UM11483

Getting Started with NXP-based Wireless Modules on i.MX 8M Quad EVK Running Linux OS

Rev. 19.0 — 23 September 2025 **User manual**

Document information

Information	Content
Keywords	i.MX 8M Quad Evaluation Kit (EVK), 88W8801-based wireless module, 88W8987-based wireless module, 88W8997-based wireless module, 88W9098-based wireless module, IW416-based wireless module, IW612-based wireless module, IW610-based module, AW693-based wireless module
Abstract	Details the bring-up of Wi-Fi and Bluetooth on NXP-based wireless modules connected with i.MX 8M Quad EVK running Linux OS.

1 About this document

This document details the enabling of wireless solutions on i.MX 8M Quad evaluation kit. The i.MX 8M Quad EVK is powered by Linux and the NXP Linux drivers are used for NXP-based wireless modules. The manual covers:

- The bring-up of i.MX 8M Quad EVK
- The configurations for the BSP image
- The hardware connection with NXP-based wireless modules
- The bring up of Wi-Fi, Bluetooth, and 802.15.4

This document specifies the hardware interconnection and software support for the i.MX 8M Quad evaluation kit and NXP-based wireless modules to enable Wi-Fi/Bluetooth functionality.

Note: *AzureWave modules AW-CM358-uSD and AW-CM276-uSD only support Wi-Fi with the i.MX 8M Quad EVK.*

Once you have enabled the Wi-Fi and/or Bluetooth interfaces, for details the Wi-Fi and Bluetooth features and configurations for i.MX 8M Quad EVK, see ref.[16].

Note: *This document does not provide a detailed description of the i.MX 8M Quad BSP nor how to generate an image and rootfs as these topics are covered in ref.[15].*

2 i.MX 8M Quad evaluation kit (EVK)

NXP i.MX 8M Quad Evaluation Kit (EVK) provides a platform for rapid evaluation of the i.MX 8MQuad, i.MX 8MDual and i.MX 8MQuadLite application processors, utilizing dual to quad Arm Cortex-A53s and single Cortex-M4 cores.

The EVK includes the hardware design files, tools and board support packages (BSPs) for Linux where:

- The i.MX 8M Quad Linux Board Support Package (BSP) supports the Linux Operating System (OS) on the i.MX 8M Quad application processors. The purpose of this software package is to enable Linux support on the i.MX 8M Quad family of Integrated Circuits (ICs) and their associated platforms. It provides the necessary software to interface the standard open-source Linux kernel to the i.MX 8M Quad hardware.
- The i.MX 8M Quad BSP is based on the latest long-term stable (LTS) version of the Linux kernel, which is enhanced with the features provided by NXP and can be accommodate customized Linux kernel configurations.

2.1 i.MX 8M Quad evaluation board

The i.MX 8M Quad evaluation board is based on the NXP i.MX 8M Quad application processor. The i.MX 8M Quad processor features an advanced implementation of the Quad Arm Cortex-A53+ ARM Cortex-M4 core which operates at speeds up to 1.5 GHz and 266 MHz respectively. The i.MX 8M Quad features an integrated power management module that reduces the complexity of an external power supply and simplifies the power sequencing. Each processor provides a 32-bit LVDDR3L/DDR4/LPDDR4 memory interface and other interfaces to connect peripherals such as HDMI, LCD, Wi-Fi, Bluetooth, and camera sensors.

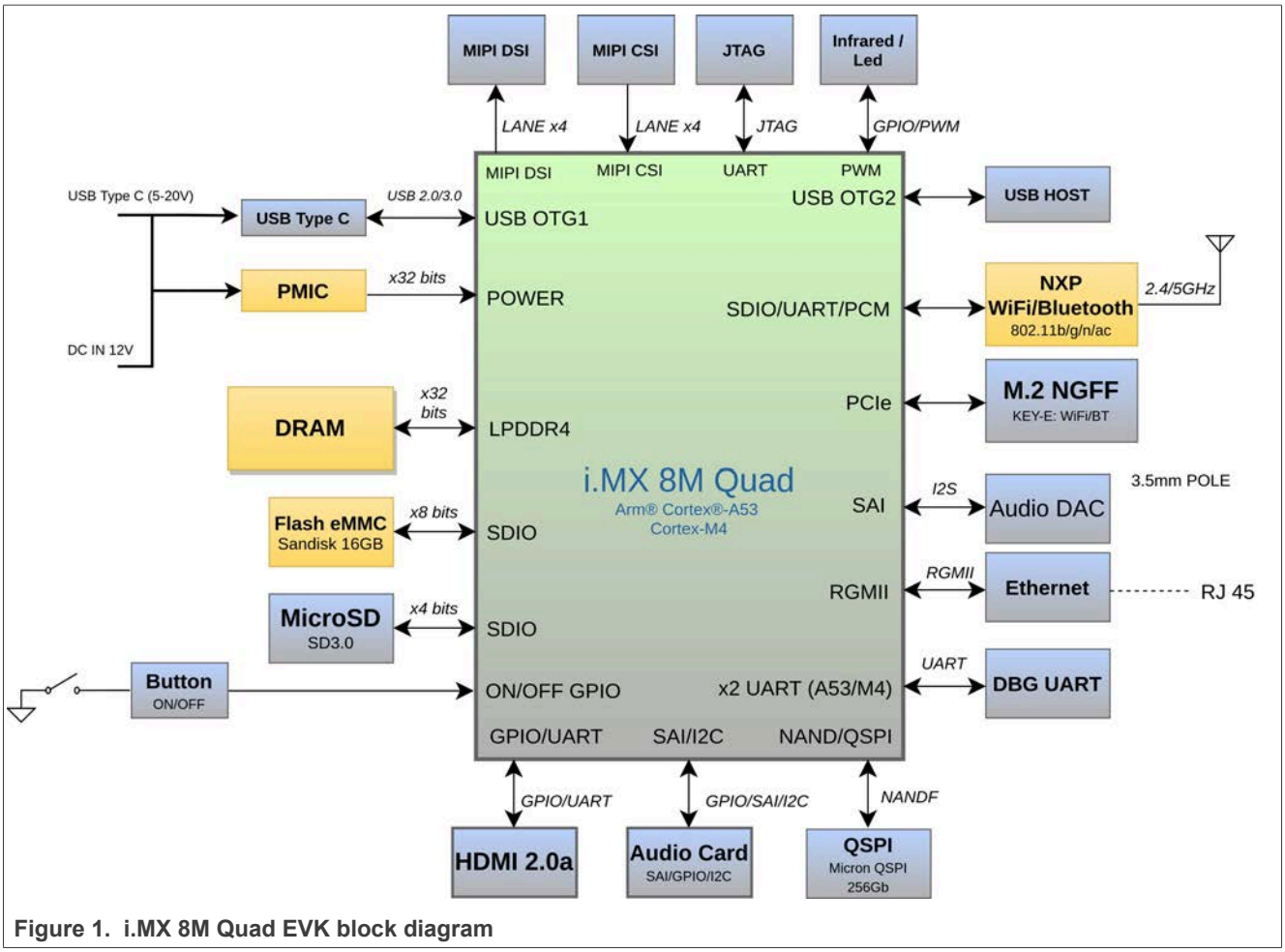

Figure 1. i.MX 8M Quad EVK block diagram

For more information about the application processor, see the data sheet and reference manual on www.nxp.com.

Table 1 lists the features of i.MX 8M Quad EVK.

Table 1. Features of i.MX 8M Quad

Features	Features (continued)
i.MX 8M Quad applications processor with five cores (4×Arm® Cortex® -A53 and 1× Cortex-M4)	M.2 connector for Wi-Fi/Bluetooth (PCIe, USB, UART, I2C and I2S)
3 GB, 32-bit LPDDR4 with 1.6 GHz clock	USB3.0 Type-A connector
eMMC 5.0, 16 GB	HDMI2.0a Type-A connector
32 MB Octal SPI NOR flash	1 Gbit/s Ethernet
Micro SD card connector	Mini-SAS MIPI-DSI connector
USB3.0 Type-C connector with PD support	2x mini-SAS MIPI-CSI connectors for Camera
USB to serial converter for debug	Infrared receiver LEDs for power indication and general-purpose use
3.5 mm audio jack for amplified speakers	JTAG 10-pin connector

2.2 i.MX 8M Quad evaluation board interfaces

Figure 2 shows the front view of i.MX 8M Quad evaluation board with pointers to the interfaces.

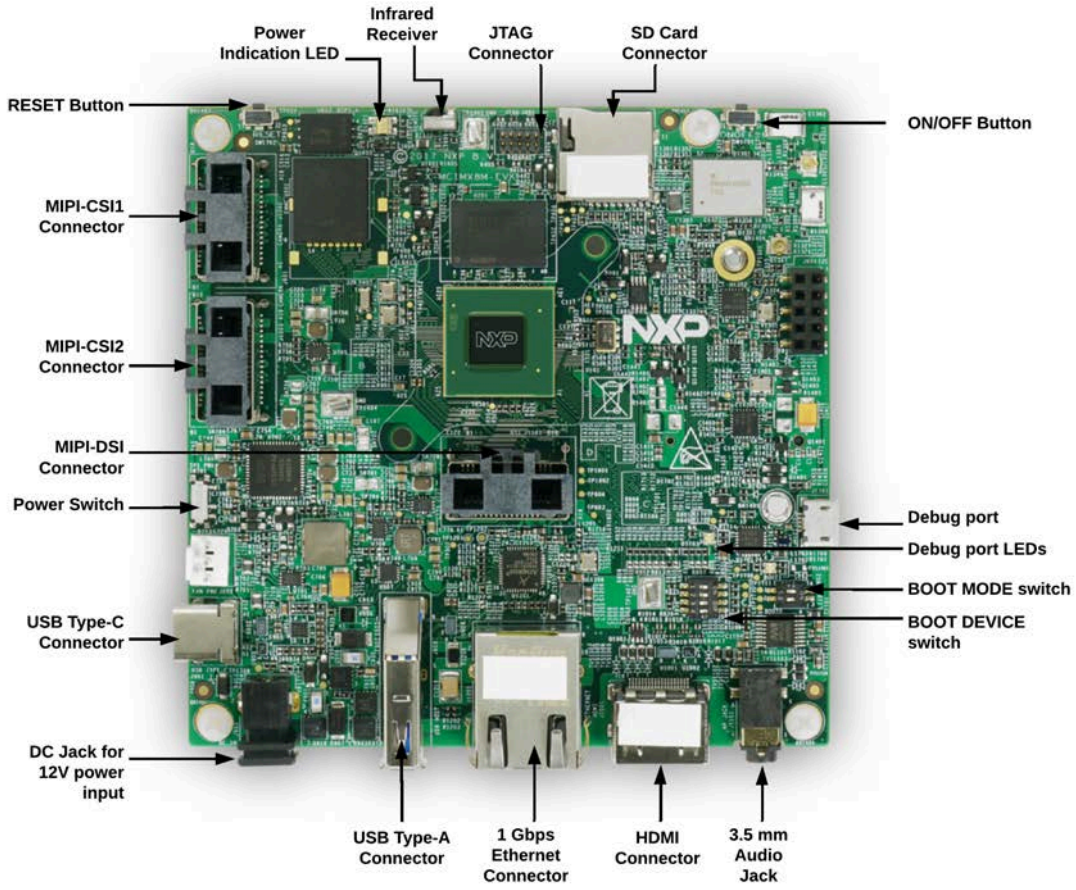

Figure 2. i.MX 8M Quad evaluation board interfaces - Front view

Figure 3 shows the back view of i.MX 8M Quad evaluation board with pointers to the interfaces.

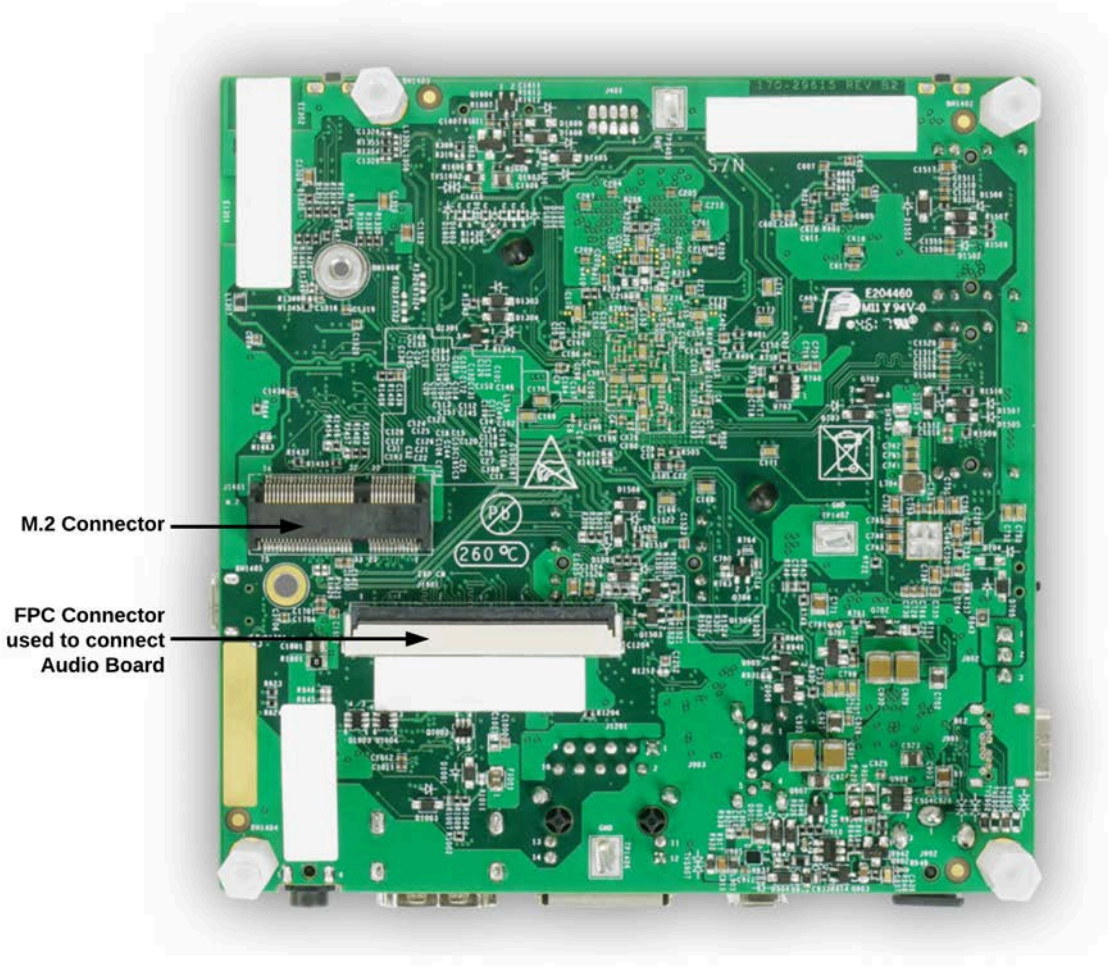

Figure 3. i.MX 8M Quad evaluation board interfaces - Back view

2.3 i.MX 8M Quad switch settings

Figure 4 shows the two switches on i.MX 8M Quad evaluation board. The *Boot Device Switch* is used to select the boot drive while the *Boot Mode Switch* is used to set the boot mode.

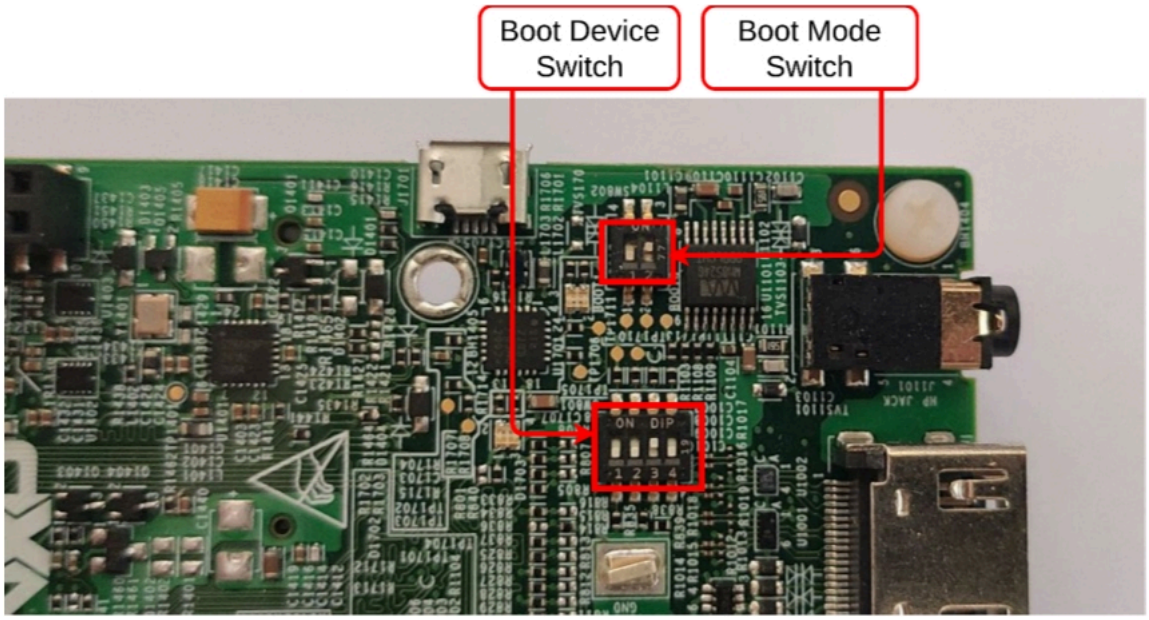

Figure 4. Boot device switch and boot mode switch on i.MX 8M Quad evaluation board

Table 2 shows the settings of the boot mode switch.

Table 2. Boot mode switch settings

Boot mode	D1	D2
Serial downloader	0	1
Internal boot	1	0

Table 3 shows the settings of the boot device switch to boot from eMMC.

Table 3. Switch settings to boot from eMMC

Configuration	D1	D2	D3	D4
Boot from eMMC	OFF	OFF	ON	OFF

3 NXP-based wireless modules

This document revision addresses the following NXP-based wireless modules.

- **88W8801**: see ref.[1] and ref.[17].
- **88W8987**: see ref.[2] and ref.[18].
- **88W8997**: see ref.[19], ref.[14], and ref.[19].
- **88W9098**: see ref.[3] and ref.[20].
- **IW416**: see ref.[5] and ref.[21].
- **IW610**: see ref.[6] and ref.[22].
- **IW612**: see ref.[7] and ref.[23].

3.1 Interface with i.MX 8M Quad application processor

Figure 5 shows the high-level block diagram of i.MX 8M Quad application processor with the Wi-Fi (SDIO/PCIe) and Bluetooth (UART) hardware interfaces used to communicate with NXP-based wireless module.

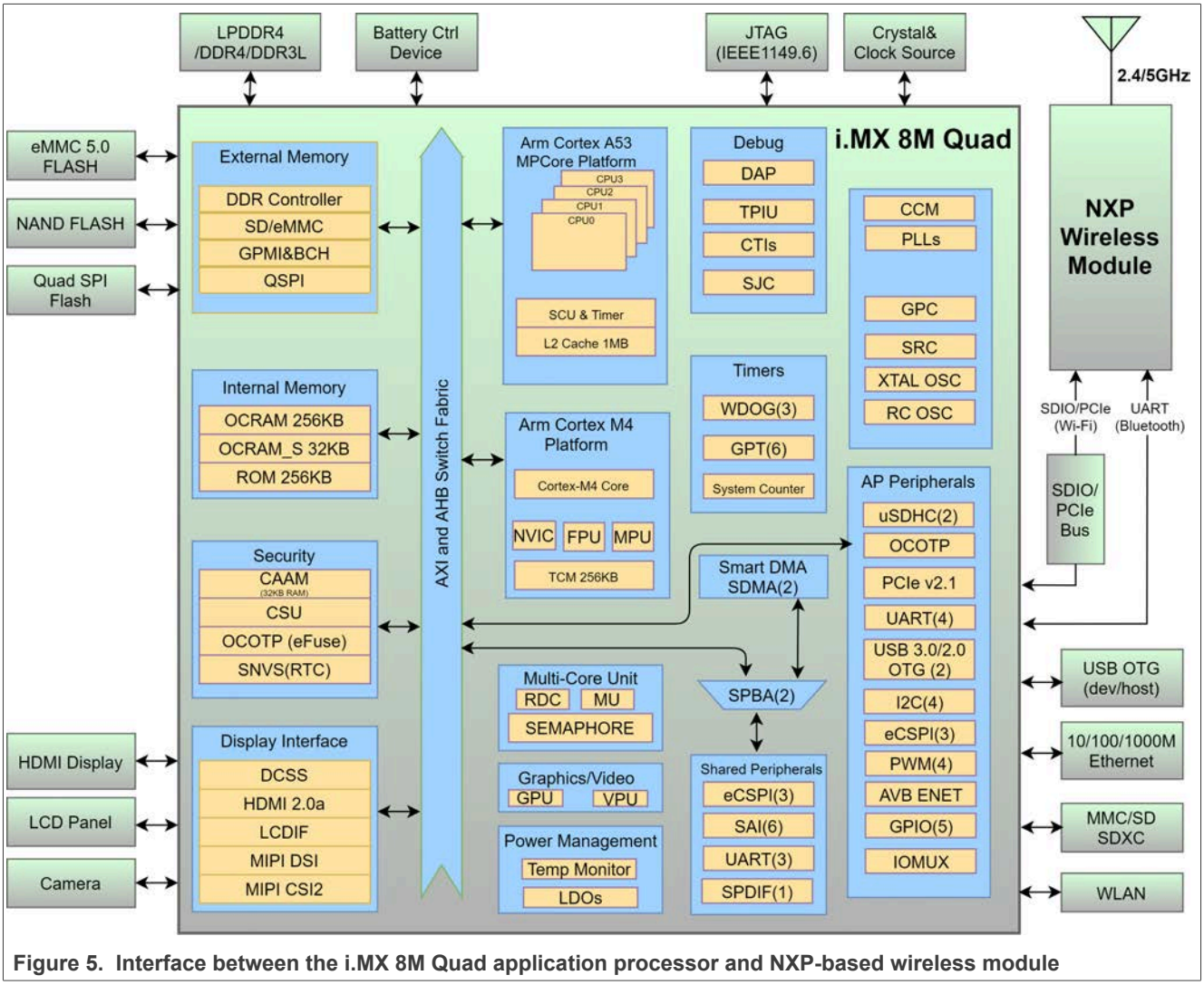

Figure 5. Interface between the i.MX 8M Quad application processor and NXP-based wireless module

3.2 Wi-Fi MM driver architecture

This section describes the architecture of the Wi-Fi Linux MM driver which supports multiple bus interfaces and Wi-Fi devices. The architecture is divided into two modules:

- **MOAL module:** OS-dependent module that includes:
  - The upper interface to the kernel/protocol stack.
  - The lower interface that registers the Wi-Fi driver to the bus driver (SDIO/PCIE) when loaded, so an SDIO/PCIE device can be detected.
  - One SHIM layer to MLAN module.
- **MLAN module:** OS-independent module that includes most of the driver handling and interface to the firmware. It also adds applicable interface operations based on the card type given by MOAL module. One option is available to use MLAN module "as is" to reduce the porting work.

Figure 6 shows the Wi-Fi MM driver architecture.

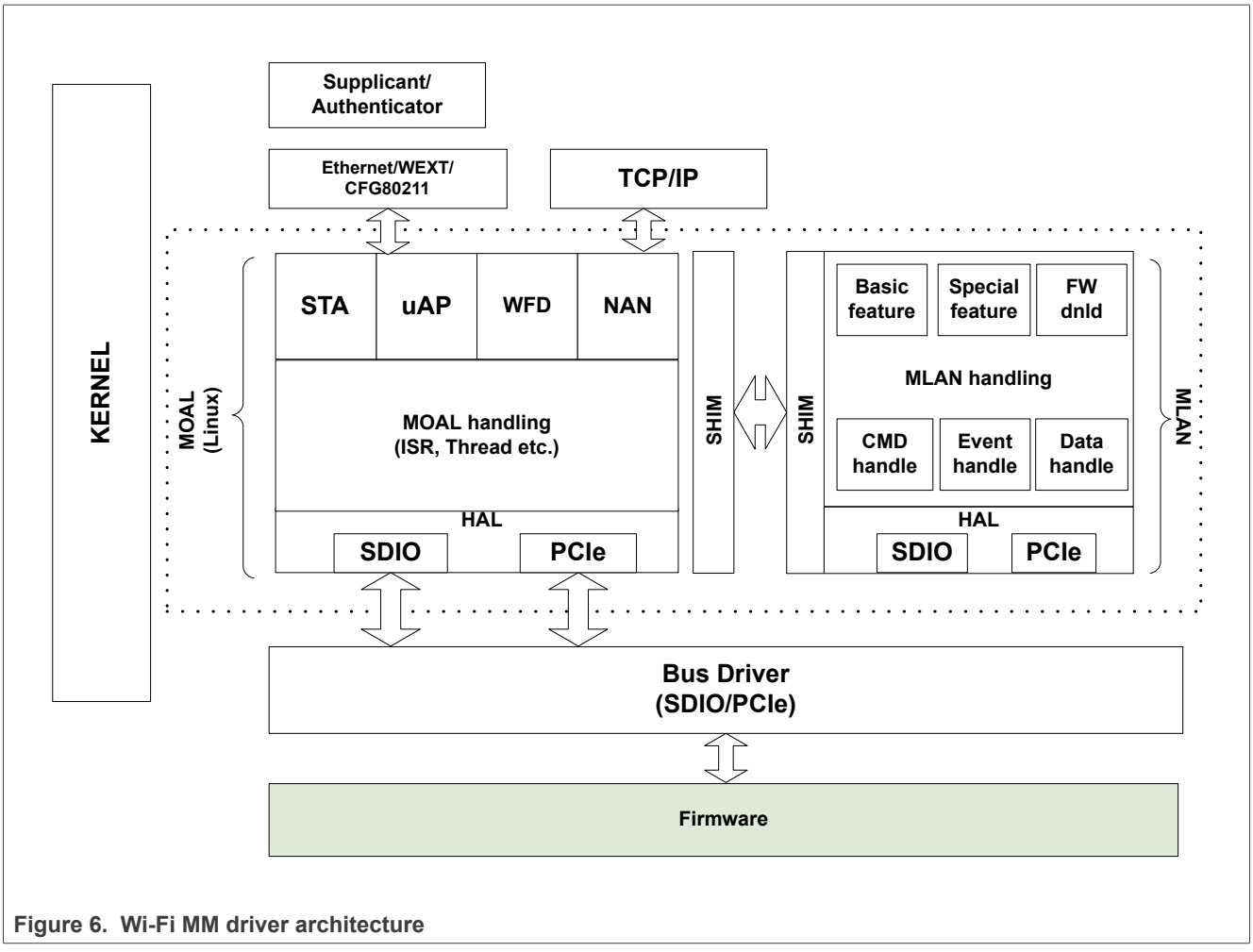

Figure 6. Wi-Fi MM driver architecture

3.2.1 MOAL module

Upper interface to kernel/protocol stack:

- Ethernet handling: standard ethernet driver module in Linux with similar functionality as an Ethernet NIC card to the kernel. The driver module handles packet transmission and reception as well as the control.
- 802.11 extensions (WE) handling: interface to the wireless extension stack provided by Linux kernel. The interface includes 802.11 specific features like scanning, association, 802.11d, 802.11h, and WMM.
- Cfg80211 handling: interface to cfg80211 stack provided by Linux kernel. The interface is a replacement for WE handling, although it can also work alongside WE handling. Multiple network interfaces are supported. The possible BSS types of network interface are STA, uAP, WFD, and NAN. The BSS list is created during the driver initialization and can be adjusted according to the device capability.

Lower interface to the bus driver:

- Bus driver interfacing: component that provides an interfacing layer between the MOAL module and the low-level bus driver module. The component registers the Wi-Fi driver to the (SDIO/PCIE) bus driver and call the APIs provided by the bus driver to access the registers and data ports.
- Independent operation of SDIO/PCIE: When a device is detected, the driver adds the applicable interface operations for the device interface.

SHIM layer to MLAN: used to convert MLAN APIs to OS-specific APIs.

3.2.2 MLAN module

The MLAN module is used for firmware downloading, command handling, data handling, event handlings, SME handling, power-save handling, and mux/demux handling, and more.

Note: *The driver or firmware API version up to 18 is supported in MLAN.*

The data sent to or received from the firmware is little-endian based. The driver is configured as little-endian by default.

3.3 Wi-Fi layer interfaces

The wireless module requires a kernel driver loaded on the i.MX 8M Quad host system and a firmware running on NXP-based wireless modules. When the interface bus driver detects the NXP-based wireless module, the MLAN module downloads the firmware binary via the interface adapter. NXP Wi-Fi driver is loaded between the bus driver and the network stack from the cfg80211 subsystem in the kernel. NXP kernel driver includes a set of controls and configurations to communicate with the user space through one of the following interfaces:

- Input/output control (IOCTL)
- Wireless Extension (Wext)
- CFG80211

The IOCTL provides a path to the user space applications *iwconfig* and *iwpriv* whereas cfg80211 interface provides a different path to the user space applications *wpa_supplicant*, *hostapd* and *iw*.

Figure 7 illustrates the Wi-Fi layer interface.

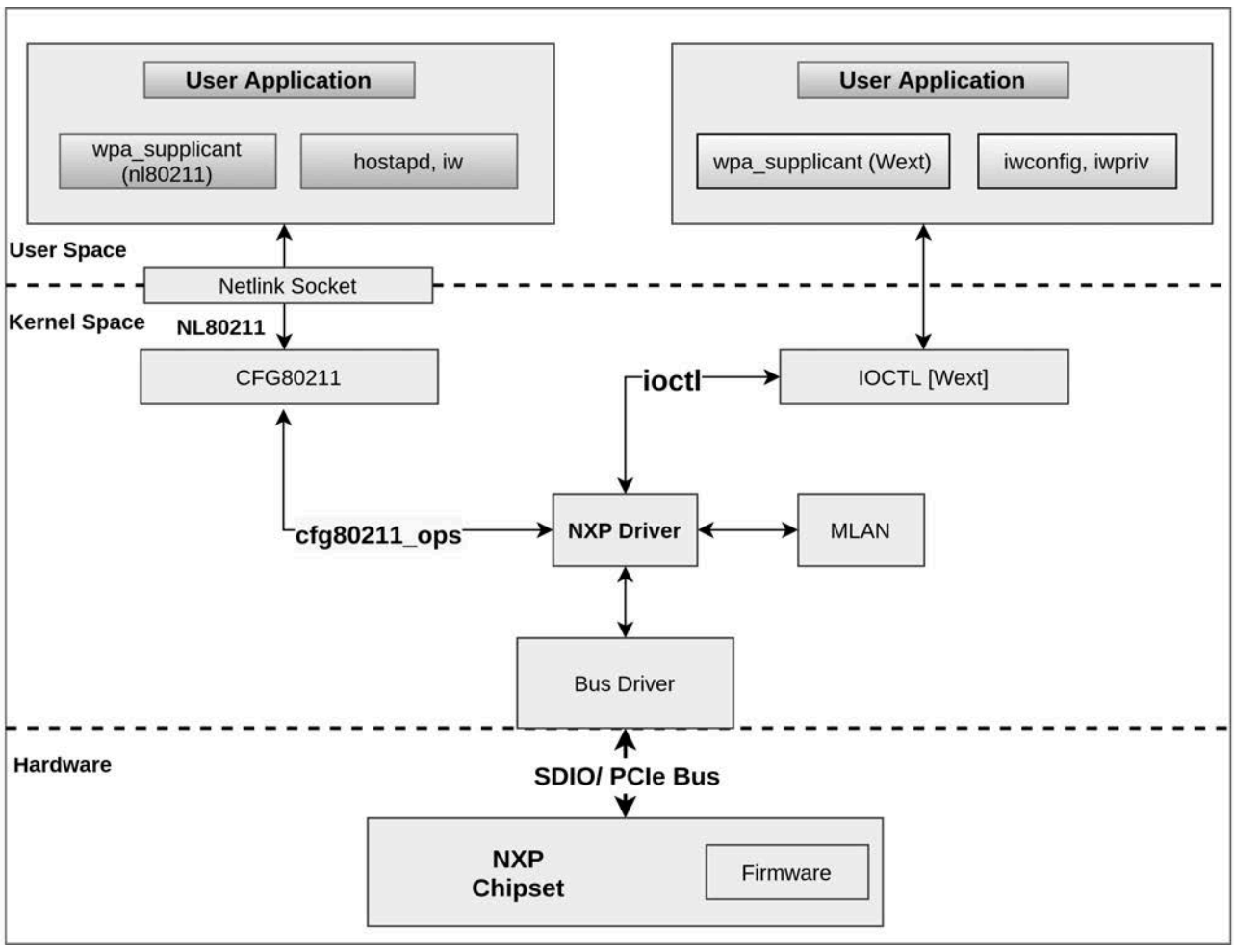

Figure 7. Wi-Fi layer interface

3.4 Bluetooth layer interfaces

Figure 8 illustrates the layers between the user applications and the NXP-based Bluetooth module. The NXP-based wireless module requires a kernel driver loaded on the i.MX 8M Quad host system and a firmware running on NXP SoC. The Wi-Fi driver loads the combo firmware. The *hci_uart* driver provides the HCI interface between the firmware and user application.

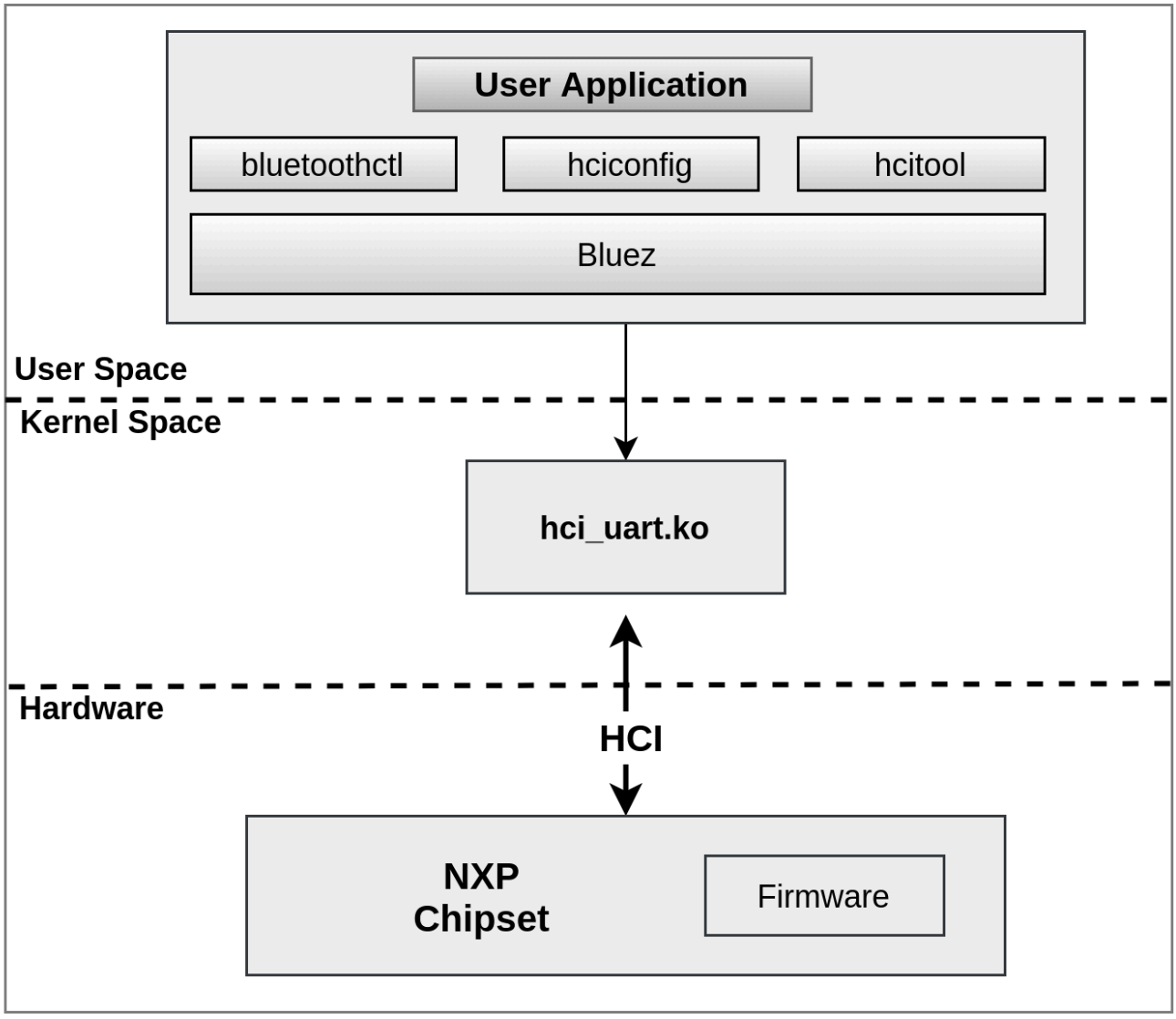

Figure 8. Bluetooth layer interface

3.5 OpenThread radio coprocessor (RCP) software architecture

Thread is an IPv6-based networking protocol designed for low-power Internet of Things devices in an IEEE802.15.4-2015 wireless mesh network. OpenThread released by Google is an open-source implementation of Thread.

The 802.15.4 subsystem of IW612 works as controller in OpenThread RCP design as illustrated in Figure 9.

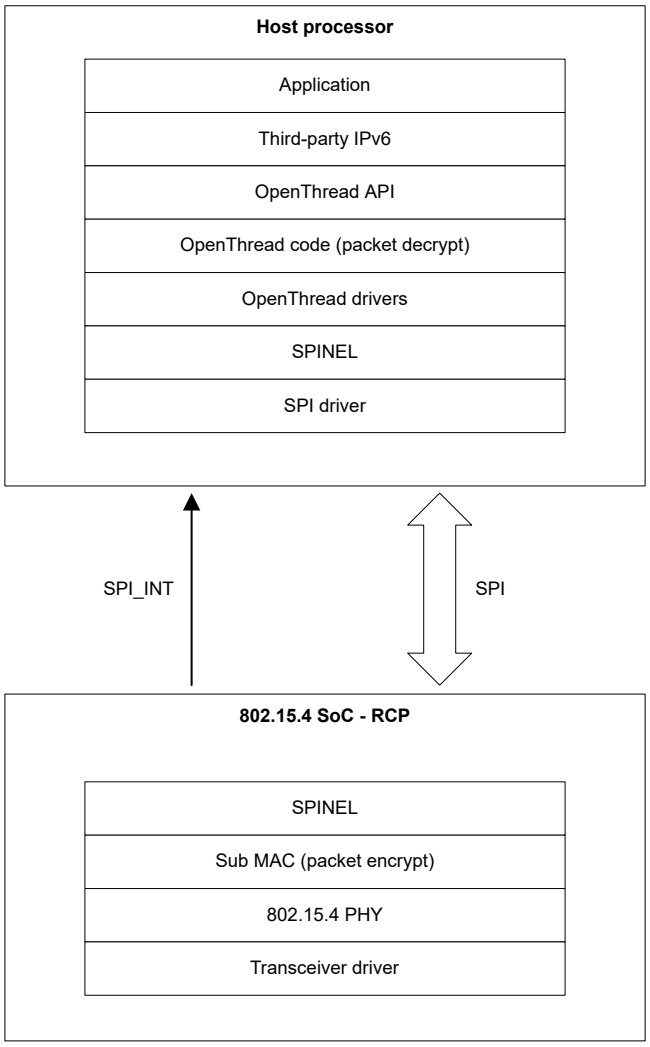

Figure 9. 802.15.4 software architecture

3.5.1 Host

The OpenThread core runs on the host processor and communicates with the controller via OpenThread Daemon (Ot-daemon) through SPI interface over the Spinel protocol (link).

Ot-daemon main features:

- Used in the radio coprocessor (RCP) design
- OpenThread POSIX build mode that runs OpenThread as a service
- UNIX socket as input and output

Clients can connect to the UNIX socket and communicate using OpenThread CLI as a protocol.

3.5.2 Controller

The controller supports:

- Spinel over SPI
- 1 MHz maximum SPI clock speed
- IEEE 802.15.4-2015 sub-MAC and PHY features as required by Thread 1.3.0
- Thread Network Leader
- TX and RX PHY PSDU

3.6 88W8987-based Azurewave AW-CM358-uSD Wi-Fi module (SDIO)

Azurewave provides a micro-SD card interface adapter (uSD-1212) with AW-CM358SM module that is compatible with the i.MX 8M Quad EVK. The AW-CM358-uSD supports Wi-Fi through a uSD device interface that conforms to the industry SDIO Full-Speed card specification and allows a host controller using the SDIO bus protocol to access the Wireless SoC device. The AW-CM358-uSD acts as the device on the SDIO bus, and features the following:

- SDIO 3.0 standard
- On-chip memory used for CIS
- Supports 1-bit SDIO and 4-bit SDIO transfer modes
- Special interrupt register for information exchange

3.6.1 Recommended antenna part

MAG.LAYERS: MSA-4008-25GC1-A2

3.6.2 Supported I/O signal level

- SDIO (3.0/2.0) supports 1.8V for I/O signal

3.6.3 Supported RF standard

Table 4. Supported RF standard

I/O voltage level	Wi-Fi	Bluetooth
AW-CM358-uSD	1x1 Wi-Fi 5 (2.4/5 GHz)	5.0

3.6.4 Supported Wi-Fi features

AW-CM358-uSD and AW-CM358MA modules share the same Wi-Fi feature set. See ref.[8].

Note: *Azurewave AW-CM358-uSD supports both Wi-Fi and Bluetooth RF standards. But as i.MX 8M Quad EVK does not include the FFC connector for Bluetooth interface, only Wi-Fi is supported for i.MX 8M Quad platform with AW-CM358-uSD module combination.*

3.6.5 Azurewave AW-CM358-uSD Wi-Fi module interface

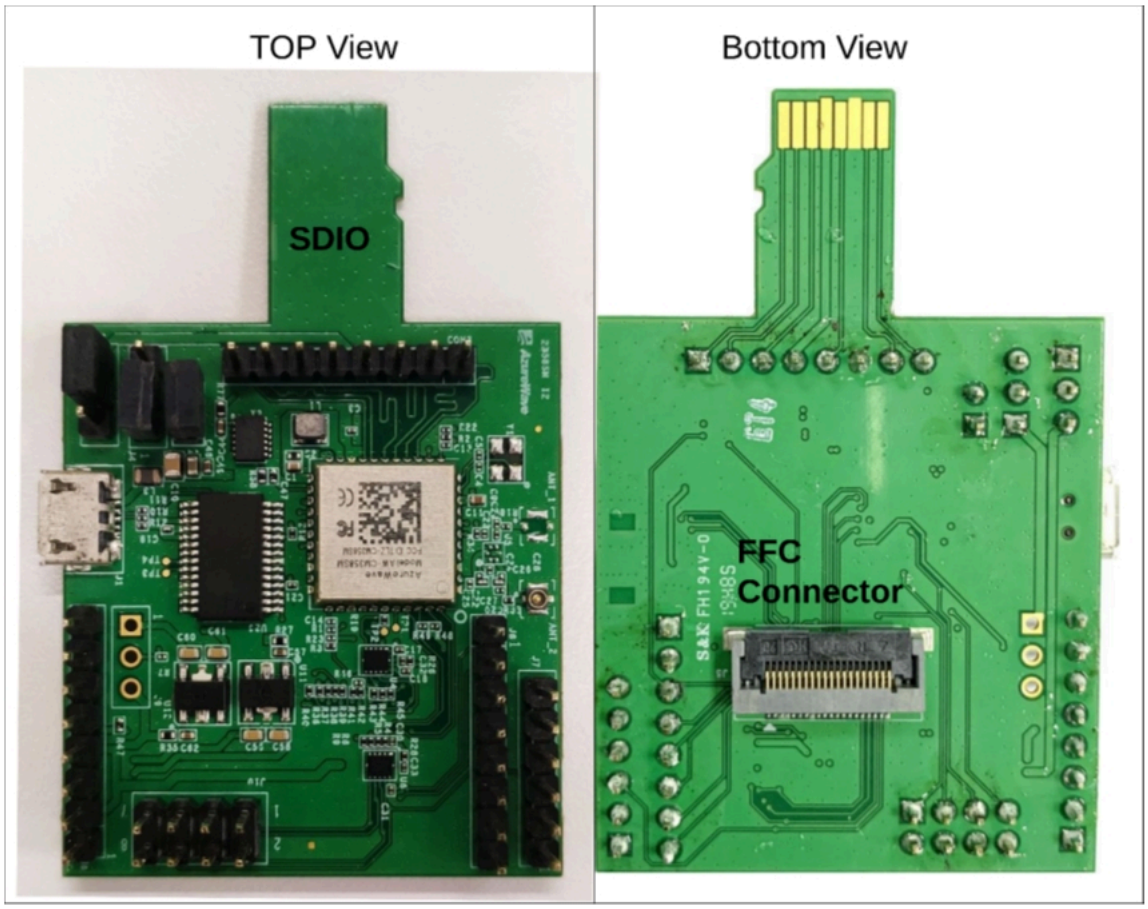

Figure 10. AzureWave AW-CM358-uSD module interface

3.6.6 Azurewave AW-CM358-uSD module jumpers and power supply

This section covers the jumper settings to configure the module with 1.8V SDIO voltage level for Wi-Fi. See Table 5 and Figure 11.

Table 5. Jumper settings for AW-CM358-uSD module

Jumper	Header pins	Description
J2	(1-2)	SDIO module power source
J4	(1-2)	1.8V SDIO voltage

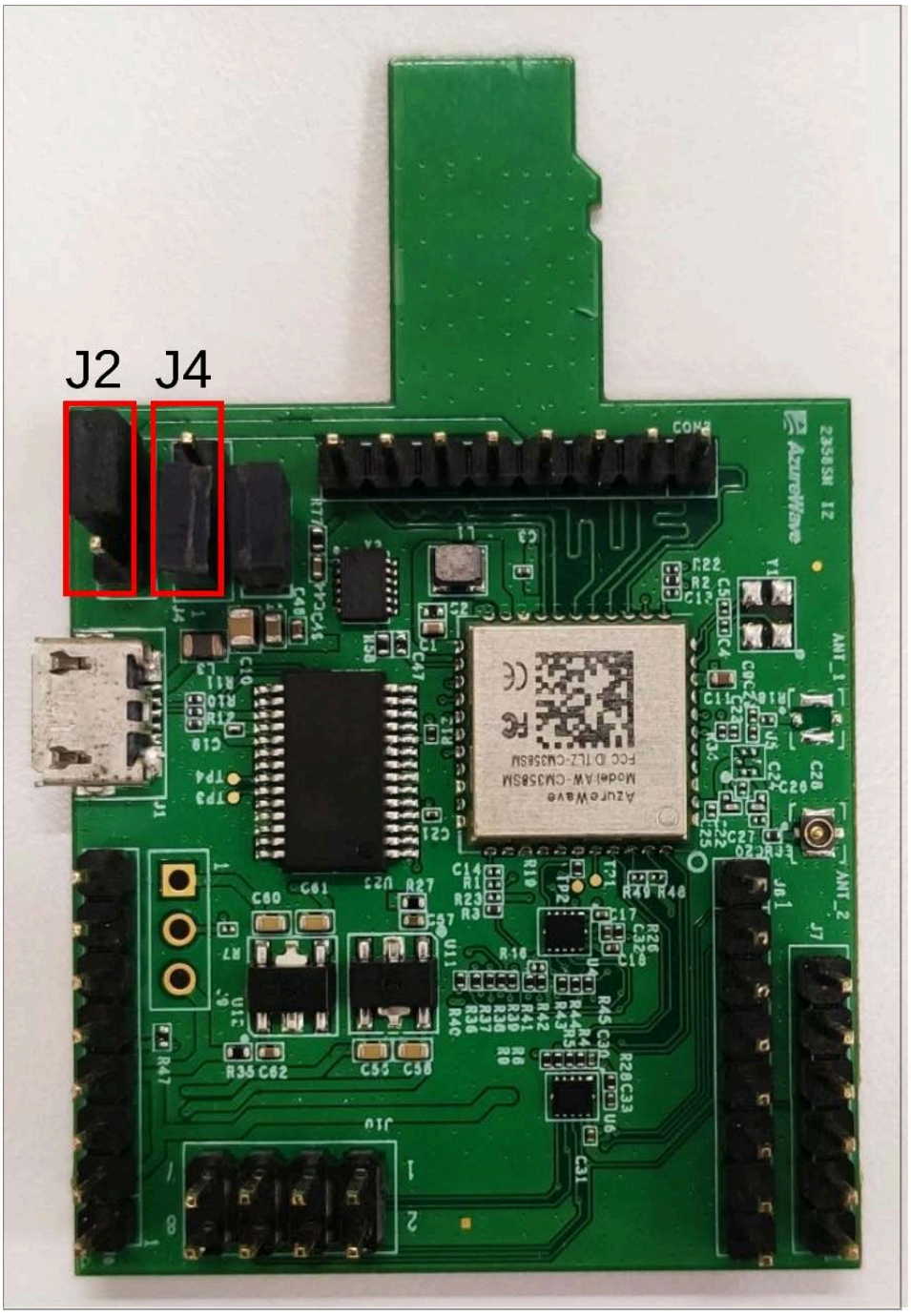

Figure 11. AzureWave AW-CM358-uSD module header positions

3.6.7 Plugging the wireless module

Plug the module into the connector slots of i.MX 8M Quad board:

- For AW-CM358MA, connect the module into the M.2 connector of the i.MX 8M Quad board and screw.
- For AW-CM358-uSD, plug the module into the SDIO card slot of the i.MX 8M Quad board.

```
[ 3632.632050] mmc1: new ultra high speed SDR104 SDIO card at address 0001
```

3.6.8 AW-CM358-uSD module setup with i.MX 8M Quad

Plug AW-CM358-uSD module into the SDIO card slot of the i.MX 8M Quad board.

```
[ 3632.632050] mmc1: new ultra high speed SDR104 SDIO card at address 0001
```

Connect the antenna.

Use a Micro USB to USB cable to connect i.MX 8M Quad EVK to the host computer running on Linux OS.

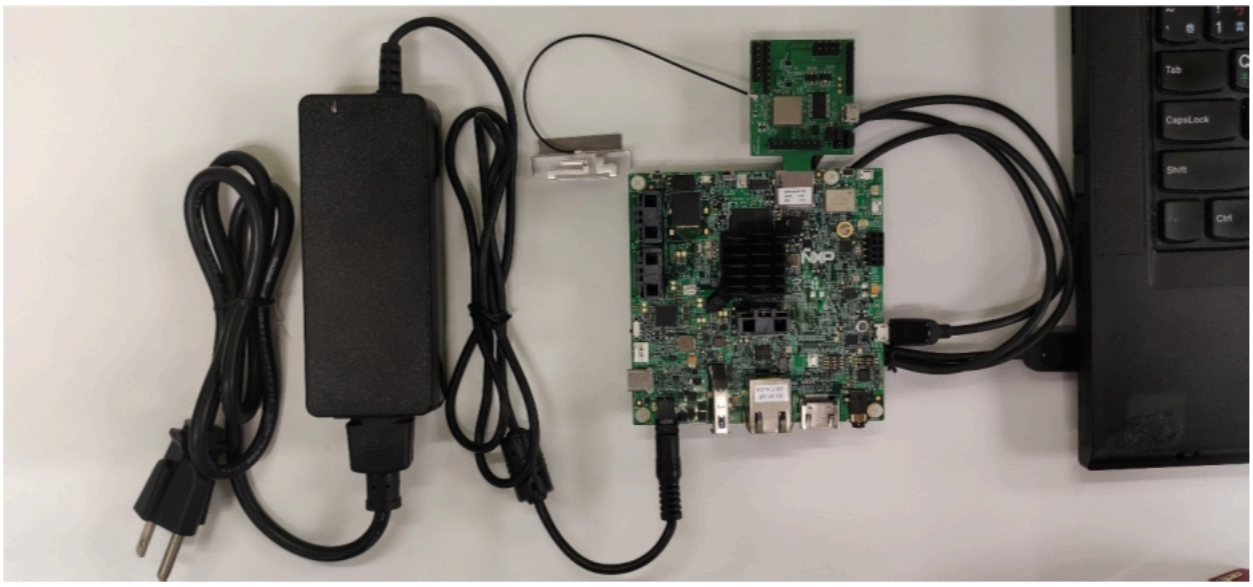

Figure 12. AzureWave AZ-CM358-uSD module and i.MX 8M Quad EVK setup

3.7 88W8987-based AzureWave AW-CM358MA module (SDIO-UART)

The AW-CM358MA module supports a SDIO device interface that conforms to the industry standard SDIO Full-Speed card specification and allows a host controller using the SDIO bus protocol to access the Wireless SoC device.

The AW-CM358MA is a Wi-Fi 5 and Bluetooth 5 combo stamp module with the M.2 adapter board that features:

- SDIO 3.0 standard
- On-chip memory used for CIS
- 1-bit SDIO and 4-bit SDIO transfer modes
- A special interrupt register for information exchange

3.7.1 Recommended antenna part

MAG.LAYERS: MSA-4008-25GC1-A2

3.7.2 Supported RF standards

Table 6. AW-CM358MA supported RF standards

Part number	Wi-Fi	Bluetooth
AW-CM358MA	1x1 Wi-Fi 5 (2.4/5GHz)	5.0

3.7.3 Supported Wi-Fi features

See ref.[8].

3.7.4 Supported Bluetooth features

See ref.[8].

3.7.5 AW-CM358MA module view

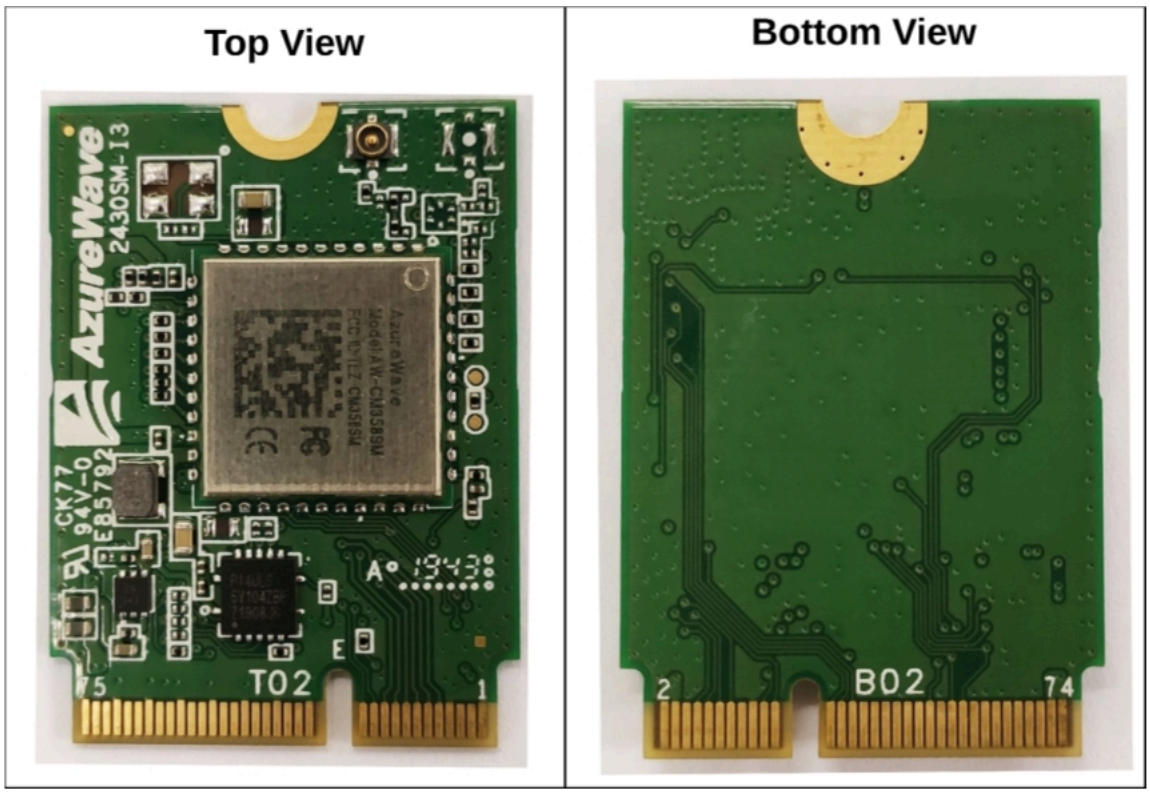

Figure 13. AzureWave AW-CM358MA module

3.7.6 Enabling SDIO on M.2 (AW-CM358MA and AW-CM276MA-SUR)

The command below sets as default the DTB file that enables the SDIO interface on the M.2 connector, and reboots the EVK to load the updated DTB file.

```
root@imx8mqevk:~# mv /run/media/mmcblk0p1/imx8mq-evk-usdhc2-m2.dtb /run/media/mmcblk0p1/
imx8mq-evk.dtb
root@imx8mqevk:~# reboot
```

Note: *This section only applies to AW-CM358MA and AW-CM276MA-SUR modules based on M.2 connector. The SDIO on M.2 DTB is necessary to enable the SDIO interface on the M.2 connector. By default the SDIO support is enabled for the SD Card slot connector. The hardware rework instructions are given in* *Section 3.7.7 "i.MX 8M Quad rework for SDIO support on M.2"**.*

3.7.7 i.MX 8M Quad rework for SDIO support on M.2

The section shows how to rework i.MX 8M Quad resistors to support SDIO on the M.2 connector for the AW-CM358MA module.

The M.2 connector on AW-CM358MA module supports SDIO interface for Wi-Fi. By default, the i.MX 8M Quad M.2 connector supports PCIe interface. To enable SDIO over M.2 on i.MX 8M Quad, remove the bridge resistors connected to the micro SD slot. Next, install the resistors on the M.2 connector (Figure 16 and Figure 18).

Note:

- *The hardware rework instructions reroute the SDIO interface from the SD card connector to the M.2 connector. A software change is also required to enable the rerouted SDIO interface. To enable the SDIO interface on the M.2 connector using a DTB file, see Section 3.7.6.*
- *The SDIO on the Micro SD card slot is disabled after the rework.*
- *To enable the SDIO over M.2 connector when using i.MX 8M Mini or i.MX 8M Nano platforms, no rework of the bridge resistors is required.*
- Silkscreen of PCBA SCH-38820

Figure 14. Silkscreen of PCBA SCH-38820 REV A

- Silkscreen of PCBA SCH-29615

Figure 15. Silkscreen of PCBA SCH-29615 REV B4

- Remove the following 0 Ω 0402 resistors: R1603, R1617, R1618, R1619, R1620 and R1621 (micro SD card J1601).

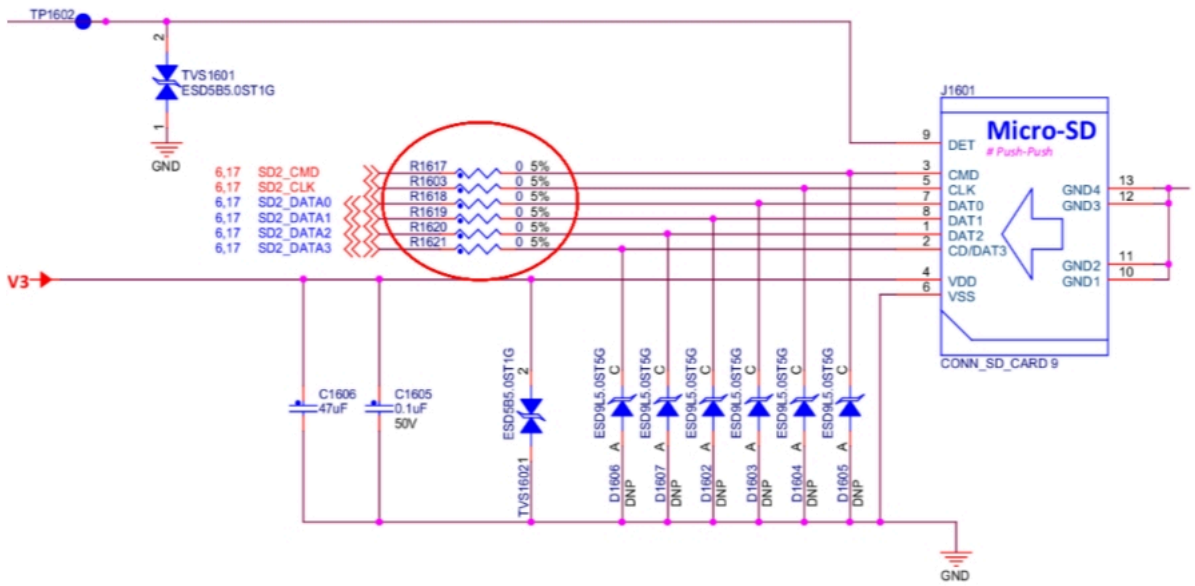

Figure 16. Resistors R1603, R1617, R1618, R1619, R1620 and R1621 (micro SD card J1601)

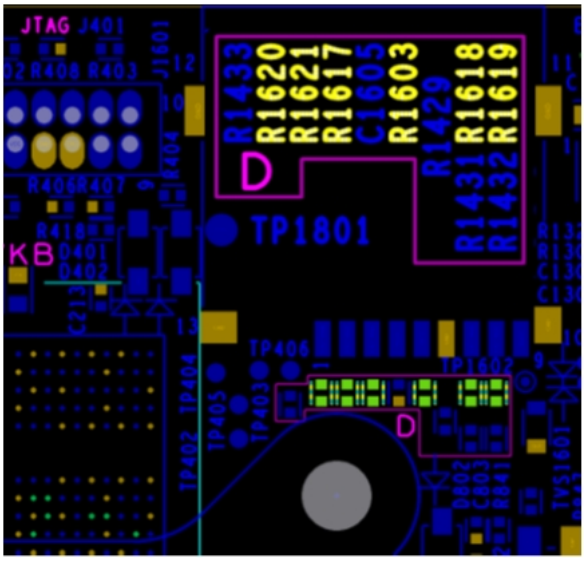

Figure 17. MicroSD Card J1601 resistors

- Install the following 0Ω 0402 resistors: R1429, R1430, R1431, R1432, R1433, R1434, R1435 and R1436 (M.2 J1401).

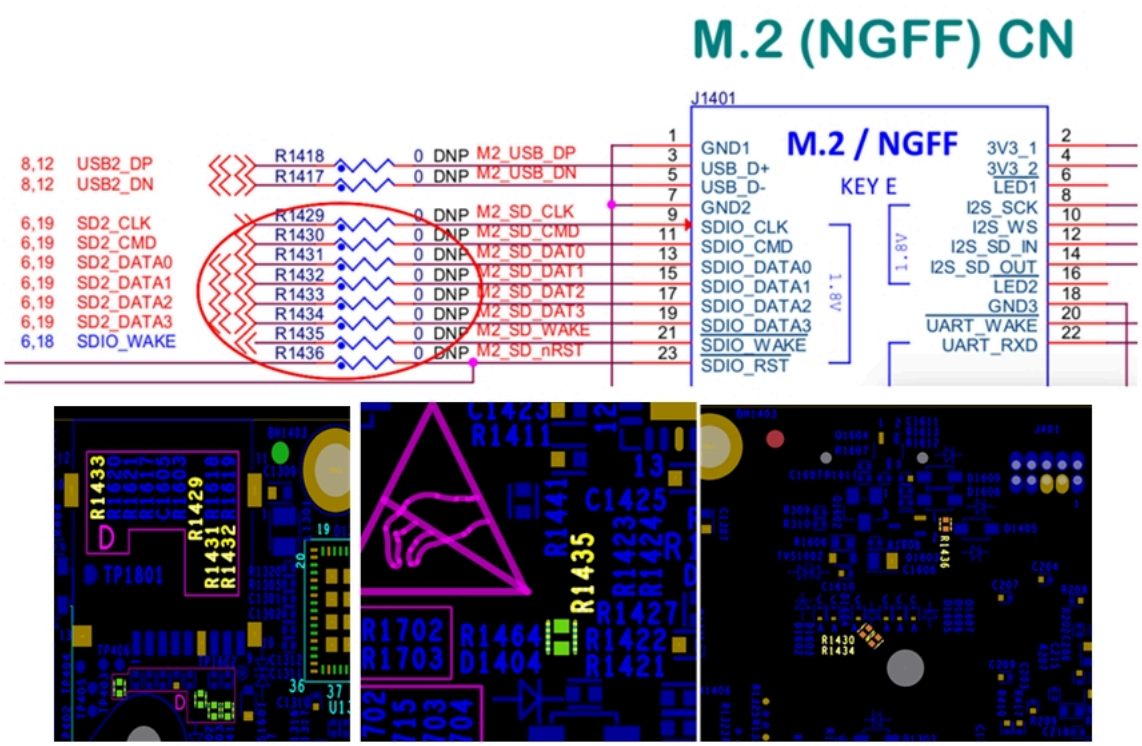

Figure 18. M.2 J1401 resistors

3.7.8 AW-CM358MA module setup with iMX 8M Quad

Connect AW-CM358MA module into the M.2 connector of the i.MX 8M Quad board and screw.

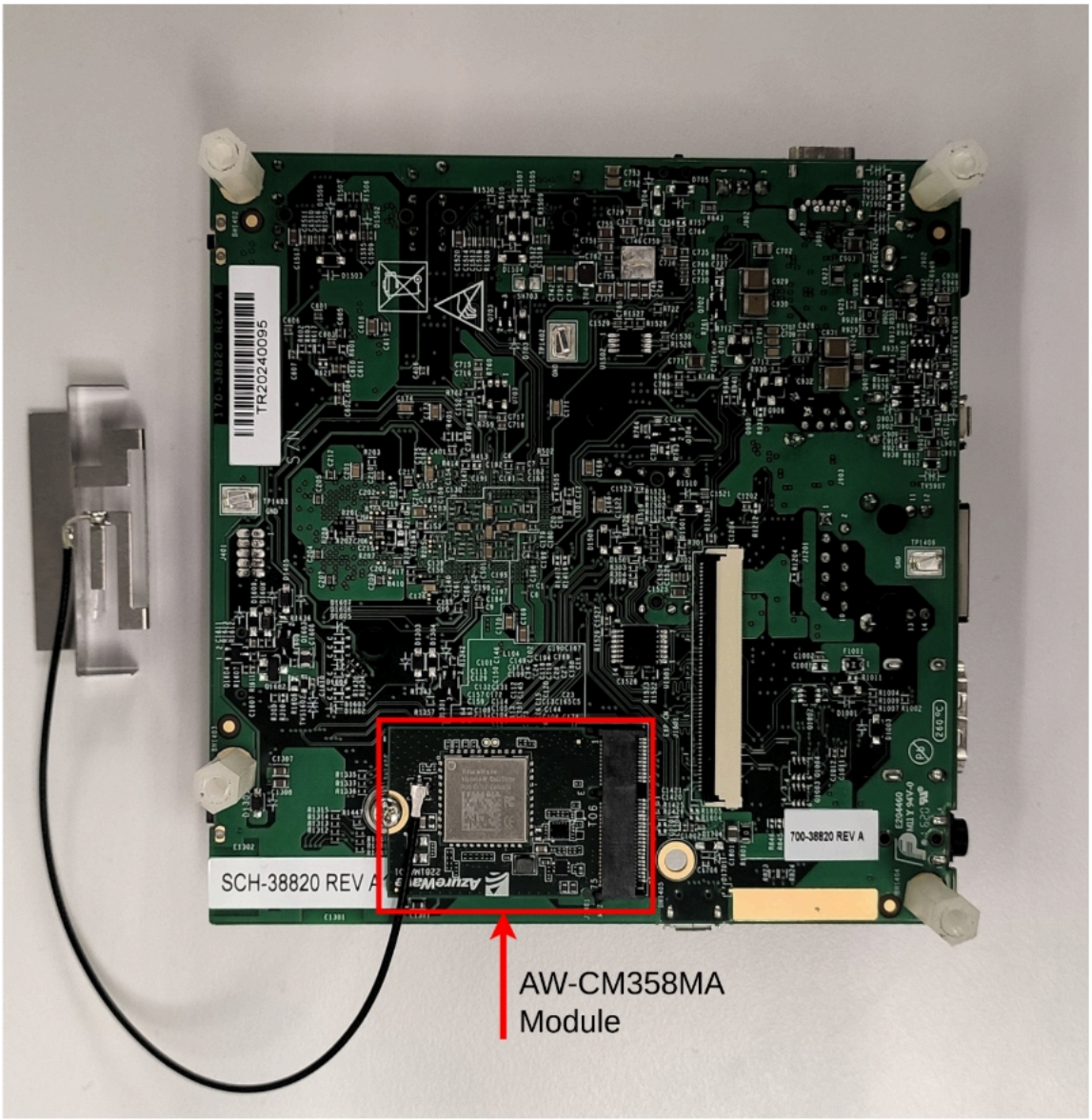

Figure 19. Azurewave AW-CM358MA module plugged into i.MX 8M Quad bottom side M.2 connector

Connect the antenna, and use a Micro USB to USB cable to connect i.MX 8M Quad EVK to the host computer running on Linux OS.

Figure 20. Azurewave AW-CM358MA module and i.MX 8M Quad setup

3.8 88W8987-based Panasonic PAN9028 module (SDIO-UART)

The PAN9028 module is a 2.4 GHz and 5 GHz ISM band Wi-Fi and Bluetooth radio module placed on the mSDIO adapter board.

The SDIO interface for Wi-Fi conforms to the industry standard SDIO Full-Speed card specification. The host controller uses SDIO bus protocol to access 88W8987 wireless SoC device. SDIO high-speed interface complies with the industry standard 16550 specification.

The UART interface for Bluetooth can be used separately.

PAN9028 Wi-Fi 5 and Bluetooth 5 combo module features:

- SDIO 3.0 standard
- On-chip memory used for CIS
- 1-bit SDIO and 4-bit SDIO transfer modes with full clock range up to 208 MHz
- A special interrupt register for information exchange

3.8.1 Supported RF standards

Table 7. PAN9028 supported RF standards

Part number	Wi-Fi	Bluetooth
PAN9028	1x1 Wi-Fi 5 (2.4/5 GHz)	5.0

3.8.2 Supported Wi-Fi features

See ref.[24].

3.8.3 Supported Bluetooth features

See ref.[24].

3.8.4 PAN9028 module view

Figure 21 shows PAN9028 module on mSDIO adapter.

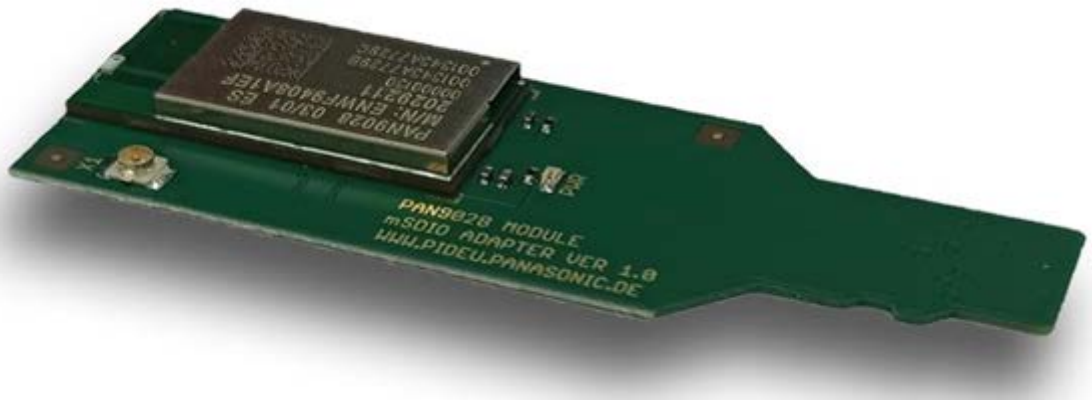

Figure 21. PAN9028 module view

3.8.5 microSDIO adapter

3.8.5.1 Functional blocks

Table 8 details the functional blocks highlighted in Figure 22.

Table 8. Functional blocks

Functional block	Description	Interface
A	microSDIO adapter	SDIO
B	PAN9028 module	—
C	On-board antenna or bottom pad selection	—
D	U.FL connector	X1

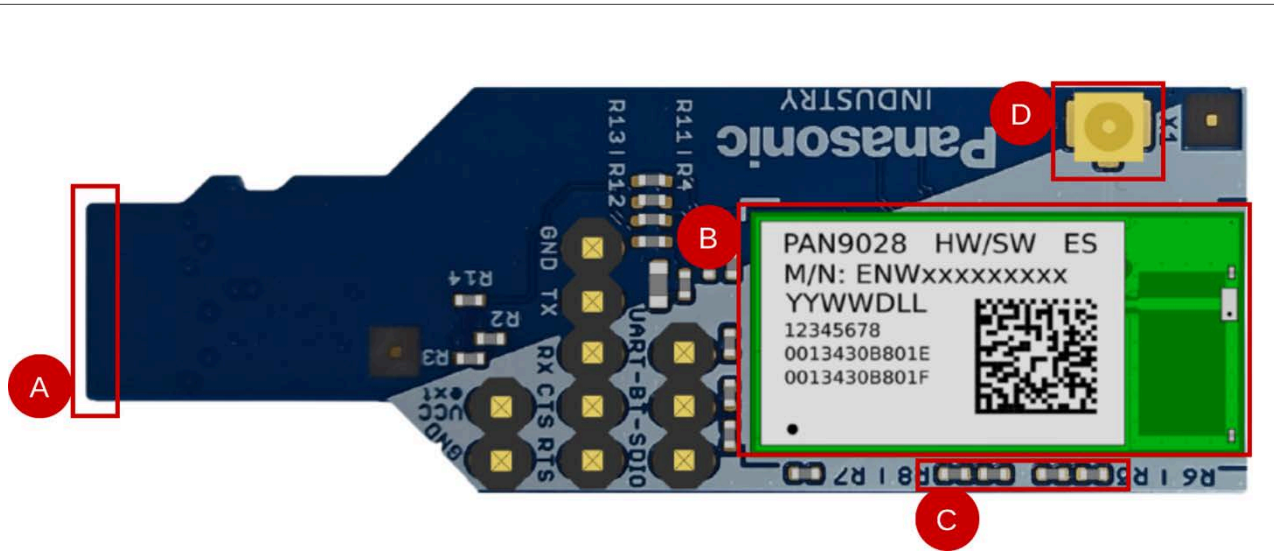

Figure 22. Functional blocks on microSDIO adapter

3.8.5.2 Resistor configurations

Table 9 details the resistor configurations on microSDIO adapter.

Table 9. Resistor configuration on microSDIO adapter

RF-Out	R6	R7
PAN9028 chip antenna	0R	Not connected
RF-UFL X1 connector	Not connected	0R

Figure 23 illustrates the resistor configurations shown in Table 9.

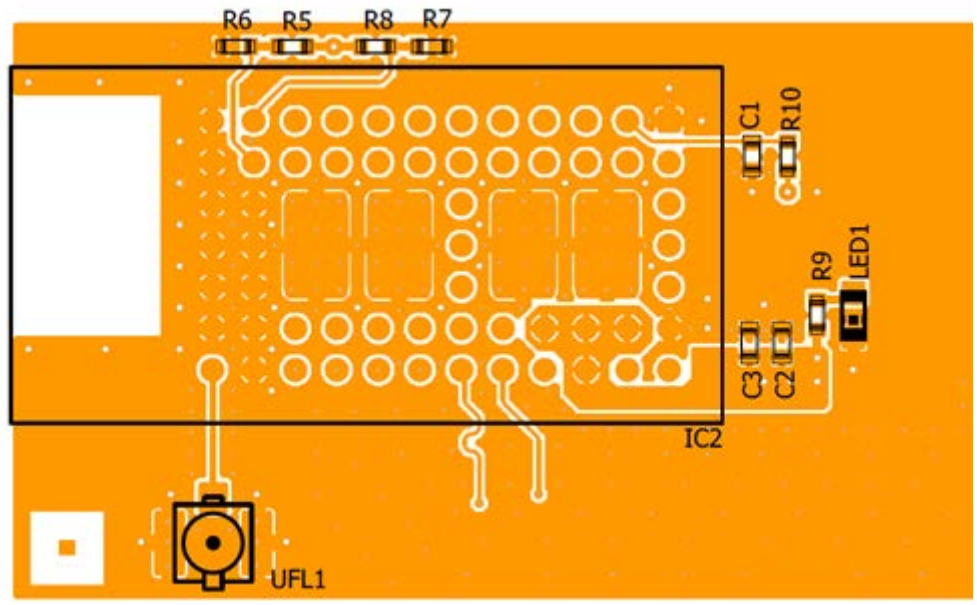

Figure 23. Resistors on microSDIO adapter

3.9 88W8997-based AzureWave AW-CM276MA-PUR module (PCIe-UART)

AzureWave AW-CM276MA-PUR supports PCIe and high-speed UART interfaces to the host processor for Wi-Fi and Bluetooth with the M.2 adapter board. See ref.[9]. AW-CM276MA-PUR module features the following:

- PCIe M.2 TYPE connector
- PCIe interface for Wi-Fi
- UART interface for Bluetooth

Figure 24. AW276MA-PUR module top and bottom views

3.9.1 Recommended antenna part

MAG.LAYERS: MSA-4008-25GC1-A2

3.9.2 Supported RF standards

Table 10. AW-CM276MA-PUR supported RF standards

Part number	Wi-Fi	Bluetooth
AW-CM276MA-PUR	2x2 Wi-Fi 5 (2.4 GHz/5 GHz)	5.0

3.9.3 Supported Wi-Fi features

See ref.[9].

3.9.4 Supported Bluetooth features

See ref.[9].

3.10 88W8997-based AzureWave module AW-CM276MA-SUR (SDIO-UART)

The AzureWave AW-CM276MA-SUR module based on 88W8997 wireless device supports a SDIO device interface that conforms to the industry standard SDIO Full-Speed card specification and allows a host controller using the SDIO bus protocol to access the wireless SoC device. The AW-CM276MA-SUR is a Wi-Fi 5 and Bluetooth 5 combo stamp module with the M.2 adaptor board that features:

- SDIO 3.0 standard
- On-chip memory used for CIS
- 1-bit SDIO and 4-bit SDIO transfer modes
- Special interrupt register for information exchange
- Allows card to interrupt host

3.10.1 Recommended antenna part

- MAG.LAYERS: MSA-4008-25GC1-A2

3.10.2 Supported RF standard

Table 11. AW-CM276MA-SUR supported RF standards

Part number	Wi-Fi	Bluetooth
AW-CM276MA-SUR	Wi-Fi 5	Bluetooth 5.0

3.10.3 Supported Wi-Fi features

See ref.[10].

3.10.4 Supported Bluetooth features

See ref.[10].

3.10.5 AzureWave AW-CM276MA-SUR Wi-Fi module interface

Figure 25. AzureWave AW-CM276MA-SUR Wi-Fi module interface

3.11 88W8997-based AzureWave AW-CM276-uSD Wi-Fi module (SDIO)

Azurewave provides a micro-SD card interface adapter (uSD-1216) with AW-CM276NF module that is compatible with the i.MX 8M Quad EVK. The AW-CM276-uSD supports Wi-Fi through uSD device interface that conforms to the industry standard SDIO Full-Speed card specification and allows a host controller using the SDIO bus protocol to access the Wireless SoC device. The AW-CM276NF acts as the device on the SDIO bus, and features the following:

- SDIO 3.0 standard
- On-chip memory used for CIS
- Supports 1-bit SDIO and 4-bit SDIO transfer modes
- Special interrupt register for information exchange
- Allows card to interrupt host

3.11.1 Recommended antenna part

- MAG.LAYERS: MSA-4008-25GC1-A2

3.11.2 Supported I/O signal level

SDIO (3.0/2.0) supports 1.8V/3.3V for I/O signal.

3.11.3 Supported RF standard

Table 12. AW-CM276-uSD supported RF standards

Part number	Wi-Fi	Bluetooth
AW-CM276NF	2x2 Wi-Fi 5	Bluetooth 5.0

3.11.4 Supported Wi-Fi features

See AzureWave AW-CM276NF data sheet.

Note: *AzureWave AW-CM276NF supports both Wi-Fi and Bluetooth RF standards. But as i.MX 8M Quad EVK does not include the FFC connector for Bluetooth interface, only Wi-Fi is supported for i.MX 8M Quad platform with AW-CM276-uSD module combination.*

3.11.5 AzureWave AW-CM276-uSD Wi-Fi module interface

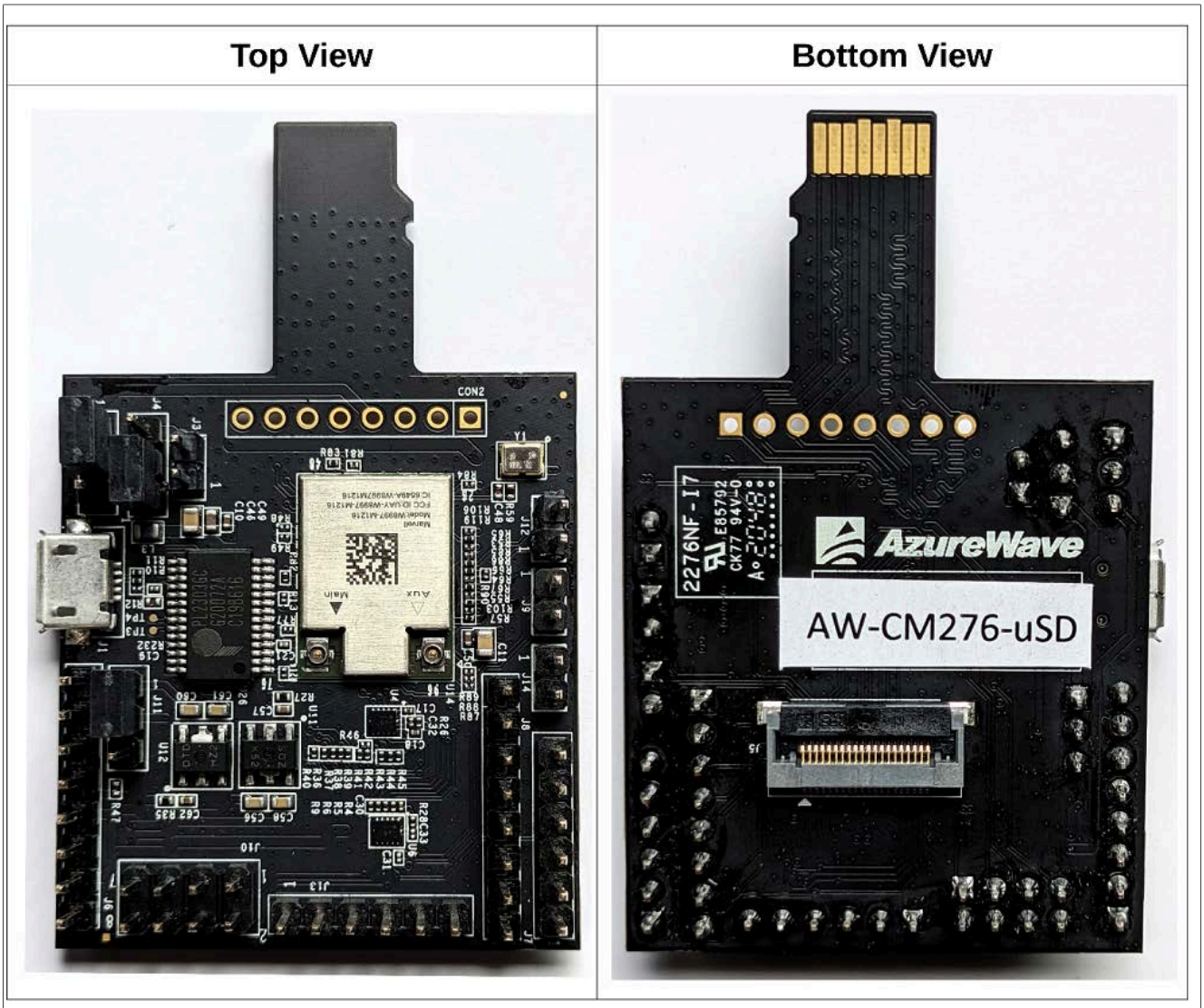

Figure 26. AzureWave AW-CM276-uSD Wi-Fi module interface

3.11.6 AzureWave AW-CM276-uSD module jumpers and power supply

This section provides the jumper settings to configure the module with 1.8 V SDIO voltage for Wi-Fi.

See Table 13 and Figure 27.

Table 13. Jumper settings for AW-CM276-uSD module

Jumper	Header pins	Description
J2	(1-2)	SDIO module power source
J4	(1-2)	1.8V VIO voltage
J11	(1-2)	1.8V SDIO_SD voltage

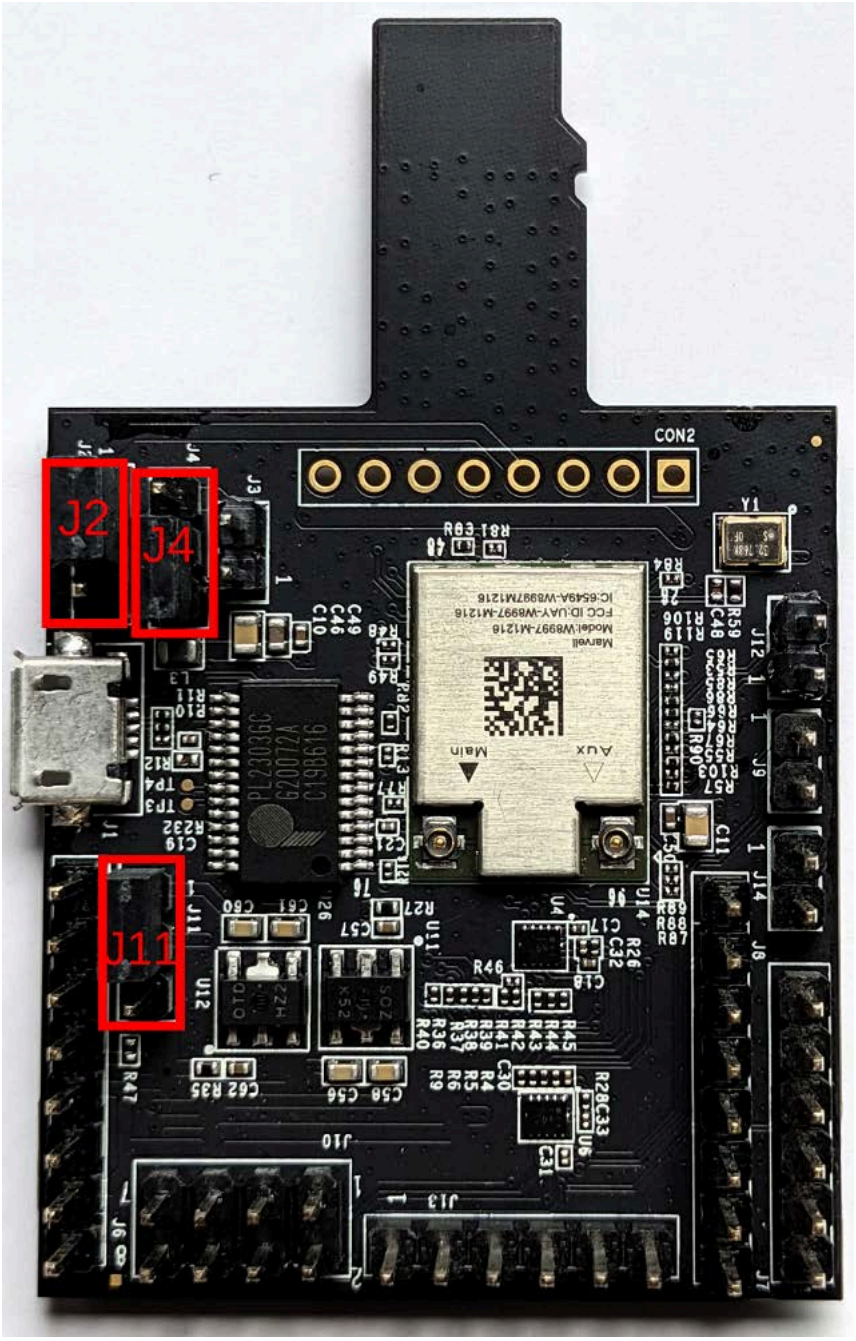

Figure 27. AzureWave AW-CM276-uSD module jumpers and power supply

3.12 88W9098-based Murata module LBEE6ZZ1 (PCIe-UART)

LBEE6ZZ1 is a wireless module from Murata based on NXP's 88W9098 wireless device. Murata provides an M.2 interface with LBEE6ZZ1 module that is compatible with the i.MX 8M Quad EVK. The LBEE6ZZ1 supports Wi-Fi through a PCIe device interface that conforms to the industry PCIe 2.0 specifications and allows a host controller using the PCIe bus protocol to access the wireless SoC device. The LBEE6ZZ1 acts as the device on the PCIe bus, and features the following:

- PCIe 2.0 interface for Wi-Fi
- UART interface for Bluetooth
- IEEE 802.11 a/b/g/n/ac/ax 2x2 MU-MIMO

3.12.1 Recommended antenna

- Dual-band antenna with u.FL connector

3.12.2 Supported RF standards

Table 14. LBEE6ZZ1 supported RF standards

Part number	Wi-Fi	Bluetooth
LBEE6ZZ1	Wi-Fi 6 2x2	5.1

3.12.3 Supported Wi-Fi features

See ref.[3].

3.12.4 Supported Bluetooth features

See ref.[3].

3.12.5 Murata LBEE6ZZ1 Wi-Fi module interface

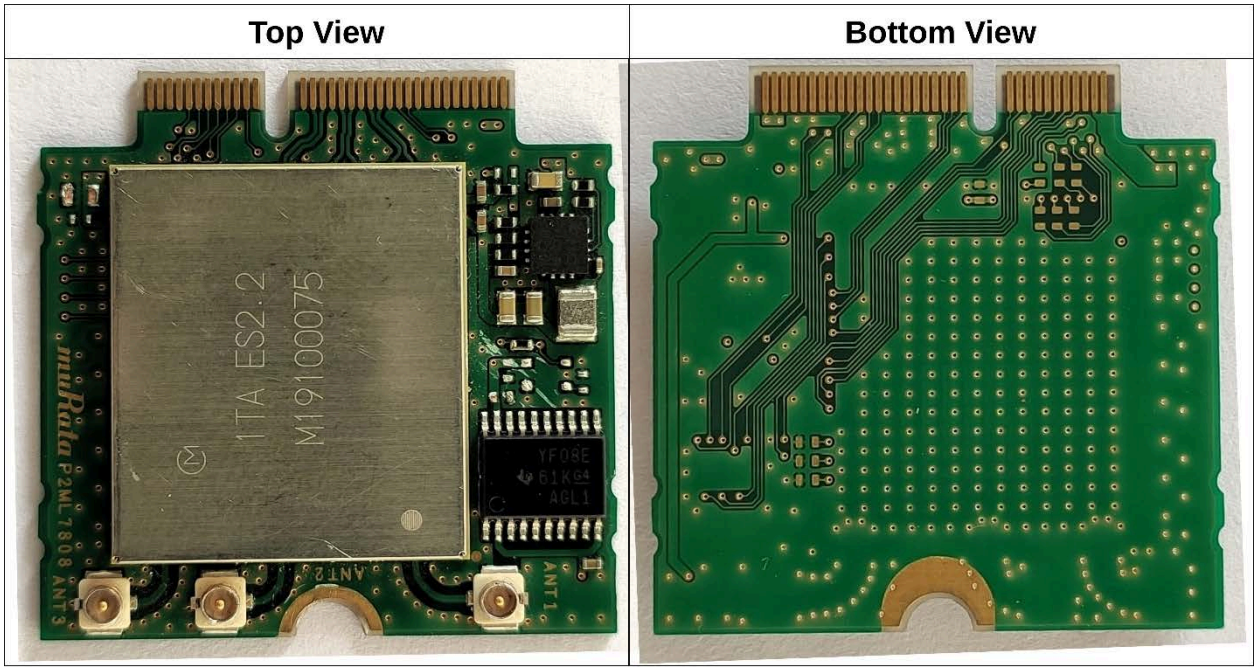

Figure 28. Murata LBEE6ZZ1 Wi-Fi module interface

3.13 88W9098-based Murata module LBEE5ZZ1XL (SDIO-UART)

The LBEE5ZZ1XL is a wireless module from Murata based on NXP's 88W9098 wireless device. Murata provides an M.2 interface with LBEE5ZZ1XL module that is compatible with the i.MX 8M Quad EVK. The LBEE5ZZ1XL supports Wi-Fi through PCIe 3.0 interface, with optional support for SDIO 3.0 standard and the Bluetooth section supports through high-speed 4-wire UART interface. The module acts as the device on the SDIO bus, and features the following:

- SDIO 3.0 standard
- 1-bit SDIO and 4-bit SDIO transfer modes
- SDIO interface for Wi-Fi
- UART interface for Bluetooth

Note: *For details on how to select Wi-Fi through SDIO interface, see ref.[12].*

3.13.1 Supported RF standards

Table 15. LBEE5ZZ1XL supported RF standards

Part number	Wi-Fi	Bluetooth
LBEE5ZZ1XL	Wi-Fi 6 2x2	5.2

3.13.2 Supported Wi-Fi features

See ref.[3].

3.13.3 Supported Bluetooth features

See ref.[3].

3.13.4 LBEE5ZZ1XL module view

Figure 29 shows the top and bottom views of LBEE5ZZ1XL module.

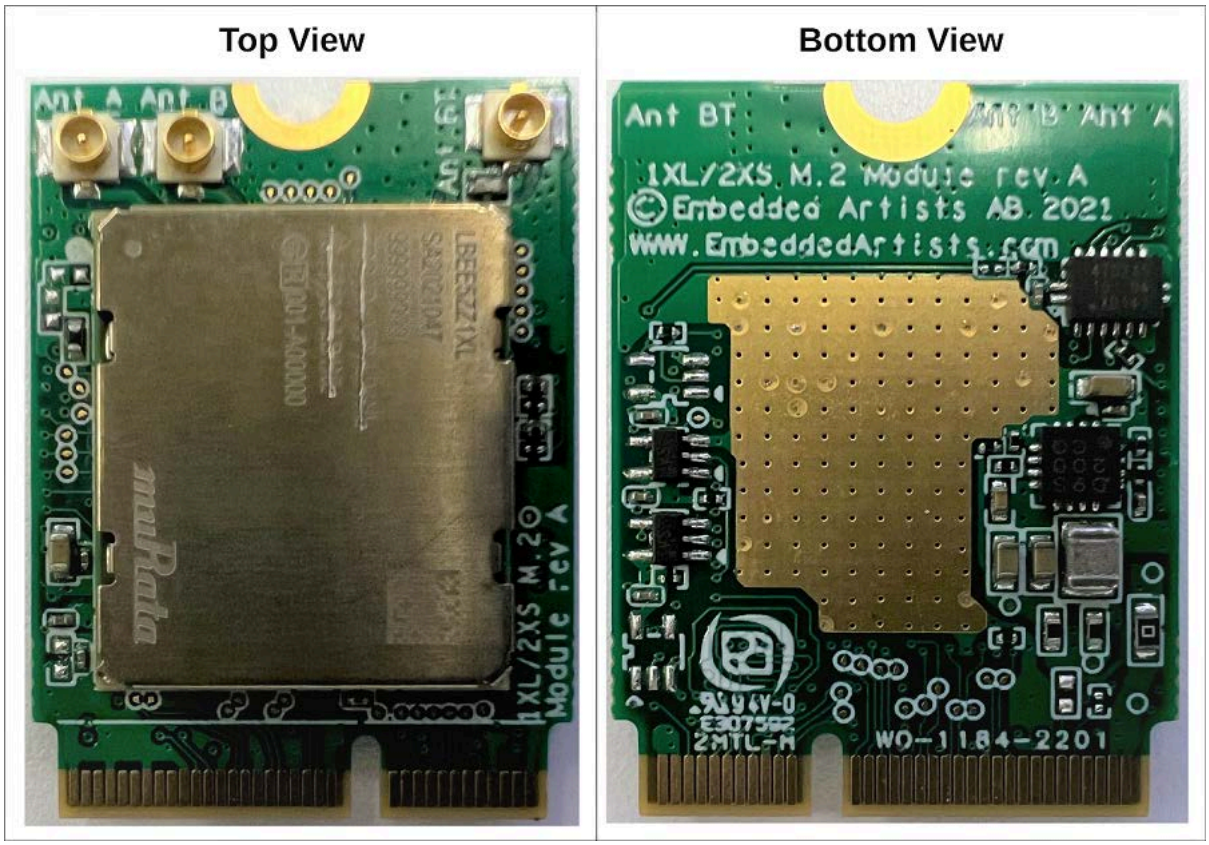

Figure 29. Top and bottom views of LBEE5ZZ1XL module

Note: *See Section 3.7.7 for the rework instructions to support SDIO interface on M.2.*

3.14 IW416-based AzureWave AW-AM510MA module (SDIO-UART)

The AW-AM510MA module supports a SDIO device interface that conforms to the industry standard SDIO Full-Speed card specification and allows a host controller using the SDIO bus protocol to access the Wireless SoC device. The AW-AM510MA is a Wi-Fi 4 and Bluetooth 5.1 combo stamp module with the M.2 adaptor board that features:

- SDIO 3.0 standard
- On-chip memory used for CIS
- 1-bit SDIO and 4-bit SDIO transfer modes
- One special interrupt register for information exchange

3.14.1 Recommended antenna part

MAG.LAYERS: MSA-4008-25GC1-A2

3.14.2 Supported RF standards

Table 16. AW-AM510MA supported RF standards

Part number	Wi-Fi	Bluetooth
AW-AM510MA	1x1 Wi-Fi 4 (2.4 GHz/5 GHz)	5.1

3.14.3 Supported Wi-Fi features

See ref.[11].

3.14.4 Supported Bluetooth features

See ref.[11].

3.14.5 AW-AM510MA module view

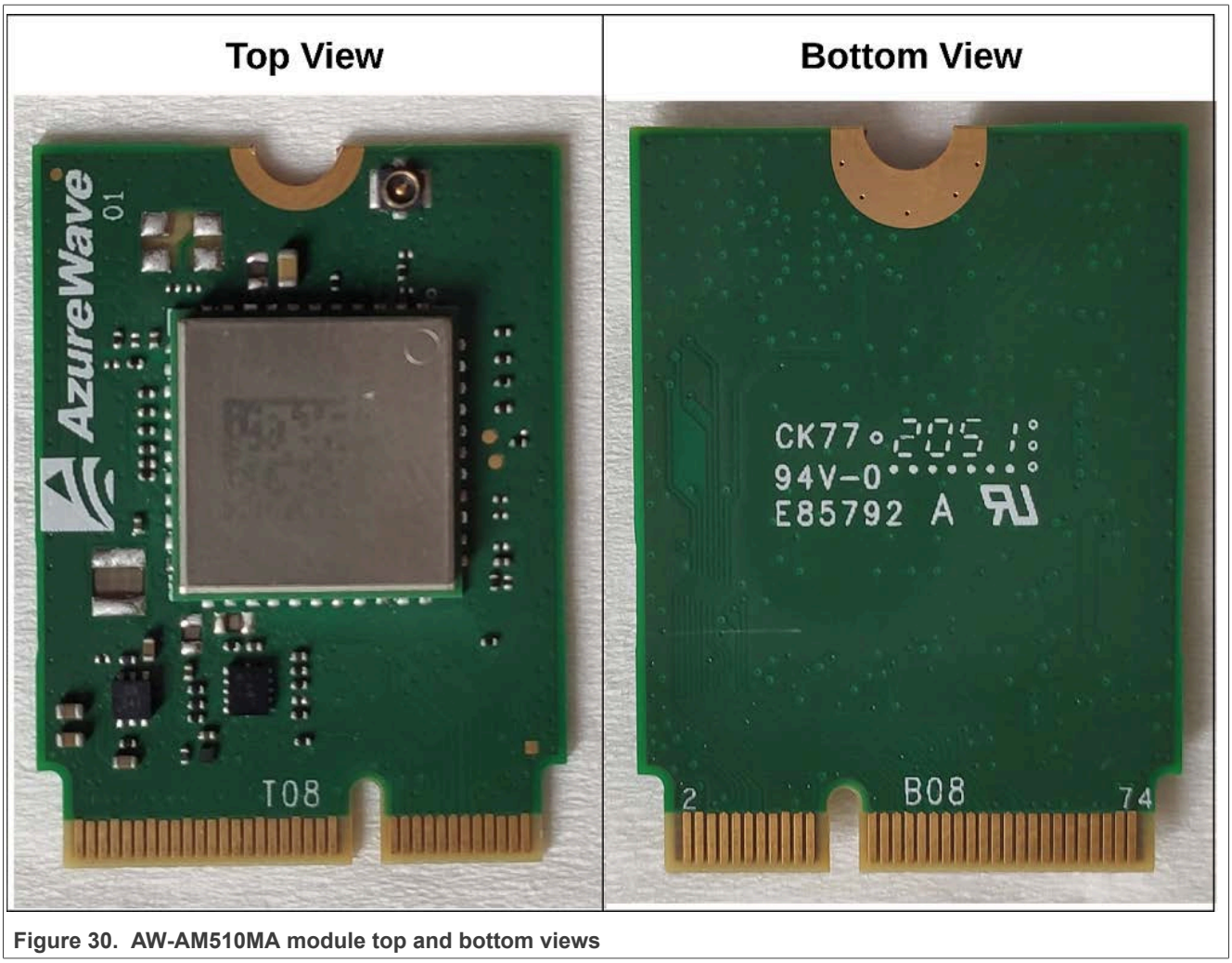

Figure 30. AW-AM510MA module top and bottom views

Note: *See Section 3.7.7 for rework instructions to support SDIO interface on M.2.*

3.15 88W8801-based Murata LBWA0ZZ2DS Wi-Fi module (SDIO)

The LBWA0ZZ2DS is a wireless module from Murata based on NXP 88W8801 Wi-Fi device. Murata provides an M.2 interface with LBWA0ZZ2DS module that is compatible with the i.MX 8M Quad EVK. The LBWA0ZZ2DS supports Wi-Fi through an SDIO device interface with SDIO 2.0 standard. The module acts as the device on the SDIO bus, and features the following:

- SDIO 2.0 standard
- 1-bit SDIO and 4-bit SDIO transfer modes
- Integrated antenna

3.15.1 Supported RF standards

Table 17. LBWA0ZZ2DS supported RF standards

Part number	Wi-Fi
LBWA0ZZ2DS	1x1 Wi-Fi 4 (2.4 GHz)

3.15.2 Supported Wi-Fi features

See ref.[13].

3.15.3 LBWA0ZZ2DS module view

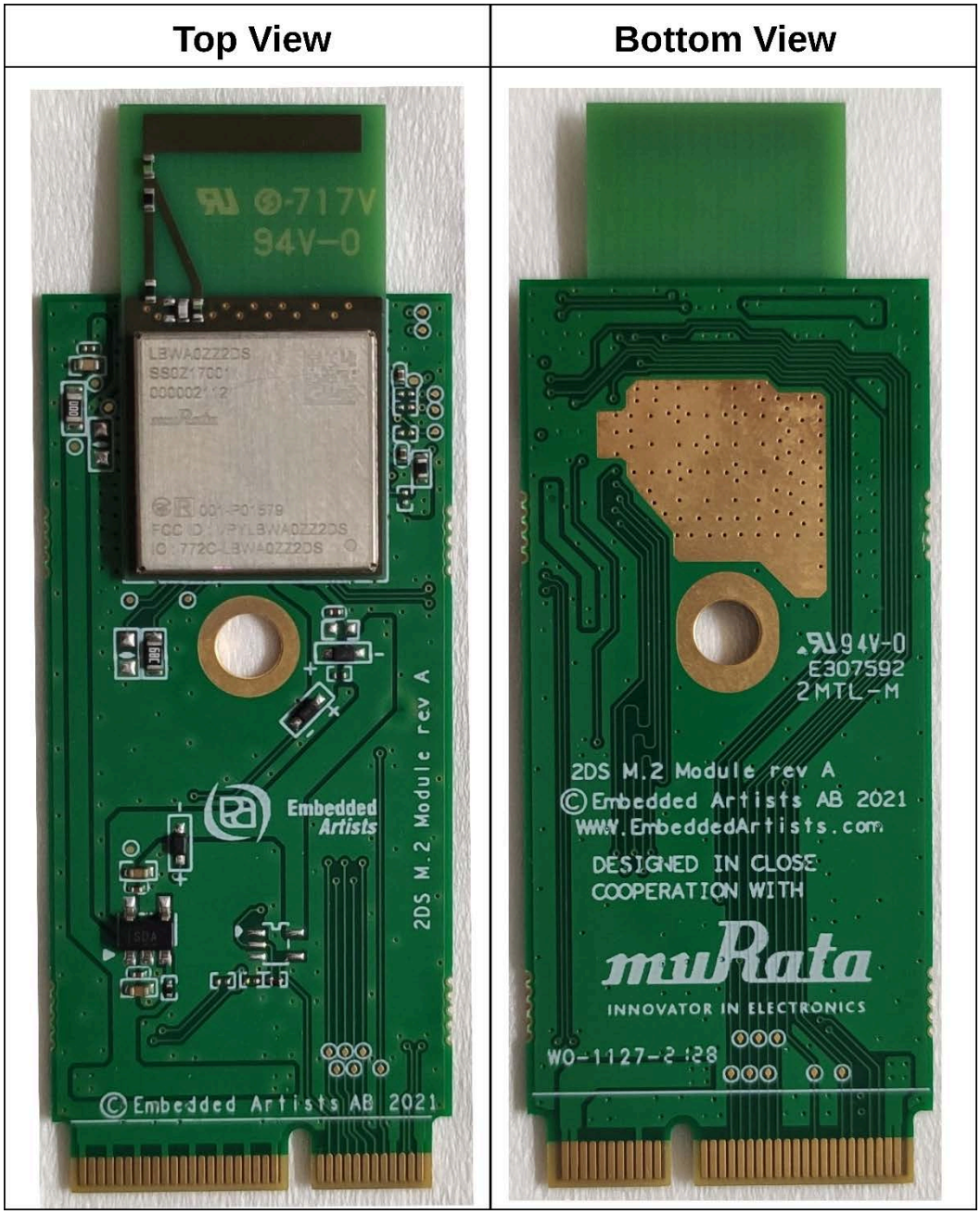

Figure 31. Top and bottom views of LBWA0ZZ2DS module

Note: *See Section 3.7.7 for rework instructions to support SDIO interface on M.2.*

3.16 IW612-based Murata module LBES5PL2EL

The LBES5PL2EL is wireless module from Murata based on NXP IW612 wireless device. Murata provides an M.2 interface with LBES5PL2EL module that is compatible with the i.MX 8M Quad EVK. The module acts as the device on the SDIO bus, and features the following:

- SDIO 3.0 standard
- 1-bit SDIO and 4-bit SDIO transfer modes
- SDIO interface for Wi-Fi
- UART interface for Bluetooth
- SPI interface for 802.15.4

3.16.1 Supported RF standards

Table 18. LBES5PL2EL supported RF standards

Part number	Wi-Fi	Bluetooth	802.15.4
LBES5PL2EL	1x1 Wi-Fi 6	5.2[1]	IEEE 802.15.4-2015

[1] Certified 5.3

3.16.2 Supported Wi-Fi features

See ref.[7].

3.16.3 Supported Bluetooth features

See ref.[7].

3.16.4 Supported 802.15.4 features

See ref.[7].

Note: *LBES5PL2EL module requires additional hardware rework to support the 802.15.4 protocol using the SPI interface. To get the hardware rework details, reach out to NXP support team.*

3.16.5 LBES5PL2EL module view

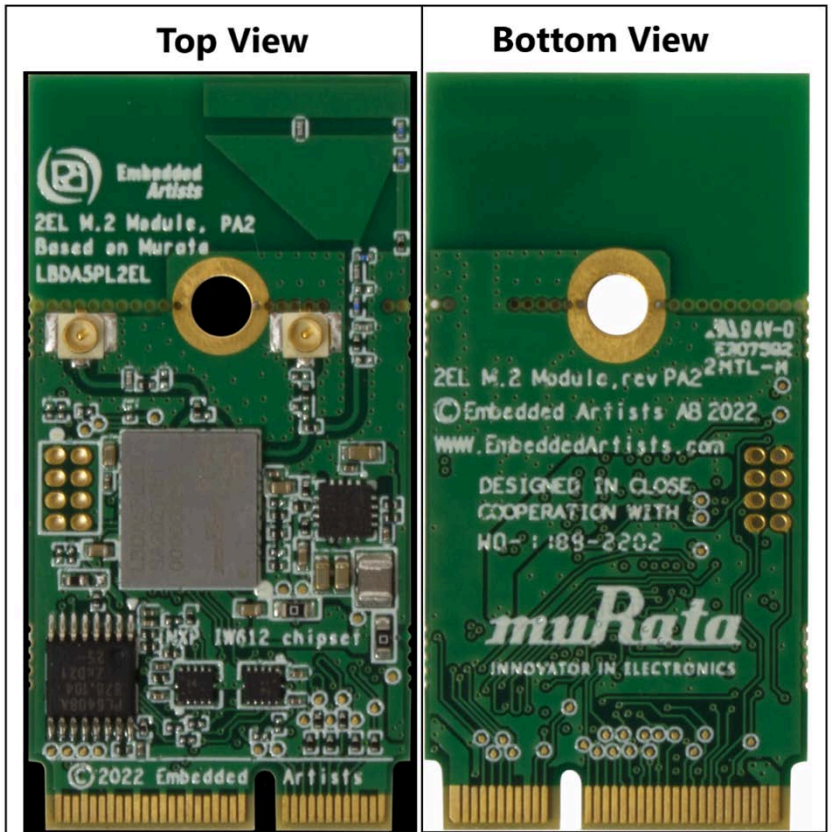

Figure 32. Top and bottom views of LBES5PL2EL module

Note: *See Section 3.7.7 for the rework instructions to support SDIO interface on M.2.*

3.17 IW610-based Murata module LBES0ZZ2LL

The LBES0ZZ2LL is the wireless module from Murata based on NXP IW610 wireless device. Murata provides an M.2 interface with LBES0ZZ2LL module that is compatible with the i.MX 8M Quad EVK. The module acts as the device on the SDIO bus, and features the following:

- SDIO 3.0 standard
- 1-bit SDIO and 4-bit SDIO transfer modes
- SDIO interface for Wi-Fi
- UART interface for Bluetooth
- SPI interface for 802.15.4
- USB interface configurable for Wi-Fi
- USB interface configurable for Bluetooth

3.17.1 How to switch between USB-USB and SDIO-UART host interface configurations

IW610 module includes a switch with sliders to set the configuration to SDIO-UART or USB-USB for Wi-Fi and Bluetooth host interfaces.

Table 19 details the settings for USB-USB and SDIO-UART host interface configurations.

Table 19. Switch settings for USB-USB and SDIO-UART configurations

Switch settings	Wi-Fi	Bluetooth	802.15.4
Both sliders on the switch in OFF position (away from the edge golden fingers)	SDIO	UART	SPI
Both sliders on the switch in ON position (towards the edge golden fingers)	USB	USB	SPI

Figure 33 shows the switches used to select the host interfaces for Wi-Fi and Bluetooth.

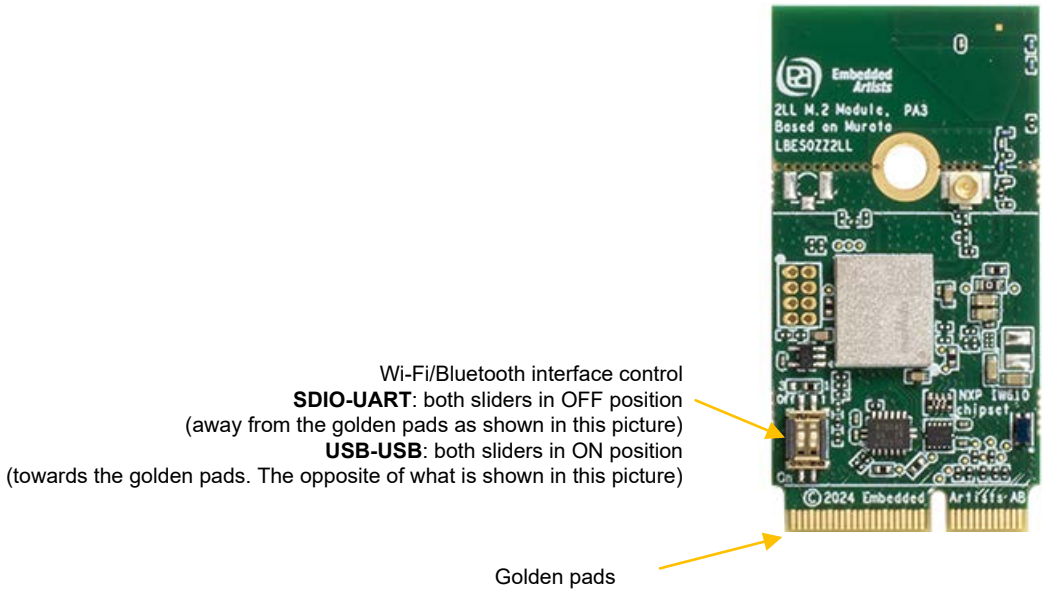

Figure 33. Switch used for Wi-Fi host interface selection on IW610 module

3.17.2 Supported RF standards

Table 20. LBES0ZZ2LL supported RF standards

Part number	Wi-Fi	Bluetooth	802.15.4
LBES0ZZ2LL	1x1 Wi-Fi 6	5.4	IEEE 802.15.4-2015

3.17.3 Supported Wi-Fi features

See IW610 data sheet ref.[6].

3.17.4 Supported Bluetooth features

See IW610 data sheet ref.[6].

3.17.5 Supported 802.15.4 features

See IW610 data sheet ref.[6].

Note: *LBES0ZZ2LL module requires additional hardware rework to support the 802.15.4 protocol using the SPI interface. To get the hardware rework details, reach out to NXP support team.*

3.17.6 LBES0ZZ2LL module view

Figure 34. Top and bottom views of LBES0ZZ2LL module

Note: *See Section 3.7.7 for the rework instructions to support SDIO interface on M.2.*

3.18 AW693-based u-blox module JODY-W683

JODY-W683 is a wireless module from u-blox based on NXP's AW693 wireless device. u-blox provides an M.2 interface with JODY-W683 module that is compatible with the i.MX 8M Quad EVK. The JODY-W683 supports Wi-Fi through a PCIe device interface that conforms to the industry PCIe 2.0 specifications and allows a host controller using the PCIe bus protocol to access the wireless SoC device.

The JODY-W683 acts as the device on the PCIe bus, and features the following:

- PCIe 2.0 interface for Wi-Fi
- UART interface for Bluetooth
- IEEE 802.11 a/b/g/n/ac/ax 2x2 MU-MIMO

3.18.1 Supported RF standards

Table 21. JODY-W683 supported RF standards

Part number	Wi-Fi	Bluetooth
JODY-W683	2x2 Wi-Fi 6E	5.4

3.18.2 Supported Wi-Fi and Bluetooth features

See AW693 data sheet (ref.[4]).

3.18.3 JODY-W683 module view

Front view

Bottom view

Figure 35. JODY-W683 module view

4 i.MX 8M EVK Linux image setup

The i.MX 8M Quad Linux BSP is a collection of binary files, source code, and support files used to create a U-Boot bootloader, a Linux kernel image, and a root file system for i.MX 8M Quad development platforms.

All the information on how to set up the Linux OS host, how to run and configure a Yocto Project, generate an image, and generate a rootfs, are covered in the i.MX Yocto Project User's Guide. On i.MX 8M Quad, four elements are needed:

- imx-boot (built by imx-mkimage), which includes SPL, U-Boot, Arm Trusted Firmware, DDR firmware, and HDMI firmware
- Linux kernel image
- A device tree file (*.dtb*) for the board being used
- A root file system (*rootfs*) for the specific Linux image

Read more in section 4.1 of *i.MX Linux® User's Guide*, document number “IMXLUG”.

4.1 Using the pre-built image

This section describes the steps to prepare eMMC to boot up an i.MX 8M Quad EVK using a Linux host machine. The pre-built image can be downloaded from the page Embedded Linux for I.MX Application Processors on NXP website. Accept NXP software license agreement when prompted.

The release contains a pre-built image that is built specifically for the board configuration. It does not run on other boards unless U-Boot, the device tree, and rootfs are changed.

Note: *A pre-built image contains all the image elements and does not require to build the image using the Yocto setup.*

The content of the extracted downloaded release archive is shown hereafter.

```
├── EULA.txt
├── fsl-image-mfgtool-initramfs-imx_mfgtools.cpio.gz.u-boot
├── GPLv2
├── Image-imx8_all.bin
├── Image-imx8mqevk.bin
├── imx8mq-evk-ak4497.dtb
├── imx8mq-evk-audio-tdm.dtb
├── imx8mq-evk-dcss-adv7535.dtb
├── imx8mq-evk-dcss-rm67191.dtb
├── imx8mq-evk-dp.dtb
├── imx8mq-evk.dtb
├── imx8mq-evk-dual-display.dtb
├── imx8mq-evk-inmate.dtb
├── imx8mq-evk-lcdif-adv7535.dtb
├── imx8mq-evk-lcdif-rm67191.dtb
├── imx8mq-evk-pcie1-m2.dtb
├── imx8mq-evk-pcie-ep.dtb
├── imx8mq-evk-pdm.dtb
├── imx8mq-evk-root.dtb
├── imx8mq-evk-rpmsg.dtb
├── imx8mq-evk-usdhc2-m2.dtb
├── imx8mq-evk-usd-wifi.dtb
├── imx-boot-imx8mqevk-sd.bin-flash_dp_evk
├── imx-boot-imx8mqevk-sd.bin-flash_evk
├── imx-boot-imx8mqevk-sd.bin-flash_evk_no_hdmi
├── imx-image-full-imx8mqevk.manifest
├── imx-image-full-imx8mqevk.tar.bz2
├── imx-image-full-imx8mqevk.wic
├── imx-image-multimedia-imx8mqevk.manifest
├── imx-image-multimedia-imx8mqevk.tar.bz2
├── imx-image-multimedia-imx8mqevk.wic
├── imx_mcore_demos
├── QUALCOMM_ATHEROS_LICENSE_AGREEMENT.pdf
├── README.uuu
├── samples
├── SCR-5.4.47_2.2.0.txt
└── uuu.auto
```

4.2 Building the image from source

See ref.[15].

4.3 Flashing the image to eMMC

The *Universal Update Utility* (*UUU*) runs on a Windows or Linux OS host and is used to download images to different devices on an i.MX board.

Note: *To flash the image on eMMC, change the switch settings to set the boot mode on serial download mode. See Table 2 "Boot mode switch settings".*

Download *UUU*

Download the latest version of UUU from https://github.com/NXPmicro/mfgtools/releases.

Copy *UUU*

Copy the uuu binary into the */usr/bin/* directory and change the permission to executable.

```
ubuntu@ubuntu-desktop:/# sudo chmod +x /usr/bin/uuu
```

UUU usage

```
ubuntu@ubuntu-desktop:/# sudo uuu -b emmc_all <bootloader> <image>
```

Using the pre-built image (Section 4.1 "Using the pre-built image"), the following command burns the complete image into eMMC:

```
ubuntu@ubuntu-desktop:/# sudo uuu <release package>.zip
```

Flash the image

Use the following format for the command:

```
ubuntu@ubuntu-desktop:/# sudo uuu -b emmc_all <bootloader> <image>
```

For example:

```
ubuntu@ubuntu-desktop:/# sudo uuu -b emmc_all imx-boot-imx8mqevk-sd.bin-flash_evk imx-
image-full-imx8mqevk.wic
```

Follow these instructions to flash i.MX 8M Quad EVK board:

- Connect a USB cable from a computer to the USB OTG/TYPE C port on the board
- Connect a USB cable from the OTG-to-UART port to the computer for console output
- Open a Terminal emulator program (for example minicom)
- Set the boot pin to serial download mode (see Section 2.3 "i.MX 8M Quad switch settings")

4.4 Booting from eMMC

To boot the i.MX 8M Quad from eMMC, set the boot switch per the settings given in Section 2.3 "i.MX 8M Quad switch settings".

4.4.1 Serial console setup

Follow these steps to setup the serial console and access the i.MX 8M Quad device terminal:

- Open the serial console and login into the device

```
ubuntu@ubuntu-desktop:/# sudo minicom -s -D /dev/ttyUSBx
```

- *ttyUSB0* or *ttyUSB01* are the serial devices. The minicom setup configuration is given below:

```
A -    Serial Device      : /dev/ttyUSBx
E -    Bps/Par/Bits       : 115200 8N1
F - Hardware Flow Control : No
G - Software Flow Control : No
```

- Save and exit.

4.4.2 Enabling SDIO on M.2 connector

Note: *This section only applies to modules based on M.2 connector with SDIO interface for Wi-Fi. The SDIO on M.2 DTB is necessary to enable the SDIO interface on the M.2 connector. By default the SDIO support is enabled for the SD Card slot connector. The hardware rework instructions are given in Section 3.7.7 "i.MX 8M Quad rework for SDIO support on M.2".*

The procedure below sets as default the DTB file that enables the SDIO interface on the M.2 connector.

- **Step 1** - Power-on the board and hit any key within 3 seconds to halt u-boot from booting and enter the u-boot console
- **Step 2** - Set `fdtfile` with the DTB file to enable SDIO interface M.2 connector on bottom of the board:

```
u-boot=> setenv fdtfile imx8mq-evk-usdhc2-m2.dtb
u-boot=> saveenv
```

Command output example:

```
Saving Environment to MMC... Writing to MMC(0)... OK
```

- **Step 3** - Reset the board

```
u-boot=> reset
```

Command output example:

```
resetting ...
```

4.4.3 Enabling PCIe on M.2 (88W9098-based Murata module LBEE6ZZ1)

The procedure below sets as default the DTB file that enables the PCIe interface on the M.2 connector.

- **Step 1** - Power-on the board and hit any key within 3 seconds to halt u-boot from booting and enter the u-boot console
- **Step 2** - Set `fdtfile` with the DTB file to enable PCIe interface M.2 connector on bottom of the board:

```
u-boot=> setenv fdtfile imx8mq-evk-pcie1-m2.dtb
u-boot=> saveenv
```

Command output example:

```
Saving Environment to MMC... Writing to MMC(0)... OK
```

- **Step 3** - Reset the board

```
u-boot=> reset
```

Command output example:

```
resetting ...
```

4.4.4 Linux OS login

The default login username for the i.MX Linux OS is *root*. There is no password.

4.4.5 Software package release version information

The software package release version information is provided in the release notes.

5 Bring-up of Wi-Fi

This section describes the bring-up steps for the Wi-Fi interfaces of the NXP-based wireless module connected to i.MX 8M Quad EVK.

5.1 Bring-up of 88W8987-based wireless module (AW-CM358MA)

Follow these instructions to load the driver modules and bring up the 88W8987-based wireless module

- Load the modules in the kernel

```
root@imx8mqevk:~# modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Verify the kernel debug messages in the command output

```
[ 2896.073550] wlan: Loading MWLAN driver
[ 2896.178399] vendor=0x02DF device=0x9149 class=0 function=1
[ 2896.183972] Attach moal handle ops, card interface type:0x105
[ 2896.189935] SD8987: init module param from usr cfg
[ 2896.194807] card_type: SD8987, config block: 0
[ 2896.199291] cfg80211_wext=0xf
[ 2896.202274] wfd_name=p2p
[ 2896.204855] max_vir_bss=1
[ 2896.207509] cal_data_cfg=none
[ 2896.210494] drv_mode = 7
[ 2896.213054] ps_mode = 2
[ 2896.215532] auto_ds = 2
[ 2896.218007] fw_name=nxp/sdiouart8987_combo_v0.bin
[ 2896.225747] SDIO: max_segs=128 max_seg_size=65535
[ 2896.230505] rx_work=1 cpu_num=4
[ 2896.233724] Attach mlan adapter operations.card_type is 0x105.
[ 2896.240142] wlan: Enable TX SG mode
[ 2896.243666] wlan: Enable RX SG mode
[ 2896.251307] Request firmware: nxp/sdiouart8987_combo_v0.bin
[ 2896.667066] Wlan: FW download over, firmwarelen=526996 downloaded 526996
[ 2897.601246] WLAN FW is active
[ 2897.604233] on_time is 2897601629500
[ 2897.638511] fw_cap_info=0x181c3f03, dev_cap_mask=0xffffffff
[ 2897.644138] max_p2p_conn = 8, max_sta_conn = 8
[ 2897.671116] wlan: version = SD8987---16.92.10.p207-MXM4X16186.p6-GPL-(FP92)
[ 2897.682184] wlan: Driver loaded successfully
```

- Verify the Wi-Fi interfaces

```
root@imx8mqevk:~# ifconfig -a
```

Command output example:

```
eth0  Link encap:Ethernet  HWaddr 00:04:9f:06:77:40
      UP BROADCAST MULTICAST DYNAMIC  MTU:1500  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:65536  Metric:1
      RX packets:90 errors:0 dropped:0 overruns:0 frame:0
      TX packets:90 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:6708 (6.5 KiB)  TX bytes:6708 (6.5 KiB)
mlan0 Link encap:Ethernet  HWaddr 00:50:43:24:83:c4
      BROADCAST MULTICAST  MTU:1500  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
p2p0  Link encap:Ethernet  HWaddr 02:50:43:24:83:c4
      BROADCAST MULTICAST  MTU:1500  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
uap0  Link encap:Ethernet  HWaddr 00:50:43:24:84:c4
      BROADCAST MULTICAST  MTU:1500  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- Get the Wi-Fi driver version

```
root@imx8mqevk:~# cat /proc/mwlan/adapter0/uap0/info
```

Command output example

```
driver_name = "uap"
driver_version = SD8987---w8987o-V0, MXXX, FP92, 16.92.10.p206-MXM5X16186.p6-GPL-(FP92)
interface_name="uap0"firmware_major_version=16.92.10
media_state="Disconnected"mac_address="70:66:55:26:8b:6b"
```

5.2 Bring-up of 88W8997-based wireless module (AW-CM276MA-PUR)

Follow these instructions to load the driver modules and bring up the 88W8987-based wireless module

- Load the modules in the kernel

```
root@imx8mqevk:~# modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Verify the kernel debug messages in the command output

```
[  139.999861] wlan: Loading MWLAN driver
[  140.004586] wlan_pcie 0000:01:00.0: enabling device (0000 -> 0002)
[  140.010904] Attach moal handle ops, card interface type: 0x204
[  140.016763] No module param cfg file specified
[  140.021239] rx_work=1 cpu_num=4
[  140.024418] Attach mlan adapter operations.card_type is 0x204.
[  140.034659] Request firmware: mrvl/pcieuart8997_combo_v4.bin
[  140.993340] FW download over, size 627620 bytes
[  141.859194] WLAN FW is active
[  141.862165] on_time is 141859539976
[  142.039695] fw_cap_info=0x18fcffa3, dev_cap_mask=0xffffffff
[  142.045296] max_p2p_conn = 8, max_sta_conn = 8
[  142.070295] wlan: version = PCIE8997-16.68.10.p16-MXM5X16215-GPL-(FP92)
[  142.077289] wlan: Driver loaded successfully
```

- Verify the Wi-Fi interfaces

```
root@imx8mqevk:~# ifconfig -a
```

Command output example:

```
eth0 Link encap:Ethernet  HWaddr 00:04:9f:06:c5:a5
      inet addr:169.254.130.90  Bcast:169.254.255.255  Mask:255.255.0.0
      inet6 addr: fe80::204:9fff:fe06:c5a5/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:89 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B)  TX bytes:30164 (29.4 KiB)

lo   Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:65536  Metric:1
      RX packets:458 errors:0 dropped:0 overruns:0 frame:0
      TX packets:458 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:29156 (28.4 KiB)  TX bytes:29156 (28.4 KiB)

mlan0 Link encap:Ethernet  HWaddr 70:66:55:9b:3a:95
      BROADCAST MULTICAST  MTU:1500  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

p2p0 Link encap:Ethernet  HWaddr 72:66:55:9b:3a:95
      BROADCAST MULTICAST  MTU:1500  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

uap0  Link encap:Ethernet  HWaddr 70:66:55:9b:3b:95
      BROADCAST MULTICAST  MTU:1500  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- Get the Wi-Fi driver version

```
root@imx8mqevk:~# cat /proc/mwlan/adapter0/uap0/info
```

Command output example

```
driver_name = "uap"
driver_version = PCIE8997-w8997o-V4, RF878X, FP68_LINUX, 16.68.10.p16-MXM5X16215-GPL-
(FP92)
interface_name="uap0"
firmware_major_version=16.68.10
media_state="Disconnected"
mac_address="70:66:55:9b:3b:95"
```

5.3 Bring-up of 88W8997-based AzureWave module (AW-CM276MA-SUR and AW-CM276-uSD)

Follow these instructions to load the driver modules and bring up the 88W8997-based wireless module.

- Load the modules in the kernel

```
root@imx8mqevk:~# modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Verify the kernel debug messages in the command output.

```
[   49.395686] wlan: Loading MWLAN driver
[   49.401313] vendor=0x02DF device=0x9141 class=0 function=1
[   49.409809] Attach moal handle ops, card interface type: 0x104
[   49.418621] SD8997: init module param from usr cfg
[   49.423457] card_type: SD8997, config block: 0
[   49.427951] ps_mode = 2
[   49.430397] auto_ds = 2
[   49.432860] cfg80211_wext=0xf
[   49.435843] cal_data_cfg=none
[   49.438811] fw_name=nxp/sdiouart8997_combo_v4.bin
[   49.443540] drv_mode = 7
[   49.446073] max_vir_bss=1
[   49.448714] wfd_name=p2p
[   49.451306] SDIO: max_segs=128 max_seg_size=65535
[   49.456036] rx_work=1 cpu_num=4
[   49.459243] Attach mlan adapter operations.card_type is 0x104.
[   49.466060] wlan: Enable TX SG mode
[   49.469570] wlan: Enable RX SG mode
[   49.479279] Request firmware: nxp/sdiouart8997_combo_v4.bin
[   50.201775] Wlan: FW download over, firmwarelen=543964 downloaded 543964
[   51.071805] WLAN FW is active
[   51.074778] on_time is 51071607569
[   51.118478] fw_cap_info=0x181c3fa3, dev_cap_mask=0xffffffff
[   51.124075] max_p2p_conn = 8, max_sta_conn = 8
[   51.148460] wlan: version = SD8997----16.92.10.p216-MM5X16247.p1-GPL-(FP92)
[   51.156963] usbcore: registered new interface driver usbxxx
[   51.162772] wlan: Driver loaded successfully
```

- Verify the Wi-Fi interfaces

```
root@imx8mqevk:~# ifconfig -a
```

Command output example:

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.0.103  netmask 255.255.255.0  broadcast 192.168.0.255
    inet6 fe80::204:9fff:fe06:c791  prefixlen 64  scopeid 0x20<link>
    ether 00:04:9f:06:c7:91  txqueuelen 1000  (Ethernet)
    RX packets 338  bytes 29157 (28.4 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 387  bytes 34568 (33.7 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop  txqueuelen 1000  (Local Loopback)
    RX packets 732  bytes 53888 (52.6 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 732  bytes 53888 (52.6 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mlan0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
    ether 00:e9:3a:0d:c2:59  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

p2p0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
    ether 02:e9:3a:0d:c2:59  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

uap0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
    ether 00:e9:3a:0d:c3:59  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- Get the Wi-Fi driver version

```
root@imx8mqevk:~# cat /proc/mwlan/adapter0/uap0/info
```

Command output example:

```
driver_name = "uap"
driver_version = SD8997----w8997o-V4, MXXX, FP92, 16.92.10.p216-MM5X16247.p1-GPL-(FP92)
interface_name="uap0"
firmware_major_version=16.92.10
media_state="Disconnected"
mac_address="00:e9:3a:0d:c3:59"
```

5.4 Bring-up of 88W9098-based Murata module LBEE6ZZ1

Follow these instructions to load the driver modules and bring up the 88W9098-based wireless module.

- Load the modules in the kernel

```
root@imx8mqevk:~# modprobe moal mod_para=nxp/wifi_mod_para.conf
```

- Verify the kernel debug message in the command output

```
[ 1934.903485] wlan: Loading MWLAN driver
[ 1934.908682] usbcore: registered new interface driver usbxxx
[ 1934.914517] wlan_pcie 0001:01:00.0: enabling device (0000 -> 0002)
[ 1934.920862] Attach moal handle ops, card interface type: 0x206
[ 1934.926883] PCIE9098: init module param from usr cfg
[ 1934.931915] card_type: PCIE9098, config block: 0
[ 1934.936630] cfg80211_wext=0xf
[ 1934.939652] wfd_name=p2p
[ 1934.942186] max_vir_bss=1
[ 1934.944828] cal_data_cfg=none
[ 1934.947809] drv_mode = 7
[ 1934.950342] ps_mode = 2
[ 1934.952801] auto_ds = 2
[ 1934.955264] fw_name=nxp/pcieuart9098_combo_v1.bin
[ 1934.959993] rx_work=1 cpu_num=4
[ 1934.963148] Attach mlan adapter operations.card_type is 0x206.
[ 1934.971277] Request firmware: nxp/pcieuart9098_combo_v1.bin
[ 1935.403742] FW download over, size 617212 bytes
[ 1936.271780] WLAN FW is active
[ 1936.274751] on_time is 1936271521056
[ 1937.621201] fw_cap_info=0x81c3fa3, dev_cap_mask=0xffffffff
[ 1937.626749] max_p2p_conn = 8, max_sta_conn = 64
[ 1937.641095] wlan: version = PCIE9098--17.92.1.p55-MM5X17247-GPL-(FP92)
[ 1937.650087] wlan_pcie 0001:01:00.1: enabling device (0000 -> 0002)
[ 1937.657779] Attach moal handle ops, card interface type: 0x206
[ 1937.665636] PCIE9098: init module param from usr cfg
[ 1937.670841] Configuration block, fallback processing
[ 1937.682944] Configuration fallback to, card_type: 0x206, blk_id: 0x0
[ 1937.689503] rx_work=1 cpu_num=4
[ 1937.694195] Attach mlan adapter operations.card_type is 0x206.
[ 1937.707333] Request firmware: nxp/pcieuart9098_combo_v1.bin
[ 1937.713705] WLAN FW already running! Skip FW download
[ 1937.718879] WLAN FW is active
[ 1937.721903] on_time is 1937718673474
[ 1937.737820] fw_cap_info=0x81c3fa3, dev_cap_mask=0xffffffff
[ 1937.743389] max_p2p_conn = 8, max_sta_conn = 64
[ 1937.757214]wlan: version = PCIE9098--17.92.1.p55-MM5X17247-GPL-(FP92)
[ 1937.765249] wlan: Driver loaded successfully
```

- Verify the Wi-Fi interfaces

```
root@imx8mqevk:~# ifconfig -a
```

Command output example

```
eth0: flags=-28669<UP,BROADCAST,MULTICAST,DYNAMIC>  mtu 1500
    ether 00:04:9f:06:c5:a5  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop  txqueuelen 1000  (Local Loopback)
    RX packets 380  bytes 25376 (24.7 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 380  bytes 25376 (24.7 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mlan0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
    ether 2c:4c:c6:f4:67:b0  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mmlan0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
    ether 2c:4c:c6:f4:67:b1  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

muap0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
    ether 2c:4c:c6:f4:68:b1  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

p2p0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
    ether 2e:4c:c6:f4:67:b0  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

p2p1: flags=4098<BROADCAST,MULTICAST>  mtu 1500
    ether 2e:4c:c6:f4:67:b1  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

uap0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
    ether 2c:4c:c6:f4:68:b0  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Note: *NXP 88W9098-based wireless modules support Wi-Fi dual radios with dual MACs and dual independent Wi-Fi interfaces created by the driver for each mode of operation.*

- Get the Wi-Fi driver version

```
root@imx8mqevk:~# cat /proc/mwlan/adapter0/uap0/info
```

Command output example

```
driver_name = "uap"
driver_version = PCIE9098--w9098o-V1, MXXX, FP92, 17.92.1.p55-MM5X17247-GPL-(FP92)
interface_name="uap0"
firmware_major_version=17.92.1
media_state="Disconnected"
mac_address="2c:4c:c6:f4:68:b0"
```

5.5 Bring-up of 88W9098-based Murata module LBEE5ZZ1XL

Follow these instructions to load the driver modules and bring up the 88W9098-based wireless module.

- Load the modules in the kernel

```
root@imx8mqevk:~# modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Verify the kernel debug messages in the command output:

```
[ 484.265296] mlan: loading out-of-tree module taints kernel.
[ 484.290981] wlan: Loading MWLAN driver
[ 484.295144] wlan: Driver loaded successfully
[ 484.295209] wlan: Register to Bus Driver...
[ 484.303818] vendor=0x02DF device=0x914D class=0 function=1
[ 484.309393] Attach moal handle ops, card interface type: 0x106
[ 484.317132] SD9098: init module param from usr cfg
[ 484.321997] card_type: SD9098, config block: 0
[ 484.326463] cfg80211_wext=0xf
[ 484.329444] max_vir_bss=1
[ 484.332065] cal_data_cfg=none
[ 484.335058] ps_mode = 1
[ 484.337524] auto_ds = 1
[ 484.339972] host_mlme=enable
[ 484.342874] mac_addr=00:50:43:20:12:34
[ 484.346636] fw_name=nxp/sdiouart9098_combo_v1.bin
[ 484.351365] SDIO: max_segs=128 max_seg_size=65535
[ 484.356090] rx_work=1 cpu_num=4
[ 484.359268] Attach mlan adapter operations.card_type is 0x106.
[ 484.365447] wlan: Enable TX SG mode
[ 484.368958] wlan: Enable RX SG mode
[ 484.376480] Request firmware: nxp/sdiouart9098_combo_v1.bin
[ 484.850401] Wlan: FW download over, firmwarelen=631336 downloaded 631336
[ 485.180998] WLAN FW is active
[ 485.198386] fw_cap_info=0x81c3fa3, dev_cap_mask=0xffffffff
[ 485.203898] max_p2p_conn = 8, max_sta_conn = 64
[ 485.224923] wlan: version = SD9098----17.92.1.p91-MM5X17293-GPL-(FP92)
[ 485.231824] Set REG 0x90002328: 0x3000 slew_rate=3
[ 485.237505] vendor=0x02DF device=0x914E class=0 function=2
[ 485.243499] Attach moal handle ops, card interface type: 0x106
[ 485.249670] SD9098: init module param from usr cfg
[ 485.254603] card_type: SD9098, config block: 1
[ 485.259113] cfg80211_wext=0xf
[ 485.262202] max_vir_bss=1
[ 485.265900] cal_data_cfg=none
[ 485.269001] ps_mode = 1
[ 485.271460] auto_ds = 1
[ 485.273944] host_mlme=enable
[ 485.276856] mac_addr=00:50:43:20:52:56
[ 485.280628] fw_name=nxp/sdiouart9098_combo_v1.bin
[ 485.285388] SDIO: max_segs=128 max_seg_size=65535
[ 485.290179] rx_work=1 cpu_num=4
[ 485.293501] Attach mlan adapter operations.card_type is 0x106.
[ 485.299954] wlan: Enable TX SG mode
[ 485.303542] wlan: Enable RX SG mode
[ 485.308758] Request firmware: nxp/sdiouart9098_combo_v1.bin
[ 485.315433] WLAN FW already running! Skip FW download
[ 485.320623] WLAN FW is active
[ 485.324163] fw_cap_info=0x81c3fa3, dev_cap_mask=0xffffffff
[ 485.329734] max_p2p_conn = 8, max_sta_conn = 64
[ 485.344597] wlan: version = SD9098----17.92.1.p91-MM5X17293-GPL-(FP92)
[ 485.351504] Set REG 0x90002328: 0x3000 slew_rate=3
[ 485.357184] wlan: Register to Bus Driver Done
```

- Verify the Wi-Fi interfaces

```
root@imx8mqevk:~# ifconfig -a
```

Command output example:

```
eth0: flags=-28605<UP,BROADCAST,RUNNING,MULTICAST,DYNAMIC>  mtu 1500
                inet 10.10.0.120  netmask 255.255.254.0  broadcast 10.10.1.255
                inet6 fe80::204:9fff:fe06:c5a5  prefixlen 64  scopeid 0x20<link>
                ether 00:04:9f:06:c5:a5  txqueuelen 1000  (Ethernet)
                RX packets 1706  bytes 247304 (241.5 KiB)
                RX errors 0  dropped 0  overruns 0  frame 0
                TX packets 259  bytes 38694 (37.7 KiB)
                TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
                inet 127.0.0.1  netmask 255.0.0.0
                inet6 ::1  prefixlen 128  scopeid 0x10<host>
                loop  txqueuelen 1000  (Local Loopback)
                RX packets 180  bytes 15992 (15.6 KiB)
                RX errors 0  dropped 0  overruns 0  frame 0
                TX packets 180  bytes 15992 (15.6 KiB)
                TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
mlan0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
                ether 00:50:43:20:12:34  txqueuelen 1000  (Ethernet)
                RX packets 0  bytes 0 (0.0 B)
                RX errors 0  dropped 0  overruns 0  frame 0
                TX packets 0  bytes 0 (0.0 B)
                TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
mmlan0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
                ether 00:50:43:20:52:56  txqueuelen 1000  (Ethernet)
                RX packets 0  bytes 0 (0.0 B)
                RX errors 0  dropped 0  overruns 0  frame 0
                TX packets 0  bytes 0 (0.0 B)
                TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
muap0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
                ether 00:50:43:20:53:56  txqueuelen 1000  (Ethernet)
                RX packets 0  bytes 0 (0.0 B)
                RX errors 0  dropped 0  overruns 0  frame 0
                TX packets 0  bytes 0 (0.0 B)
                TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
mwfd0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
                ether 02:50:43:20:52:56  txqueuelen 1000  (Ethernet)
                RX packets 0  bytes 0 (0.0 B)
                RX errors 0  dropped 0  overruns 0  frame 0
                TX packets 0  bytes 0 (0.0 B)
                TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
uap0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
               ether 00:50:43:20:13:34  txqueuelen 1000  (Ethernet)
               RX packets 0  bytes 0 (0.0 B)
               RX errors 0  dropped 0  overruns 0  frame 0
               TX packets 0  bytes 0 (0.0 B)
               TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
wfd0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
               ether 02:50:43:20:12:34  txqueuelen 1000  (Ethernet)
               RX packets 0  bytes 0 (0.0 B)
               RX errors 0  dropped 0  overruns 0  frame 0
               TX packets 0  bytes 0 (0.0 B)
               TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Note: *NXP 88W9098-based wireless modules support dual Wi-Fi radios. When the driver is loaded, it creates dual independent Wi-Fi interfaces for each mode of operation (STA, Access Point, and P2P).*

- Get the Wi-Fi driver version

```
root@imx8mqevk:~# cat /proc/mwlan/adapter0/uap0/info
```

Command output example:

```
driver_name = "uap"
driver_version = SD9098----w9098o-V1, MXXX, FP92, 17.92.1.p91-MM5X17293-GPL-(FP92)
interface_name="uap0"
firmware_major_version=17.92.1
media_state="Disconnected"
mac_address="00:50:43:20:13:34"
num_tx_bytes = 0
num_rx_bytes = 0
```

5.6 Bring-up of IW416-based AzureWave module AW-AM510MA

Follow these instructions to load the driver modules and bring up the IW416-based wireless module.

- Load the modules in the kernel

```
root@imx8mqevk:~# modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Verify the kernel debug messages in the command output:

```
[  558.903379] wlan: Loading MWLAN driver
[  559.018905] vendor=0x02DF device=0x9159 class=0 function=1
[  559.024535] Attach moal handle ops, card interface type: 0x108
[  559.030552] No module param cfg file specified
[  559.035015] SDIO: max_segs=128 max_seg_size=65535
[  559.039733] rx_work=1 cpu_num=4
[  559.042909] Attach mlan adapter operations.card_type is 0x108.
[  559.049601] wlan: Enable TX SG mode
[  559.053110] wlan: Enable RX SG mode
[  559.060287] Request firmware: nxp/sdiouart8978_combo_v0.bin
[  559.444220] Wlan: FW download over, firmwarelen=481164 downloaded 481164
[  560.855398] WLAN FW is active
[  560.858381] on_time is 560854191159
[  560.884715] fw_cap_info=0x181c0f03, dev_cap_mask=0xffffffff
[  560.890320] max_p2p_conn = 8, max_sta_conn = 8
[  560.915183] wlan: version = SDIW416---16.92.10.p233-MM5X16266.p1-GPL-(FP92)
[  560.924100] usbcore: registered new interface driver usbxxx
[  560.931152] wlan: Driver loaded successfully
```

- Verify the Wi-Fi interfaces

```
root@imx8mqevk:~# ifconfig -a
```

Command output example:

```
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
      ether 00:04:9f:06:c5:a5  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
      inet 127.0.0.1  netmask 255.0.0.0
      inet6 ::1  prefixlen 128  scopeid 0x10<host>
      loop  txqueuelen 1000  (Local Loopback)
      RX packets 524  bytes 34160 (33.3 KiB)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 524  bytes 34160 (33.3 KiB)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mlan0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      ether 48:e7:da:3c:79:d9  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

p2p0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      ether 4a:e7:da:3c:79:d9  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

uap0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      ether 48:e7:da:3c:7a:d9  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- Get the Wi-Fi driver version

```
root@imx8mqevk:~# cat /proc/mwlan/adapter0/uap0/info
```

Command output example:

```
driver_name = "uap"
driver_version = SDIW416---w8978o-V0, MXXX, FP92, 16.92.10.p233-MM5X16266.p1-GPL-(FP92)
interface_name="uap0"
firmware_major_version=16.92.10
media_state="Disconnected"
mac_address="48:e7:da:3c:7a:d9"
```

5.7 Bring-up of 88W8801-based Murata module LBWA0ZZ2DS

Follow these instructions to load the driver modules and bring up the 88W8801-based wireless module.

- Load the modules in the kernel

```
root@imx8mqevk:~# modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Verify the kernel debug messages in the `dmesg` command output:

```
[   26.576621] Attach moal handle ops, card interface type: 0x10a
[   26.583160] SD8801: init module param from usr cfg
[   26.587988] card_type: SD8801, config block: 0
[   26.592458] cfg80211_wext=0xf
[   26.595441] wfd_name=p2p
[   26.597987] max_vir_bss=1
[   26.600608] cal_data_cfg=none
[   26.603609] drv_mode = 7
[   26.606160] ps_mode = 2
[   26.608605] auto_ds = 2
[   26.611068] fw_name=nxp/sd8801_uapsta.bin
[   26.615102] SDIO: max_segs=128 max_seg_size=65535
[   26.619820] rx_work=1 cpu_num=4
[   26.622988] Attach mlan adapter operations.card_type is 0x10a.
[   26.629102] wlan: Enable TX SG mode
[   26.632613] wlan: Enable RX SG mode
[   26.642042] Request firmware: nxp/sd8801_uapsta.bin
[   27.021740] Wlan: FW download over, firmwarelen=259052 downloaded 259052
[   27.242541] WLAN FW is active
[   27.245511] on_time is 27243589010
[   27.267053] fw_cap_info=0xba3, dev_cap_mask=0xffffffff
[   27.410250] wlan: version = SD8801----14.92.36.p171-MM5X14283.p2-GPL-(FP92)
[   27.418406] usbcore: registered new interface driver usbxxx
[   27.424115] wlan: Register to Bus Driver Done
```

- Verify the Wi-Fi interfaces

```
root@imx8mqevk:~# ifconfig -a
```

Command output example:

```
eth0: flags=-28669<UP,BROADCAST,MULTICAST,DYNAMIC>  mtu 1500
        ether 00:04:9f:06:c5:a5  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 140  bytes 10736 (10.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 140  bytes 10736 (10.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mlan0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether c4:ac:59:a4:51:47  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

p2p0:  flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether c6:ac:59:a4:51:47  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

uap0:   flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether c4:ac:59:a4:52:47  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- Get the Wi-Fi driver version

```
root@imx8mqevk:~# cat /proc/mwlan/adapter0/uap0/info
```

Command output example:

```
driver_name = "uap"
driver_version = SD8801----w8801-B0, RF878X, FP92, 14.92.36.p171-MM5X14283.p2-GPL-(FP92)
interface_name="uap0"
firmware_major_version=14.92.36
media_state="Disconnected"
mac_address="c4:ac:59:a4:52:47"
```

5.8 Bring-up of IW612-based Murata module LBES5PL2EL

Follow these instructions to load the driver modules and bring up the IW612-based wireless module.

- Load the modules in the kernel

```
root@imx8mqevk:~# modprobe moal mod_para=nxp/wifi_mod_para.conf
```

- Verify the kernel debug messages in the dmesg command output:

```
[   23.671516] mlan: loading out-of-tree module taints kernel.
[   23.700968] wlan: Loading MWLAN driver
[   23.705032] wlan: Register to Bus Driver...
[   23.709451] vendor=0x0471 device=0x0205 class=0 function=1
[   23.715055] Attach moal handle ops, card interface type: 0x109
[   23.720904] rps set to 0 from module param
[   23.725021] No module param cfg file specified
[   23.729483] SDIO: max_segs=128 max_seg_size=65535
[   23.734199] rx_work=1 cpu_num=4
[   23.737488] Attach mlan adapter operations.card_type is 0x109.
[   23.743730] wlan: Enable TX SG mode
[   23.747243] wlan: Enable RX SG mode
[   23.755084] Request firmware: nxp/sduart_nw61x_v1.bin.se
[   24.045554] Wlan: FW download over, firmwarelen=817072 downloaded 817072
[   25.408086] WLAN FW is active
[   25.411089] on_time is 25408095257
[   25.424498] fw_cap_info=0x487cff03, dev_cap_mask=0xffffffff
[   25.430102] uuid: 5bf547832610530a8aae04a12655ba48
[   25.434904] max_p2p_conn = 8, max_sta_conn = 16
[   25.457626] Register NXP 802.11 Adapter mlan0
[   25.462131] wlan: uap%d set max_mtu 2000
[   25.468185] Register NXP 802.11 Adapter uap0
[   25.476310] Register NXP 802.11 Adapter wfd0
[   25.480752] wlan: version = SDIW612---18.99.1.p154.40-MM5X18368.p2-GPL-(FP92)
[   25.489216] wlan: Register to Bus Driver Done
[   25.493715] wlan: Driver loaded successfully
```

- Verify the Wi-Fi interface

```
root@imx8mqevk:~# ifconfig -a
```

Command output example:

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 192.168.9.32  netmask 255.255.255.0  broadcast 192.168.9.255
      inet6 fe80::204:9fff:fe07:5777  prefixlen 64  scopeid 0x20<link>
      ether 00:04:9f:07:57:77  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 66  bytes 14858 (14.5 KiB)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
      inet 127.0.0.1  netmask 255.0.0.0
      inet6 ::1  prefixlen 128  scopeid 0x10<host>
      loop  txqueuelen 1000  (Local Loopback)
      RX packets 204  bytes 14640 (14.2 KiB)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 204  bytes 14640 (14.2 KiB)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mlan0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      ether d0:17:69:ee:7e:2a  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

uap0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      ether d2:17:69:ee:7f:2a  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wfd0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      ether d2:17:69:ee:7e:2a  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- Get the driver version.

```
root@imx8mqevk:~# cat /proc/mwlan/adapter0/uap0/info
```

Example of command output:

```
driver_name = "uap"
driver_version = SDIW612---w9177o-V1, RF878X, FP99, 18.99.1.p154.40-MM5X18368.p2-GPL-
(FP92) interface_name="uap0"
firmware_major_version=18.99.1
```

5.8.1 Bring up the Wi-Fi interface

Use the following steps to bring up the Wi-Fi interfaces

- Invoke the command to initialize *mlan0* interface

```
root@imx8mqevk:~# ifconfig mlan0 up
root@imx8mqevk:~# ifconfig mlan0
```

Command output example:

```
mlan0   Link encap:Ethernet  HWaddr 00:50:43:24:83:c4
        UP BROADCAST MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- Invoke the command to initialize the *uap0* interface

```
root@imx8mqevk:~# ifconfig uap0 up
root@imx8mqevk:~# ifconfig uap0
```

Command output example:

```
uap0    Link encap:Ethernet  HWaddr 00:50:43:24:84:c4
        UP BROADCAST MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

5.9 Bring-up of IW610-based Murata module LBES0ZZ2LL

Step 1 – Load the driver modules.

```
root@imx8mqevk:~# modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Step 2 – Verify the kernel debug messages in the `dmesg` command output.

```
[ 2515.181788] wlan: Loading MWLAN driver
[ 2515.185959] wlan: Register to Bus Driver...
[ 2515.190353] vendor=0x0471 device=0x0215 class=0 function=1
[ 2515.196026] Attach moal handle ops, card interface type: 0x10d
[ 2515.201913] rps set to 0 from module param
[ 2515.206053] No module param cfg file specified
[ 2515.210528] SDIO: max_segs=128 max_seg_size=65535
[ 2515.215245] rx_work=1 cpu_num=4
[ 2515.218400] Enable moal_recv_amsdu_packet
[ 2515.222447] Attach mlan adapter operations.card_type is 0x10d.
[ 2515.228625] wlan: Enable TX SG mode
[ 2515.232142] wlan: Enable RX SG mode
[ 2515.236176] Request firmware: nxp/sduart_iw610.bin.se
[ 2515.241885] Wakeup device...
[ 2515.246270] WLAN FW already running! Skip FW download
[ 2515.251412] Wakeup device...
[ 2515.254455] WLAN FW is active
[ 2515.257445] on_time is 2515549379117
[ 2515.416064] fw_cap_info=0x487cbf03 fw_cap_ext=0x10b85
[ 2515.421124] uuid: 4640e5b0452b5f5bbdb50e559ca4dd0f
[ 2515.425929] uap fw ver=2.0
[ 2515.428645] max_p2p_conn = 8, max_sta_conn = 8
[ 2515.433096] wlan_set_regiontable: 2.4G 0x10
[ 2515.437289] wlan_set_regiontable: 5G 0x10
[ 2515.441381] Enable Beamforming
[ 2517.047668] wlan: version = SDIW610---18.99.5.p39-MM6X18505.p4-(FP92)
[ 2517.145579] wlan: Register to Bus Driver Done
[ 2517.149965] wlan: Driver loaded successfully
```

Step 3 – Verify the Wi-Fi interface.

```
root@imx8mqevk:~# ifconfig -a
```

Command output example:

```
eth0: flags=-28605<UP,BROADCAST,RUNNING,MULTICAST,DYNAMIC>  mtu 1500
      inet 172.29.122.114  netmask 255.255.255.0  broadcast 172.29.122.255
      inet6 fe80::204:9fff:fe06:c5a5  prefixlen 64  scopeid 0x20<link>
      ether 00:04:9f:06:c5:a5  txqueuelen 1000  (Ethernet)
      RX packets 4494  bytes 989486 (966.2 KiB)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 454  bytes 48142 (47.0 KiB)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
      inet 127.0.0.1  netmask 255.0.0.0
      inet6 ::1  prefixlen 128  scopeid 0x10<host>
      loop  txqueuelen 1000  (Local Loopback)
      RX packets 98  bytes 8552 (8.3 KiB)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 98  bytes 8552 (8.3 KiB)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mlan0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      ether 78:f5:05:7b:cc:4e  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

uap0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      ether 7a:f5:05:7b:cd:4e  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wfd0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      ether 7a:f5:05:7b:cc:4e  txqueuelen 1000  (Ethernet)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Step 4 – Get the driver version.

```
root@imx8mqevk:~# cat /proc/mwlan/adapter0/uap0/info
```

Example of command output:

```
driver_name = "uap"
driver_version = SDIW610---w9177o-V1, RF878X, FP99, 18.99.5.p39-MM6X18505.p4-GPL-
(FP92) interface_name="uap0"
firmware_major_version=18.99.5
```

5.9.1 Bring-up of LBES0ZZ2LL module using USB interface

Note: *Only IW610-based Murata 2LL module supports USB interface for Wi-Fi. USB bring-up has been validated exclusively on the i.MX91 EVK platform.*

To set the host interface configuration to USB-USB on IW610 module, see Section 3.17.1.

Steps to bring up the Wi-Fi radio using the USB interface:

Step 1 – Load the driver modules.

```
root@imx91evk:~#  modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Step 2 – Verify the kernel debug messages in the `dmesg` command output.

```
[   93.141374] mlan: loading out-of-tree module taints kernel.
[   93.179628] wlan: Loading MWLAN driver
[   93.180057] wlan: Register to Bus Driver...
[   93.184123] USB probe: idVendor=471 idProduct=214 bInterfaceNumber=0
[   93.184153] VID/PID = 471/214, Boot2 version = 4000
[   93.184180] Attach moal handle ops, card interface type: 0x40d
[   93.184191] rps set to 0 from module param
[   93.186640] USBIW610: init module param from usr cfg
[   93.186709] card_type: USBIW610, config block: 0
[   93.186719] max_vir_bss=1
[   93.186730] cal_data_cfg=none
[   93.186739] fw_name=nxp/usbusbspi_iw610.bin.se
[   93.186777] rx_work=0 cpu_num=1
[   93.186786] Enable moal_recv_amsdu_packet
[   93.186825] Attach mlan adapter operations.card_type is 0x40d.
[   93.187571] Request firmware: nxp/usbusbspi_iw610.bin.se
[   95.061301] fw_dnld: 862436 bytes downloaded
[   95.062149] usbcore: registered new interface driver usbxxx
[   95.062236] wlan: Register to Bus Driver Done
[   95.062242] wlan: Driver loaded successfully
root@imx91evk:~# [   95.267670] usb 1-1: USB disconnect, device number 2
[   95.281858] USB: unregister device
[   95.281909] Free module params
[   95.697851] usb 1-1: new high-speed USB device number 3 using ci_hdrc
[   95.847531] USB probe: idVendor=471 idProduct=215 bInterfaceNumber=0
[   95.847558] VID/PID = 471/215, Boot2 version = 3201
[   95.847566] woal_usb_probe: invalid endpoint assignment
[   95.847850] USB probe: idVendor=471 idProduct=215 bInterfaceNumber=0
[   95.847862] VID/PID = 471/215, Boot2 version = 3201
[   95.847868] woal_usb_probe: invalid endpoint assignment
[   95.849138] USB probe: idVendor=471 idProduct=215 bInterfaceNumber=0
[   95.849164] VID/PID = 471/215, Boot2 version = 3201
[   95.849192] Attach moal handle ops, card interface type: 0x40d
[   95.849204] rps set to 0 from module param
[   95.849355] USBIW610: init module param from usr cfg
[   95.849413] card_type: USBIW610, config block: 0
[   95.849422] max_vir_bss=1
[   95.849431] cal_data_cfg=none
[   95.849437] fw_name=nxp/usbusbspi_iw610.bin.se
[   95.849463] rx_work=0 cpu_num=1
[   95.849472] Enable moal_recv_amsdu_packet
[   95.849496] Attach mlan adapter operations.card_type is 0x40d.
[   95.853989] WLAN FW is active
[   95.854007] on_time is 95636193711
[   95.874333] VDLL: Request firmware: nxp/usbusbspi_iw610.bin.se
[   95.876035] VDLL image: len=22800
[   95.876369] fw_cap_info=0x487cbf03, dev_cap_mask=0xffffffff
[   95.876402] uuid: 83d861b492165a6696476e79abdc3947
[   95.876409] max_p2p_conn = 8, max_sta_conn = 8
[   95.924328] Register NXP 802.11 Adapter mlan0
```

```
[   95.924444] wlan: uap%d set max_mtu 2000
[   95.927117] Register NXP 802.11 Adapter uap0
[   95.966907] Register NXP 802.11 Adapter wfd0
[   95.967002] wlan: version = USBIW610--18.99.5.p61-MM6X18537.p9-GPL-(FP92)
[   96.066087] usbcore: registered new interface driver btusb
[   96.067023] Bluetooth: hci0: unexpected event for opcode 0x0000
[   96.730791] Bluetooth: MGMT ver 1.23
[   96.743154] audit: type=1334 audit(1741286455.117:34): prog-id=21 op=UNLOAD
[   96.743272] audit: type=1334 audit(1741286455.117:35): prog-id=20 op=UNLOAD
[   96.757259] audit: type=1334 audit(1741286455.129:36): prog-id=24 op=LOAD
[   96.760962] NET: Registered PF_ALG protocol family
[   96.761414] audit: type=1334 audit(1741286455.133:37): prog-id=25 op=LOAD
[   96.763514] audit: type=1334 audit(1741286455.137:38): prog-id=26 op=LOAD
```

5.10 Bring-up of AW693-based u-blox module JODY-W6

Step 1 – Load the driver modules.

```
root@imx8mqevk:~# modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Step 2 – Verify the kernel debug messages in the `dmesg` command output.

```
[ 1487.061572] wlan: Loading MWLAN driver
[ 1487.061896] wlan: Register to Bus Driver...
[ 1487.062082] wlan_pcie 0001:01:00.0: enabling device (0000 -> 0002)
[ 1487.062152] PCI memory map Virt0: 0000000047df3fb4 PCI memory map Virt2:
 000000006bda95d8
[ 1487.062174] Attach moal handle ops, card interface type: 0x20c
[ 1487.062183] rps set to 0 from module param
[ 1487.062306] PCIEAW693: init module param from usr cfg
[ 1487.062365] card_type: PCIEAW693, config block: 0
[ 1487.062373] max_vir_bss=1
[ 1487.062380] cal_data_cfg=none
[ 1487.062385] fw_name=nxp/pcieuartaw693_combo_v1.bin.se
[ 1487.062402] rx_work=1 cpu_num=4
[ 1487.062409] Enable moal_recv_amsdu_packet
[ 1487.062420] Attach mlan adapter operations.card_type is 0x20c.
[ 1487.067993] Request firmware: nxp/pcieuartaw693_combo_v1.bin.se
[ 1487.309304] FW download over, size 949040 bytes
[ 1487.500866] WLAN FW is active
[ 1487.500875] on_time is 1487497124959
[ 1487.521125] VDLL image: len=144600
[ 1488.376083] fw_cap_info=0x487cef03, dev_cap_mask=0xffffffff
[ 1488.376112] mclient caps: tx_ba_stream_limit 16777215, tx_mpdu_pps 43000 50000
[ 1488.376124] uuid: 513146a6ecea554dbee7e6c63171c697
[ 1488.376133] max_p2p_conn = 8, max_sta_conn = 64
[ 1488.376636] VDLL image: len=144600
[ 1488.398645] Register NXP 802.11 Adapter mlan0
[ 1488.398787] wlan: uap%d set max_mtu 2000
[ 1488.401082] Register NXP 802.11 Adapter uap0
[ 1488.414703] Register NXP 802.11 Adapter wfd0
[ 1488.414751] wlan: version = PCIEAW693-18.99.2.p145.34-MM6X18537.p9-(FP92)
[ 1488.415497] wlan_pcie 0001:01:00.1: enabling device (0000 -> 0002)
[ 1488.415590] PCI memory map Virt0: 0000000050cdb9d0 PCI memory map Virt2:
 00000000157a486a
[ 1488.415608] Attach moal handle ops, card interface type: 0x20c
[ 1488.415618] rps set to 0 from module param
[ 1488.415734] PCIEAW693: init module param from usr cfg
[ 1488.415850] card_type: PCIEAW693, config block: 1
[ 1488.415859] max_vir_bss=1
[ 1488.415868] cal_data_cfg=none
[ 1488.415873] fw_name=nxp/pcieuartaw693_combo_v1.bin.se
[ 1488.415898] rx_work=1 cpu_num=4
[ 1488.415905] Enable moal_recv_amsdu_packet
[ 1488.415920] Attach mlan adapter operations.card_type is 0x20c.
[ 1488.432168] Request firmware: nxp/pcieuartaw693_combo_v1.bin.se
[ 1488.433587] WLAN FW already running! Skip FW download
[ 1488.433609] WLAN FW is active
[ 1488.433614] on_time is 1488429863997
[ 1488.434296] VDLL image: len=144600
[ 1488.434324] fw_cap_info=0x687ccb03, dev_cap_mask=0xffffffff
[ 1488.434346] mclient caps: tx_ba_stream_limit 16777215, tx_mpdu_pps 43000 50000
[ 1488.434353] uuid: 513146a6ecea554dbee7e6c63171c697
[ 1488.434361] max_p2p_conn = 8, max_sta_conn = 64
[ 1488.434762] VDLL image: len=144600
[ 1488.448997] Register NXP 802.11 Adapter mmlan0
[ 1488.449091] wlan: muap%d set max_mtu 2000
[ 1488.456268] Register NXP 802.11 Adapter muap0
[ 1488.466877] Register NXP 802.11 Adapter mwfd0
```

```
[ 1488.466922] wlan: version = PCIEAW693-18.99.2.p145.34-MM6X18537.p9-(FP92)
```

Step 3 – Verify the status of Wi-Fi interface.

```
root@imx8mqevk:~# ifconfig -a
```

Command output example:

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 172.29.122.114 netmask 255.255.255.0 broadcast 172.29.122.255
inet6 fe80::204:9fff:fe06:c5a5 prefixlen 64 scopeid 0x20<link>
ether 00:04:9f:06:c5:a5 txqueuelen 1000 (Ethernet)
RX packets 12650 bytes 13071690 (12.4 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 845 bytes 91415 (89.2 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 245 bytes 22790 (22.2 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 245 bytes 22790 (22.2 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
mlan0: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether b8:f4:4f:90:28:a1 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
mmlan0: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether b8:f4:4f:90:28:a2 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
muap0: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether ba:f4:4f:90:29:a2 txqueuelen 8192 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
mwfd0: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether ba:f4:4f:90:28:a2 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
uap0: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether ba:f4:4f:90:29:a1 txqueuelen 8192 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
wfd0: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether ba:f4:4f:90:28:a1 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Step 4 – Get the driver version.

```
root@imx8mqevk:~# cat /proc/mwlan/adapter0/uap0/info
```

Example of command output:

```
driver_name = "uap"
driver_version = PCIEAW693-aw693w-V1, PCIE, FP99, 18.99.2.p145.34-MM6X18537.p9-(FP92)
interface_name="uap0"
firmware_major_version=18.99.2
```

6 Bring-up of Bluetooth

This section describes two different methods to bring up the Bluetooth interfaces:

- Bring up using NXP Bluetooth UART driver on the i.MX Linux BSP version L6.1.22 and later.
- Bring up using Linux Open Source UART driver on i.MX Linux BSP version earlier than L6.1.22.

6.1 Bring-up using NXP Bluetooth UART driver

This section describes the steps to bring up the Bluetooth interface using NXP Bluetooth UART driver. NXP Bluetooth UART driver is part of i.MX Linux BSP version L6.1.22 and later.

Note: *Wi-Fi SDIO driver must be installed before NXP Bluetooth UART driver.*

Step 1 - Load Wi-Fi SDIO driver with the combo firmware or Wi-Fi standalone firmware.

- Load the Wi-Fi SDIO driver with either combo or the Wi-Fi standalone firmware to initialize the Wi-Fi interface before initializing the Bluetooth interface

See Section 3.6.7 "Plugging the wireless module" and Section 5.8.1 "Bring up the Wi-Fi interface".

Step 2 - Load NXP Bluetooth UART driver

```
modprobe btnxpuart
```

Note:

- *When using the Wi-Fi standalone firmware to load the Wi-Fi SDIO driver, btnxpuart.ko driver downloads the Bluetooth only firmware within ~5 seconds.*
- *When using the combo firmware to load the Wi-Fi SDIO driver, btnxpuart.ko driver does not download the Bluetooth only firmware.*

Step 3 - Confirm the status of HCI interface

- Issue the command:

```
root@imx8mqevk:~# hciconfig hci0 up
root@imx8mqevk:~# hciconfig
```

Command output example:

```
hci0: Type: Primary Bus: UART
BD Address: 00:50:43:24:83:B7 ACL MTU: 1021:7 SCO MTU: 120:6
UP RUNNING
RX bytes:1451 acl:0 sco:0 events:84 errors:0
TX bytes:789 acl:0 sco:0 commands:84 errors:0
```

6.2 Bring-up using Linux open source UART driver

Use the following steps to bring up the Bluetooth interfaces.

Note: *Start at **step 4** if you want to load the combo firmware and initialize the Wi-Fi driver prior to initializing the Bluetooth interface. See Section 3.6.7 and Section 5.8.1. Start from **Step 1** to download Bluetooth-only firmware.*

Step 1 - Download the UART firmware loader source code

- Open IW416, IW612, 88W9098, or 88W8987 webpage on NXP website (Section 10)
- Go to the **Software** section on the product page
- Download the latest Linux software package
- Look for *UART-FW-LOADER--GPL-src.tgz* in the release package
- Copy the firmware loader to i.MX 8M Quad EVK

```
scp UART-FW-LOADER--GPL-src.tgz root@<ip address of i.MX8MQ board>:/home/root/
```

Step 2 - Compile *uartfwloader_src* on i.MX 8M Quad EVK

- Extract *UART-FW-LOADER--GPL-src.tgz*
- Go to the directory *uartfwloader_src/linux/* and issue the command:

```
make make
```

fw_loader executable is created in *uartfwloader_src/linux/* directory.

Step 3 - Download Bluetooth-only firmware to the wireless module

- Check that that the firmware is available in */lib/firmware/nxp* directory
- Use fw_loader utility to download the firmware for Bluetooth only on /dev/ttymxc2

```
./fw_loader /dev/ttymxcX 115200 0 /lib/firmware/nxp/<BT_only>.bin
```

Note: *For 88W9098-based wireless modules (except Murata modules) and for IW416-based module (AM510MA), issue the commad to download the firmware for Bluetooth with 3M baud rate.*

```
./fw_loader /dev/ttymxc2 115200 0 /lib/firmware/nxp/uartuartXXXX_bt_v1.bin 3000000
```

Step 4 - Initialize *hci0* interface

Note: *Skip **Step 4** and **Step 5**, and go to **Step 6** for 88W9098-based wireless modules (except Murata modules) and for IW416-based module (AM510MA) as these work on 3M baud rate by default.*

- Attach ttymxc2 port:

```
root@imx8mqevk:~# hciattach /dev/ttymxc2 any 115200 flow
```

Command output example:

```
Setting TTY to N_HCI line discipline
Device setup complete
```

- Bring up HCI interface:

```
root@imx8mqevk:~# hciconfig hci0 up
root@imx8mqevk:~# hciconfig
```

Command output example:

```
hci0: Type: Primary Bus: UART
BD Address: 00:50:43:24:83:B7 ACL MTU: 1021:7 SCO MTU: 120:6
UP RUNNING
RX bytes:1451 acl:0 sco:0 events:84 errors:0
TX bytes:789 acl:0 sco:0 commands:84 errors:0
```

Step 5 - Change the speed of UART interface to 3M

```
root@imx8mqevk:~# hcitool -i hci0 cmd 0x3f 0x0009 0xc0 0xc6 0x2d 0x00
```

Step 6 - Initialize *hci0* interface on 3M baud rate

```
root@imx8mqevk:~# killall hciattach
root@imx8mqevk:~# hciattach /dev/ttymxc2 any 3000000 flow
root@imx8mqevk:~# hciconfig hci0 up
```

Step 7 - Scan nearby Bluetooth devices

```
root@imx8mqevk:~# hcitool -i hci0 scan
```

Command output example:

```
Scanning ...
40:49:0F:9E:66:FE desktop
root@imx8mqevk:~#
```

6.3 Bring-up using USB (IW610-based module only)

Note: *Only IW610-based Murata 2LL module supports the USB interface. The bring-up of Bluetooth with Murata 2LL module is tested on i.MX 91 EVK only.*

To set the host interface configuration to USB-USB on IW610 module, see Section 3.17.1.

Steps to bring up the Bluetooth radio using the USB interface.

Step 1 – Load the driver modules.

```
root@imx91evk:~#  modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Step 2 – Verify the kernel debug messages in the `dmesg` command output.

```
[   93.141374] mlan: loading out-of-tree module taints kernel
[   93.179628] wlan: Loading MWLAN driver
[   93.180057] wlan: Register to Bus Driver...
[   93.184123] USB probe: idVendor=471 idProduct=214 bInterfaceNumber=0
[   93.184153] VID/PID = 471/214, Boot2 version = 4000
[   93.184180] Attach moal handle ops, card interface type: 0x40d
[   93.184191] rps set to 0 from module param
[   93.186640] USBIW610: init module param from usr cfg
[   93.186709] card_type: USBIW610, config block: 0
[   93.186719] max_vir_bss=1
[   93.186730] cal_data_cfg=none
[   93.186739] fw_name=nxp/usbusbspi_iw610.bin.se
[   93.186777] rx_work=0 cpu_num=1
[   93.186786] Enable moal_recv_amsdu_packet
[   93.186825] Attach mlan adapter operations.card_type is 0x40d.
[   93.187571] Request firmware: nxp/usbusbspi_iw610.bin.se
[   95.061301] fw_dnld: 862436 bytes downloaded
[   95.062149] usbcore: registered new interface driver usbxxx
[   95.062236] wlan: Register to Bus Driver Done
[   95.062242] wlan: Driver loaded successfully
root@imx91evk:~# [   95.267670] usb 1-1: USB disconnect, device number 2
[   95.281858] USB: unregister device
[   95.281909] Free module params
[   95.697851] usb 1-1: new high-speed USB device number 3 using ci_hdrc
[   95.847531] USB probe: idVendor=471 idProduct=215 bInterfaceNumber=0
[   95.847558] VID/PID = 471/215, Boot2 version = 3201
[   95.847566] woal_usb_probe: invalid endpoint assignment
[   95.847850] USB probe: idVendor=471 idProduct=215 bInterfaceNumber=0
[   95.847862] VID/PID = 471/215, Boot2 version = 3201
[   95.847868] woal_usb_probe: invalid endpoint assignment
[   95.849138] USB probe: idVendor=471 idProduct=215 bInterfaceNumber=0
[   95.849164] VID/PID = 471/215, Boot2 version = 3201
[   95.849192] Attach moal handle ops, card interface type: 0x40d
[   95.849204] rps set to 0 from module param
[   95.849355] USBIW610: init module param from usr cfg
[   95.849413] card_type: USBIW610, config block: 0
[   95.849422] max_vir_bss=1
[   95.849431] cal_data_cfg=none
[   95.849437] fw_name=nxp/usbusbspi_iw610.bin.se
[   95.849463] rx_work=0 cpu_num=1
[   95.849472] Enable moal_recv_amsdu_packet
[   95.849496] Attach mlan adapter operations.card_type is 0x40d.
[   95.853989] WLAN FW is active
[   95.854007] on_time is 95636193711
[   95.874333] VDLL: Request firmware: nxp/usbusbspi_iw610.bin.se
[   95.876035] VDLL image: len=22800
[   95.876369] fw_cap_info=0x487cbf03, dev_cap_mask=0xffffffff
[   95.876402] uuid: 83d861b492165a6696476e79abdc3947
[   95.876409] max_p2p_conn = 8, max_sta_conn = 8
[   95.924328] Register NXP 802.11 Adapter mlan0
```

```
[   95.924444] wlan: uap%d set max_mtu 2000
[   95.927117] Register NXP 802.11 Adapter uap0
[   95.966907] Register NXP 802.11 Adapter wfd0
[   95.967002] wlan: version = USBIW610--18.99.5.p61-MM6X18537.p9-GPL-(FP92)
[   96.066087] usbcore: registered new interface driver btusb
[   96.067023] Bluetooth: hci0: unexpected event for opcode 0x0000
[   96.730791] Bluetooth: MGMT ver 1.23
[   96.743154] audit: type=1334 audit(1741286455.117:34): prog-id=21 op=UNLOAD
[   96.743272] audit: type=1334 audit(1741286455.117:35): prog-id=20 op=UNLOAD
[   96.757259] audit: type=1334 audit(1741286455.129:36): prog-id=24 op=LOAD
[   96.760962] NET: Registered PF_ALG protocol family
[   96.761414] audit: type=1334 audit(1741286455.133:37): prog-id=25 op=LOAD
[   96.763514] audit: type=1334 audit(1741286455.137:38): prog-id=26 op=LOAD
```

6.4 UART baudrate detection and adaption (IW611 and IW612 only)

After IW611 and IW612 firmware download and during the initialization phase, the UART baudrate of IW611/IW612 is set to the default value of 115,200 bps if the OTP is not programmed, or to the value set in the OTP memory (115,200 or 3M).

To communicate with IW611 or IW612 over UART/HCI interface, the Host application must be configured with the baudrate value set for IW611 and IW612. The developers of host applications must know the baudrate value and configure the application accordingly.

With the UART baudrate detection and adaption feature, the Host application is set with the baudrate value of 115,200 bps or 3M bps. The IW611/IW612 firmware automatically detects the baudrate value in the initial HCI command from the Host application and adapts to the value for IW611/IW612.

This feature avoids configuration and maintenance issues on the Host application.

6.4.1 Requirements

- The feature applies only to UART with 4-wire communication (TX, RX, RTS, and CTS).
  - UART flow Control (RTS/CTS) must be supported on the host platform.
- The Host application must initiate the communication using one of the following HCI commands:
  - HCI Reset command (ogf:0x03, ocf:0x0003)
  - HCI Read Local Features command (ogf:0x04, ocf:0x0003)
  - HCI Read Local Version command (ogf:0x04, ocf:0x0001)

Note: *If the first HCI command is none of the above, IW611 or IW612 may not respond.*

The Host application must use either 115,200 or 3M as initial baudrate value for the detection.

6.4.2 Baudrate detection and adaption flow

The flow for UART baudrate detection and adaption is the following:

- The controller firmware configures 3M as the initial baudrate value.
- The Host application sends the first HCI command to the controller firmware over UART using 3M or 115,200 baudrate value. Refer to Section 6.4.1 for the list of supported initial HCI commands.
- The firmware detects the first HCI command and the baudrate value.
  - If the baudrate value is 3M, the firmware communicates with the Host using the 3M baudrate.
  - If the baudrate value is not 3M: the firmware first sets RTS high and the baudrate value to 115,200, then sets RTS back to low, and communicates with the Host using the 115,200 baudrate value.

Figure 36 shows the baudrate detection and adaption flow.

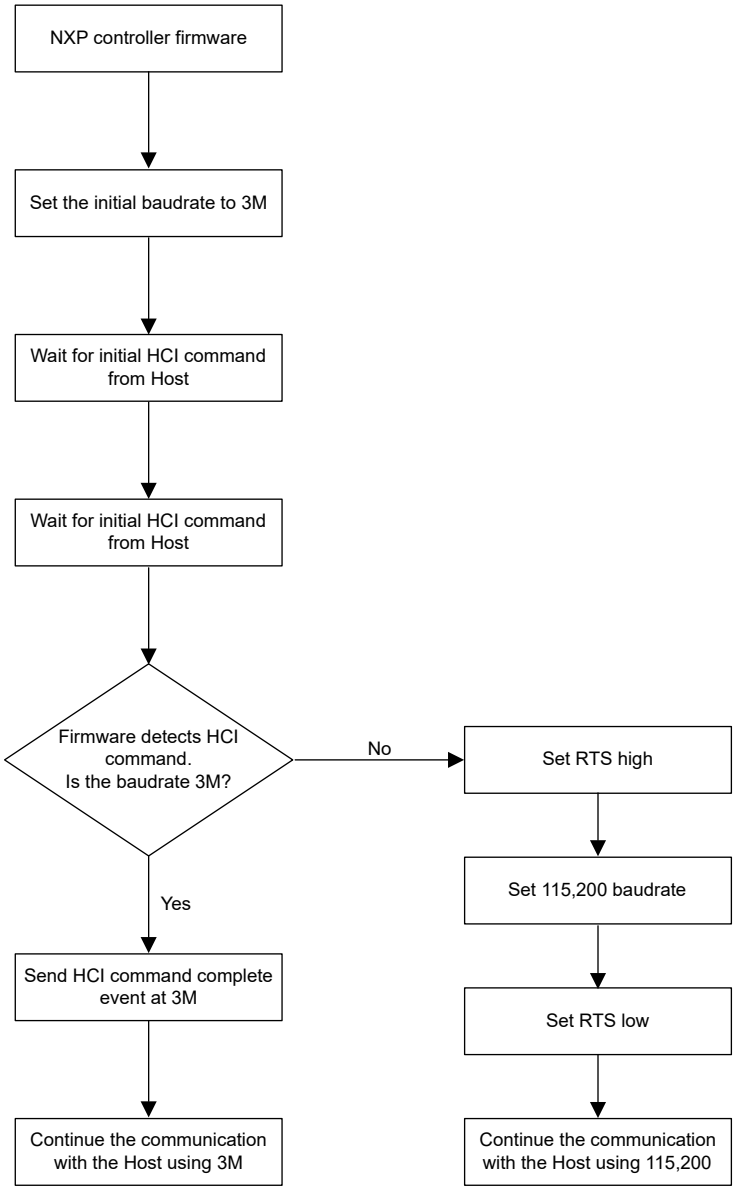

Figure 36. Baudrate detection and adaptation flow

6.4.3 Example of baudrate auto detection and adaption on IW612

In this section, the UART Bluetooth interface is brought up with 115,200 and 3M baudrate values, and IW612 firmware detects the two values configured for the host application.

Step 1 – Download the Bluetooth standalone firmware using the fw_loader utility. See Section 6.2.

Step 2 – Bring up the Bluetooth interface using 115,200 or 3M baudrate

Example of `hciattach` command with 115,200 baudrate:

```
$ root@imx8mmevk:~# hciattach /dev/ttymxc2 any 115200 flow
Device setup complete
$ root@imx8mmevk:~#
```

Example of `hciattach` command with 3M baudrate:

```
$ root@imx8mmevk:~# hciattach /dev/ttymxc2 any 3000000 flow
Device setup complete
$ root@imx8mmevk:~#
```

Note: *The hciattach utility first sends the "Read Local Supported Features" HCI command which fulfills the requirement of first HCI command (Section 6.4.1).*

Step 3 – Verify that the Bluetooth interface is up.

```
$ root@imx8mmevk:~# hciconfig hci0 up
$ root@imx8mmevk:~# hciconfig
```

Command output example:

```
hci0:   Type: Primary  Bus: UART
BD Address: C0:95:DA:00:E5:15  ACL MTU: 1021:7  SCO MTU: 120:6
UP RUNNING
RX bytes:1451 acl:0 sco:0 events:84 errors:0
TX bytes:789 acl:0 sco:0 commands:84 errors:0
```

6.5 Configure BTNXPUART driver for high-speed communication

The section shows how to set the baudrate to 4M using BTNXPUART driver. The 4M baudrate enables fast firmware downloads and high-speed HCI communication.

Note:

- *The BTNXPUART driver supports the feature from i.MX Linux BSP version LF6.12.y_2.1.0.*
- *The 88W8987, 88W9098, IW416, IW612, IW610, and AW693 wireless products support the feature.*

Step 1 – Use the device tree compiler (dtc) to compile the DTB file into a DTS file.

The dtc tool is available on Ubuntu host machines. If not, use the `apt` command to install dtc on your machine.

```
dtc -I dtb -O dts -o imx8mq-evk.dts imx8mq-evk.dtb
```

Step 2 – Update the UART Bluetooth node in the *imx8mq-evk.dts* file converted in step 1.

```
bluetooth {
        compatible = "nxp,88w8987-bt";
        max-speed = <4000000>
};
```

Step 3 – Recompile the updated DTS file into a DTB file.

The imx8mq-evk.dts file includes the 4M baudrate for Bluetooth UART.

```
dtc -I dts -O dtb -o imx8mq-evk.dtb imx8mq-evk.dts
```

Step 4 – Load the Wi-Fi SDIO driver with either the combo or the Wi-Fi standalone firmware.

The Wi-Fi interface must be initialized before the Bluetooth interface. See Section 3.6 and Section 5.8.1.

Step 5 – Load BTNXPUART driver.

```
root@imx8mqevk:~#  modprobe btnxpuart
```

Note:

- *When using the Wi-Fi standalone firmware to load the Wi-Fi SDIO driver, the btnxpuart.ko driver downloads the Bluetooth only firmware within ~5 seconds.*
- *When using the combo firmware to load the Wi-Fi SDIO driver, the btnxpuart.ko driver does not download the Bluetooth-only firmware.*
- *The btnxpuart.ko driver sends the `3f|09` vendor command to set the 4M baudrate. After a reset, the btnxpuart.ko driver sends a vendor command to switch the baud rate from 4M to the fw-init-baudrate value.*

Step 6 – Reset the HCI interface to set the baudrate to 4M.

```
root@imx8mqevk:~# hciconfig hci0 reset
root@imx8mqevk:~# hciconfig
```

Command output example:

```
hci0: Type: Primary Bus: UART
BD Address: 00:50:43:24:83:B7 ACL MTU: 1021:7 SCO MTU: 120:6
UP RUNNING
RX bytes:1451 acl:0 sco:0 events:84 errors:0
TX bytes:789 acl:0 sco:0 commands:84 errors:0
```

Step 7 – Check if the 4M baudrate is already configured.

Use the vendor-specific command to set the baudrate to 4M. If the command returns an error, the host and controller are already communicating at 4M baudrate.

```
root@imx8mqevk:~# hcitool -i hci0 cmd 3f 09 00 09 3d 00
```

Command output example:

```
< HCI Command: ogf 0x3f, ocf 0x0009, plen 4
  00 09 3D 00
> HCI Event: 0x0e plen 4
  01 09 FC 12
```

7 Bring-up of 802.15.4

Note: *In the current release, the 802.15.4 subsystem is only supported on IW612 evaluation board (EVB) with i.MX 8M Mini (8MMINILPD4-EVK@2020) host platform.*

This section describes the bring-up steps for the 802.15.4 interface of the NXP based IW612 module on i.MX 8M Mini platform.

7.1 Build ot-daemon for i.MX 8M Mini platform

This section shows how to cross-compile ot-daemon tool for i.MX 8M Mini Linux platform.

7.1.1 Download OpenThread source code from github

```
git clone https://github.com/openthread/openthread.git
cd openthread
git checkout 481a064f034496d93273f4b837d46559f7b74e5c
```

Note: *See the release notes for the latest OpenThread tag used during QA.*

7.1.2 Configure i.MX Linux SDK build environments

- Download and install i.MX 8M Linux SDK. See i.MX Yocto Project User's Guide for information on the SDK download and configuration.
- When the SDK is installed successfully, issue the command to set up i.MX Linux SDK build environment.

```
source /opt/fsl-imx-wayland/5.10-hardknott/environment-setup-cortexa53-crypto-poky-linux
```

7.1.3 Cross-compile ot-daemon and ot-ctl

- Issue the command to cross compile OpenThread source code in *openthread* directory

The cross compilation generates *ot-daemon* and *ot-ctl* at *openthread/build/posix/src/*

```
#cd openthread
#./script/bootstrap
#./script/cmake-build posix -DOT_DAEMON=ON -DOT_POSIX_CONFIG_RCP_BUS=SPI -
DOT_COMPILE_WARNING_AS_ERROR=OFF -DMBEDTLS_FATAL_WARNINGS=OFF -DOT_THREAD_VERSION=1.2
#file build/posix/src/posix/ot-*
build/posix/src/posix/ot-ctl: ELF 64-bit LSB shared object, ARM aarch64,
version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-aarch64.so.1,
BuildID[sha1]=9a228a97a831f085b2d3a6dba159808610c704d9, for GNU/Linux 3.14.0, with
 debug_info, not stripped
build/posix/src/posix/ot-daemon: ELF 64-bit LSB shared object, ARM aarch64,
version 1 (GNU/Linux), dynamically linked, interpreter /lib/ld-linux-aarch64.so.1,
BuildID[sha1]=9ab7071121f7ecaba74004d62aa811f2cc68175a, for GNU/Linux 3.14.0, with
 debug_info, not       stripped
```

7.2 Bring-up of a Thread Network

This section shows how to bring up a Thread network on i.MX 8M Mini platform.

Step 1 – Ensure that the SPI interface is enumerated when the hardware connection is ready

The software release package includes the SPI device tree file for i.MX 8M Mini platform.

- Copy SPI device tree file to i.MX 8M Mini platform and issue the reboot command to restart the device.

```
root@imx8mmevk:~# cp imx8mm-evk.dtb /run/media/mmcblk2p1/
root@imx8mmevk:~# reboot
```

- Look for SPI interface enumeration

/dev/gpiochip5 is the SPI interface for IW612 connected to i.MX 8M Mini.

```
root@imx8mmevk:~# ls -l /dev/gpio*
crw------- 1 root root 254, 0 Sep 20 10:16 /dev/gpiochip0
crw------- 1 root root 254, 1 Sep 20 10:16 /dev/gpiochip1
crw------- 1 root root 254, 2 Sep 20 10:16 /dev/gpiochip2
crw------- 1 root root 254, 3 Sep 20 10:16 /dev/gpiochip3
crw------- 1 root root 254, 4 Sep 20 10:16 /dev/gpiochip4
crw------- 1 root root 254, 5 Sep 20 10:16 /dev/gpiochip5
```

Step 2 – Load the combo firmware

Note: *You must first load the combo firmware and initialize the Wi-Fi driver prior to initializing the 802.15.4 interface. See Section 3.6.7 and Section 5.*

Step 3 – Start ot-daemon in the background

```
root@imx8mmevk:/usr/sbin# ot-daemon "spinel+spi:///dev/spidev1.0?gpio-int-device=/dev/
gpiochip5&gpio-int-line=12&gpio-reset-device=/dev/gpiochip5&gpio-reset-line=13&spi-mode=0&spi-
speed=1000000&spi-reset-delay=500&"
```

Table 22. Command parameters

Parameter	Description
`spinel+spi://`	Path to SPI interface
`gpio-int-device`	Path to the Linux sysfs-exported GPIO pin with the Interrupt signal
`gpio-init-line`	The offset index of the Interrupt signal for the associated GPIO pin
`gpio-reset-device`	Path to the Linux sysfs-exported GPIO pin for the Reset signal
`gpio-reset-line`	The offset index of Reset signal for the associated GPIO pin
`spi-mode`	SPI mode to use (0-3)
`spi-speed`	SPI speed in Hertz
`spi-reset-delay`	The delay after “RESET” assertion, in milliseconds

Step 4 – Verify the ot-daemon process

- Issue the command:

```
root@imx8mmevk:/usr/sbin# ps -aux | grep ot-daemon
```

Command output example:

```
root      652  0.3  0.1   7264  3496 ttymxc1  S    15:42   0:00 ot-daemon spinel+spi:///
dev/spidev1.0?gpio-intdevice=/dev/gpiochip5&gpiointline=12&gpio-reset-device=/dev/
gpiochip5&gpio-reset-line=13&spimode=0&spispeed=1000000&spiresetdelay=500
root      663  0.0  0.0   3008  1276 ttymxc1  S+   15:42   0:00 grep ot-daemon
root@imx8mqevk:/usr/sbin#
```

Step 5 – Verify the ot-ctl utility

- Issue the command:

```
root@imx8mmevk:/usr/sbin# ot-ctl thread stop
Done
```

7.3 Build the Kernel image and drivers to enable OTBR support for i.MX 8M Mini platform

Open Thread Border Router (OTBR) connects a Thread network to other IP-based networks such as Wi-Fi or Ethernet. A Thread network requires a Border Router to connect to other networks.

This section explains how to bring up OTBR agent on the NXP-based IW612 module with i.MX 8M Mini platform. The steps include rebuilding the Linux Kernel image and NXP drivers to comply with the new Linux Kernel image.

7.3.1 Build Linux kernel image

To enable OTBR for i.MX 8M Mini platform, two changes are required at Linux Kernel level:

- Modification of the device tree to support the IW612 hardware.
- Addition of OTBR Linux kernel network configurations.

Note: *Another Linux platform such as ubuntu laptop is required for the cross compilation of OTBR agent.*

Step 1 – Download NXP Kernel source code from GitHub.

```
git clone https://github.com/nxp-imx/linux-imx
cd linux-imx/
```

Step 2 – Download Wi-Fi source code from GitHub.

```
git clone https://github.com/nxp-imx/mwifiex.git
cd mwifiex/
```

Step 3 – Install the Linux machine dependencies.

```
apt-get install gcc-aarch64-linux-gnu flex bison
apt-get install device-tree-compiler
```

Step 4 - Configure i.MX Linux build environment.

```
export ARCH=arm64
export CROSS_COMPILE=aarch64-linux-gnu-
```

Step 5 – Compile the *.dts* file

```
cd linux-imx/
cpp -nostdinc -I include -I arch  -undef -x assembler-with-cpp arch/arm64/boot/dts/
freescale/imx8mm-evk-any_bsp_gpio_irq.dts imx8mm-evk-any_bsp_gpio_irq.dts.preprocessed
dtc -I dts -O dtb -p 0x1000 imx8mm-evk-any_bsp_gpio_irq.dts.preprocessed -o imx8mm-evk-
any_bsp_gpio_irq.dtb
```

Step 6 – Copy the imx_v8_defconfig configurations to the *.config* file.

```
cp arch/arm64/configs/imx_v8_defconfig .config
```

Step 7 – Save the configurations of the Linux kernel.

```
make menuconfig
```

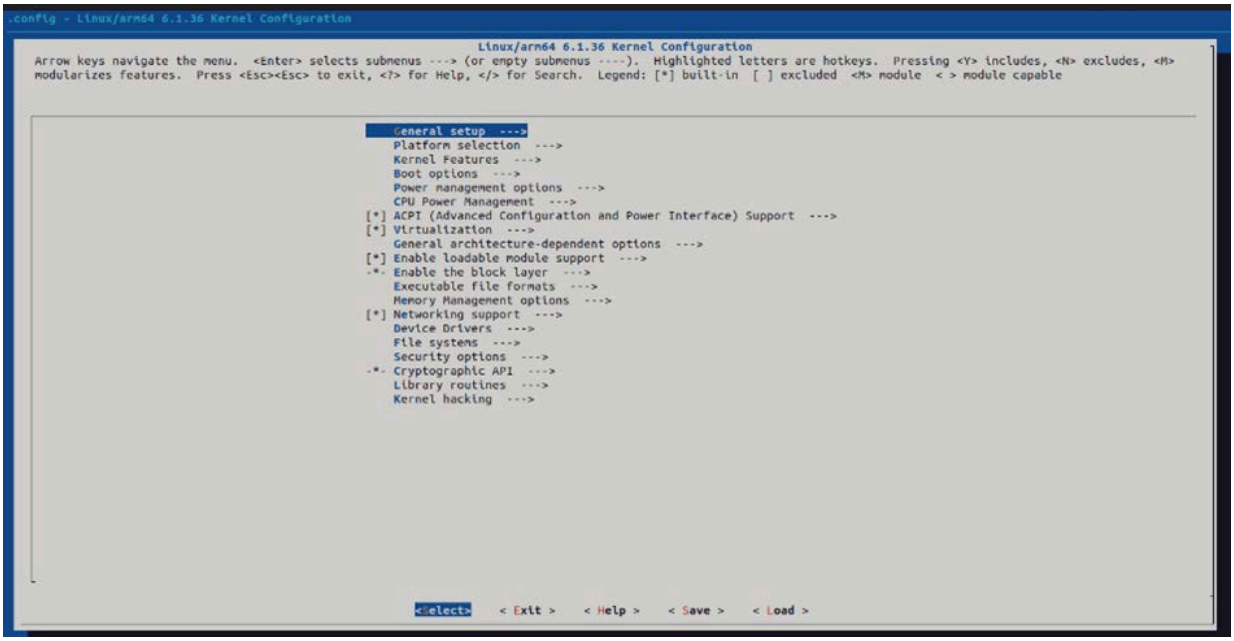

Figure 37. Configuration of the Linux kernel

- To save the updated configurations, save and exit the menuconfig.

Step 8 - Apply the Kernel configurations for OTBR.

- Apply the configurations for otbr-agent.

```
./scripts/kconfig/merge_config.sh -m .config ../0001-otbr-agent-v2-imx-v8.cfg
```

- Apply the configurations for otbr-firewall.

```
./scripts/kconfig/merge_config.sh -m .config ../0001-otbr-firewall-imx-v8.cfg
```

- Apply the configurations for sysfs gpio support.

```
./scripts/kconfig/merge_config.sh -m .config ../0001-gpio_sysfs_support.cfg
```

Note: *All the above-mentioned configuration files are available in the standard IW612 release package available on NXP website.*

Step 9 - Save the configurations and exit.

```
make menuconfig
```

Step 10 - Validate the new configurations, generate the dependencies, and modify the local version.

```
make menuconfig
```

- Go to **General setup** menu

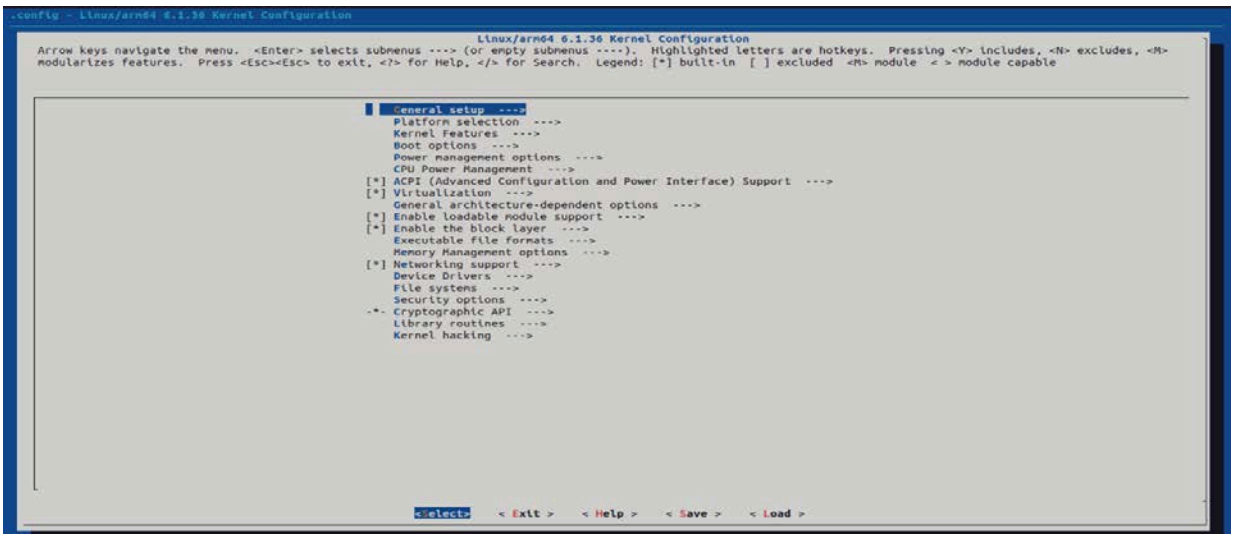

Figure 38. General setup menu

- Go to **Local version - append to kernel release** submenu and type **-otbr**.

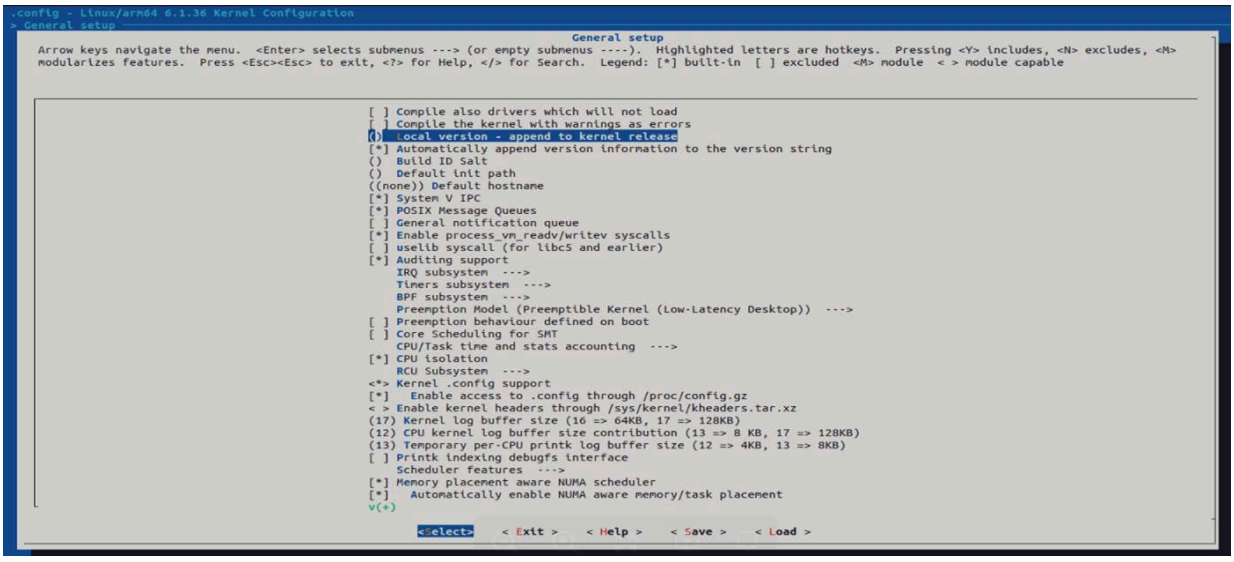

Figure 39. Location version

- Click **Yes** and exit the menuconfig.

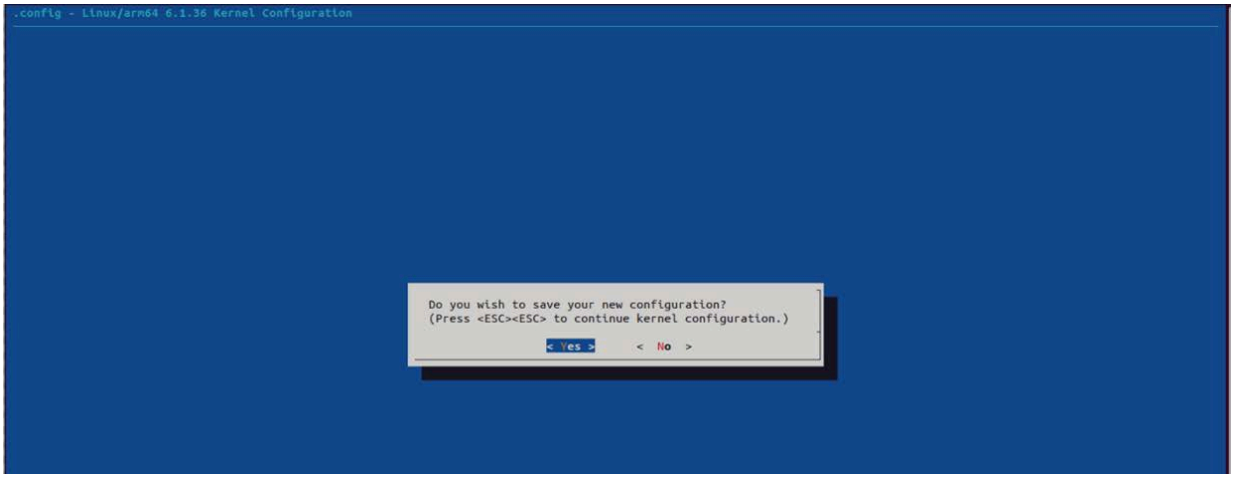

Figure 40. Exit the menuconfig

Step 11 – Check the configured kernel version.

```
grep otbr .config
```

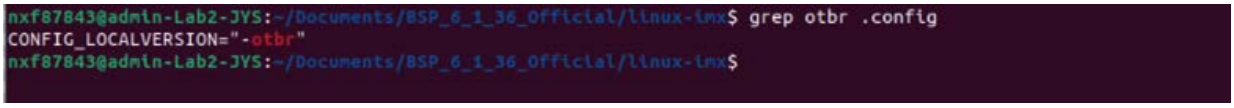

Figure 41. Check the kernel version

Step 12 – Cross compile the Linux Kernel.

```
export ARCH=arm64
export CROSS_COMPILE=aarch64-linux-gnu-
make -j8 Image modules
```

Step 13 – Install the kernel module.

- Remove existing *_install* files.

```
rm -rf _install
```

- Create new *_install* directory.

```
mkdir _install
```

- Install the kernel modules.

```
make INSTALL_MOD_PATH=./_install modules_install
```

- Remove some extra files.

```
rm _install/lib/modules/*/build
rm _install/lib/modules/*/source
```

7.3.2 Build the drivers

This section provides the steps to compile the drivers and enable the OTBR agent support.

Step 1 – Compile the drivers *mlan.ko* and *moal.ko*.

```
cd ../mwifiex/mxm_wifiex/wlan_src/
make clean
make CROSS_COMPILE=aarch64-linux-gnu- KERNELDIR=../../../linux-imx/ build
```

Step 2 - Copy *mlan.ko* and *moal.ko* in Linux Kernel modules directory.

```
cd ../../linux-imx/
mkdir _install/lib/modules/6.1.36-otbr/extra
cp ../mwifiex/mxm_wifiex/bin_wlan/*.ko _install/lib/modules/6.1.36-otbr/extra
```

Step 3 - Create Linux Kernel and Modules delivery.

```
mkdir _delivery
cp arch/arm64/boot/Image _delivery/.
cd _install/lib/modules/
zip -r ../../../_delivery/modules.zip *
```

Step 4 – Flash the i.MX 8M Mini board.

Note: *BSP version 6.1.x or greater is recommended.*

Step 5 – Copy all the files to i.MX 8M Mini platform.

```
scp Image root@<i.MX IP Addr>:/run/media/boot-mmcblk2p1/Image_otbr
scp imx8mm-evk-any_bsp_gpio_irq.dtb root@<i.MX IP Addr>:/run/media/boot-mmcblk2p1/
scp modules.zip root@<i.MX IP Addr>:/lib/modules
scp imx-linux-otbr-044-091123-0f7e849.zip imx-linux-othost-043-091123-0f7e849.zip
install_extra_deb_pkgs.sh otbr_extra_deb_packages.tar
otbr_firewall_extra_deb_packages.tar root@<i.MX IP Addr>:/home/root
```

- For kernel image, *module.zip* file and *imx8mm-evk-any_bsp_gpio_irq.dtb* file, refer the section Build Linux kernel image.
- For the files *imx-linux-otbr-044-091123-0f7e849.zip*, *imx-linux-othost-043-091123-0f7e849.zip*, *install_extra_deb_pkgs.sh otbr_extra_deb_packages.tar* and *otbr_firewall_extra_deb_packages.tar*, see IW612 release available on NXP website.

Step 6 - Power cycle the i.MX 8M Mini board.

7.4 Bring up of an OTBR agent

This section describes the bring up of an OTBR agent for the Open Thread network.

The compilation steps are available in Section 7.3.1 and Section 7.3.2.

Step 1 - Enter u-boot mode by pressing any key on bootup.

Step 2 – Set the u-boot environment variable to the compiled image and *.dtb.* file.

```
env set image Image_otbr
env set fdtfile imx8mm-evk-any_bsp_gpio_irq.dtb
```

Step 3 – Confirm the u-boot configurations and boot up the board.

- Check the environment configurations

```
env print image
env print fdtfile
```

- Save the environment and boot the board.

```
env save
boot
```

Step 4 – Check the kernel version "6.1.36-otbr".

```
uname -r
```

Figure 42. Get the kernal version

Note: *The Kernel version is "6.1.36-otbr" because the CONFIG_LOCALVERSION="-otbr" is set during Linux Kernel build.*

Step 5 – Install the customized Linux module.

```
cd /lib/modules
unzip --qq modules.zip
```

Step 6 – Unzip the OT and OTBR packages.

```
cd /home/root
unzip -qq imx-linux-othost-043-091123-0f7e849.zip
unzip -qq imx-linux-otbr-044-091123-0f7e849.zip
```

Step 7 - Install the customized OT and OTBR packages.

```
cd imx-linux-othost-043-091123-0f7e849/SPI/
cp -R * /
cd ../../imx-linux-otbr-044-091123-0f7e849/SPI/
cp -R * /
cd ../..
./install_extra_deb_pkgs.sh
rm -rf *
sync
```

Step 8 – Reboot the i.MX 8M Mini board.

Step 9 – Load the IW612 combo firmware.

- Create a customized *wifi_mod_para_iw612.conf*

```
touch wifi_mod_para_iw612.conf
cat wifi_mod_para.conf | grep IW612 -B1 -A9 > wifi_mod_para_iw612.conf
cat wifi_mod_para_iw612.conf
```

Figure 43. Customized *wifi_mod_para_iw612.conf* file

Note: *Change the name of the firmware for the secure or non-secure board.*

Step 10 - Copy a backup version of the BSP firmware to i.MX 8M Mini board.

- Create a backup copy of the current i.MX BSP firmware.

```
cd /lib/firmware/nxp/
mv sduart_nw61x_v1.bin.se sduart_nw61x_v1.bin.se_bkp
```

- Copy the new firmware files to i.MX 8M Mini platform. For example:

```
scp sduart_nw61x_v1.bin root@<IMX8 IP Addr>:/lib/firmware/nxp
scp sduart_nw61x_v1.bin.se root@<IMX8 IP Addr>:/lib/firmware/nxp
```

Step 11 - Reset the SDIO interface.

```
echo -n 30b50000.mmc > /sys/bus/platform/drivers/sdhci-esdhc-imx/unbind
sleep 1
echo -n 30b50000.mmc > /sys/bus/platform/drivers/sdhci-esdhc-imx/bind
```

Step 12 - Load the Wi-Fi drivers.

```
cd /lib/modules/6.1.36-otbr/extra/
insmod mlan.ko
insmod moal.ko mod_para=/lib/firmware/nxp/wifi_mod_para_iw612.conf
```

Step 13 – Create *wpa_supplicant.conf* file with the connecting AP information (SSID and password).

```
cd /home/root
cat << EOF > wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
network={
        ssid="nxp_matter-5G"
        psk="nxp12345"
}
EOF
```

Figure 44. Creation of *wpa_supplicant.conf* file

Note: *Use the SSID and password of your Wi-Fi access point.*

Step 14 – Bring up STA mode using *wpa_supplicant.conf* file.

```
ifconfig mlan0 up
killall wpa_supplicant
wpa_supplicant -d -B -i mlan0 -c ./wpa_supplicant.conf
Step 15 - Set the required Wi-Fi configurations.
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 2 > /proc/sys/net/ipv6/conf/all/accept_ra
```

Note: *Thread network must be run on ipV6 network.*

Step 16 – Run “otbr-agent” as a background process.

```
otbr-agent -d 6 -I wpan0 -B mlan0 'spinel+spi:///dev/spidev1.0?gpio-reset device=/
dev/gpiochip5&gpio-int-device=/dev/gpiochip5&gpio-int-line=12&gpio-reset-line=13&spi-
mode=0&spi-speed=1000000&spi-reset-delay=0' &
```

The i.MX device is ready to use the OTBR agent to connect with the Wi-Fi Access Point and Open Thread leader device.

8 Bring-up of ZigBee on i.MX 8M Mini

This section shows how to bring up ZigBee on i.MX 8M Mini EVK platform.

Note: *The 802.15.4 subsystem is only supported on IW612 evaluation board (EVB) with i.MX 8M Mini (8MMINILPD4-EVK@2020) host platform. This section describes the steps to bring up the 802.15.4 interface of NXP-based IW612 module on i.MX 8M Mini platform, and for Zigbee-only mode. Zigbee dual-pan mode is not supported.*

Step 1 – See Section 4 for how to download and flash the pre-built image for i.MX 8M Mini EVK.

Step 2 - Download the latest release of the software package ref.[23].

Step 3 - Toolchain setup:

- Follow the guidelines in i.MX Yocto Project User's Guide ref.[15] and create the repo.
- Create the toolchain.

```
# bitbake meta-toolchain
```

- Install the toolchain in X86 host-machine from the following directory.

```
<PATH_TO_BUILD_DIRECTORY>/tmp/deploy/sdk/fsl-imx-wayland-glibc-x86_64-meta-toolchain-
armv8a-imx8mm-lpddr4-evk-toolchain-6.6-scarthgap.sh
```

Step 4 - Extract the ZIP file on the Linux PC.

```
zboss-host-release-<version>-<date>-<gitRef>.zip
```

Step 5 - Set the environment for the cross compiler.

```
$ . /opt/fsl-imx-xwayland/<BSP VERSION>/environment-setup-<XXX>
$ export CC=`echo ${CC} | sed 's/-O2 -D_FORTIFY_SOURCE=2 //g'`
$ export CXX=`echo ${CXX} | sed 's/-O2 -D_FORTIFY_SOURCE=2 //g'`
$ export CPP=`echo ${CPP} | sed 's/-O2 -D_FORTIFY_SOURCE=2 //g'`
```

Step 6 – Cross compile the simple_gw application.

- Go to the *simple_gw* directory.

```
$ cd simple_gw/
```

- Compile the application.

```
$ make clean all
```

Example of command output.

```
aarch64-poky-linux-gcc -march=armv8-a+crc+crypto -mbranch-protection=standard -fstack-
protector-strong -Wformat -Wformat-security -Werror=format-security --sysroot=/opt/
fsl-imx-wayland/6.6-scarthgap/sysroots/armv8a-poky-linux -MT simple_gw.o -MMD -MP -MF
 simple_gw.d -I../../libs -I../../include -I../../include/platform -I../../include/
platform/osif -O3 -c simple_gw.c -o simple_gw.o
      aarch64-poky-linux-gcc -march=armv8-a+crc+crypto -mbranch-protection=standard -
fstack-protector-strong -Wformat -Wformat-security -Werror=format-security --sysroot=/
opt/fsl-imx-wayland/6.6-scarthgap/sysroots/armv8a-poky-linux -MT ias_cie_addon.o -MMD
 -MP -MF ias_cie_addon.d -I../../libs -I../../include -I../../include/platform -I../../
include/platform/osif -O3 -c ias_cie_addon.c -o ias_cie_addon.o
      aarch64-poky-linux-gcc -march=armv8-a+crc+crypto -mbranch-protection=standard -
fstack-protector-strong -Wformat -Wformat-security -Werror=format-security --sysroot=/
opt/fsl-imx-wayland/6.6-scarthgap/sysroots/armv8a-poky-linux -o simple_gw -Wl,--start-
group simple_gw.o ias_cie_addon.o ../../libs/libzboss.a -Wl,--end-group  -Wl,--gc-
sections -Xlinker -Map=simple_gw.map -s
```

Step 7 – Modify the script *imx-dualpan.sh* located in *<zboss-host-release-<version>-<date>-<gitRef>/scripts* directory.

Note: *The script modification is required for compatibility with i.MX 8M Mini host running Linux BSP v6.6.23.*

- On line number 273 and for `i.MX8MM|i.MX8MN`, replace **`gpiochip5`** with `gpiochip1`.
- On line number 522 and for `function flash_firmware()` replace `fw_loader_imx_lnx` with `./fw_loader_imx_lnx`.

Step 8 – Copy the compiled ZigBee application to i.MX 8M Mini platform.

- Connect an ethernet cable between i.MX 8M Mini board and the network router (or network switch).
- Connect the Linux PC to the same network router via ethernet (Figure 45).

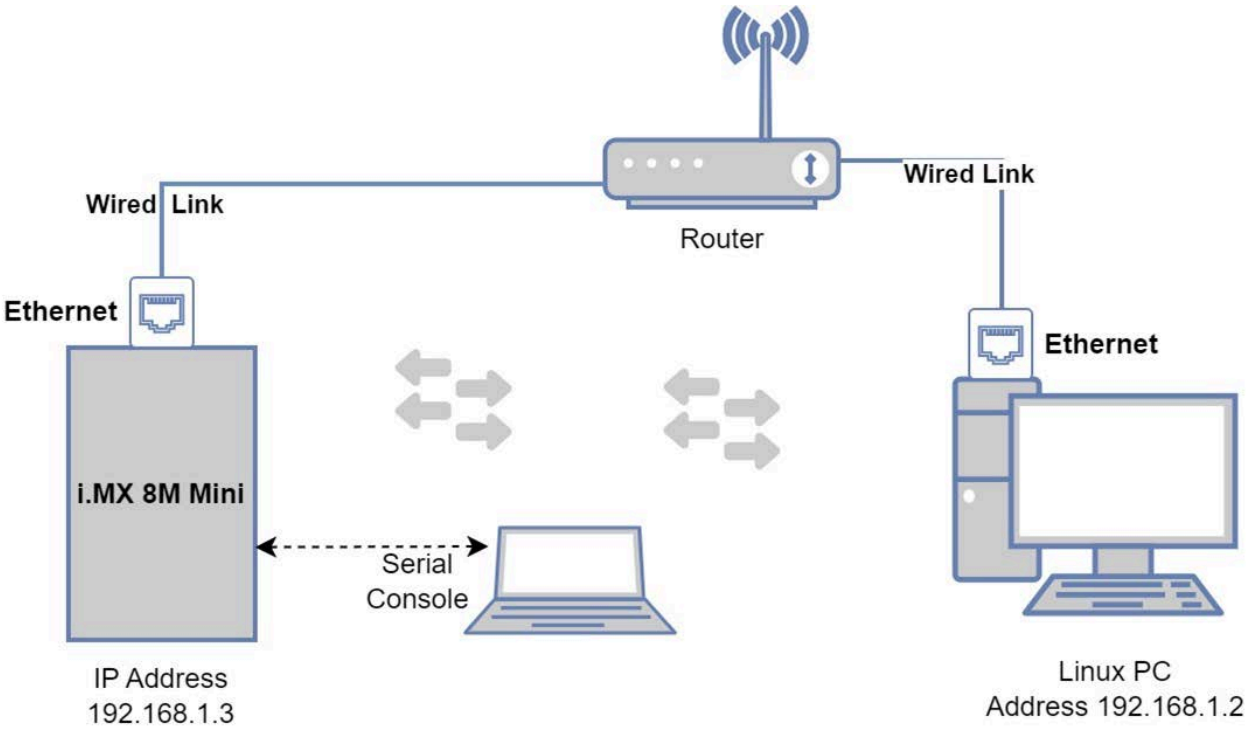

Figure 45. Setup to copy ZibgBee application to i.MX 8M Mini platform

- Get the IP address of i.MX 8M Mini board.

```
# ifconfig
```

- Transfer the `simple_gw` compiled binary from the Linux PC to i.MX 8M Mini.

```
# sudo IMX_IPADDR=<IP Address of i.MX 8M Mini> make deploy
```

Example of command output:

```
nxp_ae@95LD823-server:~/.../simple_gw$ sudo IMX_IPADDR=10.10.0.215 make deploy
# test_ip_addr:
IMX addr IMX_IPADDR: 10.10.0.215
ping -c 1 10.10.0.215
PING 10.10.0.215 (10.10.0.215) 56(84) bytes of data.
64 bytes from 10.10.0.215: icmp_seq=1 ttl=64 time=3.33 ms
--- 10.10.0.215 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.330/3.330/3.330/0.000 ms
# clean_previous:
The authenticity of host '10.10.0.215 (10.10.0.215)' can't be established.
ECDSA key fingerprint is SHA256:Oo/uPyDxq/WqfYArnB5Y4RdHuj8QtCkan1TjCpmTMRA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
WarningPermanently added '10.10.0.215' (ECDSA) to the list of known hosts.
# deploy_prebuild:
Copy on IMX the mux & run script
zb_mux        100%  131KB   5.4MB/s   00:00
imx-dualpan.sh      100%   25KB   1.4MB/s   00:00
Create on IMX the folder ~/zboss-dual-pan/prebuild
Copy on IMX the prebuild binaries
Bulb        100% 1156KB  13.8MB/s   00:00
gp_proxy       100% 1091KB  48.2MB/s   00:00
light_control       100%  836KB  60.9MB/s   00:00
light_zc       100% 1155KB  65.6MB/s   00:00
on_off_distrib_output_zr     100% 1156KB  40.2MB/s   00:00
on_off_distrib_switch_zed     100% 1156KB  65.3MB/s   00:00
on_off_output_zc      100% 1156KB  53.6MB/s   00:00
on_off_switch_zed      100%  836KB  70.3MB/s   00:00
ota_client_zr       100% 1156KB  57.2MB/s   00:00
ota_server_zc       100% 1156KB  64.9MB/s   00:00
scenes_zc       100% 1156KB  70.8MB/s   00:00
scenes_zed       100%  836KB  55.5MB/s   00:00
simple_gw       100% 1156KB  65.1MB/s   00:00
thermostat_zc       100% 1156KB  20.9MB/s   00:00
thermostat_zr       100% 1157KB  42.7MB/s   00:00
# deploy_compiled:
Create on IMX the folder ~/zboss-dual-pan/compiled
Copy on IMX the compiled binaries
simple_gw       100% 1156KB  15.1MB/s   00:00
```

Step 10 – Run the example for Zigbee-only mode over zb_mux.

Command syntax:

```
./imx-dualpan.sh --ch <channel> --zb <zb_app> [--fw <IW612-firmware>]
```

Table 23. Command parameters

Parameter	Description
--ch <channel>	OpenThread & Zigbee channel to use
--ot <ot appli>	Openthread application path and name (uart-hdlc)
--zb <zb appli>	Zigbee application path and name
--fw <firmware>	Firmware file to download (optional)
--help	Print this message

Example of command:

```
# ./imx-dualpan.sh --ch 25 --zb ./prebuild/simple_gw --fw /lib/firmware/nxp/
sduart_nw61x_v1.bin.se
```

The command starts Zigbee-only mode on channel 25 on the IW612 device.

Example of command output:

```
root@imx8mmevk:~/zboss-dual-pan# ./imx-dualpan.sh --ch 25 --zb ./prebuild/simple_gw --
fw /lib/firmware/nxp/sduart_nw61x_v1.bin.se
me:       ./imx-dualpan.sh
channel:  25
first:    zb_app
second:
zb_app:   ./prebuild/simple_gw
ot_app:
firmware: /lib/firmware/nxp/sduart_nw61x_v1.bin.se
run_level in release mode
Hardware config: i.MX8MM
Flash /lib/firmware/nxp/sduart_nw61x_v1.bin.se with w_loader_imx_lnx on /dev/ttymxc2
Push Reset button of device [enter] ?
Protocol: NXP Proprietary
FW Loader Version: M322
ComPort : /dev/ttymxc2
BaudRate: 115200
FlowControl: 0
Filename: /lib/firmware/nxp/sduart_nw61x_v1.bin.se
Second BaudRate: 3000000
ChipID is : 7601, Version is : 0
File downloaded:   936668: 1038668
Download Complete
time:4342
CTS is low
Wait for the firmware to start...
Device setup complete
bring up hci0
Can't init device hci0: Connection timed out (110)
Start zb_mux
[08:20:14.958] zb_mux: ZBOSS HOST Version RELEASE 018-052424-1b70caa
[08:20:14.958] zb_mux: TRACE_LEVEL: 0
[08:20:14.958] zb_mux: TRACE_MASK: 0
[08:20:14.960] zb_mux: dev init /tmp/ttyOpenThread nli: 0
[08:20:14.961] zb_mux: dev init /tmp/ttyZigbee nli: 2
[08:20:14.961] zb_mux: configure spi speed 1000000
Factory reset for simple_gw [y/n] ([y] or [Enter]) ? y
Factory reset: simple_gw.nvram is removed
Start Zigbee ./prebuild/simple_gw [enter] ?
[08:20:22.607] simple_gw: ZBOSS HOST Version RELEASE 018-052424-1b70caa
[08:20:22.607] simple_gw: TRACE_LEVEL: 4
[08:20:22.607] simple_gw: TRACE_MASK: 0x00000800
[08:20:22.609] simple_gw: Configure channel 25 with env MACSPLIT_CHANNEL
[08:20:22.609] simple_gw: device_type: 0: COORDINATOR, channel_mask: 02000000
[08:20:22.620] simple_gw: signal 23: ZDO PRODUCTION_CONFIG_READY, status 255
[08:20:22.705] zb_mux: dev connect /tmp/ttyZigbee nli: 2
[08:20:22.706] zb_mux: configure spi speed 1000000
[08:20:22.755] simple_gw: signal 43: MACSPLIT_DEVICE_BOOT, status 0
[08:20:22.755] simple_gw: signal 01: ZDO SKIP_STARTUP, status 0, []
[08:20:23.565] simple_gw: signal 59: NWK PERMIT_JOIN_STATUS, status 0, [duration: 0]
[08:20:23.617] simple_gw: signal 05: BDB DEVICE_FIRST_START, status 0, []
[08:20:24.069] simple_gw: signal 59: NWK PERMIT_JOIN_STATUS, status 0, [duration: 180]
[08:20:24.118] simple_gw: signal 10: BDB STEERING, status 0, []
```

9 Abbreviations

Table 24. Abbreviations

Abbreviation	Definition
AP	Access point
BSP	Board support package
BT	Bluetooth
DTB	Device tree blob
EVK	Evaluation kit
FW	Firmware
STA	Station
uSD	Micro SD
WLAN	Wireless local area network
WPA	Wi-Fi protected access

10 References

[1] Data sheet – 88W8801: 2.4 GHz Single-band 1x1 Wi-Fi 4 Solution (link)
[2] Data sheet – 88W8987: 2.4/5 GHz Dual-Band 1x1 Wi-Fi 5 and Bluetooth Solution (link)
[3] Data sheet – 88W9098: Wi-Fi 6 2x2 Concurrent Dual Wi-Fi (CDW) and Bluetooth Combo SoC (link)
[4] Data sheet – AW693: 2x2 Dual-band (5-7 GHz) Concurrent Dual Wi-Fi 6/6E, 1x1 (2.4 GHz) Wi-Fi 6, and Bluetooth Combo Solution (link)
[5] Data sheet – IW416: Dual-band 1x1 Wi-Fi 4 and Bluetooth Combo SoC (link)
[6] Data sheet – IW610: 1x1 Wi-Fi 6 and Bluetooth Low Energy/802.15.4 Solution Family (link)
[7] Data sheet – IW612: 1x1 Dual-band Wi-Fi 6, Bluetooth and 802.15.4 Tri-radio Solution (link)
[8] Data sheet – AzureWave – AW-CM358SM – IEEE 802.11a/b/g/n/ac WLAN with Bluetooth 5 Combo Stamp LGA Module (link)
[9] Data sheet – AzureWave – AW-CM276MA-PUR – IEEE 802.11a/b/g/n/ac Wireless LAN 2T2R and Bluetooth 5.0 Combo Module (M.2 2230) (link)
[10] Data sheet – AzureWave – AW-CM276MA-SUR – IEEE 802.11a/b/g/n/ac Wireless LAN 2T2R and Bluetooth 5.0 Combo Module (M.2 2230) (link)
[11] Data sheet – AzureWave – AW-AM510MA – IEEE 802.11a/b/g/n Wireless LAN + Bluetooth 5. 1 Combo Module(M.2 2230) (link)
[12] Data sheet – Murata – LBEE5ZZ1XL module specification (link)
[13] Data sheet – Murata – LBWA0ZZ2DS module specification (link)
[14] Fact sheet – 88W8997: 802.11ac wave 2 2x2 Wi-Fi Dual-band with Bluetooth SoC (link)
[15] User guide – UG10164: i.MX Yocto Project User's Guide (link)
[16] Reference manual – RM00297: Linux Software Reference Manual for Wireless Connectivity (link)
[17] Webpage – 88W8801: 2.4 GHz Single-Band 1x1 Wi-Fi® 4 (802.11n) Solution (link)
[18] Webpage – 88W8987: 2.4/5 GHz Dual-Band 1x1 Wi-Fi® 5 (802.11ac) + Bluetooth® Solution (link)
[19] Webpage – 88W8997: 2.4/5 GHz Dual-Band 2x2 Wi-Fi® 5 (802.11ac) + Bluetooth® Solution (link)
[20] Webpage – 88W9098: 2.4/5 GHz Dual-Band 2x2 Wi-Fi® 6 (802.11ax) + Bluetooth® (link)
[21] Webpage – IW416: 2.4/5 GHz Dual-Band 1x1 Wi-Fi® 4 (802.11n) + Bluetooth® Solution (link)
[22] Webpage – IW610: 2.4/5 GHz Dual-band 1x1 Wi-Fi® 6 + Bluetooth Low Energy + 802.15.4 Tri-Radio Solution (link)
[23] Webpage – IW612: 2.4/5 GHz Dual-Band 1x1 Wi-Fi® 6 (802.11ax) + Bluetooth® + 802.15.4 Tri-Radio Solution (link)
[24] Webpage – Panasonic PAN9028: Mid Range All Round Solution (link)

11 Note about the source code in the document

The example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2020-2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

12 Contact information

Use the following links for more product details, queries and support.

Web support: nxp.com/support

NXP community: community.nxp.com

iMX community: community.nxp.com/community/imx

13 Revision history

Revision history

Document ID	Date	Description
UM11483 v.19.0	23 September 2025	**NXP-based wireless modules** • Section 3.7.7 "i.MX 8M Quad rework for SDIO support on M.2": updated the intriduciton and the note. **Bring up Bluetooth** • Section 6.5 "Configure BTNXPUART driver for high-speed communication": added.
UM11483 v.18.0	27 June 2025	**NXP-based wireless modules** • Section 3.17 "IW610-based Murata module LBES0ZZ2LL": added USB-USB host interface configuration and Section 3.17.1. • Section 3.18 "AW693-based u-blox module JODY-W683": added. **Bring-up of Wi-Fi** • Section 5.1 "Bring-up of 88W8987-based wireless module (AW-CM358MA)": updated the instructions. • Section 5.2 "Bring-up of 88W8997-based wireless module (AW-CM276MA-PUR)": updated the instructions. • Section 5.3 "Bring-up of 88W8997-based AzureWave module (AW-CM276MA-SUR and AW-CM276-uSD)": updated the instructions. • Section 5.4 "Bring-up of 88W9098-based Murata module LBEE6ZZ1": updated the instructions. • Section 5.5 "Bring-up of 88W9098-based Murata module LBEE5ZZ1XL": updated the instructions • Section 5.6 "Bring-up of IW416-based AzureWave module AW-AM510MA": updated the instructions • Section 5.7 "Bring-up of 88W8801-based Murata module LBWA0ZZ2DS": updated the instructions • Section 5.8 "Bring-up of IW612-based Murata module LBES5PL2EL": updated the instructions • Section 5.9 "Bring-up of IW610-based Murata module LBES0ZZ2LL": added the information for USB-USB host interface configuration. • Section 5.10 "Bring-up of AW693-based u-blox module JODY-W6": added. **Bring-up of Bluetooth** • Section 6.3 "Bring-up using USB (IW610-based module only)": added. Section 10 "References": updated.
UM11483 v.17.0	26 March 2025	• Section 6.4 "UART baudrate detection and adaption (IW611 and IW612 only) ": added.
UM11483 v.16.0	17 December 2024	• Section 10 "References": added IW610 webpage and data sheet. • Section 5.8 "Bring-up of IW612-based Murata module LBES5PL2EL": updated the last step to get the driver version. • Section 3.17 "IW610-based Murata module LBES0ZZ2LL": added. • Section 5.9 "Bring-up of IW610-based Murata module LBES0ZZ2LL": added. • Section 7.2 "Bring-up of a Thread Network": updated **Step 3**.
UM11483 v.15.0	25 September 2024	• Section 8 "Bring-up of ZigBee on i.MX 8M Mini": added.
UM11483 v.14.0	27 March 2024	• Section 7.3 "Build the Kernel image and drivers to enable OTBR support for i.MX 8M Mini platform": added. • Section 7.4 "Bring up of an OTBR agent": added.
UM11483 v.13.0	13 December 2023	• Section 3.2 "Wi-Fi MM driver architecture": added the section.

Revision history...*continued*

Document ID	Date	Description
UM11483 v.12.0	25 September 2023	• Section 7.2 "Bring-up of a Thread Network": added **step 4** and **step 5**.
UM11483 v.11.0	26 June 2023	• Table 2 "Boot mode switch settings": updated the format • Table 3 "Switch settings to boot from eMMC": updated the format • Figure 9 "802.15.4 software architecture": updated • Table 5 "Jumper settings for AW-CM358-uSD module": added • Table 8 "Functional blocks": updated • Section 3.8.5.2 "Resistor configurations": renamed jumper as resistor • Table 13 "Jumper settings for AW-CM276-uSD module": added • Section 4.3 "Flashing the image to eMMC": updated • Section 5.8.1 "Bring up the Wi-Fi interface": updated the format • Section 6 "Bring-up of Bluetooth": updated • Section 7 "Bring-up of 802.15.4": updated the note at the beginning of the section • Section 11 "Note about the source code in the document": added
UM11483 v.10.0	30 March 2023	• Section 10 "References": added references for IW612 • Section 3 "NXP-based wireless modules": added IW612 • Section 3.5 "OpenThread radio coprocessor (RCP) software architecture": added • Section 3.16 "IW612-based Murata module LBES5PL2EL": added the section • Section 5.8 "Bring-up of IW612-based Murata module LBES5PL2EL": added the section • Section 6 "Bring-up of Bluetooth": updated • Section 7 "Bring-up of 802.15.4": added the section
UM11483 v.9.0	15 December 2022	• Section 6 "Bring-up of Bluetooth": updated the note for 88W9098-based modules
UM11483 v.8.0	29 September 2022	• Section 4.3 "Flashing the image to eMMC": updated UUU version number • Section 6 "Bring-up of Bluetooth": updated the second note with a reference to IW416-based module
UM11483 v.7.0	29 June 2022	• Section 10 "References": added the reference to Panasonic PAN9028 webpage. • Section 3.8 "88W8987-based Panasonic PAN9028 module (SDIO-UART)": added the section • Section 6 "Bring-up of Bluetooth": updated the note about the modules based on 88W9098 and IW416.

Revision history...*continued*

Document ID	Date	Description
UM11483 v.6.0	29 March 2022	Extended the scope to 88W9098-based module with SDIO host interface for Wi-Fi (LBEE5ZZ1XL) in addition to other changes: • Section 10 "References": added the reference to LBEE5ZZ1XL module data sheet • Section *Wi-Fi driver and firmware support on past Linux BSP releases*: added the information on Linux 5.10.72_2.2.0 BSP release • Section 3 "NXP-based wireless modules": added the host interface configuration to section headings • Section 3.13 "88W9098-based Murata module LBEE5ZZ1XL (SDIO-UART)": added • Section 4.3 "Flashing the image to eMMC": updated UUU version • Section 5.4 "Bring-up of 88W9098-based Murata module LBEE6ZZ1": updated the content of the configuration file for dual MAC • Section 5.5 "Bring-up of 88W9098-based Murata module LBEE5ZZ1XL": added • Section 6 "Bring-up of Bluetooth": updated the second note
UM11483 v.5.0	14 December 2021	Extended the scope to 88W8801-based wireless module • Section 10 "References": updated • Section 3.15 "88W8801-based Murata LBWA0ZZ2DS Wi-Fi module (SDIO)": added • Section 5.7 "Bring-up of 88W8801-based Murata module LBWA0ZZ2DS": added Additional changes: • Section *Wi-Fi driver and firmware support on past Linux BSP releases*: updated
UM11483 v.4.0	21 September 2021	Extended the scope to IW416-based wireless module: • Section 10 "References": updated • Section 3 "NXP-based wireless modules": updated • Section 3.14 "IW416-based AzureWave AW-AM510MA module (SDIO-UART)": added • Section 4.4.2 "Enabling SDIO on M.2 connector": added • Section 5.6 "Bring-up of IW416-based AzureWave module AW-AM510MA": added • Section 6 "Bring-up of Bluetooth": updated Other modifications: • Section *Wi-FI driver and firmware aupport on past Linux BSP releases*: added • Replaced the sections *Setting up the host* and *Building the image* with Section 4.2 "Building the image from source" • Section 4.3 "Flashing the image to eMMC": updated *UUU usage* paragraph • Section 4.4.3 "Enabling PCIe on M.2 (88W9098-based Murata module LBEE6ZZ1)": updated

Revision history...*continued*

Document ID	Date	Description
UM11483 v.3.0	12 June 2021	• Section 10 "References": updated • Extended the scope to 88W9098-based wireless module: – Section 3.12 "88W9098-based Murata module LBEE6ZZ1 (PCIe-UART)": added – Section 4.4.3 "Enabling PCIe on M.2 (88W9098-based Murata module LBEE6ZZ1)": added – Section 5.4 "Bring-up of 88W9098-based Murata module LBEE6ZZ1": added • Extended the scope to 88W8997-based wireless modules – Section 3.11 "88W8997-based AzureWave AW-CM276-uSD Wi-Fi module (SDIO)": added – Section 3.10 "88W8997-based AzureWave module AW-CM276MA-SUR (SDIO-UART)": added – Section 5.3 "Bring-up of 88W8997-based AzureWave module (AW-CM276MA-SUR and AW-CM276-uSD)": added • Section 3.7.6 "Enabling SDIO on M.2 (AW-CM358MA and AW-CM276MA-SUR)": updated • Section 6 "Bring-up of Bluetooth": updated
UM11483 v.2.0	21 January 2021	• Extended the scope to 88W8987- and 88W8997-based wireless modules • Updated the document title and overall document structure
UM11483 v.1.0	15 October 2020	Initial version

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Suitability for use in automotive applications — This NXP product has been qualified for use in automotive applications. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

Matter, Zigbee — are developed by the Connectivity Standards Alliance. The Alliance's Brands and all goodwill associated therewith, are the exclusive property of the Alliance.

Tables

Figures

Contents