Qualcomm Technologies, Inc.

QCA6574AU.LE.2.2.1 Post CS6

Release Notes

80-YC286-14 Rev. G

June 15, 2023

For additional information or to submit technical questions, go to: https://createpoint.qti.qualcomm.com

Confidential – Qualcomm Technologies, Inc. and/or its affiliated companies – May Contain Trade Secrets

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	November 2022	Initial release
B	December 2022	Updated section 10.4
C	January 2023	▪ Added host 5.04 in section 1.5, 1.6.3 and updated section 1.7.3 accordingly. ▪ Updated section 6.4 with common tools location and added a note about creating a folder.
D	April 2023	▪ Updated section 8.7
E	May 2023	▪ Updated repo location in section 10.3 and 11.3
F	June 2023	▪ Updated document for CodeLinaro ▪ Updated build id in section 1.6.1, 1.6.2, and 1.6.3 ▪ Added supported features in post CS6 release in section 3.1 ▪ Updated resolved bugs in section 4.4
G	June 2023	▪ Updated the download package location in section 9.3, 10.3, and 11.3.

Contents

Figures

Tables

1 Getting started

1.1 Purpose

This document provides information for QCA6574AU.LE.2.2.1 POST CS6 Release.

The supported ASICs are show in Table 2-1.

1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Commands to be entered appear in a different font, for example, **`copy a:*.* b:`**

Button and key names appear in bold font, for example, click **Save** or press **Enter**.

1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm® Technologies, Inc. (QTI) at https://createpoint.qti.Qualcomm.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

1.4 Release packages

Table 1-1 describes the software for this product line, which is divided into the release packages. Download the release packages separately or combined for a complete product line software set.

Table 1-1 Release packages

From chipcode.qti.Qualcomm.com	From codelinaro.org
▪ Proprietary non-HLOS software ▫ Contains proprietary source and firmware images for all non-apps processors ▫ An umbrella package built from a combined set of integrated individual component releases	▪ Open source HLOS software ▫ Contains open source for apps processor HLOS
▪ Proprietary HLOS software ▫ Contains proprietary source and firmware images for the apps processor HLOS	–

The proprietary and open source HLOS packages are from separate sources and combined according to the downloading instructions. The unique build identification (build ID) code identifies each package with the following naming convention.

<PL Image>-<Version>-<Chipset>

- <PL_Image>–CHSS.LNX_FSL.2.1
- <Version> – Variable number of digits used to represent the build ID version
- <Build Flavor> – Meta build flavor

For example, CHSS.LNX_FSL.2.1-05210-QCA6574AUARMSDIOHZ-1

1.5 Download software from Qualcomm ChipCode™ portal

The non-opensource software can be downloaded from the Qualcomm ChipCode website. Designated point of contacts in your organization can download the licensed software. The software is organized into distribution packages (distros) composed of subsystem image files. Each distro has a corresponding Git project. The Git tree includes revisions for previous builds that allows you to differentiate the changes between releases.

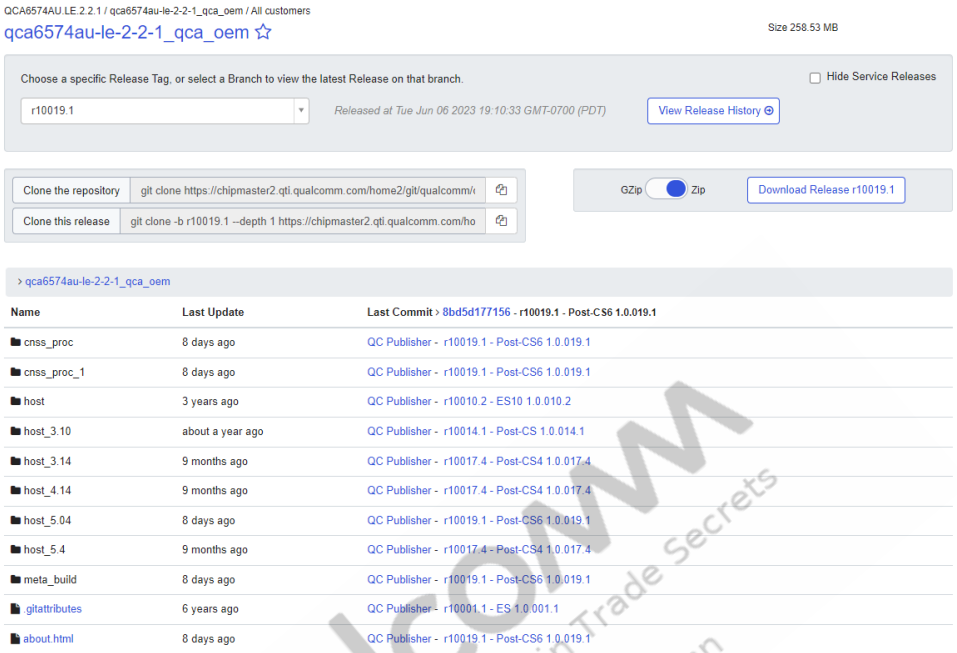
1. If you are new to the Qualcomm ChipCode, review the following link for up-to-date documentation and a set of tutorial videos:

   https://chipcode.qti.qualcomm.com/projects/help/wiki

2. Create another top-level directory on the build PC. Download the proprietary software from https://chipcode.qti.qualcomm.com.

   Select QCA6574AU.LE.2.2.1 from the products list, Click **-co-ltd/ qca6574au-le-2-2-1_qca_oem to download the code.

Download code.



3. Unzip each of the subsystem images to generate the following directory structure. In this example, <target_root> is the top-level directory.

```
<target_root>
├── cnss_proc
└── cnss_proc_1
└── host
        └── apps_proc
└── host_4.14
        └── apps_proc
└── host_5.4
        └── apps_proc
└── host_5.04
        └── apps_proc
└── meta_build
```

1.6 Download software from open source HLOS software

The Linux Board Support Package (BSP) release is in two parts, a proprietary release from ChipCode and an open-source release from the Code Linaro(CLO) site.

1.6.1 Download software from open source HLOS software for Linux Kernel 4.9

1. $ cd <target_root>/host/apps_proc
2. Locate the <build_id> tag, which identifies the corresponding open source HLOS software build. Download from CLO.

   Download repo launcher from
   https://git.codelinaro.org/clo/tools/repo/-/blob/aosp-new/repo-1/repo

Then use this repo to download source code as below

```
$repo init --no-clone-bundle -u
https://git.codelinaro.org/clo/le/le/manifest.git -b release -m
CHSS.LNX_FSL.2.1-07310-QCA6574AUARMSDIOHZ.xml --repo-
url=https://git.codelinaro.org/clo/tools/repo.git --repo-branch=aosp-
new/repo-1
```

3. Sync the source codes.

```
$repo sync -c --no-tags -j16
```

After that, <target_root>folders show as follows.

```
├── cnss_proc
└── cnss_proc_1
└── host
        └── apps_proc
              ├── .repo
              └── sources
└── host_4.14
└── host_5.4
└── meta_build
```

1.6.2 Download software from open source HLOS software for Linux Kernel 4.14

1. $ cd <target_root>/host_4.14/apps_proc
2. Locate the <build_id> tag, which identifies the corresponding open source HLOS software build. Download from CLO.

   Download repo launcher from
   https://git.codelinaro.org/clo/tools/repo/-/blob/aosp-new/repo-1/repo

   Then use this repo to download source code as below

```
$repo init --no-clone-bundle -u
https://git.codelinaro.org/clo/le/le/manifest.git -b release -m
CHSS.LNX_FSLS.4.14-05410-QCA6574AUARMSDIOHZ.xml --repo-
url=https://git.codelinaro.org/clo/tools/repo.git --repo-branch=aosp-
new/repo-1
```

3. Sync the source codes.

```
$repo sync -c --no-tags -j16
```

After that, <target_root>folders show as follows.

```
├── cnss_proc
└── cnss_proc_1
└── host
└── host_4.14
        └── apps_proc
              ├── .repo
              └── sources
└── host_5.4
└── meta_build
```

1.6.3 Download software from open source HLOS software for Linux Kernel 5.4

1. $ cd <target_root>/host_5.04/apps_proc
2. Locate the <build_id> tag, which identifies the corresponding open source HLOS software build. Download from CLO.

   Download repo launcher from
   https://git.codelinaro.org/clo/tools/repo/-/blob/aosp-new/repo-1/repo

   Then use this repo to download source code as below

   ```
   $repo init --no-clone-bundle -u
   https://git.codelinaro.org/clo/le/le/manifest.git -b release -m
   CHSS.LNX_FSL.5.0.r1-02900-QCA6574AUARMSDIOHZ.xml --repo-
   url=https://git.codelinaro.org/clo/tools/repo.git --repo-branch=aosp-
   new/repo-1
   ```

3. Sync the source codes.

   ```
   $repo sync -c --no-tags -j16
   ```

   After that, <target_root>folders show as follows.

   ```
   ├── cnss_proc
   └── cnss_proc_1
   └── host
   └── host_4.14
   └── host_5.4
   └── host_5.04
          └── apps_proc
                ├── .repo
                └── sources
   └── meta_build
   ```

1.7 Build image

1.7.1 Build image for Linux Kernel 4.9 at I.mx6 platform

Use the following commands to build Image.

1. Open the command prompt and change to the following directory:

   Linux kernel 4.9

   ```
   $cd <target_root>/host/apps_proc
   ```

2. $source sources/meta-qti-connectivity/scripts/set_bb_env.sh
3. $build-imxauto-image

After that, image is located at
<target_root>/host/apps_proc/build/tmp/deploy/images/imx6qsabresd.(Linux Kernel 4.9)

NOTE: The script set_bb_env.sh uses the Freescale imx6qasbresd (shown in Figure1-1) as the default target board. If customer uses other Freescale target board, such as imx6qsabreauto, then step2 becomes to:

$ MACHINE= imx6qsabreauto source sources/meta-qti-connectivity/scripts/set_bb_env.sh

1.7.2 Build image for Linux Kernel 4.14 at I.mx8 platform

Use the following commands to build Image.

4. Open the command prompt and change to the following directory:

   ```
   $cd <target_root>/host_4.14/apps_proc
   ```

5. $EULA=1 source sources/meta-qti-connectivity/scripts/set_imx8_env.sh
6. $build-imxauto-image

After that, image is located at <target_root>/ host_4.14/apps_proc/build/tmp/deploy/images/ imx8mqevk. (Linux Kernel 4.14)

NOTE: The script set_imx8_env.sh uses the Freescale imx8mqevk (shown in Figure1-3) as the default target board. If customer uses other Freescale target board, such as imx8qxpmek, then step2 becomes to:

$ MACHINE= imx8qxpmek EULA=1 source sources/meta-qti-connectivity/scripts/set_imx8_env.sh

1.7.3 Build image for Linux Kernel 5.4 at I.mx8 platform

Use the following commands to build Image.

1. Open the command prompt and change to the following directory:

   ```
   $cd <target_root>/host_5.04/apps_proc
   ```

2. $EULA=1 source sources/meta-qti-connectivity/scripts/set_imx8_env.sh
3. $build-imxauto-image

After that, image is located at <target_root>/host_5.04/apps_proc/build/tmp/deploy/images/imx8qxpmek. (Linux Kernel 5.4)

NOTE: The script set_imx8_env.sh uses the Freescale imx8qxpmek (shown in Figure1-3) as the default target board. If customer uses other Freescale target board, such as imx8mqevk, then step2 becomes to:

$ MACHINE= imx8mqevk EULA=1 source sources/meta-qti-connectivity/scripts/set_imx8_env.sh

1.8 Flash image

1.8.1 Flash image for Linux Kernel 4.9/4.14 at I.mx6 platform

Creating a Boot SD card

Step 1: Insert a SD card into laptop card reader slot

Step 2: Covert and copy core image file into SD card

Switch the folder path to *./build/tmp/deploy/images* and find the image with name {MACHINE NAME}, for example MACHINE=imx6qsabresd. Then follow the below instructions to copy it onto SD card for booting.

```
$ sudo dd if=standalone-auto-image-imx6qsabresd.sdcard
of=/dev/{SD_DEV_NAME}
```

NOTE: {SD_DEV_NAME} is based on the name which system recognizes. You can use "mount" command to check it. In general, it is "/dev/mmcblk0" or "/dev/sdb"

After the file transfer is completed, sync to write back into SD card before unmount it.

```
$ sudo sync
```

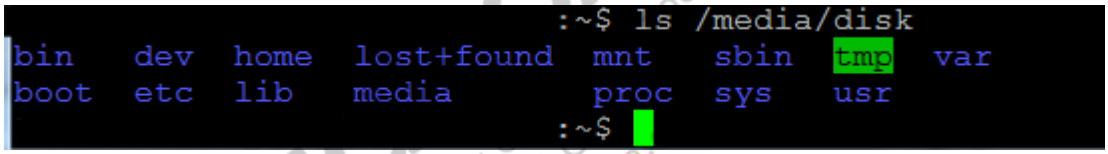
Step 3: Check the image on the SD card

- Use "mount" command to identify rootfs partition of Yocto bsp core image on SD card

```
$ mount
…
/dev/sdb2 on /media/disk type ext3 (rw,nosuid,nodev,uhelper=udisks)
…
```

It indicates that the Yocto rootfs on SD card is mounted on development machine in "/media/disk". Use *ls* command to check its root file system.

```
$ ls /media/disk
```



Step 4: Copy QCA65X4A/AU-1 firmware files to image on the SD card. The firmware files are located at <target_root>/meta_build/firmware/pcie.

```
$cd <target_root>/meta_build/firmware/pcie
$sudo cp athwlan.bin otp.bin fakeboar.bin utf.bin
/media/disk/lib/firmware/
```

Step 5: Copy QCA65X4A/AU-1 firmware log parse data to image on SD card, which is located at <target_root>/meta_build/firmware/pcie.

```
$cd <target_root>/meta_build/firmware/pcie
$sudo mkdir -p /media/disk/firmware/image/
$sudo cp Data.msc /media/disk/firmware/image/
```

Step 6: Copy and rename the host driver configuration files

```
# sudo mkdir -p /media/disk/lib/firmware/wlan
# sudo cd /media/disk/lib/firmware/wlan/
# sudo cp QCA6574AU.LE.2.2.1_Rome_PCIe_qcacld-3.0.ini qcom_cfg.ini
```

Step 7: Copy QCA65X4A/AU-3 firmware files to image on the SD card. The firmware files are located at <target_root>/meta_build/firmware/sdio.

```
$cd <target_root>/meta_build/firmware/sdio
$sudo mkdir -p /media/disk/lib/firmware/qca6574
$sudo cp qwlan30.bin otp30.bin bdwlan30.bin utf30.bin
/media/disk/lib/firmware/qca6574
```

Step 8: Copy QCA65X4A/AU-3 firmware log parse data to image on SD card, which is located at <target_root>/meta_build/firmware/sdio.

```
$cd <target_root>/meta_build/firmware/sdio
$sudo mkdir -p /media/disk/lib/firmware/qca9377
$sudo cp Data.msc /media/disk/lib/firmware/qca9377
```

Step 9: Copy and rename the host driver configuration files.

```
# sudo mkdir -p /media/disk/lib/firmware/wlan/qca6574
# sudo cd /media/disk/lib/firmware/wlan/qca6574
# sudo cp QCA6574AU.LE.2.2.1_Rome_SDIO_qcacld-3.0.ini qcom_cfg.ini
```

Note: `Data.msc` default path is `/lib/firmware/qca9377`, it can be specified by command cnss_diag option `-p`.

- wlan.ko & wlan-sdio.ko
  - Located on /lib/modules/${Kernel Version}/extra/
- wpa_supplicant and hostapd
  - Located on /usr/sbin
- QCA65X4A/AU-1 target firmware files
  - WLAN firmware, located on rootfs “/lib/firmware” directory
    - **athwlan.bin** : WLAN target firmware
    - **otp.bin** : OTP memory manipulate firmware
    - **fakeboar.bin** : the board data
    - **utf.bin** : WLAN testmod firmware
  - WLAN FW log parse data, located on rootfs “/firmware/image/” directory
    - **Data.msc:** used for parse FW log
- QCA65X4A/AU-1 host driver configuration files
  - located on rootfs “/lib/firmware/wlan” directory
    - qcom_cfg.ini
- QCA65X4A/AU-3 target firmware files
  - WLAN firmware, located on rootfs “/lib/firmware/qca6574” directory
    - **qwlan30.bin** : WLAN target firmware
    - **otp30.bin** : OTP memory manipulate firmware
    - **bdwlan30.bin** : the board data
    - **utf30.bin** : WLAN testmod firmware
  - WLAN FW log parse data, located on rootfs “/lib/firmware/qca9377” directory
    - **Data.msc:** used for parse FW log
- QCA65X4A/AU-3 host driver configuration files
  - located on rootfs “/lib/firmware/wlan/qca6574” directory
    - qcom_cfg.ini

Step 6: Remove SD Card from developing environment

1.8.2 Flash image for Linux Kernel 5.4 at I.mx8 platform

Creating a Boot SD card

Step 1: Insert a SD card into laptop card reader slot

Step 2: Covert and copy core image file into SD card

Switch the folder path to *./build/tmp/deploy/images* and find the image with name {MACHINE NAME}, for example MACHINE= imx8qxpmek. Then follow the below instructions to copy it onto SD card for booting.

```
$ bunzip2 -dk -f standalone-imx8auto-image-imx8qxpmek.wic.bz2

$ sudo dd if=standalone-imx8auto-image-imx8qxpmek.wic
of=/dev/{SD_DEV_NAME} bs=1M conv=fsync
```

NOTE: {SD_DEV_NAME} is based on the name which system recognizes. You can use "mount" command to check it. In general, it is "`/dev/mmcblk0`" or "`/dev/sdb`"

After the file transfer is completed, sync to write back into SD card before unmount it.

```
$ sudo sync
```

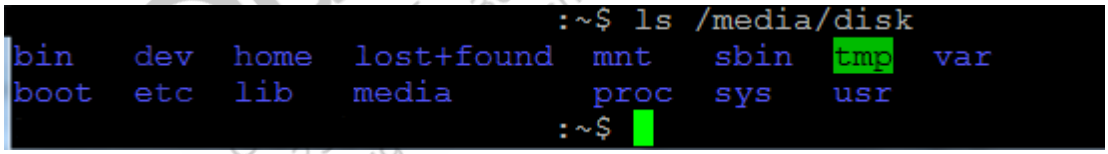
Step 3: Check the image on the SD card

- Use "mount" command to identify rootfs partition of Yocto bsp core image on SD card

```
$ mount
…
/dev/sdb2 on /media/disk type ext3 (rw,nosuid,nodev,uhelper=udisks)
…
```

It indicates that the Yocto rootfs on SD card is mounted on development machine in "/media/disk". Use *ls* command to check its root file system.

```
$ ls /media/disk
```



Step 4: Copy QCA6574 firmware files to image on the SD card. The firmware files are located at <target_root>/meta_build/firmware/pcie.

```
$cd <target_root>/meta_build/firmware/pcie
$sudo cp athwlan.bin otp.bin fakeboar.bin utf.bin Data.msc
/media/disk/lib/firmware/
```

Copy and rename the host driver configuration files

```
# sudo mkdir -p /media/disk/lib/firmware/wlan
# sudo cd /media/disk/lib/firmware/wlan/
# sudo cp QCA6574AU.LE.2.2.1_Rome_PCIe_qcacld-3.0.ini qcom_cfg.ini
```

- wlan.ko
  - Located on /lib/modules/${Kernel Version}/extra/
- wpa_supplicant and hostapd
  - Located on /usr/sbin
- QCA65X4A/AU-1 target firmware files
  - WLAN firmware, located on rootfs "/lib/firmware" directory
    - **athwlan.bin** : WLAN target firmware
    - **otp.bin** : OTP memory manipulate firmware

- **fakeboar.bin** : the board data
- **utf.bin** : WLAN testmod firmware

□ WLAN FW log parse data, located on rootfs “/firmware/image/” directory

- **Data.msc:** used for parse FW log

■ QCA65X4A/AU-1 host driver configuration files

□ located on rootfs “/lib/firmware/wlan” directory

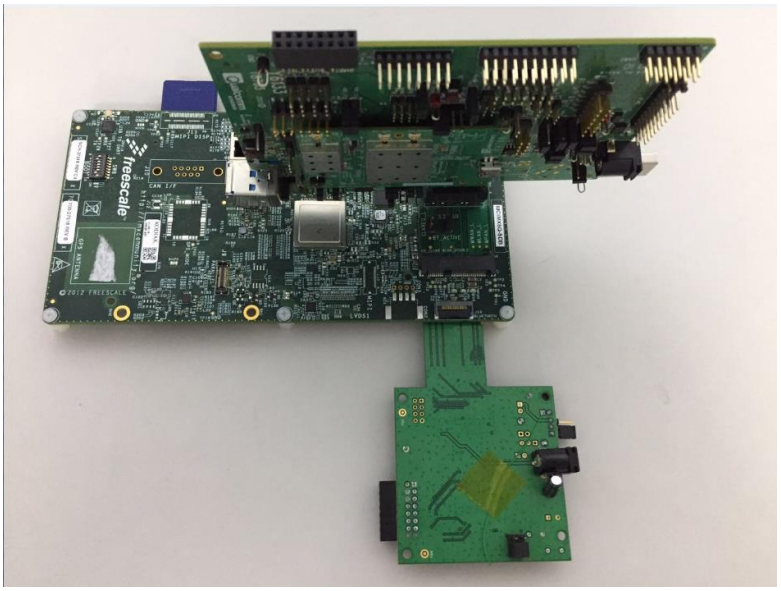
- qcom_cfg.ini

Step 6: Remove SD Card from developing environment

1.9 Hardware Setup

1.9.1 Hardware Setup for Linux Kernel 4.9/4.14 at I.mx6 platform

The Figure 1-1 shows the hardware Connection on Freescale SABRE.



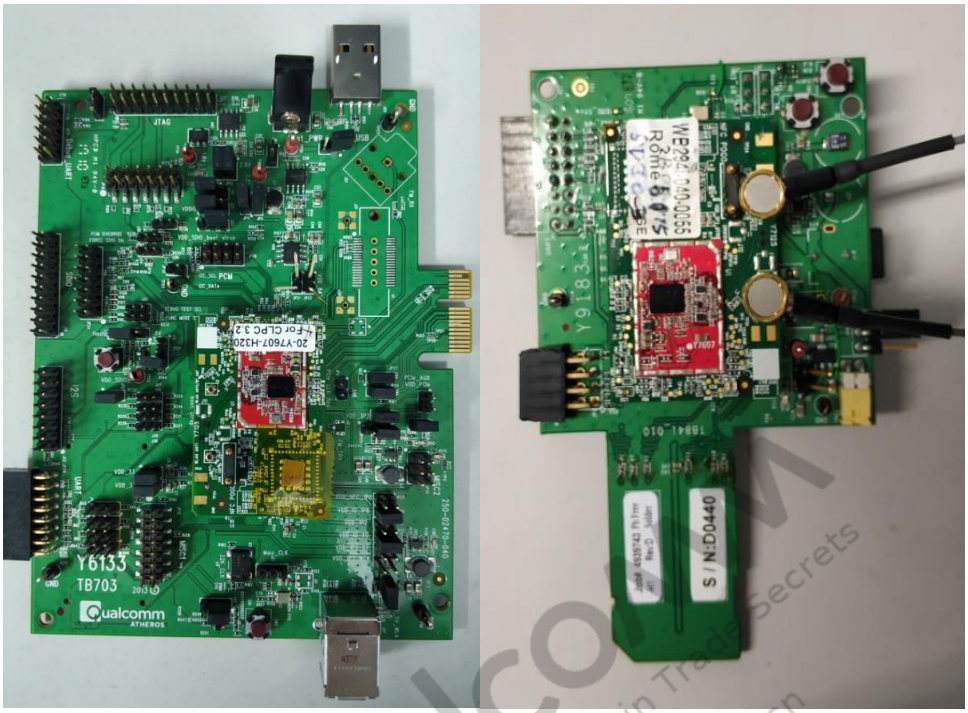
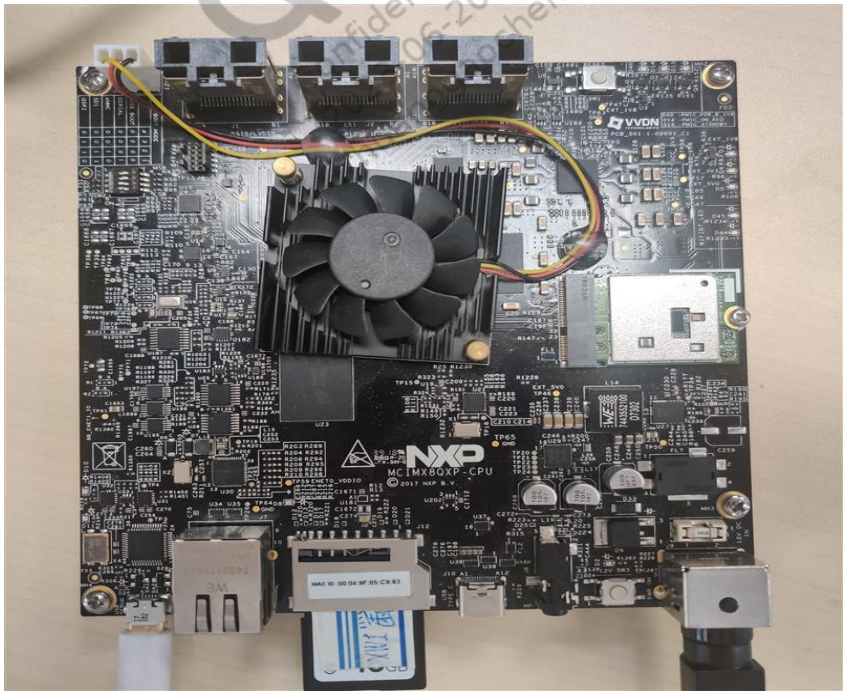


Figure 1-1 Hardware connection of Freescale

1.9.2 Hardware Setup for Linux Kernel 5.4 at I.mx8 platform

The Figure 1-1 shows the hardware Connection on imx8qxpmek.



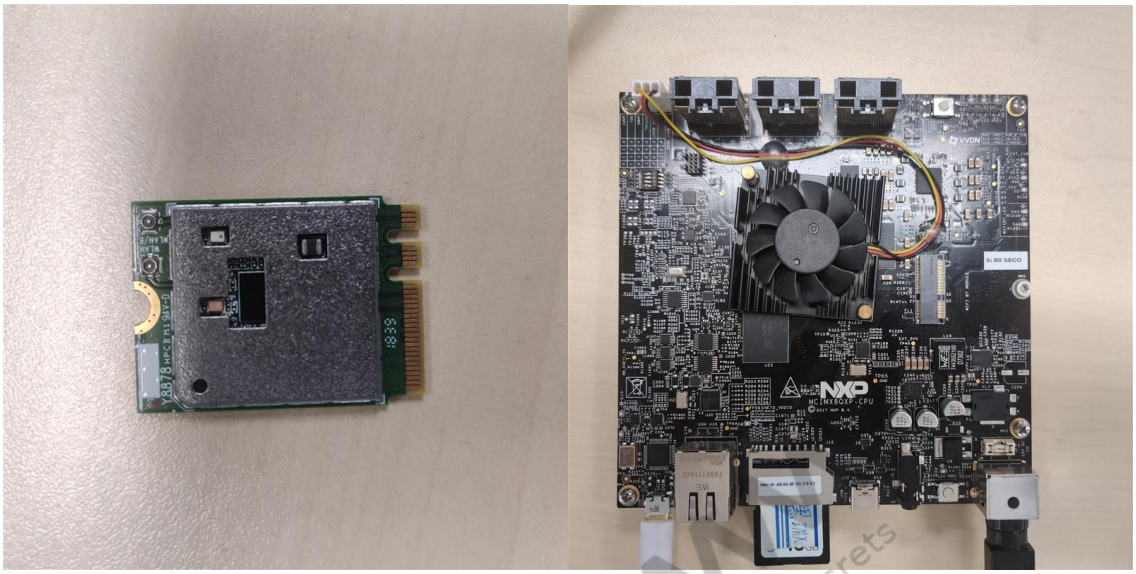


Figure 1-3 Hardware connection of imx8qxpmek

1.10 Boot up Yocto-Linux

1.10.1 Boot up Yocto-Linux for Linux Kernel 4.9/4.14 at I.mx6 platform

- Use SD boot disk to boot up Freescale iMX6 platform
- Click the following command after ‘u-boot=>’ prompt, then uboot process will go on.

```
#u-boot=> setenv mmcargs setenv bootargs console=${console},${baudrate}
${smp} root=${mmcroot} pci=nomsi

#u-boot=> saveenv

#u-boot=> run bootcmd
```

-
- Use “root” id to login Yocto-Linux

1.10.2 Boot up Yocto-Linux for Linux Kernel 4.14/5.4 at I.mx8 platform

- Use SD boot disk to boot up Freescale iMX8 platform
- Click the following command after ‘u-boot=>’ prompt, then uboot process will go on.

```
#u-boot=> setenv  mmcargs  setenv bootargs ${jh_clk}
console=${console} root=${mmcroot} pci=nomsi

#u-boot=> saveenv

#u-boot=> run bootcmd
```

- Use “root” id to login Yocto-Linux

1.11 Bring up WLAN driver

Insert kernel module “wlan.ko”

- PCIE

```
# cd /lib/modules/${Kernel Version}/extra/
# insmod wlan.ko
```

- SDIO

```
# cd /lib/modules/${Kernel Version}/extra/
# insmod wlan-sdio.ko
```

NOTE: Install wlan.ko before installing wlan-sdio.ko, since wlan tools(cnss_diag) parse FW logs in this order for dual Wi-Fi mode.

Use *ifconfig*, *iwconfig* commands to bring up WLAN interface on Linux

```
# iwconfig
```

There are two interfaces created in *wlan0* and *p2p0* as default.

```
root@imx6qsabresd:~# iwconfig
wlan1     Qcom:802.11n  ESSID:off/any  Nickname:""
          Mode:Managed  Channel:0  Access Point: Not-Associated
          Bit Rate:0 kb/s   Tx-Power=0 dBm
          RTS thr=1048576 B   Fragment thr=8000 B
          Encryption key:off

eth0      no wireless extensions.

p2p1      Qcom:802.11n  ESSID:off/any  Nickname:""
          Mode:Managed  Channel:0  Access Point: Not-Associated
          Bit Rate:0 kb/s   Tx-Power=0 dBm
          RTS thr=1048576 B   Fragment thr=8000 B
          Encryption key:off

lo        no wireless extensions.

p2p0      Qcom:802.11n  ESSID:off/any  Nickname:""
          Mode:Managed  Channel:0  Access Point: Not-Associated
          Bit Rate:0 kb/s   Tx-Power=0 dBm
          RTS thr=1048576 B   Fragment thr=8000 B
          Encryption key:off

sit0      no wireless extensions.

wlan0     Qcom:802.11n  ESSID:off/any  Nickname:""
          Mode:Managed  Channel:0  Access Point: Not-Associated
          Bit Rate:0 kb/s   Tx-Power=0 dBm
          RTS thr=1048576 B   Fragment thr=8000 B
          Encryption key:off

root@imx6qsabresd:~#
```

Figure 1-2 WLAN Interfaces

1.11.1 STA mode

Connect to Open Mode AP

1. Setup an AP in WPA-PSK mode with “Yocto” SSID and IP as 192.168.1.1 with DHCP Server enable.
2. Edit the configuration file "wpa-sta.conf" content as following from console.

```
ctrl_interface=/var/run/wpa_supplicant
```

```
update_config=1
network={
ssid="Yocto"
key_mgmt=NONE
}
```

3. Enter the below command from console to connect wlanX with AP.
   ```
   # wpa_supplicant –D nl80211 -i wlan0 -c wpa-sta.conf &
   ```
4. Enter the below command from console, after wlan0 is connected to AP, to set the IP address
   ```
   # udhcpc -i wlan0
   ```
5. Ping the AP IP address to check the connectivity
   ```
   # ping 192.168.1.1
   ```

Connect to WPA-PSK Mode AP

1. Setup an AP in WPA-PSK mode with “Yocto” SSID and IP as 192.168.1.1 with DHCP Server enable.
2. Edit the configuration file "wpa-sta.conf" content as following from console.
   ```
   ctrl_interface=/var/run/wpa_supplicant
   update_config=1
   network={
   ssid="Yocto"
   key_mgmt=WPA-PSK
   auth_alg=OPEN
   pairwise=CCMP
   group=CCMP
   psk="12345678"
   }
   ```
3. Enter the below command from console to connect wlanX with AP.
   ```
   # wpa_supplicant -D nl80211 -i wlan0 -c wpa-sta.conf &
   ```
4. Enter the below command from console, after wlan0 is connected to AP, to set the IP address
   ```
   # udhcpc -i wlan0
   ```
5. Ping the AP IP address to check the connectivity
   ```
   # ping 192.168.1.1
   ```

1.11.2 SoftAP mode

Setup 11ng SoftAP with Open Mode

1. Edit the default configuration file *“hostapd.conf”* and save it on */home/root/sbin* with *hostapd* together
   ```
   ctrl_interface=/var/run/hostapd
   interface=wlan0
   ssid=Yocto
   hw_mode=g
   ```

```
channel=1
auth_algs=1
ieee80211n=1
```

2. Run the hostapd with *"hostapd.conf".*
   ```
   # hostapd hostapd.conf &
   ```
3. Setup the IP address of SoftAP.
   ```
   #ifconfig wlan0 192.168.11.1
   ```
4. Setup STA to connect to this SoftAP with open mode and use "ping" to verify the connection.

Setup 11ng SoftAP with WPA1/2 PSK Mode

1. Edit the default configuration file “hostapd.conf" as following:
   ```
   ctrl_interface=/var/run/hostapd
   interface=wlanX
   ssid=Yocto
   hw_mode=g
   channel=1
   auth_algs=3
   ieee80211n=1
   wpa=3
   wpa_passphrase=12345678
   wpa_key_mgmt=WPA-PSK
   wpa_pairwise=CCMP
   rsn_pairwise=CCMP
   ```
2. Start the hostapd with “hostapd.conf".
   ```
   #hostapd hostapd.conf &
   ```
3. Setup the IP address for the SoftAP.
   ```
   #ifconfig wlan0 192.168.11.1
   ```
4. Setup STA to connect to this SoftAP with WPA1/2 PSK mode and use "ping" to verify the connection.

Setup 11ng SoftAP with WPA1/2 PSK Mode and turn on the dhcp server

1. Edit the default configuration file “hostapd.conf" as following:
   ```
   ctrl_interface=/var/run/hostapd
   interface=wlan0
   ssid=Yocto
   hw_mode=g
   channel=1
   auth_algs=3
   ieee80211n=1
   wpa=3
   wpa_passphrase=12345678
   wpa_key_mgmt=WPA-PSK
   wpa_pairwise=CCMP
   rsn_pairwise=CCMP
   ```
2. Start the hostapd with “hostapd.conf".

```
#hostapd hostapd.conf &
```

3. Setup the IP address for the SoftAP.
```
#ifconfig wlan0 192.168.11.1
```
4. Edit the default configuration file “udhcpd-ap.conf " and save it on */home/root/*:
```
start 192.168.11.20
end 192.168.11.254
pidfile /var/run/udhcpd-ap.pid
lease_file /var/run/udhcpd-ap.leases
opt dns 192.168.2.1
option subnet 255.255.255.0
option domain atherosowl.com
option lease 864000
opt router 192.168.11.1
interface wlan0
```
5. Create a udhcpd-ap.leases file under /var/run/ .
```
# touch /var/run/udhcpd-ap.leases
```
6. Start the udhcpd with “udhcpd-ap.conf".
```
# udhcpd udhcpd-ap.conf -fS &
```
7. Setup STA to connect to this SoftAP with WPA1/2 PSK mode and use "ping" to verify the connection.

1.11.3 P2P mode

Setup P2P Mode

1. Edit an “*p2pdev*_dual.conf” and save it on /home/root/.
```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
device_name=chias-rome-P2P
manufacturer=QCA
model_name=McK
device_type=1-0050F204-1
config_methods=display keypad push_button
p2p_listen_reg_class=81
p2p_listen_channel=1
p2p_oper_reg_class=81
p2p_oper_channel=1
p2p_go_intent=10
persistent_reconnect=1
p2p_no_group_iface=1
```
2. Edit an “empty.conf” and save it on /home/root/.
```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1
```
3. Bring up the wpa_supplicant.

```
wpa_supplicant -D nl80211 -i p2p0 -c p2pdev_dual.conf &
```

4. Enable the wpa_cli on p2p0 interface

```
./wpa_cli -i p2p0
```

Nonautonomous GO method.

1. Run find command on the peers:

```
> p2p_find 10
```

2. After the device is found successfully, run the stop command at Go/Client side:

```
> p2p_stop_find
```

3. Run connect command on both Go/Client side:

```
> p2p_connect <Mac Address of Peer> pbc
```

4. The nonautonomous GO method status messages:

```
> status
You can see the module p2p mode is “P2P GO” or “P2P Client”.
```

5. If you need to set IP address and turn on DHCP server.

```
If the module is “P2P Client” mode:
# udhcpc -i p2p0

If the module is “P2P GO” mode:
# ifconfig p2p0 192.168.11.1
# touch /var/run/udhcpd-ap.leases
# udhcpd udhcpd-P2P.conf -fS &

NOTE: You need edit udhcpd-P2P.conf on /home/root:
start 192.168.11.20
end 192.168.11.254
pidfile /var/run/udhcpd-ap.pid
lease_file /var/run/udhcpd-ap.leases
opt dns 192.168.2.1
option subnet 255.255.255.0
option domain atherosowl.com
option lease 864000
opt router 192.168.11.1
interface p2p0
```

6. Use "ping" to verify the connection.

Autonomous GO method.

1. Start an autonomous GO:

```
> p2p_group_add freq=2412
```

2. Run the “CTRL+C” to exit and run the wps_pbc command:

```
# wpa_cli -i p2p0 wps_pbc
```

3. If you need turn on DHCP server.

```
# ifconfig p2p0 192.168.11.1
# touch /var/run/udhcpd-ap.leases
# udhcpd udhcpd-P2P.conf -fS &

NOTE: You need edit udhcpd-P2P.conf on /home/root:
start 192.168.11.20
end 192.168.11.254
pidfile /var/run/udhcpd-ap.pid
lease_file /var/run/udhcpd-ap.leases
opt dns 192.168.2.1
option subnet 255.255.255.0
option domain atherosowl.com
option lease 864000
opt router 192.168.11.1
interface p2p0
```

4. Setup Mobile phone to connect module and use "ping" to verify the connection.

1.11.4 WPS mode

Setup WPS Station Mode.

1. Edit the configuration file " wps_empty.conf " as following.

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

2. Enter the below command from console to connect wlanX with WPS AP.

```
# wpa_supplicant -D nl80211 -i wlan0 -c empty.conf &
# wpa_cli -i wlan0 wps_pbc
```

3. Turn on WPS function on AP side.
4. Enter the below command from console, after wlan0 is connected to WPS AP, to set the IP address.

```
# udhcpc -i wlan0
```

Setup WPS Station Mode connect to WPS AP via Station side pin.

1. Enter the below command from console to connect wlanX with WPS AP via station side pin.

```
# wpa_supplicant -D nl80211 -i wlan0 -c empty.conf &
# wpa_cli -i wlan0 wps_pbc
```

```
# wpa_cli -i wlan0 wps_pin any
```

The pin code will appear on the console, the pin code is an 8-digit string.

2. Enter the pin code on the WPS AP side and register.
3. Enter the below command from console, after wlan0 is connected to WPS AP, to set the IP address.

```
# udhcpc -i wlan0
```

Setup WPS Station Mode connect to WPS AP via AP side pin.

1. Enter the below command from console to connect wlanX with WPS AP via AP side pin.

```
# wpa_supplicant -D nl80211 -i wlan0 -c empty.conf &
# wpa_cli wps_pbc
# wps_reg 58:6d:8f:ac:15:b6 61083721
The wps_reg command need AP side BSSID and PIN code.
```

2. Enter the below command from console, after wlan0 is connected to WPS AP, to set the IP address.

```
# udhcpc -i wlan0
```

Setup WPS AP mode, The WPS Station connection.

1. Edit an "*wps_hostapd.conf*" and save it on /home/root/.

```
driver=nl80211
interface=wlan0
ssid=wps-test
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
wpa_passphrase=12345678
ctrl_interface=/var/run/hostapd
eap_server=1
wps_state=2
ap_setup_locked=1
wps_pin_requests=/var/run/hostapd.pin-req
device_name=Wireless AP
manufacturer=Qualcomm
model_name=WAP
model_number=123
serial_number=12345
device_type=6-0050F204-1
os_version=01020300
config_methods=label display push_button keypad
```

2. Start the hostapd with "hostapd.conf".

```
# hostapd wps_hostapd.conf &
```

3. Setup the IP address for the SoftAP.

```
#ifconfig wlan0 192.168.11.1
```

4. Edit the default configuration file “udhcpd-ap.conf " and save it on */home/root*:

```
start 192.168.11.20
end 192.168.11.254
pidfile /var/run/udhcpd-ap.pid
lease_file /var/run/udhcpd-ap.leases
opt dns 192.168.2.1
option subnet 255.255.255.0
option domain atherosowl.com
option lease 864000
opt router 192.168.11.1
interface wlan0
```

5. Create a udhcpd-ap.leases file under /var/run/ .

```
# touch /var/run/udhcpd-ap.leases
```

6. Start the udhcpd with “udhcpd-ap.conf".

```
# udhcpd udhcpd-ap.conf -fS &
```

7. Trigger WPS on WPS AP side

```
# hostapd_cli -i wlan0 wps_pbc
```

8. WPS station device can connect to WPS function.

Setup WPS AP mode, The WPS Station connection via Station pin code.

1. Start the hostapd with “hostapd.conf".

```
# hostapd wps_hostapd.conf &
```

2. Setup the IP address for the SoftAP.

```
#ifconfig wlanX 192.168.11.1
```

3. Edit the default configuration file “udhcpd-ap.conf " and save it on */home/root*:

```
start 192.168.11.20
end 192.168.11.254
pidfile /var/run/udhcpd-ap.pid
lease_file /var/run/udhcpd-ap.leases
opt dns 192.168.2.1
option subnet 255.255.255.0
option domain atherosowl.com
option lease 864000
opt router 192.168.11.1
interface wlan0
```

4. Create a udhcpd-ap.leases file under /var/run/ .

```
# touch /var/run/udhcpd-ap.leases
```

5. Start the udhcpd with “udhcpd-ap.conf".

```
# udhcpd udhcpd-ap.conf -fS &
```

6. Connect via the pin code on the WPS station device.

```
# hostapd_cli -i wlan0 wps_pin any 81951741
```

1.11.5 SoftAP + SoftAP mode

Setup SoftAP session

1. Configure SoftAP function.

```
# ifconfig wlan0 up
# hostapd hostapd.conf &
```

2. Set IP address and turn on DHCP server.

```
# ifconfig wlan0 192.168.11.1
# touch /var/run/udhcpd-ap.leases
# udhcpd udhcpd-ap.conf -fS &
```

Setup another SoftAP session

3. Create a SoftAP interface.

```
# iw dev wlan0 interface add wlan1 type __ap
```

4. Configure SoftAP function.

```
# ifconfig wlan1 up
# hostapd hostapd1.conf &
```

5. Set IP address and turn on DHCP server

```
# ifconfig wlan1 192.168.12.1
# udhcpd udhcpd-ap1.conf -fS &
```

NOTE: The interface and SSID in the configuration file should be different.

1.11.6 SoftAP + STA mode

Setup SoftAP session

1. Configure SoftAP function.

```
# ifconfig wlan0 up
# hostapd hostapd.conf &
```

2. Set IP address and turn on DHCP server.

```
# ifconfig wlan0 192.168.11.1
# touch /var/run/udhcpd-ap.leases
# udhcpd udhcpd-ap.conf -fS &
```

Setup STA session

3. Create a STA interface.

```
# iw dev wlan0 interface add wlan1 type station
```

4. Configure STA function.

```
# ifconfig wlan1 up
# wpa_supplicant –D nl80211 -i wlan1 -c wpa-sta.conf &
```

5. Set IP address and turn on DHCP server

```
# udhcpc -i wlan1 &
```

1.11.7 SoftAP + P2P mode

Setup SoftAP session

1. Configure SoftAP function.

```
# ifconfig wlan0 up
# hostapd hostapd.conf &
```

2. Set IP address and turn on DHCP server.

```
# ifconfig wlan0 192.168.11.1
# touch /var/run/udhcpd-ap.leases
# udhcpd udhcpd-ap.conf -fS &
```

Setup P2P session

3. Configure P2P function and enable P2P interface.

```
# wpa_supplicant -i p2p0 -c p2pdev_dual.conf &
# wpa_cli -i p2p0
```

4. Start an autonomous GO.

```
> p2p_group_add
```

5. Run the “CTRL+C” to exit and run the wps_pbc command:

```
# wpa_cli -i p2p0 wps_pbc
```

6. If you need turn on DHCP server.

```
# ifconfig p2p0 192.168.12.1
# touch /var/run/udhcpd-ap.leases
# udhcpd udhcpd-P2P.conf -fS &
```

NOTE: If SoftAP and P2P working in same frequency, you must set the same channel.

1.11.8 Setup WPA3 security

Setup 11ng SoftAP with WPA3 SAE Mode

1. Edit the default configuration file “hostapd.conf" as following:

```
ctrl_interface=/var/run/hostapd
```

```
interface=wlan0
driver=nl80211
hw_mode=g
ieee80211n=1
channel=11
ssid=Yocto
auth_algs=3
wpa=2
wpa_passphrase=12345678
wpa_key_mgmt=SAE
rsn_pairwise=CCMP
ieee80211w=2
```

2. Start the hostapd with “hostapd.conf".

```
# hostapd hostapd.conf &
```

3. Set IP address and turn on DHCP server.

```
# ifconfig wlan0 192.168.11.1
# touch /var/run/udhcpd-ap.leases
# udhcpd udhcpd-ap.conf -fS &
```

4. Setup STA to connect to this SoftAP with WPA3 SAE mode and use "ping" to verify the connection.

Setup 11ng SoftAP with WPA3 SAE + WPA2 PSK transition Mode

1. Edit the default configuration file “hostapd.conf" as following:

```
ctrl_interface=/var/run/hostapd
interface=wlan0
driver=nl80211
hw_mode=g
ieee80211n=1
channel=11
ssid=Yocto
auth_algs=3
wpa=2
wpa_passphrase=12345678
wpa_key_mgmt=SAE WPA-PSK
rsn_pairwise=CCMP
ieee80211w=1
```

2. Start the hostapd with “hostapd.conf".

```
# hostapd hostapd.conf &
```

3. Set IP address and turn on DHCP server.

```
# ifconfig wlan0 192.168.11.1
# touch /var/run/udhcpd-ap.leases
# udhcpd udhcpd-ap.conf -fS &
```

4. Setup STA to connect to this SoftAP with WPA3 SAE or WPA2 PSK mode and use "ping" to verify the connection.

Setup 11ng SoftAP with WPA3 OWE Mode

1. Edit the default configuration file “hostapd.conf" as following:

```
ctrl_interface=/var/run/hostapd
interface=wlan0
driver=nl80211
hw_mode=g
ieee80211n=1
channel=11
ssid=Yocto
wpa=2
wpa_key_mgmt=OWE
rsn_pairwise=CCMP
ieee80211w=2
```

2. Start the hostapd with “hostapd.conf".

```
# hostapd hostapd.conf &
```

3. Set IP address and turn on DHCP server.

```
# ifconfig wlan0 192.168.11.1
# touch /var/run/udhcpd-ap.leases
# udhcpd udhcpd-ap.conf -fS &
```

4. Setup STA to connect to this SoftAP with WPA3 OWE mode and use "ping" to verify the connection.

Setup 11ng SoftAP with WPA3 OWE Transition Mode

1. Edit the default configuration file “hostapd.conf" as following:

```
ctrl_interface=/var/run/hostapd
interface=wlan0
driver=nl80211
hw_mode=g
ieee80211n=1
channel=11
ssid=Yocto
owe_transition_ifname=wlan0_1
bss=wlan0_1
ssid=Yocto_OWE
owe_transition_ifname=wlan0
wpa=2
wpa_key_mgmt=OWE
rsn_pairwise=CCMP
ieee80211w=2
ignore_broadcast_ssid=1
use_driver_iface_addr=1
```

2. Start the hostapd with “hostapd.conf".

```
# hostapd hostapd.conf &
```

3. Set IP address and turn on DHCP server.

```
# ifconfig wlan0 192.168.11.1
```

```
# touch /var/run/udhcpd-ap.leases
# udhcpd udhcpd-ap.conf -fS &
# ifconfig wlan0_1 192.168.12.1
# touch /var/run/udhcpd-ap1.leases
# udhcpd udhcpd-ap1.conf -fS &
```

NOTE: The udhcpd-ap.conf is configuration wlan0, the udhcpd-ap1.conf is configuration wlan0_1.

4. Setup STA to connect to this SoftAP with WPA3 OWE or WPA2 mode and use "ping" to verify the connection.

Connect to WPA3 SAE mode AP

1. Edit the configuration file "wpa-sta.conf" content as following from console.

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1
pmf=1
network={
ssid="SAE"
psk="12345678"
key_mgmt=SAE
pairwise=CCMP
group=CCMP
group_mgmt=AES-128-CMAC
ieee80211w=2
}
```

2. Enter the below command from console to connect wlanX with AP.

```
# wpa_supplicant -D nl80211 -i wlan0 -c wpa-sta.conf &
```

3. Enter the below command from console, after wlan0 is connected to AP, to set the IP address

```
# udhcpc -i wlan0
```

4. Ping the AP IP address to check the connectivity

```
# ping 192.168.1.1
```

Connect to WPA3 OWE mode AP

1. Edit the configuration file "wpa-sta.conf" content as following from console.

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1
pmf=1
network={
ssid="OWE"
key_mgmt=OWE
pairwise=CCMP
group=CCMP
group_mgmt=AES-128-CMAC
```

```
ieee80211w=2
owe_group=19
}
```

2. Enter the below command from console to connect wlanX with AP.

```
# wpa_supplicant –D nl80211 -i wlan0 -c wpa-sta.conf &
```

3. Enter the below command from console, after wlan0 is connected to AP, to set the IP address

```
# udhcpc -i wlan0
```

4. Ping the AP IP address to check the connectivity

```
# ping 192.168.1.1
```

1.11.9 Setup WAPI security

Connect to WAPI HEX mode AP

1. Edit the default configuration file “wpa_supplicant_hex.conf" as following:

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
network={
        ssid="WAPI-PSK-HEX"
        proto=WAPI
        key_mgmt=WAPI-PSK
        pairwise=SMS4
        group=SMS4
        wapi_key_type=1
        wapi_psk="123456789"
}
```

2. Enter the below command from console to connect wlanX with AP.

```
# wpa_supplicant -i wlan0 -c wpa_supplicant_hex.conf -Dnl80211 -ddKt &
```

3. Enter the below command from console, after wlan0 is connected to AP, to set the IP address

```
# udhcpc -i wlan0
```

4. Ping the AP IP address to check the connectivity

Connect to WAPI ASCII mode AP

1. Edit the default configuration file “wpa_supplicant_asc.conf" as following:

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
network={
        ssid="WAPI-PSK-ASCII"
        proto=WAPI
        key_mgmt=WAPI-PSK
        pairwise=SMS4
        group=SMS4
        wapi_key_type=0
        wapi_psk="123456789"
```

```
}
```

2. Enter the below command from console to connect wlanX with AP.

```
# wpa_supplicant -i wlan0 -c wpa_supplicant_asc.conf -Dnl80211 -ddKt &
```

3. Enter the below command from console, after wlan0 is connected to AP, to set the IP address

```
# udhcpc -i wlan0
```

4. Ping the AP IP address to check the connectivity

Connect to WAPI CERT mode AP

1. Edit the default configuration file “wpa_supplicant_cert.conf" as following:

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
network={
        ssid="WAPI-CERT"
        key_mgmt=WAPI-CERT
        as_cert_file="/data/as.cer"
        user_cert_file="/data/ASUE.cer"
}
```

2. Enter the below command from console to connect wlanX with AP.

```
# wpa_supplicant -i wlan0 -c wpa_supplicant_cert.conf -Dnl80211 -ddKt &
```

3. Enter the below command from console, after wlan0 is connected to AP, to set the IP address

```
# udhcpc -i wlan0
```

4. Ping the AP IP address to check the connectivity

2 Supported ASICs

This software can be used with the supported ASICs (Table 2-1) and revisions, with the indicated release quality. ASIC revisions available at the time of this release are assumed to be supported, unless otherwise indicated.

Table 2-1 Supported ASICs

ASIC hardware	
Interface	
PCIe	SDIO
QCA6564A-1	QCA6564A-3
QCA6564AU-1	QCA6564AU-3
QCA6574A-1	QCA6574A-3
QCA6574AU-1	QCA6574AU-3

Platform information

The following platforms are supported in this release:

- IMX6 SABRESDB + Y6133+ QCA65X4A/AU-1 + 30-Y9183-1+ QCA65X4A/AU-3
- IMX6 IMX8QXPMEK + Y8878(QCA65X4A/AU-1)
- Software platform version

The following software platforms are supported in this release:

- Linux kernel – Ver 4.9
- Linux kernel – Ver 4.14
- Linux kernel – Ver 5.4

3 Supported Features

This chapter describes the new software features introduced in this release and a description of those features.

3.1 General features

- Support i.MX6 Linux kernel 4.9/4.14. Wi-Fi functionality and FTM tools.
- Support i.MX8 Linux kernel 4.14/5.4. Wi-Fi functionality and FTM tools.
- Added in Post CS
  - Support WPA3 R3 SAE feature for Linux Kernel 4.9/4.14/5.4.
  - Pass WPA3 R3 SAE certification test at Linux Kernel 5.4.
- Added in Post CS5
  - Support SDIO interface for Linux Kernel 4.9/4.14/5.4
  - Support quick track tool for Linux Kernel 5.4
  - Support firmware ramdump for Linux Kernel 4.9/4.14/5.4
  - Fix Blacklist issue for Linux Kernel 4.9/4.14/5.4
- Added in Post CS6
  - Remove unnecessary logs on SDIO interface for Linux Kernel 4.9/4.14/5.4.
  - Support China's new MIIT wireless regulatory requirements for Linux Kernel 4.9/4.14/5.4.
  - Support get FW state by tool iwpriv for Linux Kernel 4.9/4.14/5.4.
  - Fix Several Quick Track certification issues for Linux Kernel 4.9/4.14/5.4.
  - Support iptables and OPENSSH at I.MX image.
  - fix Throughput issue for LK4.14 SDIO interface
  - update manifest and patch path from CAF to CLO.

3.2 WLAN features

IEEE Standards

Feature & Spec	Support Ability	Feature & Spec	Support Ability
802.11a	Yes	802.11h	Yes
802.11b	Yes	802.11i	Yes
802.11g	Yes	802.11k	Yes
802.11n	Yes	802.11r	Yes

Feature & Spec	Support Ability	Feature & Spec	Support Ability
802.11ac	Yes	802.11u	No
802.11ad	No	802.11v	No
802.11ax	No	802.11w	Yes
802.11d	Yes	802.11mc	Yes
802.11e	Yes	802.11p	No

Channel Bonding

Channel Bonding	Support Ability
40MHz	Yes
80MHz	Yes
160MHz	No

Guard Interval

Guard Interval	Support Ability
400ns	Yes
800ns	Yes

Security

Security					Support Ability
Pre-RSNA	Open System	Open System Authentication	-	-	Yes
	WEP	Open System Authentication	WEP64	ARC4	Yes
			WEP128	ARC4	Yes
		Shared Key Authentication	WEP64	ARC4	Yes
			WEP128	ARC4	Yes
RSNA	WPA Personal	Pre-shared Key Authentication	TKIP	ARC4	Yes
			CCMP	ARC4	Yes
	WPA2 Personal	Pre-shared Key Authentication	TKIP	ARC4	Yes
			CCMP	AES	Yes
	WPA3 Personal	Simultaneous Authentication of Equals	CCMP	AES	Yes
WAPI	WAPI	Pre-shared Key Authentication	SMS4	ECC192	Yes

NOTE: WAPI: STA only

Mode

Mode		Support Ability
STA		Yes
AP		Yes
Direct	P2P Group Owner	Yes

Mode		Support Ability
	P2P Client	Yes
	Multi connection	Yes

NOTE: Multi connection means multiclient connection

QoS

QoS		Support Ability
WMM	EDCA	Yes
WMM Power Save	U-APSD	Yes
WMM Voice Personal		No
WMM Admission Control		Yes

4 Bugs and limitations

This chapter lists the bugs and limitations reported for this product line:

- New – Newly reported bugs and limitations
- Ongoing – Previously reported bugs and limitations
- Resolved – Previously reported bugs and limitations that have been resolved and are no longer relevant

Known CRs are selected based on information available at the time of release taking into accounting the following:

- CR is applicable to a software product.
- CR changes are likely to be included in an upcoming release.

However, due to the dynamic nature of the development environment, the schedule, and actual contents of upcoming releases are subject to change.

CAUTION: The software does not configure the QTI baseband device (MSM™, MDM, and so on) input pins to ensure minimum sleep current for customer-defined pin mux configurations. Therefore, it is the customer's responsibility to properly configure every digital CMOS input pin to a valid voltage level before entering sleep. This is required to minimize leakage current during sleep. This applies to EBI data pins, GPIO pins, alternate functions multiplexed with GPIOs (that is, AUX_JTAG or SEL pins), and any other CMOS pin that is configured as an input. See *Configuration of Input Pins during Device Sleep* (80-VN499-7) for more details.

NOTE: For a list of all completed and known Change Requests (CRs), refer to the release history tab in Qualcomm CreatePoint for the product

4.1 New bugs

This section is not applicable to this release.

4.2 Limitations

WLAN

This section is not applicable to this release.

BT

This section is not applicable to this release.

4.3 Ongoing bugs

This section is not applicable to this release.

4.4 Resolved

Area	CR	Description
WLAN Driver	3270266	Disable Tx/Rx STBC in 1x1 mode.
WLAN Driver	2880892	qcacld-3.0: drop the Tx pkt if device is in system suspend
WLAN Driver	3496034	qcacld-2.0: Fix compile error caused by max_num_csa_counters
WLAN Driver	2558910	Replace get_monotonic_boottime() with ktime_get_boottime()

4.5 Known issues

No issue in this version.

4.6 Change request

This section is not applicable to this release.

4.7 Dependency information

This section is not applicable to this release.

5 Test reports

5.1 CNSS test report

5.1.1 WLAN coverage ratio table

This section is not applicable to this release.

5.1.2 WLAN functional sanity report

QCA6574AU.LE.2.2.1	
Test Case	**Test Result**
WLAN STA scan	Pass
WLAN STA connection in open mode	Pass
WLAN STA connection in WPA2-PSK mode	Pass
WLAN STA Roaming in open mode	Pass
WLAN start SAP in open/WPA2-PSK mode and another use STA device to connect it	Pass
Verify WLAN card 802.11 a/b/g/n basic function	Pass
Ping traffic between STA and external AP	Pass
WLAN P2P mode connection	Pass
WLAN Autonomous Go connection	Pass

5.1.3 WLAN detailed cases

This section is not applicable to this release.

5.1.4 WLAN KPI regression report

Peer Test Device	:	Wi-Fi@ AP/P2P: NETGEAR7800; STA: MacBook Pro
Test Environment	:	Chamber for ES10
	:	QCA6574AU.LE.2.2.1
HW	:	i.MX6 SABRE for Automotive platform
SW	:	Kernel: 4.9.11
		Host SW: 5.2.0.190I
		FW: WLAN.RM.4.5.3.r2-00003-QCARMSWRZ-1

Peer Test Device	:	Wi-Fi@ AP/P2P: NETGEAR7800; STA: MacBook Pro
Test Environment	:	Chamber for CS
	:	QCA6574AU.LE.2.2.1
HW	:	i.MX8 QXPMEK for Automotive platform
SW	:	Kernel: 5.4
		Host SW: 5.2.0.237G
		FW: WLAN.RM.4.5.3-00153-QCARMSWRZ-1

Mode	Traffic Type	Band/Channel	Linux 4.9 SDIO2.0 TPut (Mbps)	Linux 5.4 PCIE TPut (Mbps)
STA KPI	UDP RX	2.4G: Channel 6: 11NGHT20MHz	106	110
	UDP TX		111	124
	TCP RX		79	115
	TCP TX		101	93
	UDP RX	5G: Channel 161: VHT80MHz	110	734
	UDP TX		125	596
	TCP RX		83	661
	TCP TX		111	431
SAP KPI	UDP RX	2.4G: Channel 6: 11NGHT20MHz	105	123
	UDP TX		112	118
	TCP RX		78	115
	TCP TX		96	119
	UDP RX	5G: Channel 149: VHT80MHz	105	584
	UDP TX		125	457
	TCP RX		80	479
	TCP TX		112	454

5.2 CTS/CTSV/GTS report

This section is not applicable to this release.

5.3 Product stability report

This section is not applicable to this release.

5.4 Power dashboard

This section is not applicable to this release.

5.5 Hardware brings up test (BUT) report

This section is not applicable to this release.

5.6 WFA test report

Platform SW Version : CHSS.LNX_FSL.5.0-00400-QCA6574AUARMSDIOHZ-1

CHSS.LNX_FSL.5.0-01200-QCA6574AUARMSDIOHZ-1(for WPA3 SAE R3 feature)

Platform HW Version : iMx8

Wi-Fi Chip Model : QCA6574AU-1

5.6.1 11n-APUT report

11n-APUT	
Test Case ID	**Test Result**
4.2.6	Pass
4.2.8	Pass
4.2.9	Pass
4.2.10	Pass
4.2.21 (WMM)	Pass
4.2.23 (WMM)	Pass
4.2.26	Pass
4.2.27	Pass
4.2.35	Pass
4.2.43	Pass
4.2.47	Pass

NOTE: set ini value “sifs_burst_mask=3， gEnablefwlog=0” if the test result can’t achieve target.

5.6.2 11n-STAUT report

11n-STAUT	
Test Case ID	**Test Result**
5.2.3	Pass
5.2.5 (Only for 2.4Ghz)	Pass
5.2.11	Pass
5.2.14	Pass
5.2.26	Pass
5.2.27 (WMM)	Pass
5.2.30 (WMM)	Pass
5.2.31 (WMM)	Pass
5.2.32 (WMM)	Pass

5.2.33 (WMM)	Pass
5.2.35	Pass
5.2.38	Pass
5.2.39 (Only for 2.4Ghz)	Pass
5.2.43	Pass
5.2.51	Pass
5.2.52	Pass
5.2.55	Pass

5.6.3 11ac-STAUT report

11ac-STAUT	
Test Case ID	**Test Result**
5.2.9.1	Pass
5.2.10.1	Pass
5.2.27.1	Pass
5.2.28.1	Pass
5.2.32.1	Pass
5.2.33.1	Pass
5.2.34.1	Pass
5.2.35.1	Pass
5.2.37.1	Pass
5.2.38.1	Pass
5.2.40.1	Pass
5.2.50.1	Pass
5.2.55	Pass
5.2.57	Pass
5.2.61	Pass
5.2.62	Pass

5.6.4 11ac-APUT report

11ac-APUT	
Test Case ID	**Test Result**
4.2.1.1	Pass
4.2.2.1	Pass
4.2.5.1	Pass
4.2.5.2	Pass
4.2.20.1	Pass
4.2.21.1	Pass
4.2.25.1	Pass
4.2.26.1	Pass
4.2.30.1	Pass

4.2.40.1	Pass
4.2.43.1	Pass
4.2.48	Pass
4.2.50	Pass
4.2.51	Pass
4.2.55	Pass

5.6.5 WMMPS-STAUT report

WMMPS-STAUT	
Test Case ID	**Test Result**
5.2 (M.D)	Pass
5.3 (B.B)	Pass
5.4 (B.K)	Pass
5.5 (B.W)	Pass
5.7 (L.1)	Pass
5.8 (M.B)	Pass
5.10 (M.W)	Pass

NOTE: For case 5.2,5.7, set ini “UapsdMask=0xf ,gMaxPsPoll=5,WmmIsEnabled=1”

NOTE: For case 5.3,5.4,5.5 set ini “UapsdMask=0xf”

NOTE: For case 5.8,5.10, set ini “UapsdMask=0x3”

5.6.6 WMMPS-APUT report

WMMPS-APUT	
Test Case ID	**Test Result**
4.1	Pass
4.3 (B.H)	Pass
4.5 (A.J)	Pass
4.6 (B.M)	Pass
4.7 (L.1)	Pass
4.8 (A.Y)	Pass
4.10 (M.V)	Pass
4.12 (A.U)	Pass

5.6.7 PMF-STAUT report

PMF-STAUT	
Test Case ID	**Test Result**
5.1	Pass
5.3.3.2	Pass

5.3.3.5	Pass
5.4.3.1	Pass
5.4.3.2	Pass

5.6.8 PMF-APUT report

PMF-APUT	
Test Case ID	**Test Result**
4.3.3.2	Pass
4.3.3.4	Pass
4.5	Pass

5.6.9 WPS2-STAUT report

WPS2-STAUT	
Test Case ID	**Test Result**
5.1.1	Pass
5.1.3	Pass
5.1.5	Pass
5.1.6	Pass
5.1.8	Pass
5.1.11	Pass
5.1.12	Pass
5.1.13	Pass
5.1.14	Pass
5.1.19	Pass
5.4.1	Pass
5.4.2	Pass
5.4.3	Pass
5.4.4	Pass

5.6.10 WPS2-APUT report

WPS2-APUT	
Test Case ID	**Test Result**
4.1.1	Pass
4.1.8	Pass
4.1.13	Pass
4.2.1	Pass
4.2.2	Pass
4.2.4	Pass
4.2.10	Pass
4.2.13	Pass

4.2.14	Pass
4.3.2	Pass
4.4.1	Pass

5.6.11 KRACK report

KRACK	
Test Case ID	**Test Result**
5.2.1	Pass
5.2.2	Pass
5.2.3	Pass
5.2.4	Pass
5.2.5	Pass
5.3.1	Pass

5.6.12 FFD-STA report

FFD-STA	
Test Case ID	**Test Result**
5.2.1	Pass
5.2.2	Pass
5.3.1	Pass
5.3.2	Pass
5.4.1	Pass
5.4.2	Pass
5.4.3	Pass
5.4.4	Pass
5.5.1	Pass
5.6.1	Pass
5.6.2	Pass
5.6.3	Pass
5.6.4	Pass
5.6.5	Pass
5.7.1	Pass
5.7.2	Pass
5.8.1	Pass
5.8.2	Pass
5.8.3	Pass
5.8.4	Pass

5.6.13 FFD-AP report

FFD-AP	
Test Case ID	**Test Result**
4.2.1	Pass
4.2.2	Pass
4.3.1	Pass
4.3.2	Pass
4.4.1	Pass
4.4.2	Pass
4.4.3	Pass
4.4.4	Pass
4.5.1	Pass
4.6.1	Pass
4.6.2	Pass
4.6.3	Pass
4.6.4	Pass
4.6.5	Pass
4.7.1	Pass
4.7.2	Pass
4.8.1	Pass
4.8.2	Pass
4.8.3	Pass
4.8.4	Pass

5.6.14 WPA2_improvement-STA report

WPA2_improvement-STA	
Test Case ID	**Test Result**
5.2.2	Pass
5.2.3	Pass
5.2.4	Pass

5.6.15 WPA2_improvement-AP report

WPA2_improvement-AP	
Test Case ID	**Test Result**
4.2.2	Pass
4.2.3	Pass
4.2.4	Pass

5.6.16 WPA3_OWE-APUT report

WPA3_OWE-APUT	
Test Case ID	**Test Result**
4.2.1	Pass
4.2.2	Pass
4.2.3	Pass
4.2.4	Pass
4.2.5	Pass

5.6.17 WPA3_OWE-STAUT report

WPA3_OWE-STAUT	
Test Case ID	**Test Result**
5.2.1	Pass
5.2.2	Pass
5.2.3	Pass
5.2.4	Pass
5.2.5	Pass
5.2.6	Pass

5.6.18 WPA3_SAE-R3-APUT report

WPA3_SAE-APUT	
Test Case ID	**Test Result**
4.2.1	Pass
4.2.2.1	Pass
4.2.2.2	Pass
4.2.2.3	Pass
4.2.2.4	Pass
4.2.3.1	Pass
4.2.3.2	Pass
4.2.4	Pass
4.2.5.1	Pass
.2.5.2	Pass
4.2.6	Pass
4.2.7	Pass
4.2.8	Pass
4.3	Pass
4.4.1	Pass
4.4.2	Pass
4.4.3	Pass
4.4.4	Pass
4.6.1	Pass

5.6.19 WPA3_SAE-R3-STAUT report

WPA3_SAE-STAUT	
Test Case ID	**Test Result**
5.2.1.1	Pass
5.2.1.2	Pass
5.2.1.3	Pass
5.2.2.1	Pass
5.2.2.2	Pass
5.2.3	Pass
5.2.4	Pass
5.2.5.1	Pass
5.2.5.2	Pass
5.2.6	Pass
5.2.7	Pass
5.3	Pass
5.4.2	Pass
5.6.1	Pass

6 Setup WLAN Certification Tools

The purpose of this section is to outline procedures to be used to obtain certifications offered by the Wi-Fi Alliance. The scope is limited to the Linux Android operating system. The information is intended to provide directions while running certification tests and will be updated accordingly to reflect changes in procedures as demanded by the operating system.

6.1 Configuration

The 3rd party platform image must have the following utility for wpa certification.

- udhcpd
- udhcpc
- ping with interval setting i.e. which must supports "-i"
- sigma_dut
- cert.tools

Since FSL iMX8 platform is using yocto Linux 5.4.70, the BSP must enable busybox to support udhcpd and udhcpc. The ping with interval setting should enable iptuils as well as need adding sigma_dut recipes in local.cof of yocto project. The section 11.3 already did it to support above.

The cert.tools is the bringup script files which has been included in wlan-sigmadut recipes and could be installed in the BSP image automatically.

6.2 Bring up sigma_dut for 11n/11ac certification

1. Connect the DUT to the 3rd party platform via PCIE.
2. Start sigma_dut using script files
   - i STA mode
     (a) # ./ sta.sh

   Edit the default configuration file "sta.sh " as following:

```
$ insmod /lib/modules/5.4.70-2.3.2/extra/wlan.ko
$ ifconfig wlan0 up
$ ifconfig eth0 192.168.250.150 netmask 255.255.0.0
$ wpa_supplicant -i wlan0 -c /usr/share/misc/wifi/wpa_supplicant.conf -Dnl80211 &
sigma_dut -3 -M wlan0 -c LINUX-WCN -S wlan0 &
```

ii SoftAP mode

(a) # ./ sap.sh

Edit the default configuration file “sap.sh " as following:

```
$ insmod /lib/modules/5.4.70-2.3.2/extra/wlan.ko
$ ip addr flush dev wlan0
$ ip addr flush dev eth0
$ brctl addbr br0
$ ifconfig eth0 up
$ ifconfig wlan0 up
$ ifconfig br0 up
$ brctl addif br0 eth0
$ brctl addif br0 wlan0
$ ip link set dev br0 up
$ ifconfig br0 192.165.100.40 netmask 255.255.0.0 up
$ ifconfig wlan0 192.165.100.150 netmask 255.255.0.0 up
$ sigma_dut -M wlan0 -c LINUX-WCN -S wlan0 -b br0 -i 192.165.100.150 -k
 255.255.0.0 -T 1470 &
$ ip addr flush dev eth0
```

6.3 Bring up sigma_dut for WPA3 R3 Certification

1. Connect the DUT to the 3rd party platform via PCIE.
2. Start sigma_dut using scripts below

i STA mode

```
$ insmod /lib/modules/5.4.70-2.3.2/extra/wlan.ko
$ ifconfig wlan0 up
$ ifconfig eth0 192.168.250.150 netmask 255.255.0.0
$ wpa_supplicant -i wlan0 -c /usr/share/misc/wifi/wpa_supplicant.conf -
Dnl80211 &
$ sigma_dut -2 -M wlan0 -c LINUX-WCN -S wlan0 &
```

ii SoftAP mode

```
$ insmod /lib/modules/5.4.70-2.3.2/extra/wlan.ko
$ ifconfig wlan0 up
$ ifconfig eth0 192.168.250.254 netmask 255.255.0.0 up
$ sigma_dut -2 -M wlan0 -c LINUX-WCN -S wlan0 -i 192.165.1.20 -k
255.255.0.0 -d &
```

6.4 Support quick track tool

The code of *common-tools* is located at *<target_root>/host_5.04/apps_proc/sources/wlan-proprietary.*

After flashing image, `ctrl_app_dut` can be found at */usr/bin.*

1. Connect the DUT to the 3rd party platform via eth0.
2. Load driver and set the IP address

```
$ insmod /lib/modules/5.4.70-2.3.2/extra/wlan.ko
$ ifconfig wlan0 up
$ ifconfig wlan0 192.165.1.150
```

3. Start ctrl_app_dut(for STA mode)

```
$ ifconfig eth0 192.168.1.150
$ ctrl_app_dut -i wlan0 -s /usr/sbin/wpa_supplicant
```

4. Start ctrl_app_dut(for AP mode)

```
$ ifconfig eth0 192.168.1.150
$ ctrl_app_dut -i wlan0 -s /usr/sbin/hostapd
```

NOTE: Need create folder "/etc/misc/wifi/" before running 'ctrl_app_dut -i wlan0 -s /usr/sbin/wpa_supplicant' or 'ctrl_app_dut -i wlan0 -s /usr/sbin/hostapd'

7 Klocwork Scan report

KW issues are show in the following sections.

7.1 WLAN HOST

Analysis Time	Build	Status
04/24/2022	CHSS.LNX_FSL.5.0-01800-QCA6574AUARMSDIOHI-1	OK

7.2 WLAN Firmware

Analysis Time	Build	Status
11/03/2022	WLAN.RM.4.5.3-00200-QCARMSWRZ-1	OK

7.3 Bluetooth Firmware

Analysis Time	Build	Status
03/28/2022	BTFM.RM.2.4.1-00056-QCABTFMSWPZ-2	OK

8 WLAN tools

8.1 Purpose

This section describes:

- How to use myftm
- How to use qcmbr
- How to use cnss_diag
- How to use athdiag
- How to use pktlogconf
- How to use SSR_Demo

8.2 Myftm

Myftm is a command line tool, used for QCA65x4A/AU WLAN Factory Test Mode test. Customer can use it to perform certain testing, e.g. transmit waveform, measure RSSI, calculate PER and so on.

8.2.1 WLAN RF test setup

The Figure 8-1 shows a general test setup to perform WLAN RF test. The device under test (DUT) is a target board comprised of WLAN chipset, ARM SOC. Myftm runs on target system and works in user space. The vector signal analyzer (VSA) captures data from the DUT, and the vector signal generator(VSG) generates data to the DUT. Host PC is used to input commands to DUT's myftm tool through micro USB or serial port.

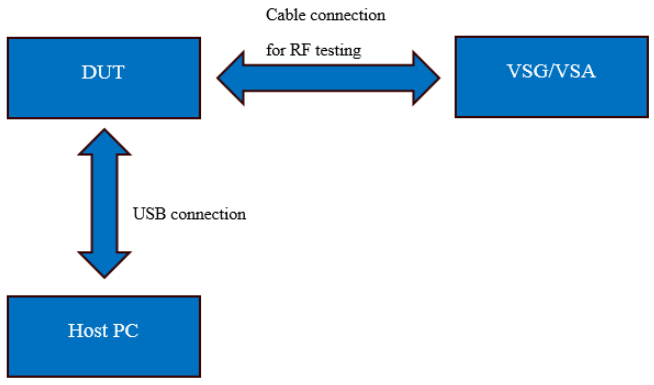


Figure 8-1 WLAN RF test with myftm tool

8.2.2 Myftm parameters

Table 8-1 myftm parameters

Parameter	Functionality	Input range	Description
-r	WlanATSetRate	1~30	The Tx/Rx data rate index to test on; please see Table 5-2 for more information
-f	WlanATSetWifiFreq	2412~5825	Frequencies (MHz)
-p	WlanATSetWifiTxPower	0~20	Tx power (dBm), such as 15
-a	WlanATSetWifiAntenna	1	Chain 0
		2	Chain 1
		3	Both Chain 0 and Chain 1 (2x2)
-G	Gain Index	0-31	Gain index is considered only if TPC is set to 4
-D	Dac Gain	-48 ~ 128	Dac Gain is considered only if TPC is set to 4
-j	Number of packets		Number of packets to be transmitted; by default, it is 0, which means to continue Tx.
-k	Aggregation	0	Disable Aggregation (default value)
		1	Enable Aggregation
-M	WLAN mode	0	TCMD_WLAN_MODE_NOHT
		1	TCMD_WLAN_MODE_HT20
		2	TCMD_WLAN_MODE_HT40PLUS
		4	TCMD_WLAN_MODE_CCK
		5	TCMD_WLAN_MODE_VHT20
		6	TCMD_WLAN_MODE_VHT40PLUS
		8	TCMD_WLAN_MODE_VHT80_0
-C	PA CFG	0 ~ 3	
-c	TPC	0	TPC_TX_PWR
		1	TPC_FORCED_GAIN
		2	TPC_TGT_PWR
		3	TPC_TX_FORCED_GAIN
		4	TPC_FORCED_GAINIDX
		5	TPC_FORCED_TGTPWR
-l	Enable long preamble	No argument	Enable long preamble; by default, short preamble is enabled
-s	Tx packet size	Valid packet size	By default, it is set to 1500
-t	WlanATSetWifiTX	0	Tx off
		1	TCMD_CONT_TX_SINE
		2	TCMD_CONT_TX_FRAME
		3	TCMD_CONT_TX_TX99
-x	WlanATSetWifiRX	0	Rx stop
		1	Rx start

Following table list data rate indexes and relevant data rates that can be specified with the -r parameter.

Table 8-2 Rate Index mapping

Rate Index	Data rate	Note
1	1M	Legacy 802.11a/b/g data rate
2	2M	
3	5.5M	
4	6M	
5	9M	
6	11M	
7	12M	
8	18M	
10	24M	
12	36M	
13	48M	
14	54M	
15	MCS0	802.11n or 802.11ac HT/VHT data rate
16	MCS1	
17	MCS2	
18	MCS3	
19	MCS4	
20	MCS5	
21	MCS6	
22	MCS7	
23	MCS8	
24	MCS9	
25	MCS10	
26	MCS11	
27	MCS12	
28	MCS13	
29	MCS14	
30	MCS15	

8.2.3 Set up WLAN driver for FTM mode

1. Power up the Freescale SBARE board
2. Connect the board to a PC via micro USB cable or serial port line. Install the corresponding drivers when prompted by Windows XP/7
3. Open the serial port tool (baud rate=115200Hz) in PC
4. Type the following WLAN commands to enter FTM mode.

```
# insmod wlan.ko con_mode=5
```

8.2.4 Tx test examples

Use the following examples to test Tx performance

Example1:

Set the WLAN to continue Tx with 1 Mbps rate on chain0.

```
# myftm –M 4 –r 1 –f 2412 –c 0 –p 15 –a 1 –t 3
```

- -M 4 – Set the WLAN mode as CCK
- -r 1 – Rate is 1 Mbps
- -f 2412 – The central frequency is set to 2412 MHz (channel 1)
- -c 0 – The TPC is TPC_TX_PWR
- -p 15– Set fix Tx power as 15 dBm
- -a 1 – Tx on chain0
- -t 3 – Tx mode is CONT_TX_99: Continuous Modulated Tx 99% Duty Cycle

Example 2:

Set the WLAN to continue to Tx with MCS9 rate on VHT80 mode on both chain0 and chain1.

```
# myftm –a 3 –M 8 –c 0 –r 24 –f 5210 –p 10 –t 3
```

- -a 3 – Tx on both chain0 and chain1
- -M 8 – Set the WLAN mode as VHT80
- -c 0 – The TPC is TPC_TX_PWR
- -r 24 – MCS9
- -f 5210 – The central frequency is set to 5210 MHz
- -p 10 – Set fix Tx power as 10 dBm
- -t 3 – Tx mode is CONT_TX_99: Continuous Modulated Tx 99% Duty Cycle

To stop Tx, use following command:

```
# myftm –t 3
```

8.2.5 Rx test examples

Use the following example to test Rx performance.

Set the WLAN to Rx CCK 5.5 M rate packet on channel 1 and the antenna on chain0.

```
# myftm –M 4 –r 3 –f 2412 –a 1 –x 1
```

- -M 4 – CCK mode
- -r 3 – 5.5 M data rate
- -f 2412 – The central frequency is 2412 MHz (channel 1)
- -a 1 – Rx packet is on chain0
- -x 1 – Start the Rx packet

After executing this command, you can control the VSA to transmit relevant packets.

Use following command to exit from Rx mode:

```
# myftm –x 0
```

NOTE: All commands to myftm tool are classified as Tx or Rx operations so all the commands end with either the -t or -x option.

8.3 Qcmbr

The Qcmbr (QDART Connectivity for non-MSMTM Based Resources) application provides the communication path between QDART tool and on-chip firmware. It's used for RF calibration and FTM testing.

Qcmbr runs on target system and works in user space. It connects QDART tool via a TCP/IP socket, thus the target board should be in the same LAN with host PC. It connects on-chip firmware via the driver and operating system via IOCTL calls. The Figure 8-2 shows the connection between QDART and Qcmbr.

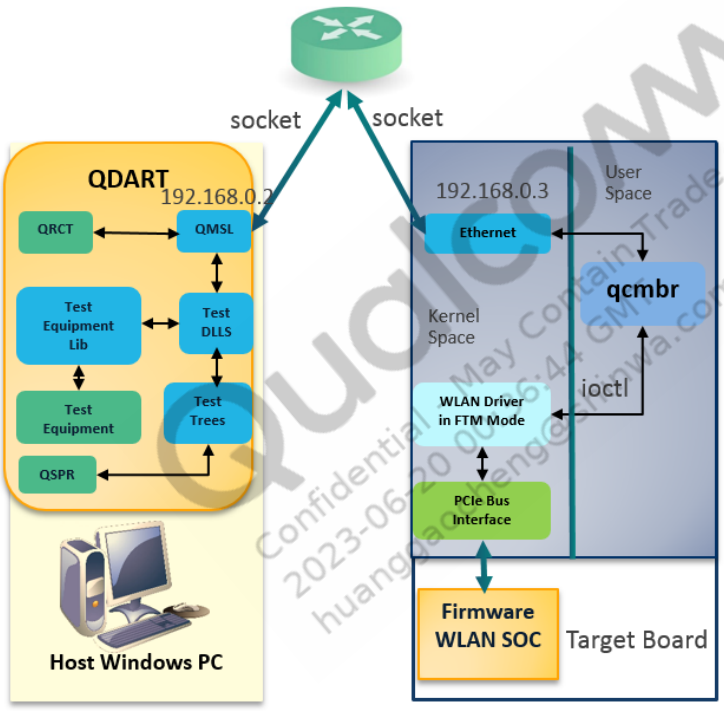


Figure 8-2 Connection between QDART and Qcmbr

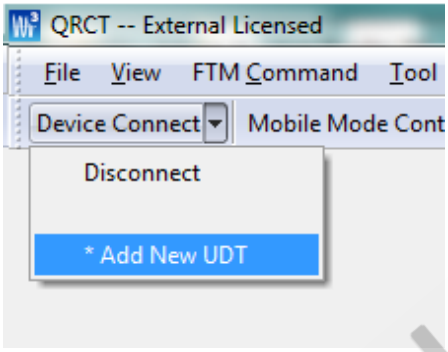
8.3.1 Set up Qcmbr for WLAN Test

1. Power up the Freescale SBARE board.
2. Connect the target board to PC via an Ethernet cable.
3. Set IP address of the target board to 192.168.0.3 and assign 192.168.0.2 to PC.
   ```
   # ifconfig eth0 192.168.0.3
   ```
4. Type the following WLAN command to enter FTM mode. Note the wlan.ko should be built with enable QCMBR.
   ```
   # insmod wlan.ko con_mode=5
   ```
5. Run Qcmbr, the default port used by Qcmbr is 2390. It can be specified by '-port xxx'.
   ```
   # sudo ./Qcmbr
   ```

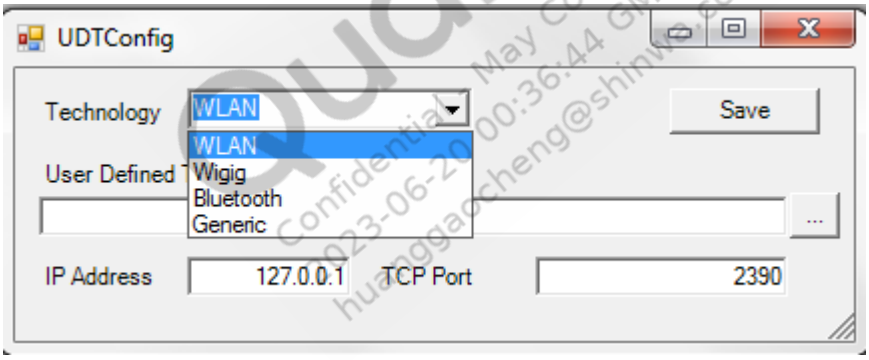
8.3.2 Set up QRCT for WLAN Test

The QDART.WIN.4.8.Installer is the full QDART package that supports other QCT chipsets. It can be obtained in https://createpoint.qti.Qualcomm.com by searching for QDART. Qualcomm Radio Control Toolkit (QRCT) tool is one component of QDART. It is a Windows toolkit that provides a GUI interface to FTM operation.

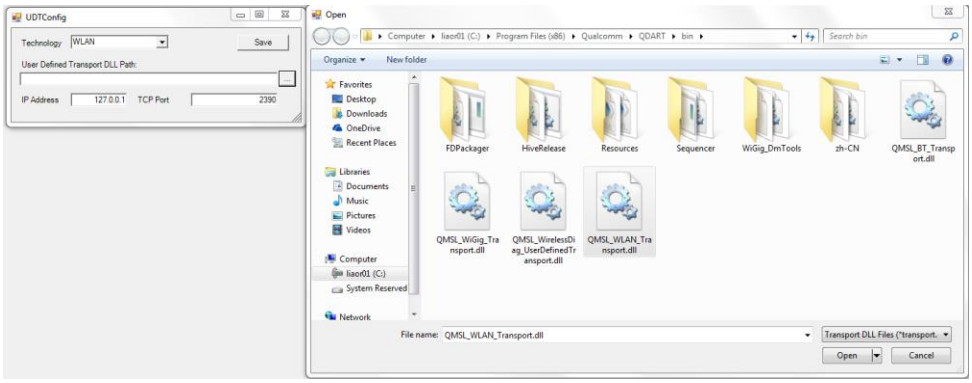
1. After installing QDART, launch QRCT in the Administrator mode
2. Select Tools > User Defined Transport
3. Select Device Connect > *Add New UDT



4. Select **WLAN** from the Technology drop-down list

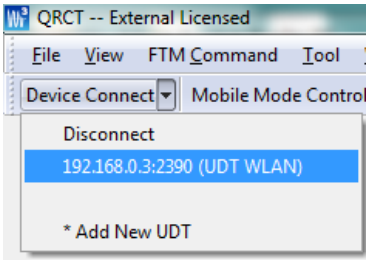


5. Open User Defined Transport DLL file as shown in the following image, select the QMSL_WLAN_Transport.dll.

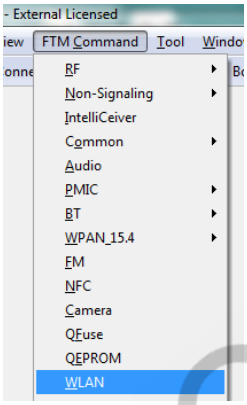


6. Input 192.168.0.3 in the IP Address item, 2390 in the TCP Port item. 192.168.0.3 is the IP address of target board. The IP address of host PC is set to 192.168.0.2, so that the PC and target board are in the same LAN. The TCP port 2390 is the default port used by Qcmbr.

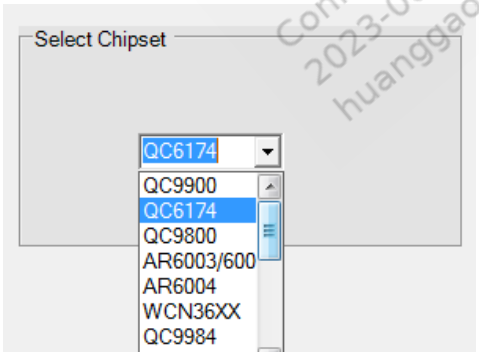
7. Click **Save**
8. Select **Device connect** > **192.168.0.3:2390(UDT WLAN)**, then similar log "processDiagPacket-succeed------Wait for Next Diag Packet---" appears on the terminal which runs Qcmbr tool.



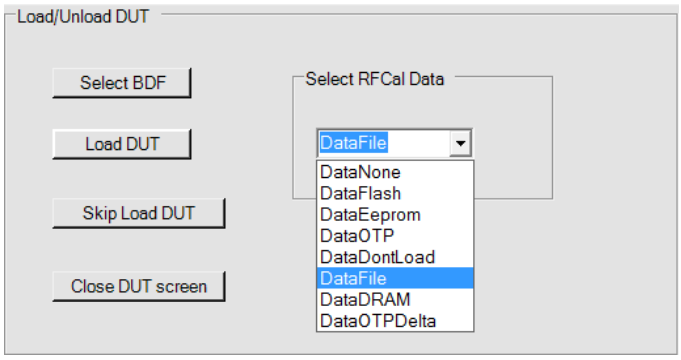
9. Select FTM Command > WLAN



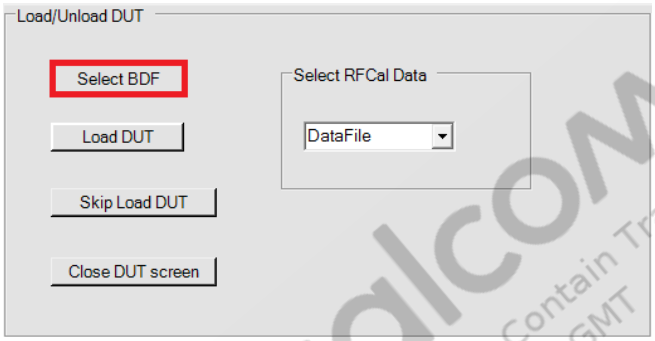
10. Select **QC6174** from the Select Chipset drop-down list



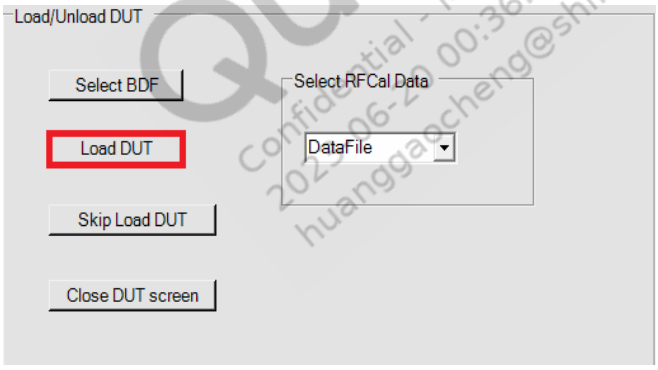
11. Select **DataFile** from the Select RFCal Data drop-down list



12. In Load/Unload DUT, click **Select BDF** and navigate to the corresponding BDF file. The bdwlan30.bin in release FW is the right BDF file.



13. Click **Load DUT**, then a DUT setting screen appears.



8.3.3 QRCT Tx test setting

After setting up the QRCT, a DUT transmitter settings screen appears as below. You can select the options of Tx Mode (Channel, TX Power Control, etc.) on the drop-down list as required. Click **SET TX ON** and to stop click **STOP TX**.

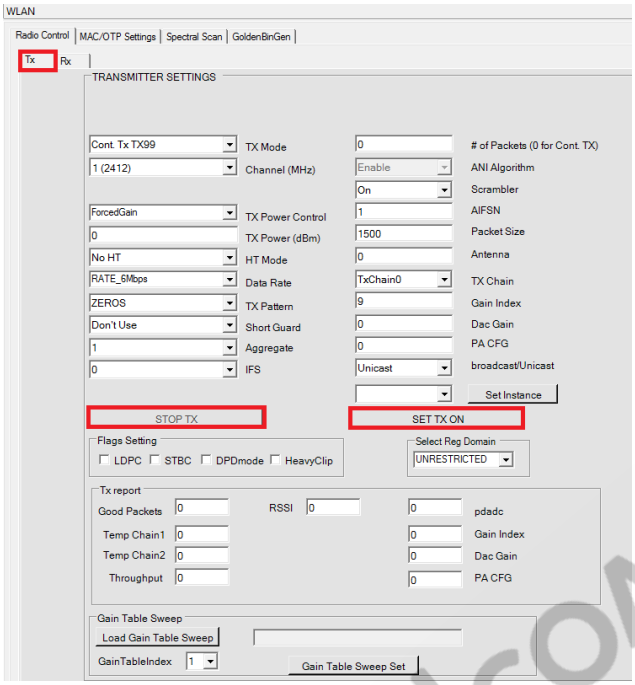


Figure 8-3 QRCT Tx test setting

8.3.4 QRCT Rx test setting

Click the **Rx** tab, then a DUT receiver setting screen appears as below. You can select the options of Rx Mode (Channel, HT Mode, etc.) as required. Then click **SET CONTINUOUS RX**. To stop RX, click the **GET RECEIVE REPORT**.

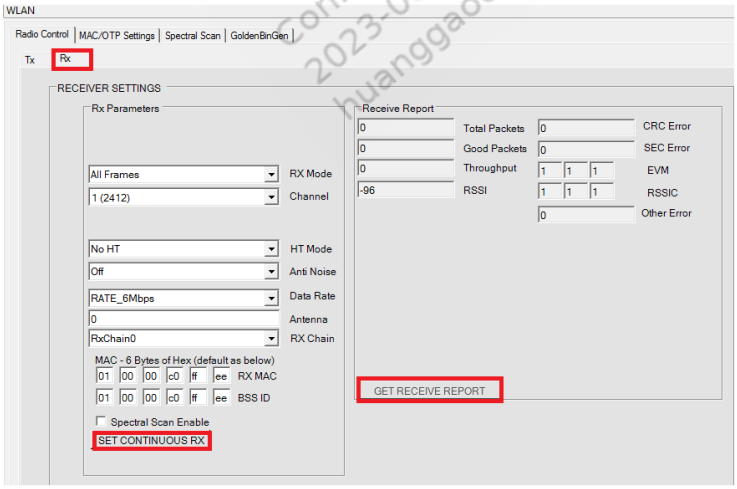


Figure 8-4 QRCT Rx test setting

8.4 Cnss_diag

Cnss_diag is a WLAN logging tool. It routes WLAN driver verbose messages and firmware debug information to console.

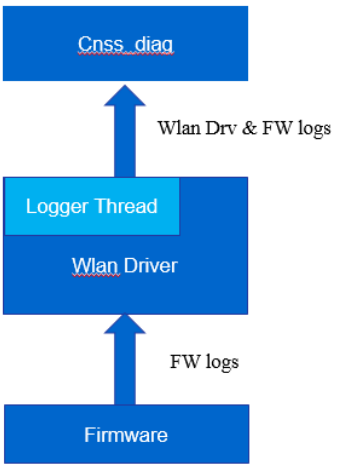


Figure 8-5 Log path from firmware to cnss_diag

Command options

```
Usage:
cnss_diag options
Options:
-f, --logfile(Currently file path is fixed)
-c, --console (prints the logs in the console)
-s, --silent (No print will come when logging)
-q, --qxdm  (prints the logs in the qxdm)
-d, --debug  (more prints in logcat, check logcat
-s ROME_DEBUG, example to use: -q -d or -c -d)
-b --buffer_size ( example to use : -b 64(in KBs)
-m --cnss_diag_config_file_loc ( example to use : -m /data/misc/cnss_diag.conf)
The options can also be given in the abbreviated form --option=x or -o x. The options can be given in any order
```

NOTE: Only “-s” and “-c” options are supported as of now.

```
# cnss_diag -f -c > driver_fw_log  &
# cat driver_fw_log
```

```
R1: FWMSG: [7815460] WMI -  WMI_CMD_NOT_HANDLED CmdId = 0x1d014
R0: FWMSG: [5488616] WMI_EVENT_SEND VdevId/EventId = 0x1d011 VdevMode = 0 EventId = 0x0
R1: FWMSG: [7815697] WMI -  WMI_EVENT_SEND VdevId/EventId = 0x1d011 VdevMode = 0 EventId = 0x0
R0: FWMSG: [5489642] WMI_EVENT_SEND VdevId/EventId = 0x1d011 VdevMode = 0 EventId = 0x0
R0: FWMSG: [5490418] WHAL_ERROR_POWER_SET  0x335 0x0 0x0 0x0 0x0
R0: FWMSG: [5490418] WHAL_ERROR_POWER_SET  0x335 0x1 0x0 0x0 0x0
R0: FWMSG: [5490418] WMI_EVENT_SEND VdevId/EventId = 0x1d00c VdevMode = 0 EventId = 0x0
R1: FWMSG: [7816722] WMI -  WMI_EVENT_SEND VdevId/EventId = 0x1d011 VdevMode = 0 EventId = 0x0
R0: FWMSG: [5490667] WMI_EVENT_SEND VdevId/EventId = 0x1d011 VdevMode = 0 EventId = 0x0
R1: FWMSG: [7817747] WMI -  WMI_EVENT_SEND VdevId/EventId = 0x1d011 VdevMode = 0 EventId = 0x0
R0: FWMSG: [5491692] WMI_EVENT_SEND VdevId/EventId = 0x1d011 VdevMode = 0 EventId = 0x0
```

NOTE: “R0: FWMSG” means the log from wlan.ko (PCIe module), “R1: FWMSG” means the log from wlan-sdio.ko (SDIO module).

Driver and firmware debug log level

cnss_diag always enables all levels of WLAN driver logging. Firmware debug log level can be statically configured by the settings of gFwDebugLogLevel, gFwDebugModuleLoglevel and gEnablefwlog parameters in qcom_cfg.ini or can be dynamically runtime configured by iwpriv dl_mod_loglevel command

- Statically configured in qcom_cfg.ini

- **gEnablefwlog**: This entry is used to enable or disable all firmware logs. It can have following values
  - 0 = disable FW logging
  - 1 = enable FW logging
- **gFwDebugLogLevel=<loglevel>:** This entry specifies the default log level for all firmware modules. Log level for each module can have value from 0 to 5. All firmware logs with log level value above the gFwDebugLogLevel will be logged.
  - 0 = DBGLOG_VERBOSE
  - 1 = DBGLOG_INFO
  - 2 = DBGLOG_INFO_LVL_1
  - 3 = DBGLOG_INFO_LVL_2
  - 4 = DBGLOG_WARN
  - 5 = DBGLOG_ERR

  If value of gFwDebugLogLevel is 0 then all logs levels of all firmware modules will be enabled except for the ones overridden by gFwDebugModuleLoglevel. By default, the value of this entry is 4 which means only warning and error logs are enabled.
- **gFwDebugModuleLoglevel=<moduleid,loglevel,……,moduleid,loglevel>**: This entry is used to override the default log level for a module. Value of module id is defined in CORE/SERVICES/COMMON/wlan_module_ids.h.

  For Example, gFwDebugModuleLoglevel=1,0,2,1,3,2,4,3,5,4 means for FW module ID 1 enable log level 0, for FW module ID 2 enable log level 1, for FW module ID 3 enable log level 2, for FW module ID 4 enable log level 3, for FW module ID 5 enable log level 4.

  By default, qcom_cfg.ini has following parameters configured to minimize performance impact and to reduce the need of reproduction.
  - gFwDebugLogLevel=4
  - gFwDebugModuleLoglevel=1,0,2,0,4,0,5,0,6,0,7,4,8,0,9,0,11,0,13,0,17,0,18,0,19,0,27,0,29,0,31,0,35,0,36,0,38,0
  - gEnablefwlog=1

■ Dynamically runtime configured by iwpriv dl_mod_loglevel command

iwpriv wlan0 dl_mod_loglevel <value of FW module ID * 10 + FW debug log level>: the command changes FW module's debug log level at runtime. E.g. iwpriv wlan0 dl_mod_loglevel 10 configures FW module ID 1 with log level 0 enabled.

Recommended firmware logs for different issue scenarios

■ Concurrency issues

- iwpriv wlan0 dl_mod_loglevel 10
- iwpriv wlan0 dl_mod_loglevel 20
- iwpriv wlan0 dl_mod_loglevel 50
- iwpriv wlan0 dl_mod_loglevel 60
- iwpriv wlan0 dl_mod_loglevel 70
- iwpriv wlan0 dl_mod_loglevel 80
- iwpriv wlan0 dl_mod_loglevel 90

- iwpriv wlan0 dl_mod_loglevel 110
- iwpriv wlan0 dl_mod_loglevel 130
- iwpriv wlan0 dl_mod_loglevel 190
- iwpriv wlan0 dl_mod_loglevel 270
- iwpriv wlan0 dl_loglevel 0

- Co-Existence issues
  - iwpriv wlan0 dl_mod_loglevel 40
- G-Scan issues
  - iwpriv wlan0 dl_mod_loglevel 10
  - iwpriv wlan0 dl_mod_loglevel 50
  - iwpriv wlan0 dl_mod_loglevel 70
  - iwpriv wlan0 dl_mod_loglevel 80
  - iwpriv wlan0 dl_mod_loglevel 90
  - iwpriv wlan0 dl_mod_loglevel 130
  - iwpriv wlan0 dl_mod_loglevel 140
  - iwpriv wlan0 dl_mod_loglevel 191
  - iwpriv wlan0 dl_mod_loglevel 500
- RTT issues
  - iwpriv wlan0 dl_mod_loglevel 220
- Autojoin issues
  - iwpriv wlan0 dl_mod_loglevel 10
  - iwpriv wlan0 dl_mod_loglevel 40
  - iwpriv wlan0 dl_mod_loglevel 50
  - iwpriv wlan0 dl_mod_loglevel 60
  - iwpriv wlan0 dl_mod_loglevel 70
  - iwpriv wlan0 dl_mod_loglevel 80
  - iwpriv wlan0 dl_mod_loglevel 90
  - iwpriv wlan0 dl_mod_loglevel 130
  - iwpriv wlan0 dl_mod_loglevel 170
  - iwpriv wlan0 dl_mod_loglevel 191
  - iwpriv wlan0 dl_mod_loglevel 270
  - iwpriv wlan0 dl_mod_loglevel 310
  - iwpriv wlan0 dl_mod_loglevel 360
  - iwpriv wlan0 dl_mod_loglevel 520
  - iwpriv wlan0 dl_mod_loglevel 530
  - iwpriv wlan0 dl_mod_loglevel 500

8.4.1 Module ID definitions

Check CORE/SERVICES/COMMON/wlan_module_ids.h for latest WLAN module definitions.

Table 8-3 WLAN module number

Typed name	Enum	Typed name	Enum
WLAN_MODULE_ID_MIN	0	WLAN_MODULE_HB	39
WLAN_MODULE_INF	0	WLAN_MODULE_TXBF	40
WLAN_MODULE_WMI	1	WLAN_MODULE_BATCH_SCAN	41
WLAN_MODULE_STA_PWRSAVE	2	WLAN_MODULE_THERMAL_MGR	42
WLAN_MODULE_WHAL	3	WLAN_MODULE_PHYERR_DFS	43
WLAN_MODULE_COEX	4	WLAN_MODULE_RMC	44
WLAN_MODULE_ROAM	5	WLAN_MODULE_STATS	45
WLAN_MODULE_RESMGR_CHAN_ MANAGER	6	WLAN_MODULE_NAN	46
WLAN_MODULE_RESMGR	7	WLAN_MODULE_IBSS_PWRSAVE	47
WLAN_MODULE_VDEV_MGR	8	WLAN_MODULE_HIF_UART	48
WLAN_MODULE_SCAN	9	WLAN_MODULE_LPI	49
WLAN_MODULE_RATECTRL	10	WLAN_MODULE_EXTSCAN	50
WLAN_MODULE_AP_PWRSAVE	11	WLAN_MODULE_UNIT_TEST	51
WLAN_MODULE_BLOCKACK	12	WLAN_MODULE_MLME	52
WLAN_MODULE_MGMT_TXRX	13	WLAN_MODULE_SUPPL	53
WLAN_MODULE_DATA_TXRX	14	WLAN_MODULE_ERE	54
WLAN_MODULE_HTT	15	WLAN_MODULE_OCB	55
WLAN_MODULE_HOST	16	WLAN_MODULE_RSSI_MONITOR	56
WLAN_MODULE_BEACON	17	WLAN_MODULE_WPM	57
WLAN_MODULE_OFFLOAD	18	WLAN_MODULE_CSS	58
WLAN_MODULE_WAL	19	WLAN_MODULE_PPS	59
WAL_MODULE_DE	20	WLAN_MODULE_SCAN_CH_PREDICT	60
WLAN_MODULE_PCIELP	21	WLAN_MODULE_MAWC	61
WLAN_MODULE_RTT	22	WLAN_MODULE_CMC_QMIC	62
WLAN_MODULE_RESOURCE	23	WLAN_MODULE_EGAP	63
WLAN_MODULE_DCS	24	WLAN_MODULE_NAN20	64
WLAN_MODULE_CACHEMGR	25	WLAN_MODULE_QBOOST	65
WLAN_MODULE_ANI	26	WLAN_MODULE_P2P_LISTEN_OFFLOAD	66
WLAN_MODULE_P2P	27	WLAN_MODULE_HALPHY	67
WLAN_MODULE_CSA	28	WAL_MODULE_ENQ	68
WLAN_MODULE_NLO	29	WLAN_MODULE_GNSS	69
WLAN_MODULE_CHATTER	30	WLAN_MODULE_WAL_MEM	70
WLAN_MODULE_WOW	31	WLAN_MODULE_SCHED_ALGO	71
WLAN_MODULE_WAL_VDEV	32	WLAN_MODULE_TX	72
WLAN_MODULE_WAL_PDEV	33	WLAN_MODULE_RX	73
WLAN_MODULE_TEST	34	WLAN_MODULE_WLM	74
WLAN_MODULE_STA_SMPS	35	WLAN_MODULE_RU_ALLOCATOR	75

Typed name	Enum	Typed name	Enum
WLAN_MODULE_SWBMISS	36	WLAN_MODULE_11K_OFFLOAD	76
WLAN_MODULE_WMMAC	37	WLAN_MODULE_ID_MAX	77
WLAN_MODULE_TDLS	38	WLAN_MODULE_ID_INVALID	78

8.5 Athdiag

The ATHDIAG user space utility provides a convenient and powerful mechanism to access target address space along with operations for read/write and save the contents of specific memory access requested to a file.

- Command options

```
usage:
athdiag commands and options:
--get --address=<target word address>
--set --address=<target word address> --[value|param]=<value>
                                      --or=<OR-ing value>
                                      --and=<AND-ing value>
--read --address=<target address> --length=<bytes> --file=<filename>
--write --address=<target address> --file=<filename>
                                   --[value|param]=<value>
--otp --read --address=<otp offset> --length=<bytes> --file=<filename>
--otp --write --address=<otp offset> --file=<filename>
--dump --target=<target name> [--hex] [--path=<pathname>]
--quiet
--device=<device name> (if not default)
The options can also be given in the abbreviated form --option=x or -o x.
The options can be given in any order.
```

- Write command can be used to change global variables in the firmware
  - Locate the file containing the symbol of interest
  - Use nm to extract symbol address
  - Use athdiag --read or --write commands to read/write the global variable
- Use device to differentiate the dual Wi-Fi module:
  - Device= /proc/cld for wlan.ko (PCIe module)
  - Device=/proc/cldqca6574 for wlan-sdio.ko (SDIO module)

Example:

```
# athdiag --read --address=0xa0000 --length=0x18000
--file=/firmware/rome_axi-1
# athdiag --read --address=0xa0000 --length=0x18000
--file=/firmware/rome_axi-3  --device=/proc/cldqca6574
```

NOTE: when error log shows “refusing to read mmio out of bounds at xxx” by using athdiag, it is the limitation of WLAN driver for PCIe interface. The default device is /proc/cld.

8.6 Pktlogconf

Pktlogconf is the utility to enable/disable pktlog collection and configure circular buffer size. It is used to debug issues in MAC layer by collecting WLAN frames transmission and receiving status. To support pktlogconf, gEnablePacketLog must be configured to 1 in qcom_cfg.ini.

- Command options

```
usage: pktlogconf [-a adapter] [-e[event-list]] [-d] [-s log-size] [-t -k -l]
                  [-b -p -i]
    -h    show this usage
    -a    configures packet logging for specific 'adapter';
          configures system-wide logging if this option is
          not specified
    -d    disable packet logging
    -e    enable logging events listed in the 'event-list'
          event-list is an optional comma separated list of one or more
          of the following: rx tx rcf rcu ani  (eg., pktlogconf -erx,rcu,tx)
    -s    change the size of log-buffer to "log-size" bytes
    -t    enable logging of TCP headers
    -k    enable triggered stop by a threshold number of TCP SACK packets
    -l    change the number of packets to log after triggered stop
    -b    enable triggered stop by a throuput threshold
    -p    enable triggered stop by a PER threshold
    -i    change the time period of counting throughput/PER
```

- Step to collect pktlog for first wlan module

```
# pktlogconf -a cld -s 10000000 -e  //set buffer size to 10MB, default
1MB
# <start the issue reproduction>
# pktlogconf -d cld    // stop pktlog collection
# touch ~/Data.dat
# cat /proc/ath_pktlog/cld > ~/DataLog1.dat
```

- Step to collect pktlog for second wlan module if it exists

```
# pktlogconf -a cld -m qca6574 -s 10000000 -e  //set buffer size to
10MB, default 1MB
# <start the issue reproduction>
# pktlogconf -d cld -m qca6574   // stop pktlog collection
# touch ~/Data.dat
# cat /proc/ath_pktlogqca6574/cldqca6574 > ~/DataLog2.dat
```

DataLog.dat contains binary data and needs script to do post-processing. Due to legal issue, the script cannot be released externally. Attach the pktlog file into QCOM Salesforce system for analysis.

NOTE: When gEnablePacketLog=1, it may degrade the KPI of SDIO module. Hence it will be configured to 0 in qcom_cfg.ini for SDIO module.

8.7 SSR_Demo

The SSR_Demo user space utility provides the function to restart the WLAN module. When SSR_Demo receives a crash event from WLAN driver through netlink routine, triggers the restart of the WLAN module. By default, it runs as a daemon.

- Command options

"module_numbers" depends on the order in which the modules are started, starting at 1.

```
usage: SSR_Demo [options]
   -n, --nodaemon  do not run as a daemon
   -d              show more debug messages (-dd for even more)
   -f <path/file>  Log output to file
   -s              Log output to syslog
   -m  --mod_name <module_name,module_numbers> specify the module name,eg. -m wlan-cnss,2
                  0 < module_name_length < 240, 0 < module_numbers <= 2
       --help      display this help and exit
```

- Usage

  ```
  #  insmod *.ko
  ```

```
root@imx6qsabresd:~# lsmod
Module                  Size  Used by
wlan                 4816850  0
root@imx6qsabresd:~# SSR_Demo -m wlan,1
root@imx6qsabresd:~# ps | grep SSR_Demo
  620 root       1700 S    SSR_Demo -m wlan,1
  636 root       2740 S    grep SSR_Demo
```

Then SSR_Demo runs as a daemon.

9 Porting WLAN to 3rd platform with Linux 4.1/4.9

9.1 Purpose

This chapter describes the procedure to port the QCA65X4A/AU WLAN to the Freescale iMX6 board with a Yocto BSP. Figure 9-1 shows the necessary components, their builds, and the release model.

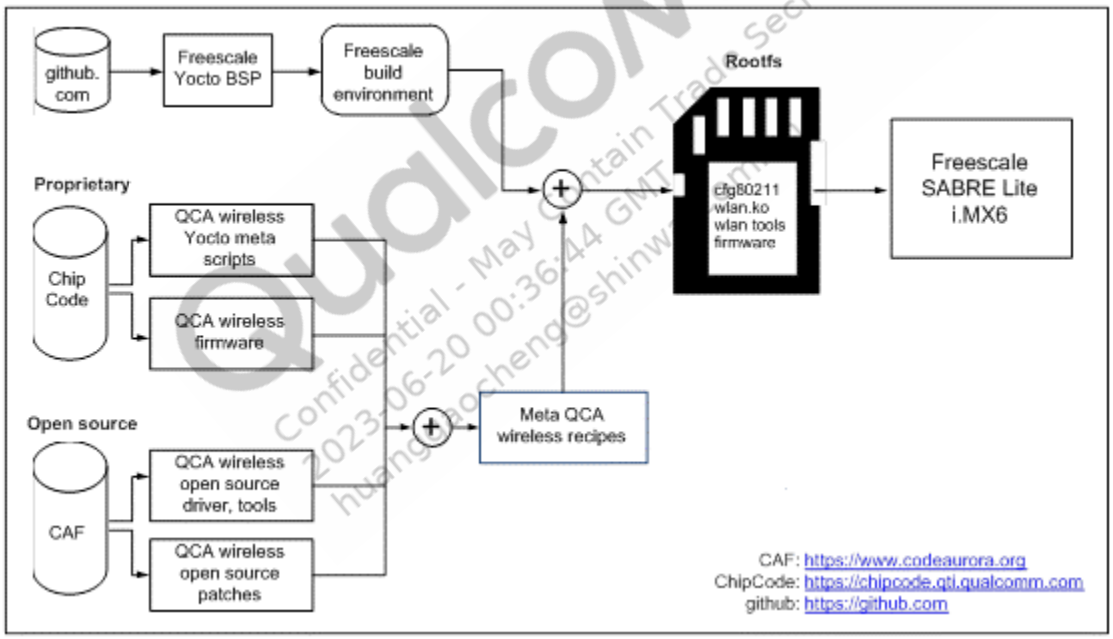


Figure 9-1 Required components

Table 9-1 lists the major components that are required to be built to support QCA65X4A/AU WLAN on Freescale iMX6 Yocto platform.

Table 9-1 Major components

Category	Action
Freescale Yocto BSP	Download from github.com
QCA65X4A/AU firmware binary	Download from QTI ChipCode portal
Ath6kl-utils(Myftm)	Download from QTI ChipCode portal
Qcmbr	Download from QTI ChipCode portal
Qcacld-utils(cnss_diag/pktlogconf/Athdiag)	Download from QTI ChipCode portal
SSR_Demo	Download from QTI ChipCode portal

Category	Action
Wlan-rtt	Download from QTI ChipCode portal
Open source meta layer for QTI connectivity	Download from CLO open source server
WLAN Host driver source code	Download from CLO open source server
Wpa_supplicant	Download from CLO open source server
Dsrc-tool	Download from CLO open source server
Sigma-dut	Download from CLO open source server
Mdm-init	Download from CLO open source server

9.2 Set up build environment

1. Install Ubuntu 12.04 LTS on an x86 Linux machine.
2. Set up the x86 build environment. The essential and graphical support packages needed to support Ubuntu distribution are shown in the following command.

   ```
   $ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath libsdl1.2-dev xterm libcurl4-openssl-dev lzop
   ```

3. Follow the steps to install repo utility.

   ```
   $ mkdir ~/bin
   $ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
   $ chmod a+x ~/bin/repo
   $ PATH=${PATH}:~/bin
   ```

9.3 Download QCA65X4A/AU recipes and Freescale BSP with Linux 4.9.11

1. Create a clean workspace:

   ```
   $ mkdir fsl-community-bsp
   $ cd fsl-community-bsp
   ```

2. Download Freescale BSP:

   ```
   $ repo init -u https://github.com/nxp-imx/imx-manifest.git -b imx-linux-morty -m imx-4.9.11-1.0.0_ga.xml
   $ repo sync
   ```

3. Download QCA open source code from CLO.

   Create a *tmp* folder outside of the *fsl-community-bsp* folder, download QCA open source code to the *tmp* folder, and then copy them to *fsl-community-bsp/sources/* folder:

   ```
   $ mkdir tmp
   $ cd tmp
   $repo init --no-clone-bundle -u https://git.codelinaro.org/clo/le/le/manifest.git -b release -m CHSS.LNX_FSL.2.1-07310-QCA6574AUARMSDIOHZ.xml --repo-url=https://git.codelinaro.org/clo/tools/repo.git --repo-branch=aosp-new/repo-1
   $repo sync -c --no-tags -j16
   ```

```
$ cp -rf ./sources/meta-qti-connectivity  ../fsl-community-bsp/sources
$ cp -rf ./sources/wlan-opensource  ../fsl-community-bsp/sources
```

4. Download WLAN firmware and QCA proprietary tool from ChipCode.

   Download the package named *qca6574au-le-2-2-1_qca_oem.git* from https://chipcode.qti.Qualcomm.com/Qualcomm/qca6574au-le-2-2-1_qca_oem/tree/r10019.1, put it under folder *fsl-community-bsp*, and then collect the firmware and recipes from the downloaded package.

   ```
   $ cd fsl-community-bsp
   $ cp -rf qca6574au-le-2-2-1_qca_oem.git/host/apps_proc/sources/wlan-proprietary/  ./sources/
   $ cp -rf qca6574au-le-2-2-1_qca_oem.git/host/apps_proc/sources/meta-qti-connectivity-prop/  ./sources/
   ```

After completing the above steps, the *fsl-community-bsp/sources* folder shows as following:

```
fsl-community-bsp/sources
                  ├── meta-qti-connectivity
                  └── wlan-opensource
                          └── dsrc-tool
                          └── mdm-init
                          └── qcacld-2.0
                          └── sigma-dut
                          └── wpa_supplicant_8
                  └── meta-qit-connectivity-prop
                  └── wlan-proprietary
                  └── ……
```

NOTE: The above sample lists only QCA WLAN meta layer, other Freescale meta layers are not listed.

WLAN firmware binary files are located at *fsl-community-bsp/qca6574au-le-2-2-1_qca_oem.git/meta_build/firmware/*, the *firmware* folder shows as follows:

```
<chipcode_root>/meta_build
                  └── firmware
                        ├── pcie
                              └──Data.msc
                              └──otp.bin
                              └──utf.bin
                              └──athwlan.bin
                              └──fakeboar.bin
                        ├── sdio
                              └──Data.msc
                              └──otp30.bin
                              └──utf30.bin
                              └──qwlan30.bin
                              └──bdwlan30.bin
                        ├── btfw32.tlv
                        ├── btnv32.bin
```

NOTE: `<chipcode_root>` means fsl-community-bsp/qca6574au-le-2-2-1_qca_oem.git.

9.4 Build iMX Core image for Linux 4.9.11

1. Make the scripts executable:
   ```
   $ cd fsl-community-bsp/source/meta-qti-connectivity-prop/scripts
   $ chmod a+x prepare_sdk_prop.sh
   $ cd fsl-community-bsp/source/meta-qti-connectivity/scripts
   $ chmod a+x prepare_sdk_4.9.11_os.sh
   ```
2. Set up the build environment (for Freescale iMX6 SABRE board, as shown in Figure 1-1):
   ```
   $ EULA=0 MACHINE=imx6qsabresd source fsl-setup-release.sh -b build -e
   x11
   ```

NOTE: If the following error occurs, use command `umask 022` to fix it:

```
ERROR: OE-core's config sanity checker detected a potential
misconfiguration. Either fix the cause of this error or at your own risk
disable the checker (see sanity.conf). Following is the list of
potential problems / advisories: Please use a umask which allows a+rx
and u+rwx.
```

3. Run prepare scripts:
   ```
   $ ../sources/meta-qti-connectivity-prop/scripts/prepare_sdk_prop.sh
   $ ../sources/meta-qti-connectivity/scripts/prepare_sdk_4.9.11_os.sh
   ```
4. Build the Yocto core image to package the WLAN into the core-image:
   ```
   $ bitbake core-image-minimal
   ```

When the build is complete, the core image is created at *tmp/deploy/images/imx6qsabresd/* directory with filename core-image-minimal-imx6qsabresd.sdcard.

9.5 Flash image

Follow the steps below to create a boot SD card:

1. Insert an SD card into the card reader.
2. Covert and copy the core image file into the SD card:

   Switch the folder path to *./build/tmp/deploy/images* and find the image with name {MACHINE NAME}, for example MACHINE=imx6qsabresd. Then follow the below instructions to copy it onto SD card for booting.
   ```
   $ sudo dd if= core-image-minimal-imx6qsabresd.sdcard
   of=/dev/{SD_DEV_NAME}
   ```

NOTE: `{SD_DEV_NAME}` is based on the name which system recognizes. You can use the **`mount`** command to check it. In general, it is "`/dev/mmcblk0" or "/dev/sdb".`

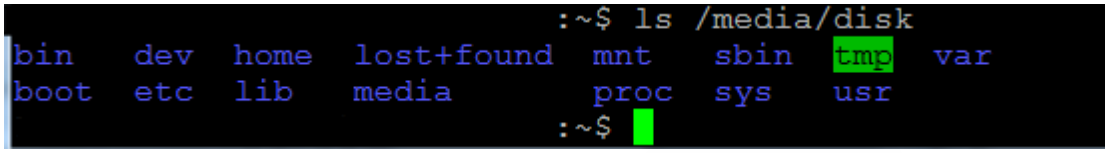
   After the file transfer is complete, sync to write back into SD card before unmounting it.
   ```
   $ sudo sync
   ```
3. Check the image on the SD card:

   Use the **`mount`** command to identify the rootfs partition of Yocto BSP core image on the SD card

```
$ mount
…
/dev/sdb2 on /media/disk type ext3 (rw,nosuid,nodev,uhelper=udisks)
…
```

It indicates that the Yocto rootfs on SD card is mounted on the development machine in "/media/disk". Use the **`ls`** command to check its root file system.

```
$ ls /media/disk
```



4. Copy QCA65X4A/AU-1 firmware files to image on the SD card. The firmware files are located at <chipcode*_root>/meta_build/firmware/pcie*.

```
$ cd <chipcode_root>/meta_build/firmware/pcie
$ sudo cp athwlan.bin otp.bin fakeboar.bin utf.bin
/media/disk/lib/firmware/
```

5. Copy QCA65X4A/AU-1 firmware log parse data to image on the SD card. The firmware log parse data is located at <*chipcode_root>/meta_build/firmware/pcie*.

```
$ cd <chipcode_root>/meta_build/firmware/pcie
$ sudo mkdir -p /media/disk/firmware/image/
$ sudo cp Data.msc /media/disk/firmware/image/
```

6. Copy QCA65X4A/AU-3 firmware files to image on the SD card. The firmware files are located at <*chipcode_root>/meta_build/firmware/sdio*.

```
$ cd <chipcode_root>/meta_build/firmware/sdio
$ sudo mkdir -p /meida/disk/lib/firmware/qca6574
$ sudo cp qwlan30.bin otp30.bin bdwlan30.bin utf30.bin
/media/disk/lib/firmware/qca6574
```

7. Copy QCA65X4A/AU-3 firmware log parse data to image on the SD card. The firmware log parse data is located at <*chipcode_root>/meta_build/firmware/sdio*.

```
$ cd <chipcode_root>/meta_build/firmware/sdio
$ sudo mkdir -p /media/disk/lib/firmware/qca9377
$ sudo cp Data.msc /media/disk/lib/firmware/qca9377
```

10 Porting WLAN to 3rd platform with Linux 4.14

10.1 Purpose

This chapter describes the procedure to port the QCA65X4A/AU WLAN to the Freescale iMX6 board with a Yocto BSP. Figure 9-1 shows the necessary components, their builds, and the release model.

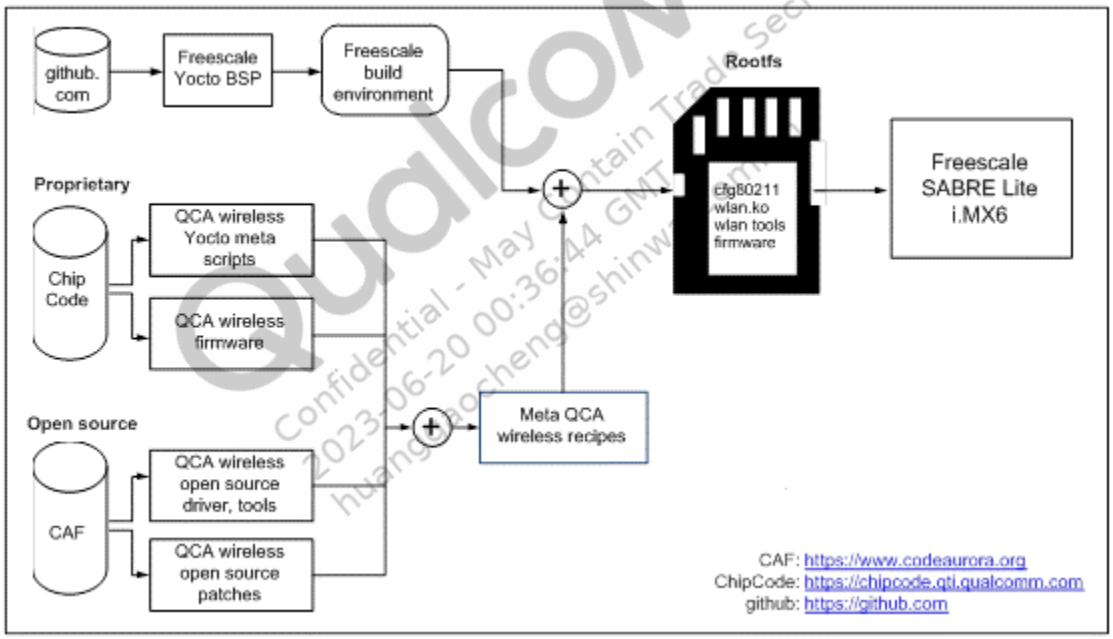


Figure 9-10-1 Required components

Table 9-1 lists the major components that are required to be built to support QCA65X4A/AU WLAN on Freescale iMX6 Yocto platform.

Table 9-10-1 Major components

Category	Action
Freescale Yocto BSP	Download from github.com
QCA65X4A/AU firmware binary	Download from QTI ChipCode portal
Ath6kl-utils(Myftm)	Download from QTI ChipCode portal
Qcmbr	Download from QTI ChipCode portal
Qcacld-utils(cnss_diag/pktlogconf/Athdiag)	Download from QTI ChipCode portal
SSR_Demo	Download from QTI ChipCode portal

Category	Action
Wlan-rtt	Download from QTI ChipCode portal
Open source meta layer for QTI connectivity	Download from CLO open source server
WLAN Host driver source code	Download from CLO open source server
Wpa_supplicant	Download from CLO open source server
Dsrc-tool	Download from CLO open source server
Sigma-dut	Download from CLO open source server
Mdm-init	Download from CLO open source server

10.2 Set up build environment.

1. Install Ubuntu 16 LTS on an x86 Linux machine.
2. Set up the x86 build environment. The essential and graphical support packages needed to support Ubuntu distribution are shown in the following command.

   ```
   $ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath libsdl1.2-dev xterm libcurl4-openssl-dev lzop
   ```

3. Follow the steps to install repo utility.

   ```
   $ mkdir ~/bin
   $ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
   $ chmod a+x ~/bin/repo
   $ PATH=${PATH}:~/bin
   ```

10.3 Download QCA65X4A/AU recipes and Freescale BSP with Linux 4.14.78

1. Create a clean workspace:

   ```
   $ mkdir fsl-community-bsp
   $ cd fsl-community-bsp
   ```

2. Download Freescale BSP and copy the setup files:

   ```
   $ repo init -u https://github.com/nxp-imx/imx-manifest.git -b imx-linux-sumo -m imx-4.14.78-1.0.0_ga.xml
   $ repo sync
   $ cp ./sources/base/setup-environment ./
   $ cp ./sources/meta-fsl-bsp-release/imx/tools/fsl-setup-release.sh  ./
   ```

3. Download QCA open-source code from CLO.

   Create a *tmp* folder outside of the *fsl-community-bsp* folder, download QCA open-source code to the *tmp* folder, and then copy them to *fsl-community-bsp/sources/* folder:

   ```
   $ mkdir tmp
   $ cd tmp
   $repo init --no-clone-bundle -u
   https://git.codelinaro.org/clo/le/le/manifest.git -b release -m
   CHSS.LNX_FSLS.4.14-05410-QCA6574AUARMSDIOHZ.xml --repo-
   ```

```
url=https://git.codelinaro.org/clo/tools/repo.git --repo-branch=aosp-
new/repo-1
$repo sync -c --no-tags -j16
$ cp -rf ./sources/meta-qti-connectivity  ../fsl-community-bsp/sources
$ cp -rf ./sources/wlan-opensource  ../fsl-community-bsp/sources
```

4. Download WLAN firmware and QCA proprietary tool from ChipCode.

   Download the package named *qca6574au-le-2-2-1_qca_oem.git* from https://chipcode.qti.Qualcomm.com/Qualcomm/qca6574au-le-2-2-1_qca_oem/tree/r10019.1, put it under folder *fsl-community-bsp*, and then collect the firmware and recipes from the downloaded package.

```
$ cd fsl-community-bsp
$ cp -rf qca6574au-le-2-2-
1_qca_oem.git/host_4.14/apps_proc/sources/wlan-proprietary/  ./sources/
$ cp -rf qca6574au-le-2-2-
1_qca_oem.git/host_4.14/apps_proc/sources/meta-qti-connectivity-prop/
./sources/
```

After completing the above steps, the *fsl-community-bsp/sources* folder shows as following:

```
fsl-community-bsp/sources
                  ├── meta-qti-connectivity
                  └── wlan-opensource
                        └── dsrc-tool
                        └── mdm-init
                        └── qcacld-2.0
                        ├── qcacld-3.0
                        └── qca-wifi-host-cmn
                        └── fw-api
                        └── sigma-dut
                        └── wpa_supplicant_8
                  └── meta-qit-connectivity-prop
                  └── wlan-proprietary
                  └── ……
```

NOTE: The above sample lists only QCA WLAN meta layer, other Freescale meta layers are not listed.

WLAN firmware binary files are located at *fsl-community-bsp/qca6574au-le-2-2-1_qca_oem.git/meta_build/firmware/*, the *firmware* folder shows as follows:

```
<chipcode_root>/meta_build
                  └── firmware
                        ├── pcie
                              └──Data.msc
                              └──otp.bin
                              └──utf.bin
                              └──athwlan.bin
                              └──fakeboar.bin
                        ├── sdio
                              └──Data.msc
                              └──otp30.bin
                              └──utf30.bin
```

```
        └──qwlan30.bin
        └──bdwlan30.bin
├── btfw32.tlv
├── btnv32.bin
```

NOTE: `<chipcode_root>` means fsl-community-bsp/qca6574au-le-2-2-1_qca_oem.git.

10.4 Build iMX Core image for Linux 4.14.78

1. Make the scripts executable:

```
$ cd fsl-community-bsp/source/meta-qti-connectivity-prop/scripts
$ chmod a+x prepare_sdk_prop.sh
$ cd fsl-community-bsp/source/meta-qti-connectivity/scripts
$ chmod a+x prepare_sdk_4.14.78_os.sh
```

2. Enable PCIE support:

   Add three kernel configuration to fsl-community-bsp/source/meta-qti-connectivity/recipes-kernel/linux-kernel/linux-imx_%.bbappend under 4.14.78 case as following:

```
--- a/recipes-kernel/linux-kernel/linux-imx_%.bbappend
+++ b/recipes-kernel/linux-kernel/linux-imx_%.bbappend
@@ -49,6 +49,9 @@ CONFIG_STACKTRACE=y
 CONFIG_BRIDGE=y
 CONFIG_TMPFS=y
 CONFIG_CNSS_LOGGER=y
+CONFIG_PCI=y
+CONFIG_PCI_IMX6=y
+CONFIG_PCI_MSI=y
 KERNEL_EXTRACONFIGS
                 };;
```

3. Set up the build environment:

```
$ EULA=1 DISTRO=fsl-imx-x11 MACHINE=imx6qsabresd source fsl-setup-release.sh -b build
```

NOTE: If the following error occurs, use command `umask 022` to fix it:

```
ERROR: OE-core's config sanity checker detected a potential
misconfiguration. Either fix the cause of this error or at your own risk
disable the checker (see sanity.conf). Following is the list of
potential problems / advisories: Please use a umask which allows a+rx
and u+rwx.
```

4. Run prepare scripts:

```
$ ../sources/meta-qti-connectivity-prop/scripts/prepare_sdk_prop.sh
$ ../sources/meta-qti-connectivity/scripts/prepare_sdk_4.14.78_os.sh
```

5. Build the Yocto core image to package the WLAN into the core-image:

```
$ bitbake core-image-minimal
```

When the build is complete, the core image is created at *tmp/deploy/images/imx6qsabresd/* directory with filename core-image-minimal-imx6qsabresd.sdcard.bz2.

10.5 Flash image

Follow the steps below to create a boot SD card:

1. Insert an SD card into the card reader.
2. Unzip and copy the core image file into the SD card:

   Switch the folder path to *./build/tmp/deploy/images* and find the image with name {MACHINE NAME}, for example MACHINE=imx6qsabresd. Then follow the below instructions to copy it onto SD card for booting.

   ```
   $ bunzip2 core-image-minimal-imx6qsabresd.sdcard.bz2
   $ sudo dd if= core-image-minimal-imx6qsabresd.sdcard
   of=/dev/{SD_DEV_NAME}
   ```

   NOTE: `{SD_DEV_NAME}` is based on the name which system recognizes. You can use the **`mount`** command to check it. In general, it is "`/dev/mmcblk0`" or "`/dev/sdb`".

   After the file transfer is complete, sync to write back into SD card before unmounting it.

   ```
   $ sudo sync
   ```

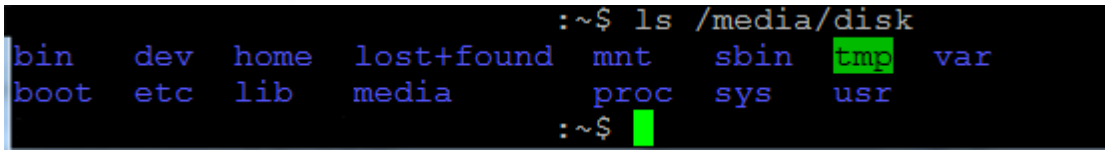
3. Check the image on the SD card:

   Use the **`mount`** command to identify the rootfs partition of Yocto BSP core image on the SD card

   ```
   $ mount
   …
   /dev/sdb2 on /media/disk type ext3 (rw,nosuid,nodev,uhelper=udisks)
   …
   ```

   It indicates that the Yocto rootfs on SD card is mounted on the development machine in "/media/disk". Use the **`ls`** command to check its root file system.

   ```
   $ ls /media/disk
   ```

10.5.1 Update WLAN firmware and host driver configuration file

1. Copy QCA65X4A/AU-1 firmware files to image on the SD card. The firmware files are located at <chipcode_*root*>*/meta_build/firmware/sdio*.

   ```
   $ cd <chipcode_root>/meta_build/firmware/sdio
   $ sudo mkdir -p /media/disk/lib/firmware/qca6574
   $ sudo cp Data.msc qwlan30.bin otp30.bin bdwlan30.bin utf30.bin
   /media/disk/lib/firmware/qca6574
   ```

2. Rename host driver configuration files in "/media/disk/lib/firmware/wlan/"

   ```
   $ sudo mkdir -p /media/disk/lib/firmware/wlan/qca6574
   ```

```
$ sudo cd /media/disk/lib/firmware/wlan
$ sudo cp QCA6574AU.LE.2.2.1_Rome_SDIO_qcacld-3.0.ini
qca6574/qcom_cfg.ini
```

- wlan-sdio.ko located on /lib/modules/${Kernel Version}/extra/
- WLAN firmware, located on rootfs "/lib/firmware/qca6574" directory
  - **qwlan30.bin**: WLAN target firmware
  - **otp30.bin** : OTP memory manipulate firmware
  - **utf30.bin** : WLAN test mode firmware
  - **bdwlan30.bin** : WLAN board data
- WLAN FW log parse data, located on rootfs "/lib/firmware/qca6574" directory
  - **Data.msc:** used to parse FW log
- host driver configuration file, located on rootfs "/lib/firmware/wlan/qca6574" directory
  - **qcom_cfg.ini:** driver configuration file.

10.5.2 Bring up WLAN driver

1. Insmod wlan host modules

```
# cd /lib/modules/${Kernel Version}/extra/
# insmod wlan-sdio.ko
```

2. Use *ifconfig* command to bring up WLAN interface on Linux

```
# ifconfig -a
# ifconfig wlan0 up
# ifconfig
wlan0     Link encap:Ethernet  HWaddr 00:03:7F:12:55:29
          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:3000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

11 Porting WLAN to 3rd platform with Linux 5.4

11.1 Purpose

This chapter describes the procedure to port the QCA65X4A/AU-1 WLAN to the Freescale iMX8 board with a Yocto BSP. Figure 9-1 shows the necessary components, their builds, and the release model.

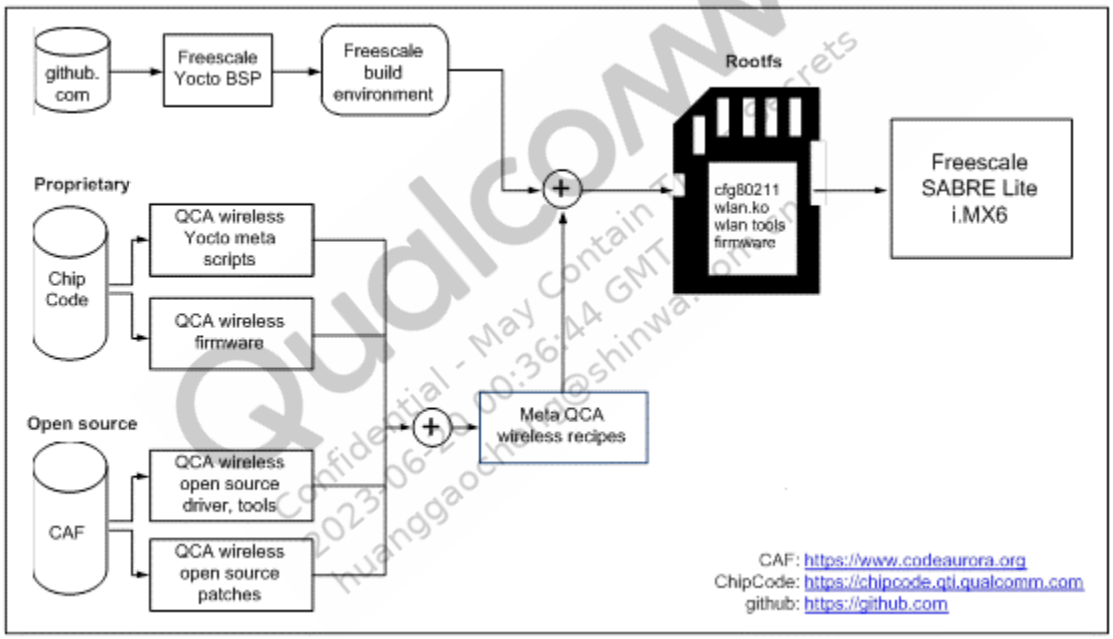


Figure 10-11-1 Required components

Table 9-1 lists the major components that are required to be built to support QCA65X4A/AU-1 WLAN on Freescale iMX6 Yocto platform.

Table 9-11-1 Major components

Category	Action
Freescale Yocto BSP	Download from github.com
QCA65X4A/AU-1 firmware binary	Download from QTI ChipCode portal
Ath6kl-utils(Myftm)	Download from QTI ChipCode portal
Qcmbr	Download from QTI ChipCode portal
Qcacld-utils(cnss_diag/pktlogconf/Athdiag)	Download from QTI ChipCode portal
SSR_Demo	Download from QTI ChipCode portal
Wlan-rtt	Download from QTI ChipCode portal
Open source meta layer for QTI connectivity	Download from CLO open source server

Category	Action
WLAN Host driver source code	Download from CLO open source server
Wpa_supplicant	Download from CLO open source server
Dsrc-tool	Download from CLO open source server
Sigma-dut	Download from CLO open source server
Mdm-init	Download from CLO open source server

11.2 Set up build environment

1. Install Ubuntu 16 LTS on an x86 Linux machine.
2. Set up the x86 build environment. The essential and graphical support packages needed to support Ubuntu distribution are shown in the following command.

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath libsdl1.2-dev xterm libcurl4-openssl-dev lzop
```

3. Follow the steps to install repo utility.

```
$ mkdir ~/bin
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ PATH=${PATH}:~/bin
```

11.3 Download QCA65X4A/AU-1 recipes and Freescale BSP with Linux 5.4

1. Create a clean workspace:

```
$ mkdir fsl-community-bsp
$ cd fsl-community-bsp
```

2. Download Freescale BSP and copy the setup files:

```
$ repo init -u https://github.com/nxp-imx/imx-manifest.git -b imx-linux-zeus -m imx-5.4.70-2.3.2.xml
$ repo sync -j32
```

3. Download QCA open-source code from CLO.

   Create a *tmp* folder outside of the *fsl-community-bsp* folder, download QCA open-source code to the *tmp* folder, and then copy them to *fsl-community-bsp/sources/* folder:

```
$ mkdir tmp
$ cd tmp
$repo init --no-clone-bundle -u https://git.codelinaro.org/clo/le/le/manifest.git -b release -m CHSS.LNX_FSL.5.0.r1-02900-QCA6574AUARMSDIOHZ.xml --repo-url=https://git.codelinaro.org/clo/tools/repo.git --repo-branch=aosp-new/repo-1
$repo sync -c --no-tags -j16
$ cp -rf ./sources/meta-qti-connectivity  ../fsl-community-bsp/sources
$ cp -rf ./sources/wlan-opensource  ../fsl-community-bsp/sources
```

4. Download WLAN firmware and QCA proprietary tool from ChipCode.

   Download the package named *qca6574au-le-2-2-1_qca_oem* from https://chipcode.qti.Qualcomm.com/Qualcomm/qca6574au-le-2-2-1_qca_oem/tree/r10019.1, put it under folder *fsl-community-bsp*, and then collect the firmware and recipes from the downloaded package.

   ```
   $ cd fsl-community-bsp
   $ cp -rf qca6574au-le-2-2-1_qca_oem/host_5.04/apps_proc/sources/wlan-proprietary/ ./sources/
   $ cp -rf qca6574au-le-2-2-1_qca_oem/host_5.04/apps_proc/sources/meta-qti-connectivity-prop/ ./sources/
   ```

After completing the above steps, the *fsl-community-bsp/sources* folder shows as following:

```
fsl-community-bsp/sources
                  ├── meta-qti-connectivity
                  └── wlan-opensource
                         └── dsrc-tool
                         └── mdm-init
                         └── qcacld-2.0
                         └── qcacld-3.0
                         └── qca-wifi-host-cmn
                         └── fw-api
                         └── sigma-dut
                         └── wpa_supplicant_8
                  └── meta-qit-connectivity-prop
                  └── wlan-proprietary
                  └── ……
```

NOTE: The above sample lists only QCA WLAN meta layer, other Freescale meta layers are not listed.

WLAN firmware binary files are located at *fsl-community-bsp/qca6574au-le-2-2-1_qca_oem.git/meta_build/firmware/*, the *firmware* folder shows as follows:

```
<chipcode_root>/meta_build
                  └── firmware
                        ├── pcie
                              └──Data.msc
                              └──otp.bin
                              └──utf.bin
                              └──athwlan.bin
                              └──fakeboar.bin
                        ├── sdio
                              └──Data.msc
                              └──otp30.bin
                              └──utf30.bin
                              └──qwlan30.bin
                              └──bdwlan30.bin
                        ├── btfw32.tlv
                        ├── btnv32.bin
```

NOTE: `<chipcode_root>` means fsl-community-bsp/qca6574au-le-2-2-1_qca_oem.git.

11.4 Build iMX Core image for Linux 5.4

1. Make the scripts executable:

```
$ cd fsl-community-bsp/source/meta-qti-connectivity-prop/scripts
$ chmod a+x prepare_sdk_prop.sh
$ cd fsl-community-bsp/source/meta-qti-connectivity/scripts
$ chmod a+x prepare_sdk_os.sh
```

2. Set up the build environment (for Freescale iMX6 EVK board):

```
$ EULA=1 MACHINE=imx6qsabresd DISTRO=fsl-imx-xwayland source ./imx-
setup-release.sh -b build
```

NOTE: If the following error occurs, use command `umask 022` to fix it:

```
ERROR: OE-core's config sanity checker detected a potential
misconfiguration. Either fix the cause of this error or at your own risk
disable the checker (see sanity.conf). Following is the list of
potential problems / advisories: Please use a umask which allows a+rx
and u+rwx.
```

3. Run prepare scripts:

```
$ ../sources/meta-qti-connectivity-prop/scripts/prepare_sdk_prop.sh
$ ../sources/meta-qti-connectivity/scripts/prepare_sdk_os.sh
```

4. Build the Yocto core image to package the WLAN into the core-image:

```
$ umask a+rx
$ bitbake core-image-minimal
```

When the build is complete, the core image is created at *tmp/deploy/images/imx6qsabresd/* directory with filename core-image-minimal-imx6qsabresd.wic.bz2.

11.5 Flash image

Follow the steps below to create a boot SD card:

1. Insert an SD card into the card reader.
2. Unzip and copy the core image file into the SD card:

   Switch the folder path to *./build/tmp/deploy/images* and find the image with name {MACHINE NAME}, for example MACHINE= imx6qsabresd. Then follow the below instructions to copy it onto SD card for booting.

```
$ bunzip2 -dk -f core-image-minimal-imx6qsabresd.wic.bz2
$ sudo dd if=core-image-minimal-imx6qsabresd.wic of=/dev/sd{SD_DEV_NAME}
bs=1M conv=fsync
```

NOTE: `{SD_DEV_NAME}` is based on the name which system recognizes. You can use the **`mount`** command to check it. In general, it is "`/dev/mmcblk0" or "/dev/sdb"`.

   After the file transfer is complete, sync to write back into SD card before unmounting it.

```
$ sudo sync
```

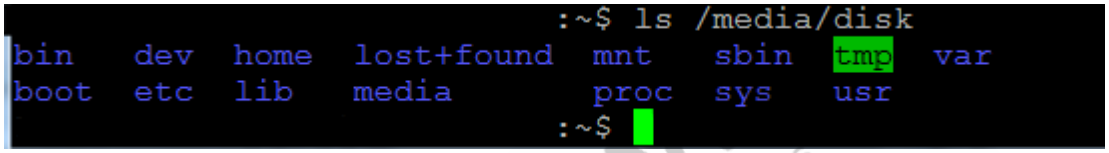
3. Check the image on the SD card:

   Use the **mount** command to identify the rootfs partition of Yocto BSP core image on the SD card

   ```
   $ mount
   …
   /dev/sdb2 on /media/disk type ext3 (rw,nosuid,nodev,uhelper=udisks)
   …
   ```

   It indicates that the Yocto rootfs on SD card is mounted on the development machine in "/media/disk". Use the **ls** command to check its root file system.

   ```
   $ ls /media/disk
   ```

11.5.1 Update WLAN firmware and host driver configuration file

4. Copy QCA65X4A/AU-1 firmware files to image on the SD card. The firmware files are located at <chipcode_*root*>*/meta_build/firmware/sdio*.

   ```
   $ cd <chipcode_root>/meta_build/firmware/sdio
   $ sudo mkdir -p /media/disk/lib/firmware/wlan-sdio
   $ sudo cp Data.msc qwlan30.bin otp30.bin bdwlan30.bin utf30.bin
   /media/disk/lib/firmware/wlan-sdio
   ```

5. Rename host driver configuration files in "/media/disk/lib/firmware/wlan/"

   ```
   $ cd /media/disk/lib/firmware/wlan/
   $ sudo mkdir -p /media/disk/lib/firmware/wlan/wlan-sdio
   $ sudo cp QCA6574AU.LE.2.2.1_Rome_SDIO_qcacld-3.0.ini wlan-
   sdio/qcom_cfg.ini
   ```

   - wlan-sdio.ko located on /lib/modules/${Kernel Version}/extra/
   - WLAN firmware, located on rootfs "/lib/firmware/wlan-sdio" directory
     - **qwlan30.bin**: WLAN target firmware
     - **otp30.bin** : OTP memory manipulate firmware
     - **utf30.bin** : WLAN test mode firmware
     - **bdwlan30.bin**: WLAN board data
   - WLAN FW log parse data, located on rootfs "/lib/firmware/wlan-sdio" directory
     - **Data.msc:** used to parse FW log
   - host driver configuration file, located on rootfs "/lib/firmware/wlan/wlan-sdio" directory
     - **qcom_cfg.ini:** driver configuration file.

11.5.2 Bring up WLAN driver

1. Insmod wlan host modules

   ```
   # cd /lib/modules/${Kernel Version}/extra/
   ```

```
# insmod wlan-sdio.ko
```

2. Use *ifconfig* command to bring up WLAN interface on Linux

```
# ifconfig -a
# ifconfig wlan0 up
# ifconfig
wlan0     Link encap:Ethernet  HWaddr 00:03:7F:12:55:29
          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:3000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

A Appendix

A.1 Kernel patch for Linux - iMX 4.9.11

Kernel patch locates at *fsl-commnuity-bsp/sources/meta-qti-connectivity/recipes-kernel/linux-kernel/files/*, the *linux-kernel* folder shows as follows:

```
Linux-kernel
├── files
│   ├──lk-4.9
│   │   ├── 0001-2221407-cfg80211-Use-new-wiphy-flag-
WIPHY_FLAG_DFS_OFFLOAD.patch
│   │   ├── 0002-2221408-mac80211-implement-HS2.0-gratuitous-ARP-
unsolicited-.patch
│   │   ├── 0003-2221409-cfg80211-export-regulatory_hint_user-API.patch
│   │   ├──  0004-2224213-net-cnss-Add-snapshot-of-cnss-driver.patch
│   │   ├──  0005-2227362-make-CNSS-work-up-for-SDIO-device-on-kernel-
4.9.patch
│   │   ├──  0006-2227831-Enable-CNSS-PCI-platform-module.patch
│   │   ├── 0007-2232783-Remove-PCI-host-controller-driver-
dependency.patch
│   │   ├──  0008-2247541-Enable-UART5-for-BT.patch
│   │   ├──  0009-2259006-Update-db.txt.patch
│   │   ├──  0010-2264510-cfg80211-fix-build-error-about-
cfg80211_roam_info.patch
│   │   ├──  0011-2268473-Fix-the-error-return-dismatch-with-qcacld2.0-
wlan-dr.patch
│   │   ├──  0012-2284816-cnss-add-pointer-null-check-before-use.patch
│   │   ├──  0013-2338459-cfg80211-Amendment-for-Use-new-wiphy-flag-
WIPHY_FLAG.patch
│   │   ├──  0014-2351541-cfg80211-Bypass-checkin-the-CHAN_RADAR-if-
DFS_OFFLOA.patch
│   │   ├──  0015-2407178-cfg80211-Fix-use-after-free-when-process-wdev-
events.patch
│   │   ├──  0016-2406371-CNSS-update-code-to-fix-blacklist-CR-and-compile-
err.patch
│   │   ├── 0017-2505736-CNSS-cnss-logger-can-be-built-even-no-CNSS-
module.patch
│   ├── linux-imx_%.bbappend
```

linux-imx_%.bbappend is used to configure the kernel and apply kernel patch.

A.2 Kernel patch for Linux - iMX 4.14.78

Kernel patch locates at *fsl-commnuity-bsp/sources/meta-qti-connectivity/recipes-kernel/linux-kernel/files/lk-4.14*, the *Linux-kernel* folder shows as follows:

```
lk-4.14
├── 0001-2748365-cfg80211-Updated-nl80211_commands-to-be-in-sync-with.patch
├── 0002-2748366-cfg80211-nl80211-Optional-authentication-offload-to-.patch
├── 0003-2748367-nl80211-Free-connkeys-on-external-authentication-fai.patch
├── 0004-2748368-nl80211-Allow-SAE-Authentication-for-NL80211_CMD_CON.patch
├── 0005-2748371-nl80211-Fix-external_auth-check-for-offloaded-authen.patch
├── 0006-2748372-cfg80211-Authentication-offload-to-user-space-in-AP-.patch
├── 0007-2762760-cfg80211-Sync-nl80211-commands-feature-with-upstream.patch
├── 0008-2748378-nl80211-Allow-set-del-pmksa-operations-for-AP.patch
├── 0009-2748379-cfg80211-nl80211-Offload-OWE-processing-to-user-spac.patch
├── 0010-2739268-Kconfig-Add-CLD_LL_CORE-configuration-for-WLAN.patch
└── 0011-2776447-cnss-Add-support-of-cnss-logger-module.patch
└── reg-qcom-call-regulatory-callback-for-self-managed-hints.patch
```

linux-imx_%.bbappend is used to configure the kernel and apply kernel patch to Freescale 4.14.78 kernel.