

# Wi-Fi®/Bluetooth® (NXP) for i.MX

Linux User Manual - Rev. 3.0



## Table of Contents

1 Introduction .....	5
2 Murata’s Customized i.MX Yocto Image Explained .....	9
2.1 Overview .....	9
2.2 Murata GitHub: Cornerstone of Implementation .....	10
2.3 “meta-murata-wireless”: Pulling it All Together .....	11
2.3.1 What does “meta-murata-wireless” do exactly? .....	11
2.3.2 The Contents of “meta-murata-wireless” .....	11
3 Building i.MX Yocto Linux .....	13
3.1 i.MX Yocto Build: Overview .....	13
3.2 i.MX Yocto Build: The Fast Track (for Beginner Level Users) .....	13
3.2.1 Install Ubuntu .....	13
3.2.2 Download Murata’s Script Files .....	13
3.2.3 Configure Ubuntu for i.MX Yocto Build .....	14
3.2.4 Murata’s i.MX Yocto Build Script .....	15
3.3 i.MX Yocto Build: Manual Steps (for Advanced Users) .....	18
3.3.1 NXP i.MX versus Murata Module Interconnect .....	18
3.3.2 Host PC Preparation .....	20
3.3.3 Yocto Project Setup .....	20
3.3.4 Fetch “meta-murata-wireless” from GitHub and copy into “Sources” .....	21
3.3.5 Install Necessary “hooks” for “meta-murata-wireless” .....	22
3.3.6 Build Murata-Customized Yocto Image for Specific i.MX Target .....	23
3.4 i.MX Yocto Build: Manual Steps “Take 2” (Quick Recap) .....	23
3.4.1 Initialize Linux i.MX Yocto Default Build Environment .....	23
3.4.2 Configuring the i.MX Target .....	24
3.4.3 Add “meta-murata-wireless” Layer and Kick off the Build .....	24
3.5 i.MX Yocto Build: Manual Steps “Take 2” for “5.10.72” .....	24
4 Using NXP driver .....	25
4.1 How-To on NXP driver .....	25
4.1.1 Using Switch Modules Script File .....	25
4.1.2 Bringing Up Wi-Fi Interface .....	25
4.1.3 “mLAN0” initialization .....	27
4.2 Using manual driver load .....	27
4.2.1 1ZM – Manual load .....	27
4.2.2 1YM-SDIO Manual load .....	27
4.2.3 1YM-PCIe Manual load .....	27

4.3 Bluetooth Interface (Standard BT-UART) .....	28
5 Hierarchy of device tree source files for i.MX6/8 platforms .....	29
6 Embedded Artists' Solution .....	30
6.1 Build Steps .....	31
6.1.1 Create Build Directory .....	31
6.1.2 Initialize repo .....	31
6.1.3 Select Machine Configuration and Distribution .....	32
6.1.4 Initialize Build .....	32
6.1.5 Start Build .....	32
6.1.6 Deploy Image .....	32
6.2 Documentation .....	33
7 Murata's Regulatory Solution for NXP modules .....	34
7.1 Description of Regulatory Solution for WLAN/BT .....	34
7.2 Tree view of the files list for Linux Patched solution .....	34
7.2.1 Description of files .....	36
7.3 Purpose of switch_regions.sh .....	36
7.3.1 Sample Description of structure SD8987 from "wifi_mod_para.conf" configured for US with 1ZM module .....	37
7.3.2 Sample Description of structure PCIE8997 from "wifi_mod_para.conf" configured for EU with 1YM module .....	37
7.4 Sample log of switch_regions.sh: (Ex: 1YM-SDIO@8M-MINI) .....	38
8 Acronyms .....	39
9 Technical Support Contact .....	40
10 Appendix A: Useful "git" commands .....	41
11 Appendix B: Useful "bitbake" commands .....	42
12 Appendix C: Example of running Host Setup for Yocto Script .....	43
13 Appendix D: Example of running Murata Wireless Build .....	44
14 References .....	46
14.1 Murata's i.MX Wireless Solutions Landing Page .....	46
14.2 Murata uSD-M.2 Adapter Datasheet (Rev B2) .....	46
14.3 Murata uSD-M.2 Adapter Datasheet (Rev B1) .....	46
14.4 Murata uSD-M.2 Adapter Datasheet (legacy Rev A) .....	46
14.5 Murata Wi-Fi/BT Solution for i.MX Hardware User Manual .....	46
14.6 Murata Wi-Fi/Bluetooth for i.MX Linux User Guide for NXP-based Module .....	46
14.7 Murata Linux Patched Solution .....	46
14.8 Murata Wi-Fi/Bluetooth for i.MX Linux Quick Start Guide for NXP-based Module .....	47
14.9 Murata's Community Forum Support .....	47
14.10 Embedded Artists' Reference Documentation .....	47

14.11 NXP Reference Documentation .....	47
Revision History.....	49

## Figures

Figure 1: i.MX 6 EVK Wi-Fi/BT Interconnect Block Diagram .....	6
Figure 2: i.MX 8QXP MEK & 8MQuad EVK Wi-Fi/BT PCIe Interconnect Block Diagram.....	6
Figure 3: i.MX 8MQuad EVK Wi-Fi/BT SDIO Interconnect Block Diagram .....	7
Figure 4: i.MX 8MPlus EVK Wi-Fi/BT Interconnect Block Diagram.....	7
Figure 5: i.MX 8M Mini EVK Wi-Fi/BT PCIe Interconnect Block Diagram .....	8
Figure 6: i.MX 8M Mini/Nano EVK Wi-Fi/BT SDIO Interconnect Block Diagram .....	8
Figure 7: Configuring dash.....	15
Figure 8: DTS hierarchy for imx6ulevk .....	29
Figure 9: DTS hierarchy for imx6ull14x14evk.....	29
Figure 10: Combine i.MX COM with Wi-Fi/BT M.2 EVB .....	30

## Tables

Table 1: Document Conventions .....	4
Table 2: Current NXP i.MX Platforms Supported .....	5
Table 3: NXP i.MX EVK Part Number / Yocto (MACHINE) target / Kernel Version Matrix .....	9
Table 4: Murata GitHub Repositories used in build .....	10
Table 5: Config DTBs for EVKs.....	12
Table 6: Important folders/files in “meta-murata-wireless”.....	12
Table 7: NXP i.MX EVK Part Number / Murata Module Interconnect .....	19
Table 8: NXP i.MX EVK / Yocto (MACHINE) target / Kernel Version Matrix .....	19
Table 9: i.MX6/8 Targets supported by Murata .....	19
Table 10: Embedded Artists' i.MX Interconnect.....	31
Table 11: Embedded Artists' supported branches.....	31
Table 12: Embedded Artists' Machine Configuration.....	32
Table 13: Embedded Artists' Landing Pages .....	33
Table 14: Embedded Artists' Datasheets and Schematics .....	33
Table 15: Embedded Artists' User Manuals and Software .....	33
Table 16: Murata regulatory files.....	36
Table 17: List of Support Resources .....	40
Table 18: Useful “git” commands: .....	41
Table 19: Useful “bitbake” commands.....	42
Table 20: Embedded Artists Documentation Listing.....	47
Table 21: NXP Reference Documentation Listing for Supported Releases .....	48

## About This Document

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

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

## Audience & Purpose

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

## Document Conventions

**Table 1** describes the document conventions.

**Table 1: Document Conventions**

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

# 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. The Murata Connectivity Modules enable developers to minimize the development time and effort for connectivity function implementation. All steps necessary to build Murata Wi-Fi/BT module-enabling software are documented here. Currently NXP i.MX Linux 5.10.52 and Linux 5.10.72 BSPs are supported.

Customized software images enabling NXP's i.MX 6 and i.MX 8 Reference Platforms is provided by Murata's "meta-murata-wireless" Yocto solution. An automated build script makes image generation very simple.

Beyond validation of NXP i.MX platforms listed in **Table 2** below, the Murata software solution is easily applied to the complete i.MX 6/8 family. Users need only customize the Linux DTB (Device Tree Blob) which configures the kernel during boot.

This release also enables a custom evaluation release for Type 1YM (88W8997) module, allowing customers to interface the WLAN-SDIO instead of the default WLAN-PCIe bus.

**Table 2: Current NXP i.MX Platforms Supported**

NXP i.MX EVK Part number	NXP i.MX EVK	Murata Modules Supported	Interconnect
<a href="#">MCIMX8QXP-CPU</a>	i.MX 8QXP MEK	1YM, 1XL	M.2 (WLAN-PCIe)
<a href="#">8MPLUSLPD4-EVK</a>	i.MX 8MPLUS EVK	1ZM, 1YM, 1XK, 1XL, 2DS	M.2: 1ZM, 1YM (PCIe and SDIO), 1XK, 1XL (PCIe), 2DS
<a href="#">MCIMX8M-EVKB</a>	i.MX 8MQuad EVK	1ZM, 1YM, 1XK, 1XL, 2DS	uSD-M.2 Adapter: 1ZM, 1YM, 1XK, 2DS (WLAN Only) M.2 (WLAN-PCIe): 1YM, 1XL
<a href="#">8MMINILPD4-EVKB</a>	i.MX 8M Mini EVK	1ZM, 1YM, 1XK, 1XL, 2DS	uSD-M.2 Adapter: 1ZM, 1YM, 1XK, 2DS M.2 (WLAN-PCIe): 1YM – WLAN Only, 1XL – WLAN Only
<a href="#">8MNANOD4-EVK</a>	i.MX 8M Nano EVK	1ZM, 1YM, 1XK, 2DS	uSD-M.2 Adapter
<a href="#">MCIMX6UL-EVKB</a>	i.MX 6UL EVK	1ZM, 1YM, 1XK, 2DS	uSD-M.2 Adapter
<a href="#">MCIMX6ULL-EVK</a>	i.MX 6ULL EVK	1ZM, 1YM, 1XK, 2DS	uSD-M.2 Adapter



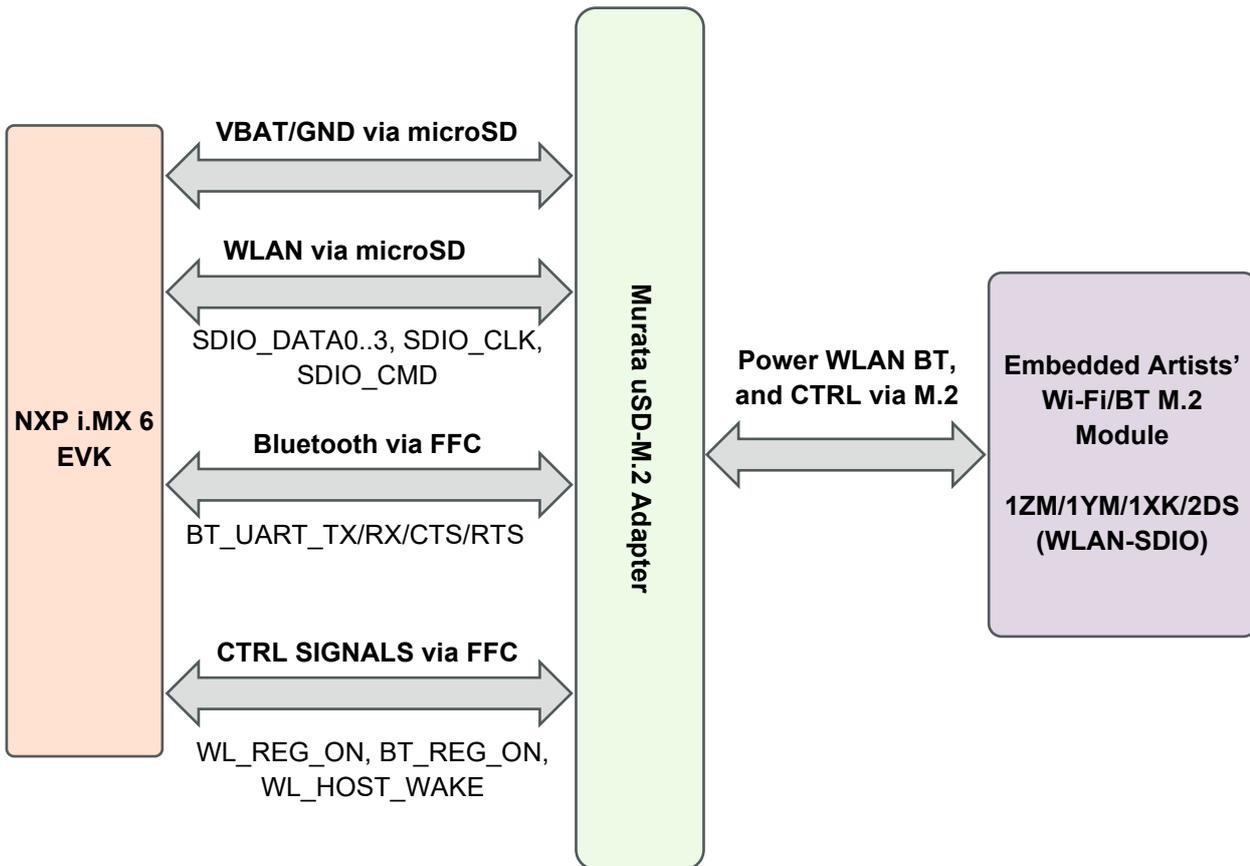
2DS Supports only WLAN. There is no Bluetooth.

**Figure 1** illustrates a high-level connection diagram between NXP i.MX 6 EVKs and Murata module. This configuration is enabled with Murata's uSD-M.2 Adapter and Embedded Artists' Wi-Fi/BT M.2 EVBs.



There are limitations on this interconnect. Chief amongst them is fixed WLAN-SDIO VIO of 3.3V on certain i.MX 6 platforms, and a maximum WLAN-SDIO clock speed of 50 MHz. Please refer to the [Linux User Guide](#), [uSD-M.2 Adapter Datasheet](#), and [Hardware User Manual](#) for more details. Given that WLAN-SDIO VIO of Type 1ZM/1YM is 1.8V only, only NXP i.MX 6UL(L) EVKs work for this configuration.

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

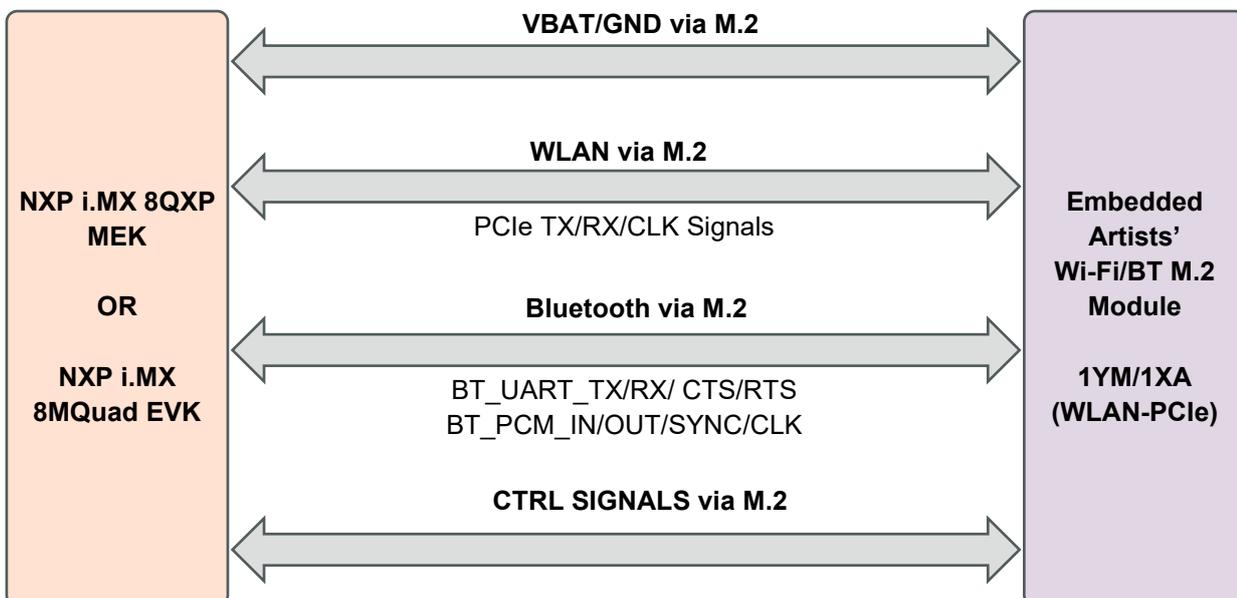


**Figure 2** shows a simplified block diagram for the i.MX 8QXP MEK & i.MX 8MQuad EVK Wi-Fi/BT interconnect. Currently Type 1YM, 1XL are supported with WLAN-PCIe interface.



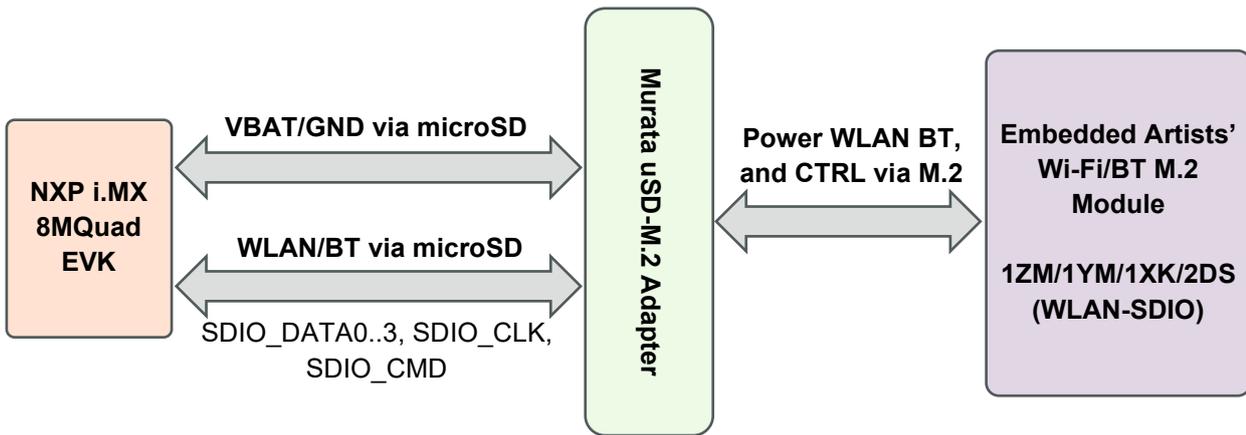
No adapter is needed for this interconnect as the M.2 EVBs can be connected directly to the i.MX 8 EVK.

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



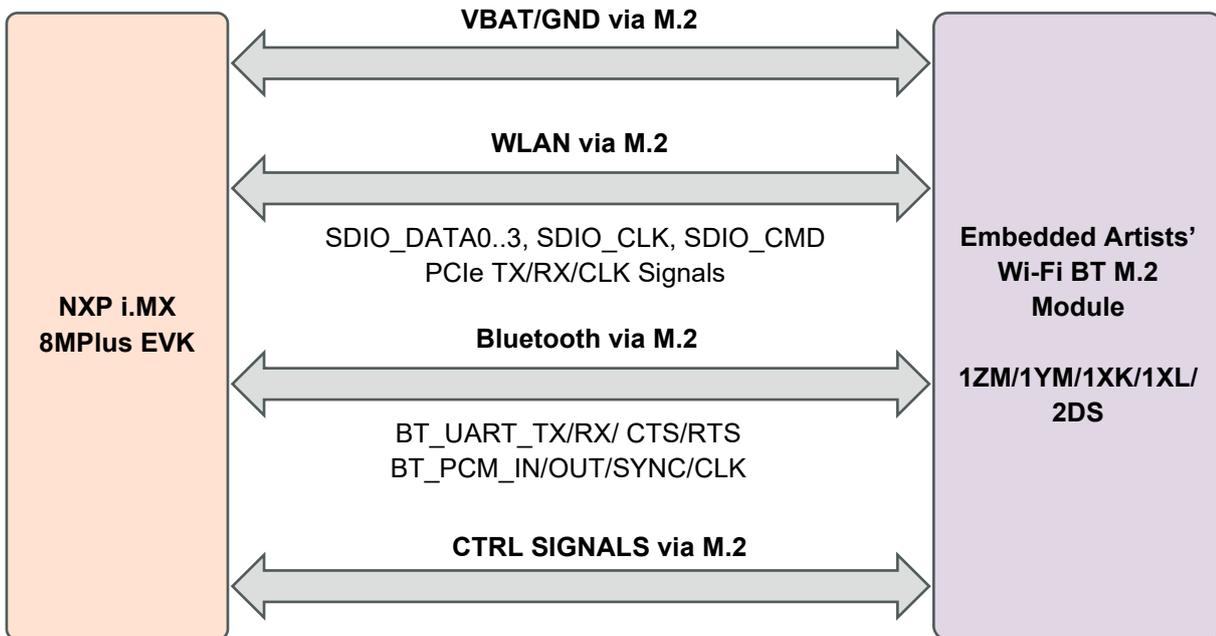
**Figure 3** shows a simplified block diagram for the i.MX 8MQuad EVK Wi-Fi/BT SDIO interconnect. Type 1ZM, 1YM, 1XK and 2DS modules are supported via the uSD-M.2 adapter on i.MX 8MQuad EVK. Note that this EVK can use onboard eMMC flash. The eMMC flash is necessary as the default uSD card slot is no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

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



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

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

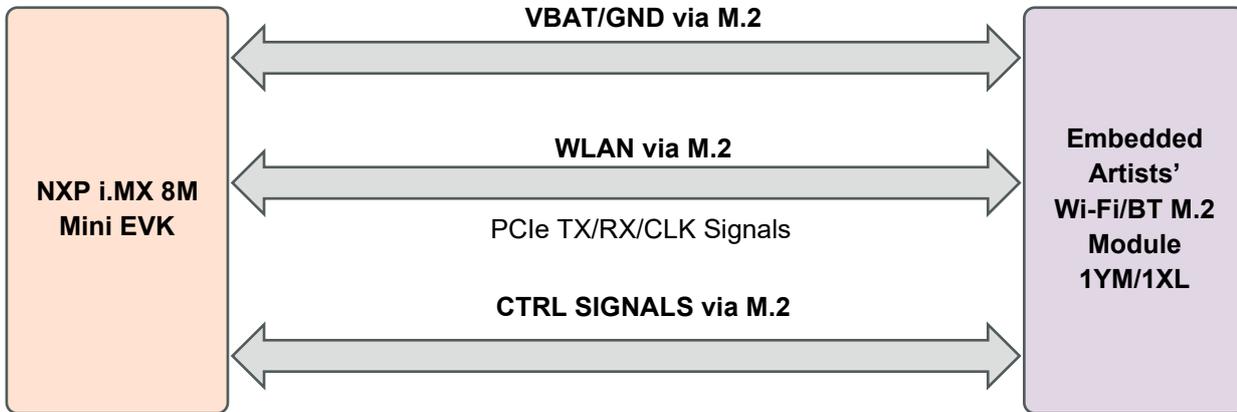


**Figure 5** shows a simplified block diagram for the i.MX 8M Mini EVK WLAN interconnect with onboard M.2 EVB option. Currently Type 1YM, 1XL are supported with WLAN-PCIe interface.



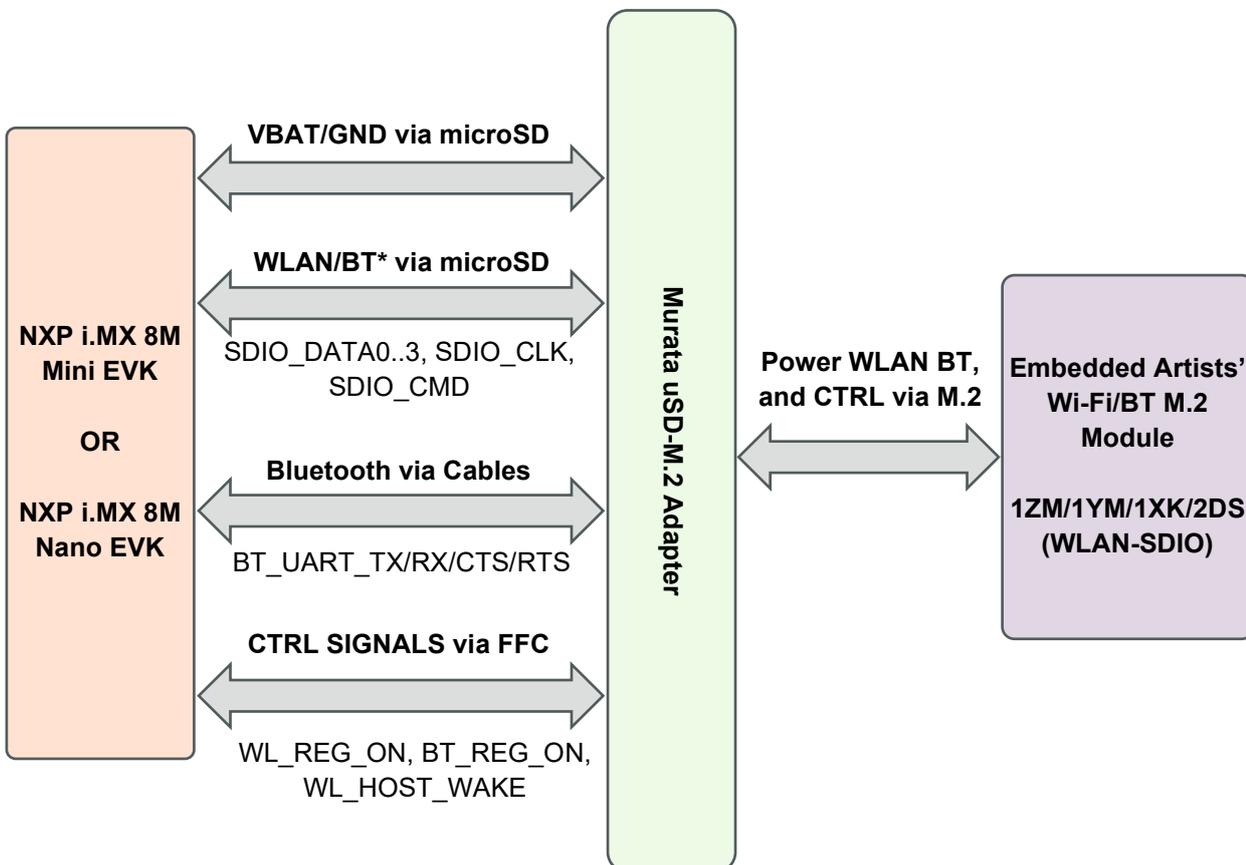
The NXP i.MX 8M Mini EVK does not bring out the Bluetooth signals to the M.2 connector – WLAN only.

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



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

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



## 2 Murata's Customized i.MX Yocto Image Explained

### 2.1 Overview

The NXP [Linux 5.10.52](#) and [Linux 5.10.72](#) BSPs for i.MX 6/8 has the WLAN driver supporting Type 1ZM (88W8987) integrated into the NXP baseline build. For Type 1YM (88W8997) only certain i.MX8 platforms have the default WLAN-PCIe driver integrated. Murata's customized "meta-murata-wireless" provides uniform Type 1ZM/1YM support on all i.MX 6/7/8 software builds.

Embedded Artists' Type 1YM M.2 EVB can be configured (via resistor strapping options – see Linux User Guide or Hardware User Manual) to support an additional evaluation mode of WLAN-SDIO/BT-SDIO (referenced as "1YM-SDIO\*" in this document). 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".

With Bluetooth already supported by integrated BlueZ stack, Murata's customized Yocto layer "meta-murata-wireless" seamlessly integrates the WLAN drivers. More specifically it 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 7**.
- Customized (evaluation-only) solution for Type 1YM configured in WLAN-SDIO/BT-SDIO mode. The eventual NXP release will integrate both WLAN-PCIe/BT-UART and 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) EVKs.
- Customized Murata RF calibration files used to correctly configure/tune Murata's modules.

With hardware interconnect not a limiting factor, Murata's customized Yocto layer supports the following NXP i.MX EVKs as outlined in **Table 3**. "MACHINE=target" is a direct reference to Yocto build. The "target" string is the keyword used to select hardware configuration for the build.

**Table 3: NXP i.MX EVK Part Number / Yocto (MACHINE) target / Kernel Version Matrix**

NXP i.MX EVK Part #	MACHINE=target	Kernel Versions Supported	Interconnect
<a href="#">MCIMX8QXP-CPU</a>	imx8qxpmev	5.10.52, 5.10.72	M.2
<a href="#">8MPLUSLPD4-EVK</a>	imx8mp-lpddr4-evk	5.10.52, 5.10.72	M.2
<a href="#">MCIMX8M-EVKB</a>	imx8mqevk	5.10.52, 5.10.72	M.2
<a href="#">8MMINILPD4-EVKB</a>	imx8mmevk	5.10.52, 5.10.72	M.2, uSD-M.2 Adapter
<a href="#">8MNANOD4-EVK</a>	imx8mnddr4evk	5.10.52, 5.10.72	uSD-M.2 Adapter
<a href="#">MCIMX6UL-EVKB</a>	imx6ulevk	5.10.52, 5.10.72	uSD-M.2 Adapter
<a href="#">MCIMX6ULL-EVK</a>	imx6ull14x14evk	5.10.52, 5.10.72	uSD-M.2 Adapter

## 2.2 Murata GitHub: Cornerstone of Implementation

“THE CORNERSTONE IS THE FIRST STONE SET IN THE CONSTRUCTION OF A MASONRY FOUNDATION, IMPORTANT SINCE ALL OTHER STONES WILL BE SET IN REFERENCE TO THIS STONE, THUS DETERMINING THE POSITION OF THE ENTIRE STRUCTURE.”

Murata GitHub is the cornerstone of the customized i.MX Yocto implementation for NXP wireless. Refer to **Table 4** for a complete listing of the Murata repositories currently used to build the i.MX Yocto image. For each repository, all available branch and release/tag names are included. All repositories are hosted at [Murata GitHub](#). For “meta-murata-wireless” GIT repository, the following branch name is used:

- i.MX architecture, Linux kernel version (Yocto codename), and driver version release.

The release names follow the branch-naming convention. When deciding on using a branch versus release/tag, the user will want to consider the following points:

- Release/tag is strongly recommended for end users who want a stable/tested version of the customized release. Murata only “tags” a branch after having run through a testing cycle on the various i.MX/module configurations.
- If the user \*just\* needs to build a reference SD card image, then it is strongly recommended to use a release/tag.
- Branches are recommended for users who need the latest updates. Users can examine the latest git commits on GitHub for any given repository. The most up-to-date branch by default is “master”. Note that Murata does limited testing on updates (new git commits) to a given branch. If the user runs into any unexpected difficulty, please post to [Murata’s Community Forum](#).

**Table 4: Murata GitHub Repositories used in build**

Murata GitHub Repository Name	Branch Names	Latest Release / Tag Names	Contents
“meta-murata-wireless”	<p>“master”</p> <p>“imx-hardknott-5-10-52”</p> <p>“imx-hardknott-5-10-72”</p> <p><a href="#">GitHub Listing</a></p>	<p>“imx-hardknott-5-10-52_r1.0”</p> <p>“imx-hardknott-5-10-72_r1.0”</p> <p><a href="#">GitHub Listing</a></p>	<p>“<i>meta-murata-wireless</i>” customized recipe layer. It drops into existing Yocto build environment. “master” branch only contains build script utilities – key starting point for user wanting to generate an image.</p>

Digging deeper into the Murata GitHub repositories, “meta-murata-wireless” repository is the “driver’s seat”. It determines which branches/releases of other branches are pulled into the Yocto build. This repository versions the Murata-customized Yocto layer which is copied into the Yocto “sources” sub-folder. In this Yocto build implementation, each i.MX/kernel implementation is independent. As such, there can be no single “master” branch that is the latest/greatest for all these unique branches. However, the “master” branch does contain a unique sub-folder “script-utils” with Linux scripts for assisting the user on automatically building a customized i.MX image.

## 2.3 “meta-murata-wireless”: Pulling it All Together

### 2.3.1 What does “meta-murata-wireless” do exactly?

Murata’s customized Yocto layer “meta-murata-wireless” provides the following enhancements / customizations:

- Comprehensive support for 1ZM/1YM/1XK/1XL/2DS on all possible i.MX targets - hardware interconnect is the only limiting factor: see **Table 7**.
- 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) EVKs.
- Customized Murata RF calibration files used to correctly configure/tune Murata’s modules.

For a Yocto primer, please refer to [this link](#) . For specifics on the i.MX Yocto implementation and their project user’s guide, please refer to **Table 21**.

### 2.3.2 The Contents of “meta-murata-wireless”

“meta-murata-wireless” is a Murata-customized Yocto layer. Each Yocto layer resides in the “sources” sub-folder. Murata’s implementation follows the Yocto “norm” with some minor exceptions due to required backport implementation. If you are planning on customizing or porting the Murata implementation, please read this section carefully. Refer to **Table 6** for a description of important folders/files in the “meta-murata-wireless” layer.

- “script-utils/latest/”: contains “Host\_Setup\_for\_Yocto.sh” and “Murata\_Wireless\_Yocto\_Build\_NXP.sh” script files. Refer to [Section 3.2](#)  for more details on how these script files accelerate the Linux image build process.
- “add-murata-layer-script/add-murata-wirless.sh”: This is the “hook”. When run, it will insert necessary code into the current i.MX Yocto build to pull in “meta-murata-wireless” layer. This script is run after i.MX Yocto build environment setup and i.MX target/graphics selection. It should only be run once for any given/unique build directory.
- “conf/layer.conf”: setting priorities for layers pulled into the Yocto build is very important. “meta-murata-wireless” is set to the highest priority (9) which guarantees that it’s packages will be compiled in if there is any conflict. As a specific example, there is “competition” between two WPA suplicant packages. One is included in “meta-murata-wireless” and the other is “meta” (i.MX default version). “meta” layer priority is 5, so the WPA suplicant package it pulls in will not be part of the image (version 2.6). Rather, the “meta-murata-wireless” WPA suplicant package (Version 2.9) with associated/configured/tested patches is compiled into the i.MX image.
- “freescale/<MACHINE>.conf”: When compiling i.MX image for the following EVKs, these machine configuration files (see **Table 5**) are critical. It pulls in the necessary DTB files to the image – otherwise the resulting SD card image would not have the correct Device Tree Blob (DTB) file to configure the kernel when the platform boots.

**Table 5: Config DTBs for EVKs**

EVK Name	Machine File Name
i.MX6UL	imx6ulevk.conf
i.MX6ULL	imx6ull14x14evk.conf
i.MX 8M Mini EVK	imx8_all.conf, layer.conf
i.MX 8M Nano EVK	imx8mnevk.conf, imx8mnddr4evk.conf, imx8mnlpddr4evk.conf

- “recipes-connectivity/murata-binaries/murata-binaries\_1.0.bb”: This recipe installs the following files:
  - Calibration file for 1YM-PCle.
  - switch\_module.sh for switching between NXP wireless (1zm, 1ym-sdio, 1ym-pcie, 1xk, 1xl and 2ds).
- “recipes-connectivity/murata-binaries/murata-binaries/switch\_module.sh”: This script file configures the WPA-Supplicant to point to default NXP wpa-supPLICANT and ensures automatic driver loading for 1ZM, 1YM-SDIO, 1YM-PCle, 1XL, 1XK and 2DS chipsets. It is placed in the location, “/usr/sbin”

**Usage:**

```
switch_module.sh <module>
# Where <module> is one of (case insensitive):
# 1zm, 1ym-sdio, 1ym-pcie, 1xl, 1xk, 2ds
```

- “recipes-connectivity/wpa-supPLICANT/wpa\_supPLICANT\_%.bbappend”: This folder contains the recipe file that configures the WPA SupPLICANT version (i.e., “wpa\_supPLICANT\_2.9.bb”). This file renames wpa supPLICANT executable to wpa\_supPLICANT.nxp.
- “recipes-kernel/linux/linux-imx\_<Kernel Version>.bbappend”: This file patches the i.MX kernel and makes any necessary changes to the kernel configuration file (“.config”).
- “recipes-kernel/linux/linux-imx-<Kernel Version>/”: This folder contains all the patches applied to the i.MX kernel. The “linux-imx\_<Kernel Version>.bbappend” recipe selects which patches to apply.

**Table 6: Important folders/files in “meta-murata-wireless”**

“meta-murata-wireless” folder/file	Notes
script-utils/latest/	Murata automated script files for host setup and i.MX Yocto build.
add-murata-layer-script/ add-murata-wireless.sh	Script file which modifies “bblayer.conf” and “local.conf”; thereby enabling the Murata-customized implementation for current “target” build.
conf/layer.conf	Enable “meta-murata-wireless” bbappend files and set priority for “meta-murata-wireless” layer.  Priority setting is important to avoid conflicts with multiple recipes in Yocto implementation such as WPA supPLICANT, Hostapd, etc.
freescale/imx6ulevk.conf freescale/imx6ull14x14evk.conf freescale/imx8_all.conf freescale/imx8mnddr4evk.conf freescale/imx8mnevk.conf	Ensures that all necessary i.MX6UL(L), i.MX 8M Mini EVK and i.MX 8M Nano EVK Device Tree Blob (DTB) files are included in boot folder of generated i.MX image.

freescale/imx8mnlpddr4evk.conf freescale/layer.conf	
recipes-connectivity/ murata-binaries/murata-binaries_1.0.bb	Installs switch module script and Calibration file for 1YM-PCIe.
recipes-connectivity/ murata-binaries/murata- binaries/switch_module.sh	Automatically switches WLAN driver loading for SDIO and PCIe NXP devices.
recipes-connectivity/ wpa-supPLICANT/wpa-supPLICANT_%.bbappend	Renames default WPA supplicant as NXP WPA supplicant (wpa_supPLICANT.nxp).
recipes-kernel/ linux/ linux-imx_<Kernel Version>.bbappend	Appends to the default i.MX recipe "meta-fsl-bsp-release/imx/meta-bsp/recipes-kernel/linux/linux-imx_<Kernel Version>.bb". This file patches the i.MX kernel and makes any necessary changes to the kernel configuration file (".config").

## 3 Building i.MX Yocto Linux

### 3.1 i.MX Yocto Build: Overview

Certain Intellectual Property issues (for 3<sup>rd</sup> party drivers that are integrated into the baseline NXP i.MX image) prevents direct access to i.MX image binaries. Therefore, it is necessary for the end user to arrive at their own "SD card" image. There are two paths to arrive at the necessary image:

1. Execute Murata automated build script. This is intended to ease the "startup" phase. However, the user needs to install Ubuntu on a machine (PC or virtual environment). If selecting this (easier option) then please refer to [Section 3.2](#).
2. Follow manually documented steps. Intended for users with reasonable familiarity of Linux and Yocto: refer to [Section 3.3](#). For users familiar with "meta-murata-wireless" implementation, all the steps from [Section 3.3](#) are repeated in a condensed version as examples in [Section 3.4](#) and [Section 3.5](#).

### 3.2 i.MX Yocto Build: The Fast Track (for Beginner Level Users)

#### 3.2.1 Install Ubuntu

First step is to install Ubuntu 14.04, 16.04, 18.04 or 20.04 (Murata's build is verified on Ubuntu 20.04 64-bit install) on the host - native PC or virtual environment like VMware. Host PC typically used requires Ubuntu 20.04/18.04/16.04/14.04 installed with 50 GB free disk space (80 GB needed for i.MX8 build). Murata has verified these build steps using Ubuntu 20.04 and 18.04 (x64). For more information on the Ubuntu download, please refer to [this link](#). The Ubuntu installation manual is provided [here](#).

#### 3.2.2 Download Murata's Script Files

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

1. Using "web browser" option to download "meta-murata-wireless" zip file and extract:
  - Click on "clone or download" button at [Murata GitHub](#).

- Now select “Download ZIP” option.
  - Once the file is downloaded, extract it with “unzip” command or folder UI.
  - Now go to the “meta-murata-wireless-master/script-utils/latest” folder where the necessary README and script files are contained.
2. Use “wget” command to pull specific files from Murata GitHub.



We need to set script files as executable afterwards with “chmod a+x” command because “wget” does not maintain the file permissions correctly.

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

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

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

chmod a+x *.sh
```

### 3.2.3 Configure Ubuntu for i.MX Yocto Build

Next step is configuring Ubuntu for Yocto build. Please run Murata’s host setup script (should already be downloaded at this stage): [Host\\_Setup\\_for\\_Yocto.sh](#) . To examine the plain ASCII text version, you can go to [this link](#)  or just hit the “Raw” button. For more information (README-NXP.txt file), just go to the [main folder](#) . The “latest” folder is used to maintain the most recent/up-to-date script.

Murata’s script installs necessary additional packages required for the Yocto build. For additional information, refer to NXP Yocto Project User’s Guide (part of [NXP Reference Documentation](#) ).

Murata’s script will prompt user for password – as supervisory access is needed to install various packages. GIT is also configured so it can be used later during the build process. For more information on first-time GIT setup, you can refer to [this link](#) .

Running the script file is straightforward. Simply invoke at Ubuntu “terminal” prompt (folder location is not important):

```
./Host_Setup_for_Yocto.sh
```

The script goes through the following stages:

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

For an example input/output sequence, refer to [Appendix C](#) .

### 3.2.3.1 Configure Ubuntu for bash

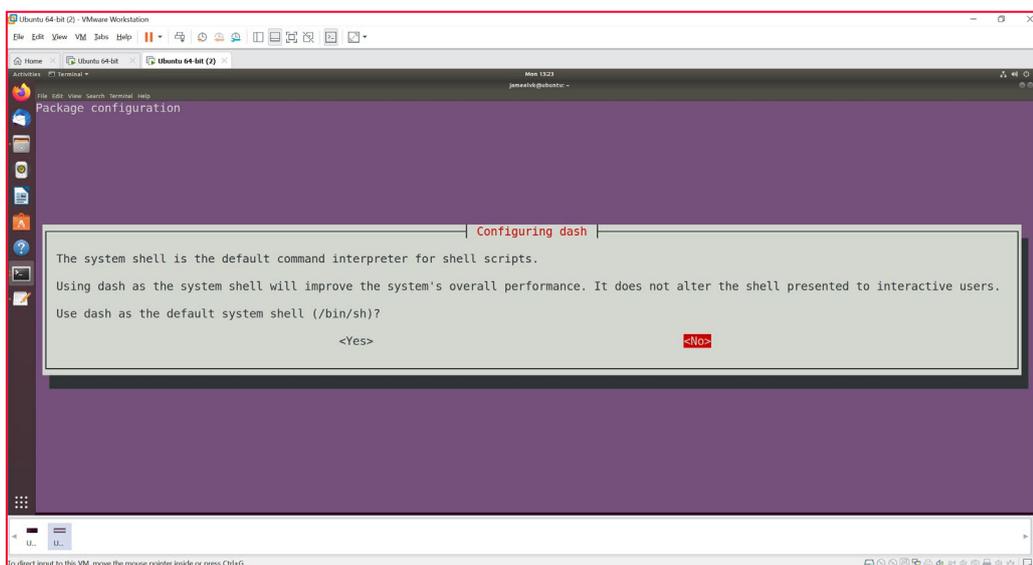
By default, Ubuntu sets the environment to use dash. It is mandatory that, User sets the default system shell to “No” when configuring dash. Follow the steps mentioned below for reconfiguring dash:

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

```
sudo dpkg-reconfigure dash
```

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

**Figure 7: Configuring dash**



## 3.2.4 Murata’s i.MX Yocto Build Script

With Ubuntu installed and configured to build i.MX Yocto, please run the build script (should already be downloaded at this stage): [Murata\\_Wireless\\_Yocto\\_Build\\_NXP.sh](#) [↗](#). For plain ASCII text version, you can go to [this link](#) [↗](#) or just hit the “Raw” button. For more information (README file), just go to the [main folder](#) [↗](#). The “latest” folder is used to maintain the most recent/up-to-date script.

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

- Installed 64-bit version of Ubuntu 20.04 (preferred), 18.04, 16.04, or 14.04.
- Ran Murata’s host setup script in [Section 3.2.3](#) [↗](#) to add necessary packages for Yocto build and configure GIT.
- Created a i.MX BSP folder specific to the desired i.MX Yocto Release. The i.MX Yocto distribution cannot build different versions of Yocto (Linux kernel) in the same folder. Currently the following Yocto releases are supported:
  - 5.10.52\_2.1.0
  - 5.10.72\_2.2.0

- Creating the i.MX BSP folder is straightforward:

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

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



When creating a i.MX BSP folder (\$BSP\_DIR or “murata-imx-bsp” used to reference this all-important folder later in this document), make sure that no parent folder contains a “.repo” folder.

Murata’s build script performs the following tasks:

- Verifies host environment (i.e., Ubuntu 14.04/16.04/18.04/20.04).
- Installs the ‘repo’ tool.
- Check to make sure script being run is the latest version.
- Prompts the user to select release type:
  - “**Stable**” corresponds to “meta-murata-wireless” release/tag (rather than a branch). Murata tests wireless functionality on i.MX platforms for each release/tag. This release type is recommended for baseline image builds or initial bring-up testing.
  - “**Developer**” corresponds to a branch which can be a “moving target”. When performing the automated build, the script file pulls the latest branch contents – as opposed to a specific GIT commit on that branch. If the user wants the latest fixes, then this is the best option to go with.



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

- Select i.MX Yocto release. As already pointed out the current i.MX BSP folder (from which script is being executed) can only support one i.MX Yocto release. If you need to test/evaluate different Yocto/kernel versions, then you must create additional folders.
- Select i.MX target: refer to **Table 3** for more details.
- Select “DISTRO and image”. This configures the graphical driver and Yocto image. For more details refer to the Yocto documentation. It is recommended to go with Murata defaults on this step – Murata has tested/validated with these images.
- Name desired build target folder name. If re-running the Murata build script, this folder name must be unique.
- Review the final configuration and accept before moving forward.
- Accept the NXP End User’s License Agreement (EULA). There is 3rd party IP included in the i.MX Yocto build. This step addresses the sub-licensing issue. During this step, the user must review a fair bit of legal documentation (by repeatedly entering space bar) or if already familiar with the EULA language, enter ‘q’ to bypass displaying the complete agreement. The final step of this EULA step prompts the user to enter “y” to accept.

- Last and final step is to confirm that user wants to kick off the final build process (invoke “bitbake <image>” command).

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

```
./Murata_Wireless_Yocto_Build_NXP.sh
```

The script goes through the following stages:

- Verifying Host Environment
- Install the ‘repo’ tool.
- Verifying Script Version
- Select Release Type:
  - Stable: Murata tested/verified release tag. Stable is the recommended default.
  - Developer: Includes latest fixes on branch. May change at any time.
- Select “Linux Kernel”
- Select Target
- Select DISTRO & Image
- Creation of Build directory
- Verify your selection
- Acceptance of End User License Agreement (EULA)
- Starting Build Now.



Depending on machine type, build may take 1-7 hours to complete.

For an example input/output sequence, refer to [Appendix D](#) .

### 3.3 i.MX Yocto Build: Manual Steps (for Advanced Users)

The user should be very familiar with the NXP Yocto Project User's Guide (part of [NXP Reference Documents release](#) ) . We are only emphasizing some important steps in this section.

Before starting the manual build process, we need to understand which targets, kernel versions, and wireless options are supported. [Section 3.3.1](#)  gives a quick overview of targets, associated supporting kernel versions, and what wireless interconnect is supported.

#### 3.3.1 NXP i.MX versus Murata Module Interconnect

To figure out the Murata wireless modules supported on a given NXP i.MX Platform and kernel version, refer to **Table 7** and **Table 8**.

Refer to **Table 9** for the supported targets: hardware interconnect, DTB file, and interrupt configuration. Refer to the [Linux User Guide](#)  and [Hardware User Manual](#)  for more details on hardware interconnect.

**Table 7** provides an overview of i.MX Reference versus Murata module matrix. An additional column is included to provide a quick NXP chipset lookup. For a given i.MX platform and Murata module, you can quickly look up the compatibility. Providing more details on the terminology used in the table:

- **“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.
- **“uSD-M.2<sup>+</sup>”**: 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.
- **“M.2”**: Only Embedded Artists' Wi-Fi/BT M.2 EVB is required. Then NXP i.MX EVK has a M.2 connector. Currently only WLAN-PCIe, BT-UART, and WLAN/BT CTRL signals are brought out on the M.2 connectors for these specific NXP i.MX EVKs. As such, only Wi-Fi/BT M.2 EVBs which support WLAN-PCIe (i.e., Type 1YM) can be used.
- **“M.2<sup>w</sup>”**: Only Embedded Artists' Wi-Fi/BT M.2 EVB is required. Then NXP i.MX EVK has a M.2 connector. Currently only WLAN-PCIe, and WLAN CTRL signals are brought out on the M.2 connectors for these specific NXP i.MX EVKs. As such, there is no Bluetooth support – only WLAN.
- 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 1YM (WLAN-SDIO/BT-UART) M.2 configuration is referenced as **“1YM-SDIO”**.

Table 7: NXP i.MX EVK Part Number / Murata Module Interconnect

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

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

NXP i.MX EVK Part #	NXP i.MX EVK	MACHINE=target	Kernel 5.10.52	Kernel 5.10.72
MCIMX8QXP-CPU <a href="#">↗</a>	i.MX 8QuadXPlus MEK	imx8qxpmeke	✓	✓
8MPLUSLPD4-EVK <a href="#">↗</a>	i.MX 8MPlus EVK	imx8mp-lpddr4-evk	✓	✓
MCIMX8M-EVKB <a href="#">↗</a>	i.MX 8MQuad EVK	imx8mqevk	✓	✓
8MMINILPD4-EVKB <a href="#">↗</a>	i.MX 8M Mini EVK	imx8mmevk	✓	✓
8MNANOD4-EVK <a href="#">↗</a>	i.MX 8M Nano EVK	imx8mnddr4evk	✓	✓
MCIMX6UL-EVKB <a href="#">↗</a>	i.MX 6UL EVK	imx6ulevk	✓	✓
MCIMX6ULL-EVK <a href="#">↗</a>	i.MX 6ULL EVK	imx6ull14x14evk	✓	✓

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

Target (MACHINE)	Hardware Config	i.MX DTB File	Interrupt Config
imx8qxpmeke	M.2 <sup>[2]</sup>	imx8qxp-mek.dtb	N/A
imx8mp-lpddr4-evk	M.2 <sup>[2]</sup>	imx8mp-evk-usdhc1-m2.dtb	SDIO
imx8mp-lpddr4-evk	M.2 <sup>[2]</sup>	imx8mp-evk.dtb	N/A
imx8mqevk	M.2 <sup>[2]</sup>	imx8mq-evk-pcie1-m2.dtb	N/A
imx8mqevk	uSD-M.2 <sup>[3]</sup>	imx8mq-evk.dtb	SDIO
imx8mmevk	M.2 <sup>W</sup> <sup>[5]</sup>	imx8mm-evk.dtb	N/A
imx8mmevk	uSD-M.2 <sup>+</sup> <sup>[4]</sup>	imx8mm-evk.dtb	SDIO
imx8mnddr4evk	uSD-M.2 <sup>+</sup> <sup>[4]</sup>	imx8mn-evk.dtb	SDIO
imx6ulevk	uSD-M.2 <sup>[3]</sup>	imx6ul-14x14-evk-btwifi.dtb <sup>[6]</sup>	SDIO
imx6ull14x14evk	uSD-M.2 <sup>[3]</sup>	imx6ull-14x14-evk-btwifi.dtb <sup>[6]</sup>	SDIO

<sup>1</sup> No Connection options available

<sup>2</sup> Wi-Fi/BT M.2 EVB plugs directly into M.2 connector

<sup>3</sup> Works with uSD-M.2 Adapter configured for 1.8V default

<sup>4</sup> Works with uSD-M.2 Adapter (Rev B1) with additional cabling

<sup>5</sup> Wi-Fi/BT M.2 EVB plugs directly into M.2 connector; but only WLAN is functional

<sup>6</sup> Custom DTB file for 6UL(L) configured for 1.8V

### 3.3.2 Host PC Preparation

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



Murata has verified these build steps using Ubuntu 18.04 (x64) and 20.04 (x64).

Next step is configuring Ubuntu for Yocto build. Refer to [Yocto Project User's Guide](#) (Section 3 "Host Setup"). When following NXP's Yocto Project User's Guide, make sure that GIT is setup properly with the commands below:

```
git config --global user.name "Your Name"
git config --global user.email "Your Email"
git config --list
```

### 3.3.3 Yocto Project Setup

The NXP Yocto Project BSP Release directory contains a "sources" directory, which contains the recipes used to build, one or more build directories, and a set of scripts used to set up the environment. The recipes used to build the project come from both the community and NXP. The Yocto Project layers are downloaded to the source directory. This sets up the recipes that are used to build the project. The following example shows how to download the NXP Yocto Project Community BSP recipe layers. Rather than use a specific folder name, we use the variable "BSP\_DIR" name to represent the base directory for Yocto Build.

Create a build directory and setup "BSP\_DIR":

```
cd ~
mkdir murata-imx-bsp
cd murata-imx-bsp
export BSP_DIR=`pwd`
```

Now we are ready to download the NXP Yocto Project Community BSP recipe layers.

In this example, we are targeting Linux 5.10.52\_2.1.0 release. Execute the following repo commands:

```
repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-
linux-hardknott -m imx-5.10.52-2.1.0.xml
repo sync
```

Once "repo sync" is completed, the source code is checked out into the directory "\$BSP\_DIR/sources". You can perform repo synchronization, with the command "repo sync", periodically to update to the latest code. If errors occur during repo initialization, try deleting the ".repo" directory and running the repo initialization command again. In [Section 3.3.4](#), we add the customized "meta-murata-wireless" to "\$BSP\_DIR/sources".

To setup build directory for Linux 5.10.52\_2.1.0, the following command syntax is used:

```
cd $BSP_DIR
DISTRO=<distro name> MACHINE=<machine name> source imx-setup-release.sh -b
<build dir>
```

One example is to build fb frame buffer end for i.MX 6UltraLite EVK:

```
cd $BSP_DIR
DISTRO=fsl-imx-fb MACHINE=imx6ulevk source imx-setup-release.sh -b build-
imx6ulevk-fb
```



When specifying the build directory (-b <build dir>), it is better to specify a unique folder name (in this example “build-imx6ulevk-fb”). Otherwise, the next time you configure a Yocto build from the same folder (i.e., invoke “imx-setup-release.sh” script from “BSP\_DIR”, it will delete your previous build.

After invoking the “imx-release-setup.sh” script, the EULA (End User License Agreement) will be presented to the user for agreement. Press “space” bar repeatedly (or “q” to skip reading entire EULA) until you reach the bottom of the agreement. Enter ‘y’ to accept the EULA agreement:

```
2.3. For NXP Licensed Software provided to you in source code
Do you accept the EULA you just read? (y/n) y
EULA has been accepted.
```

In this example, the final expected output is (after accepting license agreement):

```
Your build environment has been configured with:
MACHINE=imx6ulevk
SDKMACHINE=i686
DISTRO=fsl-imx-fb
EULA=
BSPDIR=
BUILD_DIR=.
meta-freescale directory found
<username>@ubuntu:~/murata-imx-bsp/build-imx6ulevk-fb$
```

To make code examples easier going forward, let’s create a variable for the build directory:

```
# In this example the present directory is "~/murata-imx-bsp/build-imx6ulevk-
fb"
$ export BUILD_DIR=`pwd`
```

### 3.3.4 Fetch “meta-murata-wireless” from GitHub and copy into “Sources”

Clone “meta-murata-wireless” git and checkout “imx-hardknott-5-10-52\_r1.0” release (Linux 5.10.52):

```
cd $BSP_DIR/sources
git clone https://github.com/murata-wireless/meta-murata-wireless.git
cd meta-murata-wireless
git checkout imx-hardknott-5-10-52_r1.0
```

“meta-murata-wireless” contains the following (refer to [Section 2.3.2](#)  for more details):

- “add-murata-layer-script” folder
- “conf” folder
- “freescale” folder

- “recipes-connectivity” folder
- “recipes-kernel” folder
- files (COPYING.MIT, README, README.md)

### 3.3.5 Install Necessary “hooks” for “meta-murata-wireless”

To enable “meta-murata-wireless” into the selected i.MX target Yocto build, certain Yocto configuration files must be modified. Murata provides “add-murata-wireless.sh” script file:

```
cd $BSP_DIR
sh ./sources/meta-murata-wireless/add-murata-layer-script/add-murata-
wireless.sh build-imx6ulevk-fb
```



“build-imx6ulevk-fb” is just the build folder name and not the directory.

If successful, the output will look like:

```
Welcome to Freescale Community BSP

The Yocto Project has extensive documentation about OE including a
reference manual which can be found at:
  http://yoctoproject.org/documentation

For more information about OpenEmbedded see their website:
  http://www.openembedded.org/

You can now run 'bitbake <target>'

Common targets are:
  core-image-minimal
  meta-toolchain
  meta-toolchain-sdk
  adt-installer
  meta-ide-support

# The following is a misleading statement7.
Your configuration files at build-imx6ulevk-fb have not been touched.

# The following is an additional Murata logging to clarify file modifications.
CORRECTION: Murata modified the following files
- bblayers.conf present in <BUILD_DIR>/conf
- local.conf present in <BUILD_DIR>/conf
```

Murata-Wireless setup complete. Create an image with:

```
# No errors logged and here is the bitbake command
$ bitbake fsl-image-validation-imx
```

<sup>7</sup> “\$BSP\_DIR/sources/meta-fsl-bsp-release/imx/tools/fsl-setup-release.sh” generates the “have not been touched” message. Subsequently the “add-murata-wireless.sh” script modifies the “bblayer.conf” and “local.conf”.

To double check on correct script execution, you can check the content of two files in “\$BUILD\_DIR/conf” folder: “bblayers.conf” and “local.conf”. Verify last line in “\$BUILD\_DIR/conf/bblayers.conf” is:

```
BBLAYERS += " ${BSPDIR}/sources/meta-murata-wireless "
```

Verify the last two lines in “\$BUILD\_DIR/conf/local.conf” are:

```
CORE_IMAGE_EXTRA_INSTALL += " hostapd murata-binaries iperf3 "
CORE_IMAGE_EXTRA_INSTALL += " bluez5 bluez5-noinst-tools bluez5-obex openobex
obexftp glibc-gconv-utf-16 glibc-utils cyw-supPLICANT python3"
```

### 3.3.6 Build Murata-Customized Yocto Image for Specific i.MX Target

Now that the “meta-murata-wireless” layer is “hooked” into the i.MX BSP Yocto build, we can invoke “bitbake” to build the default (Murata-verified) “fsl-image-validation-imx” image:

```
cd $BUILD_DIR
bitbake fsl-image-validation-imx
```

Things to watch out for:

- If any necessary source file cannot be fetched, the build process terminates. In such a scenario, re-execute the command “bitbake fsl-image-validation-imx” again to see if the problem resolves itself. If the problem persists, report the error to whoever maintains the failing git repository.
- The “bitbake” step is expected to take at least 1-7 hours to complete. This build process depends heavily on processor speed, RAM, hard drive access (i.e., SSD is optimal), and internet download speeds.
- Ensure that there is a minimum of 50 GB free disk space (80 GB needed for i.MX8 build).

## 3.4 i.MX Yocto Build: Manual Steps “Take 2” (Quick Recap)

This section is for users already familiar with the “meta-murata-wireless” implementation. It provides a condensed (i.e., repeat) version of [Section 3.3](#). The following example sequence shows all the necessary steps to build the Murata customized i.MX image for the following configuration:

- i.MX6UL or i.MX6ULL EVK at 1.8V VIO (both WLAN and BT interfaces).
- Murata uSD-M.2 Adapter is used, install J12 in 2-3 position for 3.3V setting, install J12 in 1-2 position for 1.8V setting.
- WLAN interface is initialized during kernel boot. To bring up interface just invoke “ifconfig wlan0 up” or configure WPA supplicant, Hostapd, etc.
- Bluetooth UART is connected and is initialized by running “hciattach” command.
- Bluetooth PCM connection is not supported.

### 3.4.1 Initialize Linux i.MX Yocto Default Build Environment

```
cd ~
mkdir murata-imx-bsp
```

```
cd murata-imx-bsp
export BSP_DIR=`pwd`
repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-
linux-hardknott -m imx-5.10.52-2.1.0.xml
repo sync
```

### 3.4.2 Configuring the i.MX Target

```
# In this example BSP_DIR is "~/murata-imx-bsp"
cd $BSP_DIR
DISTRO=fsl-imx-fb MACHINE=imx6ulevk source ./imx-setup-release.sh -b build-
imx6ulevk-fb
```

This will bring the EULA (End User License Agreement). Press “space” bar repeatedly (or “q” to skip reading entire EULA) until you reach the bottom of the agreement. Enter ‘y’ to accept the EULA agreement:

```
2.3.      For NXP Licensed Software provided to you in source code
Do you accept the EULA you just read? (y/n) y
EULA has been accepted.
```

### 3.4.3 Add “meta-murata-wireless” Layer and Kick off the Build

```
# In this example, current directory is "~/murata-imx-bsp/build-imx6ulevk-fb"
export BUILD_DIR=`pwd`
cd $BSP_DIR/sources
git clone https://github.com/murata-wireless/meta-murata-wireless.git
cd meta-murata-wireless
git checkout imx-hardknott-5-10-52_r1.0
cd $BSP_DIR
sh ./sources/meta-murata-wireless/add-murata-layer-script/add-murata-
wireless.sh build-imx6ulevk-fb
```

```
cd $BUILD_DIR
bitbake fsl-image-validation-imx
```

## 3.5 i.MX Yocto Build: Manual Steps “Take 2” for “5.10.72”

```
cd ~
mkdir murata-imx-bsp
cd murata-imx-bsp
export BSP_DIR=`pwd`
repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-
linux-hardknott -m imx-5.10.72-2.2.0.xml
repo sync

cd $BSP_DIR
DISTRO=fsl-imx-wayland MACHINE=imx8mqevk source ./imx-setup-release.sh -b
build-imx8mqevk-wayland
export BUILD_DIR=`pwd`
cd $BSP_DIR/sources
git clone https://github.com/murata-wireless/meta-murata-wireless.git
cd meta-murata-wireless
git checkout imx-hardknott-5-10-72_r1.0
```

```
cd $BSP_DIR
sh ./sources/meta-murata-wireless/add-murata-layer-script/add-murata-
wireless.sh build-imx8mqevk-wayland

cd $BUILD_DIR
bitbake fsl-image-validation-imx
```

## 4 Using NXP driver

This section is intended to provide the user with a quick “how to” when using NXP driver for the first time.

### 4.1 How-To on NXP driver

Here is a quick overview on expected output when platform is correctly configured for Murata-NXP WLAN solution. First off, let’s go through a checklist:

- Murata-customized image correctly built and (micro) SD card flashed. Even without testing WLAN/BT, make sure your finished image allows the i.MX platform to boot correctly (use default i.MX DTB file).
- i.MX hardware correctly setup: i.e., for 1.8V VIO signaling image, make sure the uSD-M.2 Adapter is correctly configured.
- Select the correct DTB file by interrupting bootloader sequence. Refer to **Table 9** for selecting the right DTB (Device Tree Blob) file.

#### 4.1.1 Using Switch Modules Script File

The steps to load the correct drivers for the M.2 board are provided below.

- Boot into Linux by entering the command, “boot”
- Login in as “root” at the prompt.
- Run the script with the module name as parameter

```
switch_module.sh <module_name>
# Where <module-name> can be lzm, lym-sdio, lym-pcie, lzk, lxl or 2ds.
```

```
switch_module.sh lzm
reboot
```

- Look for the following message after entering switch\_module.sh <module\_name> command.

```
Setting up for LZM (NXP - SDIO)
```

- Reboot by entering the command, “reboot”
- M.2 board should now have been detected and the correct kernel modules gets loaded.

#### 4.1.2 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, the WLAN

device is probed over the SDIO bus. As an example, we will bring up Wi-Fi when using following configuration:

- i.MX 6ULL EVK
- uSD-M.2 Adapter configured for 1.8V VIO (J12 jumper is set in position 1-2)
- Type 1ZM (88W8987) EVB

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.

### 4.1.3 “mLAN0” initialization

After the kernel boots, the “mLAN0” interface can be initialized by invoking the “ifconfig mLAN0 up” command. Example output shown below:

```
$ ifconfig mLAN0 up
$ ifconfig mLAN0
# "mLAN0" interface is UP. WLAN MAC Address is shown
mLAN0      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)
```

## 4.2 Using manual driver load

Following are the steps for manual loading of NXP drivers for 1ZM, 1YM-SDIO and 1YM-PCIe.

### 4.2.1 1ZM – Manual load

After setting the correct DTB file for the corresponding platform, Enter the following commands at the prompt.

```
insmod /lib/modules/5.4.47-2.2.0+g5ec03d0/kernel/net/wireless/cfg80211.ko
modprobe moal mod_para=nxp/wifi_mod_para_sd8987.conf
ifconfig mLAN0 up
```

### 4.2.2 1YM-SDIO Manual load

```
insmod /lib/modules/5.4.47-2.2.0+g5ec03d0/kernel/net/wireless/cfg80211.ko
cd /usr/share/nxp_wireless/bin_sd8997
insmod ./mLAN.ko
insmod sd8997.ko fw_name=nxp/sdsd8997_combo_v4.bin
cal_data_cfg=nxp/WlanCalData_ext_DB_W8997_1YM_ES2_Rev_C.conf cfg80211_wext=0xf
ifconfig mLAN0 up
```

### 4.2.3 1YM-PCIe Manual load

```
insmod /lib/modules/5.4.47-2.2.0+g5ec03d0/kernel/net/wireless/cfg80211.ko
cd /usr/share/nxp_wireless/bin_pcie8997
insmod ./mLAN.ko
```

```
insmod ./pcie8997.ko drv_mode=3 ps_mode=2 auto_ds=2 cfg80211_wext=0xf
fw_name=nxp/pcieuart8997_combo_v4.bin
cal_data_cfg=nxp/WlanCalData_ext_DB_W8997_1YM_ES2_Rev_C.conf

ifconfig wlan0 up
```

## 4.3 Bluetooth Interface (Standard BT-UART)

Before initializing the Bluetooth interface, the WLAN interface (wlan0) 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
```

For more specifics on controlling the Bluetooth interface, refer to the [Linux User Guide](#). Here is an example output using i.MX 6ULL EVK with Type 1ZM module:

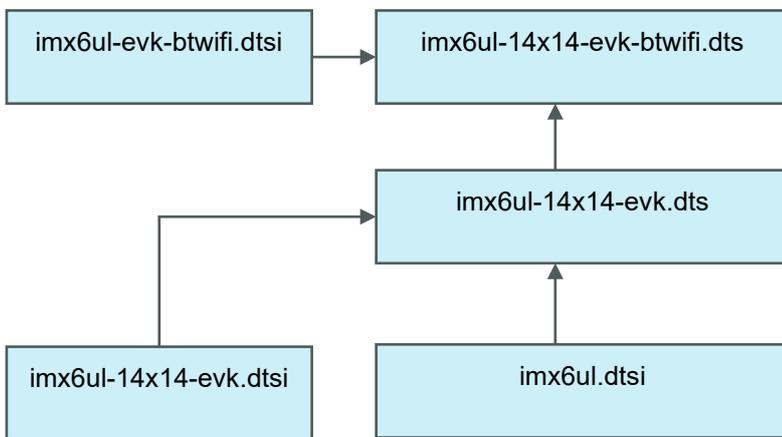
```
$ hciattach /dev/ttymx1 any 115200 flow
Setting TTY to N_HCI line discipline
Device setup complete
$ hciconfig hci0 up
$ hcitool -i hci0 cmd 0x3f 0x0009 0xc0 0xc6 0x2d 0x00
< HCI Command: ogf 0x3f, ocf 0x0009, plen 4
  C0 C6 2D 00
> HCI Event: 0x0e plen 4
  01 09 FC 00
$ killall hciattach
# Ignore the following error
[ 1669.277042] Bluetooth: hci0: sending frame failed (-49)
$ hciattach /dev/ttymx1 any -s 3000000 3000000 flow
Setting TTY to N_HCI line discipline
Device setup complete
$ hciconfig hci0 up
$ hciconfig hci0 piscan
$ hciconfig hci0 noencrypt
$ hciconfig -a
$ hcitool scan
Scanning ...
        34:F3:9A:A6:00:53          SCOTTK-HPZBOOK
```

## 5 Hierarchy of device tree source files for i.MX6/8 platforms

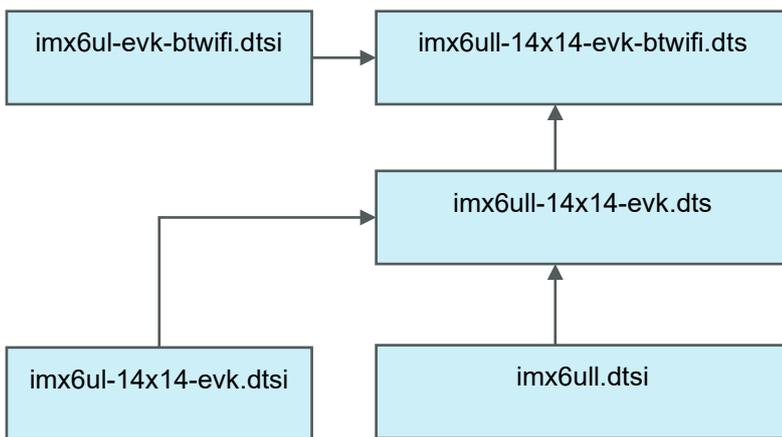
Device tree source files is a set of text files in the Linux kernel that describe the hardware of certain platform. For 32-bit platforms, they are located in /arch/arm/boot/dts/. For 64-bit platforms, they are located in /arch/arm64/boot/dts/freescale/. DTS files come with two extensions, .dtsi and .dts. \*.dtsi files are include files for device tree source. \*.dts files are device tree source files. They can be used together to describe a target platform.

The following figures shows how these DTS files are organized in the hierarchy structures.

**Figure 8: DTS hierarchy for imx6ulevk**



**Figure 9: DTS hierarchy for imx6ull14x14evk**

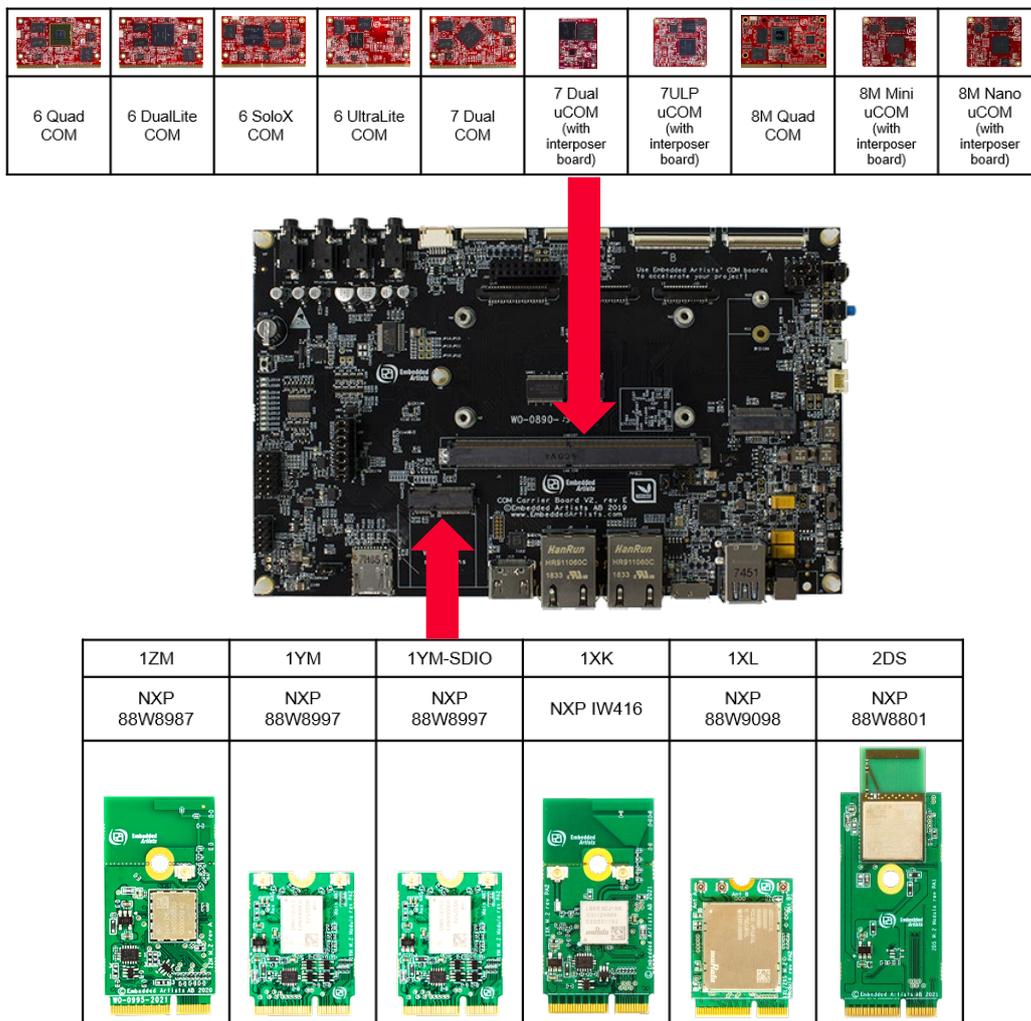


## 6 Embedded Artists' Solution

Murata has partnered with Embedded Artists to provide an easier solution for evaluating Wi-Fi/BT IOT modules. This solution is composed of three parts: Carrier board, Computer on Module (COM) board, and M.2 Evaluation boards (EVB). **Figure 12** shows that the carrier board can work with a variety of NXP i.MX6/7/8 COM boards and two different Murata EVBs (1YM configured for both WLAN-PCIe and WLAN-SDIO). With this platform, users can more easily evaluate i.MX processors against Wi-Fi/BT M.2 EVBs to find the best combination for their product. Also, Embedded Artists brings out all the test points you need for trouble shooting. With this platform, no adapter/interconnection is needed, therefore you can achieve the real potential of Murata's Wi-Fi/BT modules.

**Figure 10** shows how this platform works with COM boards and M.2 Wi-Fi/BT EVBs.

**Figure 10: Combine i.MX COM with Wi-Fi/BT M.2 EVB**



**Table 10** provides an i.MX Reference Platform versus Murata module matrix. An additional column is included to provide a quick NXP chipset lookup. As the table clearly shows, the Embedded Artist's solution provides very comprehensive interconnect options.

As we will also see in [Section 6.1](#), Embedded Artists provides not just a better hardware solution but an enhanced software solution as well. The software is well documented and easy to use.

Table 10: Embedded Artists' i.MX Interconnect

EA i.MX (u)COM	1ZM	1YM	1XK	1XL	2DS
	NXP 88W8987	NXP 88W8997	NXP 88W8997	NXP 88W9098	NXP 88W8801
<a href="#">iMX8M Quad COM</a>	M.2 <sup>[8]</sup>				
<a href="#">iMX8M Mini uCOM V2</a>	M.2 <sup>[8]</sup>				
<a href="#">iMX8M Mini uCOM V3</a>	M.2 <sup>[8]</sup>				
<a href="#">iMX8M Nano uCOM</a>	M.2 <sup>[8]</sup>				
<a href="#">iMX7 Dual COM</a>	M.2 <sup>[8]</sup>				
<a href="#">iMX7 Dual uCOM</a>	M.2 <sup>[8]</sup>				
<a href="#">iMX7ULP uCOM</a>	M.2 <sup>[8]</sup>				
<a href="#">iMX6 Quad COM</a>	NC <sup>[9]</sup>	M.2 <sup>[8]</sup>	M.2 <sup>[8]</sup>	M.2 <sup>[8]</sup>	M.2 <sup>[8]</sup>
<a href="#">iMX6 DualLite COM</a>	NC <sup>[9]</sup>	M.2 <sup>[8]</sup>	M.2 <sup>[8]</sup>	M.2 <sup>[8]</sup>	M.2 <sup>[8]</sup>
<a href="#">iMX6 SoloX COM</a>	NC <sup>[9]</sup>	M.2 <sup>[8]</sup>	M.2 <sup>[8]</sup>	M.2 <sup>[8]</sup>	M.2 <sup>[8]</sup>
<a href="#">iMX6 UltraLite COM</a>	M.2 <sup>[8]</sup>				

## 6.1 Build Steps

Please refer to Embedded artists document, [Getting-Started-with-M2-modules-and-iMX6\\_7\\_8-from-Linux-v5.10](#), Section 5 for detailed steps on building images.

### 6.1.1 Create Build Directory

Create a directory for the downloaded files (ea-bsp in the example below):

```
mkdir ea-bsp
cd ea-bsp
```

### 6.1.2 Initialize repo

Enter the command, “repo init” and “repo sync” for downloading the files:

```
repo init -u https://github.com/embeddedartists/ea-yocto-base -b <selected
branch>
repo sync
```

Refer to **Table 11** for current branches supporting Murata’s modules based on NXP chipsets.

Table 11: Embedded Artists' supported branches

GitHub Repository	Branches	Description
ea-yocto-base	ea-5.10.72	Linux image build
linux-imx	ea-5.10.72	Linux Kernel
uboot-imx	ea_v2020.04	Bootloader

<sup>8</sup> Works with onboard M.2 slot

<sup>9</sup> No Connection option (due to 1ZM only supporting 1.8V SDIO VIO)

### 6.1.3 Select Machine Configuration and Distribution

A machine configuration must be specified before a build can be started. The **Table 12** below contains the machine configurations available for Embedded Artists boards. It is also possible to find the configuration files in the directory `~/ea-bsp/sources/meta-ea/conf/machine`.

**Table 12: Embedded Artists' Machine Configuration**

i.MX EVK	Machine
<a href="#">iMX8M Quad COM</a>	imx8mqea-com
<a href="#">iMX8M Mini uCOM</a>	imx8mmea-ucom
<a href="#">iMX8M Nano uCOM</a>	imx8mnea-ucom
<a href="#">iMX7 Dual COM</a>	imx7dea-com
<a href="#">iMX7 Dual uCOM</a>	imx7dea-ucom
<a href="#">iMX7ULP uCOM</a>	imx7ulpea-ucom
<a href="#">iMX6 Quad COM</a>	imx6qea-com
<a href="#">iMX6 DualLite COM</a>	imx6dlea-com
<a href="#">iMX6 SoloX COM</a>	imx6sxea-com
<a href="#">iMX6 UltraLite COM</a>	imx6ulea-com

### 6.1.4 Initialize Build

Before starting the build, it must be initialized. In this step the build directory and local configuration files are created. A distribution must be selected when initializing the build, see [iMX Working with Yocto](#) for different alternatives. For headless setups (i.e., without a display) use `fsl-imx-fb`. Note that the i.MX8 boards do not support `fsl-imx-fb` so use `fsl-imx-wayland` instead.

In the example below the machine `imx6sxea-com`, the build directory `build_dir` and the `fsl-imx-fb` distribution (see [iMX Working with Yocto](#) for other distributions) is selected.

```
DISTRO=fsl-imx-fb MACHINE=imx6sxea-com source ea-setup-release.sh -b build_dir
```

### 6.1.5 Start Build

Invoke the command below to start the build. Please note that depending on the capabilities of your host computer building an image can take many hours.

```
bitbake ea-image-base
```

### 6.1.6 Deploy Image

Please refer to Embedded Artists document, [Getting-Started-with-M2-modules-and-iMX6\\_7\\_8-from-Linux-v5.10](#), Section 5.3 for deploying images.

## 6.2 Documentation

Embedded Artists also provide every document you need on their website. **Table 13** provides the list of documents you might need during the evaluation process.

**Table 13: Embedded Artists' Landing Pages**

Landing Pages	Notes
<a href="#">Embedded Artists' Website</a>	The Art of Embedded Systems Development – made EASY™
<a href="#">i.MX 6/7/8 COM Boards</a>	Listing of Computer-on-Module boards.
<a href="#">i.MX 6/7/8 COM Carrier Board V2</a>	Main baseboard which all the COM boards plug into.
<a href="#">Getting Started with i.MX 6/7/8 Developer's Kit V2/V3</a>	How to bring up i.MX 6/7/8 Dev Kit (V2/V3).
<a href="#">M.2 Module Family</a>	Top level listing of 1ZM, 1YM, 1XL, 1XK, 2DS M.2 EVBs.
<a href="#">Application Development on an i.MX Developer's Kit</a>	Description of C/C++, Python, Node.js, and Qt5 development.
<a href="#">Devices and Peripherals on an i.MX Kit</a>	Description of how to work with peripherals and devices.

**Table 14** includes the links to the datasheets and schematics of COM boards and the EVBs.

**Table 14: Embedded Artists' Datasheets and Schematics**

Datasheets and Schematics	Notes
<a href="#">i.MX 6/7/8 COM Carrier Board V2 Datasheet</a>	Comprehensive definition of COM Carrier (baseboard).
<a href="#">i.MX 6/7/8 COM Carrier Board V2 Schematics</a>	Complete schematics including clear definition of uSD-M.2 Adapter.
<a href="#">i.MX 8 COM Carrier Board V3 Datasheet</a>	Comprehensive definition of COM Carrier (baseboard).
<a href="#">i.MX 8 COM Carrier Board V3 Schematics</a>	Complete schematics including clear definition of uSD-M.2 Adapter.
<a href="#">M.2 SDIO Interface Schematic</a>	Reference schematic for customers designing in WLAN-SDIO M.2 EVB.
<a href="#">M.2 PCIe Interface Schematic</a>	Reference schematic for customers designing in WLAN-PCIe M.2 EVB.
<a href="#">EACOM Board Specification Guide</a>	Comprehensive definition of Embedded Artists' Computer-On-Module's.
<a href="#">1ZM M.2 Module Datasheet</a>	Comprehensive details on 1ZM Wi-Fi/BT M.2 Module.
<a href="#">1YM M.2 Module Datasheet</a>	Comprehensive details on 1YM Wi-Fi/BT M.2 Module.
<a href="#">1XK M.2 Module Datasheet</a>	Comprehensive details on 1XK Wi-Fi/BT M.2 Module.
<a href="#">1XL M.2 Module Datasheet</a>	Comprehensive details on 1XL Wi-Fi/BT M.2 Module.
<a href="#">2DS M.2 Module Datasheet</a>	Comprehensive details on 2DS Wi-Fi M.2 Module.

**Table 15** provides the link to the required manual documents and the software.

**Table 15: Embedded Artists' User Manuals and Software**

User Manuals and Software	Notes
<a href="#">Getting Started with M.2 Modules and i.MX 6/7/8</a>	Comprehensive document covering all major topics associated with using Wi-Fi/BT M.2 EVBs on EA's i.MX 6/7/8 Dev Kits.
<a href="#">i.MX Working with Yocto</a>	Comprehensive guide on building Linux images using Yocto framework.

<a href="#">Linux i.MX Images Download</a> 	Pre-compiled images using “uuu” tool: allows users to easily flash i.MX platforms with latest Linux images with integrated Wi-Fi/BT support.
<a href="#">Wi-Fi/BT M.2 EVB Primer</a> 	Introduction and drill-down on M.2 interface.

## 7 Murata’s Regulatory Solution for NXP modules

### 7.1 Description of Regulatory Solution for WLAN/BT

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

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

**BT:** Bluetooth Tx power is configured by using hcitool / “bt\_power\_config\_1.sh” script file.

### 7.2 Tree view of the files list for Linux Patched solution

NXP\_Linux\_Patched\_Solution\_for\_5.10.\*\*\*/

```

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

```

```

| | | | └── txpower_US.bin
| | | | └─ 1ZM
| | | | | └── db.txt
| | | | | └── ed_mac.bin
| | | | | └── regulatory.bin
| | | | | └── txpower_CA.bin
| | | | | └── txpower_EU.bin
| | | | | └── txpower_JP.bin
| | | | | └── txpower_US.bin
| | | | └─ 2DS
| | | | | └── db.txt
| | | | | └── ed_mac.bin
| | | | | └── regulatory.bin
| | | | | └── txpower_CA.bin
| | | | | └── txpower_EU.bin
| | | | | └── txpower_JP.bin
| | | | | └── txpower_US.bin
| | | | └─ 32_bit
| | | | | └── crda
| | | | | └── libreg.so
| | | | | └── regdbdump
| | | | └─ 64_bit
| | | | | └── crda
| | | | | └── libreg.so
| | | | | └── regdbdump
| | | |
| | | | └─ bt_power_config_1.sh
| | | | └─ regulatory.rules
| | | | └─ wifi_mod_para_murata.conf
| | |
| | └── README.txt
| └── switch_regions.sh

```

```

|
├── patch_files
│   ├── imx6ul-14x14-evk-btwifi.dtb
│   ├── imx6ul-14x14-evk-btwifi.dts
│   ├── imx6ull-14x14-evk-btwifi.dtb
│   └── imx6ull-14x14-evk-btwifi.dts

```

## 7.2.1 Description of files

**Table 16: Murata regulatory files**

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

## 7.3 Purpose of switch\_regions.sh

The switch\_regions.sh script file allows the user to easily switch the regulatory region. To use the script, copy the folder, "NXP\_Linux\_Patched\_Solution\_for\_5.10.\*/murata" from the [Murata patched solution](#) to "/lib/firmware/nxp" in the root file system. This is only required for the first time.

Execute the script file, "switch\_regions.sh" for deploying regulatory solution.

Ex: switch\_regions.sh <Module\_name> <Country\_code>

The following options are supported.

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

### 7.3.1 Sample Description of structure SD8987 from “wifi\_mod\_para.conf” configured for US with 1ZM module

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

### 7.3.2 Sample Description of structure PCIE8997 from “wifi\_mod\_para.conf” configured for EU with 1YM module

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



ed\_mac.bin is only applicable for EU country code.

## 7.4 Sample log of switch\_regions.sh: (Ex: 1YM-SDIO@8M-MINI)

For country code CA (Canada):

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

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

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

For country code US (United States):

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

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

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

## 8 Acronyms

Acronym	Meaning
1YM	AKA "1YM-PCIe", indicates that M.2 EVB is strapped for WLAN-PCIe/BT-UART
1YM-SDIO	Type 1YM M.2 EVB strapped for WLAN-SDIO/BT-UART
1YM-SDIO*	Type 1YM M.2 EVB strapped for WLAN-SDIO/BT-SDIO
AP	Access Point
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BSP	Board Support Package
BT	Bluetooth
CE	Conformité Européenne
CLI	Command Line Interface
CLK	Clock
CMD	Command
COM	Computer on Module
CPU	Central Processing Unit
CRDA	Central Regulatory Domain Agent
CTRL	Control
CTS	Clear to Send
CYW	Cypress, now Infineon
DHCP	Dynamic Host Configuration Protocol
DIP	Dual In-line Package
DTB	Device Tree Blob: Kernel reads in at boot time for configuration.
DTS	Device Tree Source
EA	Embedded Artists designs, manufactures and distributes current <a href="#">Wi-Fi/BT M.2 EVBs</a> . EA also have enhanced i.MX developer kits which provide comprehensive support for <a href="#">Murata modules</a> .
eMMC	Embedded Multi-Media Controller: integrated flash memory and controller on single die.
EULA	End User License Agreement
EVB	Evaluation Board (Embedded Artists' Wi-Fi/BT module)
EVK	Evaluation Kit (includes EVB + Adapter)
EVKB	Evaluation Kit Board
FCC	Federal Communications Commission
FFC	Flat Flex Cable
FW	Firmware
GIT	Global Information Tracker
GND	Ground
GPIO	General Purpose Input/Output
HCI	Host Controller Interface
I2S	Inter-IC Sound
IC	Industry Canada
IFX	Infineon
IRQ	Interrupt Request Line
LED	Light Emitting Diode
MAC	Medium Access Control
MEK	Multisensory Enablement Kit
MIMO	Multiple Input Multiple Output
NVRAM	Non-Volatile Random-Access Memory
OOB	Out of Band
O/S	Operation System
P2P	Peer-to-Peer
PC	Personal Computer
PCB	Printed Circuit Board
PCIe	PCI Express
PCM	Pulse Code Modulation
PSK	Pre-Shared Key
RF	Radio Frequency

RSSI	Received Signal Strength Indicator
RTS	Request to Send
RX	Receive
SABRE	Smart Application Blueprint for Rapid Engineering
SD	Secure Digital
SDB	SABRE Development Board
SDIO	Secure Digital Input Output
SSID	Service Set Identifier
STA	Station
SW	Software
SYNC	Synchronization
TELEC	Telecom Engineering Center
TX	Transmit
UART	Universal Asynchronous Receiver/Transmitter
UHS	Ultra-High Speed
UI	User Interface
USB	Universal Serial Bus
uSD	Micro SD
uSD-M.2	Micro SD to M.2 Adapter
VBAT	Voltage of the Battery
VIO	Input Offset Voltage
VMware	Virtual Machine Software
Wi-Fi	Wireless LAN: "Wi-Fi" is a registered trademark of Wi-Fi Alliance
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access

## 9 Technical Support Contact

**Table 17** lists all the support resources available for the Murata Wi-Fi/BT solution.

**Table 17: List of Support Resources**

Support Site	Notes
<a href="#">Murata Community Forum</a> 	<b>Primary support point for technical queries.</b> This is an open forum for all customers. Registration is required.
<a href="#">Murata i.MX Landing Page</a> 	No login credentials required. Murata documentation covering hardware, software, testing, etc. is provided here.
<a href="#">Murata uSD-M.2 Adapter Landing Page</a> 	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.
<a href="#">Murata Module Landing Page</a> 	No login credentials required. Murata documentation covering all NXP-based Wi-Fi/BT modules is provided here.

## 10 Appendix A: Useful “git” commands

Git is a free and open-source distributed version control system. It is important for the user to have a reasonable understanding of Git because Murata’s implementation relies heavily on [GitHub](#): a remote Git repository. Some people may be new to using “git” commands or to remote Git repositories such as GitHub. Here is a [good primer](#) on using “git”. For more details, refer to the [main Git page](#). Commonly used “git” commands are shown in **Table 18**.

**Table 18: Useful “git” commands:**

“git” command	Description
git config --list	List Git identity: username and email address.
git config --global --unset-all user.name	Remove Git username setting.
git config --global --unset-all user.email	Remove Git user email address setting.
git config --global user.name “<User Name>”	Configure Git username. Is done as part of first-time Git setup. Example: git config --global user.name “Fred Jones”
git config --global user.email “<User Email>”	Configure Git user email address. Is done as part of first-time Git setup. Example: git config --global user.email “fj@gmail.com”
git clone <remote_URL>	Creates a local working copy of an existing remote repository.
git branch -a	Lists all available branches of current local repository.
git tag	Lists all the available release tags.
git checkout <branch_name>	Check out a specific branch or release/tag in local repository.
git status	Lists the current branch (which is checked out) in local repository.
git reset --hard	Reset current Git repository content to default branch settings. NOTE: there are two “-” characters before “hard” switch.
git log	Shows the chronological commit history for local repository.
git clean -fd	Will remove all untracked directories and the files within them. It will start at the current working directory and will iterate through all subdirectories.

Providing more details on certain “git” commands:

- **“git config”** commands: Git needs to be configured prior to using it. At a minimum, the username and email address must be configured. For more details refer [here](#).
- **“git clone”**: This command is one of the key steps in building the customized Murata i.MX Yocto image. After configuring the baseline i.MX Yocto environment, the user must install the “meta-murata-wireless” layer in the Yocto “sources” folder by invoking “git clone <https://github.com/murata-wireless/meta-murata-wireless.git>”.
- **“git branch -a”**: This command is useful to see which branches are currently available. Once you have the correct spelling of a given branch, then you can invoke the “git checkout” command to pull your desired release.
- **“git tag”**: For users who want to use formal configured/tested release. This command will list all the current release tags. Typically run only in “meta-murata-wireless” repository.
- **“git checkout”**: This command checks out a given branch or release tag. It can only be run after a git repository is created. In the Murata-customized build flow, we are invoking “git checkout” after “git clone” which creates a local copy of the “meta-murata-wireless”

repository. “git clone/checkout” are also useful to pull specific versions of NVRAM, firmware, or patch files. “git checkout” can be used to easily switch branches in any given repository.

- **“git status”**: This confirms which branch/release you are on. It also indicates if there have been any changes to the files: both tracked (files checked out as part of the current branch) and un-tracked (new files that you may have added).
- **“git reset --hard”**: This command resets the content of the current Git repository to the default for current “<branch\_name>”. This is a useful command if you implemented certain changes that you want to back out. You can also invoke “git reset <filename>” to reset (restore) just that one file.



Note the double “-” character entry before “hard” switch.

## 11 Appendix B: Useful “bitbake” commands

Quoting from the BitBake User Manual:

“FUNDAMENTALLY, BITBAKE IS A GENERIC TASK EXECUTION ENGINE THAT ALLOWS SHELL AND PYTHON TASKS TO RUN EFFICIENTLY AND IN PARALLEL WHILE WORKING WITHIN COMPLEX INTER-TASK DEPENDENCY CONSTRAINTS. ONE OF BITBAKE’S MAIN USERS, OPENEMBEDDED , TAKES THIS CORE AND BUILDS EMBEDDED LINUX SOFTWARE STACKS USING A TASK-ORIENTED APPROACH.”

Or to be more succinct, BitBake is the “Swiss Army Knife” when using Yocto or OpenEmbedded.

**Table 19** provides a list of essential “bitbake” commands.

**Table 19: Useful “bitbake” commands**

Yocto commands	Description
bitbake -c devshell virtual/kernel	To access kernel source
bitbake <recipe> -c devshell	Open a new shell where necessary system values already defined for recipe
bitbake-layers show-layers	To see the layers and their priorities.
bitbake -e <recipe>   grep ^PV	To check which version actually got selected for the recipe.
bitbake -c fetch <recipe>	To fetch source code. Source code can be found in downloads folder.
bitbake -b <recipe.bb> -c compile -D	To force compilation of the selected recipe.
bitbake -b <recipe.bb>	To perform build on the selected recipe.
bitbake -b <recipe.bb> -c clean	To perform clean on the selected recipe.
bitbake fsl-image-validation-imx	To create SD card image.
bitbake virtual/kernel -c menuconfig	Interactive kernel configuration
bitbake <image > -g -u depexp	Show the package dependency for <i>image</i> .



“bitbake” commands must be invoked from the Yocto build directory.

## 12 Appendix C: Example of running Host Setup for Yocto Script

```

$ ./Host_Setup_for_Yocto.sh
Murata: setup script to check Ubuntu installation and install
      additional host packages necessary for Yocto build
=====

1) Verifying Host Environment
-----
Murata: Verified Linux Distribution:  Ubuntu
Murata: Verified Ubuntu Release:      16.04

2) Verifying Host Script Version
-----
Fetching latest script from Murata Github.
Cloning "https://github.com/murata-wireless/meta-murata-wireless.git"
Creating "meta-murata-wireless" subfolder.
Latest: 06182018
Current: 06182018.....PASS

3) Installing Essential Yocto host packages
-----

Murata: Installing Essential Yocto Project host packages.
      sudoers-privileged user will be prompted for password...
[sudo] password for skerr:
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.1ubuntu2).
diffstat is already the newest version (1.61-1).
gawk is already the newest version (1:4.1.3+dfsg-0.1).
gcc-multilib is already the newest version (4:5.3.1-1ubuntu1).
unzip is already the newest version (6.0-20ubuntu1).
chrpath is already the newest version (0.16-1).
socat is already the newest version (1.7.3.1-1).
texinfo is already the newest version (6.1.0.dfsg.1-5).
git-core is already the newest version (1:2.7.4-0ubuntu1.9).
libssl1.2-dev is already the newest version (1.2.15+dfsg1-3ubuntu0.1).
wget is already the newest version (1.17.1-1ubuntu1.5).
0 upgraded, 0 newly installed, 0 to remove and 342 not upgraded.
Murata: Installing i.MX layers host packages...
Reading package lists... Done
Building dependency tree
Reading state information... Done
autoconf is already the newest version (2.69-9).
automake is already the newest version (1:1.15-4ubuntu1).
g++ is already the newest version (4:5.3.1-1ubuntu1).
gcc is already the newest version (4:5.3.1-1ubuntu1).
libglu1-mesa-dev is already the newest version (9.0.0-2.1).
make is already the newest version (4.1-6).
sed is already the newest version (4.2.2-7).
xterm is already the newest version (322-1ubuntu1).
asciidoc is already the newest version (8.6.9-3).
docbook-utils is already the newest version (0.6.14-3ubuntu1).
groff is already the newest version (1.22.3-7).
help2man is already the newest version (1.47.3).
lzop is already the newest version (1.03-3.2).

```

```
python-pysqlite2 is already the newest version (2.7.0-1).
repo is already the newest version (1.12.32-2).
texi2html is already the newest version (1.82+dfsg1-5).
coreutils is already the newest version (8.25-2ubuntu3~16.04).
curl is already the newest version (7.47.0-1ubuntu2.15).
cvs is already the newest version (2:1.12.13+real-15ubuntu0.1).
desktop-file-utils is already the newest version (0.22-1ubuntu5.2).
libgl1-mesa-dev is already the newest version (18.0.5-0ubuntu0~16.04.1).
libsdl1.2-dev is already the newest version (1.2.15+dfsg1-3ubuntu0.1).
subversion is already the newest version (1.9.3-2ubuntu1.3).
mercurial is already the newest version (3.7.3-1ubuntu1.2).
0 upgraded, 0 newly installed, 0 to remove and 342 not upgraded.
Murata: Installing i.MX layers host packages for a Ubuntu 16.04 or 14.04 host
setup only...
Reading package lists... Done
Building dependency tree
Reading state information... Done
u-boot-tools is already the newest version (2016.01+dfsg1-2ubuntu5).
0 upgraded, 0 newly installed, 0 to remove and 342 not upgraded.
Murata: Ubuntu Linux host environment verified, necessary host packages
installed...

4) GIT Configuration:
-----
Scott Kerr
*****@murata.com

user.name : Scott Kerr
user.email: *****@murata.com

Do you want to proceed with user name and email ID? Y/n: Y
Murata: please verify git username and email address prior to running Yocto
build.
Currently configured as...
user.name=Scott Kerr
user.email=*****@murata.com
Murata: ready to build "meta-murata-wireless" customized i.MX Linux image!
```

## 13 Appendix D: Example of running Murata Wireless Build

The following shows the output of building an image for [NXP i.MX 8MMini EVK](#), with [Murata 1ZM module](#), running [5.10.52](#) kernel release, using the Murata build script.

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

```
./Murata_Wireless_Yocto_Build_NXP.sh
```

2. Install the repo tool.

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

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

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

4. Select Hardknott Yocto release running Linux 5.10.52.

```
Select which entry? 0
```

```
Selected : 5.10.52
```

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

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

6. Proceed with the default distro and image selections.

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

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

7. Enter build\_8mm as build directory name.

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

8. Verify selection and start the build.

```
i.MX Yocto Release           : 5.10.52_2.1.0
Yocto branch                 : hardknott
Target                       : imx8mmevk
NXP i.MX EVK Part Number    : 8MMINILPD4-EVK
meta-murata-wireless Release Tag: imx-hardknott-5-10-52_r1.0
DISTRO                       : fsl-imx-wayland
Image                        : fsl-image-validation-imx
Build Directory              : build_8mm

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

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

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

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

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

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

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

12. Once the build is complete, the image will be available in ~/linux-imx/build\_8mm/tmp/deploy/images/imx8mmevk/ folder. Look for the file with ".wic.bz2" extension.

## 14 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.

### 14.1 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.

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

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

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

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

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

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

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

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

### 14.6 Murata Wi-Fi/Bluetooth for i.MX Linux User Guide for NXP-based Module

The [User Guide](#) details steps to get Murata Wi-Fi/BT NXP chipset-based solution up and running quickly on i.MX 6/8 EVKs.

### 14.7 Murata Linux Patched Solution

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

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

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

## 14.9 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.

## 14.10 Embedded Artists' Reference Documentation

Embedded Artists designed the 1ZM/1YM/1XK/1XL/2DS M.2 EVBs in close collaboration with Murata. Refer to [this main landing page](#)  for more information.



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

**Table 20** lists some relevant documents published by Embedded Artists.

**Table 20: Embedded Artists Documentation Listing**

Documentation Filename	Note
<a href="#">Wi-Fi/BT M.2 EVB Primer</a> 	Introduction and drill-down on M.2 interface
<a href="#">M.2 SDIO Interface Schematic</a> 	Reference schematic for customers designing in WLAN-SDIO M.2 EVB.
<a href="#">M.2 PCIe Interface Schematic</a> 	Reference schematic for customers designing in WLAN-PCIe M.2 EVB.
<a href="#">1ZM M.2 Module Datasheet</a> 	Comprehensive details on 1ZM Wi-Fi/BT M.2 Module.
<a href="#">1YM M.2 Module Datasheet</a> 	Comprehensive details on 1YM Wi-Fi/BT M.2 Module.
<a href="#">1XK M.2 Module Datasheet</a> 	Comprehensive details on 1XK Wi-Fi/BT M.2 Module.
<a href="#">1XL M.2 Module Datasheet</a> 	Comprehensive details on 1XL Wi-Fi/BT M.2 Module.
<a href="#">2DS M.2 Module Datasheet</a> 	Comprehensive details on 2DS Wi-Fi M.2 Module.

## 14.11 NXP Reference Documentation

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

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

**Table 21** provides the following information on all releases supported:

- Supported kernel release
- Documentation link(s)
- Corresponding Yocto release name

**Table 21: NXP Reference Documentation Listing for Supported Releases**

Kernel release	NXP documentation link	Yocto name
5.10.52_2.1.0	<a href="#">Rev. L5.10.52_2.1.0_BSP</a> 	Hardknott
5.10.72_2.2.0	<a href="#">Rev. L5.10.72_2.2.0_BSP</a> 	Hardknott

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

- i.MX\_Yocto\_Project\_User's\_Guide.pdf / IMXLXOCTOUG
- i.MX\_Linux\_User's\_Guide.pdf / IMXLUG
- i.MX\_Linux\_Reference\_Manual.pdf / IMXLXRM
- i.MX\_Linux\_Release\_Notes.pdf / IMXLXRN

## Revision History

Revision	Date	Author	Change Description
1.0	Jan 07, 2021	TF	Initial Release.
2.0	Jan 24, 2022	TF	Updated supported hardware and software configurations.
3.0	May 19, 2022	TF	Migrated to new format.



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

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

Specifications are subject to change without notice.