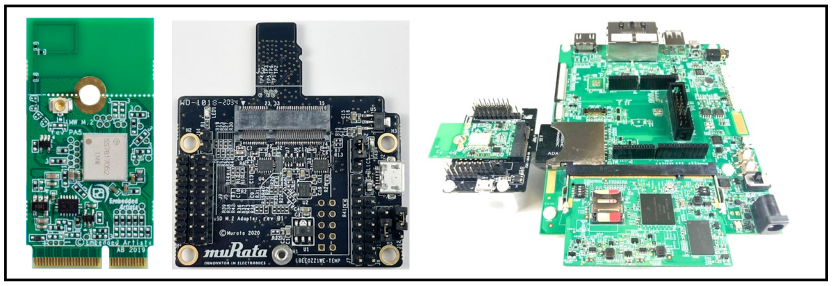
E2B-93-1092

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

Linux User Manual



Revision History

Revision	Date	Author	Change Description
1.0	Sept 7, 2015	S Kerr	Initial Release
2.0	Nov 7, 2015	S Kerr	Changes for L3.14.38_6UL GA BSP Release and support modified Murata Wi-Fi/BT EVK Definition.
3.0	March 1, 2016	S Kerr	Incorporated changes for NXP Linux 3.14.52 GA BSP Release. Added support for hostapd and Broadcom firmware package.
4.0	Feb 14, 2017	S Kerr	Renamed document to "Murata Wi-Fi/BT Solution for i.MX Linux User Manual". Incorporated changes for NXP Linux 4.1.15_2.0.0 GA BSP release. Modified NXP Linux 3.14.52_1.1.0 GA BSP release to build in bcmdhd WLAN driver, thereby matching 4.1.15_2.0.0 configuration. Added instructions for Murata source patch release which addresses errata/features on both releases. Added support for new i.MX 7Dual SDB, i.MX 6ULL EVK and Murata Type 1CK.
5.0	Dec 23, 2017	S Kerr J Kareem	Complete revision for integrating new Cypress "***fmac***" driver release. Added support for SDIO/UART 1.8V VIO signaling on i.MX6UL(L) and i.MX6SX platforms. Rollout of Murata customized Yocto build for i.MX BSP L4.9.11_1.0.0 release.
5.1	Jan 12, 2018	S Kerr J Kareem	Revise for "murata-wireless" GitHub and revised branches/releases. Add support for i.MX 7ULP EVK.
5.2	March 23, 2018	S Kerr J Kareem	Add support for new ***"fmac"*** release, codename ***"battra"***. Support latest i.MX hardware: NXP i.MX8MQuad EVK. Provide quick/easy steps for generating Linux image with automated scripting. Update build steps with Code Aurora repository paths. Provide explanation of how ***"meta-murata-wireless"*** customized layer works. Add support for i.MX Yocto Linux 4.1.15 "krogoth" branch.
5.3	April 24, 2018	S Kerr	Updated hyperlink to "***fmac battra***" release on Cypress website.
5.4	June 19, 2019	S Kerr	Updated for Linux 4.9.88 and "manda" version of "fmac" driver.
6.0	Jan 29, 2021	TF	Updated for Linux 4.9.123, 4.14.98 and 5.4.47. Added Embedded Artists' hardware solution, i.MX6/8 DTS file hierarchy, reference to 1XA and detailed backport steps. Changed document name from "Murata Wi-Fi/BT Solution for i.MX User Manual (Linux)" to "Murata Wi-Fi/BT (CYW) Solution for i.MX Linux User Manual".

This page is intentionally left blank.

TABLE OF CONTENTS

1 INTRODUCTION 6

Acronyms 10
1.1 References 11
1.1.1 Murata's i.MX Wireless Solutions Landing Page 11
1.1.2 Murata uSD-M.2 Adapter Datasheet 11
1.1.3 Murata Wi-Fi/BT Solution for i.MX Hardware User Manual 11
1.1.4 Murata Wi-Fi/BT (CYW) Solution for i.MX Linux User Guide 11
1.1.5 Murata Wi-Fi/BT (CYW) Solution for i.MX Linux Quick Start Guide 11
1.1.6 Murata's Community Forum Support 11
1.1.7 Embedded Artists' M.2 Modules Landing Page 11
1.1.8 NXP Reference Documentation 12

2 MURATA'S CUSTOMIZED I.MX YOCTO IMAGE EXPLAINED 13

2.1 Overview 13
2.2 Murata GitHub: Cornerstone of "fmac" Implementation 15
2.3 "*fmac*" Backport Implementation 18
2.4 "*meta-murata-wireless*": Pulling it All Together 19
2.4.1 What does "*meta-murata-wireless*" do exactly? 19
2.4.2 The Contents of "meta-murata-wireless" 19
2.4.3 Dynamic *"fmac"* Backport Steps Explained 23
2.4.4 Staging Directories: Important Note! 24

3 FINALLY… BUILDING I.MX YOCTO LINUX! 25

3.1 i.MX Yocto Build: Overview 25
3.2 i.MX Yocto Build: The Fast Track (In a Hurry or Linux Beginner?) 25
3.2.1 Install Ubuntu 25
3.2.2 Download Murata's Script Files 26
3.2.3 Configure Ubuntu for i.MX Yocto Build 26
3.2.4 Murata's i.MX Yocto Build Script 27
3.3 i.MX Yocto Build: Manual Steps (Advanced Users) 29
3.3.1 Image Build Configuration with i.MX Target Selection 30
3.3.2 Host PC Preparation 33
3.3.3 Yocto Project Setup 33
3.3.4 Fetch "meta-murata-wireless" from GitHub and copy into "Sources" 35
3.3.5 Install Necessary "hooks" for "meta-murata-wireless" 35
3.3.6 Build Murata-Customized Yocto Image for Specific i.MX Target 37
3.4 i.MX Yocto Build: Manual Steps "Take 2" (Quick Recap) 37
3.4.1 Initialize Linux i.MX Yocto Default Build Environment 37
3.4.2 Configuring the i.MX Target 38
3.4.3 Add "meta-murata-wireless" Layer and Kick off the Build 38
3.5 i.MX Yocto Build: Manual Steps "Take 2" for "5.4.47" with FMAC "Zigra" 39
3.5.1 For i.MX8 platform 39
3.5.2 For i.MX6/7 platforms 39
3.6 i.MX Yocto Build: Manual Steps "Take 2" for "4.14.98" with FMAC "Zigra" 40
3.6.1 For i.MX8 platform 40
3.6.2 For i.MX6/7 platforms 41
3.7 i.MX Yocto Build: Manual Steps "Take 2" for "4.9.123" with FMAC "Zigra" 42
3.7.1 For i.MX8 platform 42
3.7.2 For i.MX6/7 platforms 43
3.8 i.MX Yocto Build: Manual Steps "Take 2" for "4.1.15" with FMAC "manda" 44
3.8.1 For i.MX6/7 platforms 44

4 BLUETOOTH PATCH FILE CONSIDERATIONS 45

5 USING "FMAC" DRIVER 45

5.1 How-To on "fmac" driver 45
5.1.1 Using Switch Modules Script File 46

5.1.2 Successful *“fmac”* load message 46
5.1.3 *“wlan0”* initialization 46

6 BUILDING LINUX KERNEL WITH FMAC STANDALONE 47

7 APPLYING CYPRESS PATCHES TO HOSTAPD AND WPA SUPPLICANT 48

8 BUILDING HOSTAPD THROUGH YOCTO 50

9 BUILDING WPA SUPPLICANT THROUGH YOCTO 50

10 HIERARCHY OF DEVICE TREE SOURCE FILES FOR I.MX6/8 PLATFORMS 51

11 EMBEDDED ARTISTS’ SOLUTION 55

12 DYNAMIC BACKPORTING 58

12.1 Install stand-alone toolchain for cross-compiling 58
12.2 Build the kernel image 61
12.3 Build the cypress driver/ backports modules 66
12.4 Device tree 67
12.5 Driver and related files 69
12.6 Flash a demo image 69
12.7 Modify Bootloader 70
12.8 Modify filesystem 71
12.9 Load driver 71

13 TECHNICAL SUPPORT RESOURCES 72

14 APPENDIX A: USEFUL “*GIT*” COMMANDS 73

15 APPENDIX B: USEFUL “BITBAKE” COMMANDS 75

16 APPENDIX C: EXAMPLE OF RUNNING HOST SETUP FOR YOCTO SCRIPT 76

17 APPENDIX D: EXAMPLE OF RUNNING MURATA WIRELESS BUILD 78

LIST OF TABLES

Table 1: Current NXP i.MX Platforms Supported 6
Table 2: Acronyms used in Linux User Manual 10
Table 3: NXP Reference Documentation Listing for Supported Releases 12
Table 4: NXP i.MX EVK Part Number / Yocto (MACHINE) target / Kernel Version Matrix 14
Table 5: Murata GitHub Repositories used in “fmac” build 16
Table 6: Config DTBs for EVKs 20
Table 7: Important folders/files in “meta-murata-wireless” 22
Table 8: NXP i.MX EVK Part Number / Murata Module Interconnect 31
Table 9: i.MX6/7/8 Targets supported by Murata 32
Table 10: Embedded Artists' i.MX InterConnect 56
Table 11: Embedded Artists’ Landing Pages 56
Table 12: Embedded Artists’ Datasheets and Schematics 57
Table 13: Embedded Artists’ User Manuals and Software 57
Table 14: List of Support Resources 72
Table 15: Useful “*git*” commands 73
Table 16: Useful “*bitbake*” commands 75

LIST OF FIGURES

Figure 1: i.MX6 EVK Interconnect Block Diagram 7
Figure 2: i.MX 7Dual SDB Block Diagram 8
Figure 3: i.MX 8QXP MEK & 8MQuad EVK WLAN/BT Interconnect Block Diagram 8
Figure 4: i.MX 8M Mini EVK WLAN PCIe Interconnect Block Diagram 9
Figure 5: i.MX 8M Mini/Nano EVK WLAN/BT Interconnect Block Diagram 9
Figure 6: Configuring dash 27
Figure 7: DTS hierarchy for imx6ulevk 51
Figure 8: DTS hierarchy for imx6ull14x14evk 52
Figure 9: DTS hierarchy for imx6sxevk 52
Figure 10: DTS hierarchy for imx6qsabresd 53
Figure 11: DTS hierarchy for imx8mmevk 53
Figure 12: DTS hierarchy for imx8mnevk 54
Figure 13: Combine i.MX COM with Wi-Fi/BT M.2 EVB 55

1 Introduction

Murata has partnered with NXP Semiconductors N.V., Cypress Semiconductor Corporation, and Embedded Artists AB to offer a complete Wi-Fi and Bluetooth connectivity environment for building world class Internet-connected products. The Murata Connectivity Modules enable developers to minimize the development time and effort for connectivity function implementation. This Linux User Manual provides details on building all necessary software for enabling Murata Wi-Fi/Bluetooth on reference NXP i.MX platforms. The latest release supports NXP i.MX Linux 4.1.15 for i.MX 6/7, Linux 4.9.123, Linux 4.14.98 and Linux 5.4.47 BSP's for i.MX 6/7/8. This manual details Murata's customized wireless Yocto layer which allows the user to easily build their desired i.MX image while pulling in the WLAN FMAC driver, configured/patched WPA supplicant & Hostapd, firmware files, NVRAM files, and Bluetooth patch files.

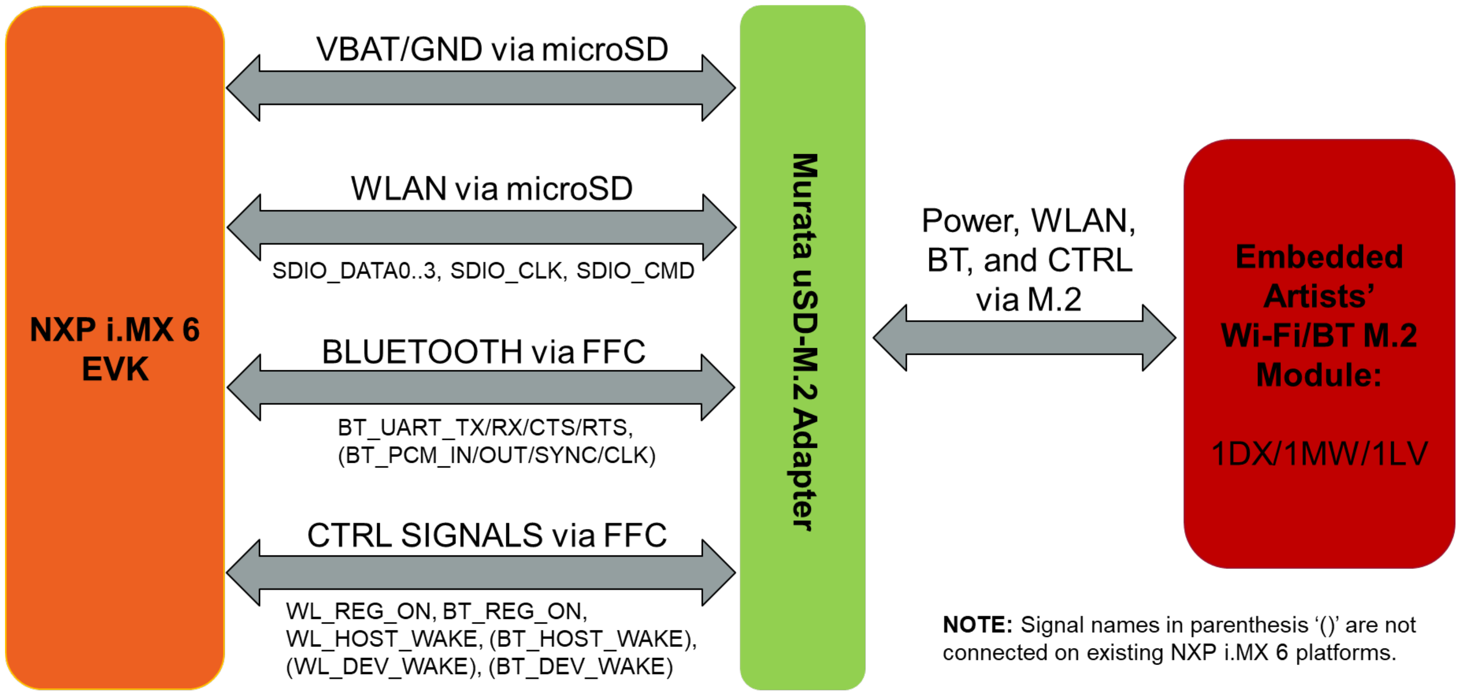
Table 1: Current NXP i.MX Platforms Supported

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

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

Figure 1 illustrates a high-level connection diagram between NXP i.MX 6 EVK's and Murata module. This configuration is enabled with Murata's uSD-M.2 Adapter and Embedded Artists' Wi-Fi/BT M.2 EVB's. Note that there are limitations on this interconnect. Chief amongst them is fixed WLAN-SDIO VIO of 3.3V on certain platforms, and a maximum WLAN-SDIO clock speed of 50 MHz. Please refer to the Murata Linux User Guide, Murata uSD-M.2 Adapter Datasheet, and Murata Hardware User Manual for more details.

Figure 1: i.MX6 EVK Interconnect Block Diagram


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

NOTE: Only i.MX6UL(L) EVK support 1.8V VIO signaling with J12 jumper on uSD-M.2 Adapter set to position 1-2. For more details on Wi-Fi throughput dependency on SDIO bus speed and hardware modifications necessary for 1.8V VIO signaling, please refer to the Murata Hardware User Manual.

Figure 2 shows a simplified block diagram for the i.MX 7Dual SDB. It shows the WLAN, Bluetooth and control signals between the processor and the module. No adapter is required for the i.MX 7Dual SDB[3]. This platform has the Murata ZP module soldered down on the board. Both Wi-Fi and Bluetooth interfaces are supported on this platform. Unlike the 3.3V VIO Interconnect limitation on some i.MX6 platforms, there is no inherent "legacy" restriction on the i.MX7D platform which limits SDIO throughput. The Murata ZP module supports a high throughput (SDIO 3.0 mode – UHS) over SDIO bus resulting in a much better performance. Please reference the NXP i.MX7 schematics for specifics: download package here[4]. Note that the ***Murata Type ZP module is no longer promoted.*** A suggested replacement is Type 1MW.

[3] Although an external module can be connected to the i.MX7D SDB, Murata does not recommend this given extensive rework required.
[4] For Wi-Fi/BT schematics on i.MX 7Dual SDB, refer to page 13 of "sch-28590_i.mx7d_saber_rev_2.pdf" document.

Figure 2: i.MX 7Dual SDB Block Diagram

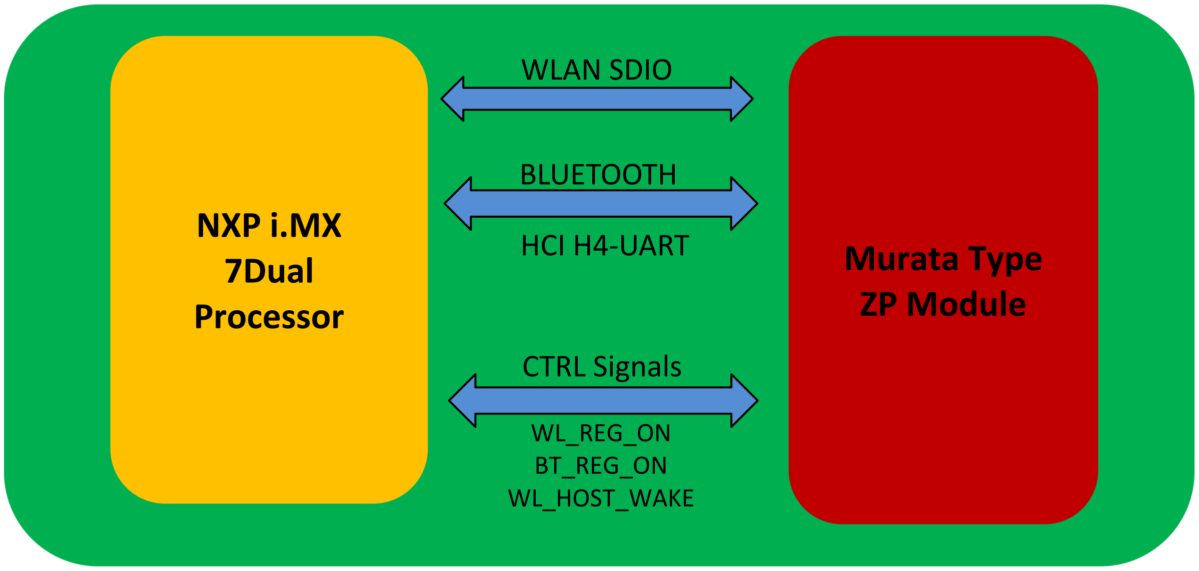


Figure 3 shows a simplified block diagram for the i.MX 8QXP MEK & i.MX 8MQuad EVK Wi-Fi/BT interconnect. Currently Type 1CX and 1XA are supported with WLAN-PCIe interface. No adapter is needed for this interconnect as the M.2 EVBs can be connected directly to the i.MX 8 EVK.

Figure 3: i.MX 8QXP MEK & 8MQuad EVK WLAN/BT Interconnect Block Diagram

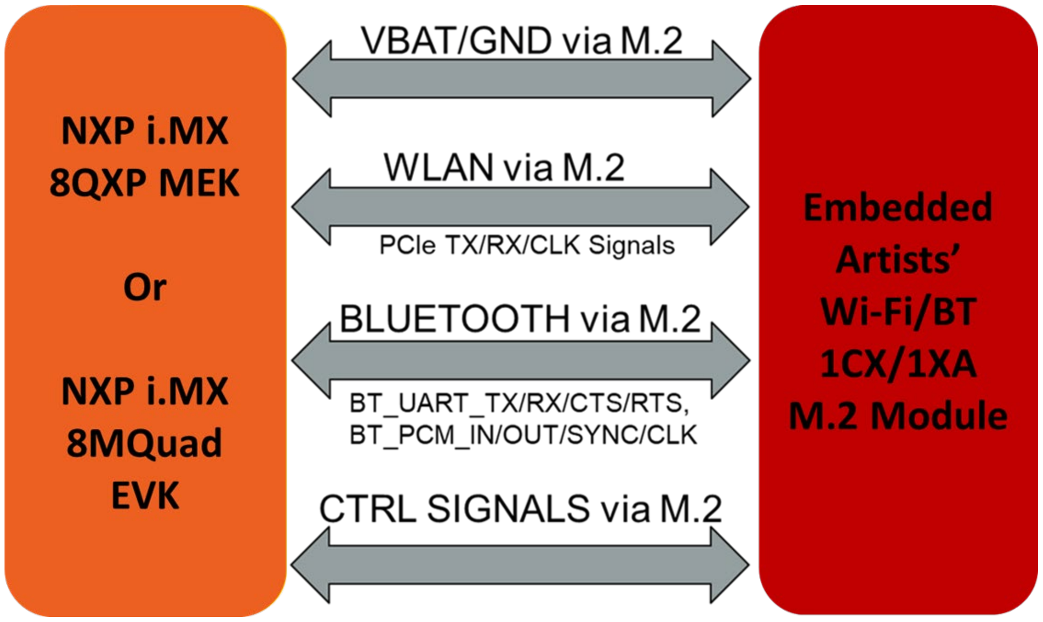


Figure 4 shows a simplified block diagram for the i.MX 8M Mini EVK WLAN interconnect with onboard M.2 EVB option. Currently, Type 1CX and 1XA M.2 Modules (WLAN Only) are supported with 2x2 802.11ac MIMO and WLAN-PCIe interface. Note that the NXP i.MX 8M Mini EVK **does not bring out** the Bluetooth signals to the M.2 connector – **WLAN only**.

Figure 5 shows a simplified block diagram for the i.MX 8M Mini/Nano EVK interconnect with Murata's uSD-M.2 Adapter option (with Embedded Artists' Wi-Fi/BT M.2 EVB). Only WLAN-SDIO based modules are supported in this configuration: 1MW, 1DX and 1LV. To properly support this (WLAN-SDIO VIO @1.8V; BT-UART VIO @3.3V) interconnect, Rev B1 uSD-M.2 Adapter needs to be used given that it correctly level shifts the Bluetooth and WLAN/BT control signals between the M.2 EVB and the NXP i.MX 8M Mini EVK.

Figure 4: i.MX 8M Mini EVK WLAN PCIe Interconnect Block Diagram

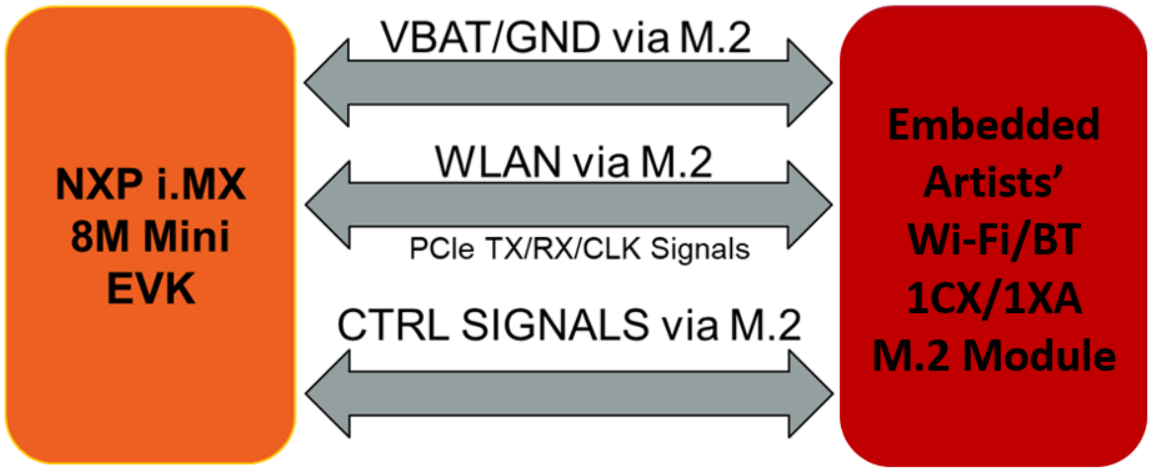
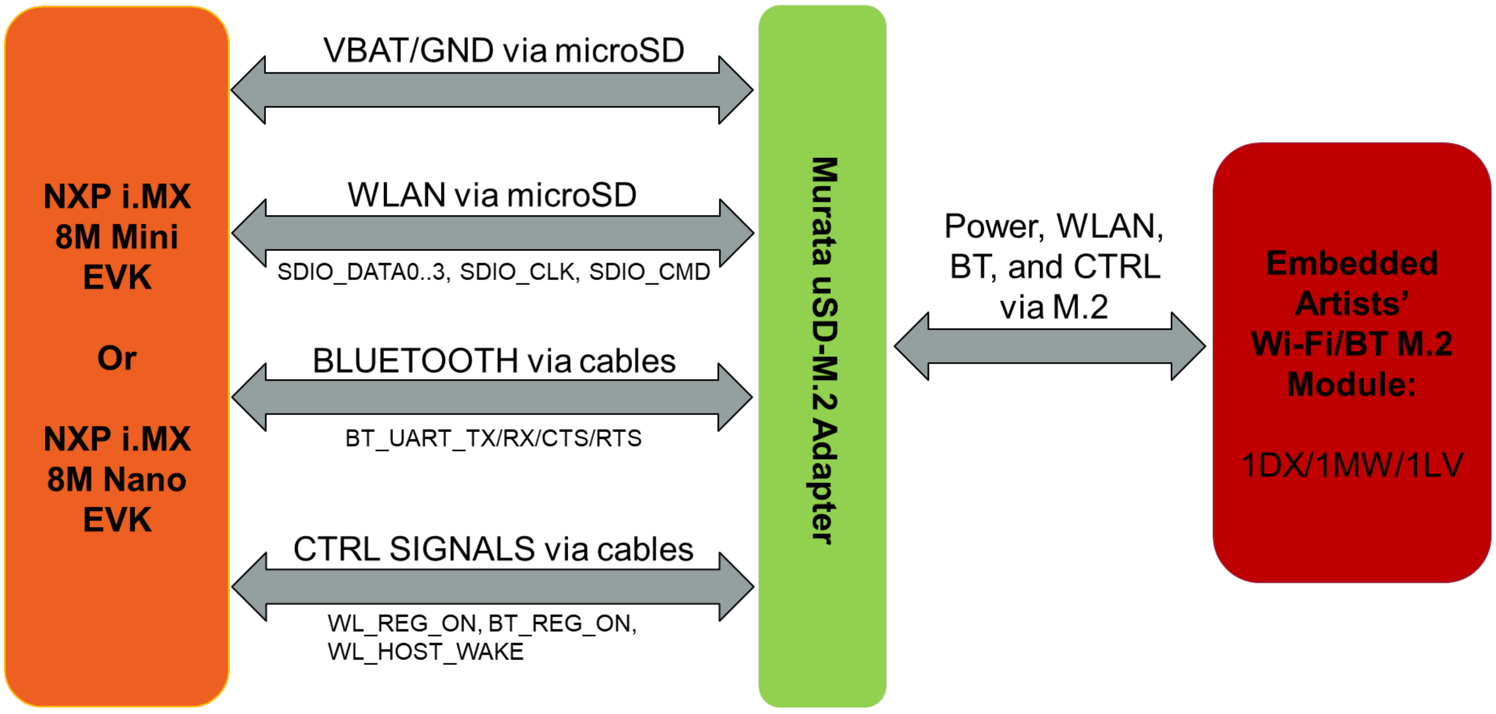


Figure 5: i.MX 8M Mini/Nano EVK WLAN/BT Interconnect Block Diagram



Acronyms

Table 2: Acronyms used in Linux User Manual

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

1.1 References

1.1.1 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.

1.1.2 Murata uSD-M.2 Adapter Datasheet

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

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

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

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

This User Guide details steps to get Murata Wi-Fi/BT Cypress chipset-based solution up and running quickly on i.MX 6/7/8 EVK's.

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

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

1.1.6 Murata's Community Forum Support

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

1.1.7 Embedded Artists' M.2 Modules Landing Page

This website landing page provides latest/comprehensive information on Embedded Artists' M.2 Evaluation Boards which enable Murata Wi-Fi/BT modules for easy evaluation.

1.1.8 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 3 provides the following information on all releases supported:

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

Table 3: NXP Reference Documentation Listing for Supported Releases

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

Each archive downloadable contains the following:

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

2 Murata's Customized i.MX Yocto Image Explained

2.1 Overview

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

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

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

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

There are three versions of ***"fmac"*** currently supported: ***"v4.14 manda"***, ***"v4.14 kong"*** and ***"v5.4 zigra"***. Note that ***"manda"***, ***"kong"*** and "***zigra***" are the Cypress codenames denoting ***"fmac"*** release version. ***"v4.12"***, ***"v4.14"*** and ***"v5.4"*** are the latest kernel versions supported by the releases

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

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

Murata's customized Yocto layer ("***meta-murata-wireless***") supports the following NXP i.MX EVK's as outlined in **Table 4**. ***"MACHINE=target"*** is a direct reference to Yocto build. The ***"target"*** string is the keyword used to select hardware configuration for the build (important knowledge when running Murata build script). With the newer EVK's (i.MX 8QXP, i.MX 8MQuad, i.MX 8M Mini, i.MX 8M Nano, and i.MX 7ULP) only certain kernel versions are supported. From this table, you can find a proper version of Linux Kernel for your target platform. You can then refer to **Table 9** for the support of interrupt configuration, corresponding DTB (Device Tree Blob) files, hardware interconnect configuration (either module onboard, uSD-M.2 Adapter & M.2 EVB, or just M.2 EVB), and if the hardware configuration is limited to 3.3V VIO on WLAN SDIO interface (default WLAN SDIO VIO is 1.8V). The hardware 3.3V VIO WLAN SDIO interface limitation only applies to certain legacy i.MX 6 platforms (6QP, 6Q, 6DL, 6SX). The 3.3V VIO signaling requires the uSD-M.2 Adapter to be configured in a "3.3V VIO Override mode". Not all Wi-Fi/BT M.2 EVB's support 3.3V VIO on WLAN-SDIO interface.

Table 4: NXP i.MX EVK Part Number / Yocto (MACHINE) target / Kernel Version Matrix

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

2.2 Murata GitHub: Cornerstone of "fmac" 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 ***"fmac"*** i.MX Yocto implementation. Refer to **Table 5** for a complete listing of the Murata repositories 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: https://github.com/murata-wireless. Depending on the repository, the branch naming convention varies as follows:

- ***"fmac"*** version (*"**zigra**", "**kong**"*, ***"manda"***). It is strongly recommended to use ***"zigra"*** (latest release).
- Linux kernel (Yocto codename. i.e. ***"zeus"***, ***"sumo"***, ***"rocko-mini", "krogoth"***) and ***"fmac"*** versions.
- i.MX architecture (i.MX8 versus i.MX6/7), Linux kernel, and ***"fmac"*** versions.

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 ***"fmac"*** 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.

- The ***"master"*** branch in ***"cyw-fmac"*** repository corresponds to the latest ***"fmac"*** release – in this case ***"zigra"***. If the user is checking for latest patches to ***"fmac"*** driver, then it is recommended to check ***"drivers/net/wireless/broadcom/brcm80211"*** folder in https://github.com/murata-wireless/cyw-fmac.

Table 5: Murata GitHub Repositories used in “fmac” build

Murata GitHub Repository Name	Branch Names	Latest Release / Tag Names	Contents
“meta-murata-wireless”	"master" “imx-zeus-zigra” “imx-sumo-zigra” “imx-sumo-kong” “imx-sumo-manda” “imx-rocko-mini-zigra” “imx-rocko-mini-kong” “imx-rocko-mini-manda” “imx-krogoth-zigra”, “imx-krogoth-kong” “imx-krogoth-manda” GitHub Listing	“imx-zeus-zigra_r1.0” “imx-sumo-zigra_r1.0” “imx-sumo-kong_r1.1” “imx-sumo-manda_r1.2” “imx-rocko-mini-zigra_r1.0” “imx-rocko-mini-kong_r1.0” “imx-rocko-mini-manda_r2.2” “imx-krogoth-zigra_r1.0” “imx-krogoth-kong_r1.0” “imx-krogoth-manda_r2.2” GitHub Listing	***“meta-murata-wireless”*** 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.
“cyw-fmac”	“master” “imx-zeus-zigra” “imx-sumo-zigra” “imx-sumo-kong” “imx-sumo-manda” “imx-rocko-mini-zigra” “imx-rocko-mini-kong” “imx-rocko-mini-manda” “imx-krogoth-zigra” “imx-krogoth-kong” “imx-krogoth-manda” GitHub Listing	“imx-zeus-zigra_r1.0” “imx-sumo-zigra_r1.0” “imx-sumo-kong_r1.0” “imx-sumo-manda_r1.0” “imx-rocko-mini-zigra_r1.0” “imx-rocko-mini-kong_r1.0” “imx-rocko-mini-manda_r2.2” “imx-krogoth-zigra_r1.0” “imx-krogoth-kong_r1.0” “imx-krogoth-manda_r2.2” GitHub Listing	“***fmac***” backports tarball extracted from official Cypress release with some modifications. Links to original Cypress releases provided below. ***“zigra”:*** https://community.cypress.com/docs/DOC-20044 ***“kong”:*** https://community.cypress.com/docs/DOC-19000 ***“manda”:*** https://community.cypress.com/docs/DOC-15932
“cyw-fmac-fw”	“master” “zigra” “kong” “manda” GitHub Listing		WLAN production firmware files (including any applicable “CLM blob” files) for following modules: SN8000, 1FX, 1DX/1LN, 1BW, 1LV, ZP/1CK, 1MW/1LC/1HK, 1BB, 1CX/1DK and 1XA.
“cyw-fmac-nvram”	“master” “zigra” “kong” “manda” GitHub Listing		WLAN NVRAM files (customized by Murata with Cypress guidance) for following modules: SN8000, 1FX, 1DX/1LN, 1BW, 1LV, ZP/1CK, 1MW/1LC/1HK, 1BB, 1CX/1DK and 1XA.
“cyw-fmac-utils-imx32”	“master”,“zigra”,“kong”, “manda”,GitHub Listing		“wl” tool pre-compiled for i.MX 32-bit.
“cyw-fmac-utils-imx64”	“master”,“zigra”,“kong”, “manda”,GitHub Listing		“wl” tool pre-compiled for i.MX 64-bit.
“cyw-bt-patch”	“master” “zeus-zigra” “sumo-zigra” “sumo-kong” "sumo-manda” “imx-rocko-mini-zigra” “imx-rocko-mini-kong” “imx-rocko-mini-manda” “krogoth-zigra” “krogoth-sumo” “krogoth-manda” GitHub Listing		Bluetooth patch files (*.hcd) which are used during Linux/BlueZ “hciattach” call to configure BT core. Bluetooth patch files for following modules: 1DX/1LN, 1BW, 1LV, ZP/1CK, 1MW/1LC/1HK, 1BB, 1CX/1DK and 1XA.

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/fmac 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.

- ***"cyw-fmac"*** contains the backport tool source used to backport the ***"fmac"*** driver and the actual WLAN ***"fmac"*** driver code: see sub-folder ***"/drivers/net/wireless/broadcom/brcm80211/brcmfmac"***. The backport tool source is customized by Cypress to correct various issues/errata. Unlike the ***"brcmfmac"*** integrated into the Linux kernel GIT (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git), this release is configured/tested/released by the Cypress team. The integrated backport tool (refer to https://backports.wiki.kernel.org/index.php/Main_Page) allows the ***"fmac"*** driver to be "dropped into" any Linux kernel version from 3.0 to 4.14("***manda***")/4.14 ("***kong***")/ 5.4 ("***zigra***"). The ***"meta-murata-wireless"*** layer uses this repository to dynamically backport the ***"fmac"*** driver to the various supported i.MX kernel versions. **NOTE:** the i.MX kernel does include "***drivers/net/wireless/broadcom/brcm80211/brcmfmac***" – please ignore this driver source code. It is very important that this **internal** "***brcmfmac***" driver is disabled to avoid any conflicts with the Cypress "***fmac***" implementation.

- Except "***master***" (and minor revisions to "***meta-murata-wireless***"), there is a one-to-one mapping of branches between the ***"meta-murata-wireless"*** and "***cyw-fmac***" repositories. i.e. "***imx-zeus-zigra***" branch of ***"meta-murata-wireless"*** backports/compiles the ***"fmac"*** source code from the same branch in "***cyw-fmac***". **NOTE:** ***"meta-murata-wireless"*** does not compile the ***"fmac"*** source from "***master***" branch of "***cyw-fmac***" repository. This is because each ***"meta-murata-wireless"*** branch is dependent not just on the ***"fmac"*** version, but also the CPU selected (i.e. "imx8" or "imx") and the Yocto version (i.e. "***krogoth***", "***rocko-mini***", "***sumo***", or "***zeus***").

- The "***cyw-bt-patch***" repository uses branch names keying off the kernel (Yocto codename) and ***"fmac"*** versions. This is done to better manage kernel-specific issues - particularly regarding BlueZ stack implementation (default i.MX Yocto stack implementation).

- The remaining repositories ("***cyw-fmac-fw***", "***cyw-fmac-nvram***", "***cyw-fmac-utils-imx32***", and "***cyw-fmac-utils-imx32***") key off the Cypress ***"fmac"*** driver codename – in this case ***"zigra"***, "***kong***", or "***manda***". ***"meta-murata-wireless"*** only pulls from the Cypress codename branch; it does not pull from "***master***". If the user needs the (not yet officially released) updates from a given "***master***" branch, they should download the files from Murata GitHub manually and install them on their Linux file system (or modify their own customized Yocto build).

- The "***cyw-fmac-nvram***" repository contains additional NVRAM files (text format) to provide comprehensive support for all Murata modules. This is necessary due to how the WLAN driver (***"fmac"***) is implemented. When the ***"fmac"*** driver loads, it looks for specific filename formats for both WLAN NVRAM and optional regulatory conformance file (CLM Blob) in the "***/lib/firmware/brcm***" folder:

  - ***"brcmfmac" + <CYW chipset number> + <"-sdio" or "-pcie"> + ".txt"***
  - ***"brcmfmac" + <CYW chipset number> + <"-sdio" or "-pcie"> + ".clm_blob"***

  There are additional NVRAM files which are not the "Cypress Chipset Default" module. The default modules include: ZP (CYW4339), and 1CX (CYW4356). **<u>NOTE:</u>** The **"fmac"** driver requires that if the user needs to use a "non-default" Murata module, then the NVRAM file being copied into the ***"/lib/firmware/brcm"*** folder needs to have the module-identifying string removed. For more details refer to the User Guide.

  <u>NOTE:</u> Starting from ***"fmac"*** version, ***"zigra"*** *and later*, when the ***"fmac"*** driver loads, it looks for specific filename formats for both WLAN NVRAM and optional regulatory conformance file (CLM Blob) in the "***/lib/firmware/cypress***" folder:
  - ***"cyfmac" + <CYW chipset number> + <"-sdio" or "-pcie"> + ".txt"***
  - ***"cyfmac" + <CYW chipset number> + <"-sdio" or "-pcie"> + ".clm_blob"***

- The "***cyw-bt-patch***" repository contains additional Bluetooth patch files necessary to comprehensively support all Murata modules. This is necessary due to the BT Stack (BlueZ) implementation. To initialize the BT core, the BlueZ function "***hciattach***" is called and searches a Bluetooth patch file with format: ***"BCM" + <CYW chipset number> + <CYW chipset version> + <Addl Chars> + ".hcd"***

  <u>NOTE:</u> "BCM" not "CYW" is first character string. The "***Addl Chars***" field provides flexibility, thereby allowing Murata to add the module name so Bluetooth patch files can be distinguished between different modules with same Cypress chipset. Unfortunately, this "free field" is not possible with the ***"fmac"*** driver implementation. For more details refer to Linux User Guide.

2.3 "*fmac*" Backport Implementation

Many users may already be familiar with "***brcmfmac***" open source driver. This WLAN driver is an established part of the Linux kernel. If the user clones the Linux kernel repository at https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git, the latest "***brcmfmac***" source code is at "***/drivers/net/wireless/broadcom/brcm80211/brcmfmac***". Both Broadcom and Cypress contribute to this release, upstreaming their latest code updates and fixes. By comparison to the *true* open source "***brcmfmac***" driver, the key attributes to consider regarding the Cypress ***"fmac"*** release include:

- The driver release follows formal software development practices. It is configured, tested, and released every three (3) to four (4) months. Currently three releases are available: ***"zigra"***, "***kong"***, and ***"manda"***. It is recommended that the user updates to latest ***"fmac"*** driver version.

- Key components such as WLAN firmware binaries, along with WPA supplicant and Hostapd (including patches) are tested and released with the ***"fmac"*** driver.

- The ***"fmac"*** driver release is integrated with Linux backport tool. For more details on backport refer to https://backports.wiki.kernel.org/index.php/Main_Page. The key attraction here is that the *same* Cypress ***"fmac"*** source code release can be evaluated on different kernel versions. Also, the backport tool provides "automatic" porting – something which the ***"meta-murata-wireless"*** layer invokes dynamically during the Yocto i.MX image build.

- Lastly, Cypress and Murata **<u>only support</u>** the formal ***"fmac"*** driver release.

2.4 “*meta-murata-wireless*”: Pulling it All Together

2.4.1 What does “*meta-murata-wireless*” do exactly?

Quick recap on the essential ingredient in arriving at the correctly configured, ***“fmac”***-enabled i.MX image. Murata’s customized Yocto layer ***“meta-murata-wireless”*** seamlessly disables any previous WLAN driver and pulls in the ***“fmac”*** (officially supported) driver implementation. More specifically it provides the following enhancements/customizations:

- Pull Cypress ***“fmac”*** driver and run backports tool during Yocto build to generate necessary driver modules.
- Additional/necessary patches to Cypress ***“fmac”*** driver for i.MX implementation.
- i.MX Linux kernel customizations to support ***“fmac”*** driver with OOB IRQ interrupts.
- Support 1.8V VIO signaling with NXP i.MX6UL(L) EVK.
- Support Wi-Fi/BT enablement on microSD slot of NXP i.MX 8M Mini/Nano EVK’s.
- Fine tune DTS files so that optimal WLAN-SDIO throughput is achieved.
- WLAN production firmware files. For manufacturing test firmware (necessary for regulatory testing), please contact Murata directly. If no direct contact is available to you, then please post query to Murata’s Community Forum at https://community.murata.com.
- Murata NVRAM files for correctly configuring wireless module RF characteristics.
- Example Bluetooth patch files which allow customers to initial Bluetooth evaluation.
- WL tool binary necessary for interoperability and RF testing.
- Hostapd configuration (Cypress-specific version for a given “***fmac***” release) with specific Cypress’ patch release.
- Hostap-conf enablement.
- Hostap-utils enablement.
- WPA-supplicant configuration (Cypress-specific version for a given “fmac” release[6]) with Cypress’ specific patch release.
- Wi-Fi Direct (P2P) enablement.

For a Yocto primer, please refer to https://wiki.yoctoproject.org/wiki/Main_Page. For specifics on the i.MX Yocto implementation and their project user’s guide, please refer to **Table 3.**

2.4.2 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 7** for a description of important folders/files in the ***“meta-murata-wireless”*** layer.

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

- ***"script-utils/latest/":*** contains "***Host_Setup_for_Yocto.sh***" and "***Murata_Wireless_Yocto_Build.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 supplicant 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 supplicant package it pulls in will not be part of the image (version 2.5). Rather, the ***"meta-murata-wireless"*** WPA supplicant package (Version 2.9 for "***kong***" and later fmac, Version 2.6 otherwise) with associated/configured/tested patches is compiled into the i.MX image.

- ***"freescale/<MACHINE>.conf":*** When compiling i.MX image for the following EVK's, these machine configuration files are critical. The "conf" files (**see Table 6**) pull 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 6: Config DTBs for EVKs

EVK Name	Machine File Name
i.MX6UL	imx6ulevk.conf
i.MX6ULL	imx6ull14x14evk.conf
i.MX 6SX SDB	imx6sxsabresd.conf
i.MX 6DualLite SDB	imx6dlsabresd.conf
i.MX 6QuadPlus SDB	imx6qpsabresd.conf
i.MX 6Quad SDB	imx6qsabresd.conf
i.MX 8M Mini EVK	imx8_all.conf, layer.conf
i.MX 8M Nano EVK	imx8mnevk.conf, imx8mnddr4evk.conf, imx8mnlpddr4evk.conf

- ***"recipes-connectivity/hostapd"***: This folder contains the recipe file that configures the Hostapd version (i.e. "***hostapd_2.9.bb***" or "***hostapd_2.6.bb***"). The "***hostapd***" sub-folder contains Hostapd patches, and necessary configuration files. **NOTE:** every release of ***"fmac"*** is configured for a specific Hostapd. As such, it is **extremely important** that the user does not deviate from this Hostapd configuration.

- ***"recipes-kernel/backporttool-linux/backporttool-linux_1.0.bb"***: This recipe dynamically backports the ***"fmac"*** driver during the Yocto build. For more details, refer to **Section 12**.

- ***"recipes-connectivity/murata-binaries/murata-binaries_1.0.bb"***: This recipe installs the following files:
    - WLAN firmware/regulatory files from https://github.com/murata-wireless/cyw-fmac-fw repository into "***/lib/firmware/cypress***" folder.
    - Bluetooth ("****.hcd***") patch files from https://github.com/murata-wireless/cyw-bt-patch into "***/etc/firmware***" folder.
    - WLAN NVRAM files from https://github.com/murata-wireless/cyw-fmac-nvram into "***/lib/firmware/cypress"*** folder.
    - "***wl***" tool binary from https://github.com/murata-wireless/cyw-fmac-utils-imx32 or https://github.com/murata-wireless/cyw-fmac-utils-imx64 into "***/usr/sbin***" folder. **NOTE:** the "***wl***" tool binary differs for i.MX 32-bit and 64-bit architecture.

  NOTE: Prior ***"fmac"*** release ***"kong"*** and older, WLAN firmware/regulatory files and NVRAM files will be placed into "***/lib/firmware/brcm"*** folder.

- ***"recipes-connectivity/murata-binaries/murata-binaries/switch_module.sh"***: This script file ensures automatic driver loading for CYW-SDIO and CYW-PCIe chipsets. It is placed in the location, ***"/usr/sbin"***
  Usage: "***switch_module.sh cyw***"

- ***"recipes-connectivity/wpa-supplicant"***: This folder contains the recipe file that configures the WPA Supplicant version (i.e. "***wpa_supplicant_2.9.bb***" or "***wpa-supplicant_2.6.bb***"). The "***wpa-supplicant***" sub-folder contains WPA Supplicant patches, and necessary configuration files. **NOTE:** every release of ***"fmac"*** is configured for a specific WPA Supplicant. As such, it is **extremely important** that the user does not deviate from this WPA Supplicant configuration.

- ***"recipes-kernel/linux-firmware/linux-firmware_20190815.bbappend "***: removes "***lib/firmware/brcm***" folder and all its contents. This ensures that a clean install (with correct NVRAM, firmware, and regulatory files) is done by the "***murata-binaries_1.0.bb***" recipe.

- ***"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 7: 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 ***"fmac"*** implementation for current "***target***" build.
conf/layer.conf	Enable ***"meta-murata-wireless"*** bbappend files and set priority for ***"meta-murata-wireless"*** layer. **<u>NOTE</u>**: 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/imx6sxsabresd.conf freescale/imx6dlsabresd.conf freescale/imx6qpsabresd.conf freescale/imx6qsabresd.conf freescale/imx8_all.conf freescale/imx8mnddr4evk.conf freescale/imx8mnevk.conf freescale/imx8mnlpddr4evk.conf freescale/layer.conf	Ensures that all necessary i.MX6UL(L), i.MX6 SX SDB, i.MX6 Dual Lite SDB, i.MX 6Quad SDB, 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.
recipes-connectivity/ hostapd/	Configure and patch Hostapd.
recipes-connectivity/ murata-binaries/murata-binaries_1.0.bb	Installs switch module script, WLAN firmware, Bluetooth patch, WLAN NVRAM, WLAN regulatory (CLM_blob), and "***wl***" binary files on root file system.
recipes-connectivity/ murata-binaries/murata-binaries/switch_module.sh	Automatically switches WLAN driver loading for SDIO and PCIe CYW devices.
recipes-connectivity/ cyw-supplicant/	Configure and patch WPA supplicant for CYW.
recipes-connectivity/ wpa-supplicant/wpa-supplicant_%.bbappend	Renames default WPA supplicant as NXP WPA supplicant (wpa_supplicant.nxp).
recipes-kernel/ backporttool-linux/ backporttool-linux_1.0.bb	Recipe that generates WLAN driver files (including cross-compilation step) and copies them to root file system. From <u>***"/lib/modules/${KERNEL_VERSION}/kernel/"*** folder:</u> - ***"drivers/net/wireless/broadcom/brcm80211/brcmfmac/brcmfmac.ko"*** - ***"drivers/net/wireless/broadcom/brcm80211/brcmutil/brcmutil.ko"*** - ***"compat/compat.ko"*** - ***"net/wireless/cfg80211.ko"***
recipes-kernel/ linux-firmware/ linux-firmware_20190815.bbappend	Appends to the default i.MX recipe "***poky/meta/recipes-kernel/linux-firmware/linux-firmware_git.bb***" which pulls various firmware files and writes them to ***"/etc/firmware"*** folder. The bbappend file removes the ***"/lib/firmware/brcm"*** folder and all its contents. This is important so that only the specific configured contents (NVRAM, firmware, etc.) for the given ***"fmac"*** driver implementation is written to this folder.
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")***.

2.4.3 Dynamic *“fmac”* Backport Steps Explained

This section explains the most difficult implementation piece of “***meta-murata-wireless***” – especially for individuals new to Yocto. There are two phases involved in building the ***“fmac”*** driver using “***backporttool-linux_1.0.bb”*** recipe. They are:

- **1st Phase:** Generation of “***.config***” file.

- **2nd Phase:** Generation of “***.ko***” (Kernel Object – WLAN driver) files.

Each of the two phases is broken down to three steps: input, build and output; with specific details on each step. To provide more specifics, we use the “***imx-zeus-zigra_r1.0***” release as an example. Refer to **Table 5** for more branch and release information or browse the branches/releases directly on Murata GitHub: https://github.com/murata-wireless/meta-murata-wireless.

2.4.3.1 1st Phase: Generation of “.config” file

Input:

Source file for building ***“fmac”*** driver is fetched from Murata GitHub location:
git clone https://github.com/murata-wireless/cyw-fmac.git.

Yocto framework fetch the source files and places it in the following location:

“<fsl-bsp-release-dir>/<build-dir>/tmp/work/imx6ulevk-poky-linux-gnueabi/backporttool-linux/1.0-r0/git”.

Build:

Key instructions:

i) the ***DEPENDS += “virtual/kernel”*** line notifies Yocto build framework to first compile Linux kernel and then compile “***backporttool-linux***” recipe.

ii) Compilation of source code is performed using native make (x86_64-linux) to produce ***“.config”*** file through the functions do_configure_prepend and do_configure_append.

Steps for compilation:

Function ***“do_configure_prepend()”*** performs the compilation and generates .config file.

Following commands are used to generate ***“.config”*** file:

CC=${BUILD_CC} oe_runmake defconfig-brcmfmac defconfig-brcmfmac

Key Parameters are explained as follows:

- “***oe-runmake***”: make compilation tool
- “***defconfig-brcmfmac***”: default configuration file for using brcmfmac (AKA ***“fmac”***)

Output:

Generated "***.config***" file can be found in: ***"<fsl-bsp-release-dir>/<build-dir>/ tmp/work/imx6ulevk-poky-linux-gnueabi/backporttool-linux/1.0-r0/git"***.

2.4.3.2 2nd Phase: Generation of ".ko" (Kernel Object – WLAN Driver) files using "backporttool-linux_1.0.bb" recipe.

Input:

All output files from 1st phase are used as input for 2nd phase.

FROM: ***"<fsl-bsp-release-dir>/<build-dir>/ tmp/work/imx6ulevk-poky-linux-gnueabi/backporttool-linux/1.0-r0/git"***

TO: ***"<fsl-bsp-release-dir>/<build-dir>/ tmp/work/imx6ulevk-poky-linux-gnueabi/backporttool-linux/1.0-r0/git"***

Build:

Steps for compilation:

The following commands are used to generate "***.ko***" files:

```
oe_runmake KERNEL_PATH=${STAGING_KERNEL_DIR}   \
           KERNEL_SRC=${STAGING_KERNEL_DIR}    \
           KERNEL_VERSION=${KERNEL_VERSION}    \
           CC="${KERNEL_CC}" LD="${KERNEL_LD}" \
           AR="${KERNEL_AR}" \
           ${MAKE_TARGETS}
```

During this stage, target toolchain (Ex: "***imx6ulevk-poky-linux-gnueabi***") is used to build the backports tool and produce necessary "***.ko***" ("***compat", "cfg80211", "brcmutil", "brcmfmac"***) driver files.

Output:

Generated "***.ko***" files are found in sub-folders of "***backporttool-linux-1.0***"*:*

- **"compat.ko"**: ***/compat/***
- ***"cfg80211.ko":*** ***/net/wireless/***
- ***"brcmutil.ko":*** ***/drivers/net/wireless/broadcom/brcm80211/brcmutil/***
- ***"brcmfmac.ko":*** ***/drivers/net/wireless/broadcom/brcm80211/brcmfmac/***

2.4.4 Staging Directories: Important Note!

"***backporttool-linux_1.0.bb***" recipe use staged kernel directory paths. i.e. It needs kernel source and kernel build output paths used by "***linux-imx***" recipe. The reason for using staged kernel directory is because "***linux-imx***" recipe makes the kernel source code and libraries available for use by other recipes through Staging. ***"linux-imx"*** recipe removes the temporary build outputs after backport recipes finishes building "***.ko***" WLAN driver files. Hence, it is recommended, that the user does not build the ***"fmac"*** driver recipes standalone.

If any changes are made to the kernel source, it is strongly recommended that the user always builds the kernel using the command, "**bitbake fsl-image-validation-imx".** This will, in turn, build ***"fmac"*** backports as well – **because** backports depends on "***linux-imx***" recipe.

3 Finally… Building i.MX Yocto Linux!

3.1 i.MX Yocto Build: Overview

Certain Intellectual Property issues (for 3rd party drivers that are integrated into the baseline NXP i.MX image) prevents direct access to i.MX image binaries with built-in "fmac" release. 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:

a) 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**.

b) 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** to **Section 3.8**.

3.2 i.MX Yocto Build: The Fast Track (In a Hurry or Linux Beginner?)

3.2.1 Install Ubuntu

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

NOTE: Murata has verified these build steps using Ubuntu 16.04 and 18.04 (x64). For more information on the Ubuntu download, please refer to this link: https://www.ubuntu.com/download/desktop. The Ubuntu installation manual is provided here.

3.2.2 Download Murata's Script Files

With Ubuntu installed, we need to get the script files downloaded. There are a couple of quick options here:

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

OR:

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

```
wget --no-check-certificate --content-disposition https://github.com/murata-wireless/meta-murata-
wireless/raw/master/script-utils/latest/README.txt
wget --no-check-certificate --content-disposition https://github.com/murata-wireless/meta-murata-
wireless/raw/master/script-utils/latest/Host_Setup_for_Yocto.sh
wget --no-check-certificate --content-disposition https://github.com/murata-wireless/meta-murata-
wireless/raw/master/script-utils/latest/Murata_Wireless_Yocto_Build.sh
chmod a+x *.sh
```

3.2.3 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 file), just go to the main folder: https://github.com/murata-wireless/meta-murata-wireless/tree/master/script-utils/latest. The "latest" folder is used to maintain the most recent/up-to-date script.

Murata's script installs necessary additional packages required for the Yocto build. For additional information, refer to NXP Yocto Project User's Guide (part of NXP Reference Documents release). Murata's script will prompt user for password – as supervisory access is needed to install various packages. GIT is also configured so it can be used later during the build process. For more information on first-time GIT setup, you can refer to this link.

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

```
./Host_Setup_for_Yocto.sh
```

The script goes through the following stages:

```
1) Verifying Host Environment
2) Verifying Host Script Version
3) Installing Essential Yocto host package s
4) GIT Configuration: verifying User name and email ID
```

For an example input/output sequence, refer to Appendix C.

3.2.3.1 Configure Ubuntu for bash

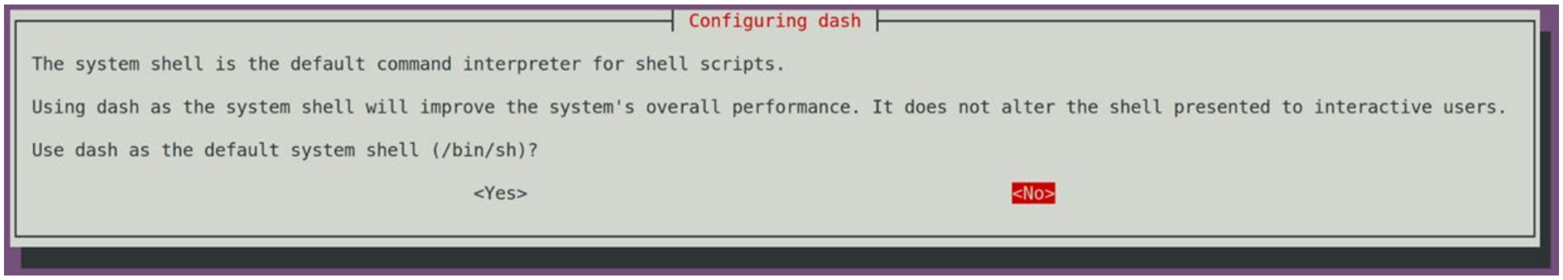
By default, Ubuntu sets the environment to use dash. It is mandatory that, User sets the default system shell to "No" when configuring dash. Follow the steps mentioned below for reconfiguring dash.

- Open "Terminal" App in Ubuntu 18.04 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 6**.

Figure 6: Configuring dash



```
Configuring dash

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

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

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

                    <Yes>                              <No>
```

3.2.4 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.sh". For plain ASCII text version, you can go to this link or just hit the "Raw" button. For more information (README file), just go to the main folder: https://github.com/murata-wireless/meta-murata-wireless/tree/master/script-utils/latest. The "latest" folder is used to maintain the most recent/up-to-date script.

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

- Installed 64-bit version of Ubuntu 18.04 (preferred), 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:

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

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

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

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

Murata's build script performs the following tasks:

- Verifies host environment (i.e. Ubuntu 18.04/16.04/14.04).
- Check to make sure script being run is latest version.
- Prompts user to select release type:
    - "**Stable**" corresponds to "***meta-murata-wireless***" release/tag (rather than branch). Murata tests wireless functionality on i.MX platforms for each release/tag. This release type is recommended for baseline image builds or initial bring-up testing.
    - "**Developer**" corresponds to a branch which can be a "moving target". When performing the automated build, the script file pulls the latest branch contents – as opposed to a specific GIT commit on that branch. If the user wants the latest fixes, then this is the best option to go with. **NOTE**: Murata only runs "spot" tests before submitting fixes/enhancements to the branch.
- Select the 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 have to** create additional folders.
- Select the wireless solution. For newer kernel releases, Murata provides Wi-Fi support for both Cypress/Infineon and NXP chipset-based modules. For older kernels, this option is not provided as only Cypress/Infineon based chipsets are supported.
- Select the "***fmac***" release. The script displays both the "***fmac***" codename and latest kernel version supported by that release. Currently, three "***fmac***" releases are supported: "***manda***", "***kong***" and "***zigra***" (most recent and up to date regarding fixes and enhancements). Murata strongly recommends using "***zigra***" release.
- Select i.MX target: refer to **Section 3.3.1** or **Table 9** for more details.
- Regarding the "**non-UHS**" option, this forces the SDIO mode to not go "Ultra High Speed" or switch to SDIO 3.0 mode (even if the target supports it).
- Select the "DISTRO and image". This configures the graphical driver and Yocto image. For more details, reference 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 final configuration and accept before moving forward.
- Accept the NXP/Freescale End User's License Agreement (EULA). There is 3rd party IP included in the i.MX Yocto build. This step addresses the sub-licensing issue. During this step the user must review a fair bit of legal documentation (by repeatedly entering space bar) or if already familiar with the EULA language, enter 'q' to bypass displaying the complete agreement. The final step of this EULA step prompts the user to enter "y" to accept.
- Last and final step is to confirm that user want to kick off 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**"):

```
./Murata_Wireless_Yocto_Build.sh
```

The script goes through the following stages:

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

For an example input/output sequence, refer to Appendix D.

3.3 i.MX Yocto Build: Manual Steps (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 Image Build Configuration with i.MX Target Selection

To figure out the Murata wireless modules supported on a given NXP i.MX Platform and kernel version, refer to **Table 4: NXP i.MX EVK Part Number / Yocto (MACHINE) target / Kernel Version Matrix** and **Table 8: NXP i.MX EVK Part Number / Murata Module Interconnect**.
Refer to **Table 9** for the supported targets: hardware interconnect, DTB file, interrupt configuration, and required 3.3V VIO is listed. Refer to the Linux User Guide and Hardware User Manual for more details on hardware interconnect.

Table 8 shows Murata module interconnect on various NXP i.MX Reference Platforms. Cypress chipset for each module is displayed. For a given i.MX platform and Murata module, you can quickly look up the compatibility. Details on the terminology used in the table are provided below.

- "**NC**" means "No Connect". This is due to one or both of the following reasons:
    - VIO incompatible: Wi-Fi/BT M.2 Module requires VIO voltage level that the NXP i.MX Hardware cannot provide.
    - Physical bus (i.e. SDIO, PCIe, UART) and/or WLAN/Bluetooth control line interconnect is not available.
- "**uSD-M.2**": Murata's uSD-M.2 Adapter provides interconnect to the Embedded Artists' Wi-Fi/BT M.2 Module. NXP i.MX Host WLAN SDIO is configured for 1.8V VIO default. The Bluetooth UART, and WLAN/BT control signals from i.MX Host are configured at 3.3V VIO (hardware limitation due to fixed voltage rails). Rev B1 Adapter level shifts the BT UART and some of the WLAN/BT control signals. Although Rev A Adapter does not level shift BT UART (and some WLAN/BT control signals), it can still be used where shown for the i.MX 6UL and i.MX 6ULL EVK's.
- "**uSD-M.2^{+}**": Murata's uSD-M.2 Adapter (Rev B1) provides interconnect to the Embedded Artists' Wi-Fi/BT M.2 Module. However, additional cabling to connect Bluetooth UART and WLAN/BT control signals is required for NXP i.MX 8M Mini EVK and 8M Nano EVK. The cable (Jumper Wire F/F 6") is easily obtained through Distribution channel (example Digi-Key part numbers 1568-1644-ND or 1568-1513-ND). See Linux User Guide for more details.
- "**uSD-M.2-3.3V**": Murata's uSD-M.2 Adapter provides interconnect to the Embedded Artists' Wi-Fi/BT M.2 Module. NXP i.MX Host WLAN SDIO is configured for (fixed at) 3.3V VIO. The Bluetooth UART, and WLAN/BT control signals from i.MX Host are configured at 3.3V VIO. Either Rev B1 or Rev A uSD-M.2 Adapter can be used in this case – with correct jumper setting for 3.3V override mode.
- "**M.2**": Only Embedded Artists' Wi-Fi/BT M.2 EVB is required. The NXP i.MX EVK has a M.2 connector. Currently only WLAN-PCIe, BT-UART, and WLAN/BT CTRL signals are brought out on the M.2 connectors for these specific NXP i.MX EVK's. As such, ***only*** Wi-Fi/BT M.2 EVB's which support WLAN-PCIe (i.e. 1CX and 1XA) can be used.
- "**M.2^{W}**": Only Embedded Artists' Wi-Fi/BT M.2 EVB is required. The NXP i.MX EVK has a M.2 connector. Currently only WLAN-PCIe, and WLAN CTRL signals are brought out on the M.2 connectors for these specific NXP i.MX EVK's. As such, there is no Bluetooth support – only WLAN.

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

Table 8: NXP i.MX EVK Part Number / Murata Module Interconnect

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

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

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

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

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

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

MD = Murata Module is soldered down

NC = No Connection options available

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

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

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

3.3.2 Host PC Preparation

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

NOTE: Murata has verified these build steps using Ubuntu 16.04 and 18.04 (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 Freescale Yocto Project Community BSP recipe layers.

In this example, we are targeting Linux 5.4.47_2.2.0 release. Execute the following repo commands:

```
repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-zeus -m imx-5.4.47-2.2.0.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 5.4.47_2.2.0, 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
```

NOTE: 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:

```
$ export BUILD_DIR=`pwd`    🡐 in this example "~/murata-imx-bsp/build-imx6ulevk-fb"
```

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

Clone ***“meta-murata-wireless”*** git and checkout “***imx-zeus-zigra_r1.0***” release (Linux 5.4.47):

```
cd $BSP_DIR/sources
git clone  https://github.com/murata-wireless/meta-murata-wireless.git
cd meta-murata-wireless
git checkout imx-zeus-zigra_r1.0
```

“meta-murata-wireless” contains the following (refer to **Section 2.4.2** 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
```

NOTE: “***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

Your configuration files at build-imx6ulevk-fb have not been touched.   🡐 misleading statement[7]
BSPDIR=
BUILD_DIR=.
meta-freescale directory found

CORRECTION: Murata modified the following files  🡐 Additional Murata logging to clarify file modifications.
  - bblayers.conf present in <BUILD_DIR>/conf
  - local.conf present in <BUILD_DIR>/conf
  - imx6ulevk.conf present in sources/meta-freescale/conf/machine
  - imx6ull14x14evk.conf present in sources/meta-imx/meta-bsp/conf/machine
  - imx6sxsabresd.conf ./sources/meta-freescale/conf/machine
  - imx6qsabresd.conf ./sources/meta-imx/meta-bsp/conf/machine
  - imx6qpsabresd.conf ./sources/meta-imx/meta-bsp/conf/machine
  - imx6dlsabresd.conf ./sources/meta-imx/meta-bsp/conf/machine
  - imx8mnevk.conf ./sources/meta-imx/meta-bsp/conf/machine
  - imx8_all.conf ./sources/meta-imx/meta-bsp/conf/machine
  - imx8mnddr4evk.conf ./sources/meta-imx/meta-bsp/conf/machine
  - imx8mnlpddr4evk.conf ./sources/meta-imx/meta-bsp/conf/machine
  - layer.conf ./sources/meta-imx/meta-bsp/conf
```

```
Murata-Wireless setup complete. Create an image with:
```

```
   $ bitbake fsl-image-validation-imx  🡐 no errors logged and here is the bitbake command
```

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 += " hostap-conf hostap-utils hostapd murata-binaries iperf3 backporttool-linux kernel-modules-pcie8997 linux-firmware-pcie8997 kernel-modules-sdio8997 "
CORE_IMAGE_EXTRA_INSTALL += " bluez5 bluez5-noinst-tools bluez5-obex openobex obexftp glibc-gconv-utf-16 glibc-utils cyw-supplicant python3"
```

[7] "$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".

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.
- SDIO in-band (default) and OOB IRQ (WL_HOST_WAKE) both supported in DTB files. Refer to Hardware User Manual on OOB IRQ rework required to make this configuration functional. To configure for OOB IRQ, choose the "***m2-oob***" version of DTB file.
- 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-zeus -m imx-5.4.47-2.2.0.xml
repo sync
```

3.4.2 Configuring the i.MX Target

```
cd $BSP_DIR    🡐 in this example "~/murata-imx-bsp"
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

```
export BUILD_DIR=`pwd`    🡐 in this example "~/murata-imx-bsp/build-imx6ulevk-fb"
cd $BSP_DIR/sources
git clone  https://github.com/murata-wireless/meta-murata-wireless.git
cd meta-murata-wireless
git checkout imx-zeus-zigra_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 “5.4.47” with FMAC “Zigra”

3.5.1 For i.MX8 platform

```
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-zeus -m imx-5.4.47-
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-zeus-zigra_r1.0

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

cd $BSP_DIR/sources/meta-murata-wireless/recipes-kernel/linux
cp linux-imx_5.4.bbappend.8MQ linux-imx_5.4.bbappend

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

3.5.2 For i.MX6/7 platforms

```
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-zeus -m imx-5.4.47-
2.2.0.xml
repo sync
cd $BSP_DIR
DISTRO=fsl-imx-wayland MACHINE=imx6ulevk source imx-setup-release.sh -b build-imx6ulevk-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-zeus-zigra_r1.0
```

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

cd $BUILD_DIR

bitbake core-image-base
```

3.6 i.MX Yocto Build: Manual Steps “Take 2” for “4.14.98” with FMAC “Zigra”

3.6.1 For i.MX8 platform

```
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-sumo -m imx-4.14.98-
2.0.0_ga.xml
repo sync

cd $BSP_DIR
DISTRO=fsl-imx-wayland MACHINE=imx8mqevk source ./fsl-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-sumo-zigra_r1.0

cd $BSP_DIR
chmod 777 sources/meta-murata-wireless/add-murata-layer-script/add-murata-wireless.sh
sh ./sources/meta-murata-wireless/add-murata-layer-script/add-murata-wireless.sh build-imx8mqevk-
wayland

cd $BSP_DIR/sources/meta-murata-wireless/recipes-kernel/linux
cp linux-imx_4.14.98.bbappend.8MQ linux-imx_4.14.98.bbappend
```

COMMENT: Renaming recipes files kernel-module-qca6174_2.0.bb and kernel-module-qca9377_2.0.bb, to kernel-module-qca6174_2.0.**bbx** and kernel-module-qca9377_2.0.**bbx**
This will disable Qualcomm CLD recipe files.

```
mv $BSP_DIR/sources/meta-fsl-bsp-release/imx/meta-bsp/recipes-kernel/kernel-modules/kernel-module-
qca6174_2.1.bb $BSP_DIR/sources/meta-fsl-bsp-release/imx/meta-bsp/recipes-kernel/kernel-
modules/kernel-module-qca6174_2.1.bbx
```

```
mv $BSP_DIR/sources/meta-fsl-bsp-release/imx/meta-bsp/recipes-kernel/kernel-modules/kernel-module-
qca9377_2.1.bb $BSP_DIR/sources/meta-fsl-bsp-release/imx/meta-bsp/recipes-kernel/kernel-
modules/kernel-module-qca9377_2.1.bbx

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

3.6.2 For i.MX6/7 platforms

```
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-sumo -m imx-4.14.98-
2.3.0_ga.xml
repo sync
cd $BSP_DIR
DISTRO=fsl-imx-wayland MACHINE=imx6ulevk source ./fsl-setup-release.sh -b build-imx6ulevk-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-sumo-zigra_r1.0

cd $BSP_DIR
chmod 777 sources/meta-murata-wireless/add-murata-layer-script/add-murata-wireless.sh
sh ./sources/meta-murata-wireless/add-murata-layer-script/add-murata-wireless.sh build-imx6ulevk-wayland


cd $BUILD_DIR

bitbake core-image-base
```

3.7 i.MX Yocto Build: Manual Steps "Take 2" for "4.9.123" with FMAC "Zigra"

3.7.1 For i.MX8 platform

```
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-rocko -m imx-4.9.123-
2.3.0-8mm_ga.xml
repo sync

cd $BSP_DIR
DISTRO=fsl-imx-wayland MACHINE=imx8mqevk source ./fsl-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-rocko-mini-zigra_r1.0

cd $BSP_DIR
chmod 777 sources/meta-murata-wireless/add-murata-layer-script/add-murata-wireless.sh
sh ./sources/meta-murata-wireless/add-murata-layer-script/add-murata-wireless.sh build-imx8mqevk-
wayland

cd $BSP_DIR/sources/meta-murata-wireless/recipes-kernel/linux
cp linux-imx_4.9.123.bbappend.8MQ linux-imx_4.9.123.bbappend
```

COMMENT: Renaming recipes files kernel-module-qca6174_2.0.bb and kernel-module-qca9377_2.0.bb, to kernel-module-qca6174_2.0.**bbx** and kernel-module-qca9377_2.0.**bbx**
This will disable Qualcomm CLD recipe files.

```
mv $BSP_DIR/sources/meta-fsl-bsp-release/imx/meta-bsp/recipes-kernel/kernel-modules/kernel-module-
qca6174_2.0.bb $BSP_DIR/sources/meta-fsl-bsp-release/imx/meta-bsp/recipes-kernel/kernel-
modules/kernel-module-qca6174_2.0.bbx
mv $BSP_DIR/sources/meta-fsl-bsp-release/imx/meta-bsp/recipes-kernel/kernel-modules/kernel-module-
qca9377_2.0.bb $BSP_DIR/sources/meta-fsl-bsp-release/imx/meta-bsp/recipes-kernel/kernel-
modules/kernel-module-qca9377_2.0.bbx
```

COMMENT: Copying necessary backport recipes and Murata binaries recipes for i.MX8 platform

```
cp -f $BSP_DIR/sources/meta-murata-wireless/freescale/backporttool-linux_1.0.bb@imx8
$BSP_DIR/sources/meta-murata-wireless/recipes-kernel/backporttool-linux/backporttool-linux_1.0.bb
cp -f $BSP_DIR/sources/meta-murata-wireless/freescale/murata-binaries_1.0.bb@imx8
$BSP_DIR/sources/meta-murata-wireless/recipes-connectivity/murata-binaries/murata-binaries_1.0.bb
cd $BUILD_DIR

bitbake fsl-image-validation-imx
```

Known Issue: Sometimes, it is possible that, you may encounter following error, when you try to create an image with "core-image-base". The error does not occur with "fsl-image-validation-imx".

```
ERROR: backporttool-linux-1.0-r0 do_make_scripts: Function failed: do_make_scripts (log file is located at
/home/jkareem/testing/imx8mmevk-bsp/build-dir/tmp/work/imx8mmevk-poky-linux/backporttool-linux/1.0-
r0/temp/log.do_make_scripts.20350)
ERROR: Logfile of failure stored in: /home/jkareem/testing/imx8mmevk-bsp/build-dir/tmp/work/imx8mmevk-
poky-linux/backporttool-linux/1.0-r0/temp/log.do_make_scripts.20350
```

Solution: continue the build with following 2 statements from <BSP_DIR>.

```
cd $BSP_DIR
source sources/base/setup-environment <BUILD_DIR>
```

3.7.2 For i.MX6/7 platforms

```
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-rocko -m imx-4.9.123-
2.3.0-8mm_ga.xml
repo sync

cd $BSP_DIR
DISTRO=fsl-imx-wayland MACHINE=imx6ulevk source ./fsl-setup-release.sh -b build-imx6ulevk-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-rocko-mini-zigra_r1.0
```

```
cd $BSP_DIR
chmod 777 sources/meta-murata-wireless/add-murata-layer-script/add-murata-wireless.sh
sh ./sources/meta-murata-wireless/add-murata-layer-script/add-murata-wireless.sh build-imx6ulevk-wayland

cd $BUILD_DIR
bitbake core-image-base
```

3.8 i.MX Yocto Build: Manual Steps “Take 2” for “4.1.15” with FMAC “manda”

3.8.1 For i.MX6/7 platforms

```
cd ~
mkdir murata-imx-bsp
cd murata-imx-bsp
export BSP_DIR=`pwd`
repo init -u https://source.codeaurora.org/external/imx/fsl-arm-yocto-bsp.git -b imx-4.1-krogoth -m imx-
4.1.15-2.0.0.xml
repo sync

cd $BSP_DIR
DISTRO=fsl-imx-x11 MACHINE=imx6ulevk source ./fsl-setup-release.sh -b build-imx6ulevk-x11
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-krogoth-manda_r2.2
cd $BSP_DIR
chmod 777 sources/meta-murata-wireless/add-murata-layer-script/add-murata-wireless.sh
sh ./sources/meta-murata-wireless/add-murata-layer-script/add-murata-wireless.sh build-imx6ulevk-x11

cd $BUILD_DIR

bitbake core-image-base
```

4 Bluetooth Patch file Considerations

The Murata customized build configures example Bluetooth patch files ***("*.hcd")*** in the "***/etc/firmware***" folder. Like WLAN, the file naming convention is very important. The default Bluetooth stack implementation is BlueZ. When the "hciattach" command is invoked, the BT core is interrogated for the Chip ID. That Chip ID is then used to select the correct Bluetooth patch file to download.

All Cypress customized Bluetooth patch files start with a "***CYW***" string which is changed by "***meta-murata-wireless***" to "***BCM***" (during file system build) for the "hciattach" call to work correctly. You can see a listing of the Bluetooth patch files at https://github.com/murata-wireless/cyw-bt-patch. Here is an example (Bluetooth bring-up/scan) sequence if you are using Type 1MW:

⇨ First boot the platform and login as "root".

```
cd /etc/firmware
mv CYW4345C0.1MW.hcd BCM4345C0.1MW.hcd
hciattach /dev/ttymxc1 bcm43xx 3000000 flow -t 20 ← "/dev/ttymxc1" is correct UART for i.MX6UL(L) EVK
hciconfig hci0 up
hcitool scan
```

NOTE: After kernel boot, no toggling of BT_REG_ON is necessary given the "modem" construct in the i.MX DTS file. i.e. The kernel boot sequence should guarantee correct toggling of BT_REG_ON during power on, so that the Bluetooth core is in the correct state.

5 Using "fmac" driver

This section is intended to provide the user with a quick "how to" when using ***"fmac"*** driver for the first time.

5.1 How-To on "fmac" driver

Here is a quick overview on expected output when platform is correctly configured for Murata-Cypress 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).
- Correct WLAN firmware, CLM blob (if applicable), and WLAN NVRAM files in "***/lib/firmware/cypress***" folder.
- 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.

5.1.1 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 "cyw" string as parameter
  switch_module.sh cyw

```
switch_module.sh cyw
reboot
```

Look for the following message after entering ***switch_module.sh cyw*** Command.

```
Setting up for Cypress
```

- Reboot by entering the command, ***"reboot"***
- m.2 board should now have been detected and the correct kernel modules gets loaded.

5.1.2 Successful *"fmac"* load message

The ***"fmac"*** driver consists of loadable modules. If the "***dtb***" file enables WLAN interface and the Murata WLAN hardware is connected, then the ***"fmac"*** driver should automatically load during kernel boot. Look for the following key logging messages:

```
brcmf_c_preinit_dcmds: Firmware: BCM43012/2 wl0: May 26 2020 00:53:41 version 13.10.271.245 (fdac85a)
FWID 01-3e4c636c
```

5.1.3 *"wlan0"* initialization

After the kernel boots, the "***wlan0***" interface can be initialized by one of the following methods.

- "***ifconfig***" command:

```
ifconfig wlan0 up
```

- WPA supplicant: First bring up WPA supplicant and then invoke "***wpa_cli***":

```
wpa_supplicant -i wlan0 -D nl80211 -c /etc/wpa_supplicant.conf -B
wpa_cli -i wlan0
```

Refer to the Linux User Guide for more details on bringing up WLAN or Bluetooth and testing it.

6 Building Linux Kernel with FMAC Standalone

DISCLAIMER: This is for customer reference only. Murata only supports our "meta-murata-wireless" solution that provides a custom build within Yocto.

The following steps show how to build "fmac" driver standalone. Linux kernel version 4.14.52 with "manda" version of "fmac" driver is used as an example.

1) Download Linux stable kernel (v4.14.52) source using following command.

```
wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.14.52.tar.gz
tar zxvf linux-4.14.52.tar.gz
cd linux-4.14.52
```

2) Fetch Cypress patch files (cypress-patch-v4.14.52-2018_0928.tar.gz) for FMAC from the following public link (https://community.cypress.com/docs/DOC-15932). The patch file is available inside the downloaded zip file.

3) Copy the file (cypress-patch-v4.14.52-2018_0928.tar.gz) to the folder "linux-4.14.52".

4) Untar the file, cypress-patch-v4.14.52-2018_0928.tar.gz. Patch files will be placed in the folder, "cypress-patch".

5) In "linux-4.14.52" folder, apply cypress patches with below bash commands

```
for i in cypress-patch/*.patch; do patch -p1 < $i; done
```

6) Source the necessary toolchain for arm 32bit/64bit.

```
source /opt/fsl-imx-x11/4.9.88-2.0.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```

7) Use the default defconfig (imx_v6_v7_defconfig) to create .config file.

```
make imx_v6_v7_defconfig
```

8) Set kernel .config and enable below options, then compile kernel image

```
CONFIG_BRCMUTIL=y
CONFIG_BRCMFMAC=y
CONFIG_BRCMFMAC_PROTO_BCDC=y
CONFIG_BRCMFMAC_SDIO=y
CONFIG_BRCMFMAC_PCIE=y
CONFIG_BRCMFMAC_PROTO_MSGBUF=y   ← Add this macro manually
```

9) Build the new kernel image using the command.

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

10) Create modules using the following command.

```
make modules_install INSTALL_MOD_PATH=/home/<user-name>/tempModules
```

11) Copy kernel to the boot folder. Copy the contents from "tempModules/lib/Modules" to rootfs (/lib/modules/).

12) Boot the system with the new kernel and modules

7 Applying Cypress Patches to Hostapd and WPA Supplicant

This section describes the building of "hostapd" and "wpa_supplicant" packages for "kong" release.

1) Download the source files for hostap_2_9.

```
wget https://w1.fi/cgit/hostap/snapshot/hostap_2_9.tar.gz
tar zxvf hostap_2_9.tar.gz
cd hostap_2_9
```

2) Fetch Cypress patch file tarball (*cypress-hostap_2_9-2020_0115.tar.gz*) for Integrated Hostapd + wpa_supplicant from the public link https://community.cypress.com/docs/DOC-19000 (inside the downloaded zip file). Place it in the "hostap_2_9" folder.

3) Apply all the patches.

```
tar zxvf cypress-hostap_2_9-2020_0115.tar.gz
for i in cypress-hostap_2_9/*.patch; do patch -p1 < $i; done
```

----sample output---

```
patching file wpa_supplicant/wpa_supplicant.c
patching file wpa_supplicant/rrm.c
patching file src/drivers/driver_nl80211_event.c
patching file wpa_supplicant/wpas_glue.c
patching file src/drivers/nl80211_copy.h
patching file src/drivers/driver.h
patching file src/drivers/driver_nl80211_capa.c
patching file src/drivers/driver.h
patching file src/drivers/driver_nl80211.c
patching file wpa_supplicant/wpa_supplicant.c
patching file src/crypto/tls_openssl.c
patching file src/drivers/nl80211_copy.h
patching file src/drivers/driver.h
patching file src/drivers/driver_nl80211.c
patching file src/drivers/driver_nl80211_capa.c
patching file src/ap/beacon.c
patching file src/ap/hostapd.c
patching file src/ap/wpa_auth.c
patching file src/ap/wpa_auth.h
patching file src/ap/wpa_auth_glue.c
patching file src/drivers/driver.h
patching file src/drivers/driver_nl80211.c
patching file src/ap/beacon.c
patching file src/ap/wpa_auth.h
patching file src/ap/wpa_auth_glue.c
patching file src/ap/wpa_auth_ie.c
patching file src/ap/wpa_auth_glue.c
```

4) Source appropriate toolchain. Example shown below.

```
source /opt/fsl-imx-x11/4.9.11-1.0.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```

5) Build hostapd

```
cd hostapd
cp defconfig .config
make clean
make
```

--sample output—

```
 LD  hostapd
 CC  hostapd_cli.c
 CC  ../src/common/wpa_ctrl.c
 CC  ../src/common/cli.c
 CC  ../src/utils/edit_simple.c
 LD  hostapd_cli
```

6) Build wpa_supplicant

```
cd ../wpa_supplicant
cp defconfig .config
make clean
make
```

-----sample output—

```
LD  wpa_supplicant
CC  wpa_cli.c
CC  ../src/common/wpa_ctrl.c
CC  ../src/common/cli.c
CC  ../src/utils/edit_simple.c
```

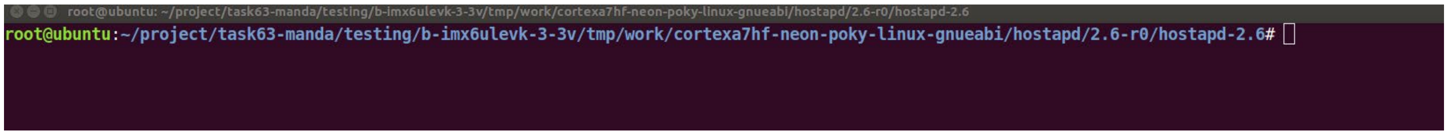
```
LD  wpa_cli
CC  wpa_passphrase.c
LD  wpa_passphrase
```

8 Building Hostapd through Yocto

1) After you have created a SD card image through the script or by any of the methods, enter the following command from the build directory.

```
<build-dir>$ bitbake -c devshell hostapd
```
[8]

This will open another new shell window as shown below.



```
root@ubuntu:~/project/task63-manda/testing/b-imx6ulevk-3-3v/tmp/work/cortexa7hf-neon-poky-linux-gnueabi/hostapd/2.6-r0/hostapd-2.6#
```

2) User can build "hostapd" using the new shell.

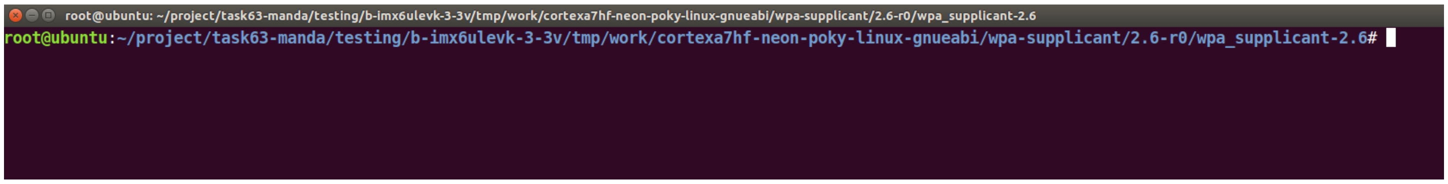
```
cd hostapd
make clean
make
```

9 Building WPA Supplicant through Yocto

1) After you have created a SD card image through the script or by any of the methods, enter the following command from the build directory. Example shown below.

```
<build-dir>$ bitbake -c devshell wpa-supplicant
```
[8]

NOTE: This will open another new shell window as shown below.



```
root@ubuntu:~/project/task63-manda/testing/b-imx6ulevk-3-3v/tmp/work/cortexa7hf-neon-poky-linux-gnueabi/wpa-supplicant/2.6-r0/wpa_supplicant-2.6#
```

[8] **NOTE:** You might have to source the setup-environment file to set up the build environment, if not done already.
<linux-imx>$ source setup-environment <build_dir>

2) User can build "wpa_supplicant" using the new shell.

```
cd wpa_supplicant
make clean
make
```

10 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/. 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.

Since Murata launched uSD-M.2 Adapter, we also provide software support for this new hardware, which is done via new set of dtb files (a dtb file is the "compiled" binary version of a dts file; this is the file that is actually used by the kernel). In addition to the standard dtb files, you can find dtb files that end with btwifi-m2-oob, or btwifi-m2 in Murata's customized images. These dtb files are developed to work with uSD-M.2 Adapter and are verified on several platforms that support uSD-M.2 Adapter, i.e. imx6ulevk, imx6ull14x14evk, imx6sxsabresd, imx6qsabresd, imx8mmevk, imx8mnevk. The following figures shows how these dts files are organized in the hierarchy structures.

Figure 7: DTS hierarchy for imx6ulevk

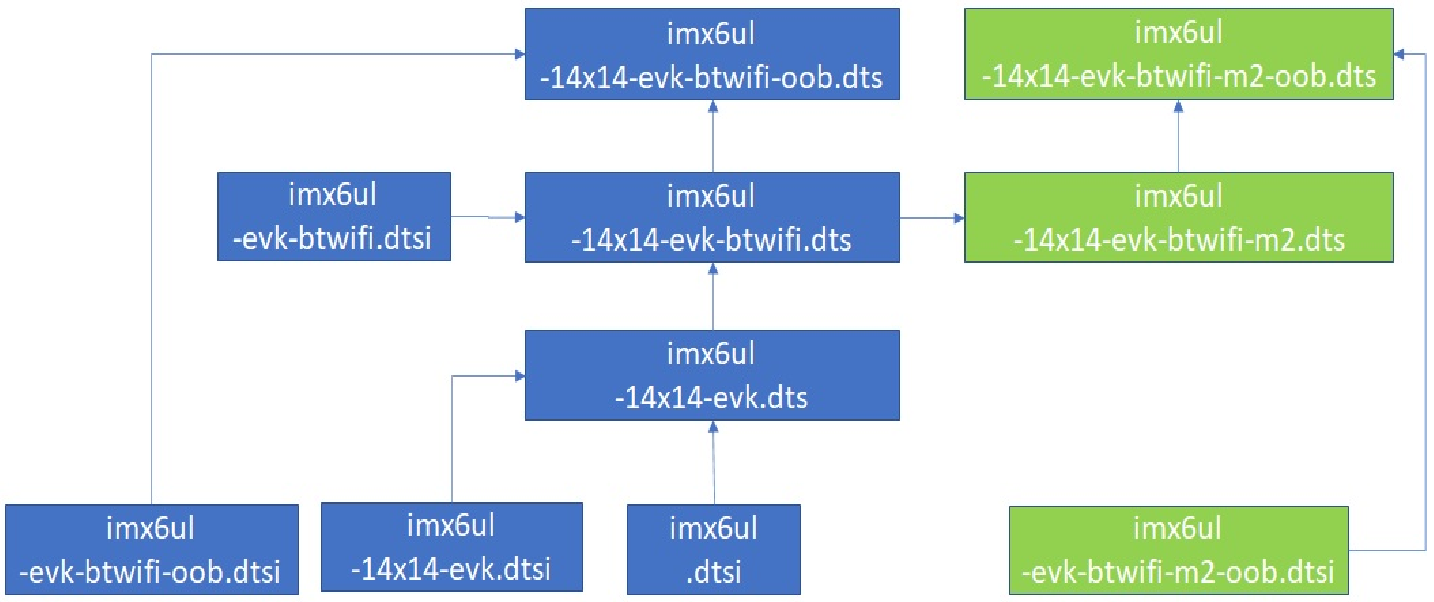

imx6ul -14x14-evk-btwifi-oob.dts
imx6ul -14x14-evk-btwifi-m2-oob.dts
imx6ul -evk-btwifi.dtsi
imx6ul -14x14-evk-btwifi.dts
imx6ul -14x14-evk-btwifi-m2.dts
imx6ul -14x14-evk.dts
imx6ul -evk-btwifi-oob.dtsi
imx6ul -14x14-evk.dtsi
imx6ul .dtsi
imx6ul -evk-btwifi-m2-oob.dtsi

Figure 8: DTS hierarchy for imx6ull14x14evk

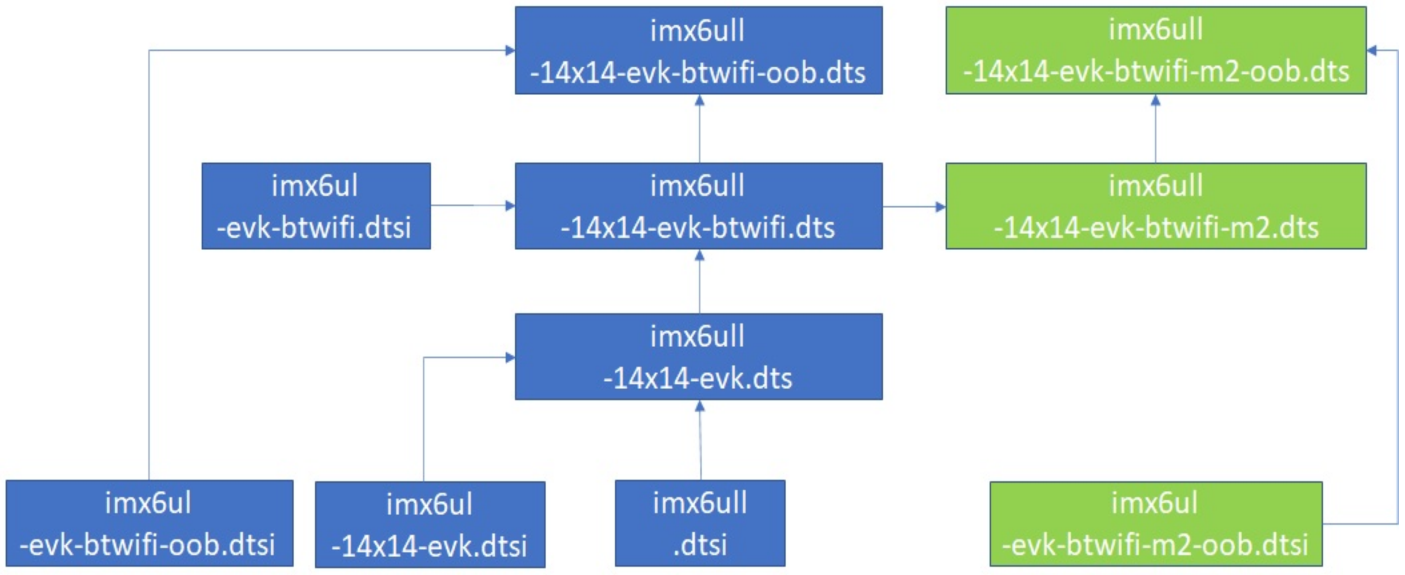

imx6ull -14x14-evk-btwifi-oob.dts
imx6ull -14x14-evk-btwifi-m2-oob.dts
imx6ul -evk-btwifi.dtsi
imx6ull -14x14-evk-btwifi.dts
imx6ull -14x14-evk-btwifi-m2.dts
imx6ull -14x14-evk.dts
imx6ul -evk-btwifi-oob.dtsi
imx6ul -14x14-evk.dtsi
imx6ull .dtsi
imx6ul -evk-btwifi-m2-oob.dtsi

Figure 9: DTS hierarchy for imx6sxevk

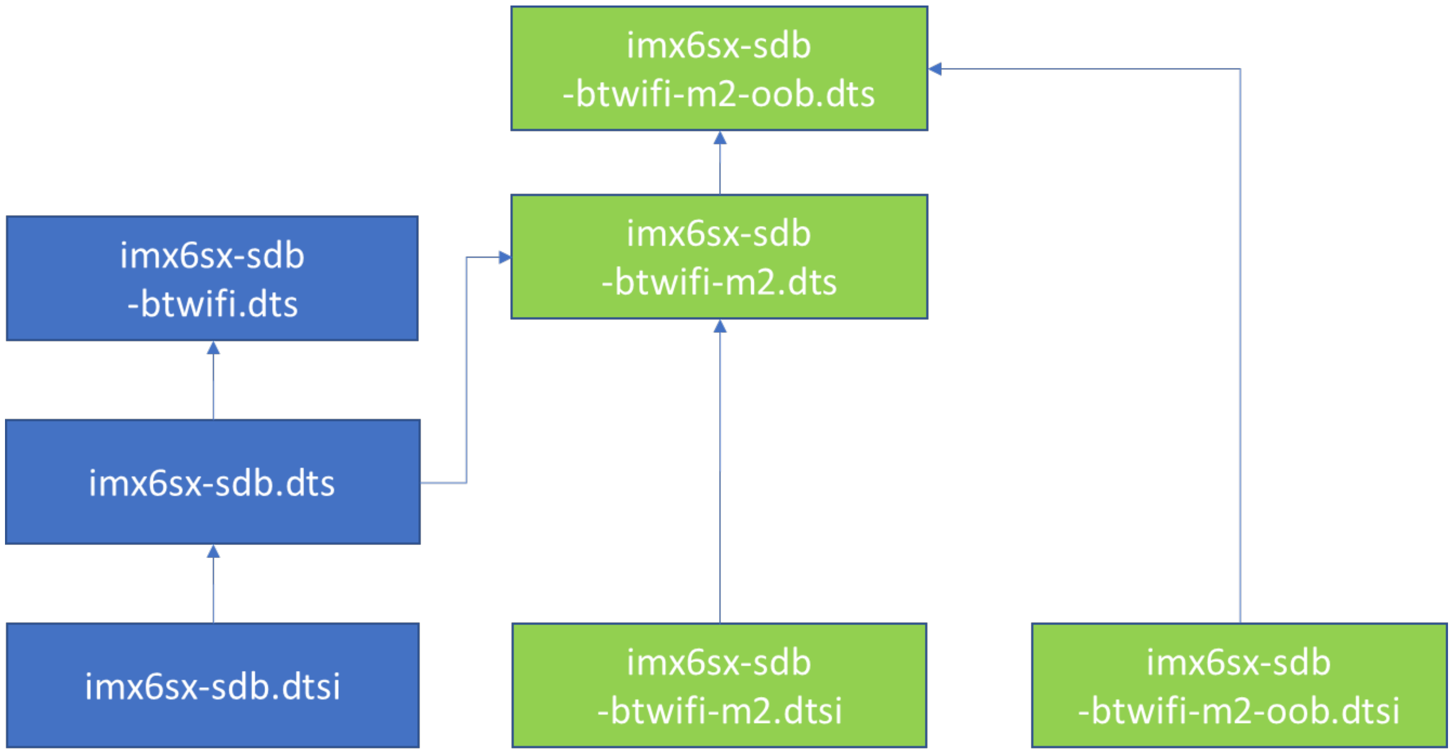

imx6sx-sdb -btwifi-m2-oob.dts
imx6sx-sdb -btwifi.dts
imx6sx-sdb -btwifi-m2.dts
imx6sx-sdb.dts
imx6sx-sdb.dtsi
imx6sx-sdb -btwifi-m2.dtsi
imx6sx-sdb -btwifi-m2-oob.dtsi

Figure 10: DTS hierarchy for imx6qsabresd

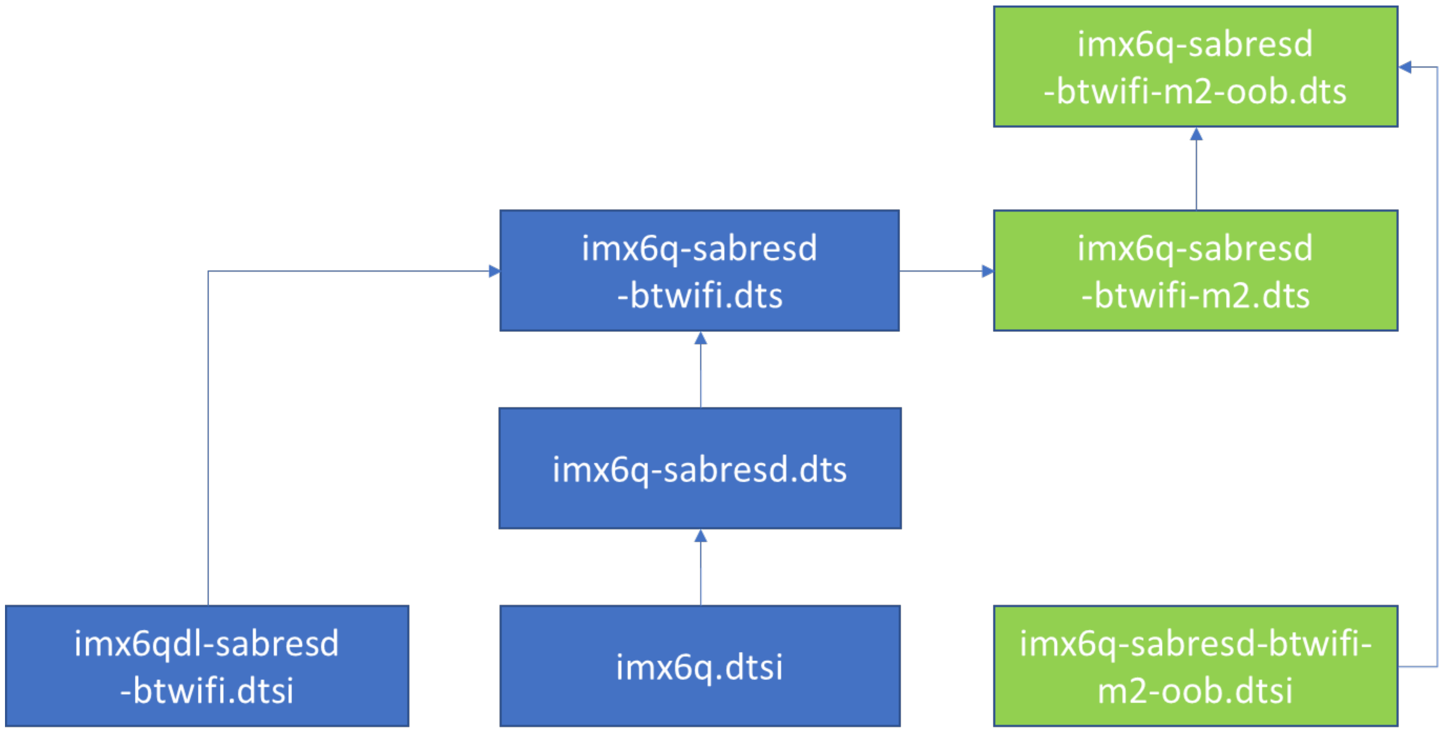

imx6q-sabresd -btwifi-m2-oob.dts
imx6q-sabresd -btwifi.dts
imx6q-sabresd -btwifi-m2.dts
imx6q-sabresd.dts
imx6qdl-sabresd -btwifi.dtsi
imx6q.dtsi
imx6q-sabresd-btwifi-m2-oob.dtsi

Figure 11: DTS hierarchy for imx8mmevk

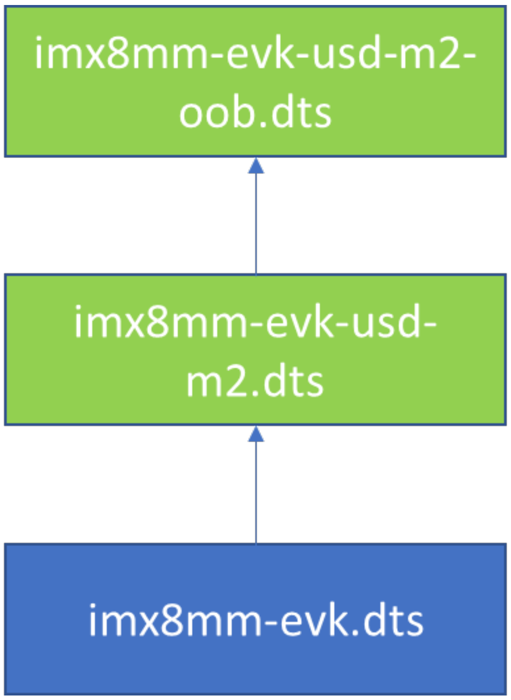
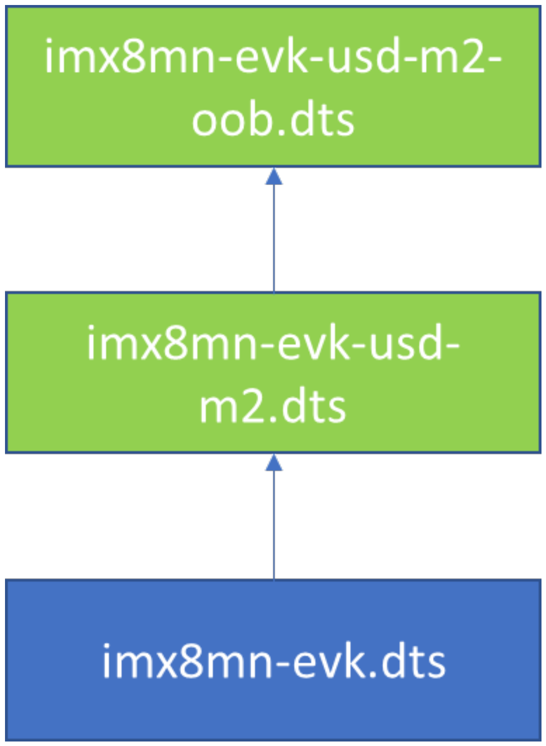


Figure 12: DTS hierarchy for imx8mnevk



11 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 board (EVB). **Figure 13** shows that the carrier board can work with a variety of NXP i.MX6/7/8 COM boards and four different Murata based EVBs. With this platform, users can easily evaluate multiple processors against multiple 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 see the real potential of Murata's module.
Figure 13 shows how this platform works with COM boards and M.2 Wi-Fi/BT EVBs.

Figure 13: Combine i.MX COM with Wi-Fi/BT M.2 EVB

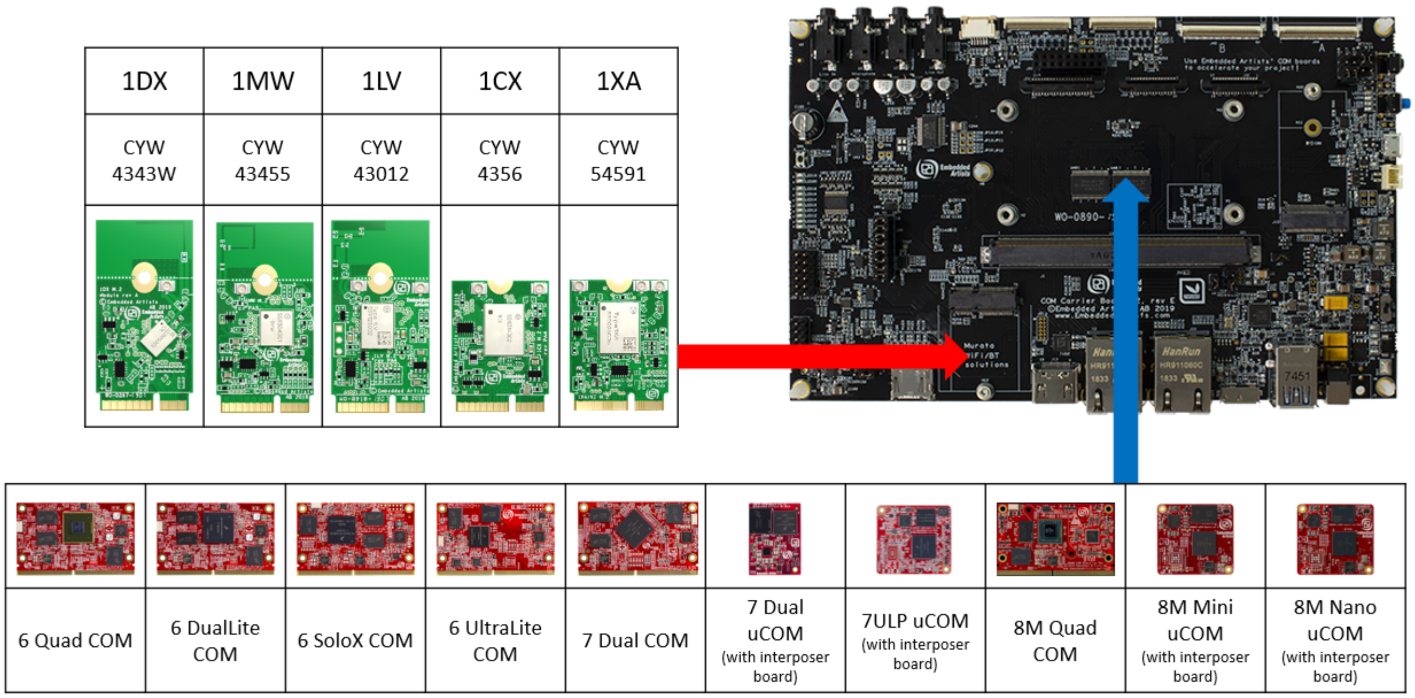

1DX
1MW
1LV
1CX
1XA
CYW 4343W
CYW 43455
CYW 43012
CYW 4356
CYW 54591
6 Quad COM
6 DualLite COM
6 SoloX COM
6 UltraLite COM
7 Dual COM
7 Dual uCOM (with interposer board)
7ULP uCOM (with interposer board)
8M Quad COM
8M Mini uCOM (with interposer board)
8M Nano uCOM (with interposer board)

Table 10 provides an i.MX Reference Platform versus Murata module compatibility matrix. An additional column is included to provide a quick Cypress chipset lookup. Beside the well-designed hardware, Embedded Artists also provides all the document you need on their website. **Table 11** provides the list of landing pages you might need during the evaluation process.

Table 10: Embedded Artists' i.MX InterConnect

EA i.MX (u)COM	1DX	1MW	1LV	1CX	1XA
	CYW4343W	CYW43455	CYW43012	CYW4356	CYW54591
iMX8M Quad COM	M.2	M.2	M.2	M.2	M.2
iMX8M Mini uCOM	M.2	M.2 / MD	M.2	M.2	M.2
iMX8M Nano uCOM	M.2	M.2 / MD	M.2	NC	NC
iMX7 Dual COM	M.2	M.2	M.2	M.2	M.2
iMX7 Dual uCOM	M.2	M.2	M.2	M.2	M.2
iMX7ULP uCOM	M.2	M.2	M.2 / MD	NC	NC
iMX6 Quad COM	M.2	M.2	$M.2^{v}$	M.2	M.2
iMX6 DualLite COM	M.2	M.2	$M.2^{v}$	M.2	M.2
iMX6 SoloX COM	M.2	M.2	$M.2^{v}$	M.2	M.2
iMX6 UltraLite COM	M.2	M.2	M.2	NC	NC
M.2 = Works with onboard M.2 slot. MD = Module soldered down. NC = No Connect. $M.2^{V}$ = These platforms have fixed 3.3 V VIO for WLAN-SDIO. Although 1LV is 1.8V only, testing has yielded reliable results.					

Table 11: Embedded Artists' Landing Pages

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

Table 12 includes the links to the datasheets and schematics of COM boards and the EVBs.

Table 12: Embedded Artists' Datasheets and Schematics

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

Table 13 provides the link to the required manual documents and the software.

Table 13: Embedded Artists' User Manuals and Software

User Manuals and Software	Notes
Getting Started With M.2 Modules and i.MX 6/7/8	Comprehensive document covering all major topics associated with using Wi-Fi/BT M.2 EVB's on EA's i.MX 6/7/8 Dev Kits.
i.MX Working with Yocto	Comprehensive guide on building Linux images using Yocto framework.
Linux i.MX Images Download	Pre-compiled images using "uuu" tool: allows users to easily flash i.MX platforms with latest Linux images with integrated Wi-Fi/BT support.
Wi-Fi/BT M.2 EVB Primer	Introduction and drill-down on M.2 interface.

12 Dynamic Backporting

The backporting steps are documented in Cypress' README file in their tar ball release. To make it clearer, the whole process is explained in detail below. Cypress' Varan fmac release is used as an example (i.e. backport fmac driver to Linux Kernel 4.9.88). The hardware configuration for this example is NXP's i.MX6ULL EVK with Murata uSD-M.2 Adapter and Embedded Artists' 1MW M.2 EVB.

12.1 Install stand-alone toolchain for cross-compiling

1) Create a build directory and get the BSP

```
$ mkdir imx-yocto-bsp
$ cd imx-yocto-bsp
$ repo init -u https://source.codeaurora.org/external/imx/imx-manifest -b imx-linux-rocko -m imx-4.9.88-2.0.0_ga.xml
$ repo sync
```

----sample output---

```
…
remote: Total 135491 (delta 85), reused 149 (delta 75), pack-reused 135309
Fetching projects:  77% (7/9) meta-openembeddedremote: Enumerating objects: 6048, done.
remote: Total 6048 (delta 0), reused 0 (delta 0), pack-reused 6048
Fetching projects:  88% (8/9) meta-freescale-3rdpartyremote: Total 481470 (delta 360371), reused 481006 (delta 360028)
Fetching projects: 100% (9/9), done.
Checking out projects: 100% (9/9), done.
repo sync has finished successfully.
```

2) Configure the build

```
$ DISTRO=fsl-imx-x11 MACHINE=imx6ull14x14evk source fsl-setup-release.sh -b build-imx6ull14x14evk-x11
```

3) Accept EULA by entering y

```
1.4. "Software Content Register" means the documentation accompanying the

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

----sample output---

```
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

Your build environment has been configured with:

   MACHINE=imx6ull14x14evk
   SDKMACHINE=i686
   DISTRO=fsl-imx-x11
   EULA=
BSPDIR=
BUILD_DIR=.
meta-freescale directory found
```

4) Build the image for populating toolchain

```
$ bitbake meta-toolchain
```

----sample output---

```
NOTE: Your conf/bblayers.conf has been automatically updated.
Parsing recipes: 100%
|#########################################################################################
##########################################################################| Time: 0:00:37
Parsing of 2423 .bb files complete (0 cached, 2423 parsed). 3264 targets, 218 skipped, 8 masked, 0
errors.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION          = "1.36.0"
```

```
BUILD_SYS           = "x86_64-linux"
NATIVELSBSTRING      = "ubuntu-16.04"
TARGET_SYS          = "arm-poky-linux-gnueabi"
MACHINE             = "imx6ull14x14evk"
DISTRO              = "fsl-imx-x11"
DISTRO_VERSION      = "4.9.88-2.0.0"
TUNE_FEATURES       = "arm armv7ve vfp thumb neon callconvention-hard cortexa7"
TARGET_FPU          = "hard"
meta
meta-poky           = "HEAD:0ec241873367e18f5371a3ad9aca1e2801dcd4ee"
meta-oe
meta-multimedia      = "HEAD:dacfa2b1920e285531bec55cd2f08743390aaf57"
meta-freescale       = "HEAD:49ac225a38f6d84519798e3264f2e4d19b84f70a"
meta-freescale-3rdparty = "HEAD:1d6d5961dbf82624b28bb318b4950a64abc31d12"
meta-freescale-distro = "HEAD:0ec6d7e206705702b5b534611754de0787f92b72"
meta-bsp
meta-sdk            = "HEAD:d65692ecb3a4136fc1cc137152634e8633ddb3c6"
meta-browser        = "HEAD:d6f9aed41c73b75a97d71bff060b03a66ee087b1"
meta-gnome
meta-networking
meta-python
meta-filesystems     = "HEAD:dacfa2b1920e285531bec55cd2f08743390aaf57"
meta-qt5            = "HEAD:32bb7d18a08d1c48873d7ab6332d4cc3815a4dff"

Initialising tasks: 100%
|###########################################################################################
#######################################################################| Time: 0:00:10
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
WARNING: chrpath-native-0.16-r0 do_fetch: Failed to fetch URL
https://alioth.debian.org/frs/download.php/file/3979/chrpath-0.16.tar.gz, attempting MIRRORS if
available
WARNING: bzip2-native-1.0.6-r5 do_fetch: Checksum mismatch for local file
/home/skerr/dynamic_backporting/imx-yocto-bsp/downloads/bzip2-1.0.6.tar.gz
Cleaning and trying again.
WARNING: bzip2-native-1.0.6-r5 do_fetch: Renaming /home/skerr/dynamic_backporting/imx-yocto-
bsp/downloads/bzip2-1.0.6.tar.gz to /home/skerr/dynamic_backporting/imx-yocto-
bsp/downloads/bzip2-1.0.6.tar.gz_bad-checksum_ce39c4c60f64756a23010c2718754245
WARNING: bzip2-native-1.0.6-r5 do_fetch: Checksum failure encountered with download of
http://www.bzip.org/1.0.6/bzip2-1.0.6.tar.gz - will attempt other sources if available
WARNING: popt-native-1.16-r3 do_fetch: Checksum mismatch for local file
/home/skerr/dynamic_backporting/imx-yocto-bsp/downloads/popt-1.16.tar.gz
Cleaning and trying again.
WARNING: popt-native-1.16-r3 do_fetch: Renaming /home/skerr/dynamic_backporting/imx-yocto-
bsp/downloads/popt-1.16.tar.gz to /home/skerr/dynamic_backporting/imx-yocto-
bsp/downloads/popt-1.16.tar.gz_bad-checksum_9e09d0bf772649c5ba385c48ba2d3547
```

```
WARNING: popt-native-1.16-r3 do_fetch: Checksum failure encountered with download of
http://rpm5.org/files/popt/popt-1.16.tar.gz - will attempt other sources if available
WARNING: nativesdk-libpng-1.6.31-r0 do_fetch: Failed to fetch URL
http://downloads.sourceforge.net/project/libpng/libpng16/1.6.31/libpng-1.6.31.tar.xz, attempting
MIRRORS if available
WARNING: nativesdk-shadow-4.2.1-r0 do_fetch: Failed to fetch URL http://pkg-
shadow.alioth.debian.org/releases/shadow-4.2.1.tar.xz, attempting MIRRORS if available
NOTE: Tasks Summary: Attempted 3090 tasks of which 0 didn't need to be rerun and all succeeded.

Summary: There were 9 WARNING messages shown.
```

After the build is done, `fsl-imx-x11-glibc-x86_64-meta-toolchain-cortexa7hf-neon-toolchain-4.9.88-2.0.0.sh` can be found in <build dir>/tmp/deploy/sdk. This file will be used to install the cross-compile toolchain for Linux Kernel 4.9.88.

5) Install the toolchain

```
$ cd <build dir>/tmp/deploy/sdk
$ sudo ./ fsl-imx-x11-glibc-x86_64-meta-toolchain-cortexa7hf-neon-toolchain-4.9.88-2.0.0.sh
```

If you select the default settings the toolchain will be installed in /opt/fsl-imx-x11/4.9.88-2.0.0. You can select a different folder using the option flag -d

```
$ sudo ./ fsl-imx-x11-glibc-x86_64-meta-toolchain-cortexa7hf-neon-toolchain-4.9.88-2.0.0.sh -d <target
folder path>
```

12.2 Build the kernel image

1) Get the BSP kernel source available. Clone the linux-imx repository from code aurora and checkout the imx_4.9.88_2.0.0_ga branch.

```
$ git clone https://source.codeaurora.org/external/imx/linux-imx

$ cd linux-imx

$ git checkout imx_4.9.88_2.0.0_ga
```

----sample output---

```
$ git clone https://source.codeaurora.org/external/imx/linux-imx

Cloning into 'linux-imx'...

remote: Enumerating objects: 9247082, done.

remote: Counting objects: 100% (9247082/9247082), done.

remote: Compressing objects: 100% (1355205/1355205), done.
```

```
remote: Total 9247082 (delta 7843647), reused 9240870 (delta 7837444)
Receiving objects: 100% (9247082/9247082), 1.52 GiB | 18.06 MiB/s, done.
Resolving deltas: 100% (7843647/7843647), done.
Checking connectivity... done.
Checking out files: 100% (38165/38165), done.

$ cd linux-imx/
$ git checkout imx_4.9.88_2.0.0_ga
Checking out files: 100% (61743/61743), done.
Branch imx_4.9.88_2.0.0_ga set up to track remote branch imx_4.9.88_2.0.0_ga from origin.
Switched to a new branch 'imx_4.9.88_2.0.0_ga'
```

2) Set up build environment and kernel configuration

```
$ source /opt/fsl-imx-x11/4.9.88-2.0.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
$ make imx_v7_defconfig
```

----sample output---

```
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
```

3) Edit .config to build cfg80211 as module, by making the following two changes on .config file.

```
CONFIG_CFG80211=m
CONFIG_BCMDHD=n
```

4) Set up the kernel configuration

```
$ make oldconfig
```

----sample output---

```
*
* Restart config...
*
*
* Wireless LAN
*
Wireless LAN (WLAN) [Y/n/?] y
 ADMtek devices (WLAN_VENDOR_ADMTEK) [Y/n/?] y
 Atheros/Qualcomm devices (WLAN_VENDOR_ATH) [Y/n/?] y
  Atheros wireless debugging (ATH_DEBUG) [N/y/?] n
  Atheros 802.11n wireless cards support (ATH9K) [N/m/?] n
  Atheros HTC based wireless cards support (ATH9K_HTC) [N/m/?] n
  Linux Community AR9170 802.11n USB support (CARL9170) [N/m/?] n
  Atheros mobile chipsets support (ATH6KL) [N/m/?] n
  Atheros AR5523 wireless driver support (AR5523) [N/m/?] n
  Atheros 802.11ac wireless cards support (ATH10K) [N/m/?] n
  Qualcomm Atheros WCN3660/3680 support (WCN36XX) [N/m/?] n
 Atmel devices (WLAN_VENDOR_ATMEL) [Y/n/?] y
  Atmel at76c503/at76c505/at76c505a USB cards (AT76C50X_USB) [N/m/?] n
 Broadcom devices (WLAN_VENDOR_BROADCOM) [Y/n/?] y
  Broadcom 43xx wireless support (mac80211 stack) (B43) [N/m/?] n
  Broadcom 43xx-legacy wireless support (mac80211 stack) (B43LEGACY) [N/m/?] n
  Broadcom IEEE802.11n PCIe SoftMAC WLAN driver (BRCMSMAC) [N/m/?] n
  Broadcom IEEE802.11n embedded FullMAC WLAN driver (BRCMFMAC) [N/m/?] n
 Broadcom FullMAC wireless cards support v1.141 (BCMDHD) [N/m/y/?] n
  Broadcom FullMAC wireless cards support v1.363 (BCMDHD_1363) [N/m/y/?] (NEW)
 Cisco devices (WLAN_VENDOR_CISCO) [Y/n/?] y
 Intel devices (WLAN_VENDOR_INTEL) [Y/n/?] y
```

```
Intersil devices (WLAN_VENDOR_INTERSIL) [Y/n/?] y
  IEEE 802.11 for Host AP (Prism2/2.5/3 and WEP/TKIP/CCMP) (HOSTAP) [Y/n/m/?] y
    Support downloading firmware images with Host AP driver (HOSTAP_FIRMWARE) [N/y/?] n
  Softmac Prism54 support (P54_COMMON) [N/m/?] n
Marvell devices (WLAN_VENDOR_MARVELL) [Y/n/?] y
  Marvell 8xxx Libertas WLAN driver support (LIBERTAS) [N/m/?] n
  Marvell 8xxx Libertas WLAN driver support with thin firmware (LIBERTAS_THINFIRM) [N/m/?] n
  Marvell WiFi-Ex Driver (MWIFIEX) [N/m/?] n
MediaTek devices (WLAN_VENDOR_MEDIATEK) [Y/n/?] y
  MediaTek MT7601U (USB) support (MT7601U) [N/m/?] n
Ralink devices (WLAN_VENDOR_RALINK) [Y/n/?] y
  *
  * Ralink driver support
  *
  Ralink driver support (RT2X00) [N/m/?] n
Realtek devices (WLAN_VENDOR_REALTEK) [Y/n/?] y
  Realtek 8187 and 8187B USB support (RTL8187) [N/m/?] n
  *
  * Realtek rtlwifi family of devices
  *
  Realtek rtlwifi family of devices (RTL_CARDS) [N/m/?] n
  RTL8723AU/RTL8188[CR]U/RTL819[12]CU (mac80211) support (RTL8XXXU) [N/m/?] n
Redpine Signals Inc devices (WLAN_VENDOR_RSI) [Y/n/?] y
  Redpine Signals Inc 91x WLAN driver support (RSI_91X) [N/m/?] n
STMicroelectronics devices (WLAN_VENDOR_ST) [Y/n/?] y
  CW1200 WLAN support (CW1200) [N/m/?] n
Texas Instrument devices (WLAN_VENDOR_TI) [Y/n/?] y
  TI wl1251 driver support (WL1251) [N/m/?] n
  TI wl12xx support (WL12XX) [N/m/?] n
  TI wl18xx support (WL18XX) [N/m/?] n
  TI wlcore support (WLCORE) [N/m/?] n
ZyDAS devices (WLAN_VENDOR_ZYDAS) [Y/n/?] y
  USB ZD1201 based Wireless device support (USB_ZD1201) [N/m/?] n
```

```
   ZyDAS ZD1211/ZD1211B USB-wireless support (ZD1211RW) [N/m/?] n
  Simulated radio testing tool for mac80211 (MAC80211_HWSIM) [N/m/?] n
  Wireless RNDIS USB support (USB_NET_RNDIS_WLAN) [N/m/?] n
#
# configuration written to .config
#
```

5) Build the Linux kernel image

```
$ make zImage -j 8
```

----sample output---

```
…
  LDS     arch/arm/boot/compressed/vmlinux.lds
  AS      arch/arm/boot/compressed/head.o
  LZO     arch/arm/boot/compressed/piggy_data
  CC      arch/arm/boot/compressed/misc.o
  CC      arch/arm/boot/compressed/decompress.o
  CC      arch/arm/boot/compressed/string.o
  SHIPPED arch/arm/boot/compressed/hyp-stub.S
  SHIPPED arch/arm/boot/compressed/lib1funcs.S
  SHIPPED arch/arm/boot/compressed/bswapsdi2.S
  SHIPPED arch/arm/boot/compressed/ashldi3.S
  AS      arch/arm/boot/compressed/hyp-stub.o
  AS      arch/arm/boot/compressed/lib1funcs.o
  AS      arch/arm/boot/compressed/ashldi3.o
  AS      arch/arm/boot/compressed/bswapsdi2.o
  AS      arch/arm/boot/compressed/piggy.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
```

12.3 Build the cypress driver/ backports modules

1) Download Cypress' Varan release here
2) Unzip cypress-fmac-v4.14.77-2019_1031.zip
3) Untar the Cypress backports package

```
$ tar zxvf cypress-backports-v4.14.77-2019_1031-module-src.tar.gz
```

A folder named v4.14.77-backports will be created.

```
$ cd v4.14.77-backports
```

4) (Native) compile local tools and generate .config (in a new terminal without sourcing Yocto toolchain settings)

```
$ MY_KERNEL=~/BackportDemo/linux-imx/
$ make KLIB=$MY_KERNEL KLIB_BUILD=$MY_KERNEL defconfig-brcmfmac
```

----sample output---

```
make[2]: 'conf' is up to date.
#
# configuration written to .config
#
```

5) (Cross) compile kernel modules

```
$ source /opt/fsl-imx-x11/4.9.88-2.0.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
$ make KLIB=$MY_KERNEL KLIB_BUILD=$MY_KERNEL modules
```

----sample output---

```
…
  LD [M]  /home/skerr/v4.14.77-backports /net/wireless/cfg80211.o
  Building modules, stage 2.
  MODPOST 4 modules
  CC      /home/skerr/v4.14.77-backports/compat/compat.mod.o
  LD [M]  /home/skerr/v4.14.77-backports/compat/compat.ko
  CC      /home/skerr/v4.14.77-
backports/drivers/net/wireless/broadcom/brcm80211/brcmfmac/brcmfmac.mod.o
  LD [M]  /home/skerr/v4.14.77-
backports/drivers/net/wireless/broadcom/brcm80211/brcmfmac/brcmfmac.ko
  CC      /home/skerr/v4.14.77-
backports/drivers/net/wireless/broadcom/brcm80211/brcmutil/brcmutil.mod.o
  LD [M]  /home/skerr/v4.14.77-
backports/drivers/net/wireless/broadcom/brcm80211/brcmutil/brcmutil.ko
  CC      /home/skerr/v4.14.77-backports/net/wireless/cfg80211.mod.o
  LD [M]  /home/skerr/v4.14.77-backports/net/wireless/cfg80211.ko
```

6) The built kernel modules are available here
   - compat/compat.ko
   - net/wireless/cfg80211.ko
   - drivers/net/wireless/broadcom/brcm80211/brcmutil/brcmutil.ko
   - drivers/net/wireless/broadcom/brcm80211/brcmfmac/brcmfmac.ko

12.4 Device tree

In this example, we use uSD-M.2 Adapter with 1MW EVB. To make it work on NXP's i.MX6ULL, we need dtb files for it. Murata has a patch file on GitHub for Linux Kernel 4.9.88. It will generate dts file for uSD-M.2 Adapter. The following steps show how to download it, apply it, and then build dtb file for uSD-M.2 Adapter.

1) Get the patch file for dtb files from Murata's GitHub and place it in the linux-imx folder.

```
$ wget https://raw.githubusercontent.com/murata-wireless/meta-murata-wireless/imx-rocko-
manda/recipes-kernel/linux/linux-imx-4.9.88/0002-murata-dts-3.3v.patch
$ cp 0002-murata-dts-3.3v.patch $MY_KERNEL
```

2) Apply the patch file

```
$ cd $MY_KERNEL
$ patch -p1 < 0002-murata-dts-3.3v.patch
```

----sample output---

```
patching file arch/arm/boot/dts/Makefile
patching file arch/arm/boot/dts/imx6dl-sabresd-btwifi-m2-oob.dts
patching file arch/arm/boot/dts/imx6dl-sabresd-btwifi-m2.dts
patching file arch/arm/boot/dts/imx6dqscm-qwks-rev3-btwifi.dtsi
patching file arch/arm/boot/dts/imx6q-sabresd-btwifi-m2-oob.dts
patching file arch/arm/boot/dts/imx6q-sabresd-btwifi-m2-oob.dtsi
patching file arch/arm/boot/dts/imx6q-sabresd-btwifi-m2.dts
patching file arch/arm/boot/dts/imx6qdl-sabresd-btwifi-m2.dtsi
patching file arch/arm/boot/dts/imx6qdl-sabresd-btwifi.dtsi
patching file arch/arm/boot/dts/imx6qp-sabresd-btwifi-m2-oob.dts
patching file arch/arm/boot/dts/imx6qp-sabresd-btwifi-m2.dts
patching file arch/arm/boot/dts/imx6sx-sdb-btwifi-m2-oob.dts
patching file arch/arm/boot/dts/imx6sx-sdb-btwifi-m2-oob.dtsi
patching file arch/arm/boot/dts/imx6sx-sdb-btwifi-m2.dts
patching file arch/arm/boot/dts/imx6sx-sdb-btwifi-m2.dtsi
patching file arch/arm/boot/dts/imx6sx-sdb-btwifi.dts
patching file arch/arm/boot/dts/imx6sxscm-evb-btwifi.dtsi
patching file arch/arm/boot/dts/imx6ul-14x14-evk-btwifi-m2-oob.dts
patching file arch/arm/boot/dts/imx6ul-14x14-evk-btwifi-m2.dts
patching file arch/arm/boot/dts/imx6ul-14x14-evk.dts
patching file arch/arm/boot/dts/imx6ul-evk-btwifi-m2-oob.dtsi
patching file arch/arm/boot/dts/imx6ul-evk-btwifi-oob.dtsi
patching file arch/arm/boot/dts/imx6ul-evk-btwifi.dtsi
patching file arch/arm/boot/dts/imx6ull-14x14-evk-btwifi-m2-oob.dts
patching file arch/arm/boot/dts/imx6ull-14x14-evk-btwifi-m2.dts
patching file arch/arm/boot/dts/imx6ull-14x14-evk.dts
patching file arch/arm/boot/dts/imx7d-sdb.dts
```

3) Build dtb files. Dtb files that end with m2 or m2-oob are developed for uSD-M.2 Adapter. We will use imx6ull-14x14-evk-btwifi-m2.dtb in this example.

```
$ make dtbs
```

----sample output---

```
…
 DTC    arch/arm/boot/dts/imx6ull-14x14-ddr3-arm2-wm8958.dtb
 DTC    arch/arm/boot/dts/imx6ull-14x14-evk.dtb
 DTC    arch/arm/boot/dts/imx6ull-14x14-evk-btwifi.dtb
 DTC    arch/arm/boot/dts/imx6ull-14x14-evk-btwifi-oob.dtb
 DTC    arch/arm/boot/dts/imx6ull-14x14-evk-btwifi-m2.dtb      🡐 example DTB
 DTC    arch/arm/boot/dts/imx6ull-14x14-evk-btwifi-m2-oob.dtb
 DTC    arch/arm/boot/dts/imx6ull-14x14-evk-emmc.dtb
 DTC    arch/arm/boot/dts/imx6ull-14x14-evk-gpmi-weim.dtb
 DTC    arch/arm/boot/dts/imx6ull-14x14-evk-usb-certi.dtb
…
```

12.5 Driver and related files

Firmware, NVRAM and clm_blob files are needed to make the Murata module work. This example uses 1MW EVB whose chip id is 43455. The following three files are needed.

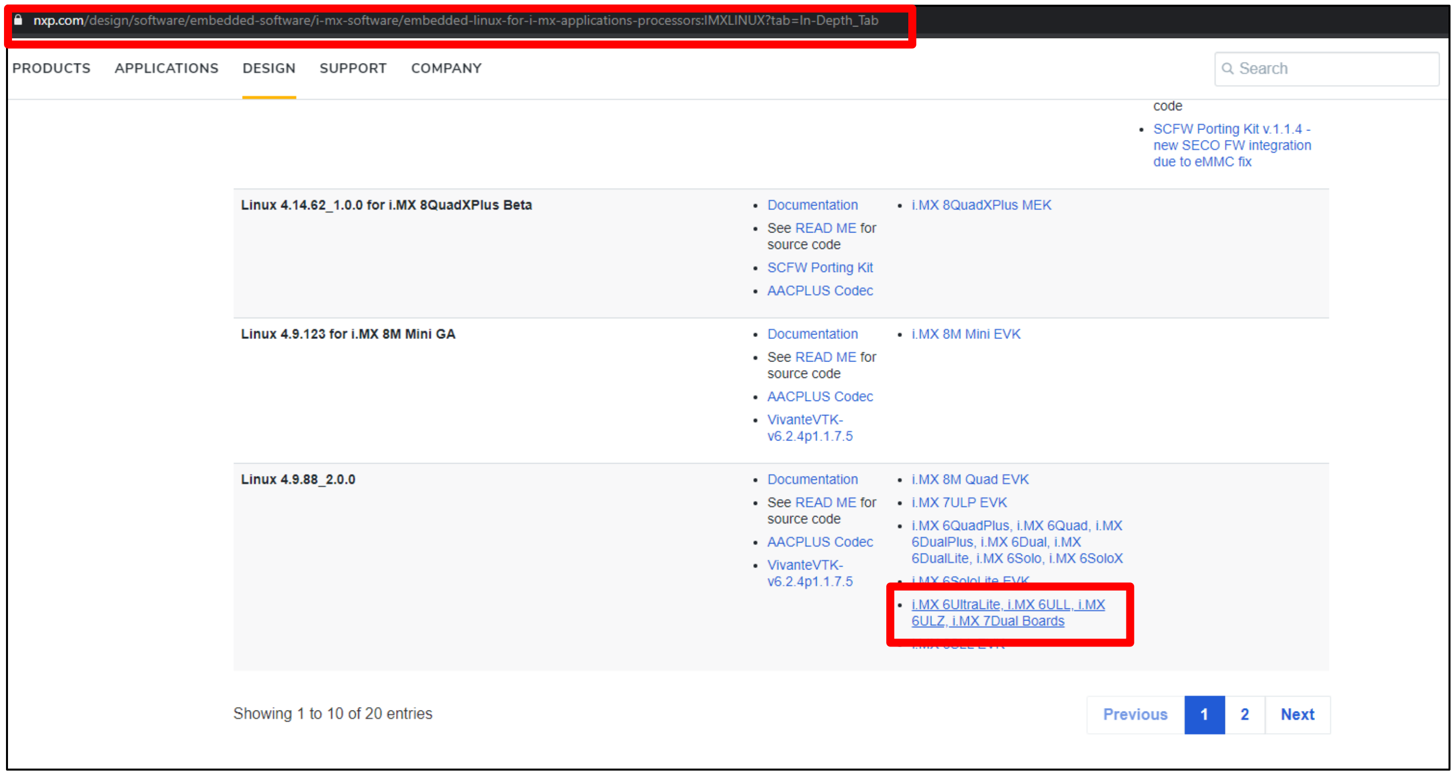
- brcmfmac43455-sdio.bin: Firmware file for Murata Type 1MW EVB. This file can be found in Cypress' tarball release, cypress-fmac-v4.14.77-2019_1031.zip.
- brcmfmac43455-sdio.clm_blob: clm_blob for Murata Type 1MW EVB. This file can be found in Cypress' tarball release, cypress-fmac-v4.14.77-2019_1031.zip.
- brcmfmac43455-sdio.txt: NVRAM file for Murata Type 1MW EVB. This file can be found on Murata's GitHub: https://github.com/murata-wireless/cyw-fmac-nvram/tree/manda. Make sure to rename it correctly to take effect. The naming convention is: brcmfmac<chip id>-sdio.txt. and the renamed file should be kept with the firmware file and clm_blob for later usage.

12.6 Flash a demo image

At this point, we have all the files needed for backporting ready which are kernel image, dtb file for uSD-M.2 Adapter, driver, NVRAM, and clm_blob file for 1MW EVB, and the four .ko files. Now we need a uSD card to flash NXP's demo image there and then drop in the files we have prepared. For convenience, lets get the demo image for imx6ullevk from NXP. Please visit the following webpage for the demo image.

https://www.nxp.com/design/software/embedded-software/i-mx-software/embedded-linux-for-i-mx-applications-processors:IMXLINUX?tab=In-Depth_Tab

(Scroll down / go to next page if the image cannot be seen)


nxp.com/design/software/embedded-software/i-mx-software/embedded-linux-for-i-mx-applications-processors:IMXLINUX?tab=In-Depth_Tab

PRODUCTS APPLICATIONS DESIGN SUPPORT COMPANY

Search

			code • SCFW Porting Kit v.1.1.4 - new SECO FW integration due to eMMC fix
Linux 4.14.62_1.0.0 for i.MX 8QuadXPlus Beta	• Documentation • See READ ME for source code • SCFW Porting Kit • AACPLUS Codec	• i.MX 8QuadXPlus MEK	
Linux 4.9.123 for i.MX 8M Mini GA	• Documentation • See READ ME for source code • AACPLUS Codec • VivanteVTK-v6.2.4p1.1.7.5	• i.MX 8M Mini EVK	
Linux 4.9.88_2.0.0	• Documentation • See READ ME for source code • AACPLUS Codec • VivanteVTK-v6.2.4p1.1.7.5	• i.MX 8M Quad EVK • i.MX 7ULP EVK • i.MX 6QuadPlus, i.MX 6Quad, i.MX 6DualPlus, i.MX 6Dual, i.MX 6DualLite, i.MX 6Solo, i.MX 6SoloX • i.MX 6SoloLite EVK • i.MX 6UltraLite, i.MX 6ULL, i.MX 6ULZ, i.MX 7Dual Boards • i.MX 6ULL EVK	

Showing 1 to 10 of 20 entries

Previous 1 2 Next

Note that the demo image on NXP's website is a hybrid one. It can be used for i.MX6UltraLite, i.MX6ULL, and i.MX 7Dual. So, we need to flash the right u-boot image for i.MX6ULL.

1) Unzip the image, flash it to SD card.

```
$ bunzip2 -dk -f fsl-image-validation-imx-xwayland-imx6ul7d.sdcard.bz2
```

2) Flash the image. Before doing that, please use "dmesg" command to make sure you point to the right name. In this case, the uSD card is enumerated as sdb, but it might be different on your machine. So, modify the command accordingly.

```
$ sudo dd if=fsl-image-validation-imx-xwayland-imx6ul7d.sdcard of=/dev/sdb bs=1M conv=fsync
```

3) Flash the uboot image.

```
$ sudo dd if=u-boot-imx6ull14x14evk_sd.imx of=/dev/sdb bs=1k seek=1 conv=fsync
```

12.7 Modify Bootloader

1) Copy kernel image and the dtb file to boot loader.

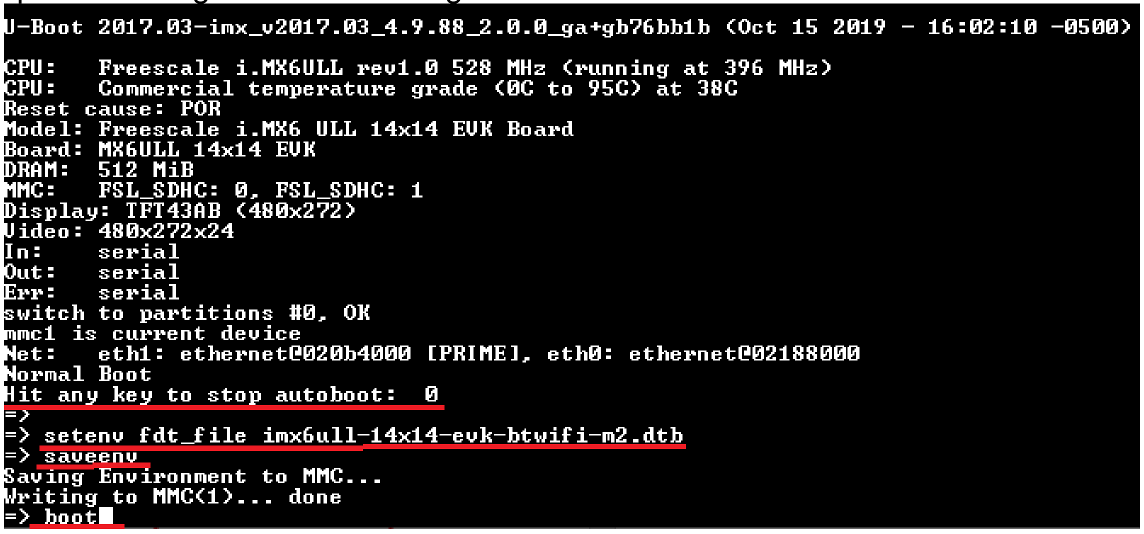
```
$ cd $MY_KERNEL
$ sudo cp arch/arm/boot/zImage /media/skerr/Boot\ imx6ul/
$ sudo cp arch/arm/boot/dts/imx6ull-14x14-evk-btwifi-m2.dtb /media/skerr/Boot\ imx6ul/
```

12.8 Modify filesystem

1) In the SD card, create a folder 4.9.88 under /lib/modules/, then drop in the four .ko files built in **Section 12.3** here.
2) In the SD card, create a brcm folder under /lib/firmware. Then drop in the firmware, nvram, and clm_blob files obtained in **Section 12.5** here.
3) The SD card is now ready to be used to boot the platform.

12.9 Load driver

1) Power on the platform.
2) Interrupt the booting and select the right dtb file. Check the screenshot below for an example.



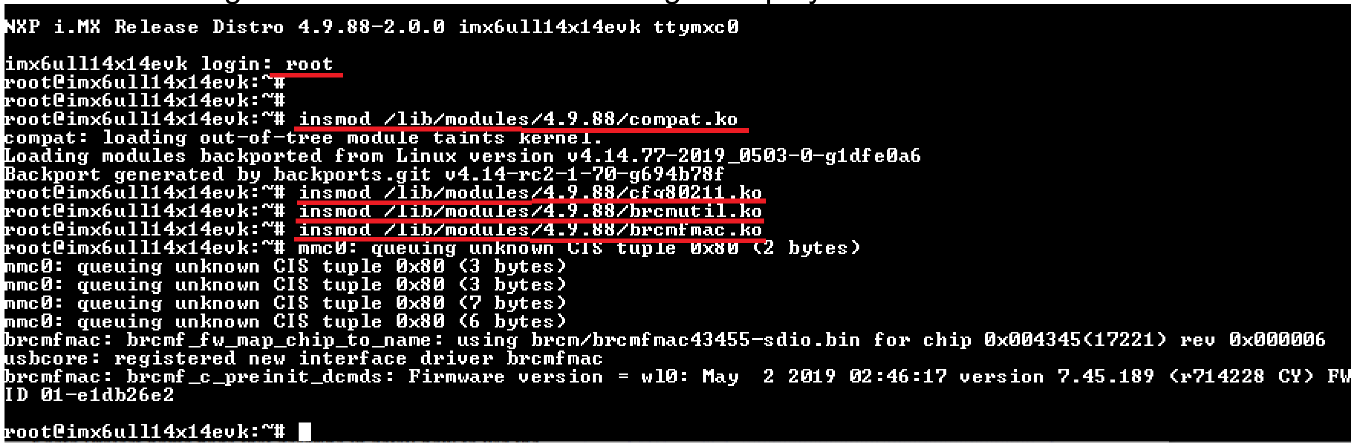
```
U-Boot 2017.03-imx_v2017.03_4.9.88_2.0.0_ga+gb76bb1b (Oct 15 2019 - 16:02:10 -0500)

CPU:   Freescale i.MX6ULL rev1.0 528 MHz (running at 396 MHz)
CPU:   Commercial temperature grade (0C to 95C) at 38C
Reset cause: POR
Model: Freescale i.MX6 ULL 14x14 EVK Board
Board: MX6ULL 14x14 EVK
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Display: TFT43AB (480x272)
Video: 480x272x24
In:    serial
Out:   serial
Err:   serial
switch to partitions #0, OK
mmc1 is current device
Net:   eth1: ethernet@020b4000 [PRIME], eth0: ethernet@02188000
Normal Boot
Hit any key to stop autoboot:  0
=>
=> setenv fdt_file imx6ull-14x14-evk-btwifi-m2.dtb
=> saveenv
Saving Environment to MMC...
Writing to MMC(1)... done
=> boot
```

3) Insmod the four ".ko" files.

```
$ insmod /lib/modules/4.9.88/compat.ko
$ insmod /lib/modules/4.9.88/cfg80211.ko
$ insmod /lib/modules/4.9.88/brcmutil.ko
$ insmod /lib/modules/4.9.88/brcmfmac.ko
```

The following screenshot shows the messages displayed on success.

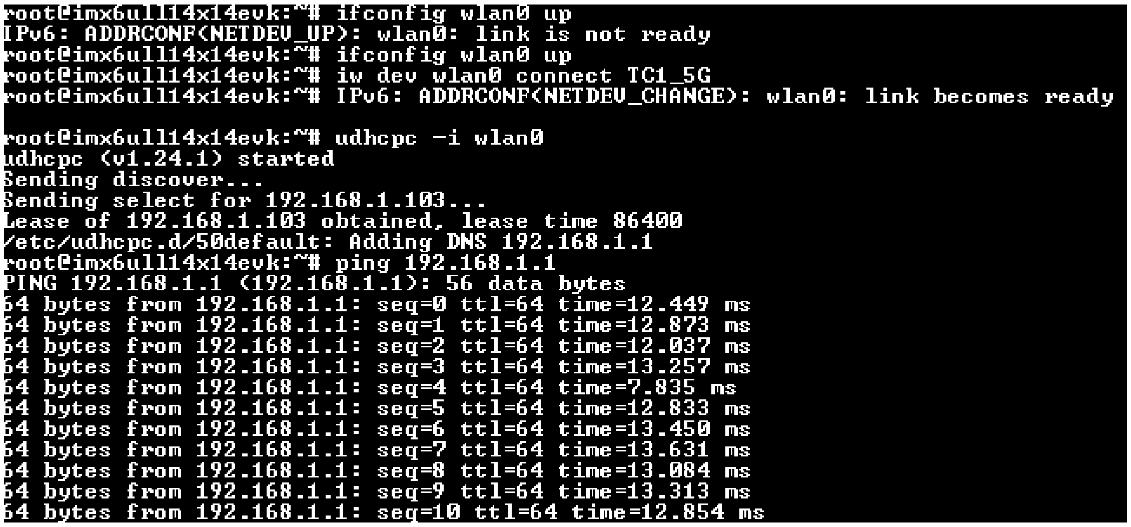


```
NXP i.MX Release Distro 4.9.88-2.0.0 imx6ull14x14evk ttymxc0

imx6ull14x14evk login: root
root@imx6ull14x14evk:~#
root@imx6ull14x14evk:~#
root@imx6ull14x14evk:~# insmod /lib/modules/4.9.88/compat.ko
compat: loading out-of-tree module taints kernel.
Loading modules backported from Linux version v4.14.77-2019_0503-0-g1dfe0a6
Backport generated by backports.git v4.14-rc2-1-70-g694b78f
root@imx6ull14x14evk:~# insmod /lib/modules/4.9.88/cfg80211.ko
root@imx6ull14x14evk:~# insmod /lib/modules/4.9.88/brcmutil.ko
root@imx6ull14x14evk:~# insmod /lib/modules/4.9.88/brcmfmac.ko
root@imx6ull14x14evk:~# mmc0: queuing unknown CIS tuple 0x80 (2 bytes)
mmc0: queuing unknown CIS tuple 0x80 (3 bytes)
mmc0: queuing unknown CIS tuple 0x80 (3 bytes)
mmc0: queuing unknown CIS tuple 0x80 (7 bytes)
mmc0: queuing unknown CIS tuple 0x80 (6 bytes)
brcmfmac: brcmf_fw_map_chip_to_name: using brcm/brcmfmac43455-sdio.bin for chip 0x004345(17221) rev 0x000006
usbcore: registered new interface driver brcmfmac
brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: May  2 2019 02:46:17 version 7.45.189 (r714228 CY) FW
ID 01-e1db26e2

root@imx6ull14x14evk:~#
```

4) Test the Wi-Fi module to make sure it is up and running.



```
root@imx6ull14x14evk:~# ifconfig wlan0 up
IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
root@imx6ull14x14evk:~# ifconfig wlan0 up
root@imx6ull14x14evk:~# iw dev wlan0 connect TC1_5G
root@imx6ull14x14evk:~# IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready

root@imx6ull14x14evk:~# udhcpc -i wlan0
udhcpc (v1.24.1) started
Sending discover...
Sending select for 192.168.1.103...
Lease of 192.168.1.103 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 192.168.1.1
root@imx6ull14x14evk:~# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=12.449 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=12.873 ms
64 bytes from 192.168.1.1: seq=2 ttl=64 time=12.037 ms
64 bytes from 192.168.1.1: seq=3 ttl=64 time=13.257 ms
64 bytes from 192.168.1.1: seq=4 ttl=64 time=7.835 ms
64 bytes from 192.168.1.1: seq=5 ttl=64 time=12.833 ms
64 bytes from 192.168.1.1: seq=6 ttl=64 time=13.450 ms
64 bytes from 192.168.1.1: seq=7 ttl=64 time=13.631 ms
64 bytes from 192.168.1.1: seq=8 ttl=64 time=13.084 ms
64 bytes from 192.168.1.1: seq=9 ttl=64 time=13.313 ms
64 bytes from 192.168.1.1: seq=10 ttl=64 time=12.854 ms
```

13 Technical Support Resources

Table 14 lists all the support resources available for the Murata Wi-Fi/BT solution.

Table 14: List of Support Resources

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

14 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 "***fmac***" implementation relies heavily on GitHub (https://github.com/): 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: https://git-scm.com/. Commonly used "***git***" commands are shown in **Table 15**.

Table 15: 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 to: https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup.

- “***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***” and “***cyw-fmac***” repositories.

- “***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***”: **NOTE:** The double “**-**” character entry before “***hard***” switch. 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.

15 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[9], 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 16** provides a list of essential "***bitbake***" commands.

Table 16: 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 *image*.

NOTE: "***bitbake***" commands must be invoked from the Yocto build directory.

[9] OpenEmbedded and Yocto are similar embedded Linux distributions. "meta-murata-wireless" can be easily leveraged in OpenEmbedded as well.

16 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-priviledged 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).
libsdl1.2-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
skerr@murata.com
```

```
user.name : Scott Kerr
user.email: skerr@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=skerr@murata.com

Murata: ready to build "meta-murata-wireless" customized i.MX Linux image!
```

17 Appendix D: Example of running Murata Wireless Build

The following shows the output of building an image for NXP i.MX 6ULL EVK with Murata Type 1MW module, running 5.4.47 kernel release, using the Murata build script.

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

```
./Murata_Wireless_Yocto_Build.sh
```

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

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

3. Select Zeus Yocto release running Linux 5.4.47.

```
Select which entry? 0
Selected : 5.4.47
```

4. Select "CYW" for wireless solution (this configures image for 1MW).

```
Select which entry? 1
Selected : CYW
```

5. Select "Zigra" FMAC version.

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

6. Select i.MX 6ULL EVK as target platform.

```
Select your entry: 2
Selected target: imx6ull14x14evk
```

7. Proceed with the default distro and image selections.

```
Murata default DISTRO & Image pre-selected are:
DISTRO: fsl-imx-fb
Image:  core-image-base
```

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

8. Enter build-imx6ullevk as build directory name.

```
Enter build directory name: build-imx6ullevk
```

9. Verify selection and start the build.

```
i.MX Yocto Release                : 5.4.47_2.2.0 GA
Yocto branch                      : zeus
Wireless                          : CYW
fmac version                      : zigra
Target                            : imx6ull14x14evk
NXP i.MX EVK Part Number          : MCIMX6ULL-EVK
meta-murata-wireless Release Tag: imx-zeus-zigra_r1.0
DISTRO                            : fsl-imx-fb
Image                             : core-image-base
Build Directory                   : build-imx6ullevk

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

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

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

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

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

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

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

13. Once the build is complete, the image will be available in **~/linux-imx/build-imx6ullevk/tmp/deploy/images/imx6ullevk/** folder. Look for the file with “.wic.bz2” extension.