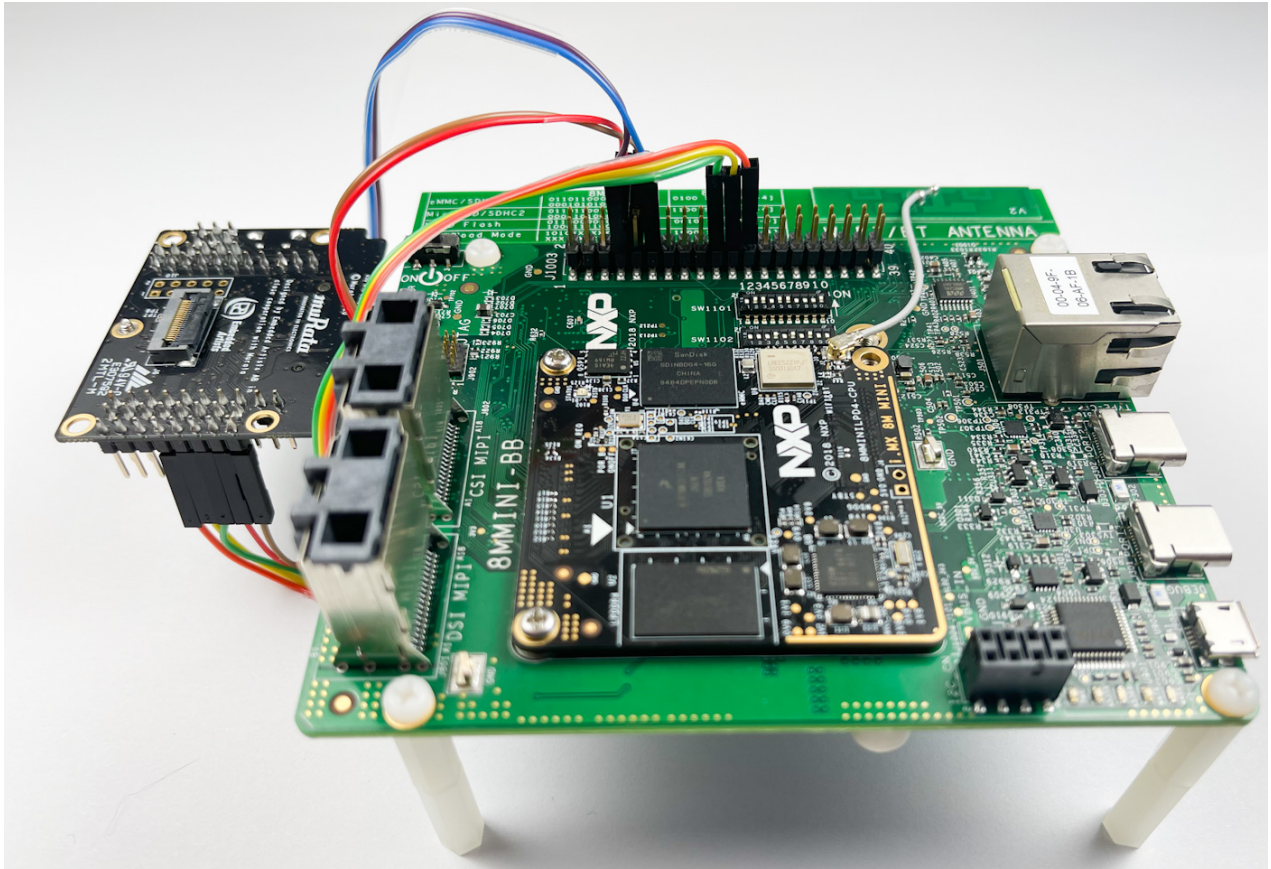


Figure 27: Additional Hex Standoff (Digi-Key part number RPC3570-ND)



7 Test/Verification of Wi-Fi and Bluetooth

Now the kernel should be booting correctly on the NXP i.MX platform with Murata module being correctly initialized (correct DTB configured and NXP 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. The relevant folders and files are summarized in **Table 13**.

Table 13: Embedded Wi-Fi/Bluetooth Files

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

7.1 Wi-Fi Interface Test/Verification

7.1.1 Useful Environment Setup on NXP Linux

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

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

7.1.2 Switch module script

Murata has included a script file in its images for customers to easily set up and switch between EVBs. **The usage is as shown below:**

```
switch_module.sh <module>
```

Where <module> can be:

- **1ZM:** Sets up the EVK for 1ZM module
- **1YM-PCle:** Sets up the EVK for PCIe based 1YM module
- **1YM-SDIO:** Sets up the EVK for SDIO based 1YM module

The “***switch_modules.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.

Enter the command, “***reboot***”, for automatic loading of necessary drivers.

```
reboot
```

The “***switch_module.sh <module>***” and reboot 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 (and has been set up for the correct EVB type using the “***switch_module.sh***” script), the WLAN driver is loaded automatically. As part of driver loading sequence (in this example), the WLAN device is probed over the SDIO bus. As an example, we will bring up Wi-Fi when using **following configuration:**

- NXP i.MX 6ULL EVK
- Murata’s uSD-M.2 Adapter
- Embedded Artists’ Type 1ZM M.2 Module (EVB)
- 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
mmc0: new high speed SDIO card at address 0001
...
cfg80211: Loading compiled-in X.509 certificates for regulatory database
cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
wlan: Loading MWLAN driver
vendor=0x02DF device=0x9149 class=0 function=1
Attach moal handle ops, card interface type: 0x105
SD8987: init module param from usr cfg
card_type: SD8987, config block: 0
cfg80211_wext=0xf
wfd_name=p2p
max_vir_bss=1
cal_data_cfg=none
drv_mode = 7
ps_mode = 2
auto_ds = 2
fw_name=nxp/sdiouart8987_combo_v0.bin
SDIO: max_segs=128 max_seg_size=65535
rx_work=0 cpu_num=1
Attach mlan adapter operations.card_type is 0x105.
wlan: Enable TX SG mode
wlan: Enable RX SG mode
Request firmware: nxp/sdiouart8987_combo_v0.bin
Wlan: FW download over, firmwarelen=526996 downloaded 526996
WLAN FW is active
on_time is 17660343376
fw_cap_info=0x181c3f03, dev_cap_mask=0xffffffff
max_p2p_conn = 8, max_sta_conn = 8
imx-sdma 20ec000.sdma: loaded firmware 3.5
wlan: version = SD8987---16.92.10.p207-MXM4X16186.p6-GPL-(FP92)
wlan: Driver loaded successfully
```

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

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

Now invoke “***ifconfig wlan0 up***” command to initialize the “***wlan0***” (WLAN) interface.


```
$ ifconfig wlan0 up
```

```
$ ifconfig wlan0
```

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

← WLAN MAC Address

← “wlan0” interface is UP

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, a simple method for scanning is presented: using the 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*”, “*iperf*”, etc.). Very attenuated signals will be in the high 80’s or 90’s (see “*RSSI*” value). If close to an Access Point, the returned “*RSSI*” value should be between -30 and -50 dBm for a properly configured setup.

“*iw*” is the default Linux command line tool for controlling a WLAN interface. One useful link to learn more about “*iw*” is on the “**Linux Wireless wiki**” here:

<https://wireless.wiki.kernel.org/en/users/documentation/iw>. In the following example of listing WLAN devices and performing a scan, one active AP has SSID of “**Murata_5G**”. Here are expected results:

```
$ iw dev ← list available WLAN devices
```

```
phy#0
```

```
Interface p2p0
```

```
ifindex 9
```

```
wdev 0x3
```

```
addr d6:53:83:be:4b:04
```

```
type managed
```

```
txpower 24.00 dBm
```

```
Interface uap0
```

```
ifindex 8
```

```
wdev 0x2
```

```
addr d4:53:83:be:4c:04
```

```
type AP
```

```
txpower 24.00 dBm
```

```
Interface wlan0
```

```
ifindex 7
```

```
wdev 0x1
```

```
addr d4:53:83:be:4b:04
```

```
type managed
```

```
txpower 24.00 dBm
```

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

wlan: wlan0 START SCAN

wlan: SCAN COMPLETED: scanned AP count=13

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.
```

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

NOTE: In the following test sequences of using “*iw*” command, the WLAN interface is not assigned an IP address. That is done later in **Section 7.1.7** where connectivity testing is performed.

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

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

Check status of connection with “*iw*” link command:

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

  bss flags:
  dtim period: 1
  beacon int: 100
```

7.1.6 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, we need to follow these steps:

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

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

```
$ killall wpa_supplicant
$ wpa_supplicant -i mlan0 -D nl80211 -c /etc/wpa_supplicant.conf -B
Successfully initialized wpa_supplicant
rfkill: Cannot open RFKILL control device
rfkill: Cannot get wiphy information
IPv6: ADDRCONF(NETDEV_CHANGE): mlan0: link becomes ready
```

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=2 ttl=64 time=10.053 ms
64 bytes from 192.168.1.1: seq=4 ttl=64 time=12.364 ms
^C ← Enter <CTRL-C> to terminate ping session
--- 192.168.1.1 ping statistics ---
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-lh" 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:aff: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)
mlan0 Link encap:Ethernet HWaddr b8:d7:af:56:61:fc ← existing "mlan0" 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-mlan0-0 192.168.2.1 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 **“uaputl.exe”** application to configure the i.MX/Murata Wi-Fi platform as a “Soft AP” or “Wi-Fi hot spot”. Only unsecured configuration is presented. First off, we need to kill the WPA supplicant:

```
$ killall wpa_supplicant
```

The uap0 interface is present to be used as a Soft AP interface. We need to turn it on.

```
$ ifconfig uap0 192.168.1.1 up
```

The following uaputl.exe commands can now be invoked to bring up a Soft AP with SSID “Test_SSID”:

```
cd /usr/share/nxp_wireless/bin_mxm_wifiex
./uaputl.exe -i uap0 sys_cfg_rates 0x8C 0x98 0xB0 0x12 0x24 0x48 0x60 0x6C
./uaputl.exe -i uap0 vhtcfg 2 3 1 0x33D07130 0xFFFFE 0xFFFFE
./uaputl.exe -i uap0 sys_cfg_channel 44 2
./uaputl.exe -i uap0 sys_cfg_ssid Test_SSID
./uaputl.exe -i uap0 bss_start
```

We can now associate from another client device and ping the Wi-Fi hotspot.

7.2 Bluetooth Interface Test/Verification

Before initializing the Bluetooth interface, the WLAN interface (mLAN0) must be brought up first. For the standard/default configuration of BT-UART interconnect (i.e. 1ZM or 1YM), 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 “*modem_reset*” construct in DTS file, the Bluetooth core should come up in the correct state to be initialized (i.e. as kernel boots, the Bluetooth core is reset and is taken out of reset). Here is the default command sequence to verify Bluetooth functionality:

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

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 1ZM module:

```
$ hciattach /dev/ttymx1 any 115200 flow
Setting TTY to N_HCI line discipline
Device setup complete
$ hciconfig hci0 up
$ hcitool -i hci0 cmd 0x3f 0x0009 0xc0 0xc6 0x2d 0x00
< HCI Command: ogf 0x3f, ocf 0x0009, plen 4
  C0 C6 2D 00
> HCI Event: 0x0e plen 4
  01 09 FC 00
$ killall hciattach
[ 1669.277042] Bluetooth: hci0: sending frame failed (-49) ← ignore this error
$ hciattach /dev/ttymx1 any -s 3000000 3000000 flow
Setting TTY to N_HCI line discipline
Device setup complete
$ hciconfig hci0 up
$ hciconfig hci0 piscan
$ hciconfig hci0 noencrypt
$ hciconfig -a
$ 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	
i.MX 8M Mini/Nano EVK (uSD)	GPIO140; UART3	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.

7.2.1 Bringing Up 1YM-SDIO* Bluetooth Interface

Different steps are required to initialize Bluetooth over SDIO interface on 1YM (“1YM-SDIO*” or WLAN_SDIO/BT-SDIO configuration). Note this configuration is only for ***evaluation testing only***. Before initializing the Bluetooth interface, the WLAN interface (mlan0) must be brought up first. Enter the following commands for bringing up Bluetooth over SDIO interface.

```
$ cd /usr/share/nxp_wireless/bin_sd8997_bt
$ insmod bt8997.ko
$ hciconfig hci0 up
$ hcitool -i hci0 cmd 3f ee 01 XX
$ hcitool scan
```

Sample outputs after executing the above commands are provided below for reference.

```
$ cd /usr/share/nxp_wireless/bin_sd8997_bt
$ insmod bt8997.ko
BT: Loading driver
BT FW is active(0)
BT: FW already downloaded!
BT: Driver loaded successfully

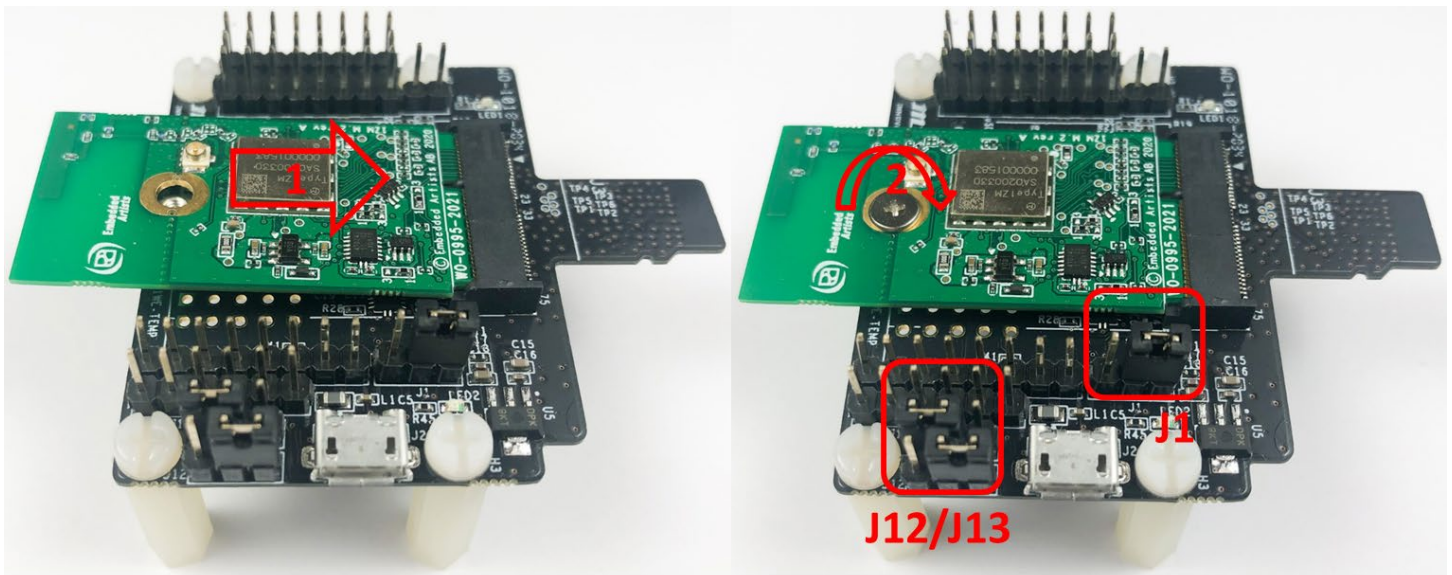
$ hciconfig hci0 up
$ hcitool -i hci0 cmd 3f ee 01 XX
< HCI Command: ogf 0x3f, ocf 0x00ee, plen 2
01 00
> HCI Event: 0x0e plen 4
01 EE FC 00
$ hcitool scan
Scanning ...
EC:CE:D7:C2:C6:55    Work phone
```

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 28**), make sure to (#1) firmly insert it before using M.2 screw to (#2) secure it in place. Important Jumpers (J12, J13, and J1) are highlighted.

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

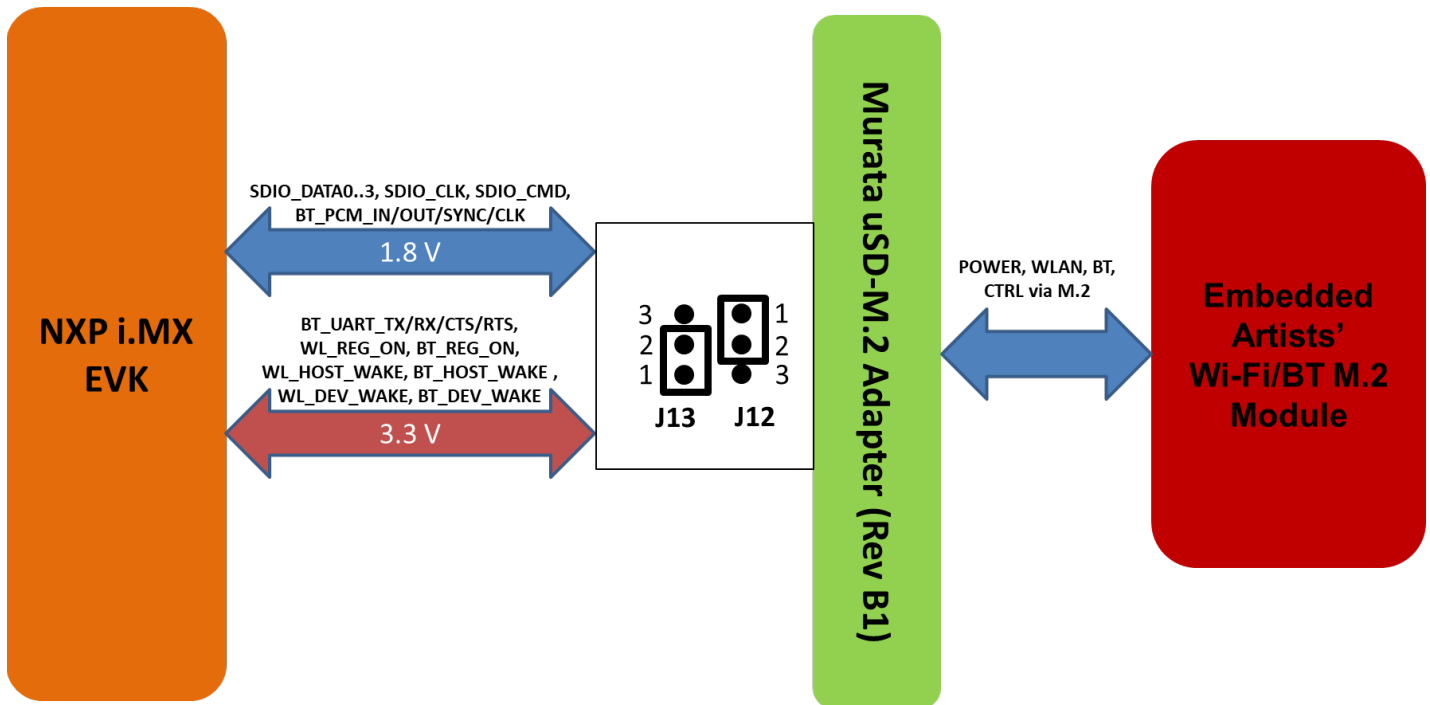


8.2 Configuring uSD-M.2 Adapter Jumpers for Correct VIO Signaling

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

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

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



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

On both legacy NXP i.MX 6 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.

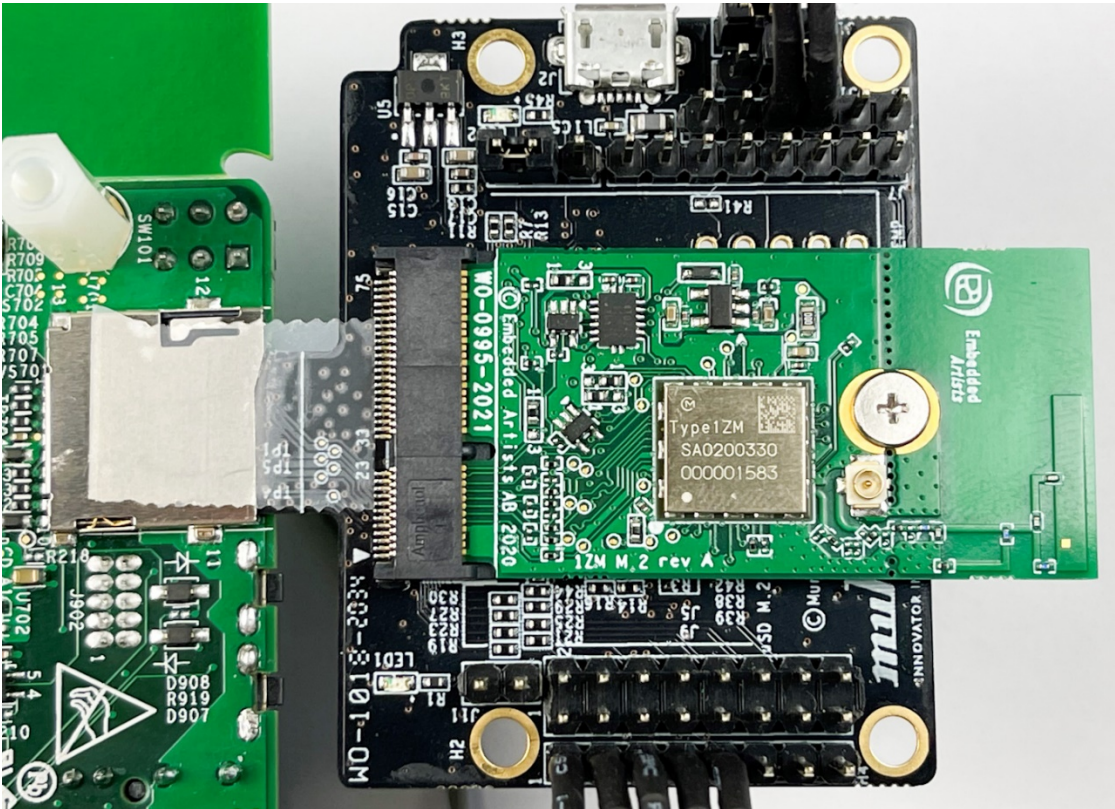
To properly secure the uSD-M.2 Adapter interconnect on the i.MX 6 EVK's, Murata **strongly recommends** to simply tape the uSD Adapter-SD Card connection and the SD Card-EVK connection as shown in **Figure 30**. Note that taping the SD Card-EVK connection makes the platform a little less flexible to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

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

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



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



8.4 uSD-M.2 Adapter High-Level Description

Figure 32 and **Figure 33** show the features on the uSD-M.2 Adapter; with text explanation in **Table 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 32: uSD-M.2 Adapter Features (Top View)

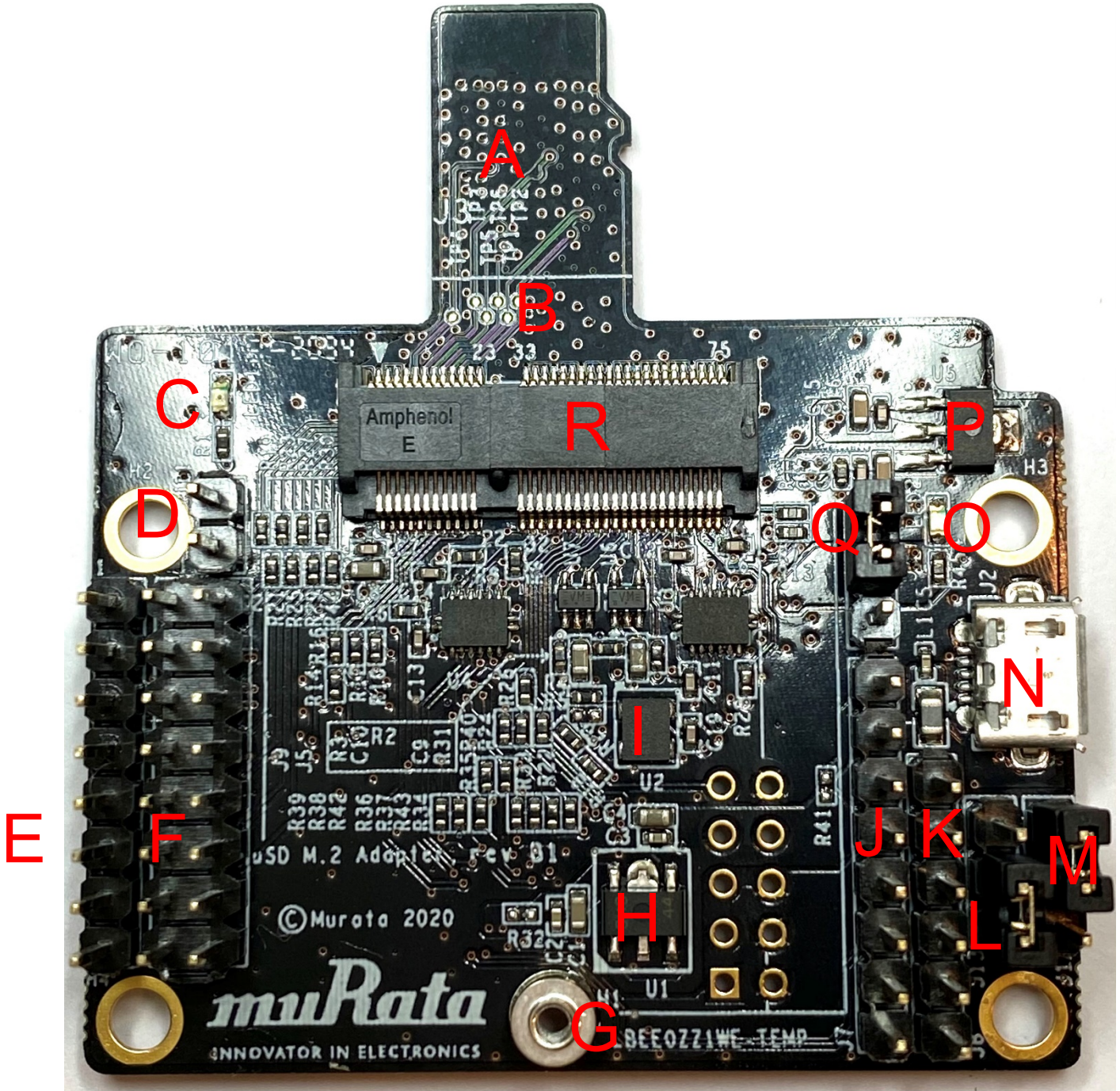
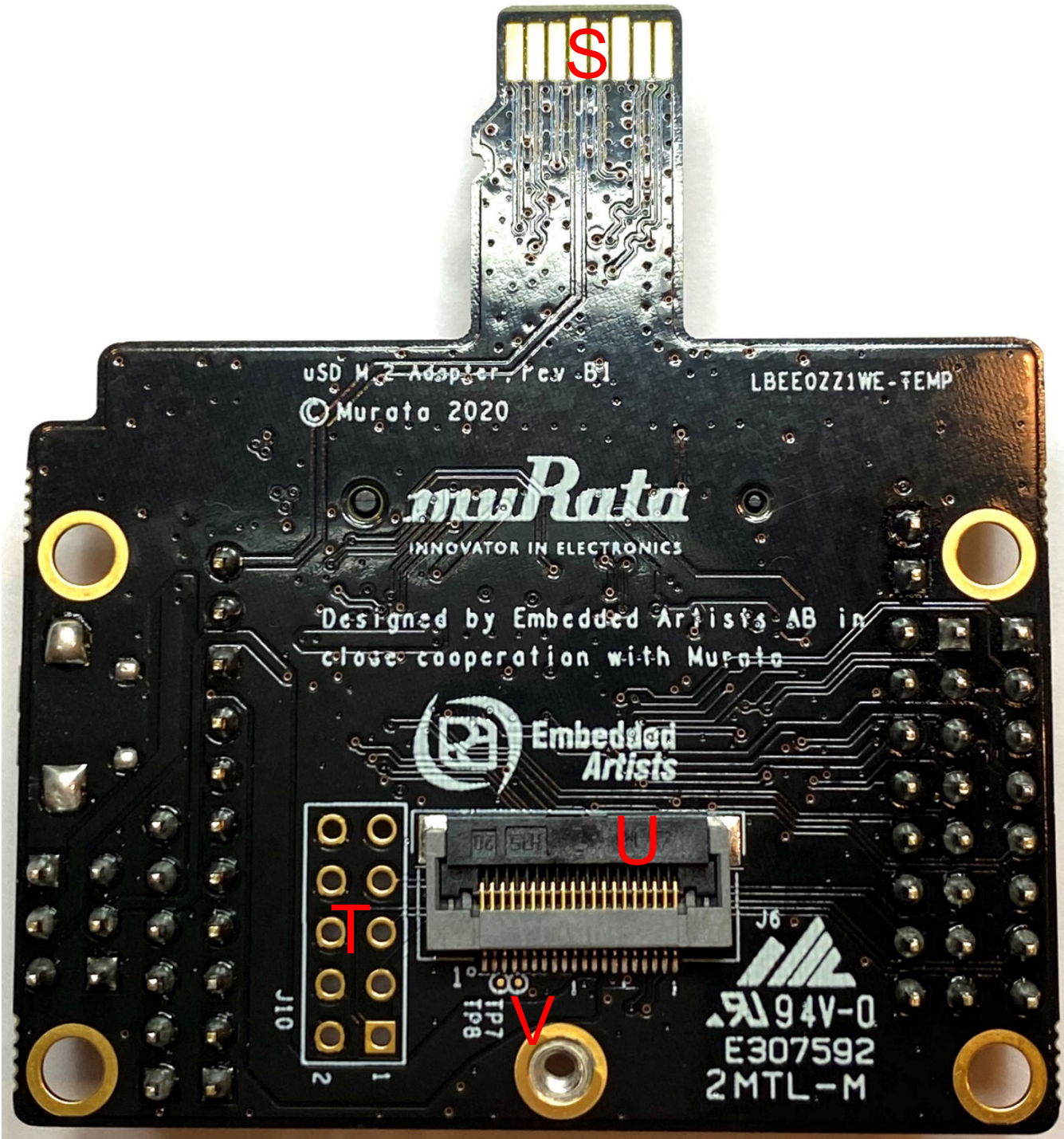


Figure 33: uSD-M.2 Adapter Features (Bottom View)



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




Embedded Artists designs, manufactures, and distributes all Wi-Fi/Bluetooth M.2 modules featuring Murata's mass-market modules (currently 1ZM and 1YM for NXP-based solutions). Murata partners very closely with Embedded Artists to test/validate all Wi-Fi/BT M.2 Modules. Customers can easily obtain these M.2 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/.

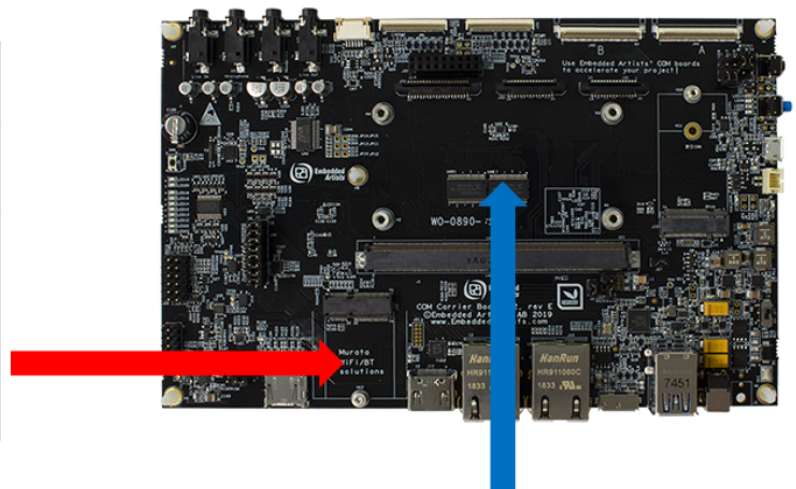
10 Embedded Artists' i.MX + Wireless Solution

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

1ZM	1YM	1YM-SDIO
NXP 88W8987	NXP 88W8997	NXP 88W8997
		













									
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	1ZM	1YM
	NXP 88W8987	NXP 88W8997
<u>iMX8M Quad COM</u>	M.2	M.2
<u>iMX8M Mini uCOM</u>	M.2	M.2
<u>iMX8M Nano uCOM</u>	M.2	M.2
<u>iMX7 Dual COM</u>	M.2	M.2
<u>iMX7 Dual uCOM</u>	M.2	M.2
<u>iMX7ULP uCOM</u>	M.2	M.2
<u>iMX6 Quad COM</u>	NC	M.2
<u>iMX6 DualLite COM</u>	NC	M.2
<u>iMX6 SoloX COM</u>	NC	M.2
<u>iMX6 UltraLite COM</u>	M.2	M.2

M.2 = Works with onboard M.2 slot
 NC = No Connection option (due to 1ZM only supporting 1.8V SDIO VIO)

Table 17: Embedded Artists' Landing Pages

Landing Pages	Notes
<u>Embedded Artists' Website</u>	The Art of Embedded Systems Development – made EASY™
<u>i.MX 6/7/8 COM Boards</u>	Listing of Computer-on-Module boards.
<u>i.MX 6/7/8 COM Carrier Board V2</u>	Main baseboard which all the COM boards plug into.
<u>Getting Started with i.MX 6/7/8 Developer's Kit V2</u>	How to bring up i.MX 6/7/8 Dev Kit (V2).
<u>M.2 Module Family</u>	Top level listing of 1ZM and 1YM M.2 EVB.
<u>Application Development on an i.MX Developer's Kit</u>	Description of C/C++, Python, Node.js, and Qt5 development.
<u>Devices and Peripherals on an i.MX Kit</u>	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
<u>i.MX 6/7/8 COM Carrier Board V2 Datasheet</u>	Comprehensive definition of COM Carrier (baseboard).
<u>i.MX6/7/8 COM Carrier Board V2 Schematics</u>	Complete schematics including clear definition of uSD-M.2 Adapter.
<u>M.2 SDIO Interface Schematic</u>	Reference schematic for customers designing in WLAN-SDIO M.2 EVB.
<u>M.2 PCIe Interface Schematic</u>	Reference schematic for customers designing in WLAN-PCIe M.2 EVB.
<u>EACOM Board Specification Guide</u>	Comprehensive definition of Embedded Artists' Computer-On-Module's.
<u>1ZM M.2 Module Datasheet</u>	Comprehensive details on 1ZM Wi-Fi/BT M.2 Module.
<u>1YM M.2 Module Datasheet</u>	Comprehensive details on 1YM 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/BT solution.

Table 20: List of Support Resources

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