E2B-93-1091

Murata Wi-Fi/BT Solution for i.MX

Quick Start Guide (Linux)

Revision History

Revision	Date	Author	Change Description
1.0	Sept 1, 2015	S Kerr G Mohiuddin	Initial Release
1.1	Sept 6, 2015	S Kerr	Removed software compile/build dependency. User can bring up NXP platform by downloading necessary files before flashing bootable SD card. Refer to Linux User Guide on software build procedures.
2.0	Nov 7, 2015	S Kerr	Modified Murata Wi-Fi/BT EVK definition. This simplifies bring-up on NXP i.MX6 Platforms. Incorporated changes for i.MX6UL 3.14.38 GA BSP Release.
4.0	Feb 14, 2017	S Kerr	Renamed document to “Murata Wi-Fi/BT Solution for i.MX Quick Start Guide (Linux)”. 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 binary patch release which addresses errata/features on both releases. Provided support for new i.MX 7Dual SDB, i.MX 6ULL EVK, and Murata Type 1CK. Expanded WLAN test verification section.
5.0	Feb 26, 2018	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’s. Add support for new i.MX 8MQuad EVK.
5.1	June 19, 2019	S Kerr	Updated to support Linux 4.9.88 and “manda” version of “fmac” driver. Minor changes to support links.

This page intentionally left blank.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

1 Introduction

Murata has partnered with Cypress Semiconductor Corporation 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 document provides details on enabling Murata Wi-Fi/Bluetooth solutions on reference NXP i.MX platforms. The latest release supports NXP i.MX Linux 4.1.15, and 4.9.11 GA BSP's for i.MX6/7, Linux L4.9.51 for i.MX 8MQuad Beta and Linux 4.9.88 for i.MX6/7/8. The following steps are described:

- How to build SD card image (using Murata customized script file) which enables Cypress "***fmac***" WLAN driver (while disabling the legacy "bcmdhd" driver). This "***fmac***" driver and associated components (WPA supplicant, Hostapd, WLAN firmware and NVRAM files, Bluetooth patchfiles, etc.) is currently not enabled in the default Linux i.MX Yocto image.
- As part of the build process, the user may configured customized i.MX image to support 1.8V VIO signaling (necessary for UHS SDIO 3.0 operation) on the i.MX 6SoloX and 6UL(L) reference platforms (also required to interface to Type 1LV EVB).
- If using Type 1BW (CYW43340), reference the Linux User Manual to configure the "***fmac***" driver image to handle Out-Of-Band (OOB) edge-sensitive interrupts.
- Associated hardware modifications are described so the Murata Wi-Fi/BT EVK is fully functional on a NXP i.MX platform for desired VIO signaling and interrupt configuration.
- Connect Murata Wi-Fi/BT EVK to i.MX Reference platform and power up.
- Initialize/configure WLAN and Bluetooth interfaces.
- Exercise WLAN and Bluetooth functionality.

Note the current NXP i.MX Platforms supported include:

- i.MX 8MQuad Evaluation Kit
- i.MX 7Dual SABRE Development Board
- i.MX 6QuadPlus SABRE Development Board
- i.MX 6Quad/DualLite SABRE Development Board
- i.MX 6SoloX SABRE Development Board
- i.MX 6SoloLite Evaluation Kit
- i.MX 6UltraLite (or i.MX 6UL) Evaluation Kit
- i.MX 6ULL Evaluation Kit

A high-level connection Diagram for the Murata Interconnect kit (i.MX6 platforms) is provided in **Figure 1**. All the Murata Wi-Fi/BT modules enabled by this release are shown[1]. The Murata Wi-Fi/BT kit for i.MX6 enables this configuration by providing two custom-built Adapter boards. Refer to **Section 9** for more details on the V1/V2 i.MX InterConnect Adapters.
No V1/V2 adapter kit is required for the i.MX 7Dual SDB[2]. 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 very 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[3]. **Figure 2** shows a simplified block diagram for the i.MX 7Dual SDB. **NOTE:** only the i.MX6UL(L) and i.MX6SX platforms support 1.8V VIO signaling with specific modifications to the V1/V2 InterConnect adapter. 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 1: Murata i.MX6 Interconnect Kit Interfaces

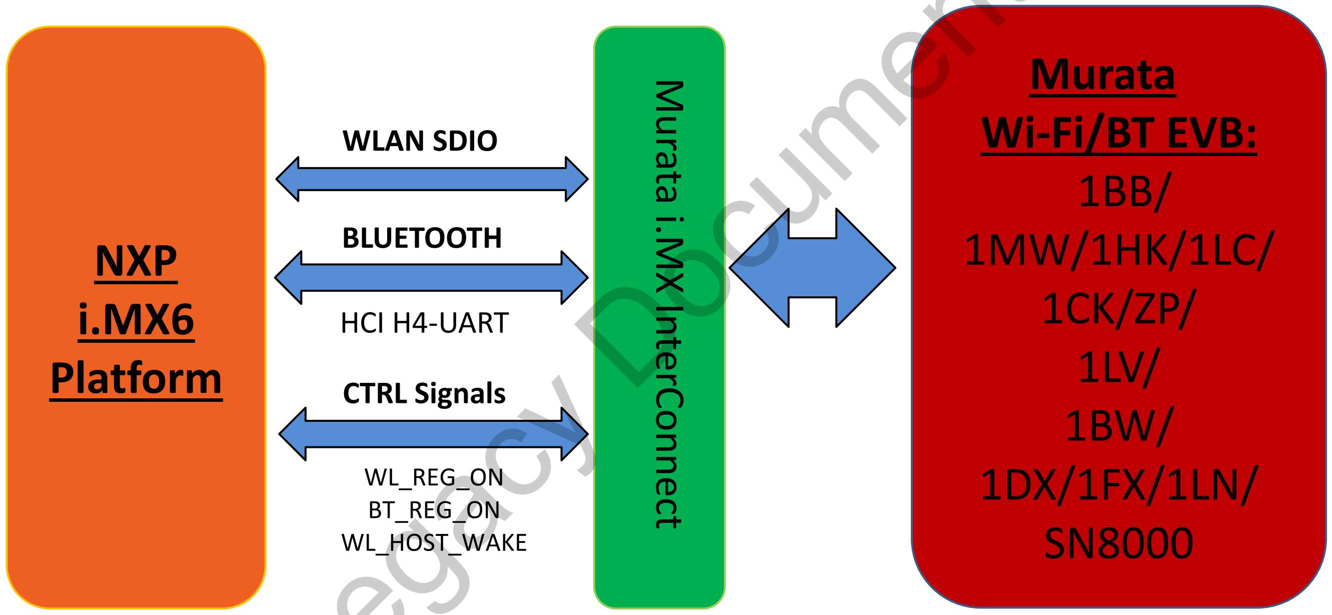


Figure 3 shows a simplified block diagram for the i.MX 8MQuad EVK Wi-Fi/BT interconnect. Currently both (CYW4356) modules supported are 2x2 802.11ac MIMO with PCIe interface. Some minor rework must be done to fully enable the Bluetooth PCM option on the i.MX8MQuad EVK. However, the basic WLAN PCIe and Bluetooth UART interfaces are supported with no hardware modifications. To obtain either Type 1CX or 1DK EVB with necessary M.2 adapter board, please contact Murata directly.

[1] Some of the modules shown are only available directly from Murata.
[2] Although an external module can be connected to the i.MX7D SDB, Murata does not recommend this given extensive rework required.
[3] 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

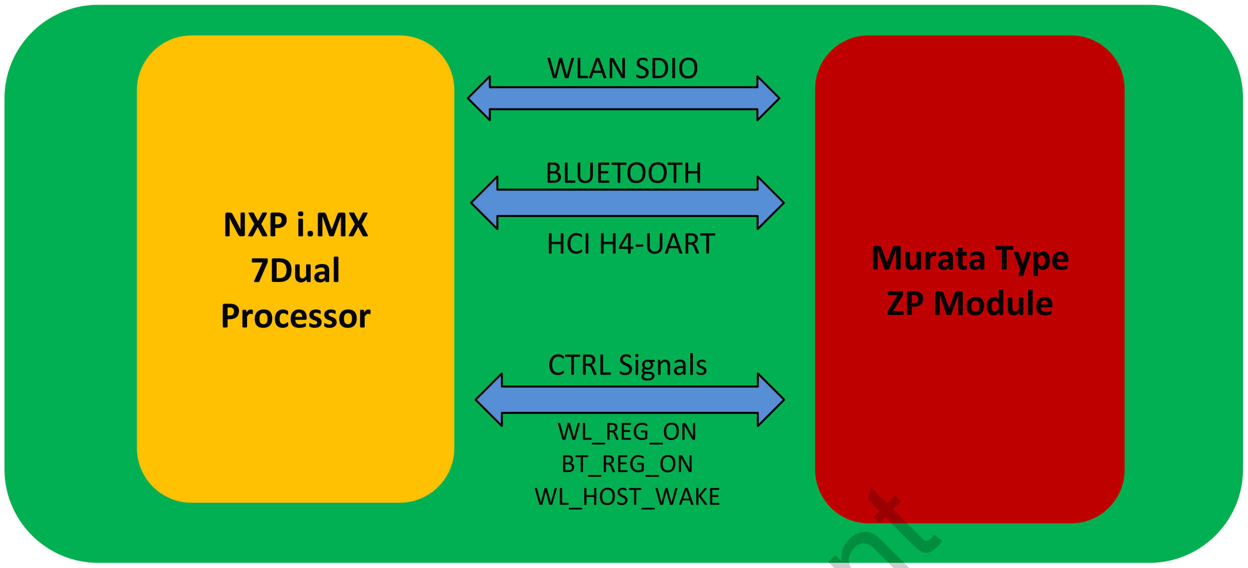
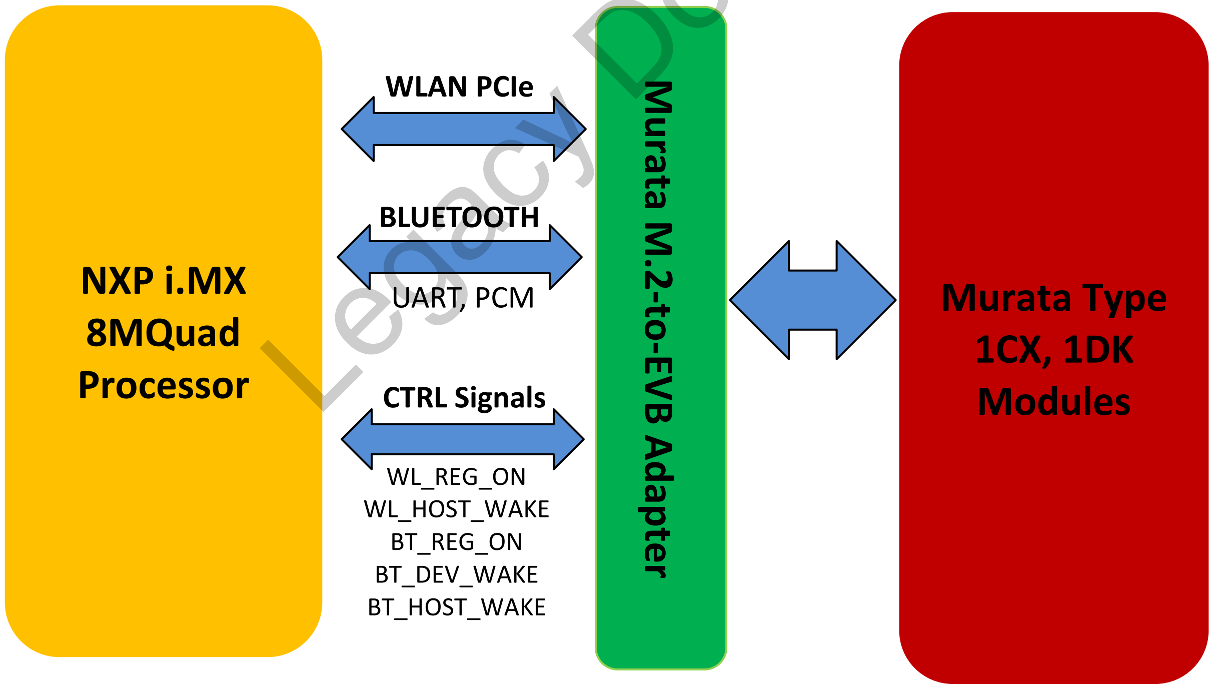


Figure 3: i.MX 8MQuad EVK Wi-Fi/BT Interconnect Block Diagram



1.1 Acronyms

Table 1: Acronyms used in Quick Start Guide

Acronym	Meaning
API	Application Programming Interface
DTB	Device Tree Blob: Kernel reads in at boot time for configuration.
EVB	Evaluation Board (Murata module on custom PCB)
EVK	Evaluation Kit (includes EVB + Adapter)
FW	Firmware
GPIO	General Purpose Input/Output
PC	Personal Computer
PCIe	PCI Express
SW	Software
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

1.2 References

1.2.1 Murata Linux User Manual

Murata Wi-Fi/BT Solution for i.MX Linux User Manual 5.4, "Murata Wi-Fi & BT Solution for i.MX Linux User Manual 5.4.pdf".

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

This manual is a key document given the hard requirement for users to build the i.MX image and flash a (micro) SD card prior to booting the i.MX platform.

The Murata Linux User Manual is available here: https://wireless.murata.com/imx.

1.2.2 Murata Hardware User Manual

Murata Wi-Fi/BT Solution for i.MX Hardware User Manual 1.0, "Murata Wi-Fi & BT Solution for i.MX Hardware User Manual 1.0.pdf".

This manual provides HW specifics for the various i.MX platforms supported:

- Murata V1/V2 i.MX InterConnect Adapters.
- Signals required to interface Murata Wi-Fi/BT module solution.
- i.MX6UL(L), and i.MX6SX 1.8V VIO customizations.
- i.MX7D SDB Wi-Fi/BT interconnect to onboard Type ZP module.

The Murata Hardware User Manual is available here: https://wireless.murata.com/imx.

1.2.3 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 a 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 2 provides the following information on all releases supported:

- document filename
- document number
- document release revision
- date of release
- hyperlink to NXP document download

Table 2: NXP Reference Documentation Listing

Document Filename / Number	4.9.88_2.0.0	4.9.51 MQ Beta	4.9.11_1.0.0	4.1.15_2.0.0
i.MX_Yocto_Project_User's_Guide.pdf / IMXLXYOCTOUG	Rev. L4.9.88_2.0.0-ga, 05/2018	Rev. L4.9.51_imx8mq-beta; 12/2017	Rev. L4.9.11_1.0.0-ga+mx8-alpha; 09/2017	Rev. L4.1.15_2.0.0-ga; 102016
i.MX_Linux_User's_Guide.pdf / IMXLUG				
i.MX_Linux_Reference_Manual.pdf / IMXLXRM				
i.MX_Linux_Release_Notes.pdf / IMXLXRN				

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

Table 3 provides an i.MX Reference versus Murata module matrix. An additional column is included to provide a quick Cypress chipset lookup for the different Murata module names. For a given i.MX platform and Murata module, you can quickly look up the compatibility. Providing more details on the terminology used in the table:

- "**NC**" means not compatible. This is due to one or both of the following reasons:
  - VIO incompatible: module requires VIO voltage level that the i.MX HW cannot provide.
  - Bus (SDIO, PCIe, UART) interconnect not available.

- "**DRW**" means difficult rework required. This configuration is not recommended and not supported by Murata.

- "**Y**" or **GREEN** text means that the configuration **is** supported. The entries with green text provide the following information:
  - VIO supported with corresponding interrupt configuration. There are two VIO's (1.8V and 3.3V). When configuring some of the i.MX platforms (i.MX 6UL(L) and 6SoloX) for 1.8V VIO operation, there is a limitation on control signal voltage levels. More specifically the WL_REG_ON, BT_REG_ON, WL_HOST_WAKE pins are configured for 3.3V at i.MX processor end (even though rest of platform is running @1.8V). To ensure reliability the interrupt configuration is changed to SDIO in-band to avoid possible unreliability issues (i.e. WL_HOST_WAKE driven at 1.8V VIO with i.MX pin (GPIO) configured for 3.3V VIO).
  - To further explain the VIO/interrupt configuration. "Out" denotes out-of-band signaling: this is when WL_HOST_WAKE line is used and the i.MX requires a dedicated GPIO to monitor the interrupt. "In" denotes SDIO-in-band signaling. The i.MX SDIO host controller monitors for interrupt status on the SDIO_DATA_1 line.
  - Maximum SDIO clock frequency. Some modules only support SDIO 2.0, thereby limiting SDIO clock frequency to 50 MHz. When the i.MX platform only supports 3.3V VIO over SDIO bus, then that also limits SDIO bus frequency to 50 MHz. For i.MX/module configurations that both support SDIO 3.0 (UHS mode), then the SDIO clock can go up to 200 MHz (i.MX 6SoloX and 6UL). **NOTE:** The i.MX 6ULL platform has a maximum SDIO clock of 132 MHz.

- The i.MX 8MQuad (with Murata Type 1CX) supports WLAN PCIe interface with BT UART and PCM. A special Murata M.2 interconnect board is required for this combination. PCM is only enabled by minor rework on the i.MX 8MQuad platform.
  NOTE: for more specifics on HW interconnect, please reference the Hardware User Manual.

Table 4 provides more details on each Murata Wi-Fi/BT solution:

- WLAN and Bluetooth support capability.
- Actual module dimensions.
- Picture of Murata Wi-Fi/BT EVB. Note that picture 1CK is actual module (it uses an interposer board to hook up to V1/V2 InterConnect Adapter).

Table 3: i.MX EVK / Murata Module Matrix

Murata Module	CYW Chipset	6UL(L)	6SL	6SX	6Q(P)/DL	7D	8MQ
1FX	43364	1.8V=In 3.3V=Out 50 MHz	3.3V=Out 50 MHz	1.8V=In 3.3V=Out 50 MHz	3.3V=Out 50 MHz	DRW	NC
1DX / 1LN	4343W	1.8V=In 3.3V=Out 50 MHz	3.3V=Out 50 MHz	1.8V=In 3.3V=Out 50 MHz	3.3V=Out 50 MHz	DRW	NC
1BW	43340	1.8V=In 3.3V=Out 50 MHz	3.3V=Out 50 MHz	1.8V=In 3.3V=Out 50 MHz	3.3V=Out 50 MHz	DRW	NC
ZP / 1CK	4339	1.8V=In 3.3V=Out 200 MHz	3.3V=Out 50 MHz	1.8V=In 3.3V=Out 200 MHz	3.3V=Out 50 MHz	ZP onboard	NC
1MW	43455	1.8V=In 3.3V=Out 200 MHz	3.3V=Out 50 MHz	1.8V=In 3.3V=Out 200 MHz	3.3V=Out 50 MHz	DRW	NC
1LV	43012	1.8V=In 50 MHz	NC	1.8V=In 50 MHz	NC	DRW	NC
1BB	4354	1.8V=In 3.3V=Out 200 MHz	3.3V=Out 50 MHz	1.8V=In 3.3V=Out 200 MHz	3.3V=Out 50 MHz	DRW	NC
1CX / 1DK	4356	NC	NC	DRW	DRW	DRW	Y

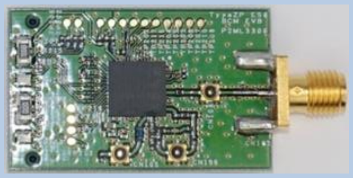
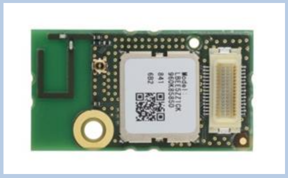
Y = Supported   NC = Not Compatible   DRW = Difficult Rework Req'd (Not Supported)

Additional information is provided for WLAN-SDIO configurations.

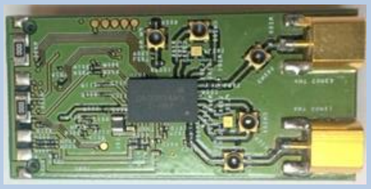
The VIO and Interrupt Configuration is listed as: <VIO>=<In | Out>
VIO is 1.8V or 3.3V with corresponding supported interrupt configuration.
"In" is SDIO-in-band interrupts; "Out" is Out-of-Band IRQ (Host GPIO Req'd)

MAX SDIO Clock frequency is listed:
For 3.3V VIO; MAX SDIO clock frequency is 50 MHz.
MAX frequency for dual 1.8/3.3V support only applies to 1.8V VIO (i.e. UHS).

Table 4: Murata Wi-Fi/BT EVK's Supported

Murata Module	CYW Chipset	Wi-Fi/BT Support	Dimensions	Part Number Or Availability	EVB Picture[4]
1FX	**43364**	802.11 b/g/n (Wi-Fi Only)	6.95 x 5.15mm H: 1.1mm	LBWA1KL1FX-TEMP-DS-SD	
1DX	**4343W**	802.11 b/g/n BT/BLE	6.95 x 5.15mm H: 1.1mm	LBEE5KL1DX-TEMP-DS-SD	
1BW	**43340**	802.11 a/b/g/n BT/BLE	8.0 x 7.5mm H: 1.1mm	LBEH5DU1BW-TEMP-DS-SD	
ZP	**4339**	802.11a/b/g/n/ac(1x1) BT/BLE	7.8 x 7.4mm H: 1.0mm	LBEH5HMZPC-TEMP-DS-SD	
1CK	**4339**	802.11a/b/g/n/ac(1x1) BT/BLE	33.0 x 18.0mm H: 7.55mm	LBEE5ZZ1CK-TEMP-DS-SD	
1MW	**43455**	802.11a/b/g/n/ac(1x1) BT/BLE	7.9 x 7.3mm H: 1.1mm	Q2/'18	
1LV	**43012**	802.11a/b/g/n BT/BLE (AC friendly)	10.0 x 7.2mm H: 1.1mm	Q3/'18	

[4] Picture of 1CK is actual module (includes PCB with integrated antenna and connector).

Murata Module	CYW Chipset	Wi-Fi/BT Support	Dimensions	Part Number or Availability	EVB Picture
1BB	4354	802.11a/b/g /n/ac(2x2) BT/BLE	9.2 x 5.7mm H: 1.2mm	Contact Murata	
1CX / 1DK	4356	802.11a/b/g /n/ac(2x2) BT/BLE	11.5 x 8.8mm H: 1.05mm	Contact Murata	

For the i.MX6 platforms, the Murata Wi-Fi/BT EVK contents are listed in **Table 5**. By contrast the i.MX 7Dual SDB has the Murata Type ZP module soldered down -- also documented in this quick start guide. One notable NXP i.MX platform not documented in this Quick Start Guide is the WaRP7 which is a "community" supported platform – more information here. The WaRP7 has the i.MX 7Solo processor mated with a Murata Type 1DX module. **Table 5** shows the contents for the Murata Wi-Fi/BT EVK. The specific kit contents differ based on module. Some important notes:

- Type 1CK includes interposer board that connects Type 1CK module (finished product) to the V1/V2 Adapter. Type 1CK kit does not come with external antenna option: **chip antenna already on module**. The 1CK module has a built-in test connector for conducted testing[5].
- All kits include V1/V2 adapters necessary for connecting different i.MX6 platforms.
- Type ZP/1BW kits include dual-band antenna; Type 1DX/1FX kits include single-band.

Murata Wi-Fi/BT EVK's supported on NXP i.MX6 Platforms are listed in

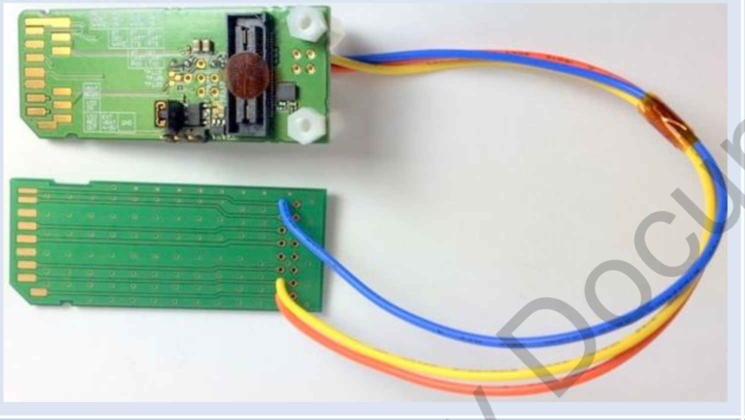
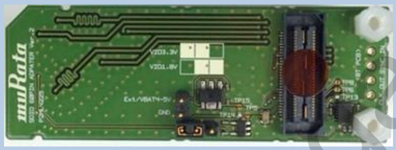
Table 4. Five (5) different modules are ***currently*** available. If you are having difficulty obtaining the desired Murata EVK, contact Murata for additional support. Alternatively click on "Order part number" hyperlinks listed in

Table 4 to bring up the Murata module webpage. Now click on "purchase" tab and scroll down to list currently available kits.

For Murata Interconnect kit connection diagram refer to **Figure 1: Murata i.MX6 Interconnect Kit Interfaces**. Murata Wi-Fi/BT kit for i.MX6 enables this configuration by providing two custom-built Adapter boards. Two adapter boards are provided in each kit allowing the user to bring up the Wi-Fi/Bluetooth interfaces in the easiest manner possible on any i.MX6 platform.

[5] Relevant Murata part numbers are: 1CK test connector is MM8030-2610; test cable for conducted testing is MXHQ87WJ3000.

Table 5: Murata Wi-Fi/BT EVK (for i.MX6) Kit Contents

Part Number	Picture of Contents	Description of Contents
1		Type 1CK/ZP/1BW/1DX/1FX Murata Wi-Fi/BT EVB. Type ZP is pictured. See **Table** 4 for specific part numbers.
		Type 1CK EVB includes an additional interposer board which connects to V1/V2 Adapter. Type 1CK module (LBEE5ZZ1CK) is FCC certified with built-in antenna and test connector.
2		**Murata i.MX InterConnect V1:** SD pins (DAT0..7) provide both Wi-Fi SDIO and Bluetooth UART connection. Wired SD Card Extender connects control signals: WL_REG_ON, BT_REG_ON, and WL_HOST_WAKE.
3		**Murata i.MX InterConnect V2:** SD pins provide Wi-Fi SDIO; ribbon cable connection provides Bluetooth UART and control signals: WL_REG_ON, BT_REG_ON, and WL_HOST_WAKE. Flexible 50 mm ribbon cable included.
4		Type ZP/1BW EVK's have 2.4/5.0 GHz Whip/Tilt SMA Antenna (for dual-band Wi-Fi) <u>**OR**</u> Type 1DX/1FX EVK's have 2.4 GHz Whip SMA Antenna (for single band Wi-Fi).

In addition to the i.MX6 InterConnect Kit (Murata Wi-Fi/BT EVK), there is also support provided on the i.MX 7Dual SDB. 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 default (3.3V VIO) InterConnect solution on some of the i.MX6 platforms (refer to **Table 3**), there is no inherent "legacy" restriction on the i.MX7 platform which can limit SDIO throughput. The Murata ZP module supports a very high throughput (SDIO 3.0 mode – UHS) over SDIO bus. Refer to the NXP i.MX7 schematics for specifics on the Wi-Fi/BT interconnect: download package here[6]. **Figure 2** shows a simplified block diagram for the i.MX 7Dual SDB.

Lastly, support is also provided for the i.MX 8MQuad EVK. However, this platform only supports PCIe WLAN interface. As such, the user must contact Murata directly to obtain the Type 1CX/1DK EVB and associated M.2 interposer board which are ***not*** available in the Distribution channel. **Figure 5** shows the i.MX 8MQuad interconnect block diagram.

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

3.1 "fmac" Solution Overview

The default NXP i.MX Linux 4.1.15, and 4.9.11 GA BSP's for i.MX6/7, Linux L4.9.51 for i.MX 8MQuad Beta and Linux 4.9.88 for i.MX6/7/8 integrate the legacy Cypress WLAN ***"bcmdhd"*** driver and have limited Bluetooth support[7]. The new implementation of Cypress' WLAN driver is referred to as ***"fmac"***. Please note that there is a distinct difference between 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 ***"fmac"*** release leverages the Linux Backports implementation to integrate the WLAN driver into the desired Linux kernel version.

Murata's customized Yocto layer ***"meta-murata-wireless***" seamlessly disables the existing ***"bcmdhd"*** 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.
- Provide ultra-high speed (UHS) SDIO operation for WLAN interface to i.MX6UL(L), i.MX6SX, and i.MX7ULP platforms with 1.8V VIO configuration.
- WLAN production firmware files. For manufacturing test firmware (necessary for RF/regulatory testing), please contact Murata directly.
- Murata NVRAM files for correctly configuring WLAN RF.
- Example Bluetooth patchfiles.
- WL tool binary necessary for interoperability and RF testing.
- Hostapd (Version 2.6) configuration with specific patch release.
- Hostap-conf enablement.
- Hostap-utils enablement.
- WPA-supplicant (Version 2.6) configuration with specific patch release.
- Wi-Fi Direct (P2P) enablement.

[6] For Wi-Fi/BT schematics on i.MX 7Dual SDB, refer to page 13 of sch-28590_i.mx7d_saber_rev_2.pdf document.
[7] Default BlueZ stack and Bluetooth driver is operational with Murata modules but NXP default image does not include all the necessary or correct Bluetooth patchfiles.

There are four versions of ***"fmac"*** currently supported: ***"v4.12 orga"***, ***"v4.14 battra", "v4.14 mothra"*** *and* ***"v4.14 manda"***. ***"orga"***, ***"battra", "mothra"*** *and* ***"manda"*** are the Cypress codenames denoting ***"fmac"*** release version. ***"v4.12"*** and ***"v4.14"*** are the latest kernel versions supported by either release (either ***"fmac"*** release can be backported to kernel version 3.0). To abbreviate references to specific versions of ***"fmac"***, Murata uses *just* the Cypress codename – i.e. ***"orga"*** / ***"battra" / "mothra" / "manda"***.

<u>**NOTE:**</u> It is strongly recommended to use the latest ***"fmac"*** release version: currently this is ***"manda"***.

3.2 Specific i.MX Target Support Details

Murata's customized Yocto layer supports the following NXP i.MX EVK's as outlined in **Table 6**. ***"MACHINE=target"*** is a direct reference to Yocto build[8]. 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 8MQuad) only certain kernel versions are supported.

Table 6: NXP i.MX EVK / Yocto (MACHINE) target / Kernel Version Matrix

i.MX EVK	MACHINE=target	Kernel Versions Supported	Notes
i.MX 8MQuad EVK	imx8mqevk	4.9.88, 4.9.51 Beta	EVB via M.2 interconnect.
i.MX 7ULP EVK	imx7ulpevk	4.9.88, 4.9.11	V1 Interconnect. Rework required.
i.MX 7Dual SDB	imx7dsabresd	4.9.88, 4.9.11, 4.1.15	Onboard ZP is supported.
i.MX 6QuadPlus SDB	imx6qpsabresd	4.9.88, 4.9.11, 4.1.15	V2 Interconnect.
i.MX 6Quad SDB	imx6qsabresd	4.9.88, 4.9.11, 4.1.15	V2 Interconnect.
i.MX 6DualLite SDB	imx6dlsabresd	4.9.88, 4.9.11, 4.1.15	V2 Interconnect.
i.MX 6SX SDB	imx6sxsabresd	4.9.88, 4.9.11, 4.1.15	V1 Interconnect.
i.MX 6SL EVK	imx6slevk	4.9.88, 4.9.11, 4.1.15	V1 Interconnect.
i.MX 6UL EVK (14x14, 9x9)	imx6ulevk	4.9.88, 4.9.11, 4.1.15	V2 Interconnect. Builds image for i.MX6ULL EVK as well.
i.MX 6ULL EVK (14x14)	imx6ull14x14evk	4.9.88, 4.9.11, 4.1.15	V2 Interconnect.
i.MX 6ULL EVK (9x9)	imx6ull9x9evk	4.9.88, 4.9.11, 4.1.15	V2 Interconnect.

To confirm support on desired i.MX Target (i.e. EVK) and i.MX Linux version number, refer to either **Table 7**, **Table 8**, or **Table 9**: i.MX reference interrupt configuration, DTB file, and Murata Adapter (V1/V2) are identified. <u>**Note:**</u> the first harmonized i.MX BSP release that supports all i.MX 8/7/6 architectures is Linux 4.9.88_2.0.0 GA.

Table 7: i.MX8 Targets supported on Linux 4.9.88_2.0.0 GA & 4.9.51 Beta Releases

Target (MACHINE)	NXP i.MX DTB File	Adapter
Imx8mqevk	fsl-imx8mq-evk-pcie1-m2.dtb	M.2

[8] Refer to Yocto Project User's Guide for your kernel version. **Table 2** provides documentation package download links.

Table 8: i.MX6/7 Targets supported on Linux 4.9.88_2.0.0 & 4.9.11_1.0.0 GA Releases

Target (MACHINE)	Interrupt Configuration	NXP i.MX DTB File	Adapter	1.8V VIO Option
imx7dsabresd	OOB IRQ	imx7d-sdb.dtb	Integrated	Yes (default)
imx6qpsabresd	OOB IRQ	imx6qp-sabresd-btwifi.dtb	V2	No
imx6qsabresd	OOB IRQ	imx6q-sabresd-btwifi.dtb	V2	No
imx6dlsabresd	OOB IRQ	imx6dl-sabresd-btwifi.dtb	V2	No
imx6sxsabresd	OOB for 3.3V; SDIO in-band for 1.8V	imx6sx-sdb-btwifi.dtb	V1	Yes
imx6slevk	OOB IRQ	imx6sl-evk-btwifi.dtb	V1	No
imx6ulevk	OOB IRQ	imx6ul-14x14-evk-btwifi-oob.dtb imx6ul-9x9-evk-btwifi-oob.dtb	V2	Yes
imx6ulevk	SDIO in-band	imx6ul-14x14-evk-btwifi.dtb imx6ul-9x9-evk-btwifi.dtb	V2	Yes
imx6ull14x14evk	OOB IRQ	imx6ull-14x14-evk-btwifi-oob.dtb	V2	Yes
imx6ull14x14evk	SDIO in-band	imx6ull-14x14-evk-btwifi.dtb	V2	Yes
imx6ull9x9evk	OOB IRQ	imx6ull-9x9-evk-btwifi-oob.dtb	V2	Yes
imx6ull9x9evk	SDIO in-band	imx6ull-9x9-evk-btwifi.dtb	V2	Yes
Imx7ulpevk	OOB IRQ	imx7ulp-evk.dtb	V1	Yes

Table 9: i.MX6/7 Targets supported on Linux 4.1.15_2.0.0 GA Release

Target (MACHINE)	Interrupt Configuration	NXP i.MX DTB File	Adapter	1.8V VIO Option
imx7dsabresd	OOB IRQ	imx7d-sdb.dtb	Integrated[9]	Yes (default)
imx6qpsabresd	OOB IRQ	imx6qp-sabresd-btwifi.dtb	V2	No
imx6qsabresd	OOB IRQ	imx6q-sabresd-btwifi.dtb	V2	No
imx6dlsabresd	OOB IRQ	imx6dl-sabresd-btwifi.dtb	V2	No
imx6sxsabresd	OOB for 3.3V; SDIO in-band for 1.8V	imx6sx-sdb-btwifi.dtb	V1	Yes
imx6slevk	OOB IRQ	imx6sl-evk-btwifi.dtb	V1	No
imx6ulevk	OOB IRQ	imx6ul-14x14-evk-btwifi-oob.dtb imx6ul-9x9-evk-btwifi-oob.dtb	V2	Yes
imx6ulevk	SDIO in-band	imx6ul-14x14-evk-btwifi.dtb imx6ul-9x9-evk-btwifi.dtb	V2	Yes
imx6ull14x14evk	OOB IRQ	imx6ull-14x14-evk-btwifi-oob.dtb	V2	Yes
imx6ull14x14evk	SDIO in-band	imx6ull-14x14-evk-btwifi.dtb	V2	Yes
imx6ull9x9evk	OOB IRQ	imx6ull-9x9-evk-btwifi-oob.dtb	V2	Yes
imx6ull9x9evk	SDIO in-band	imx6ull-9x9-evk-btwifi.dtb	V2	Yes

[9] i.MX 7Dual SDB has Murata Module ZP soldered down onto platform.

3.3 Murata “fmac” Customized i.MX Yocto Image Build

The NXP i.MX image contains third party IP which is sub-licensed via a click-through EULA (when either downloading an i.MX validation/demo image or building the image from source). As such Murata cannot make this image available directly to customers. Previously, customers could download the “bcmdhd” enabled i.MX image directly from NXP’s website. However, given the transition to “***fmac***”, the user must now build the Yocto Linux image. As detailed by the Murata Linux User Manual, Murata employs a customized “meta-murata-wireless” layer to make this customized Yocto build as simple as possible. Nonetheless end users still must configure a Linux build environment and follow specific steps to arrive at the desired image for a given i.MX target and Murata WLAN/BT configuration.

Murata has greatly simplified the build requirement by providing scripts for Ubuntu host setup and customized Yocto build – scripts easily downloadable from Murata’s Github. Steps for downloading, configuring and invoking these scripts are detailed here.

3.3.1 Install Ubuntu

First step is to install Ubuntu 12.04, 14.04 or 16.04 (Murata’s build is verified on Ubuntu 16.04 64-bit install) on the host - native PC or virtual environment like VMware. Host PC typically used has Ubuntu 16.04/14.04/12.04 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 (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.

3.3.2 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 “clone or download” button at: https://github.com/murata-wireless/meta-murata-wireless.
- Now select “Download ZIP” option.
- Once the file is downloaded, extract it with “unzip” command or folder UI.
- Now go to the “meta-murata-wireless-master/cyw-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/cyw-script-utils/latest/README.txt

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

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

chmod a+x *.sh
```

3.3.3 Configure Ubuntu for i.MX Yocto Build

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

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

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

```
./Host_Setup_for_Yocto.sh
```

The script goes through the following stages:

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

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

3.3.4 Murata's i.MX Yocto Build Script

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

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

- Installed 64-bit version of Ubuntu 16.04 (preferred), 14.04, or 12.04.
- Ran Murata's host setup script in **Section 3.3.3** to add necessary packages for Yocto build and configure GIT.
- Created a i.MX BSP folder ***specific*** to the desired i.MX Yocto Release. The i.MX Yocto distribution **cannot** build different versions of Yocto (Linux kernel) in the same folder. Currently the following Yocto releases are supported:
  - 4.9.88_2.0.0 GA
  - 4.9.51 8MQuad Beta
  - 4.9.11_1.0.0 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 12.04/14.04/16.04).
- 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 "***fmac***" release. The script displays both the "***fmac***" codename and latest kernel version supported by that release. Currently two "***fmac***" releases are supported: "***battra***" (most recent and up-to-date regarding fixes and enhancements), and "***orga***". Murata strongly recommends using "***battra***" release.
- 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 must** create additional folders.
- Select i.MX target: refer to **Table 6** for more details.
- If the i.MX target selected supports alternative VIO options (i.e. i.MX6UL(L), i.MX6SoloX, and i.MX7ULP all support 1.8V VIO signaling), then the script prompts the user for desired configuration. 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). Only use this if you want to slow down the SDIO clock for debugging purposes. **NOTE:** non-default VIO (i.e. 1.8V instead of 3.3V) requires hardware rework.
- 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**" – already created by this point):

```
./Murata_Wireless_Yocto_Build.sh
```

The script goes through the following stages:

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

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

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

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

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

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

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

With SD card image built and located, refer to **Section 4** for preparing bootable SD card.

3.4 Additional Software Considerations

3.4.1 Out-Of-Band IRQ on i.MX 6UL/6ULL EVK’s

The default interrupt configuration for the Linux 4.1.15, and 4.9.11 releases is to use OOB IRQ. However, there are two exceptions to this default interrupt configuration: the i.MX 6UL and i.MX 6ULL EVK’s. SDIO in-band interrupts have been configured (as an option) for both these platforms due to the default i.MX hardware configuration.

Murata recommends that the user runs OOB IRQ (if possible) on either of these two platforms, the necessary rework is quite straightforward: refer to the Hardware User Manual for specific instructions. This document provides the necessary software steps to run either interrupt configuration.

<u>**NOTE:**</u> For OOB IRQ configuration on i.MX6UL/ULL EVK, the second (of two) Ethernet ports is disabled due to hardware conflict (documented in Hardware User Manual steps).

3.4.2 Murata Support for Edge-Sensitive Interrupts in OOB IRQ Configuration

When running OOB IRQ’s, the default configuration is for level-sensitive interrupts. However, Type 1BW generates edge-sensitive interrupts over WL_HOST_WAKE line. To support edge-sensitive interrupts, the user needs to refer to the Linux User Manual for specific steps to modify the relevant DTS file.

3.4.3 UHS SDIO 3.0 operation on i.MX 6UL(L) and 6SoloX Platforms

To achieve high WLAN throughput performance for 802.11ac modules, it is necessary to configure the SDIO bus to 1.8V VIO signaling level for SDIO 3.0 (UHS) mode. However due to hardware limitations on the i.MX interconnect, there is a voltage level issue for WL_HOST_WAKE (used in OOB IRQ configuration). The WLAN module drives this line at 1.8V but the i.MX host CPU has a fixed voltage rail of 3.3V VIO. As such the OOB IRQ operation is not guaranteed to work. Murata recommends that the user configures for SDIO in-band interrupts for this configuration.

4 Preparing Bootable SD Card for i.MX with Murata Wi-Fi/BT EVK

4.1 Linux PC Steps to Flash SD Card

Now that the SD card image is built, we can now flash the (micro) SD card used for booting the i.MX platform. Insert the (micro) SD card into host machine (PC). It is imperative that the (micro) SD card comes up as "/dev/sdx" device. If it does not, then you may require a USB to (micro) SD card adapter as shown in **Figure 4**. This Kingston device (MobileLite G4) provides direct plug-ins for microSD and SD cards. It supports USB 3.0 and UHS SD cards – allowing very fast transfer speeds. With the "right" (micro) SD Card Reader/Writer and UHS (micro) SD card, flashing a 1 GB i.MX image can be done in 10~20 seconds versus 1~2 minutes (or more).

Figure 4: USB to SD Card Reader/Writer Adapter

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

```
dmesg
```

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

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

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

NOTE: Before running next command, make sure you have selected the correct device. Otherwise you may unintentionally wipe/erase your hard drive! Substitute the correct (micro) SD device name for "***/dev/sdx***" in "***dd***" command line below.

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

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

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

4.2 Windows PC Steps to Flash SD Card

In case you need to later flash the same SD card image using a Windows PC, the following steps have been included. Windows utilities such as "Win32 Disk Imager" or "NetBSD Disk Image Tool" can be used to flash the (micro) SD card. For example, when using "Win32 Disk Imager", follow these steps:

- After bringing up "Win32 Disk Imager" program[10], click on the folder icon/button and navigate to the location of the desired "*.sdcard" file. You need to change "*.img" file type to "*.*" to select/open the "*.sdcard" file.
- Select the "Device" button and select the drive letter corresponding to the (micro) SD card: formatting SD card may be necessary beforehand with Windows low level utilities[11].
- Now click the "Write" button. A warning window will pop up that warns that the device being written to may be corrupted.
- Upon completion, a window with "Write Successful" should appear.
- Click "OK" on "Write Successful" window.
- Now click "Exit" on "Win32 Disk Imager" window.
- To be safe, you may elect to "eject" the SD card removable memory device before removing it.

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

5 Murata Wi-Fi/BT Bring-Up on i.MX6 Platforms

Any of the (WLAN-SDIO) Murata Wi-Fi/BT EVK's listed in
Table 4 can be connected to the i.MX6 Platforms. The following sub-sections details steps for bringing up Wi-Fi/BT on the four major variants of the i.MX6 Platform. The specific steps described only vary slightly for the Murata Type 1CK, ZP, 1BW, 1DX, and 1FX modules given the variances in EVB used and antenna.

Murata Type 1CK EVK provides an additional step with connecting the interposer board. **Figure 5** shows a block diagram of the Type 1CK interconnect. Refer to **Figure 6** for the **correct orientation** of Type 1CK module and interposer board.

Figure 5: Murata Type 1CK Interconnect Block Diagram

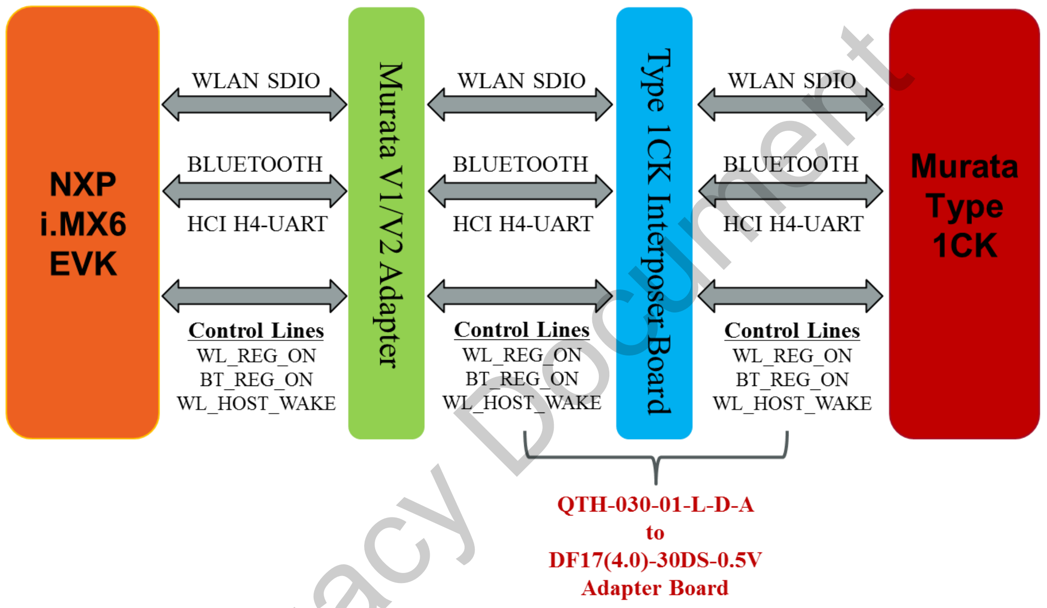
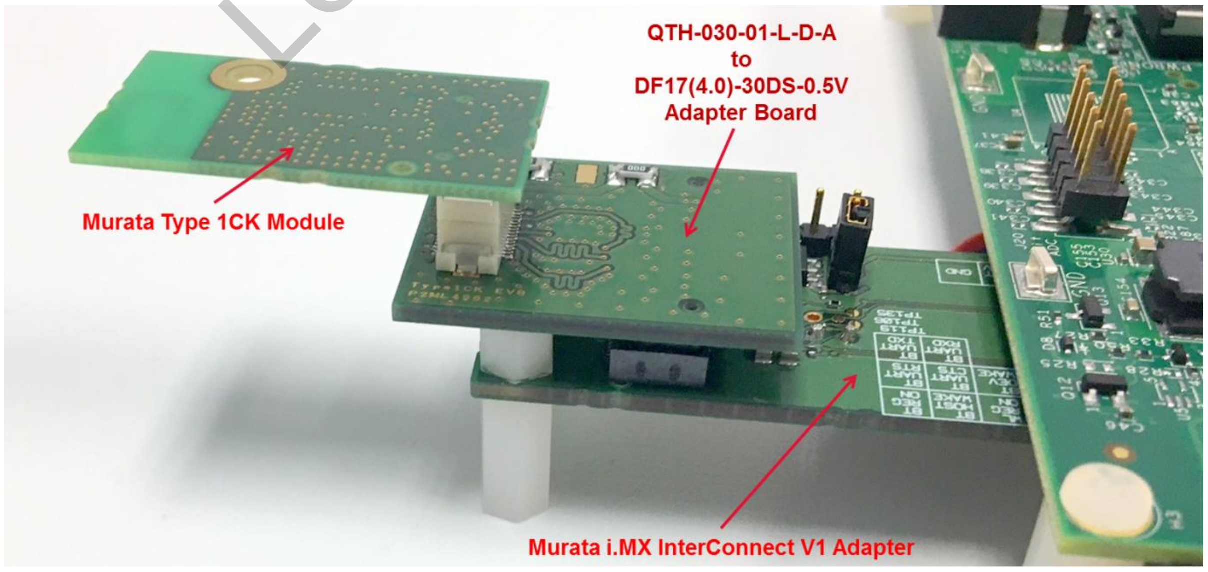


Figure 6: Murata Type 1CK Interconnect to NXP i.MX 6SoloX



QTH-030-01-L-D-A to DF17(4.0)-30DS-0.5V Adapter Board
Murata Type 1CK Module
Murata i.MX InterConnect V1 Adapter

5.1 Connecting to i.MX 6SoloX SDB

Referring to **Table 6,** the V1 Adapter is the correct adapter for this platform. V1 Adapter rework is only required for 1.8V VIO operation (necessary for SDIO 3.0 UHS mode). The Murata Wi-Fi/BT EVB is connected via the SD3/SD2 slots: see **Figure 7** below.

[1] Ensure no power is applied to i.MX 6SoloX SDB. Connect J16 micro-USB port to PC and start terminal emulator: “minicom” on Linux or “tera term” on Windows. Set port to 115200-N-8-1.
[2] Check that VIO setting on V1 Adapter (Part #2 in **Table 5**) is set to desired VIO: 3.3V (VBAT_SDIO) or 1.8V (VIO_REG_OUT). Refer to **Section 0** for specifics.
[3] Insert V1 Adapter board into SD3 slot and SD Card Extender into SD2 slot. Note the orientation as shown in **Figure 7**.
[4] Connect antenna (Part #4) to the SMA connector of the Murata EVB (Part #1).
[5] Now you can connect the EVB to the 60-pin Samtec connector on the V1 Adapter board.
[6] Prepare SD card to boot platform per **Section 4**. Insert SD card, power on platform and interrupt at u-boot. Now type:

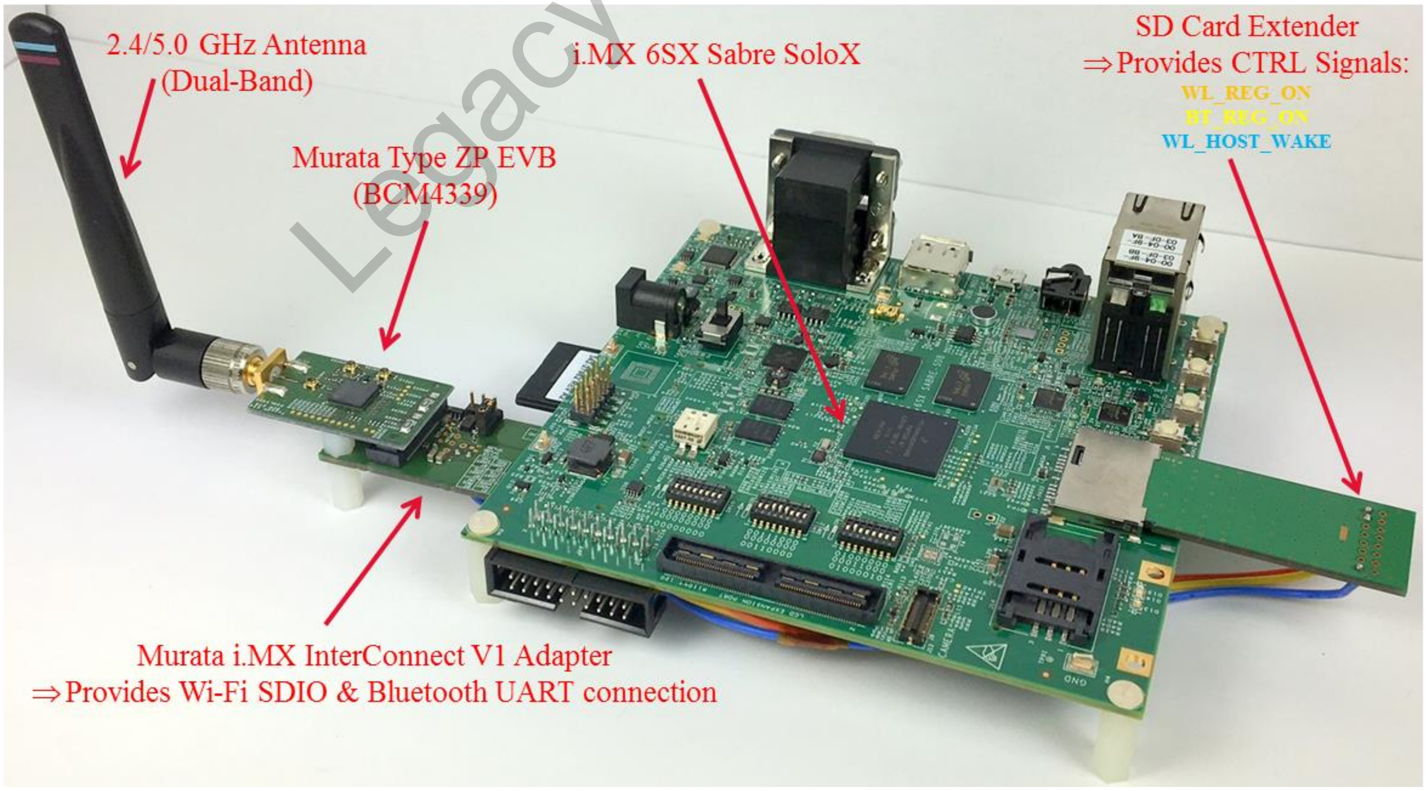
```
setenv fdt_file imx6sx-sdb-btwifi.dtb
```

Now save the u-boot configuration and boot the platform:

```
saveenv
boot     ← causes platform to boot kernel
```

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

Figure 7: i.MX 6SoloX SDB with V1 Adapter and Type ZP EVB



5.2 Connecting to i.MX 6SoloLite EVK

Referring to **Table 6**, V1 Adapter is the only solution that will work for this platform. Murata Wi-Fi/BT EVB is connected via SD1 slot with control signals connected from SD3 slot using SD Card Extender: see **Figure 8** below.

[1] Ensure no power is applied to i.MX 6SL EVK. Connect J26 micro-USB port to PC and start terminal emulator: "minicom" on Linux or "tera term" on Windows. Set port to 115200-N-8-1.
[2] Check that VIO setting on V1 Adapter (Part #2 in **Table 5**) is set to the supported voltage of 3.3V (VBAT_SDIO). Refer to **Section 0** for specifics
[3] Insert V1 Adapter board into SD1 slot and SD Card Extender into SD3 slot. Note the orientation as shown in **Figure 8**.
[4] Connect antenna (Part #4) to the SMA connector of the Murata EVB (Part #1).
[5] Now you can connect the EVB to the 60-pin Samtec connector on the V1 Adapter board.
[6] Prepare SD card to boot platform per **Section 4**. Insert SD card, power on platform and interrupt at u-boot. Now type:

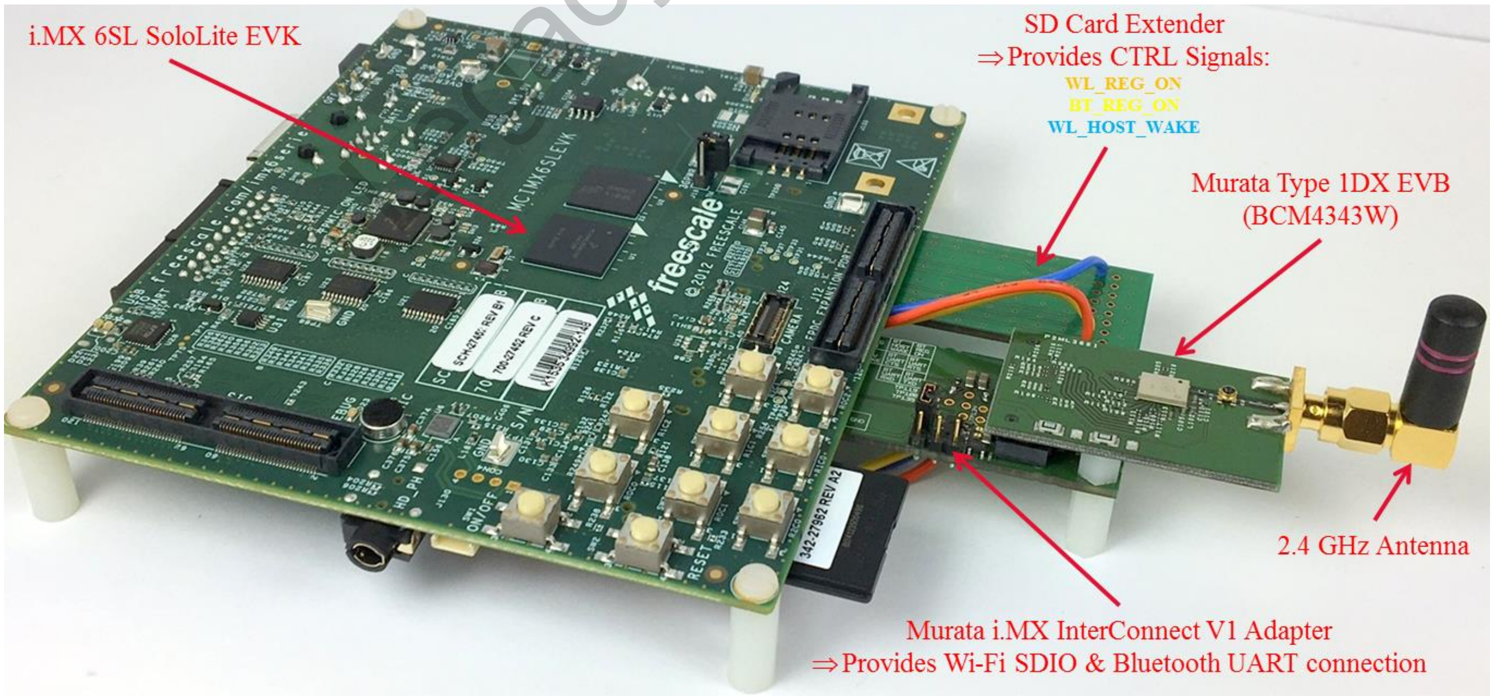
```
setenv fdt_file imx6sl-evk-btwifi.dtb
```

Now save the u-boot configuration and boot the platform:

```
saveenv
boot    🡐 causes platform to boot kernel
```

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

Figure 8: i.MX 6SoloLite EVK with V1 Adapter and Type 1DX EVB



5.3 Connecting to i.MX 6Q/DL SDB/SDP or i.MX 6 QP SDB

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

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

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

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

5.3.2 Specific Hardware Considerations for i.MX 6QP SDB

Depending on the revision, rework **may be required** for the i.MX 6QuadPlus SDB. This rework connects the Bluetooth UART and Wi-Fi/BT control signals (WL_REG_ON, BT_REG_ON, and WL_HOST_WAKE). Revision B of i.MX 6QP SDB populates the necessary resistors for connecting BT UART and Wi-Fi/BT control signals. If your board has revision earlier than B (i.e. A2) then you will have to populate the necessary resistors. Refer to the Hardware User Manual for necessary rework. For the Rev A2 board, page 15 of the NXP schematic (SPF-28857_A2.pdf) correctly captures the necessary rework to be done. Note the much simpler rework on the i.MX 6QB SDB given that the no special "crossing" of TX and RX resistor pads is necessary. Those schematic notes are repeated below.

NOTE: To use J13, populate resistors R209 - R213 and depopulate the SPI NOR FLASH U14.

5.3.3 Murata Wi-Fi/BT EVK Bring-Up on i.MX 6Q/DL SDB/SDP or i.MX 6 QP SDB

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

[1] Ensure no power is applied to i.MX 6QP SDB or i.MX 6Q/DL SDB/SDP. Connect J509 micro-USB port to PC and start terminal emulator: “minicom” on Linux or “tera term” on Windows. Set port to 115200-N-8-1.
[2] Check that VIO setting on Murata i.MX6 Interconnect V2 Adapter (Part #3 in **Table 5**) is set to 3.3V (VBAT_SDIO). Refer to **Section 9.2.1** for specifics.
[3] After connecting ribbon cable to both adapter and i.MX6 platform, insert Adapter board into SD2 slot. Note the orientation as shown in **Figure 9**.
[4] Connect antenna (Part #4) to the SMA connector of the Murata EVB (Part #1).
[5] Now you can connect the EVB to the 60-pin Samtec connector on the Adapter board.
[6] Prepare SD card to boot platform per **Section 4**. Insert SD card, power on platform and interrupt at u-boot. Now type:

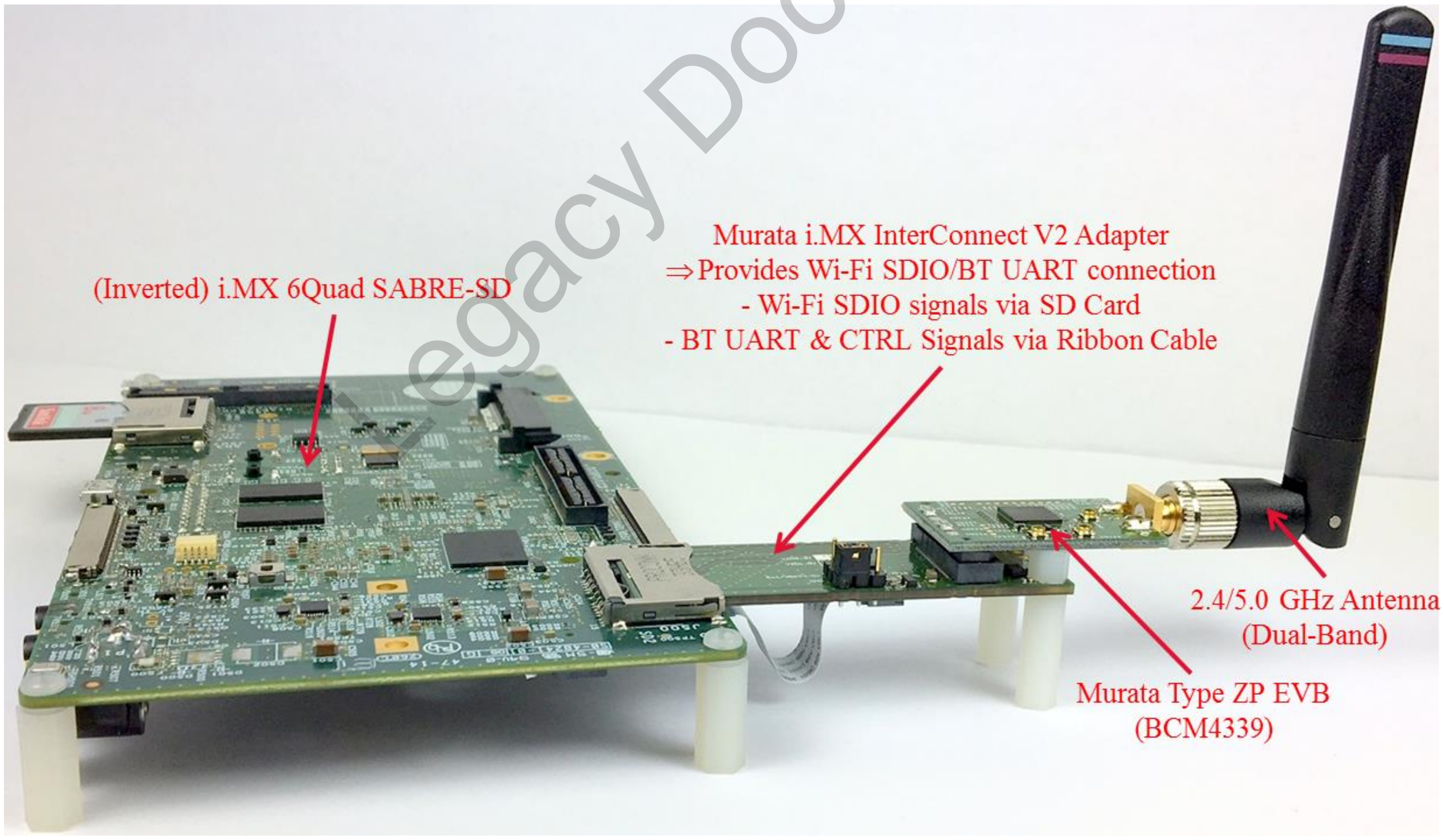
```
setenv fdt_file imx6q-sabresd-btwifi.dtb (or imx6dl-sabresd-btwifi.dtb or imx6qp-sabresd-btwifi.dtb)
```

Now save the u-boot configuration and boot the platform:

```
saveenv
boot  ← causes platform to boot kernel
```

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

Figure 9: i.MX 6Quad/DualLite SDB (Inverted) with V2 Adapter and Type ZP EVB


(Inverted) i.MX 6Quad SABRE-SD
Murata i.MX InterConnect V2 Adapter
⇒Provides Wi-Fi SDIO/BT UART connection
- Wi-Fi SDIO signals via SD Card
- BT UART & CTRL Signals via Ribbon Cable
2.4/5.0 GHz Antenna (Dual-Band)
Murata Type ZP EVB (BCM4339)

5.4 Connecting to i.MX 6UltraLite EVK or i.MX 6ULL EVK

The i.MX 6UL and i.MX 6ULL EVK's share the same baseboard, with V2 Adapter as interconnect. V2 Adapter rework is required to support 1.8V VIO operation (necessary for SDIO 3.0 UHS mode).

[1] Ensure no power is applied to i.MX 6UltraLite or i.MX 6ULL EVK. Connect J1101 micro-USB port to PC and start terminal emulator: "minicom" on Linux or "tera term" on Windows. Set port to 115200-N-8-1.
[2] Check that VIO setting on V2 Adapter (Part #3 in **Table 5**) is set to desired voltage: 3.3V or 1.8V. Refer to **Section 9.2** for specifics.
[3] Connect antenna (Part #4) to the SMA connector of the Murata EVB (Part #1).
[4] Connect the EVB to the 60-pin Samtec connector on the Adapter board.
[5] Connect ribbon cable at both ends before inserting Murata EVK into SD1 slot. Note the orientation as shown in **Figure 10**. Make sure that the adapter clicks in correctly.
[6] Prepare micro SD card to boot platform per **Section 4**. Insert SD card, power on platform and interrupt at u-boot. Now type (remove "<..>" string for SDIO in-band interrupts; include "-oob" string for OOB IRQ configuration):

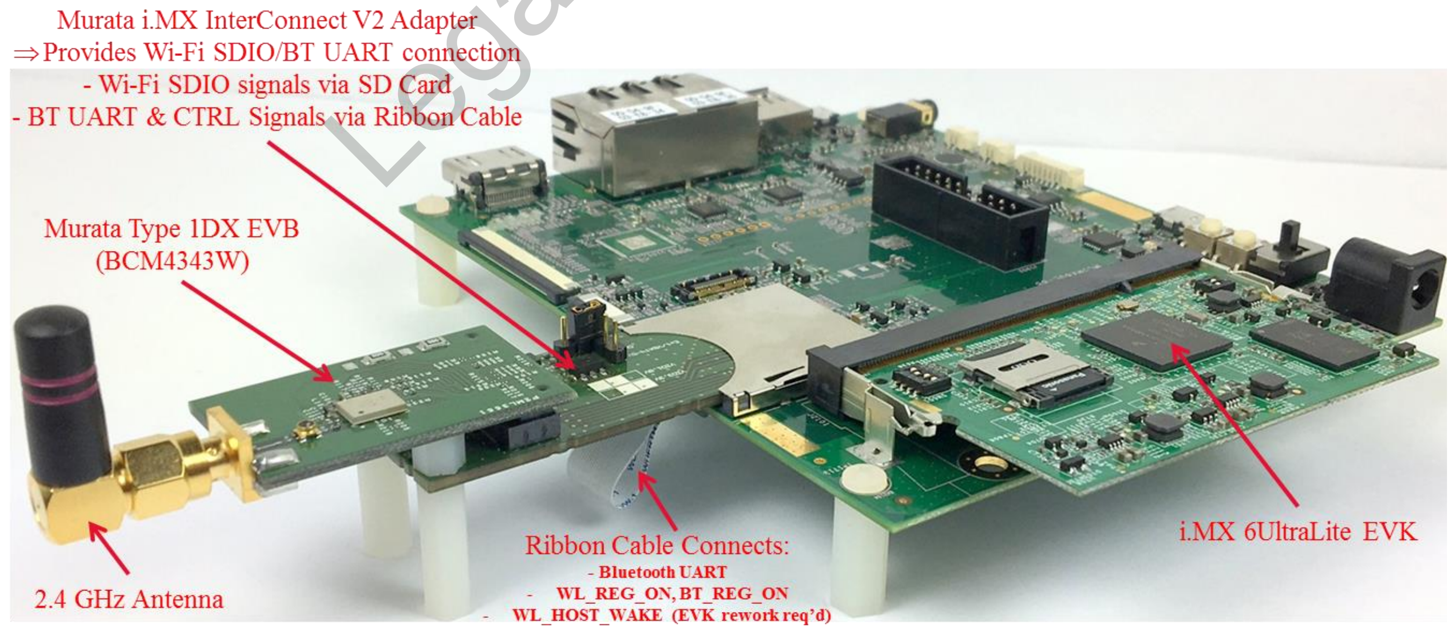
```
setenv fdt_file imx6ul-14x14-evk-btwifi<-oob>.dtb
(OR imx6ul-9x9-evk-btwifi<-oob>.dtb, imx6ull-14x14-evk-btwifi<-oob>.dtb, imx6ull-9x9-evk-btwifi<-oob>.dtb)
```

Now save the u-boot configuration and boot the platform:

```
saveenv
boot   🡐 causes platform to boot kernel
```

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

Figure 10: i.MX 6UltraLite EVK with V2 Adapter and Type 1DX EVB

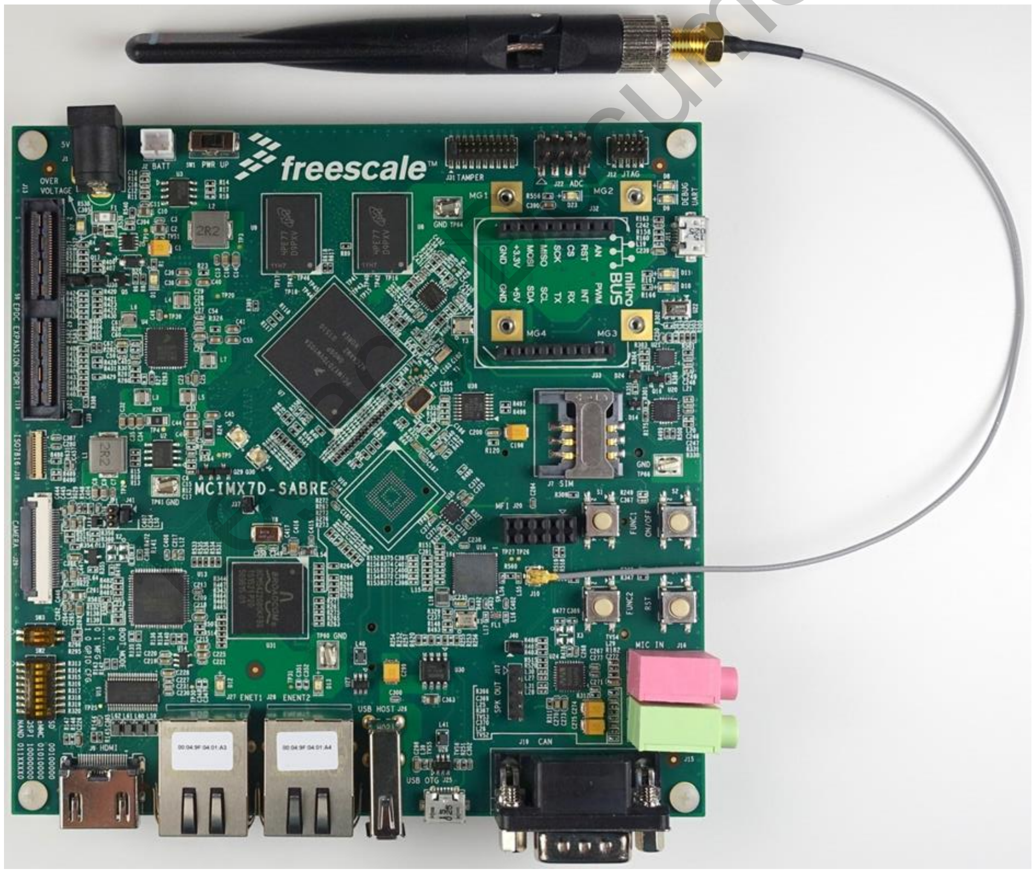

Murata i.MX InterConnect V2 Adapter
⇒Provides Wi-Fi SDIO/BT UART connection
- Wi-Fi SDIO signals via SD Card
- BT UART & CTRL Signals via Ribbon Cable
Murata Type 1DX EVB
(BCM4343W)
2.4 GHz Antenna
Ribbon Cable Connects:
- Bluetooth UART
- WL_REG_ON, BT_REG_ON
- WL_HOST_WAKE (EVK rework req'd)
i.MX 6UltraLite EVK

6 Bringing up Wi-Fi/BT on i.MX 7Dual SABRE Development Board

The NXP i.MX 7Dual SDB (see **Figure 11**) has the Murata Type ZP module soldered down on the board. As such there is no need to use the Murata InterConnect Kit. For customers wanting to evaluate another Murata module other than Type ZP, the SD card slot interface on the i.MX7D SDB is not recommended. The SD card slot is required for booting the platform[12].

[1] The i.MX 7Dual SABRE Development Board does not ship with an antenna. As such the WLAN sensitivity (RF reception) is attenuated by approximately 30 dBm. Murata strongly recommends that the user obtain the necessary antenna to operate Wi-Fi/BT properly[13].
[2] Connect J11 micro-USB port to PC and start terminal emulator: “minicom” on Linux or “tera term” on Windows. Set port to 115200-N-8-1.
[3] Prepare SD card to boot platform per **Section 4**.
[4] Insert SD card and power on platform: no need to interrupt u-boot as the default DTB file integrates support for Wi-Fi and Bluetooth.
[5] Refer to **Section 8** to test/verify Wi-Fi and Bluetooth functionality.

Figure 11: i.MX 7Dual SDB with Murata Type ZP



[12] The i.MX 7Dual SDB can be configured to flash u-boot and the kernel to the onboard QSPI-NOR. The root file system can then be NFS-mounted. However this configuration (to free up SD card slot) is very complicated and not supported in this document.
[13] Possible antenna/cable solution would be the following: ANT-DB1-RAF-SMA and 931-1188-ND.

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

The NXP i.MX 8MQuad EVK (see **Figure 12**) provides a secondary Wi-Fi/BT solution on the underside via a M.2 connector. Murata provides a M.2 interposer board that allows the customer to evaluate a PCIe-based WLAN solution such as Type 1CX or 1DK. The M.2 interconnect provides WLAN PCIe, BT UART, control signals, and optionally BT PCM (if necessary rework done on i.MX 8MQuad EVK). Please contact Murata directly for more support on this platform. Bring-up steps are straightforward:

[1] Connect J1701 micro-USB port to PC and start terminal emulator: "minicom" on Linux or "tera term" on Windows. Set port to 115200-N-8-1.
[2] Prepare micro SD card to boot platform per **Section 4**.
[3] Insert SD card and power on platform and interrupt at u-boot. Now type:

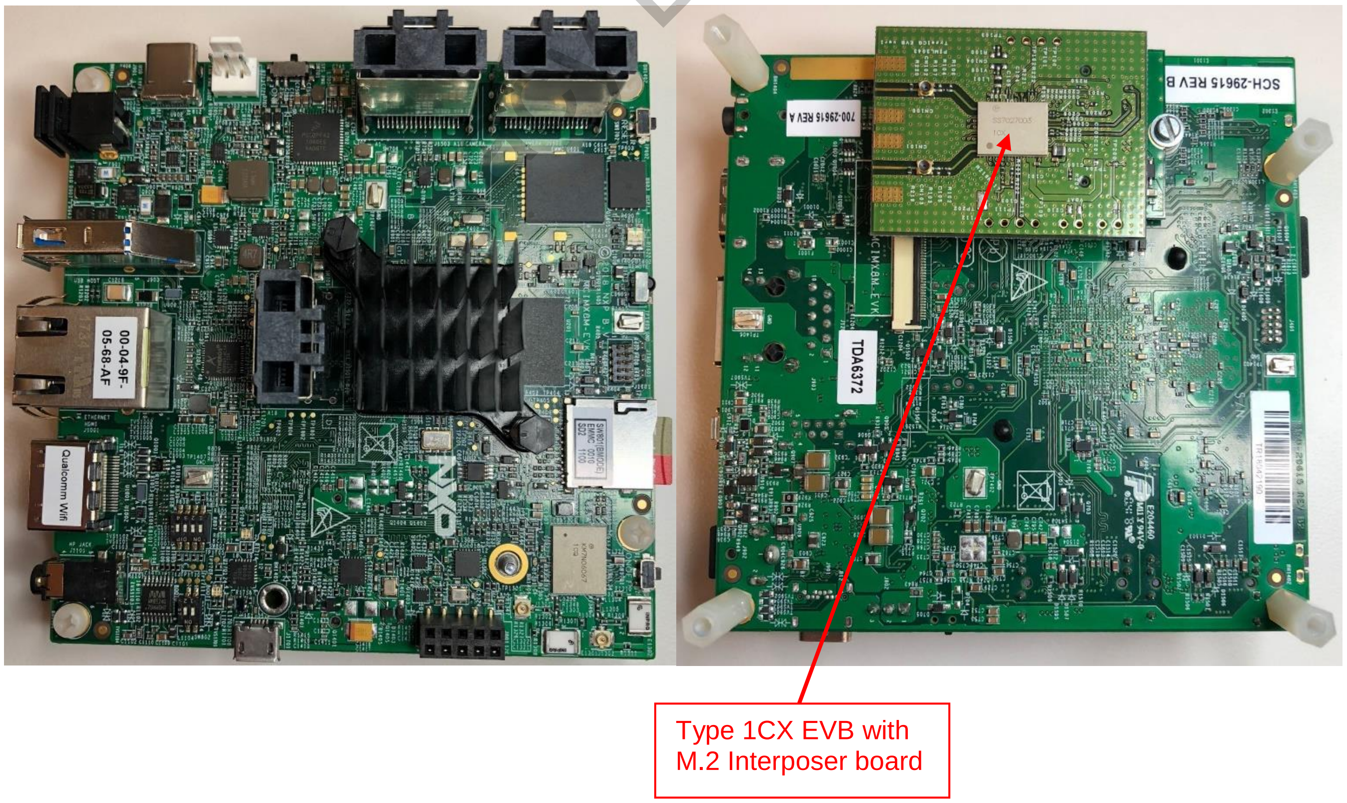
```
setenv fdt_file fsl-imx8mq-evk-pcie1-m2.dtb
```

Now save the u-boot configuration and boot the platform:

```
saveenv
boot  ← causes platform to boot kernel
```

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

Figure 12: i.MX 8MQuad with Type 1CX (top & bottom views)



8 Test/Verification of Wi-Fi and Bluetooth

Now the kernel should be booting correctly on the platform with Murata Wi-Fi/BT EVK pugged in (i.e. already set correct DTB file when interrupting u-boot). Next steps are to verify Wi-Fi and Bluetooth functionality. The NXP i.MX (Murata modified) images include all the necessary files to support Wi-Fi and Bluetooth bring-up/testing/verification. The relevant folders and files are summarized in **Table 10: Embedded Wi-Fi/Bluetooth Files**.

Table 10: Embedded Wi-Fi/Bluetooth Files

Filename or Folder	Details
/lib/firmware/brcm/brcmfmac<chipset>-sdio.bin	“fmac” firmware file for WLAN SDIO chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-sdio.txt	“fmac” nvram file for WLAN SDIO chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-sdio.clm_blob	“fmac” regulatory binary for WLAN SDIO chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-pcie.bin	“fmac” firmware file for WLAN PCIe chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-pcie.clm_blob	“fmac” regulatory binary for WLAN PCIe chipset/module.
/lib/firmware/brcm/brcmfmac<chipset>-pcie.txt	“fmac” nvram file for WLAN PCIe chipset/module.
/lib/firmware/BCM4354A2.1CX.hcd	Type 1CX (CYW4356) Bluetooth patchfile.
/etc/firmware/CYW43012C0.1LV.hcd	Type 1LV (CYW43012) Bluetooth patchfile.
/etc/firmware/CYW4350C0.1BB.hcd	Type 1BB (CYW4354) Bluetooth patchfile.
/etc/firmware/CYW4345C0.1MW.hcd	Type 1MW (CYW43455) Bluetooth patchfile.
/etc/firmware/BCM4335C0.ZP.hcd	Type ZP and 1CK (both CYW4339 based) Bluetooth patchfile.
/etc/firmware/BCM43341B0.1BW.hcd	Type 1BW (CYW43340) Bluetooth patchfile.
/etc/firmware/BCM43430A1.1DX.hcd	Type 1DX (CYW4343W) Bluetooth patchfile.
/usr/sbin/wl	WL tool is used for Wi-Fi RF and manufacturing tests. Also useful tool for initial bring-up and debug.
/usr/sbin/iw	Linux “iw” executable.
/usr/sbin/wpa_supplicant	WPA supplicant executable.
/usr/sbin/wpa_cli	WPA CLI tool.
/usr/bin/wpa_passphrase	WPA Passphrase generator.
/etc/wpa_supplicant.conf	WPA supplicant configuration file.
/usr/sbin/hostapd	Hostapd executable – manages wireless link in Soft AP mode.
/usr/sbin/hostapd_cli	Hostapd CLI tool.
/etc/hostapd.conf	Hostapd configuration file.
/etc/udhcpd.conf	DHCP server configuration file.
/etc/network/interfaces	Modify this file to automatically bring up “wlan0” interface. Also assign static IP address if desired. Currently not configured.
/usr/bin/hciattach	“hciattach” binary – used for initializing Bluetooth UART connection.
/usr/bin/hciconfig	“hciconfig” binary – used for configuring Bluetooth interface.
/usr/bin/hcitool	“hcitool” binary – used for controlling Bluetooth interface.
/usr/bin/iperf3	iPerf throughput test tool.

8.1 Wi-Fi Interface Test/Verification

8.1.1 Useful Environment Setup on NXP Linux

Once the kernel has booted, there are a couple of quick commands (sequence is important) which make the terminal console easier to work on:

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

8.1.2 Bringing Up Wi-Fi Interface

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

- i.MX 6ULL EVK
- V2 Adapter configured for 1.8V VIO
- Type 1MW (CYW43455) EVB
- Murata i.MX image compiled for i.MX6ULL @1.8V

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

```
sdhci: Secure Digital Host Controller Interface driver
sdhci: Copyright(c) Pierre Ossman
sdhci-pltfm: SDHCI platform and OF driver helper
sdhci-esdhc-imx 2190000.usdhc: allocated mmc-pwrseq
sdhci-esdhc-imx 2190000.usdhc: assigned as wifi host     🡐 "wifi-host" descriptor in "mmc0" entry
Murata: mmc_power_up: Setting 1.8V for Index: 0          🡐 forces 1.8V VIO operation in kernel
mmc0: SDHCI controller on 2190000.usdhc [2190000.usdhc] using ADMA
Murata: mmc_sdio_init_card: Skipping 1.8V setting for Index: 0
mmc0: queuing unknown CIS tuple 0x80 (2 bytes)           🡐 WLAN device enumeration
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: new ultra high speed SDR104 SDIO card at address 0001     🡐 switch to UHS mode (SDR104)
…
Murata: mmc_sdio_init_card: Skipping 1.8V setting for Index: 0
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)
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: Murata Customized Version: imx-morty-battra_r1.0;
…
brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: Feb  7 2018 02:58:50 version 7.45.151 (r683645 CY) FWID
01-baba21a4
```

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

- "***brcmfmac***" string identifies "fmac" driver log messages
- "***brcmfmac43455-sdio.bin***" indicates the WLAN firmware file being loaded.
- "***Murata Customized Version***" indicates that "***meta-murata-wireless***" was used to generate this image. The branch or release/tag information is included.
- "***Firmware version***" indicates specific version of firmware being loaded by "***fmac***".

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

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

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

NOTE: no IP address is assigned yet to the "wlan0" interface. That will be done later **Section 8.1.6**.

8.1.3 STA/Client Mode: Scan for Visible Access Points

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

8.1.3.1 Using "wl" Tool

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

- "WL Tool Instructions" on "My Murata->Common Documents"
- "WL Tool for Embedded 802.11 Systems" on Cypress Linux Support Portal.

Now that Wi-Fi interface is up and running (having loaded the default “***brcmfmac<chipset>-sdio.bin***” – STA/Client mode firmware), let’s do some basic testing to verify functionality. The “***wl scan***” command will initiate an active probe of all visible SSID’s. The “***wl scanresults***” will return a list of visible SSID’s.

In following example, one active AP has SSID of “***Murata_5G***”. Here are expected results:

```
$ wl scan
$ wl scanresults
brcmfmac: brcmf_cfg80211_vndr_cmds_dcmd_handler: oversize return buffer 130048
SSID: "Murata_5G"  🡐 Broadcast SSID, with RSSI (received signal strength) one line below
Mode: Managed   RSSI: -38 dBm   SNR: 0 dB     noise: 0 dBm    Flags: RSSI on-channel  Channel: 36/80
BSSID: 84:1B:5E:F6:A7:60      Capability: ESS
Supported Rates: [ 6(b) 9 12(b) 18 24(b) 36 48 54 ]
Extended Capabilities: BSS_Transition
VHT Capable:
        Chanspec: 5GHz channel 42 80MHz (0xe02a)
        Primary channel: 36     🡐 Channel Number
        HT Capabilities: 40Mhz SGI20 SGI40  🡐 802.11ac bandwidth (40 MHz in this case)
        Supported HT MCS : 0-23
        Supported VHT MCS:
                NSS1 Tx: 0-9   Rx: 0-9
                NSS2 Tx: 0-9   Rx: 0-9
                NSS3 Tx: 0-9   Rx: 0-9
WPS: V2.0 Configured
VS_IE:dd7c0050f204104a0001101044000102103b000103104700101b52c4d5ffb10ad163f39b91a979382c102
1000d4e4554474541522c20496e632e102300055236333030102400055236333030104200043435333610540
00800060050f204000110110005523633303010080002200810 3c000103104900060 0372a000120
VS_IE:dd090010180200001c0000
VS_IE:dd180050f20201018c0003a4000027a400004243bc0062326600

…… etc.  🡐 More SSID listings follow here.
```

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

8.1.3.2 Using "iw" Linux Command

"***iw***" is the default Linux command line tool for controlling a WLAN interface. It provides an alternative to "***wl***" tool. One useful link to learn more about "***iw***" is on the "**Linux Wireless wiki**" here. In following example of listing WLAN devices and performing a scan, one active AP has SSID of "***Murata_5G***". Here are expected results:

```
$ iw dev  🡐 list available WLAN devices
phy#0
        Interface wlan0
                ifindex 6
                wdev 0x1
                addr b8:d7:af:56:61:fc
                type managed
                channel 36 (5180 MHz), width: 20 MHz, center1: 5180 MHz
                txpower 31.00 dBm
```

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

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

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

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

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

NOTE: In the following test sequences of using "***wl***" tool or "***iw***" command, the WLAN interface is not assigned an IP address. That is done later in **Section 8.1.6** where connectivity testing is performed.

8.1.4.1 Using "wl" Tool

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

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

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

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

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

```
$ wl assoc  ← check association status
SSID: "Murata_5G"
Mode: Managed   RSSI: -44 dBm   SNR: 0 dB      noise: -91 dBm  Flags: RSSI on-channel  Channel: 36/80
BSSID: 84:1B:5E:F6:A7:60       Capability: ESS
Supported Rates: [ 6(b) 9 12(b) 18 24(b) 36 48 54 ]
Extended Capabilities: BSS_Transition
VHT Capable:
        Chanspec: 5GHz channel 42 80MHz (0xe02a)
        Primary channel: 36
        HT Capabilities: 40Mhz SGI20 SGI40
        Supported HT MCS : 0-7
        Supported VHT MCS:
                NSS1 Tx: 0-9   Rx: 0-9
                NSS2 Tx: 0-9   Rx: 0-9
                NSS3 Tx: 0-9   Rx: 0-9
WPS: V2.0 Configured
VS_IE:dd310050f204104a000110104400010210470010 1b52c4d5ffb10ad163f39b91a979382c103c000103104
9000600372a000120
VS_IE:dd090010180200001c0000
VS_IE:dd180050f20201018c0003a4000027a400004243bc0062326600
VS_IE:dd7c0050f204104a00011010440001 02103b000103104700101b52c4d5ffb10ad163f39b91a979382c102
1000d4e4554474541522c20496e632e1023000552363330301024000552363330301042000434353336105 40
00800060050f2040001101100055236333030100800022008103c0001031049000600372a000120
```

8.1.4.2 Using “iw” Linux Command

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

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

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

```
$ iw dev wlan0 link
Connected to 84:1b:5e:f6:a7:60 (on wlan0)
        SSID: Murata_5G
        freq: 5180
        RX: 1944 bytes (8 packets)
```

```
    TX: 0 bytes (0 packets)
    signal: -44 dBm

    bss flags:
    dtim period:    2
    beacon int:     100
```

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

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

<u>**Important Dependencies:**</u>

- WLAN device must be configured correctly (refer to **Section 8.1.2**).
- WPA supplicant must be up and running.

8.1.5.1 Using "wpa_cli" Command

"***wpa_cli***" can only be invoked once the "***wlan0***" interface is configured and the WPA supplicant is running. The user can follow this command sequence to establish a secure client connection to an Access Point with WPA2-PSK authentication.

Prior to running "***wpa_cli***", you might like to back up default/previous "***/etc/wpa_supplicant.conf***" file in case you overwrite it:

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

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

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

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

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

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

```
> remove_network all  🡐 tear down any existing network connections
OK
<3>CTRL-EVENT-DISCONNECTED bssid=b0:00:b4:65:e0:60 reason=3 locally_generated=1
> status              🡐 check status
wpa_state=INACTIVE
p2p_device_address=ba:d7:af:56:61:fc
address=b8:d7:af:56:61:fc
uuid=a3178764-c106-57ee-8ec2-6bf2e0607166
> scan                  🡐 initiate a scan
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>CTRL-EVENT-NETWORK-NOT-FOUND
> scan_results          🡐  list results of scan
bssid / frequency / signal level / flags / ssid
84:1b:5e:f6:a7:60    5180   -38    [WPA2-PSK-CCMP][WPS][ESS]    Murata_5G
84:1b:5e:f6:a7:61    2412   -36    [WPA2-PSK-CCMP][WPS][ESS]    Murata_2G
…
> add_network  🡐 add a network. This returns integer value which is then used for setting parameters.
0
> set_network 0 ssid "Murata_5G"             🡐 set SSID to “Murata_5G”
OK
> set_network 0 psk "your_passphrase"        🡐 set WPA passphrase
OK
> enable 0  🡐 enable network connection. If ssid and passphrase set correctly, connection will be established.
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>Trying to associate with SSID 'Murata_5G'
<3>Associated with 84:1b:5e:f6:a7:60             🡐 connection established
<3>CTRL-EVENT-CONNECTED - Connection to 84:1b:5e:f6:a7:60 completed [id=0 id_str=]
<3>CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
 > status               🡐 now verify that connection is established
bssid=84:1b:5e:f6:a7:60
freq=5180
ssid=Murata_5G
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
```

```
wpa_state=COMPLETED
p2p_device_address=ba:d7:af:56:61:fc
address=b8:d7:af:56:61:fc
uuid=a3178764-c106-57ee-8ec2-6bf2e0607166
> save_config      🡐 save current configuration: this overwrites "/etc/wpa_supplicant.conf" file!
OK
> quit             🡐 exit wpa_cli interactive mode
```

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

```
$ more /etc/wpa_supplicant.conf

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

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

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

8.1.5.2 Using "wpa_supplicant.conf" file

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

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

network={
        key_mgmt=NONE
}
```

With the default configuration, the WPA supplicant will establish a connection with any random Access Point that has no authentication scheme enabled (i.e. "open").

Using "***Murata_5G***" SSID example, the relevant/modified contents of the "/***etc/wpa_supplicant.conf***" file (already shown in **Section 8.1.5.1)** is:

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

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

To establish a secured WPA2-PSK connection by only modifying "***/etc/wpa_supplicant.conf***" file, we need to follow these steps:

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

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

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

Verify that connection is re-established with Access Point:

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

        bss flags:
        dtim period:    2
        beacon int:     100
```

8.1.6 STA/Client Mode: Basic WLAN Connectivity Testing

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

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

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

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

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

```
$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=12.686 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=10.227 ms
64 bytes from 192.168.1.1: seq=2 ttl=64 time=10.053 ms
64 bytes from 192.168.1.1: seq=3 ttl=64 time=10.341 ms
64 bytes from 192.168.1.1: seq=4 ttl=64 time=12.364 ms
^C     🡐 Enter <CTRL-C> to terminate ping session
--- 192.168.1.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss   🡐 Indicates that no packets were dropped
round-trip min/avg/max = 10.053/11.134/12.686 ms
```

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

8.1.7 Wi-Fi Direct Testing

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

Important Dependencies:

- WLAN device must be configured correctly (refer to **Section 8.1.2**).
- WPA supplicant must be up and running.

Prior to running "***wpa_cli***", you might like to back up default/previous "***/etc/wpa_supplicant.conf***" file in case you overwrite it:

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

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

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

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

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

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

```
$ ifconfig
…
p2p-wlan0-0 Link encap:Ethernet  HWaddr ba:d7:af:56:e1:fc       🡐 new P2P interface
       inet6 addr: fe80::b8d7:afff:fe56:e1fc/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:0 errors:0 dropped:0 overruns:0 frame:0
       TX packets:23 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:0 (0.0 B)  TX bytes:4525 (4.4 KiB)
```

```
wlan0     Link encap:Ethernet  HWaddr b8:d7:af:56:61:fc          🡐 existing “wlan0” interface
          inet addr:192.168.1.3  Bcast:192.168. 1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:1903 errors:0 dropped:0 overruns:0 frame:0
          TX packets:769 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:367182 (358.5 KiB)  TX bytes:76384 (74.5 KiB)
```

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

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

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

8.1.8 Soft AP or Wi-Fi Hot Spot Testing

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

Important Dependencies:

- “***hostapd***”, “***hostapd.conf***”, and “***udhcpd.conf***” files are present in file system: these are part of standard Murata i.MX customized release - see **Table 10**.

First off, we need to kill the WPA supplicant:

```
$ killall wpa_supplicant
```

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

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

After invoking the “hostapd” command, the following log is expected:

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

We can now associate from another client device and ping the “wlan0” interface (in this example 192.168.1.1).

For authenticating in secure fashion, just change the “/etc/hostapd.conf” file to uncomment/configure the “wpa” and “wpa_passphrase” lines correctly:

```
“#wpa=1” ➔ “wpa=1”
“#wpa_passphrase=secret passphrase” ➔ “wpa_passphrase=password123”
```

8.1.9 WLAN Manufacturing or RF Testing

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

- Obtain manufacturing test firmware tarball.
- Mount flashed (micro) SD card on Linux host PC.
- “***cd***” to “***lib/firmware/brcm***” folder.
- Switch user to “***root***”.
- Create “***mfgtest***” and “***production***” sub-folders in “***/lib/firmware/brcm***” folder.
- Backup existing “****.bin***” and ***“*.clm_blob***” files to “***production***” sub-folder.
- Copy manufacturing test firmware files (from “mfgtest” sub-folder) into “***lib/firmware/brcm***” folder.
- Insert (micro) SD card on i.MX platform and boot.
- Verify that “***WLTEST***” is logged when “***fmac***” driver loads.

On host, insert (micro) SD card and configure sub-folder for production and manufacturing test firmware:

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

On i.MX Target, insert modified (micro) SD card, boot platform and note firmware log message:

```
brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: Feb  7 2018 02:58:50 version 7.45.151 (r683645
CY) FWID 01-baba21a4
```

Now copy over “***mfgtest***” sub-folder contents to “***/lib/firmware/brcm***”:

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

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

```
brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: Feb  7 2018 02:55:41 version 7.45.151 (r683645
CY WLTEST) FWID 01-58ee80e2
```

NOTE: the version number of firmware for production and manufacturing test **should not differ**. In this example, both firmware versions are “***7.45.151***”.

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

```
$ killall wpa_supplicant
```

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

Once RF testing is done, you can easily revert to the normal production firmware:

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

8.2 Bluetooth Interface Test/Verification

For Murata modules supporting Bluetooth, we can verify the HCI UART connection by invoking “***hciattach***”, bringing up the interface with “***hciconfig***” and then invoking “***hcitool scan***” to see what Bluetooth devices are visible. The Bluetooth test commands vary depending on which UART port is connected to Bluetooth. With the new “***modem_reset***” construct in DTS(I) files, the Bluetooth core should come up in correct state to be initialized (i.e. as kernel boots, the Bluetooth core is reset and then taken out of reset). When the BlueZ “***hciattach***” call is invoked, a Bluetooth patchfile is pulled from the “/etc/firmware/” folder – refer to **Table 10** for more details. Here is the default command sequence to verify Bluetooth functionality:

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

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

Table 11 lists the BT_REG_ON and UART ports for the various i.MX platforms. Here is example output using i.MX 6Quad SDB with Type ZP module (no BT_REG_ON toggle):

```
$ hciattach /dev/ttymxc4 bcm43xx 3000000 flow -t 20
bcm43xx_init
Set Controller UART speed to 3000000 bit/s
Flash firmware /etc/firmware/BCM4335C0.ZP.hcd
Set Controller UART speed to 3000000 bit/s
Device setup complete
$ hciconfig hci0 up
$ hcitool scan
Scanning ...
        78:F7:BE:72:07:E6       SDK's GS4
```

Table 11: GPIO and UART Settings for Bluetooth Tests

i.MX6 Platform	GPIO/UART Configuration	Notes
i.MX 8MQuad EVK	GPIO69; UART3	
i.MX 7Dual SDB	GPIO119; UART6	
i.MX 6Q(P)/DL SDB/SDP	GPIO2; UART5	
i.MX 6SoloX SDB	GPIO171; UART3	
i.MX 6SoloLite EVK	GPIO145; UART4	
i.MX 6UL/ULL EVK	GPIO508; UART2	GPIO508 does not allow its direction to be set. Always output.

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

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

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

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

Lastly, here is an example sequence of toggling BT_REG_ON using i.IMX7D SDB platform, before invoking the BT functionality (verification) calls:

```
echo 119 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio119/direction
echo 0 > /sys/class/gpio/gpio119/value
sleep 0.1
echo 1 > /sys/class/gpio/gpio119/value
hciattach /dev/ttymxc5 bcm43xx 3000000 flow -t 20
hciconfig hci0 up
hcitool scan
```

9 Murata i.MX InterConnect V1/V2 Adapter Boards

To ensure correct functioning of Murata Wi-Fi/BT EVK, it is of key importance to check the adapter configuration based upon jumper settings and short pads open/closed. In configuring optional VIO voltage of 1.8V for i.MX 6UL(L) EVK or i.MX 6SoloX platforms, special attention must be spent on following all the necessary steps.

9.1 Murata i.MX InterConnect V1 Adapter

9.1.1 V1 Adapter 3.3V VIO Configuration: Default

Refer to **Figure 13** for default configuration on V1 Adapter board. VIO jumper should be in **RED** position – set for VBAT_SDIO which is approximately 3.3V (i.e. VIO = VBAT). Note that there are **no closed** (soldered) short-pads on V1 Adapter from top view. Of course, you will see the soldered connections for WL_HOST_WAKE, WL_REG_ON, and BT_REG_ON. Two figures are presented for the bottom side of the Murata i.MX InterConnect V1 Adapter. This is done to give unobstructed views of all the short pads. Refer to **Figure 14: Murata i.MX InterConnect V1 Adapter – Bottom #1** and **Figure 15: Murata i.MX InterConnect V1 Adapter – Bottom** #2 for default configuration on V1 Adapter Board. Compare your adapter with the short pads (open versus closed). There should be a one-to-one mapping. The short pad selections on bottom of adapter board connect Bluetooth UART through to SD_DAT4..7 pins (TP109, TP122, TP116, and TP130 closed/soldered). TP133 short pad is closed/soldered to connect VBAT_SDIO (voltage supply from SD VDD Pin #4) to VBAT_IN which powers the Murata Wi-Fi/BT EVB. The other power supply option is to use an external power supply: short TP134 (with TP133 open) and connect external supply to TP131 and TP132 (marked “EXT VBAT 4-5V” and “GND” on silkscreen – see **Figure 13**. The V1 Adapter is pre-wired to SD Card Extender. This is done to provide “plug ‘n play” interoperability with NXP i.MX 6SoloX SABRE-SD and i.MX 6SoloLite EVK. The connected signals are BT_REG_ON (yellow), WL_REG_ON (orange), and WL_HOST_WAKE (blue). This allows direct mapping to i.MX6 GPIO’s to these control signals. For additional specific information on default configuration, refer to the Hardware User Manual.

Figure 13: Murata i.MX InterConnect V1 Adapter – Top

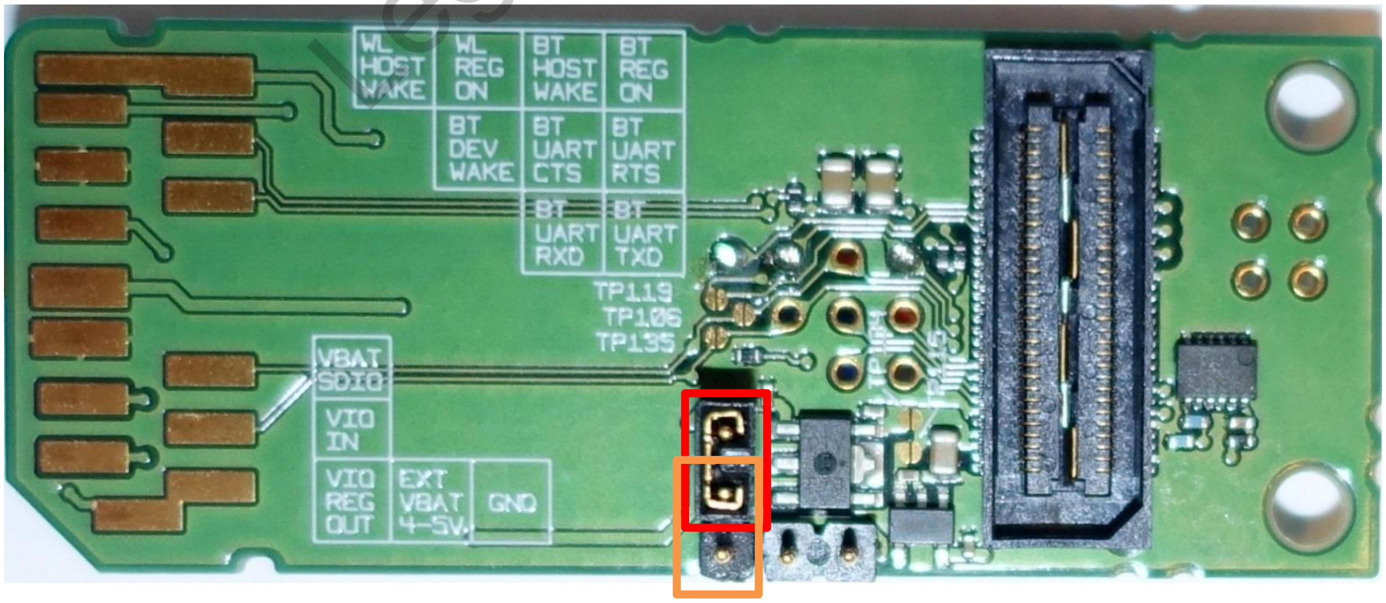


Figure 14: Murata i.MX InterConnect V1 Adapter – Bottom #1

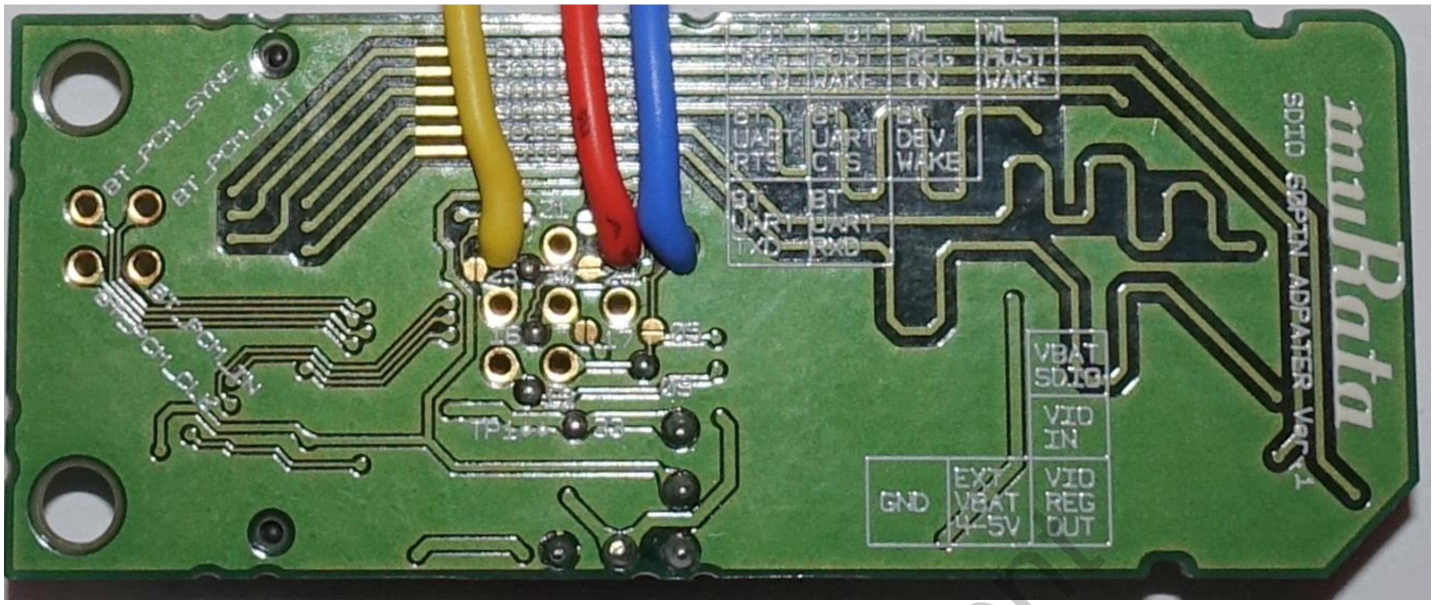
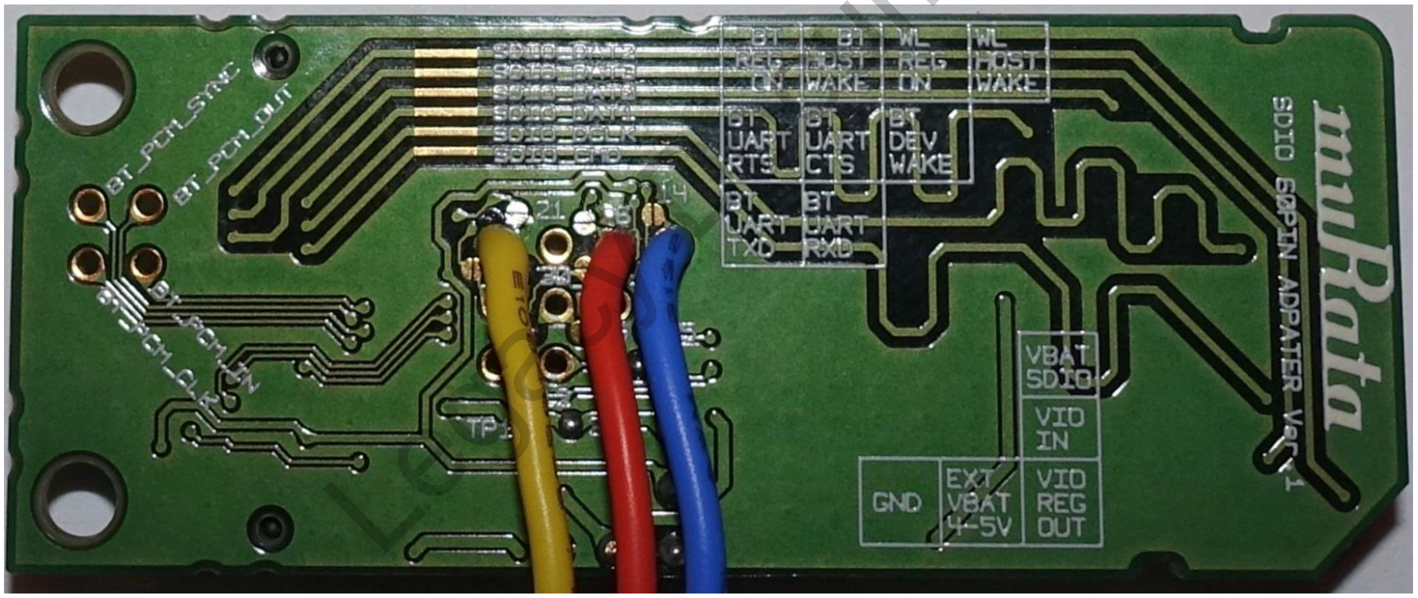


Figure 15: Murata i.MX InterConnect V1 Adapter – Bottom #2

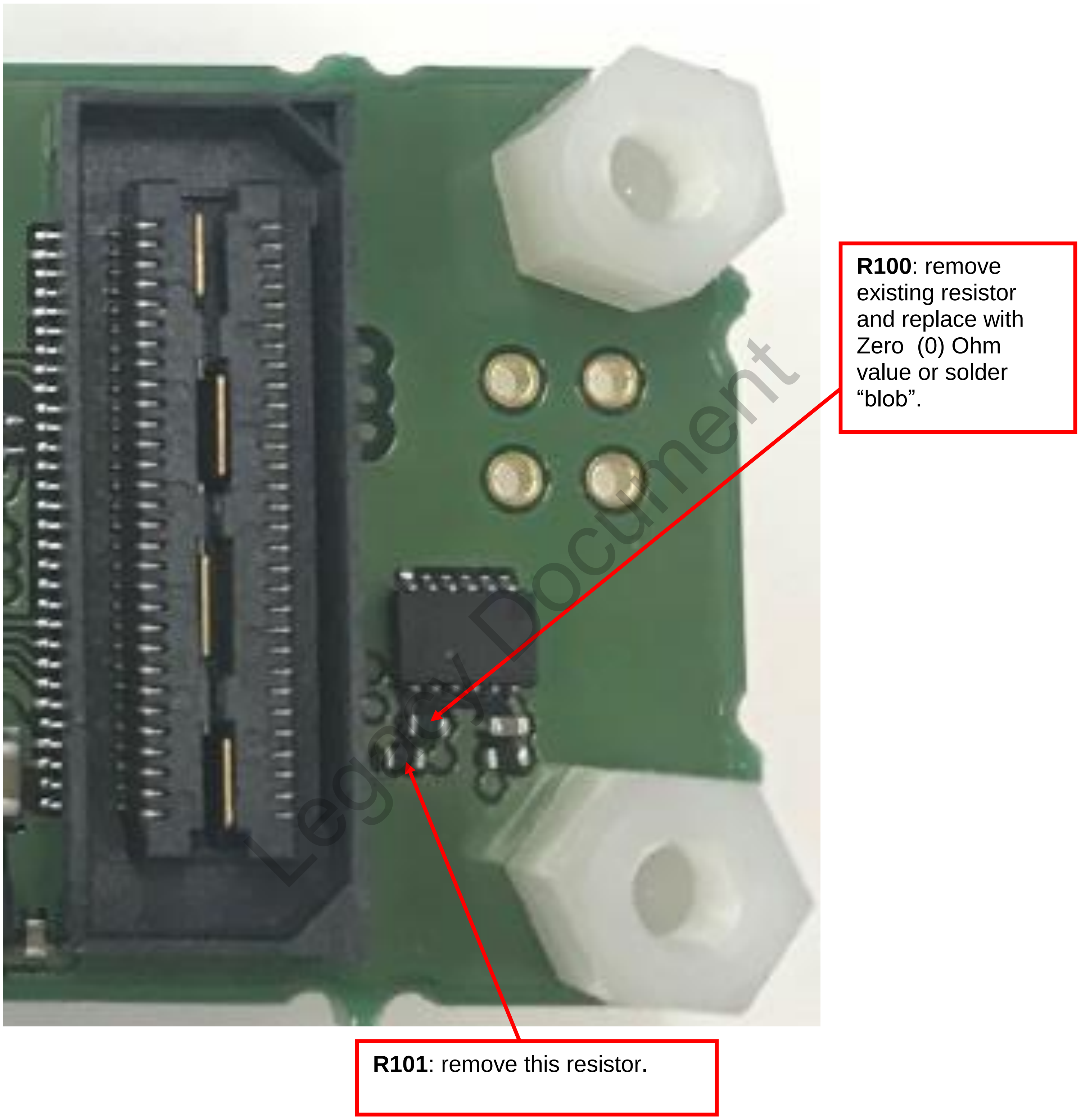


9.1.2 V1 Adapter 1.8V VIO Configuration: Necessary Changes

To configure the V1 Adapter for 1.8V VIO operation (from default 3.3V configuration), perform the following steps:

- Solder (close) TP115 – refer to **Figure 13** for location of test point.
- Move VIO jumper to **ORANGE** position from default **RED** position – refer to **Figure 13**.
- Referring to
- **Figure** 16, remove R101 and change R100 value to zero (0) Ohms or short it with solder "blob".

Figure 16: Murata i.MX InterConnect V1 Adapter – SLOW CLK Close-Up


R100: remove existing resistor and replace with Zero (0) Ohm value or solder "blob".
R101: remove this resistor.

9.2 Murata i.MX InterConnect V2 Adapter

9.2.1 V2 Adapter 3.3V VIO Configuration: Default

To ensure correct functioning of Murata Wi-Fi/BT EVK, it is of key importance to check the adapter configuration based upon jumper settings and short pads open/closed. V2 Adapter is much simpler than V1. However, it is still important to check all connections to make sure they match defaults. Refer to **Figure 17**: **Murata i.MX InterConnect Adapter V2 Adapter – Top**. VIO jumper should be in **RED** position – set for VBAT_SDIO which is approximately 3.3V (i.e. VIO = VBAT). Only two short pads are close on top: TP14 and TP13. TP14 short pad is closed/soldered to connect VBAT_SDIO (voltage supply from SD VDD Pin #4) to VBAT_IN which powers the Murata Wi-Fi/BT EVB. The other power supply option is to use an external power supply: short TP15 (with TP14 open) and connect external supply to TP11 and TP12 (marked "Ext/VBAT4-5V" and "GND" on silkscreen - see **Figure 17**). TP13 connects the BT_REG_ON control signal.

Figure 17: Murata i.MX InterConnect Adapter V2 Adapter – Top

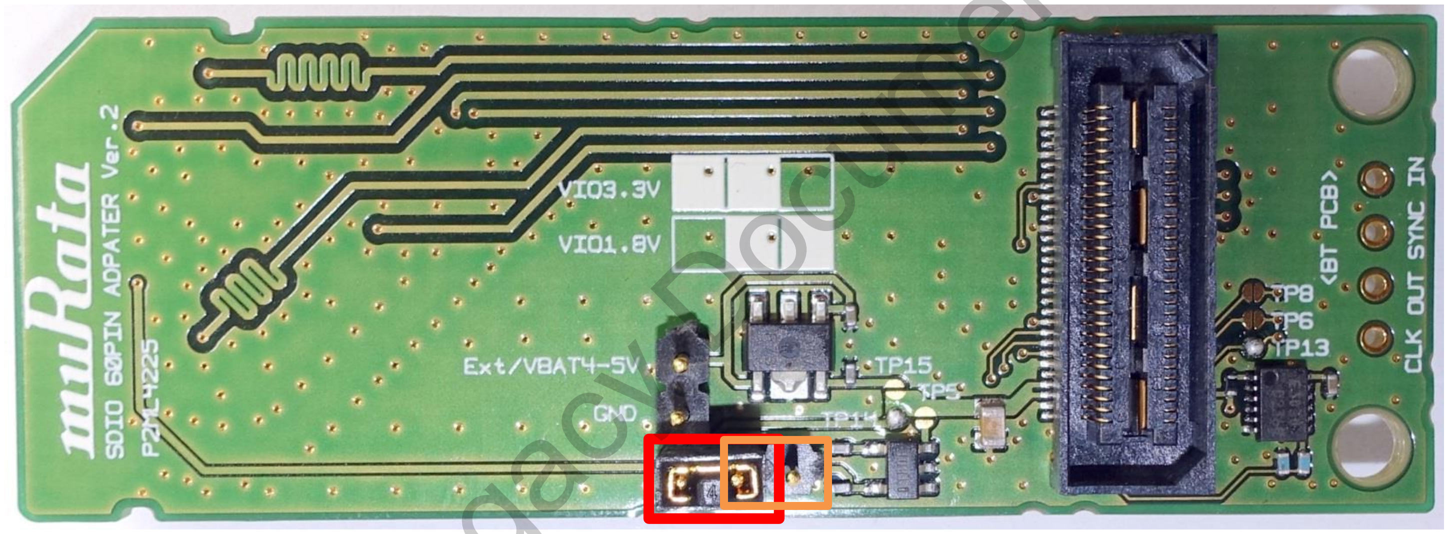
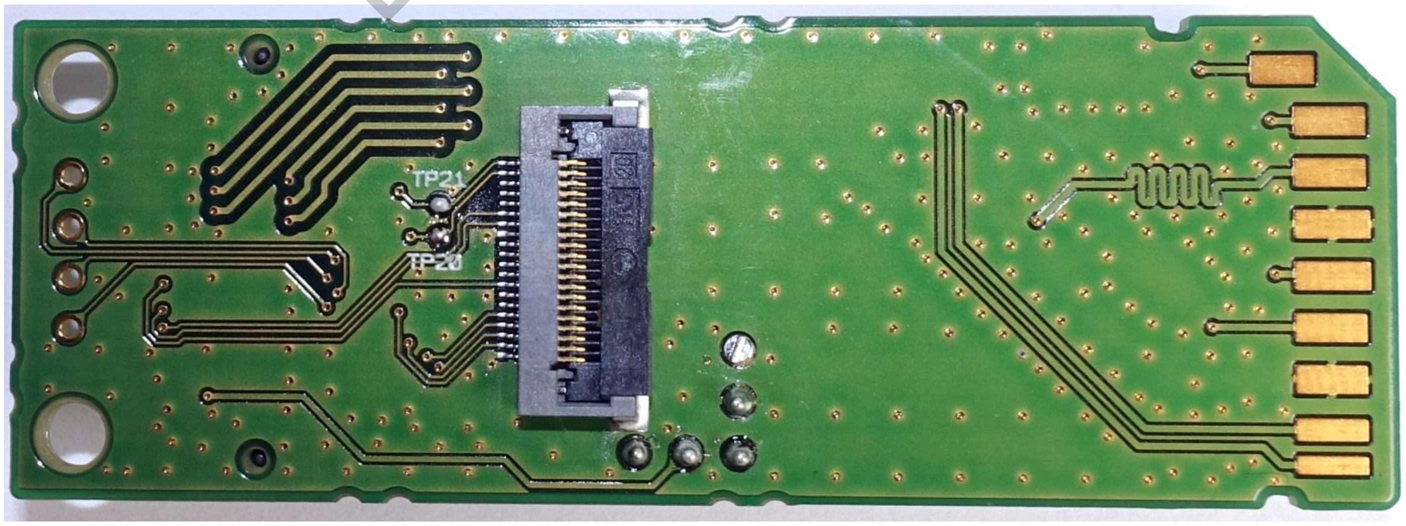


Figure 18: Murata i.MX InterConnect Adapter V2 Adapter - Bottom



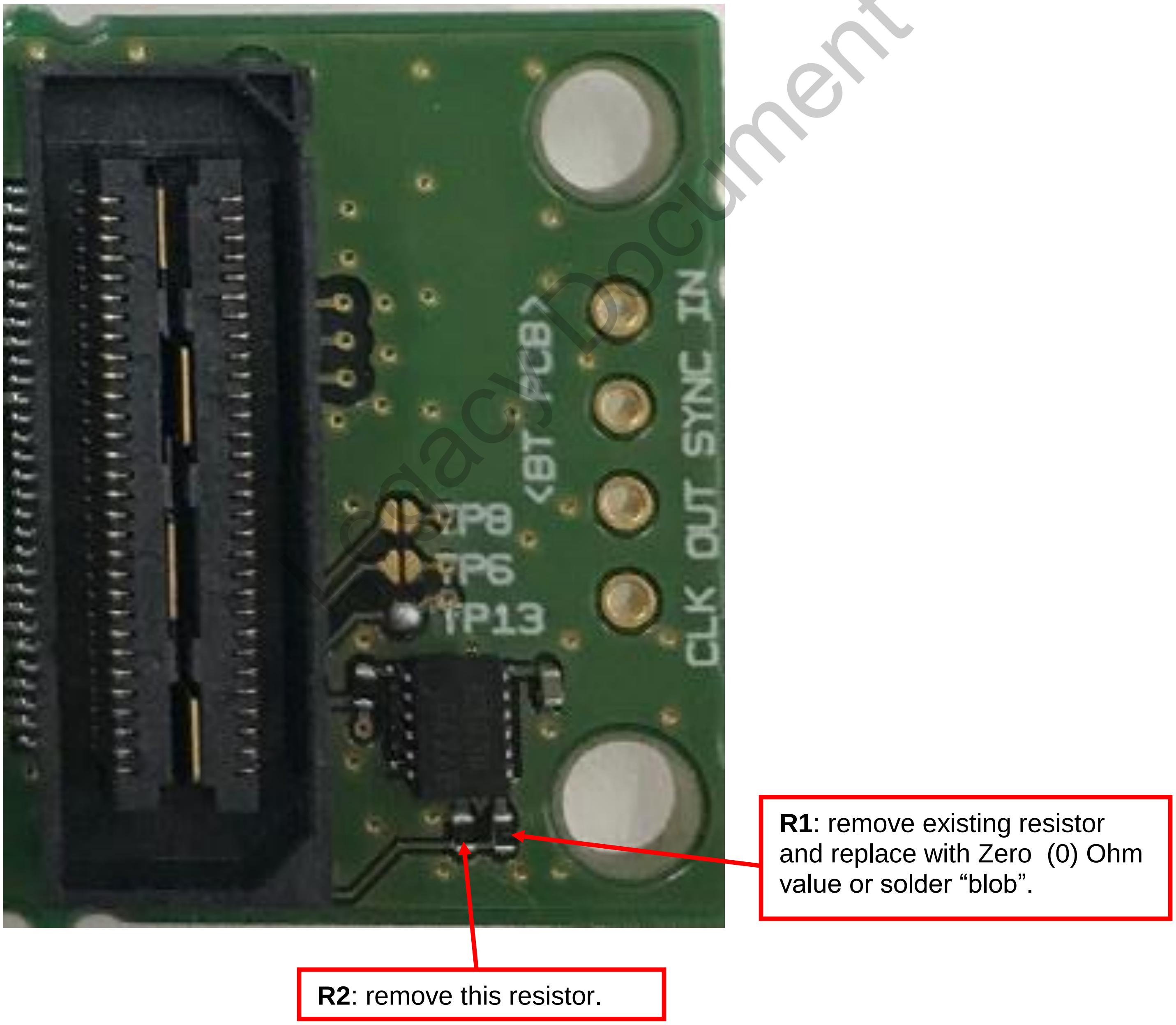
Refer to **Figure 18**: **Murata i.MX InterConnect Adapter V2 Adapter - Bottom**. Both TP20 and TP21 short pads are closed to connect WL_REG_ON and WL_HOST_WAKE respectively. For additional specific information on default configuration, refer to the Hardware User Manual.

9.2.2 V2 Adapter 1.8V VIO Configuration: Necessary Changes

To configure the V2 Adapter for 1.8V VIO operation (from default 3.3V configuration), perform the following steps:

- Solder (close) TP5 – refer to **Figure 17** for location of test point.
- Move VIO jumper to **ORANGE** position from default **RED** position – refer to **Figure 17**.
- Referring to **Figure 19**, remove R2 and change R1 value to zero (0) Ohms or short it with solder “blob”.

Figure 19: Murata i.MX InterConnect V2 Adapter – SLOW CLK Close-Up



<BT PCB>
CLK OUT SYNC IN
TP8
TP6
TP13
R1: remove existing resistor and replace with Zero (0) Ohm value or solder “blob”.
R2: remove this resistor.

10 Technical Support Contact

Table 12 below lists all the support resources available for the NXP/Cypress/Murata i.MX Wi-Fi/BT solution. There is a dedicated Murata website (no login required) in addition to a dedicated imxfaq@murata.com email alias. All website/email addresses are hyperlinked in the "Support Site" column below.

Table 12: List of Support Resources

Support Site	Notes
Murata i.MX Landing Page	**No** login credentials required: http://wireless.murata.com/imx. This is an excellent starting point to understand all hardware/software configurations supported. All supporting Murata i.MX Wireless documentation is provided here. Links to other Murata wireless pages which provide specifics on various module solutions.
NXP i.MX Community Forum	Go to: https://community.nxp.com/community/imx. Login credentials required. Numerous forum postings on interfacing Murata wireless solutions on NXP i.MX platforms. NXP and Murata teams support this forum.
Cypress Linux Support Forum	Go to: https://community.cypress.com/community/linux. Login credentials required. Includes support forum, chipset datasheets, application notes, etc. Cypress and Murata Teams support this forum.
Murata i.MX FAQ Email	i.MX FAQ email: imxfaq@murata.com. Supported by Murata Team. Typically used to support issues accessing support sites, this email address can also be used to escalate issues not answered on any of the support forums.

11 Additional Useful Links

In addition to **Table 12** listings of support resources, **Table 13** provides some useful links.

Table 13: Additional Useful Links

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