Qualcomm Linux Wi-Fi Guide

80-70018-14 AB

April 10, 2025

Contents

1 Wi-Fi overview

Qualcomm® Linux® provides Wi-Fi functionality, features, and configurable parameters for developing applications.

Wi-Fi is a wireless networking technology that uses the IEEE 802.11 protocol. It lets electronic devices like smartphones, wearables, laptops, desktops, and other consumer electronics to connect to the Internet without physical cables.

In Qualcomm Linux, Wi-Fi functionality is enabled through the ath11k driver. The system-on-chip (SoC), Wi-Fi chipset, and driver support for the development kits are as follows.

Development kit	Hardware SoCs	Wi-Fi chipset	Driver support
• Qualcomm RB3 Gen 2 Vision Development Kit • Qualcomm RB3 Gen 2 Core Development Kit	• QCS6490	• WCN6750 • WCN6856	• ath11k
• Qualcomm RB3 Gen 2 Lite Vision Development Kit • Qualcomm RB3 Gen 2 Lite Core Development Kit	• QCS5430	• WCN6750 • WCN6856	• ath11k

Development kit	Hardware SoCs	Wi-Fi chipset	Driver support
• Qualcomm Dragonwing™ IQ-9075 Evaluation Kit (EVK)	• QCS9075	• NFA765A Wi-Fi module over M.2	• ath11k
• Qualcomm IQ-8 Beta EVK	• QCS8275	• QCA6698AQ	• ath11k

The Qualcomm connectivity chipset has various components, as shown in the following figure.

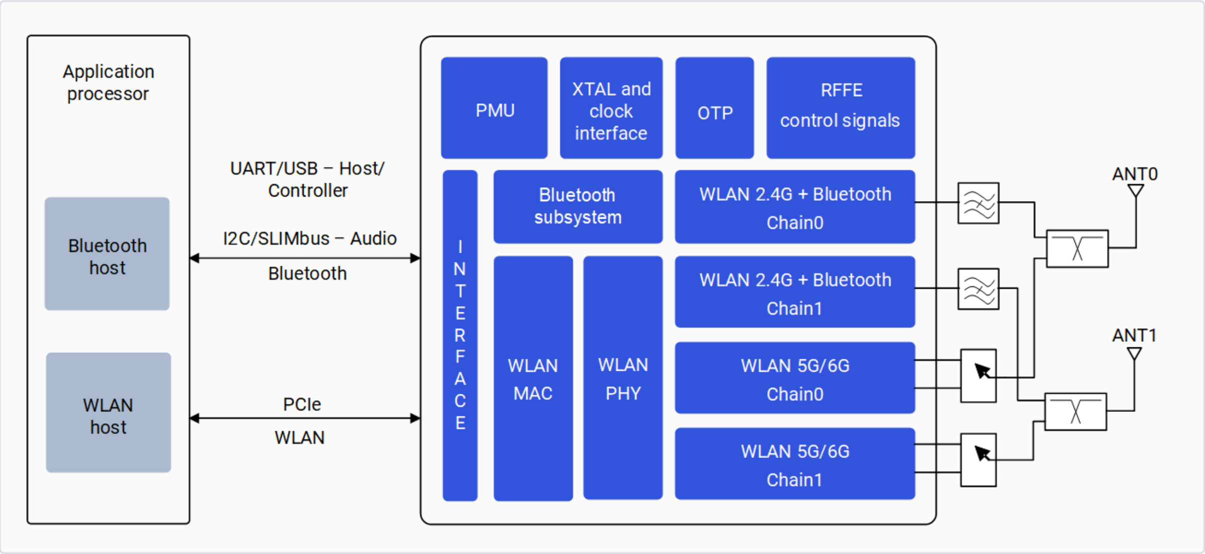


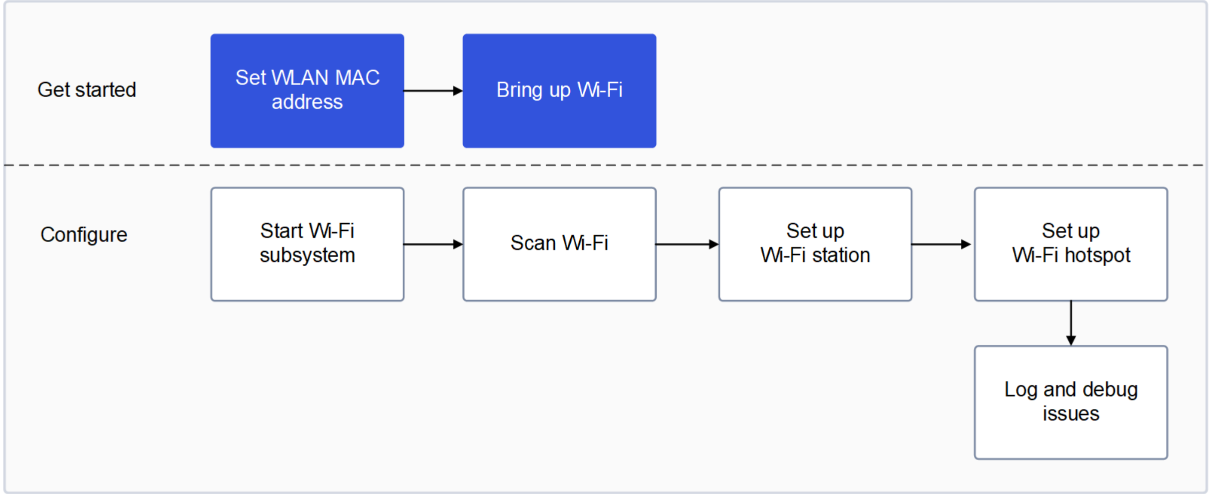
Figure1 Figure : Qualcomm connectivity chipset block diagram

2 Get started with Wi-Fi

2.1 Prerequisites

Before you begin, see Qualcomm Linux Build Guide for common infrastructure setup and build workflows.

This information describes the prerequisites to bring up WLAN functionality, such as setting up a Secure Socket Shell (SSH) connection and verifying and storing the Media Access Control (MAC) address. The following workflow shows how to get started with the Wi-Fi bring up procedure followed by configuring Wi-Fi according to your requirements.



Set up UART/SSH

To set up debug Universal Asynchronous Receiver/Transmitter (UART) and Wi-Fi, and to connect to SSH, see Sign in using SSH.

Note: When using the Secure Copy Protocol (SCP) to transfer files to or from a device, a password prompt appears. Enter `oelinux123` in the password field.

Set WLAN MAC address

Note: This is an optional step. If you want to have your own MAC address and not the default MAC address, perform the following steps:

- By default, the factory writes the WLAN MAC address in the one-time programmable (OTP) memory of the network card.

  You can modify the MAC address using the following commands, if required. In these commands, replace `wlanX` with the interface index name 0 or 1. For example, `wlan0` or `wlan1`.

  ```
  ifconfig wlanX down
  ifconfig wlanX hw ether xx:xx:xx:xx:xx:xx
  ifconfig wlanX up
  ```

- The MAC address that you set persists until a restart or the next power cycle and takes precedence over the factory-set MAC address stored in OTP memory.

3 Wi-Fi features

The Qualcomm Linux Wi-Fi software provides support for various Wi-Fi features through the ath11k driver. This driver uses the mac80211 API and lets the 802.11ax protocol.

ath11k supports the following features:

- 2.4 GHz, 5 GHz, and 6 GHz Wi-Fi bands
- Peak PHY data rate of 2.9 Gbps, 1 K QAM
- Station (STA) mode and Access Point (AP) mode

3.1 Wi-Fi capabilities

The following table provides a feature matrix of the Wi-Fi capabilities that WCN6750, WCN6856, and QCA6698AQ support.

Table : Wi-Fi software feature matrix

Feature	Description	WCN6750	WCN6856	QCA6698AQ
Wi-Fi band	2.4 GHz, 5 GHz, and 6 GHz	✓	✓	✓
Operational mode	• Wi-Fi STA • Wi-Fi SoftAP	✓	✓	✓
Maximum bandwidth	• Up to 40 MHz channel bandwidth for 2.4 GHz • Up to 160 MHz channel bandwidth for 5 GHz and 6 GHz	✓	✓	✓

Feature	Description	WCN6750	WCN6856	QCA6698AQ
Dual Band Simultaneous (DBS)	• DBS/Non-DBS • Maximum stream and bandwidth supported	• Non-DBS • 2-stream (2 x 2) 2.4 GHz 40 MHz + (2 x 2) 5 GHz/6 GHz 160 MHz	• DBS • 4-stream (2 x 2) 2.4 GHz 40 MHz + (2 x 2) 5 GHz/6 GHz 160 MHz	• DBS • 4-stream (2 x 2) 2.4 GHz 40 MHz + (2 x 2) 5 GHz/6 GHz 160 MHz
Unrestricted 160 MHz channels supported	Seven channels	✓	✓	✓
Peak PHY data rate	2.9 Gbps, 1 K QAM	✓	✓	✓

Feature	Description	WCN6750	WCN6856	QCA6698AQ
802.11ax	• Uplink/downlink (UL/DL) Multiple-User Multiple-Input and Multiple-Output (MU-MIMO) • Uplink Orthogonal Frequency Division Multiple Access (UL/DL OFDMA), Uplink Orthogonal Frequency Division Multiple Access Random Access (UL-OFDMA UORA) • Multiple- Basic Service Set Identifier (BSSID) and Multiple-Traffic Identifier (TID) • Spatial reuse • 8-stream sounding • Target Wake Time (TWT)	✓	✓	✓
Security	Wi-Fi Protected Access 3 (WPA3)	✓	✓	✓
WFA mandatory certifications[1]	• STA mode • AP mode	✓	✓	✓

3.2 Operating bands

The WCN6750, WCN6856, and QCA6698AQ Wi-Fi chipsets support 2.4 GHz, 5 GHz, and 6 GHz operating bands.

3.3 Operating modes

The Wi-Fi software operates in the following modes.

[1] For more information about the list of WFA mandatory certifications, see https://www.wi-fi.org/certification and https://www.wi-fi.org/certification/programs.

Table : Supported Wi-Fi operating modes

Mode	Description
STA mode	In STA mode, a device connects to an AP within a Wi-Fi network and communicates with other devices in the network. This mode is standard for wireless devices in a Wi-Fi connection.
Hotspot mode	Hotspot mode enables a device to provide backhaul (Internet) connectivity to Wi-Fi clients using a cellular link (LTE). The device establishes this connection through its lightweight hotspot interface. In hotspot mode, the device can: • Communicate with other Wi-Fi clients connected to the same hotspot. • Communicate with the hotspot device. • Share the WAN connection of the device.

3.4 Scan

A Wi-Fi scan compiles a list of nearby Access Points (APs) for a device. There are two primary scan modes: active and passive. Additionally, WLAN chipsets can trigger other scan policies.

The following table provides the types of scans and their descriptions.

Table : Supported scan modes

Scan	Description
Active scan	The Wi-Fi initiates a broadcast probe request (Tx) and listens for probe responses from APs. All APs, except for those with a hidden Service Set Identifier (SSID), respond to the broadcast probe request. Active scans are used on all channels for 2.4 GHz, non-Dynamic Frequency Selection (DFS) channels for 5 GHz, and Preferred Scanning Channel (PSC) for 6 GHz.
Passive scan	The client conserves power by not transmitting packets acitvely. The STA waits on each channel for approximately 100 ms (plus an additional ±10 ms for channel change) to listen to beacons broadcasted by APs. During this dwell time, the STA receives all beacons, and scans SSIDs from the APs on that channel.
Legacy scan	Scans one channel at a time.
Split scan	Alternates scanning between the home channel and foreign channel.
Burst scan	Scans multiple channels in a burst.
Agile scan	Simultaneously scans two passive channels reducing DFS channel scan time by half.

Scan	Description
Roaming scan	Initiates the move of the STA from a congested channel of the currently connected AP to a better AP. Triggers include Received Signal Strength Indicator (RSSI), missed beacons, channel traffic conditions, and AP conditions.

3.5 DBS operation

WLAN uses multiple MAC addresses available within the chipset for DBS. The WCN6856 and QCA6698AQ chips support DBS, allowing simultaneous scanning at both 2.4 GHz and 5 GHz frequencies.

Additionally, a scan channel can run in parallel with other Wi-Fi operations on different bands. This concurrent operation involves two 802.11 MAC/Physical (PHY)/radio pipes functioning on separate radio bands: 5 GHz, 6 GHz (PHY A), and 2.4 GHz (PHY B).

3.6 Coexistence

The 2.4 GHz Industrial, Scientific, and Medical (ISM) band is shared by Bluetooth®, Wi-Fi, and Mobile Wireless Standards (MWS) Long Term Evolution (LTE). Coexistence software monitors the states of Wi-Fi, Bluetooth, and LTE to determine the coordination methodology that best aligns with expectations for each link. For example, it considers quality of service (QoS) for Bluetooth links and throughput for Wi-Fi.

4 Wi-Fi software architecture

The Wi-Fi software architecture comprises the following components.

Table : Wi-Fi software components

Component	Description
User space	• Facilitates Wi-Fi parameter configuration and setup. • Supports factory testing and debugging.
Kernel space	• Facilitates communication between the driver and the application layers.
Driver	• Facilitates communication between the operating system (OS) and the wireless network hardware. • Initializes the firmware and downloading Board Data Files (BDF) required to bring up the WLAN RF. • Manages the control path and data path required for the 802.11 protocol.
Firmware	• A software component embedded in the Wi-Fi hardware device. • Manages low-level communication with the hardware. • Handles 802.11 protocol-based tasks like scanning for networks, establishing connections, and transmitting data over Wi-Fi hardware.

4.1 ath11k driver software architecture

The following figure shows the high-level Wi-Fi software architecture with ath11k driver:

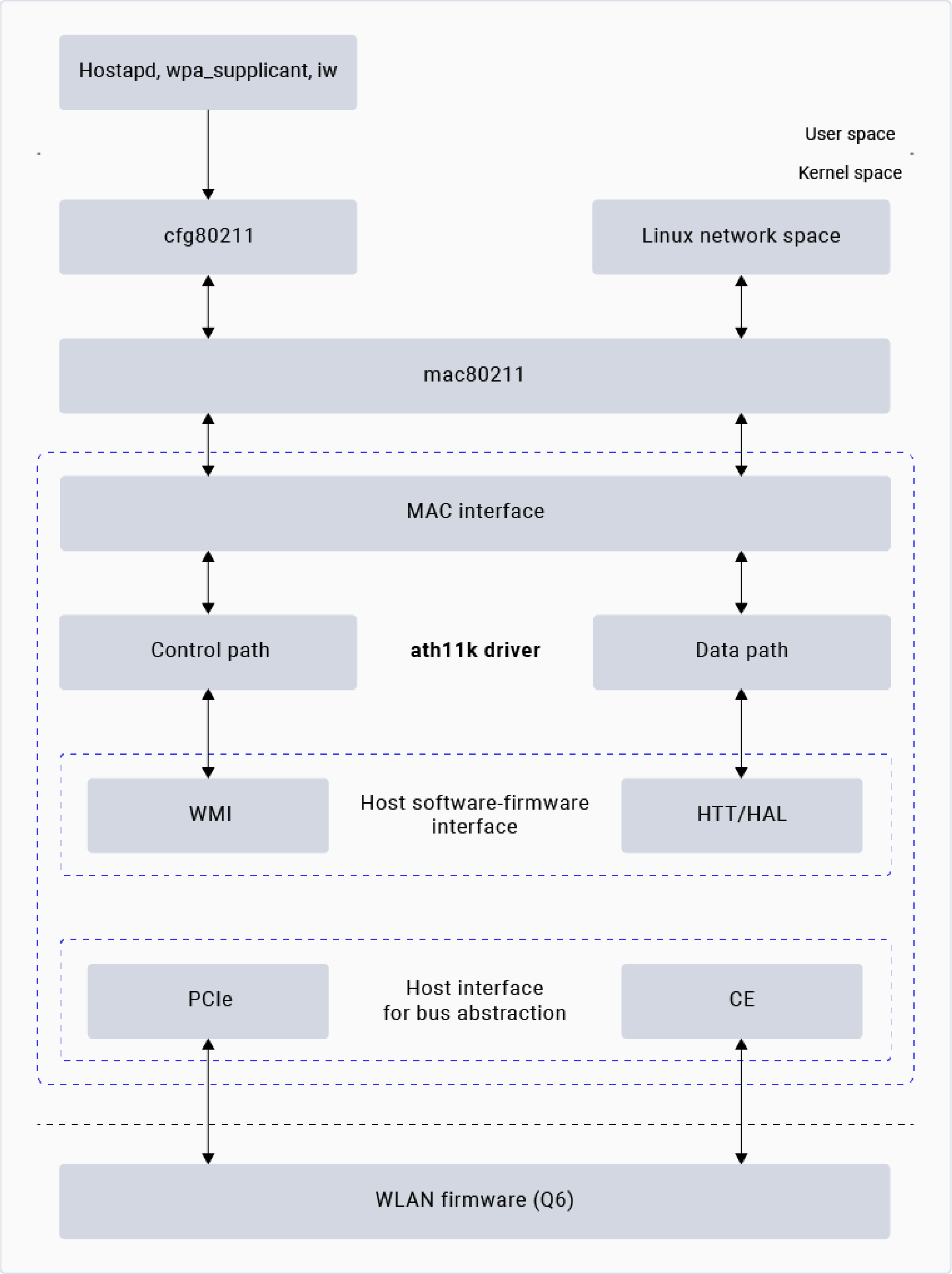


Figure1 Figure : Wi-Fi software architecture with ath11k driver

User space components

The user space contains the following key components:

Table : User space components

Component	Description
User space	• Facilitates Wi-Fi parameter configuration and setup. • Supports factory testing and debugging.
Kernel space	• Facilitates communication between the driver and the application layers.
Driver	• Facilitates communication between the operating system (OS) and the wireless network hardware. • Initializes the firmware and downloading Board Data Files (BDF) required to bring up the WLAN RF. • Manages the control path and data path required for the 802.11 protocol.
Firmware	• A software component embedded in the Wi-Fi hardware device. • Manages low-level communication with the hardware. • Handles 802.11 protocol-based tasks like scanning for networks, establishing connections, and transmitting data over Wi-Fi hardware.

Kernel space components

The kernel space contains the following key components:

Table : Kernel space components

Component	Description
cfg80211	• Manages the configuration of wireless devices. • Interfaces with mac80211, which uses the hardware-specific lower level ath11k driver.
Network stack	• A network stack provided by the OS. • Supports functions to manage the driver and handle various TCP/IP protocols as data flows to the driver.
mac80211	• Implements open-source WLAN protocols as part of the Linux kernel. • Handles the registration and configuration of the network subsystem through cfg80211. cfg80211 implements

Component	Description

ath11k driver components

The ath11k driver contains the following key components:

Table : ath11k driver components

Component	Description
MAC interface	• Acts as an interface between mac80211 and lower layers of ath11k driver. • Interfaces with mac80211 through ath11k_ops. • Sends data and management frames to the data path. • Sends configuration commands to the WMI path.
Control path	• Performs operations such as scan, authentication, and association.
Data path	• Manages the data traffic transmitted and received by WLAN. • Supports various features to ensure optimal performance.
Host software-firmware interface	• Includes WMI, HTT, and HAL.
WMI/HTT	• Implements high-level protocols between the host driver and firmware running on WLAN Qualcomm Hexagon™ Digital Signal Processor (Hexagon) on the SoC. • Typically, the WMI interface is used for control path transactions, and HTT is used for data path transactions.
HAL	• Abstracts hardware data structures and implements the recipe for direct MMIO-based register access and memory access. • Depends on the hardware or chip. • Interfaces between data path driver and WLAN hardware in a highly optimized packet path implementation.

Component	Description
Host interface for bus abstraction	• Provides host-target bus abstraction services. • Abstracts the peripheral control interface express (PCIe).
PCIe	• A hardware interface in the driver.
Copy engine (CE)	• Implements hardware pipe, Standard Ring Interface (SRNG) based communication channel between the host driver and firmware running on the CE interface over PCIe.

Firmware components

The WLAN firmware runs on a dedicated processor (Q6). The WLAN firmware binary contains the following key components:

- Various 802.11 protocols and the statemachine of software modules
- Performance-related software algorithms like rate adaptation, coexistence, MIMO, and OFDMA
- Core MAC hardware programming recipes
- Uniform MAC/PHY service interface to upper layers
- Core PHY and RF hardware programming recipes. For example, RF calibration, BDF, and RADAR.

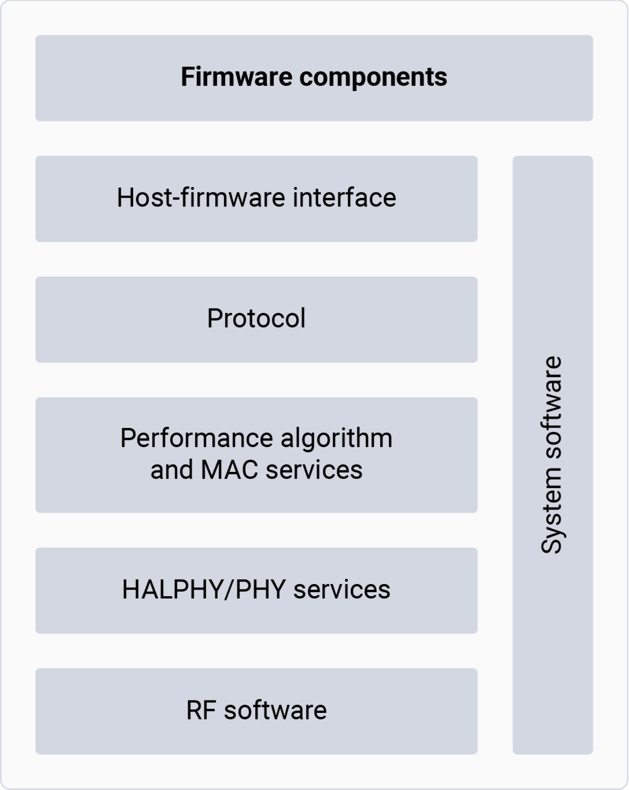


Figure2 Figure : Firmware components

The following table provides the list of firmware components and their description.

Table : Firmware components

Component	Description
Host-firmware I/F	• Wireless Module Interface (WMI) – Control plane API definitions between host and firmware • Host Target Interface (HTT) – Data plane Application Programming Interface (API) definitions between host and firmware
Protocol	• Implements various 802.11 standards protocols and state machines, offloads, host command, scan algorithms and event handling
Performance algorithm and MAC services	• Performance algorithm implements performance related software algorithms like rate adaptation, transmit scheduler, Multi-User (MU) scheduler, MU grouping, Resource Unit (RU) allocator, Air Time Fairness (ATF)and hardware IP specific: Tx and Rx Data path, MAC control path sequences • MAC register programming and descriptor setup/completion, coexistence hardware recipes, chip power hardware recipes
HAL PHY	Common software infrastructure for PHY/RFA hardware and HAL-PHY interface to firmware.
PHY/RF software	• PHY/RFA hardware component access interface implementation modules containing hardware programming sequences/recipes specific to a given PHY/RFA and platform hardware IP
System software	• Implements all platform specific SoC related services

5 Wi-Fi software

The following table lists the source code locations of the ath11k driver components.

Table : Source code location of Wi-Fi software components

Component	Source code location
mac80211	`net/mac80211`
ath11k driver	• `drivers/net/wireless/ath/ath11k` • https://github.com/torvalds/linux/tree/master/drivers/net/wireless/ath/ath11k
wpa_supplicant	`source:SRC_URI = "git://w1.fi/hostap.git;protocol=https;branch=main"`
Supplicant configuration file	`sources/mdm-init/upstream`

For more information about ath11k, see:

- ath11k: https://wireless.docs.kernel.org/en/latest/en/users/drivers/ath11k.html
- mac80211: https://wireless.wiki.kernel.org/en/developers/documentation/mac80211
- cfg80211: https://wireless.wiki.kernel.org/en/developers/documentation/cfg80211

5.1 CLD driver

By default, Qualcomm Linux lets WLAN functionality through the ath11k WLAN driver across all chipsets. However, in RB3 Gen 2, you have the option to use the Convergence Linux Driver (CLD), which offers value-added features such as Wi-Fi Direct mode and 4K QAM.

To debug any issues related to the CLD driver, see Logs for CLD driver.

The following table lists the source code location of software components with the CLD driver:

Component	Source code location
Wi-Fi subsystem platform driver (icnss)	`sources/wlan/platform/`
Qualcomm Wi-Fi host driver	• `sources/wlan/qcacld-3.0` • `sources/wlan/qca-wifi-host-cmn`
wpa_supplicant	`sources/wlan/wpa_supplicant_8`
Supplicant configuration file	`sources/mdm-init/wlan_qcm6490/`

Set WLAN MAC address in software with CLD driver

Consider the following points about the WLAN MAC address:

- By default, the factory writes the WLAN MAC address in the OTP memory of the network card.
- You can modify the MAC address, if required using the following procedure. The MAC address that you set persists until a restart or the next power cycle. It takes precedence over the factory-set MAC address stored in OTP memory.
- If no factory-set MAC address is available, the software generates a random MAC address during each boot cycle.

Before you begin, ensure that the device is connected to the host over UART as described in Connect to a UART shell.

To set a unique WLAN MAC address in software with a CLD driver, perform the following steps in the debug UART serial console:

1. Allow read/write access by running the following command:

   ```
   mount -o rw,remount /
   ```

2. Create a `wlan_mac.bin` file in the `/lib/firmware/wlan/qca_cld/` directory by running the following commands:

   ```
   cd /lib/firmware/wlan/qca_cld/
   ```

   Note: A default `wlan_mac.bin` file is unavailable.

3. Open the `wlan_mac.bin` file in a text editor.

   For example, to open the `wlan_mac.bin` file in VI editor, run the following command:

```
vi /lib/firmware/wlan/qca_cld/wlan_mac.bin
```

4. Enter the MAC address in the `wlan_mac.bin` file in the following format:

   ```
   Intf0MacAddress=<IP_0>
                       Intf1MacAddress=<IP_1>
                       Intf2MacAddress=<IP_2>
                       Intf3MacAddress=<IP_3>
                       END
   ```

   For example,

   ```
   Intf0MacAddress=000AF58989FF
                           Intf1MacAddress=000AF58989FE
                           Intf2MacAddress=000AF58989FD
                           Intf3MacAddress=000AF58989FC
                           END
   ```

5. Save the `wlan_mac.bin` file.

6. Get read/write/executable permission for the `wlan_mac.bin` file by running the following command:

   ```
   chmod 777 wlan_mac.bin
   ```

   Sample output

   ```
   root@qcm6490:/# mount -o rw, remount /
   root@qcm6490:/# cd /lib/firmware/wlan/qca_cld/
   root@qcm6490:/lib/firmware/wlan/qca_cld# vi wlan mac.bin
   root@qcm6490:/lib/firmware/wlan/qca _cld# chmod 777 wlan_mac.bin
   root@qcm6490:/lib/firmware/wlan/qca_cld# cat wlan_mac.bin
   Intf(MacAddress=000AF58989FF
   IntflMacAddress=000AF58989FE
   Intf2MacAddress=000AF58989FD
   Intf3MacAddress=000AF58989FC
   END
   root@qcm6490:/lib/firmware/wlan/qca_cld#
   ```

7. Add the `read_mac_addr_from_mac_file=1` INI parameter in the `/lib/firmware/wlan/qca_cld/WCNSS_qcom_cfg.ini` file.

   For more information about updating the INI parameter, see Customize Wi-Fi parameters in software with CLD driver.

8. Restart the device.

Customize Wi-Fi parameters in software with CLD driver

You can customize the CLD Wi-Fi host driver with various configurations using the `WCNSS_qcom_cfg.ini` file.

The Wi-Fi host driver reads this INI file during its initialization.

Any change made to the INI file reflects only after the device restarts. The INI file used by the Wi-Fi host driver is available in the `/lib/firmware/wlan/qca_cld/WCNSS_qcom_cfg.ini` path.

To customize the Wi-Fi parameters according to your requirements, perform the following steps:

Note: Ensure that you enable SSH by following the steps mentioned in Use SSH.

1. Run the SSH using the following command:

   ```
   ssh root@<device_IP_address>
   ```

   For example, if the IP address of the device is `192.168.1.22`, run the following command:

   ```
   ssh root@192.168.1.22
   ```

   ```
   mount -o remount rw /
   ```

   Note: Ensure that the RB3 Gen 2 device and the peer client device are in the Wi-Fi vicinity.

2. To pull the default `WCNSS_qcom_cfg.ini` file from the device, run the following commands:

   ```
   scp -r root@192.168.1.22:/lib/firmware/wlan/qca_cld/WCNSS_qcom_
   cfg.ini .
   ```

3. Modify the `WCNSS_qcom_cfg.ini` file.

4. To push the `WCNSS_qcom_cfg.ini` file to the device, run the following commands:

   ```
   scp -r WCNSS_qcom_cfg.ini  root@192.168.1.22:/lib/firmware/wlan/
   qca_cld
   reboot
   ```

5. Power cycle the device.

   The information of all the INI configurations is available at `sources/wlan/qcacld-3.0/core/hdd/inc/hdd_config.h`. Use the following configurations for testing:

   - `gDot11Mode=8` for 11ac only mode support.

- `gEnableBmps=0` to disable beacon mode power save (bmps).
- `gEnableDFSChnlScan=0` to avoid scanning the DFS channel.

Switch to CLD driver

Note: The following step is applicable only when the Qualcomm RB3 Gen 2 Development Kit has a WCN6750 Wi-Fi chipset.

To switch from ath11k to CLD driver, use the following two patches.

- Apply the following patch, for which the source file is at `sources/wlan/wlan-devicetree`:

```
diff --git a/qcm6490-wlan-idp.dtsi b/qcm6490-wlan-idp.dtsi
index 5a50eda..bb15319 100644
--- a/qcm6490-wlan-idp.dtsi
+++ b/qcm6490-wlan-idp.dtsi
@@ -60,7 +60,6 @@
        qcom,vdd-1.3-rfa-config = <1256000 1500000 0 0 0>;
        qcom,smem-states = <&wlan_smp2p_out 0>;
        qcom,smem-state-names = "wlan-smp2p-out";
-       status = "disabled";

        icnss_cdev_apss: qcom,icnss_cdev1 {
                #cooling-cells = <2>;
diff --git a/qcm6490-wlan-rb3.dtsi b/qcm6490-wlan-rb3.dtsi
index a716fa2..57f5637 100644
--- a/qcm6490-wlan-rb3.dtsi
+++ b/qcm6490-wlan-rb3.dtsi
@@ -60,7 +60,6 @@
        qcom,vdd-1.3-rfa-config = <1256000 1500000 0 0 0>;
        qcom,smem-states = <&wlan_smp2p_out 0>;
        qcom,smem-state-names = "wlan-smp2p-out";
-       status = "disabled";

        icnss_cdev_apss: qcom,icnss_cdev1 {
                #cooling-cells = <2>;
diff --git a/qcm6490-wlan-upstream.dtsi b/qcm6490-wlan-upstream.dtsi
```

```
index b7f219a..7f81991 100644
--- a/qcm6490-wlan-upstream.dtsi
+++ b/qcm6490-wlan-upstream.dtsi
@@ -4,10 +4,5 @@
  */
 &wifi {
        memory-region = <&wlan_fw_mem>, <&wlan_ce_mem>;
-       status = "okay";
-};
-
-&remoteproc_wpss {
-       firmware-name = "ath11k/WCN6750/hw1.0/wpss.mdt";
-       status = "okay";
+       status = "disabled";
 };
diff --git a/qcs5430-wlan-rb3.dtsi b/qcs5430-wlan-rb3.
dtsi
index 0dab35c..f50a644 100644
--- a/qcs5430-wlan-rb3.dtsi
+++ b/qcs5430-wlan-rb3.dtsi
@@ -60,7 +60,6 @@
        qcom,vdd-1.3-rfa-config = <1256000 1500000 0 0 0>
;
        qcom,smem-states = <&wlan_smp2p_out 0>;
        qcom,smem-state-names = "wlan-smp2p-out";
-       status = "disabled";

        icnss_cdev_apss: qcom,icnss_cdev1 {
                #cooling-cells = <2>;
diff --git a/qcs5430-wlan-upstream.dtsi b/qcs5430-wlan-
upstream.dtsi
index b7f219a..7f81991 100644
--- a/qcs5430-wlan-upstream.dtsi
+++ b/qcs5430-wlan-upstream.dtsi
@@ -4,10 +4,5 @@
  */
 &wifi {
        memory-region = <&wlan_fw_mem>, <&wlan_ce_mem>;
-       status = "okay";
-};
-
-&remoteproc_wpss {
-       firmware-name = "ath11k/WCN6750/hw1.0/wpss.mdt";
-       status = "okay";
+       status = "disabled";
 };
```

- Revert the following patch, for which the source file is at `sources/mdm-init/wlan_qcm6490`

```
diff --git a/wlan_qcm6490/wlan b/wlan_qcm6490/wlan
index 54059ee..d23fdfd 100644
--- a/wlan_qcm6490/wlan
+++ b/wlan_qcm6490/wlan
@@ -32,7 +32,11 @@
# SPDX-License-Identifier: BSD-3-Clause-Clear
export MODULE_BASE=/lib/modules/`uname -r`
-export MODNAME=wlan
+#To load qcacld driver make MODNAME=wlan
+#export MODNAME=wlan
+
+#MODNAME=ath11k_ahb indicates upstream ath11k driver is
being loaded
+export MODNAME=ath11k_ahb
export RETRY_LIMIT=20
HELP="Usage $0 {start | stop | restart} <ap | sta,ap>"
DUMP_TO_KMSG=/dev/kmsg
@@ -65,7 +69,6 @@
    wpa_supplicant -Dnl80211 -iwlan0 -ddd -c /etc/wpa_
supplicant.conf -t -f /tmp/wpa_supplicant-log.txt &
-   dhcpcd wlan0
         cnt=0
         while ! [ -d /var/run/wpa_supplicant ]
@@ -100,8 +103,9 @@
     stop)
         echo "Stopping WLAN..." > $DUMP_TO_KMSG
    killall wpa_supplicant
-   killall dhcpcd
-   rmmod $MODNAME
+        if [ "$MODNAME" = "wlan" ]; then
+            rmmod $MODNAME
+        fi
         ;;
     restart)
```

After applying the patches, ensure that you recompile the entire image and flash it as described in the Qualcomm Linux Build Guide.

6 Configure Wi-Fi

This information describes the setup and configuration procedures required to establish a Wi-Fi connection.

6.1 Start the Wi-Fi subsystem

If the device is started in STA mode, Wi-Fi is active by default, that is, the Wi-Fi host driver and wpa_supplicant are launched during the boot process of the device.

1. Ensure that the device is connected to the host over UART by performing the steps listed in Connect to a UART shell.
2. To confirm if the Wi-Fi host driver is loaded, run the following command on the debug UART console:

```
iw dev
```

3. Search for the `wlan0` interface in the output:

Sample output

```
root@qcm6490:~# iw dev
phy#0
Interface wlan0
ifindex 5
wdev 0x1
addr 12:78:04:00:ae:15
type managed
channel 60 (5300 MHz), width: 80 MHz, center1: 5290 MHz
txpower 20.00 dBm
multicast TXQ:
qsz-byt qsz-pkt flows drops marks overlmt hashcol tx-bytes   tx-
packets
 0       0       0     0     0     0       0        0         0
```

4. To verify if wpa_supplicant is active, run the following command:

```
ps -ef | grep wpa_supplicant
```

The following output indicates that the wpa_supplicant has been successfully enabled:

```
root        1415       1  0 00:00 ?        00:00:00 wpa_
supplicant -Dnl80211 -iwlan0 -ddd -c /etc/wpa_supplicant.conf -f
/tmp/wpa_supplicant-log.txt
```

6.2 Scan for Wi-Fi access points

A Wi-Fi scan finds the APs listed in the vicinity that must be initiated through the `nmcli` command line tool.

To initiate a Wi-Fi scan using `nmcli` run the following command:

```
nmcli dev wifi list 2>&1 | less
```

Sample output

```
IN-USE  BSSID              SSID           MODE   CHAN  RATE
      SIGNAL  BARS SECURITY
        20:DB:AB:98:57:EE  JioPrivateNet  Infra  9     130
Mbit/s  89          ****             WPA2 802.1X
        20:DB:AB:98:57:EF  JioNet         Infra  9     130
Mbit/s  89          ****              --
        20:DB:AB:98:57:E1  JioPrivateNet  Infra  44    270
Mbit/s  80          ***              WPA2 802.1X
        20:DB:AB:98:57:E0  JioNet         Infra  44    270
Mbit/s  77          ***               --
        F0:61:C0:FB:96:A1  QGuest         Infra  11    260
Mbit/s  75          ***               --
        F0:61:C0:FB:96:A0  Hydra          Infra  11    260
Mbit/s  75          ***              WPA2
        20:DB:AB:9E:CA:CF  JioNet         Infra  5     130
Mbit/s  72          ***           --
        F0:61:C0:FB:96:B0  Hydra          Infra  140   540
Mbit/s  72          ***              WPA2
        F0:61:C0:FB:96:B1  Pandora        Infra  140   540
Mbit/s  72          ***              WPA2 802.1X
        F0:61:C0:FB:96:B2  QGuest         Infra  140   540
Mbit/s  72          ***               --
        20:DB:AB:9E:CA:CE  JioPrivateNet  Infra  5     130
Mbit/s  69          ***
```

Note: To exit from the standard input shell, press **CTRL+C**.

6.3 Set up a Wi-Fi station

In the STA mode, the RB3 Gen 2 device operates as a client and it can connect to a wireless hotspot or router.

Establish a wireless connection through the `nmcli` command line tool.

To set up the Wi-Fi in STA mode, perform the following steps:

1. Ensure that the device is connected to the host over UART by performing the steps listed in Connect to a UART shell.

2. To establish a connection to an AP, run the following command:

```
nmcli dev wifi connect <WiFi-SSID> password <WiFi-password>
```

For example, run the command

```
root@qcs6490-rb3gen2-vision-kit:~# nmcli dev wifi connect
QualcommWiFi password 1234567890
```

where, the Wi-Fi SSID is `QualcommWiFi` and the Wi-Fi password is `1234567890`.

Sample output

```
Device 'wlan0' successfully activated with 'df4250eb-45f6-4ce2-
bd90-a2513e016536'
```

Note: If you see a network error message while running the command, then run one of the following commands to trigger the Wi-Fi scan and verify the intended AP.

```
nmcli dev wifi list
```

```
iw dev wlan0 scan
```

3. To verify the connection state, run the following command:

```
root@qcs6490-rb3gen2-vision-kit:~# nmcli general status
```

Sample output

```
STATE  CONNECTIVITY  WIFI-HW  WIFI    WWAN-HW  WWAN
connected  full     enabled  enabled  enabled  enabled
```

4. To verify the connection status, perform the following steps:

   a. To view the device status, run the following command:

   ```
   root@qcs6490-rb3gen2-vision-kit:~# nmcli dev status
   ```

   Sample output

   ```
   DEVICE  TYPE      STATE        CONNECTION
   wlan0   wifi      connected    QualcommWiFi
   eth0    ethernet  unavailable  --
   eth1    ethernet  unavailable  --
   can0    can       unmanaged    --
   lo      loopback  unmanaged    --
   ```

   b. To view the additional connection information, run the following command:

   ```
   nmcli device show wlan0
   ```

   Sample output

   ```
   GENERAL.DEVICE:                         wlan0
   GENERAL.TYPE:                           wifi
   GENERAL.HWADDR:                         00:03:7F:12:F7:F7
   GENERAL.MTU:                            1500
   GENERAL.STATE:                          100 (connected)
   GENERAL.CONNECTION:                     QualcommWiFi
   GENERAL.CON-PATH:                       /org/freedesktop/
   NetworkManager/ActiveConnection/5
   IP4.ADDRESS[1]:                         192.168.117.130/24
   IP4.ADDRESS[2]:                         192.168.117.131/24
   IP4.GATEWAY:                            192.168.117.126
   IP4.ROUTE[1]:                           dst = 192.168.117.0/
   24, nh = 0.0.0.0, mt = 3005
   IP4.ROUTE[2]:                           dst = 192.168.117.0/
   24, nh = 0.0.0.0, mt = 600
   IP4.ROUTE[3]:                           dst = 0.0.0.0/0, nh =
   192.168.117.126, mt = 3005
   IP4.DNS[1]:                             192.168.117.126
   IP6.ADDRESS[1]:                         2401:4900:658c:d8b0:
   3a86:b071:fd59:7ade/64
   IP6.ADDRESS[2]:                         2401:4900:658c:d8b0:
   ```

```
37d5:d37f:675c:3313/64
IP6.ADDRESS[3]:                         fe80::c930:1be0:3ac0:
496c/64
IP6.ADDRESS[4]:                         fe80::28a6:3dc0:f535:
75f1/64
IP6.GATEWAY:                            fe80::14c1:74ff:feef:
f40f
IP6.ROUTE[1]:                           dst = fe80::/64, nh =
::, mt = 256
IP6.ROUTE[2]:                           dst = fe80::/64, nh =
::, mt = 1024
IP6.ROUTE[3]:                           dst = 2401:4900:658c:
d8b0::/64, nh = ::, mt = 3005
IP6.ROUTE[4]:                           dst = ::/0, nh =
fe80::14c1:74ff:feef:f40f, mt = 3005
IP6.DNS[1]:                             2401:4900:658c:d8b0::
dc
standard input
```

c. Verify if the IP address is assigned on the `wlan0`interface by running the `ifconfigwlan0` command in the UART console.

```
root@qcs6490-rb3gen2-vision-kit:~# ifconfig wlan0
```

Sample output

```
wlan0     Link encap:Ethernet  HWaddr 00:03:7F:12:F7:F7
inet addr:192.168.117.131  Bcast:192.168.117.255  Mask:255.
255.255.0
inet6 addr: 2401:4900:658c:d8b0:3a86:b071:fd59:7ade/64 Scope:
Global
inet6 addr: fe80::c930:1be0:3ac0:496c/64 Scope:Link
inet6 addr: fe80::28a6:3dc0:f535:75f1/64 Scope:Link
inet6 addr: 2401:4900:658c:d8b0:37d5:d37f:675c:3313/64 Scope:
Global
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:1365 errors:0 dropped:0 overruns:0 frame:0
TX packets:1141 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:127462 (124.4 KiB)  TX bytes:160302 (156.5 KiB)
```

d. Verify if the AP or router is connected to the Internet by pinging the following public DNS IP:

```
root@qcs6490-rb3gen2-vision-kit:~# ping 8.8.8.8
```

Sample output

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=55 time=174 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=55 time=96.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=55 time=84.8 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time
2003ms
rtt min/avg/max/mdev = 84.809/118.472/173.684/39.352 ms
```

Note: If you are connected to one network, but want to use a different connection, you can disconnect by switching the connection to down by specifying the SSID `nmcli con down ssid/uuid`. Alternatively, if you have multiple connections with the same SSID, use the UUID.

Note: To connect to another saved connection, simply pass the up option in the `nmcli` command line tool by running the following command.

```
nmcli con up ssid/uuid
```

Ensure that you specify the SSID or UUID of the new network that you want to connect to.

Note: To exit from the standard input shell, press **CTRL+C**.

For more information about `nmcli`, see https://www.linux.org/docs/man1/nmcli.html and https://networkmanager.dev/docs/api/latest/nmcli.html.

6.4 Set up a Wi-Fi hotspot

Ensure that you enable SSH by following the steps listed in Sign in using SSH.

Set up a Wi-Fi hotspot in SoftAP mode by performing the following steps:

Run the SSH using the following command:

```
ssh root@<device_IP_address>
```

For example, if the IP address of the device is `192.168.1.22`, run the following command:

```
ssh root@192.168.1.22
```

Note: If there is a WCN6750 chipset, ensure that the Wi-Fi SoftAP operating channel configured on the Qualcomm RB3 Gen 2 Development Kit is the same as the one on the Wi-Fi STA operating channel (excluding the DFS channel).

Note: Use either STA or SAP with a DFS channel because STA+SAP SCC DFS concurrency is currently unavailable.

By default the `hostapd.conf` file is available at `/etc` location. This file configures the RB3 Gen 2 device as a Wi-Fi SoftAP operational mode with SSID as `QSoftAP` and password as `1234567890.`

1. Add the Wi-Fi SoftAP operational mode interface by running the following command:

   ```
   iw dev wlan0 interface add wlan1 type managed
   ```

2. Reconfigure Wi-Fi SoftAP operational mode by performing the following steps:

   a. Open a new command prompt/terminal.

   b. Pull the default `hostapd.conf` file from the device by running the following command:

   ```
   scp -r root@<IP_address>:<source_file_path> <destination_
   file_path>
   ```

   To pull a file to the current file path, enter the `<destination_file_path>` as `.` in the command.

   For example, the IP address of the device is `192.168.1.22`, to pull `hostapd.conf` from `/etc`, run the following command:

   ```
   scp -r root@192.168.1.22:/etc/hostapd.conf .
   ```

   c. Update the `hostapd.conf` file with the required SSID, password, and relevant hostapd configurations.

   Note: Update `hw_mode` accurately while modifying the Wi-Fi channel. For more information about hostapd configuration, see https://w1.fi/cgit/hostap/plain/hostapd/README.

d. Push the file to the device by running the following command:

```
scp -r hostapd.conf root@192.168.1.22:/etc
```

e. To verify that the configuration parameters are updated, run the following command in the device for the `hostapd.conf` file:

```
<cat /etc/hostapd.conf>
```

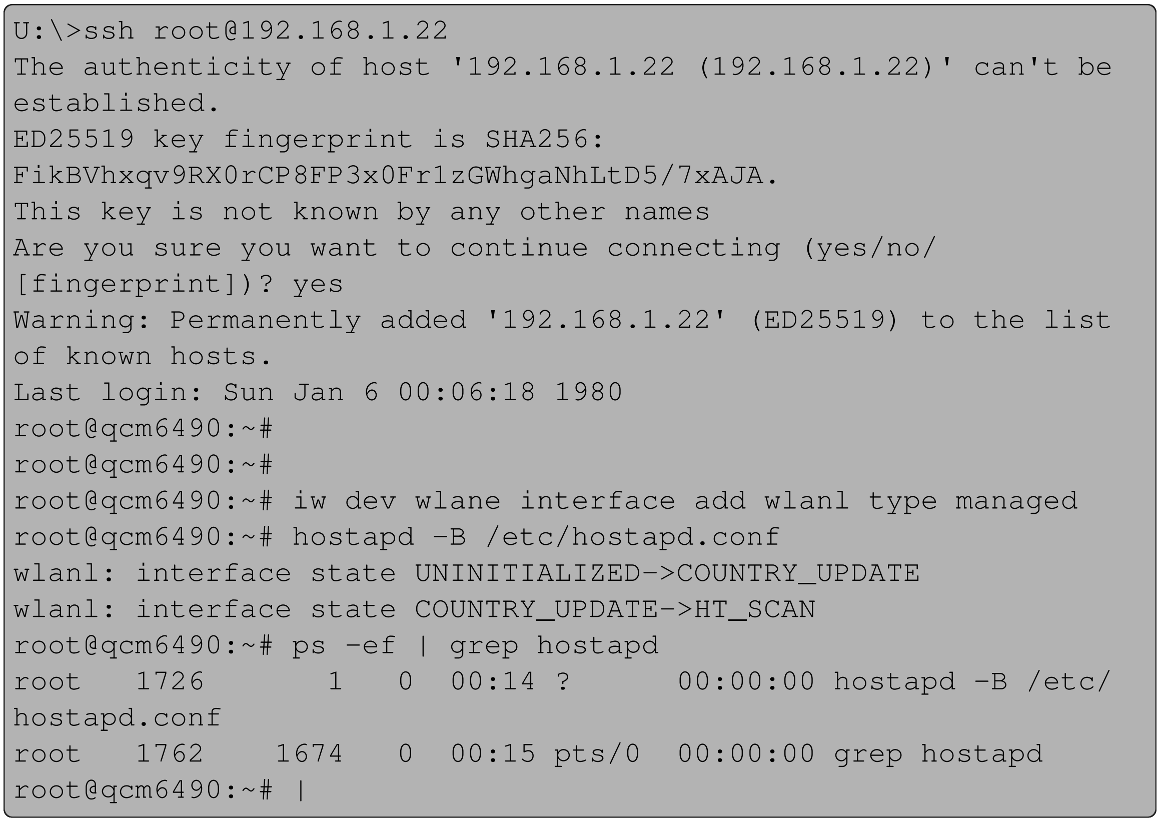
3. Set a MAC address for `wlan1` on the Qualcomm RB3 Gen 2 Development Kit that has a WCN6750 Wi-Fi chipset. The following example can be used as a reference while setting up a MAC address.

```
ifconfig wlan1 hw ether 00:03:7F:12:B5:B6
```

4. To set up the SoftAP, run the following command in the `ssh shell`:

```
hostapd -B /etc/hostapd.conf
```

The following output indicates that the AP is set up:



```
U:\>ssh root@192.168.1.22
The authenticity of host '192.168.1.22 (192.168.1.22)' can't be
established.
ED25519 key fingerprint is SHA256:
FikBVhxqv9RX0rCP8FP3x0Fr1zGWhgaNhLtD5/7xAJA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/
[fingerprint])? yes
Warning: Permanently added '192.168.1.22' (ED25519) to the list
of known hosts.
Last login: Sun Jan 6 00:06:18 1980
root@qcm6490:~#
root@qcm6490:~#
root@qcm6490:~# iw dev wlane interface add wlanl type managed
root@qcm6490:~# hostapd -B /etc/hostapd.conf
wlanl: interface state UNINITIALIZED->COUNTRY_UPDATE
wlanl: interface state COUNTRY_UPDATE->HT_SCAN
root@qcm6490:~# ps -ef | grep hostapd
root    1726       1   0  00:14 ?      00:00:00 hostapd -B /etc/
hostapd.conf
root    1762    1674   0  00:15 pts/0  00:00:00 grep hostapd
root@qcm6490:~# |
```

5. To start the Dynamic Host Configuration Protocol (DHCP) server on the Wi-Fi hotspot interface, run the following commands:

   ```
   brctl addbr br0
   ```

   ```
   brctl addif br0 wlan1
   ```

   ```
   ifconfig br0 192.168.225.1 netmask 255.255.255.0 up
   ```

   ```
   killall dnsmasq
   ```

   ```
   dnsmasq --conf-file=/etc/dnsmasq.conf --dhcp-leasefile=/var/run/dnsmasq.leases --addn-hosts=/data/hosts --pid-file=/var/run/dnsmasq.pid -i br0 -I lo -z --dhcp-range=br0,192.168.225.20,192.168.225.60,255.255.255.0,43200 --dhcp-hostsfile=/data/dhcp_hosts --dhcp-option-force=6,192.168.225.1 --dhcp-script=/bin/dnsmasq_script.sh
   ```

6. To establish a connection with the `hostapd_cli`, run the following command in SSH:

   ```
   hostapd_cli -i wlan1 -p /var/run/hostapd
   ```

   Monitor the `hostapd_cli` console for the Wi-Fi STA connection notifications such as `AP-STA-CONNECTED`, `EAPOL-4WAY-HS-COMPLETED`.

   Sample output

   ```
   root@qcm6490:~# hostapd_cli -i wlanl -p /var/run/hostapd
   hostapd_cli v2.11-devel
   Copyright (c) 2004-2022, Jouni Malinen <j@wl.fi> and contributors
   This software may be distributed under the terms of the BSD License.
   See README for more details.

   Interactive mode
   > <3>AP-STA-CONNECTED aa: a4: fd: 8b: ec: 90
   <3>EAPOL-4WAY-HS-COMPLETED aa: a4: fd: 8b:ec:90

   > list_sta
   aa: a4: fd: 8b:ec:90
   ```

7. To verify the connection status, connect to SoftAP from other devices.

   For example, connect to SoftAP from a mobile device by performing the following steps:

a. On the mobile device, go to the Wi-Fi settings.

b. Wait for the Wi-Fi STA to detect SoftAP (with the SSID **QSoftAP**).

c. Select SoftAP and enter the corresponding `wpa_passphrase` that was configured for SoftAP on the RB3 Gen 2 device, then connect.

Sample output

```
> status
state=ENABLED
phy=phyR freq=2412
num_sta_non_erp=0
num_sta_no_short_slot_time=0
num_sta_no_short_preamble=0
olbc=0
num_sta_ht_no_gf=0 num_sta_no_ht=0
num_sta_ht_20_mhz=0
num_sta_ht40_intolerant=0
olbc_ht=0
ht_op_mode=0x0
hw_mode=g
country_code=US
country3=0x20
cac_time_seconds=0
cac_time_left_seconds=N/A
channel=1
edmg_enable=0 edmg_channel=0
secondary_channel=0
ieee80211n=1
ieee80211ac=0
ieee80211ax=0
ieee80211be=0
beacon_int=100
dtim_period=2
ht_caps_info=000c
ht_mcs_bitmask=ffff0000000000000000
supported_rates-02 04 0b 16 Oc 12 18 24 30 48 60 6c
max_txpower=30
bss[0]=wlan1
bssid[0]=00:03:7f:95:8e:8e
ssid [0]=QSoftAP
num_sta[0]=1
> |
```

d. To verify the connection, ping the IP address of the mobile from the RB3 Gen 2 device in the `ssh shell`.

The following output indicates that the Wi-Fi connection has been established successfully and the data transfer has begun:

```
sh-5.1# ping 192.168.1.42
PING 192.168.1.42 (192.168.1.42): 56 data bytes
64 bytes from 192.168.1.42: seq=0 ttl=64 time-11.175 ms
64 bytes from 192.168.1.42: seq=1 ttl=64 time=14.528 ms
64 bytes from 192.168.1.42: seq=2 ttl=64 time=29.735 ms
64 bytes from 192.168.1.42: seq=3 ttl=64 time=223.822 ms
64 bytes from 192.168.1.42: seq-4 ttl=64 time-23.675 ms
^C
192.168.1.42 ping statistics ---
7 packets transmitted, 5 packets received, 28% packet loss
round-trip min/avg/max = 11.175/60.587/223.822 ms
sh-5.1#
```

Alternatively, you can verify the Wi-Fi connection status in the **Settings** of the connected device. For example, to get the IP address of a mobile device connected to the RB3 Gen 2 SoftAP, perform the following steps:

i. Go to **Settings** > **Wi-Fi**.

ii. Verify the SSID of the AP.

6.5 Stop a Wi-Fi hotspot

Note: Ensure that the Wi-Fi hotspot has been set up.

To stop the hotspot, do the following in SSH:

1. Stop the hostapd by performing the following steps:

   a. To stop the hostapd process, run the following command:

   ```
   killall hostapd
   ```

   b. To disable the interface, run the following command:

   ```
   ifconfig wlan1 down
   ```

2. To delete `ctrl_interface`, run the following command:

```
rm -rf /var/run/hostapd/wlan1
```

The Wi-Fi hotspot stops successfully.

7 Debug Wi-Fi issues

If you experience issues with configuring or using Wi-Fi features, you can report them to Qualcomm. You can also collect logs from various modules for troubleshooting purposes.

7.1 Report Wi-Fi issues

To report an issue to Qualcomm, provide the following details.

- Provide setup information

Information type	**Description**
Platform information	Platform name and software version
Wi-Fi AP information	Model name, channel/mode/bandwidth/security, and firmware version
Related logs	For more information, see Logs for ath11k.
Test setup, steps, and expected result	Test procedures used while reporting the issue.

- Package all log files for debugging and report them to Qualcomm.

  The following table lists the logs required to report an issue.

Log name	**Protocol issues**	**Throughput issues**
Host driver logs	Required	Required
Firmware logs	Required	Required
Wireless sniffer logs	Recommended	Required
Supplicant logs	Required	Required

Logs for ath11k

The Wi-Fi subsystem offers tools and logs to assist with troubleshooting issues.

7.2 Pull Wpa_supplicant logs

If you observe any issue, run the following command to collect wpa_supplicant logs:

```
scp -r root@<ip_addr>:/tmp/wpa_supplicant-log.txt .
```

For example, if the IP address of the DUT is `192.168.1.22`, run the following command:

```
scp -r root@192.168.1.22:/tmp/wpa_supplicant-log.txt .
```

7.3 Pull Wi-Fi host driver and firmware logs

To pull the Wi-Fi host driver and firmware logs for the ath11k driver, do the following:

1. Run the `trace-cmd` tool on the DUT as follows:

   - For the entire log, run the following command:

     ```
     trace-cmd record -e ath11k_htt_pktlog -e ath11k_htt_ppdu_
     stats -e ath11k_htt_rxdesc -e ath11k_log_err -e ath11k_log_
     warn -e ath11k_log_info -e ath11k_wmi_cmd -e ath11k_wmi_event
     -e ath11k_log_dbg -e ath11k_log_dbg_dump -e ath11k_wmi_diag -
     e ath11k_ps_timekeeper
     ```

   - For firmware log, run the following command:

     ```
     trace-cmd record -e ath11k_wmi_diag
     ```

2. Run the intended test case or reproduce the issue.

3. Stop `trace-cmd` by pressing **Ctrl + C**.

   The `trace.dat` log appears as follows:

   Sample output

   ```
   root@qcm6490:/tmp#
   root@qcm6490:/tmp# trace-cmd record -e ath1lk_htt_pktlog -e
   ath1lk_htt_ppdu_stats -e athllk_htt_rxdesc -e ath11k_log_err -e
   ath11k_log_warn -e athllk_log_info
    -e ath1lk_wmi_cmd -e ath11k_wmi_event -e ath1lk_log_dbg -e
   ```

```
ath1lk_log_dbg_dump -e ath11k_wmi_diag -e athllk_ps_timekeeper
Hit Ctrl^C to stop recording
^CCPUO data recorded at offset=0x7fc000
151552 bytes in size
CPU1 data recorded at offset=0x821000
163840 bytes in size
CPU2 data recorded at offset=0x849000
0 bytes in size
CPU3 data recorded at offset=0x849000
0 bytes in size
CPU4 data recorded at offset=0x849000
0 bytes in size
CPU5 data recorded at offset=0x849000
0 bytes in size
CPU6 data recorded at offset=0x849000
0 bytes in size
CPU7 data recorded at offset=0x849000
0 bytes in size
root@qcm6490:/tmp# ls -al
ls: cannot access 'ssgtzd': Permission denied
total 8600
drwxrwxrwt. 10 root  root  280 Sep 16  06:21 .
drwxr-xr-x. 22 root  root  4096 Sep  9 10:34 ..
srw-rw-rw-. 1 system system   0 Apr 28  2022 property-vault.
socket
s?????????? ? ?       ?       ?            ? ssgtzd
drwx------. 3 root   root   60 Apr 28 2022 systemd-private-
e9b8fea895944dd5a0fd05770af20d6e-ModemManager.service-52N7rb
drwx------. 3 root   root   60 Apr 28 2022 systemd-private-
e9b8fea895944dd5a0fd05770af20d6e-bluetooth.service-1nYL2F
drwx------. 3 root   root   60 Apr 28 2022 systemd-private-
e9b8fea895944dd5a0fd05770af20d6e-chronyd.service-UmA8vV
drwx------. 3 root   root   60 Jan  6 1980 systemd-private-
e9b8fea895944dd5a0fd05770af20d6e-ninfod.service-nSxLqH
drwx------. 3 root   root   60 Jan  6 1980 systemd-private-
e9b8fea895944dd5a0fd05770af20d6e-rdisc.service-voag6w
drwx------. 3 root   root   60 Apr 28 2022 systemd-private-
e9b81ea895944dd5a0fd05770af20d6e-systemd-logind.service-kHhqPi
drwx------. 3 root   root   60 Jan  6 1980 systemd-private-
e9b8fea895944dd5a0fd05770af20d6e-systemd-resolved.service-hQ2tR3
drwx------. 3 root   root   60 Jan  1 1970 systemd-private-
e9b8fea895944dd5a0fd05770af20d6e-systemd-timesyncd.service-
Ghf2u0
-rw-r--r--. 1 root   root 8687616 Sep 16 06:21 **trace.dat**
```

```
-rw-r-----. 1 root   root 114229 Sep 16 06:04 wpa_supplicant-
log.txt
```

4. Copy the `trace.dat` log from the DUT to the Ubuntu host device.
5. Get the host and firmware logs as follows:
   - To get the host log from the `trace.dat` log, run the following command:

     ```
     trace-cmd report trace.dat > host.txt
     ```

     Sample output

     ```
     username@hu-username-hyd:/local/mnt/workspace/ath11k$ ls -al
     total 18464
     drwxrwxr-x 2 username users    4096 Aug 15 08:48 .
     drwxrwxrwt 12 root root      4096 Aug 15 08:48 ..
     -rw-r--r-- 1 username users 18898944 Aug 15 08:48 trace.dat
     username@hu-username-hyd:/local/mnt/workspace/ath11k$ trace-
     cmd report trace.dat > host.txt
     username@hu-username-hyd:/local/mnt/workspace/ath11k$ ls -al
     total 19500
     drwxrwxr-x 2 username users    4096 Aug 15 08:48 .
     drwxrwxrwt 12 root  root     4096 Aug 15 08:48 ..
     -rw-rw-r-- 1  username users 1059505 Aug 15 08:48 host.txt
     -rw-r--r-- 1  username users 18898944 Aug 15 08:48 trace.dat
     username@hu-username-hyd:/local/mnt/workspace/ath11k$
     ```

     You can install the `trace-cmd` tool by running the following command in the Ubuntu terminal:

     ```
     apt-get install trace-cmd
     ```

   - To get the firmware logs, share the `trace.dat` log and issue details on the support portal.

7.4 Capture driver logs in dmesg

The ath11k commands for dmesg are as follows:

```
enum ath11k_debug_mask {
          ATH11K_DBG_AHB              = 0x00000001,
          ATH11K_DBG_WMI              = 0x00000002,
          ATH11K_DBG_HTC              = 0x00000004,
          ATH11K_DBG_DP_HTT           = 0x00000008,
```

```
          ATH11K_DBG_MAC           = 0x00000010,
          ATH11K_DBG_BOOT          = 0x00000020,
          ATH11K_DBG_QMI           = 0x00000040,
          ATH11K_DBG_DATA          = 0x00000080,
          ATH11K_DBG_MGMT          = 0x00000100,
          ATH11K_DBG_REG           = 0x00000200,
          ATH11K_DBG_TESTMODE      = 0x00000400,
          ATH11K_DBG_HAL           = 0x00000800,
          ATH11K_DBG_PCI           = 0x00001000,
          ATH11K_DBG_DP_TX         = 0x00002000,
          ATH11K_DBG_DP_RX         = 0x00004000,
          ATH11K_DBG_CE            = 0x00008000,
  };
```

You can use each command to get a specific log. For example,

- To get WMI-related driver logs, run the following command:

  ```
  echo 0x2 > /sys/module/ath11k/parameters/debug_mask
  ```

- To get host target communication (HTC) related logs, run the following command:

  ```
  echo 0x4 > /sys/module/ath11k/parameters/debug_mask
  ```

7.5 Wireless sniffer

A wireless sniffer intercepts the Wi-Fi frames over the air. Run the Wireshark software on any Linux or Windows device that can sniff the Wi-Fi frames.

7.6 Collect firmware dump

When the device crashes, the firmware dump is saved as a BIN file at the `/var/spool/crash` directory.

```
sh-5.1# cd /var/spool/crash/
sh-5.1# ls
0000:01:00.0_2022-04-28_17-51-45.bin
```

Share this dump file on the support portal.

Logs for CLD driver

The Wi-Fi subsystem offers tools and logs to assist with troubleshooting issues.

7.7 Pull Wi-Fi host driver and firmware logs

Logs from the Wi-Fi host driver and firmware are in the `/var/persist/wlan_logs` directory.

The `/usr/sbin/cnss_diag` module facilitates this logging.

To pull these logs, perform the following steps:

1. For continuous logging before any Wi-Fi operation, run the `cnss_diag` in the background and then run the following command:

   ```
   cnss_diag -s -f -q &
   ```

   Sample output

   ```
   root@qcm6490:~# cnss_diag -s -f -q &
   [1] 4285
   root@qcm6490:~# diag: Diag_LSM_Init: invoked for pid: 4285 with
   init_count: 0
   diag:successfully connected to socket 3
   diag: Diag_LSM_Init: done for pid: 4285 with init_count: 1
   Diag LSM init done!diag: Successfully registered commands with
   the driver
   initialize_cnsslog_timer: create cnss log timer
   cld80211 ctx init done, proceed to add mcast groupscnss_open_
   log_file: create directory /var/persist/wlan_logs/ ret = 0
   errno= 2failed to open file a+ mode or file size 0 is less than
   max_file_size 31457280
   cnss_open_log_file: directory /var/persist/wlan_logs/
   createdfailed to open file a+ mode or file size 0 is less than
   max_file_size 31457280
   radio_id:0 or gdiag_header is invalidradio_id:1 or gdiag_header
   is invalidcnss_open_log_file: directory /var/persist/wlan_logs/
   createdloaded msc path:/lib/firmware/qcom/qcm6490/Data.msc
   chipset:0
   loaded msc path:/lib/firmware/qcom/qcm6490/Data.msc chipset:6750
   process_cnss_diag_msg: start cnss log timer
   ```

2. Pull the Wi-Fi logs at `/var/persist/wlan_logs` by running the following command in a separate command prompt:

```
scp -r root@192.168.1.22:/var/persist/wlan_logs/ .
```

Sample output

```
C:\Users\username\Downloads\K2L>scp -r root@10.92.180.250:/var/
persist/wlan_logs/.
cnss_fw_logs_current.txt
                            100%  22MB  6.3MB/s  00:03
cnss_fw_logs_000.txt
                            100%  30MB  5.8MB/s  00:05
txrx_pktlog_current.dat
                            100%   0    0.0KB/s  00:00
host_driver_logs_current.txt
                            100%  540KB 6.4MB/s  00:00
cnss_fw_logs_001.txt
                            100%  30MB  5.8MB/s  00:05
```

8 References

Document title	DCN
Qualcomm Linux Build Guide	80-70018-254

Acronyms/terms	Definition
AP	Access Point
API	Application Programming Interface
ATF	Air Time Fairness
BDF	Board Data File
BSSID	Basic Service Set Identifier
DBS	Dual Band Simultaneous
DFS	Dynamic Frequency Selection
DHCP	Dynamic Host Configuration Protocol
DL	Downlink
DUT	Device Under Test
EAP	Extensible Authentication Protocol
FTM	Factory Test Mode
HTC	Host Target Communication
HTT	Host Target Interface
ISM	Industrial, Scientific, and Medical
LOWI	Location Wi-Fi
LTE	Long Term Evolution
MAC	Media Access Control
MU	Multi-User
MU-MIMO	Multi-User Multiple-Input and Multiple-Output
MWS	Mobile Wireless Standards
OFDMA	Orthogonal Frequency Division Multiple Access
P2P	Peer-to-Peer
PCAT	Product Configuration Assistant Tool
PCIe	Peripheral control interface express
PID	Process Identification
QRCT	Qualcomm Radio Control Toolkit
RADIUS	Remote Authentication Dial-in User Service

Acronyms/terms	Definition
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
RU	Resource Unit
SoC	System on Chip
SRNG	Standard Ring Interface
SS	Subsystem
STA	Station mode
SSH	Secure Socket Shell
SSID	Service Set Identifier
TID	Traffic Identifier
TWT	Target Wake Time
UART	Universal Asynchronous Receiver/Transmitter
UL	Uplink
UL-OFDMA	Uplink Orthogonal Frequency Division Multiple Access
UORA	Uplink Orthogonal Frequency Division Multiple Access Random Access
WFA	Wi-Fi Alliance
WMI	Wireless Module Interface

LEGAL INFORMATION

Your access to and use of this material, along with any documents, software, specifications, reference board files, drawings, diagnostics and other information contained herein (collectively this "Material"), is subject to your (including the corporation or other legal entity you represent, collectively "You" or "Your") acceptance of the terms and conditions ("Terms of Use") set forth below. If You do not agree to these Terms of Use, you may not use this Material and shall immediately destroy any copy thereof.

1) Legal Notice.

This Material is being made available to You solely for Your internal use with those products and service offerings of Qualcomm Technologies, Inc. ("**Qualcomm Technologies**"), its affiliates and/or licensors described in this Material, and shall not be used for any other purposes. If this Material is marked as "**Qualcomm Internal Use Only**", no license is granted to You herein, and You must immediately (a) destroy or return this Material to Qualcomm Technologies, and (b) report Your receipt of this Material to qualcomm.support@qti.qualcomm.com. This Material may not be altered, edited, or modified in any way without Qualcomm Technologies' prior written approval, nor may it be used for any machine learning or artificial intelligence development purpose which results, whether directly or indirectly, in the creation or development of an automated device, program, tool, algorithm, process, methodology, product and/or other output. Unauthorized use or disclosure of this Material or the information contained herein is strictly prohibited, and You agree to indemnify Qualcomm Technologies, its affiliates and licensors for any damages or losses suffered by Qualcomm Technologies, its affiliates and/or licensors for any such unauthorized uses or disclosures of this Material, in whole or part.

Qualcomm Technologies, its affiliates and/or licensors retain all rights and ownership in and to this Material. No license to any trademark, patent, copyright, mask work protection right or any other intellectual property right is either granted or implied by this Material or any information disclosed herein, including, but not limited to, any license to make, use, import or sell any product, service or technology offering embodying any of the information in this Material.

THIS MATERIAL IS BEING PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESSED, IMPLIED, STATUTORY OR OTHERWISE. TO THE MAXIMUM EXTENT PERMITTED BY LAW, QUALCOMM TECHNOLOGIES, ITS AFFILIATES AND/OR LICENSORS SPECIFICALLY DISCLAIM ALL WARRANTIES OF TITLE, MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, COMPLETENESS OR ACCURACY, AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MOREOVER, NEITHER QUALCOMM TECHNOLOGIES, NOR ANY OF ITS AFFILIATES AND/OR LICENSORS, SHALL BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY EXPENSES, LOSSES, USE, OR ACTIONS HOWSOEVER INCURRED OR UNDERTAKEN BY YOU IN RELIANCE ON THIS MATERIAL.

Certain product kits, tools and other items referenced in this Material may require You to accept additional terms and conditions before accessing or using those items.

Technical data specified in this Material may be subject to U.S. and other applicable export control laws. Transmission contrary to U.S. and any other applicable law is strictly prohibited.

Nothing in this Material is an offer to sell any of the components or devices referenced herein.

This Material is subject to change without further notification.

In the event of a conflict between these Terms of Use and the *Website Terms of Use* on www.qualcomm.com, the *Qualcomm Privacy Policy* referenced on www.qualcomm.com, or other legal statements or notices found on prior pages of the Material, these Terms of Use will control. In the event of a conflict between these Terms of Use and any other agreement (written or click-through, including, without limitation any non-disclosure agreement) executed by You and Qualcomm Technologies or a Qualcomm Technologies affiliate and/or licensor with respect to Your access to and use of this Material, the other agreement will control.

These Terms of Use shall be governed by and construed and enforced in accordance with the laws of the State of California, excluding the U.N. Convention on International Sale of Goods, without regard to conflict of laws principles. Any dispute, claim or controversy arising out of or relating to these Terms of Use, or the breach or validity hereof, shall be adjudicated only by a court of competent jurisdiction in the county of San Diego, State of California, and You hereby consent to the personal jurisdiction of such courts for that purpose.

2) Trademark and Product Attribution Statements.

Qualcomm is a trademark or registered trademark of Qualcomm Incorporated. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the U.S. and/or elsewhere. The Bluetooth® word mark is a registered trademark owned by Bluetooth SIG, Inc. Other product and brand names referenced in this Material may be trademarks or registered trademarks of their respective owners.

Snapdragon and Qualcomm branded products referenced in this Material are products of Qualcomm Technologies, Inc. and/or its subsidiaries. Qualcomm patented technologies are licensed by Qualcomm Incorporated.