Qualcomm Technologies, Inc.

Qualcomm® Robotics RB3 Platform

Linux User Guide

Rev. A

February 11, 2019

This technical data may be subject to U.S. and international export, re-export, or transfer (“export”) laws. Diversion contrary to U.S. and international law is strictly prohibited.

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

Revision history

Revision	Date	Description
A	February 2019	Initial release
B	June 2019	Update SNPE/FastCV SDK

Contents

1 RB3 Platform Linux SDK

The Linux SDK for the Qualcomm Robotics RB3 Platform is built by Thundercomm. The platform is based on Yocto Rocko with Linux Kernel 4.9and GCC 6.5.

This document provides information on how to obtain, build, and program SDA845 software into the RB3 Platform.

For more information about software updates and the software developmentkit (SDK) for your host machine operating system please visit:

https://www.thundercomm.com/app_en/product/1544580412842651

1.1 RB3 Platform Linux SDK support

For support, create a request in the Qualcomm Robotics RB3 Platform forum:

https://www.thundercomm.com/forum/category/1/qualcomm-robotics-rb3-platform

2 Set Up the Development Environment

This chapter describes how to set up the Linux development environment on an LU/Windows host machine.

- For compiling code using Qualcomm SDK, LU 14.04 is required.
- For flashing firmware images, an LU/Windows machine is needed.

2.1 Required hardware, software, and equipment

The following table lists the hardware, software, and other equipment required to install and run the software.

Item description	Version	Source/vendor	Purpose
Standalone system minimum requirement: ▪ 8 GB RAM ▪ Quadcore CPU NOTE: Lower specs will increase compilation time. Ideal compilation time required for a system with 16 GB RAM and Intel i7-2600 @3.4 GHz is about an hour.	–	–	SDK Build machine
LU 14.04 LTS Linux distribution for 64-bit architecture	14.04 LTS	LU Community/ Canonical, Ltd.	SDK build host OS
Repo	–	SDK Open Source Project	SDK source management tool
Python	Python 2.7.6 with sqlite3 library	Python.org	Building subsystem
QTI USB driver	QUD.WIN 1.1 Installer 10032.1 or later	https://www.thundercomm.com/app_en/product/1544580412842651#doc	QTI USB WWAN Driver installer for Microsoft Windows
Android SDK tools (ADB, Fastboot)	r10 or later ADB 1.0.29 or later	Android open-source project	ADB and Fastboot tools for Windows

NOTE: Seethe release notes for current versions of the ARM toolchain and Qualcomm® Hexagon™ LLVM toolchain.

2.2 Install LU 14.04 (64-bit) system

Prerequisite: You must be able to log in as root or use sudo to have root permissions during the installation.

1. Create an installation CD (the CD image is LU-14.04.2-desktop-amd64.iso) and install it on the computer following the instructions at:

   http://www.ubuntu.com/download/desktop/install-ubuntu-desktop

2. Perform a software update using one of the following options:

   □ In the GUI, select System>Administration >Update Manager.

   □ From a shell command line:

   a. Edit the source config file to directly enable the universe and multiverse sources and disable the LU installation CD source.

   ```
   sudo vi /etc/apt/sources.list
   ```

   b. From the command line, perform the package list update and package upgrades.

   ```
   sudo apt-get update
   sudo apt-get upgrade
   ```

   c. Use apt-get to install the additional required packages.

   d. See https://source.android.com/source/initializing.html for the required packages. In addition to the packages mentioned in the link, install libssl-dev.

   ```
   sudo apt-get install libssl-dev
   ```

   e. Make bash the default shell (shell /bin/sh to invoke bash) using one of the following options:

   - Use sudo dpkg-reconfigure dash command and reconfigure the package.
   - Manually change the symlink /bin/sh > dash to /bin/sh> bash. Use the sudo rm /bin/sh command to remove
   - Use sudo ln -s /bin/bash /bin/sh

For more information, see the LU Wiki page at: https://wiki.ubuntu.com/DashAsBinSh

2.3 Install ADB, Fastboot, and USB host interface

The Fastboot tool communicates with the RB3 bootloader and allows you to flash images onto the board. This section provides instructions for installing ADB, Fastboot, and USB drivers on your host machine.

2.3.1 On the Windows host machine

Install ADB and Fastboot

Google currently does not offer a standalone Windows Installer for Fastboot. Instead Fastboot is part of the full Android Studio integrated development environment (IDE) installation.

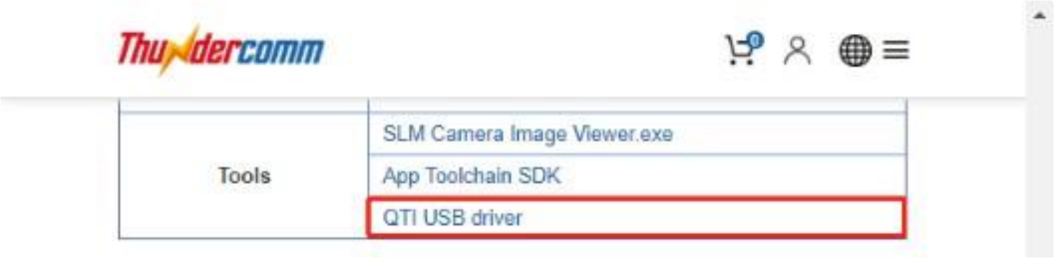
1. Go to http://developer.android.com/sdk/installing/
2. Follow the instructions to install the standalone SDK Tools. During the installation, Fastboot and ADB drivers are installed

NOTE: If you want to install Fastboot without the full Android Studio installation you can find third party installers on the web.

Install the USB drivers

1. Add the system environment path for ADB and Fastboot.
2. Install the Windows drivers. Windows will usually update the drivers automatically from the server. You can also downloadhere:

   https://www.thundercomm.com/app_en/product/1544580412842651#doc



Tools	SLM Camera Image Viewer.exe
	App Toolchain SDK
	QTI USB driver

2.3.2 On the Linux (LU) host

Install ADB and Fastboot

To install ADB and Fastboot, execute the following command:

```
sudo apt-get install android-tools-adb
sudo apt-get install android-tools-fastboot
```

Install and configure the USB driver

Setup the udev rules on your Linux PC as follows:

1. Login as root or sudo and navigate to the directory /etc/udev/rules.d/
2. Add the following rules in file 99-android.rules:

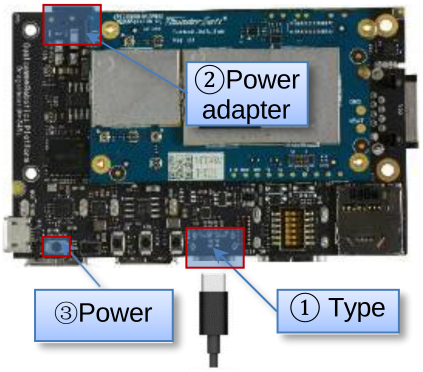
```
# fastboot for Robotics DragonBoard 845c
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", MODE="0777", GROUP="adm"
# adb for Robotics DragonBoard 845c
SUBSYSTEM=="usb", ATTR{idVendor}=="05c6", MODE="0777", GROUP="adm"
```

3. Restart the udev service:

```
$ build@ubuntu$ sudo chmod a+r /etc/udev/rules.d/99-android.rules
$ build@ubuntu$ sudo service udev restart
```

4. Reconnect RB3 to PC with USB cable.
5. Connect the power adapter.
6. Press Power and the system will start.

```
Server@: ~$ adb devices
List of devices attached
7a7d0e08         device
```



2.4 Required build environment

Recommendation: Use LU 14.04 along with gcc/g++ version 4.8.

1. Run following commands to ensure gcc/g++ 4.8 is installed:

```
$ gcc --version
$ g++ --version
```

2. If your host machine has an older version, complete the following steps to upgrade to 4.8:

   a. Run the following commands:

   ```
   $ sudo add-apt-repository ppa:ubuntu-toolchain-r/test
   $ sudo apt-get update
   $ sudo apt-get install gcc-4.8-multilib g++-4.8-multilib
   $ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-
   4.8 100
   $ sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-
   4.8 100
   ```

   b. Select 4.8 in the prompt shown and run the following command:

   ```
   $ sudo update-alternatives --config gcc
   ```

   c. Select 4.8 in the prompt as shown:

   ```
   $ sudo update-alternatives --config g++
   ```

3. To install the dependency library, run the following commands:

   ```
   $ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-
   multilib build-essential chrpath libsdl1.2 -dev xterm openssl libssh-
   dev libssl-dev
   $ sudo cpan install XML::Simple
   ```

4. Check that the bash shell is in use:

   ```
   $ sudo rm /bin/sh
   $ sudo ln -sf /bin/bash /bin/sh
   ```

NOTE: To run the build command, the following packages must be installed: diffstat, makeinfo, and chrpath.

For build environment dependencies, go to the Yocto Project Active Release Documentation page at https://www.yoctoproject.org/docs/, select **YP Core – Rocko 2.4.4**, and view the *Yocto Project Quick Start Guide*.

3 Program the Firmware

The following method is the recommended procedure for downloading your experimental version of self-compiled OS on RB3. A host PC is needed to program the board.

The procedure for both Windows and Linux host systems is described.

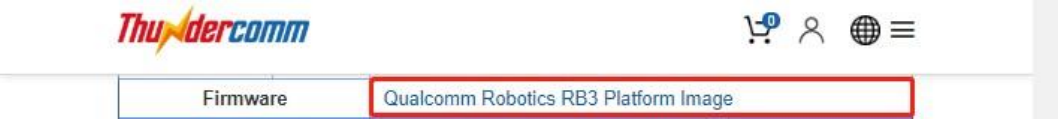
3.1 Program system images using Fastboot

NOTE: Ensure you have installed the required ADB and Fastboot drivers. Your board must be detected on your host machine.

To install Linux from a host PC complete the following steps:

1. Download the RB3 fastboot images package from the Thundercomm website and unzip it.

   https://www.thundercomm.com/app_en/product/1544580412842651#doc

2. Entering to Fastboot
   a. Press and hold **VOL-** then press **Power** once quickly to force the device to enter Fastboot mode
   b. Alternatively "adb reboot bootloader" from the command prompt
3. Plug the USB cable into the Device Type C port.
4. Confirm that Fastboot is active as follows:
   a. From the Windows command shell, run:

```
$fastboot devices
dae93bbb    fastboot
```

   b. From Linux, run:

```
$ sudo fastboot devices
dae93bbb    fastboot
```

5. Flash images:
   a. From the Windows command shell, run:

```
$ flash_all.bat
```

   b. From Linux, run:

```
$ sudo flash_all.sh
```

After the script is executed, the board will reboot automatically.

4 RB3 Platform Features and Use Cases

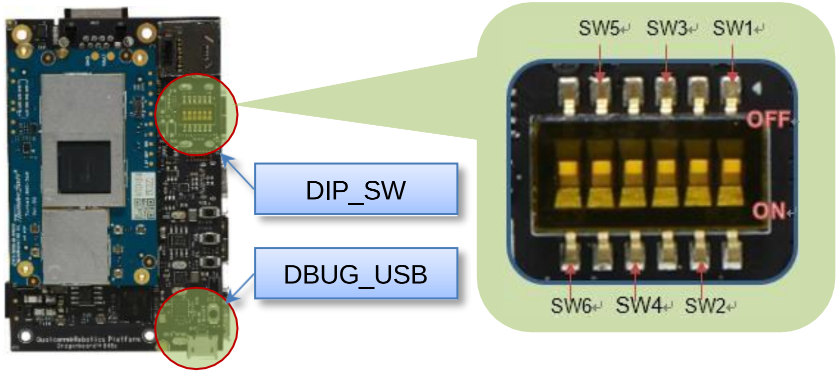
This chapter presents RB3 Platform features and use cases.

4.1 Set up the serial port

To enable DBUG_USB:

1. Turn ON SW2 on switch DIP_SW:

2. Log in using the following account credentials:

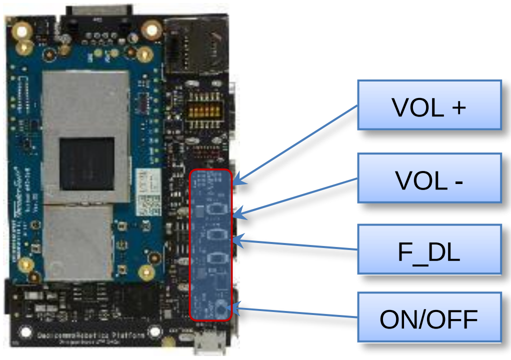


```
… …
[  OK  ] Started start dsp variants.
[  OK  ] Reached target Multi-User System.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started Update UTMP about System Runlevel Changes.

robot 201812300250 sda845 ttyMSM0

sda845 login: root
Password: 123456
```

NOTE: SElinux is turned on by default. Don't support to login as the root user. If need to log in, please close the SElinux as the instructions below :
a. Connect the host PC to the type C via the usb cable.
b. $ adb shell
  # setenforce 0

4.2 Button events

The following figure shows the button positions.



Key	Device node
VOL +	/dev/input/event2
VOL -	/dev/input/event0
F_DL	/dev/input/event2
ON/OFF	/dev/input/event0

NOTE: F_DL is forced download mode (emergency download)

1. Get interrupt event:

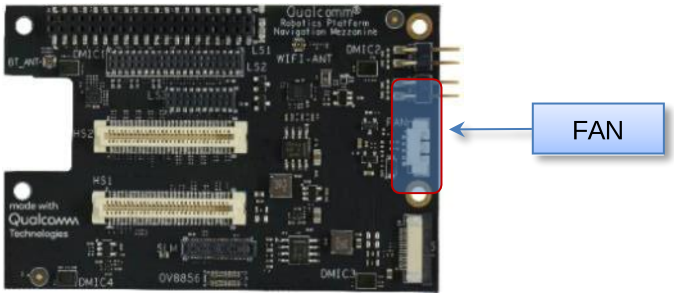
```
$ adb shell
# hexdump /dev/input/event0
```

- Press the “key -" button. The terminal displays the following information:

```
# hexdump /dev/input/event0
0000000 6a1d 5c2f 0000 0000 c499 000d 0000 0000
0000010 0001 0072 0001 0000 6a1d 5c2f 0000 0000
0000020 c499 000d 0000 0000 0000 0000 0000 0000
```

4.3 Set up the FAN control interface

The following figure shows the FAN interface position.



1. Enable FAN: (1/2/3 indicate different speed levels: 1minimum, 3maximum):

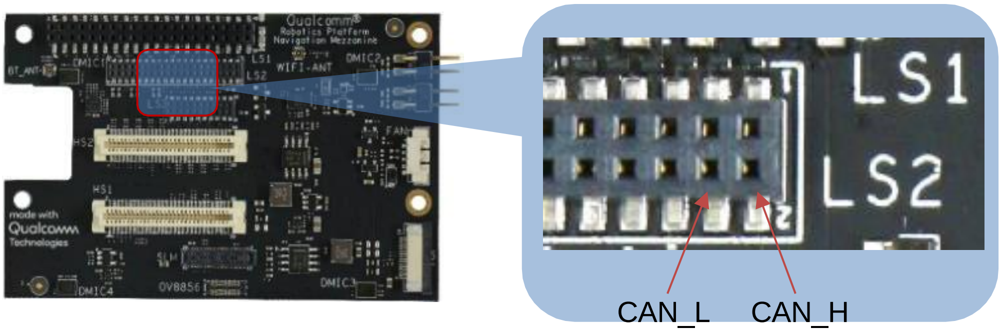
```
$ adb shell
# echo 1 > /sys/kernel/fan/speed
# echo 2 > /sys/kernel/fan/speed
# echo 3 > /sys/kernel/fan/speed
```

2. Disable FAN:

```
$ adb shell
# echo 0 > /sys/kernel/fan/speed
```

4.4 Configure CAN interface

The following figure shows the CAN interface positions.



1. Connect CAN_L and CAN_H for loopback test.
2. Configure the CAN device:

```
$adb shell
#ip link set can0 down                              // Disable CAN;
#ip link set can0 up type can bitrate 800000        // Enable CAN;
```

3. Perform the data loopback test.

   a. Receive data:

```
$adb shell
#candump can0
interface = can0, family = 29, type = 3, proto = 1
```

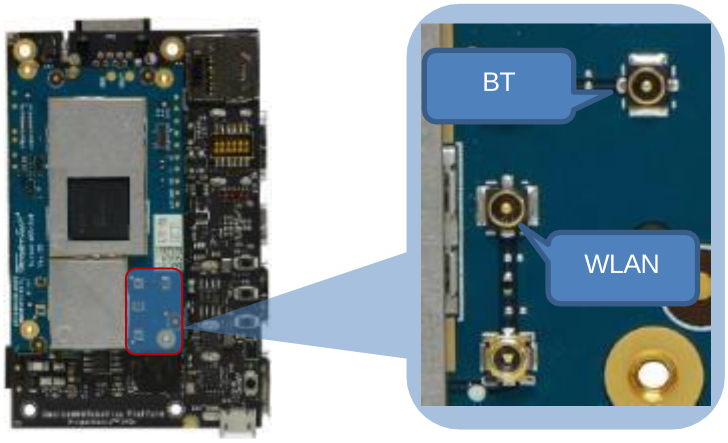
   b. Open another terminal, send data:

```
$adb shell
#cansend can0 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
```

See Section 5.4 for the source location and compilation of the command

4.5 Connectivity

The following figure shows the WLAN and BT connectivity.



4.5.1 Set up Wi-Fi

1. Confirm the antenna is properly connected.
2. To verify connectivity while the device is in Station mode,execute the following commands

```
$ adb shell
# vi /data/misc/wifi/wpa supplicant.conf
```

Fill in the ssid and psk of wifi as follows.

```
update_config=1
eapol_version=1
ap_scan=1
fast_reauth=1
pmf=1
p2p_add_cli_chan=1
network={
     ssid=”wifissid1”
     psk=”wifipsk1”
}
Network={
     ssid=”wifissid2”
     psk=”wifipsk2”
}
```

3. Restart the device and connect to wifi.

Run the following command to confirm that the device is connected to wifi.The log of IP address acknowledgement proves the connection issuccessful:

```
$ adb shell
#ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 00:0A:F5:83:66:EF
          inet addr:192.168.43.92  Bcast:192.168.43.255
Mask:255.255.255.0
```

```
          inet6 addr: fe80::20a:f5ff:fe83:66ef%1736140884/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3000
          RX bytes:1864 (1.8 KiB)  TX bytes:1828 (1.7 KiB)
# ping www.qualcomm.com
PING www.qualcomm.com (23.59.93.51): 56 data bytes
64 bytes from 23.59.93.51: seq=0 ttl=48 time=171.722 ms
64 bytes from 23.59.93.51: seq=1 ttl=48 time=163.070 ms
64 bytes from 23.59.93.51: seq=2 ttl=48 time=244.932 ms
64 bytes from 23.59.93.51: seq=3 ttl=48 time=166.135 ms
```

NOTE: If ping does not work, check the firewall and try to ping outside the firewall.

4.5.2 Bluetooth Generic Access Profile (GAP)

1. Before running btapp, run the btproperty in the background (run only once at the beginning):

```
$adb shell
# btapp
```

2. After running btapp, type **gap_menu** and press **Enter**.

```
gap_menu
***************** Menu *******************
         enable
         disable
         inquiry
         cancel_inquiry
         pair<space><bt_address>       eg. pair 00:11:22:33:44:55
         unpair<space><bt_address>     eg. unpair 00:11:22:33:44:55
         inquiry_list
         bonded_list
         get_state
         get_bt_name
         get_bt_address
         set_bt_name<space><bt name>     eg. set bt name MDM Fluoride
         set_le_bt_name<space><bt name>  eg. set_le_bt_name
MDM_LE_Fluoride
         main_menu
 *****************************************
```

4.5.3 Enable Bluetooth

After running btapp, input “**enable**” and press “**Enter**”enable

```
enable
killall: wcnssfilter: no process killed
killall: btsnoop: no process killed
killall: qcbtdaemon: no process killed
/bin/sh: qcbtdaemon: not found
 BtHfpAgMsgHandler event = 1028
ACDB -> No .acdb files found in /etc/acdbdata/!
ACDB -> found 0 form factor & soundcard independant files
… …
ACDB -> MBHC ACDB_PID_GENERAL_CONFIG
ACDB -> MBHC ACDB_PID_PLUG_REMOVAL_DETECTION
ACDB -> MBHC ACDB_PID_PLUG_TYPE_DETECTION
ACDB -> MBHC ACDB_PID_BUTTON_PRESS_DETECTION
ACDB -> MBHC ACDB_PID_IMPEDANCE_DETECTION
send vbat data
ACDB -> VBAT ACDB_PID_ADC_CAL
ACDB -> VBAT ACDB_PID_GAIN_PROC
send vbat data, calling convert_vbat_data
Vbat Registers Size: 17
copied vbat cal size =72
BT State is ON
```

4.5.4 Start inquiry

After running enable, type **inquiry** and press **Enter** to start inquiry.

```
inquiry
 Inquiry Started
Device Found details:
Found device Addr: 28:11:a5:01:00:a2
Found device Name: LE-Bose SoundSport
Device class is: 7936
Device Found details:
Found device Addr: e4:ba:d9:10:00:c9
Found device Name: 360FLY4K_00C8
Device class is: 7936
Device Found details:
Found device Addr: 28:11:a5:24:01:05
Found device Name: LE-reserved_N
Device class is: 7936
Inquiry Stopped automatically
```

NOTE: To cancel inquiry, issue the following command while the inquiry in progress:cancel_inquiry

4.5.5 Check the inquiry list

After running inquiry, type**inquiry_list** and press **Enter** to check the list.

```
inquiry_list
**************************** Inquiry List*******************************
LE-Bose SoundSport                               28:11:a5:01:00:a2
360FLY4K_00C8                                    e4:ba:d9:10:00:c9
LE-reserved_N                                    28:11:a5:24:01:05
**************************** End of List ******************************
```

4.5.6 Pair outgoing SSP

- Use the following command to pair outgoing SSP:

```
pair<bd_address>
```

- To accept or reject the outgoing pairing for the following example (pair e4:ba:d9:10:00:c9), type **Yes** or **No** and press **Enter**.

```
pair e4:ba:d9:10:00:c9
************************************************
 BT pairing request::Device iPhone::Pairing Code:: 281155
************************************************
 ** Please enter yes / no **
yes
************************************************
 Pairing state for 360FLY4K_00C8 is BONDED
************************************************
```

4.5.7 Check the bonded list

1. After running btapp, type **bonded_list** and press **Enter** to check the bonded list:

```
bonded_list
************************** Bonded Device List *************************
360FLY4K_00C8                                       a4:f1:e8:c6:2f:b4
**************************** End of List *****************************
```

2. Disconnect bonded, type **disable** and press **Enter** disable:

```
disable
killall: qcbtdaemon: no process killed
 BtHfpAgMsgHandler event = 1029
killall: wcnssfilter: no process killed
 BT State is OFF
```

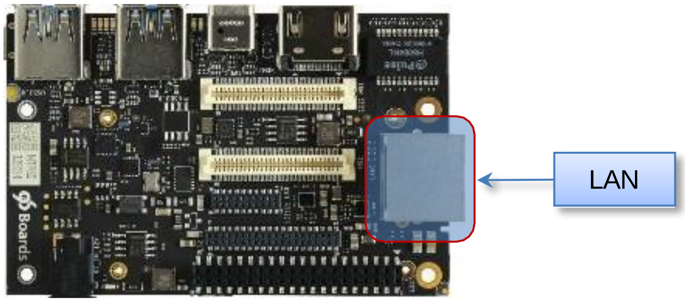
3. To exit from btapp, navigate to the main menu and enter the following command:

```
exit
```

See Section 5.4 for the source location and compilation of the command

4.6 Ethernet

The following figure shows the LAN port.



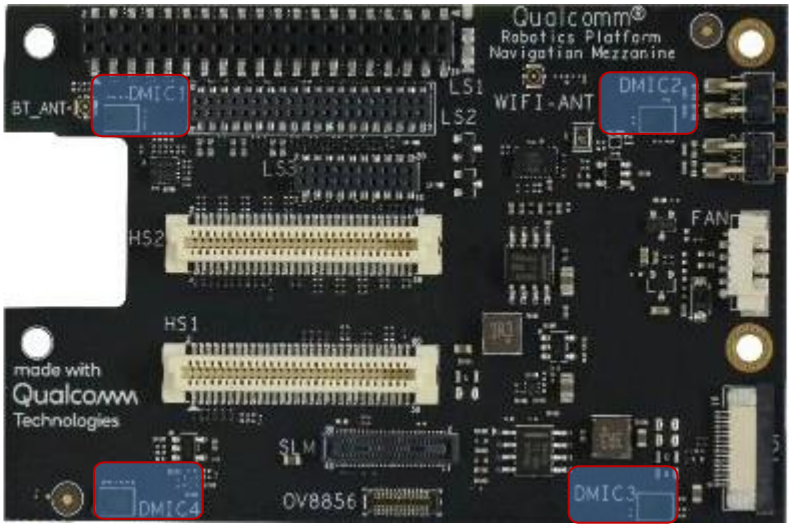
- Connect the LAN cable to the LAN port.

```
$ adb shell
# ifconfig
enp1s0u3  Link encap:Ethernet  HWaddr 00:0E:C6:81:79:01
          inet addr:192.168.7.196  Bcast:192.168.7.255
Mask:255.255.255.0
          inet6 addr: fe80::20e:c6ff:fe81:7901%1819682900/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:319 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:34988 (34.1 KiB)  TX bytes:1774 (1.7 KiB)

# ping www.qualcomm.com
PING www.qualcomm.com (140.205.16.110): 56 data bytes
64 bytes from 140.205.16.110: seq=0 ttl=40 time=38.816 ms
64 bytes from 140.205.16.110: seq=1 ttl=40 time=42.177 ms
64 bytes from 140.205.16.110: seq=2 ttl=40 time=38.260 ms
^C
--- www.qualcomm.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 38.260/39.751/42.177 ms
```

NOTE: If ping does not work, check the firewall and try to ping outside the firewall.

4.7 Digital microphone

The following figure shows the digital microphone (DMIC) positions:



1. The board has four DMICs. Use dmic-ctl to perform separate DMIC enable, disable, and record operations.

```
$adb shell
# dmic-ctl
/usr/bin/dmic-ctl <MDIC_NUMBER><SWITCH>
DMIC_NUMBER: [0, 1, 2, 3]
SWITCH: enable, disables
```

2. In the following example DMIC1 is used.

   a. To enable DMIC1:

   ```
   $ adb shell
   # dmic-ctl 0 enable
   ```

   b. To start recording, enter arecord and press Enter:

   ```
   # arecord /data/dmic0_test.wav -f S16_LE -c 1 -r 48000 -d 10
   ```

   c. The command parameter setting only records for 10 seconds. After 10 seconds, the recording stops.

   d. Move the recording file to the PC for playback verification or use a tool like Audacity to view the waveform.

   ```
   $ adb pull /data/dmic0_test.wav .
   ```

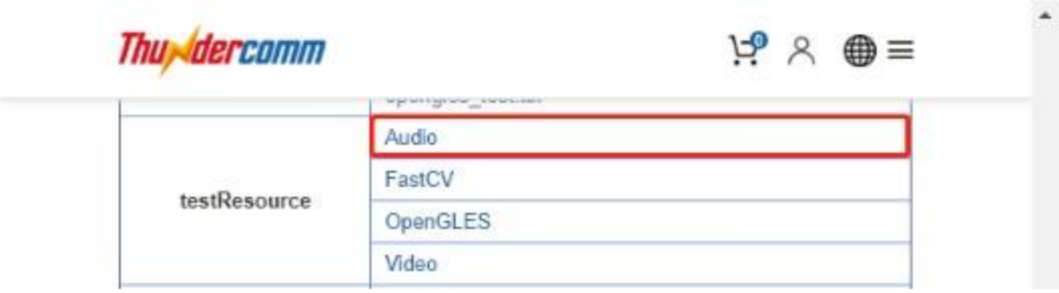
   e. To disable DMIC1:

   ```
   $ adb shell
   #dmic-ctl 1 disables
   ```

4.8 Audio

1. To verify the audio decoding functionality download the test files fromhere.

   https://www.thundercomm.com/app_en/product/1544580412842651#doc



testResource	Audio
	FastCV
	OpenGLES
	Video

And See Section 5.5for the source location and compilation of the command

2. Unzip to extract the contents of the file:
   - □ Audio_Decode_WAV_Stereo_48KHz_16Bit.wav;
   - □ Audio_Decode_AAC-LC_adts_32KHz_stereo.aac;
   - □ Audio_Decode_HE-AAC V1_stereo_32KHz.aac;
   - □ Audio_Decode_HE-AAC V2_stereo_44.1KHz.m4a;
   - □ Audio_Decode_MP3v1_32KHz_cbr_stereo.mp3;

4.8.1 Playback WAV using amixer + aplay

```
$ adb push Audio_Decode_WAV_Stereo_48KHz_16Bit.wav /data/
```

1. To enable speaker:

```
$ adb shell
# spk-ctl enable
```

2. To play:

```
$ adb shell
# aplay /data/Audio Decode WAV Stereo 48KHz 16Bit.wav
```

3. To disable speaker:

```
$ adb shell
# spk-ctl disables
```

4.8.2 Playback AAC_LC using hal_play_test_64bit

```
$ adb push Audio_Decode_AAC-LC_adts_32KHz_stereo.aac /data/
# hal_play_test_64bit -f /data/Audio_Decode_AAC-LC_adts_32KHz_stereo.aac -t
4 -d 2 -v 0.3 -r 32000 -c 2 -a 1
```

4.8.3 HE-AAC-V1 using hal_play_test_64bit

```
$ adb push Audio_Decode_HE-AAC V1_stereo_32KHz.aac /data/
# hal_play_test_64bit -f /data/Audio_Decode_HE-AAC\ V1_stereo_32KHz.aac -t
4 -d 2 -v 0.3 -r 16000 -c 2 -a 2
```

4.8.4 HE-AAC-V2 using hal_play_test_64bit

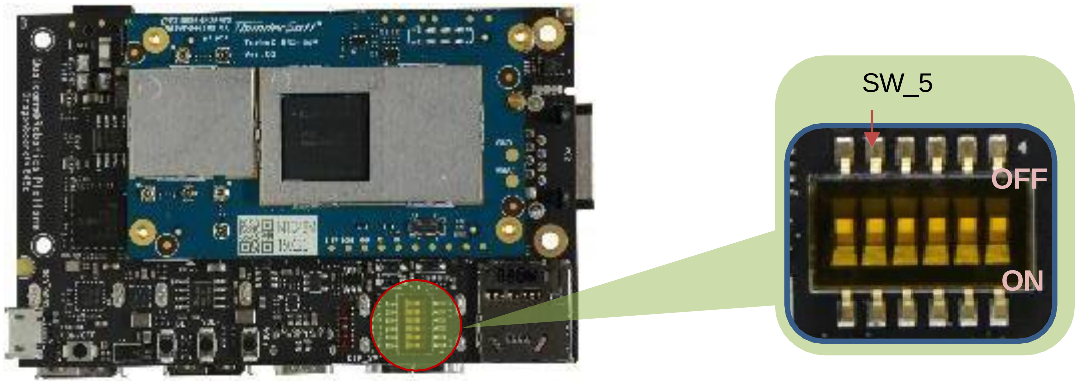
```
$ adb push Audio_Decode_HE-AAC V2_stereo_44.1KHz.m4a /data/
# hal_play_test_64bit -f /data/Audio_Decode_HE-AAC\ V2_stereo_44.1KHz.m4a -
t 3 -d 2 -v 0.3 -r 22050 -c 2 -a 3
```

4.8.5 Mp3 using hal_play_test_64bit

```
$ adb push Audio_Decode_MP3v1_32KHz_cbr_stereo.mp3 /data/
# hal_play_test_64bit -f /data/Audio_Decode_MP3v1_32KHz_cbr_stereo.mp3 -t 2
-d 2 -v 0.3 -r 32000 -c 2 -a 1
```

4.9 Sensors

The device supports five sensor types: accelerometer, gyroscope, magnetometer, proximity, and light. The sensors are situated in two groups.



The following table shows the sensor modules enabled using the SW_5 switch.

SW_5	Sensor	Sensor_name	Sensor type ID
ON	Magnetometer	ak0991x Magnetometer Wakeup	2
	Pressure	icp101xx Pressure Sensor Wakeup	6
OFF	Accelerometer	icm4x6xx Accelerometer Wakeup	1
	Gyroscope	icm4x6xx Gyroscope Wakeup	4
	Proximity	ltr559 Proximity Sensor Wakeup	8
	Light	ltr559 Ambient Light Sensor Wakeup	5

NOTE: Whenever the SW_5 switch is toggled, the device must be rebooted.

See Section 5.6 for the source location and compilation of the command

4.9.1 Get sensor data via sns_hal_batch

To get sensor data via sns_hal_batch:

```
$ adb shell
# sns_hal_batch --help
Usage: sns_hal_batch [OPTIONS]...

-h --help                 Print this message
-l --listsensors             List all available sensors and their
attributes
-o --output               the output file to write the sensor values to
                        default: /data/local/sns_hal_batch.out

Providing no parameter options runs the interactive command line
interface
Providing 1 or more parameters to sns_hal_batch will run the following
sequence:
        - set batching parameters for the sensor: (sampling rate, report
rate)
        - activate the sensor
        - wait for the specified duration
        - deactivate the sensor

The parameters, as well as their default values, are as follows:
        -s  --sensor            the android sensor type enumeration
value
                                default: 1 for
android.sensor.accelerometer
        -w  --wakeup          flag for wakeup or non-wakeup sensor
                                        w for wakeup
                                        n for non-wakeup
                                        d for don't care or default
                                default: d
        -sr --samplingrate        the sampling rate (in Hz)
                                default: 5 Hz
        -rr --reportrate           the report rate (in Hz)
                                use 0 for no batching
                                (report events as available)
                                default: 0
        -d  --duration           the duration (in seconds) to run the
sensor for
                                default: 10 seconds
```

4.9.2 Get accelerometer data

To get accelerometer data, set SW_5 to OFFand SW_6 to ONposition:

NOTE: whenever SW_5 switch is toggled, device has to be rebooted

```
# sns_hal_batch -s 1 -sr 10 -rr 10 -d 30
HAL open
HAL module_api_version: 0x1
HAL hal_api_version   : 0x256
HAL hal_id            : sensors
HAL hal_name          : QTI Sensors HAL Module
HAL hal_author        : Qualcomm Technologies, Inc.
get_sensors_list took 9815969424 nanoseconds
batch success
activate success
Activated sensor [Type: 1] icm4x6xx Accelerometer Wakeup (d) for 30
seconds, sampling at 10.000000 Hz and reporting at 10.000000 Hz
[Type:  1] android.sensor.accelerometer (wakeup)
        Name:icm4x6xx Accelerometer Wakeup Vendor:TDK-Invensense
Version:275 Handle:17
        maxRange: 156.906403 resolution: 0.000299 power: 0.240000 mA
        minDelay: 2000 us maxDelay: 1000000 us
        fifoReservedEventCount: 3000 fifoMaxEventCount: 10000
        requiredPermission:
Sleeping for 30 seconds before deactivating and exiting
88390.952725, android.sensor.accelerometer/icm4x6xx Accelerometer
Wakeup, 1548935833753.566636, 0.143171, -1.052250, -9.740259, 0.000000,
-4645832325944140928, latency(ms): 16897808328346.937705
```

4.9.3 Get gyroscope data

To get gyroscope data, set SW_5 to OFFand SW_6 to OFF position:

NOTE: whenever SW_5 switch is toggled, device has to be rebooted

```
# sns_hal_batch -s 4 -sr 10 -rr 10 -d 30
HAL open
HAL module_api_version: 0x1
HAL hal_api_version   : 0x256
HAL hal_id            : sensors
HAL hal_name          : QTI Sensors HAL Module
HAL hal_author        : Qualcomm Technologies, Inc.
get_sensors_list took 9877721611 nanoseconds
batch success
activate success
Activated sensor [Type: 4] icm4x6xx Gyroscope Wakeup (d) for 30 seconds,
sampling at 10.000000 Hz and reporting at 10.000000 Hz
[Type:  4] android.sensor.gyroscope (wakeup)
```

```
        Name:icm4x6xx Gyroscope Wakeup Vendor:TDK-Invensense Version:275
Handle:11
        maxRange: 34.905556 resolution: 0.000067 power: 0.570000 mA
        minDelay: 2000 us maxDelay: 1000000 us
        fifoReservedEventCount: 0 fifoMaxEventCount: 10000
        requiredPermission:
Sleeping for 30 seconds before deactivating and exiting
288323.834784, android.sensor.gyroscope/icm4x6xx Gyroscope Wakeup,
1548936033672.105552, 0.000239, 0.001253, 0.006258, 0.000000,
4225550931719807308, latency(ms): 16897808328361.280848
```

4.9.4 Get magnetometer data

To get data from the magnetometer, set SW_5 to ON position:

NOTE: whenever SW_5 switch is toggled, device has to be rebooted

```
# sns_hal_batch -s 2 -sr 10 -rr 10 -d 30
HAL open
HAL module_api_version: 0x1
HAL hal_api_version   : 0x256
HAL hal_id            : sensors
HAL hal_name          : QTI Sensors HAL Module
HAL hal_author        : Qualcomm Technologies, Inc.
get_sensors_list took 10010516142 nanoseconds
batch success
activate success
Activated sensor [Type: 2] ak0991x Magnetometer Wakeup (d) for 30
seconds, sampling at 10.000000 Hz and reporting at 10.000000 Hz
[Type:  2] android.sensor.magnetic_field (wakeup)
        Name:ak0991x Magnetometer Wakeup Vendor:akm Version:10058
Handle:3
        maxRange: 4912.000000 resolution: 0.150000 power: 1.100000 mA
        minDelay: 10000 us maxDelay: 1000000 us
        fifoReservedEventCount: 600 fifoMaxEventCount: 10000
        requiredPermission:
Sleeping for 30 seconds before deactivating and exiting
510179.881783, android.sensor.magnetic_field/ak0991x Magnetometer
Wakeup, 1550460581087.400447, -20.989037, 26.326185, 14.180205,
0.000000, 4743024914702264716, latency(ms): 16896284002802.032952
```

4.9.5 Get pressure sensor data

To get data from the pressure sensor, set SW_5 to ON position.

NOTE: whenever SW_5 switch is toggled, device has to be rebooted

```
# sns_hal_batch -s 6 -sr 10 -rr 10 -d 30
HAL open
HAL module_api_version: 0x1
HAL hal_api_version   : 0x256
HAL hal_id            : sensors
HAL hal_name          : QTI Sensors HAL Module
HAL hal_author        : Qualcomm Technologies, Inc.
get_sensors_list took 9613565153 nanoseconds
batch success
activate success
Activated sensor [Type: 6] icp101xx Pressure Sensor Wakeup (d) for 30
seconds, sampling at 10.000000 Hz and reporting at 10.000000 Hz
[Type:  6] android.sensor.pressure (wakeup)
Name:icp101xx Pressure Sensor Wakeup Vendor:TDK-Invensense Version:65544
Handle:41
maxRange: 1150.000000 resolution: 0.000100 power: 0.010000 mA
minDelay: 40000 us maxDelay: 1000000 us
fifoReservedEventCount: 300 fifoMaxEventCount: 10000
requiredPermission:
Sleeping for 30 seconds before deactivating and exiting
182936428.889120, android.sensor.pressure/icp101xx Pressure Sensor
Wakeup, 1547642389965.965156, 1012.952209, 0.000000, 0.000000, 0.000000,
1149058289, latency(ms): 16899284620172.475580
```

4.9.6 Get light sensor data

To get data from the light sensor, set SW_5 to OFF position.

NOTE: whenever SW_5 switch is toggled, device has to be rebooted

```
Whenever SW_5 switch is toggled, device has to be rebooted
# sns_hal_batch -s 5 -sr 10 -rr 10 -d 30
HAL open
HAL module_api_version: 0x1
HAL hal_api_version   : 0x256
HAL hal_id            : sensors
HAL hal_name          : QTI Sensors HAL Module
HAL hal_author        : Qualcomm Technologies, Inc.
get_sensors_list took 3367598176 nanoseconds
batch success
activate success
Activated sensor [Type: 5] ltr559 Ambient Light Sensor Wakeup (d) for 30
seconds, sampling at 10.000000 Hz and reporting at 10.000000 Hz
[Type:  5] android.sensor.light (wakeup)
        Name:ltr559 Ambient Light Sensor Wakeup Vendor:Lite-On
Version:256 Handle:25
        maxRange: 1.000000 resolution: 0.100000 power: 0.110000 mA
        minDelay: 0 us maxDelay: 0 us
        fifoReservedEventCount: 0 fifoMaxEventCount: 10000
        requiredPermission:
```

```
Sleeping for 30 seconds before deactivating and exiting
1123650.618788, android.sensor.light/ltr559 Ambient Light Sensor Wakeup,
1546110672090.233843, 0.000000, 0.000000, 0.000000, 0.000000, 0,
latency(ms): 16900634525269.936561
```

4.9.7 Get proximity sensor data

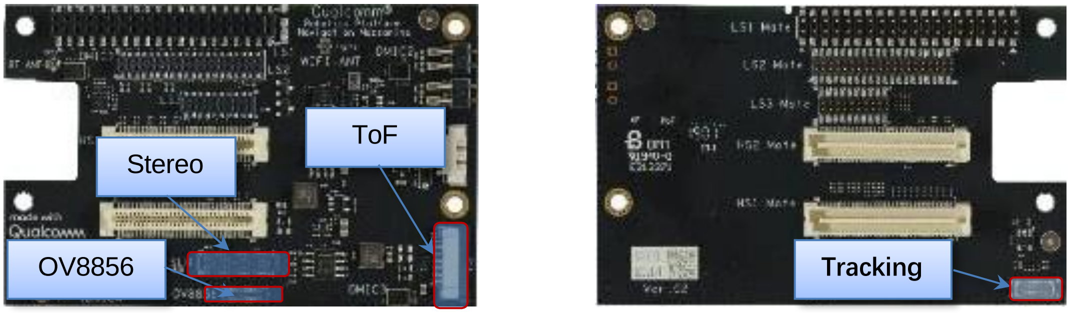
To get data from the proximity sensor, set SW_5 to OFF position:

NOTE: whenever SW_5 switch is toggled, device has to be rebooted

```
Whenever SW_5 switch is toggled, device has to be rebooted
# sns_hal_batch -s 8 -sr 10 -rr 10 -d 30
HAL open
HAL module_api_version: 0x1
HAL hal_api_version   : 0x256
HAL hal_id            : sensors
HAL hal_name          : QTI Sensors HAL Module
HAL hal_author        : Qualcomm Technologies, Inc.
get_sensors_list took 9779863069 nanoseconds
batch success
activate success
Activated sensor [Type: 8] ltr559 Proximity Sensor Wakeup (d) for 30
seconds, sampling at 10.000000 Hz and reporting at 10.000000 Hz
[Type:  8] android.sensor.proximity (wakeup)
Name:ltr559 Proximity Sensor Wakeup Vendor:Lite-On Version:256 Handle:27
maxRange: 1.000000 resolution: 0.100000 power: 0.150000 mA
minDelay: 0 us maxDelay: 0 us
fifoReservedEventCount: 300 fifoMaxEventCount: 10000
requiredPermission:
Sleeping for 30 seconds before deactivating and exiting
237290.761418, android.sensor.proximity/ltr559 Proximity Sensor Wakeup,
1547459690500.729312, 1.000000, 0.000000, 0.000000, 0.000000,
1065353216, latency(ms): 16899284620499.583722
```

4.10 Camera

The device supports four camera interfaces as shown in the following figure:



The sensor ID changes depending on the number of camera sensors connected simultaneously.

Use the following matrix to calculate the ID for the camera devices attached in various combinations:

SENSOR ID MATRIX					
x = sensor not connected	camera sensor type				
No of sensors connected	Sr. No	Stereo	ToF	OV8856	Tracking (OV7251)
1	1.1	0	x	x	x
	1.2	x	0	x	x
	1.3	x	x	0	x
	1.4	x	x	x	0
2	2.1	0	1	x	x
	2.2	0	x	1	x
	2.3	0	x	x	1
	2.4	x	0	1	x
	2.5	x	0	x	1
	2.6	x	x	0	1
3	3.1	0	1	2	x
	3.2	0	1	x	2
	3.3	0	x	1	2
	3.4	x	0	1	2
4	4.1	0	1	2	3

For example:

- Sr.No 1.4: Here we are connecting only one of the supported sensors. “cameraID = 0” will select “Tracking”
- Sr.No 2.3: Here we are connecting stereo camera along with tracking(2 camera sensor scenario). In this case, “cameraID=1” will select “Tracking” while “cameraID=0” will select “Stereo”

The hal3_test application is available to capture image streams provided by the cameras. This application is written using the HAL3 API.

See Section 5.7 for the source location and compilation of the command

- To run the hal3_testapplication, log in to the device as a root (using serial or adb shell).

```
$ adb shell
#  hal3_test -h
Enter Camera Testing
            ================== Camera Test Version  ==================
             CAMTEST_SHA1    : 0aea4e256a34e2eaa014d251c2a794b684c70e00
             CAMTEST_BUILD_TS: 10/24/2018 12:6:35
             CAMTESTHOSTNAME : ecbld-bd116-lnx.qualcomm.com
             CAMBUILD_IP     : 10.225.16.179/23
            ==========================================================
opt:h
 usage: hal3_test [-h] [-f command.txt]
 -h       show usage
 -f       using commands in file

command in program:
<order>:[Params]
 Orders:
  A: ADD a camera device for test
>>A:id=0,psize=1920x1080,pformat=yuv420,ssize=1920x1080,sformat=jpeg
  U: Update meta setting
>>U:manualaemode=1
  D: Delete current camera device
>>D
  S: trigger Snapshot and dump preview/video
>>Ss:2 num  take num(1,2,..) picture(s), eg. s:1
  s: trigger Snapshot
>> s:num  take num(1,2,..) picture(s), eg. S:2 S:2
  v: triger video that switch from preview
>>v:id=0,psize=1920x1080,pformat=yuv420,vsize=1920x1080,ssize=1920x1080,
sformat=jpeg
  p: trigger dump Preview
>> p:num  dump num(1,2,..) preview frame(s), eg. p:3 p:2
  M: set Metadata dump tag
>>M:expvalue=1,scenemode=0
Q: Quit
```

4.10.1 Main Camera (OV8856)

NOTE: The default directory where hal3_test saves files is: /data/misc/camera/.

- Start the camera with preview mode (size:1920x1080, format:YUV420)
  - pszie: set preview size.
  - pformat: set preview format and support format: yuv420, yuv_ubwc, raw10,raw8,raw16.
  - P:1, dump one preview frame.

    ```
    $ adb shell
    # hal3_test
    CAM0>> A:id=2,psize=1920x1080,pformat=yuv420
    CAM0>> P:1
    CAM0>> D
    CAM0>> Q
    # exit
    $ adb pull /data/misc/camera/”filename”  .
    ```

- Set the camera auto awb mode.
  - manualawbmode: set manual abw mode(0:off 1:auto)
  - file=ALOGE, tag=MyTest: enable monitor abw status, using ALOG to output log with MyTest tag.

    ```
    $ adb shell
    # hal3_test
    CAM0>> A:id=2,psize=1920x1080,pformat=yuv420
    CAM0>> M:awbmode=1,file=ALOGE,tag=MyTest
    CAM0>> U:manualawbmode=1
    CAM0>> P:1
    CAM0>> D
    CAM0>> Q
    # exit
    $ adb pull /data/misc/camera/”filename”  .
    ```

- Start preview with manual ae mode on.
  - manualaemode: set manual aemode(0:off, 1:on)

    ```
    $ adb shell
    # hal3_test
    CAM0>> A:id=2,psize=1920x1080,pformat=yuv420
    CAM0>> M:aemode=1,file=ALOGE,tag=MyTest
    CAM0>> U:manualaemode=1

    //Test under Lowlight
    CAM0>> P:5

    //switch to  Brightlight
    CAM0>> P:5
    ```

```
CAM0>> D
CAM0>> Q
# exit
$ adb pull /data/misc/camera/”filename”  .
```

- Start preview with snapshot (size:3264x2448,format:jpeg)
  - sszie: set snap shot size.
  - sformat: set snap shot format, support format: yuv420, jpeg, raw10,raw16.

```
$ adb shell
# hal3_test
CAM0>>
A:id=2,psize=1920x1080,pformat=yuv420,ssize=3264x2448,sformat=jpeg
CAM0>> P:1
CAM0>> s:1
CAM0>> D
CAM0>> Q
# exit
$ adb pull /data/misc/camera/”filename”  .
```

- Snapshot with auto awb mode.

```
$ adb shell
# hal3_test
CAM0>>
A:id=2,psize=1920x1080,pformat=yuv420,ssize=3264x2448   ,sformat=jpeg
CAM0>> M:awbmode=1,file=ALOGE,tag=MyTest
CAM0>> U:manualawbmode=1
CAM0>> P:1
CAM0>> s:1
CAM0>> D
CAM0>> Q
# exit
$ adb pull /data/misc/camera/”filename”  .
```

- Snapshot_Camera_Auto Flicker Detection
  - manualantimode: set manual antimode(0:off,1:50HZ,2:60HZ,3:auto)

```
zslmode: en/disable zsl mode(0:disalbe,1:enable)
$ adb shell
# hal3_test
CAM0>>
A:id=2,psize=1920x1080,pformat=yuv420,ssize=3264x2448,sformat=jpeg
CAM0>> M:zslmode=1,ae_antimode=1,file=ALOGE,tag=MyTest
CAM0>> U:manualantimode=3,manualzslmode=1
CAM0>> s:1
CAM0>> U:manualantimode=2,manualzslmode=1
CAM0>> s:1
CAM0>> U:manualantimode=1,manualzslmode=1
CAM0>> s:1
CAM0>> U:manualantimode=0,manualzslmode=1
```

```
CAM0>> s:1
CAM0>> D
CAM0>> Q
# exit
$ adb pull /data/misc/camera/”filename”  .
```

- Snapshot_ Camera_Digital Zoom
  - manualcropregion: set zoom parameters(leftxtopxwidthxheight),
  - e.g., 1306x979x653x490(crop region left:1306, top:979,width:653,height:490)

```
$ adb shell
# hal3_test
CAM0>>
A:id=2,psize=1920x1080,pformat=yuv420,ssize=3264x2448,sformat=jpeg
CAM0>> M:zslmode=1,zoomvalue=1,file=ALOGE,tag=MyTest
CAM0>> U:manualzslmode=1,manualcropregion=0x0x3264x2448
CAM0>> P:1
CAM0>> s:1
CAM0>> U:manualzslmode=1,manualcropregion=1306x979x653x490
CAM0>> P:1
CAM0>> s:1
CAM0>> U:manualzslmode=1,manualcropregion=1428x1071x408x306
CAM0>> P:1
CAM0>> s:1
CAM0>> D
CAM0>> Q
# exit
$ adb pull /data/misc/camera/”filename”  .
```

- Snapshot_Camera_MFNR
  - Turn on MFNR:

```
$ adb root
$ adb remount
$ adb shell mount -o rw,remount /
$ adb shell echo "overrideEnableMFNR=TRUE" >
/vendor/etc/camera/camxoverridesettings.txt
$ adb shell echo "advanceFeatureMask=0x02" >>
/vendor/etc/camera/camxoverridesettings.txt

$ adb shell
# hal3_test
CAM0>>
A:id=2,psize=1920x1080,pformat=yuv420,ssize=3264x2448,sformat=jpeg
CAM0>> M:zslmode=1,file=ALOGE,tag=MyTest
CAM0>> U:manualzslmode=1
CAM0>> P:1
CAM0>> s:1
CAM0>> D
CAM0>> Q
```

```
# exit
$ adb pull /data/misc/camera/”filename”  .
```

- Turn off MFNR:

```
$ adb shell "echo "overrideEnableMFNR=False" >
/vendor/etc/camera/camxoverridesettings.txt"
$ adb shell "echo "advanceFeatureMask=0x01" >>
/vendor/etc/camera/camxoverridesettings.txt"
$ adb reboot

$ adb shell
# hal3_test
CAM0>>
A:id=2,psize=1920x1080,pformat=yuv420,ssize=3264x2448,sformat=jpeg
CAM0>> M:zslmode=1,file=ALOGE,tag=MyTest
CAM0>> U:manualzslmode=1
CAM0>> P:1
CAM0>> s:1
CAM0>> D
CAM0>> Q
# exit
$ adb pull /data/misc/camera/”filename”  .
```

- Camera video recording with fps:30,encoder:h264, size:1920x1080,bitrate:16Mbit.
  - vsize: set video size.
  - codectype: select encoder(0:H264/avc, 1:H265/hevc)
  - fpsrange: set sensor fps. If you want a variable fps mode, you can use format like this fpsrange=min-max.
  - bitrate: set encoder output bitrate(Mb)

```
$ adb shell
# hal3_test
CAM0>>
A:id=2,psize=1920x1080,pformat=yuv420,vsize=1920x1080,ssize=3264x2448
,sformat=jpeg,fpsrange=30-30,codectype=0,bitrate=16
CAM0>> D
CAM0>> Q
```

- Camera video recording with fps:30,encoder:h265, size:1920x1080,bitrate:8Mbit

```
$ adb shell
# hal3_test
CAM0>>
A:id=2,psize=1920x1080,pformat=yuv420,vsize=1920x1080,ssize=3264x2448,sf
ormat=jpeg,fpsrange=30-30,codectype=1,bitrate=8
CAM0>> D
CAM0>> Q
```

- Camera video recording with MCTF on,

  The video file is saved on /data/misc/camera/ path and can get this file by adb pull cmd, this video can be play with various video player on PC that support H264 decoding.

```
$ adb root
$ adb remount
$ adb shell mount -o rw,remount /
$ adb shell "echo "advanceFeatureMask=0x01">
vendor/etc/camera/camxoverridesettings.txt"
$ adb shell "echo logVerboseMask=0xFFFFFFFF>>
/vendor/etc/camera/camxoverridesettings.txt"
$ adb shell "echo enableMCTF=TRUE>>
/vendor/etc/camera/camxoverridesettings.txt"

$ adb shell
# hal3_test
CAM0>>
A:id=2,psize=1920x1080,pformat=yuv420,vsize=1920x1080,ssize=3264x2448,sf
ormat=jpeg,fpsrange=30-30,codectype=0
CAM0>> P:1
CAM0>> D
CAM0>> Q
```

- Turn off MCTF:

```
$ adb shell "echo "advanceFeatureMask=0x01">
/vendor/etc/camera/camxoverridesettings.txt"
$ adb shell "echo logVerboseMask=0xFFFFFFFF>>
/vendor/etc/camera/camxoverridesettings.txt"
$ adb shell "echo "enableMCTF=False">>
/vendor/etc/camera/camxoverridesettings.txt"
$ adb reboot
$ adb shell
# hal3_test
CAM0>>
A:id=2,psize=1920x1080,pformat=yuv420,vsize=1920x1080,ssize=3264x2448,sf
ormat=jpeg,fpsrange=30-30,codectype=0
CAM0>> P:1
CAM0>> D
CAM0>> Q
```

- Camera video recording and live shot.

```
$ adb shell
# hal3_test
CAM0>>
A:id=2,psize=1920x1080,pformat=yuv420,ssize=3264x2448,sformat=jpeg,vsize
=1920x1080,codectype=0,fpsrange=30-30
CAM0>> P:1
CAM0>> s:1
CAM0>> D
CAM0>> Q
```

4.10.2 Tracking (OV7251)

- Preview_Camera_640*480 dump raw10

```
$ adb shell
# hal3_test
CAM0>> a:id=3,psize=640x480,pformat=raw10
CAM0>> P:1
CAM0>> D
CAM0>> Q
#exit
$ adb pull /data/misc/camera/”filename”  .
```

- Preview_Camera_640*480@100fps dump raw8

```
$ adb shell
echo "enableInternalHALPixelStreamConfig=TRUE" >>
/vendor/etc/camera/camxoverridesettings.txt
# hal3_test
CAM0>>   A:id=3,psize=640x480,pformat=raw8,fpsrange=100-100
CAM0>> U:frameduration=10000000,manual_exp=7785000,manual_iso=800
CAM0>> P:1
CAM0>> U:frameduration=10000000,manual_exp=4000000,manual_iso=800
CAM0>> P:1
CAM0>> U:frameduration=10000000,manual_exp=2000000,manual_iso=800
CAM0>> P:1
CAM0>> D
CAM0>> Q
#exit
$ adb pull /data/misc/camera/”filename”  .
```

4.10.3 Check raw picture with “imageJ”(https://imagej.nih.gov/ij/)tools.Stereo camera

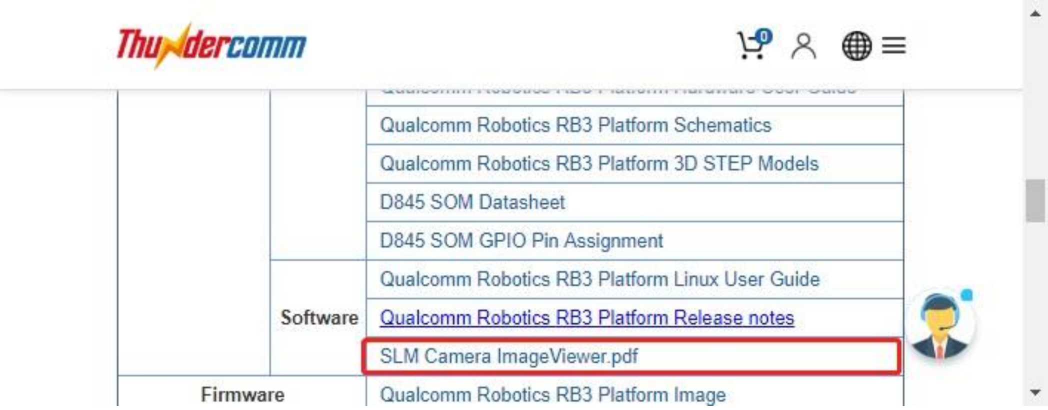
- Depth Camera_Preview_720P_ Dump RAW:

```
$ adb shell setprop persist.al.miniisp.camx.stop 1
$ adb shell setprop persist.al.camera.depth.dump 1
$ adb shell setprop persist.al.dump.count 5
$ adb shell setprop persist.al.scid 4

# hal3_test
CAM0>>
A:id=0,psize=1280x720,pformat=yuv420,ssize=1280x720,sformat=jpeg,altek=1
CAM0>> Quit
# exit
$ adb pull /data/misc/camera/”filename” .
```

- Use AltekImageViewer.exe tool to check raw files.

  Get AltekImageViewer.exeand refer to altek ImageViewer_1.30w-v01.pdf file for how to use AltekImage Viewer.

https://www.thundercomm.com/app_en/product/1544580412842651#doc



		Qualcomm Robotics RB3 Platform Hardware User Guide
		Qualcomm Robotics RB3 Platform Schematics
		Qualcomm Robotics RB3 Platform 3D STEP Models
		D845 SOM Datasheet
		D845 SOM GPIO Pin Assignment
	Software	Qualcomm Robotics RB3 Platform Linux User Guide
		Qualcomm Robotics RB3 Platform Release notes
		SLM Camera ImageViewer.pdf
Firmware		Qualcomm Robotics RB3 Platform Image

4.10.4 ToF Dump RAW

- ToF camera 640x480 dump raw 2vc iRGB+depth:

```
$ adb shell
# testapp
# 1             //Select image type
# 0             //Select synchronization method, 0:Internal  1: External
# 0             //Select mode, 0:near  1:far
# 10            //Select the number of frames saved
# 3             //Select ini_ofst_delay
# 0             //Select idle_delay
# ls /output/
bg_0001.raw  bg_0007.raw     depth_0003.raw  depth_0009.raw  ir_0005.raw
bg_0002.raw  bg_0008.raw     depth_0004.raw  depth_0010.raw  ir_0006.raw
bg_0003.raw  bg_0009.raw     depth_0005.raw  ir_0001.raw     ir_0007.raw
bg_0004.raw  bg_0010.raw     depth_0006.raw  ir_0002.raw     ir_0008.raw
bg_0005.raw  depth_0001.raw   depth_0007.raw  ir_0003.raw    ir_0009.raw
bg_0006.raw  depth_0002.raw   depth_0008.raw  ir_0004.raw    ir_0010.raw
$ adb pull /output/ .
```

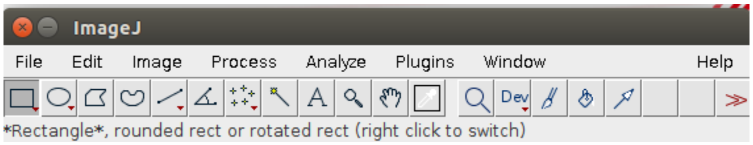
Use imageJ tool to check raw files.

Get imageJ at:

https://imagej.nih.gov/ij/

- Guide of imageJ:

1. From the File menu, select **Open**.



Rectangle, rounded rect or rotated rect (right click to switch)

2. Use the following configuration.

depth : 640 x 480.

ir or bg : 320 x 240.

image type : 16-bits unsigned.

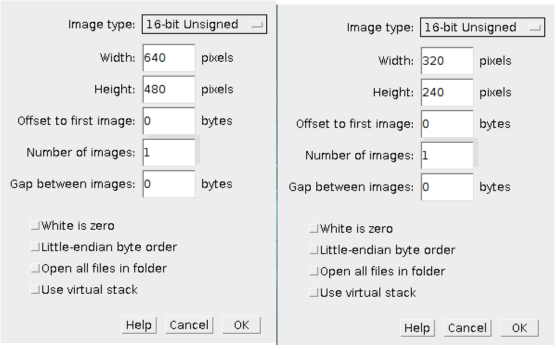
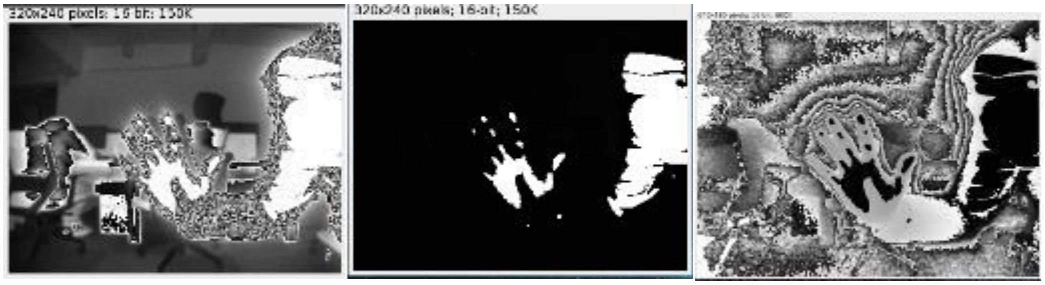


Image type: 16-bit Unsigned
Width: 640 pixels
Height: 480 pixels
Offset to first image: 0 bytes
Number of images: 1
Gap between images: 0 bytes
☐ White is zero
☐ Little-endian byte order
☐ Open all files in folder
☐ Use virtual stack

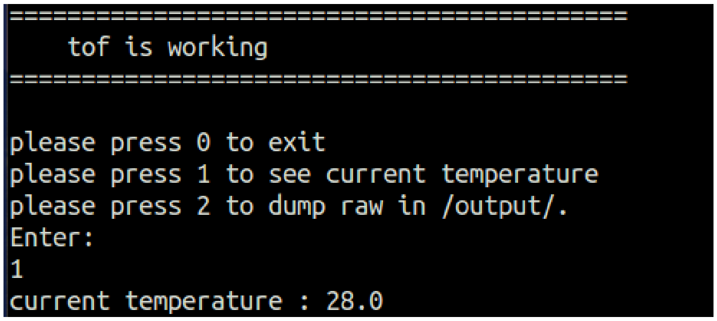
Image type: 16-bit Unsigned
Width: 320 pixels
Height: 240 pixels
Offset to first image: 0 bytes
Number of images: 1
Gap between images: 0 bytes
☐ White is zero
☐ Little-endian byte order
☐ Open all files in folder
☐ Use virtual stack

3. An image similar to the following will be displayed(ir + bg + depth):



- TOF camera temperature feature:

```
$ adb shell
# testapp
# 0            //Select mode, 0:near   1: far
# 1            //Display the current temperature of the TOF camera, as
shown below.
```



```
==========================================
    tof is working
==========================================

please press 0 to exit
please press 1 to see current temperature
please press 2 to dump raw in /output/.
Enter:
1
current temperature : 28.0
```

4.10.5 Three camera concurrency mode

Connect the camera sensors in the following table before running the test app. To enable each camera, seethe camera IDshown:

Camera module name	Stereo	ToF	OV8856	Tracking
Camera ID	0	1	2	3
Whether Depend on ISP	Yes	No	Yes	No

There are only two ISP interfaces on Robotics DK. As soon as ToF is enabledboth the ISPsare automatically selected for ToF consumption. During the concurrency mode, enable other camera sensors before enabling ToF/Tracking. This prevents the ToFcamera from occupying the ISP interface on the Robot DK.

- Enable preview of Main+Tracking+ToF:

```
adb shell
# hal3_test
A:id=2,psize=1920x1080,pformat=yuv420
P:1
A:id=3,psize=640x480,pformat=raw10
P:1
A:id=1,psize=640x480,pformat=raw16,dsize=640x480,dformat=raw16
P:1
D
Q
```

- Enable preview of Main+Stereo (depth)+ToF:

```
adb shell
#hal3_test
A:id=2,psize=1920x1080,pformat=yuv420
P:1
A:id=0,psize=1280x720,pformat=yuv420,ssize=1280x720,sformat=jpeg,altek=1
A:id=1,psize=640x480,pformat=raw16,dsize=640x480,dformat=raw16
P:1
D
Q
```

- Enable preview of Tracking+Stereo (depth)+ToF:

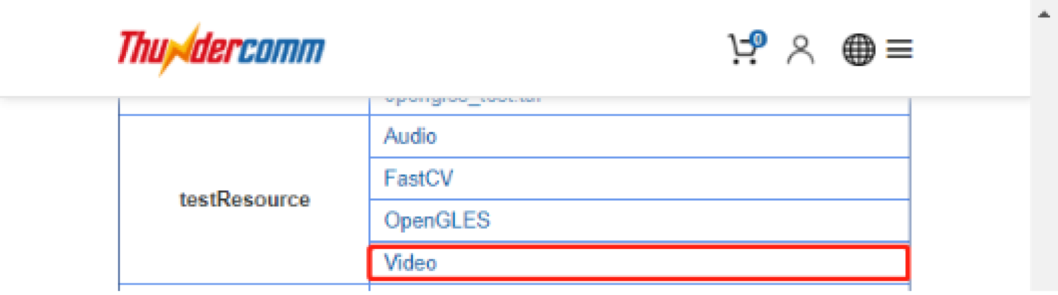
```
adb shell
#hal3_test
A:id=3,psize=640x480,pformat=raw10
P:1
A:id=0,psize=1280x720,pformat=yuv420,ssize=1280x720,sformat=jpeg,altek=1
A:id=1,psize=640x480,pformat=raw16,dsize=640x480,dformat=raw16
P:1
D
Q
```

NOTE: Choose only one resolution for each camera for example. More resolution for each camera please check the spec of camera module.

4.11 Video

Download the test file from:

https://www.thundercomm.com/app_en/product/1544580412842651#doc



testResource	Audio
	FastCV
	OpenGLES
	Video

- □ H264_3840_2160_60fps.264;
- □ H265_3840_2160_60fps.265
- □ 1920_1080_60fps.mp4;
- □ 1920_1080.yuv

4.11.1 OMX

- ■ Configure the macro environment, mainly used to set the input and output video path.
  - □ MasterConfg.xml:

```
<xml>
<InputFileRoot>/data/input/</InputFileRoot>
<OutputFileRoot>/data/output/</OutputFileRoot>
</xml>
```

  - □ SampleDecode.xml:

```
<xml>
<DecodeSession>
<TestCaseID>3840_2160</TestCaseID>
<CompressionFormat>VIDEO_CodingAVC</CompressionFormat>
<InputFile>H264_3840_2160_60fps</InputFile>
<session_mode>dec</session_mode>
<SinkType>FILE</SinkType>
<PlaybackMode>1</PlaybackMode>
<SourceHeight>2160</SourceHeight>
<SourceWidth>3840</SourceWidth>
<OutputHeight>2160</OutputHeight>
<OutputWidth>3840</OutputWidth>
</DecodeSession>
</xml>
```

- ■ Decode command:

```
$ adb shell mkdir /data/input /data/output
$ adb push MasterConfg.xml /data/
$ adb push H264 3840 2160 60fps /data/input/
$ adb push SampleDecode.xml /data/

$ adb shell
# mm-vidc-omx-test /data/ /data/SampleDecode.xml
Frame Num= 299 file_offset=0x0 frame_size=0x40f0 pBuffer=0xebac1000
fileoffset:0x4b909b status:0x0
Frame Num= 300 file_offset=0x0 frame_size=0x40f0 pBuffer=0xeb401000
fileoffset:0x4bd18b status:0x0
VT_CONSOLE RunTest::69 Test passed
# ls -al /data/output/
-rw-rw-rw-  1 root root 940032000 Dec 29 20:10 3840_2160.yuv
```

4.11.2 GST

- See Section 5.8.2 for the source location and compilation of the command
- Environmental configuration:

```
$ adb shell mkdir /data/input /data/output
$ adb push 1920_1080_60fps.mp4 /data/input
$ adb push 1920_1080.yuv /data/input
$ adb shell
# export GST_REGISTRY=/data/gstreamer-1.0/registry.$(uname -m).bin
# export GST_REGISTRY_UPDATE=no
# gst-inspect-1.0 > /dev/null
```

- Decoder/Encoder
  - □ Decoder:SW (Software Decoder):

```
# gst-launch-1.0 -e filesrc
location=/data/input/1920_1080_60fps.mp4 ! qtdemux name=demux
demux. ! queue ! h264parse ! avdec_h264 ! filesink
location=/data/output/1920_1080_60.yuv

Setting pipeline to PAUSED ...
Pipeline is PREROLLING ...
Redistribute latency...
Redistribute latency...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
Got EOS from element "pipeline0".
Execution ended after 0:00:08.570580309
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Freeing pipeline ...

#ls /data/output
1920_1080_60.yuv
```

□ Decoder:HW (Hardware Decoder):

```
# gst-launch-1.0 -e filesrc
location=/data/input/1920_1080_60fps.mp4 ! qtdemux name=demux
demux. ! queue ! h264parse ! omxh264dec ! filesink
location=/data/output/1920_1080_60hw.yuv

Setting pipeline to PAUSED ...
Pipeline is PREROLLING ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
Got EOS from element "pipeline0".
Execution ended after 0:00:08.476530986
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Freeing pipeline ...

#ls /data/output
1920_1080_60hw.yuv
```

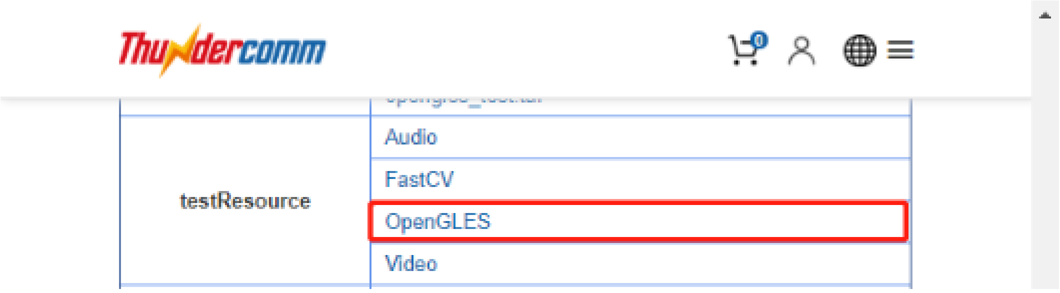
□ Encoder:HW (Hardware Encoder):

```
# gst-launch-1.0 -e filesrc location=/data/input/1920_1080.yuv !
videoparse width=1920 height=1080 format=nv12 framerate=60 !
omxh264enc target-bitrate= 800000000 quant-p-frames=59 quant-b-
frames=0 control-rate=variable ! 'video/x-
h264,streamformat=(string)byte-stream,profile=high' ! h264parse !
filesink location=/data/output/1920_1080_60.h264

Setting pipeline to PAUSED ...
Pipeline is PREROLLING ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
Got EOS from element "pipeline0".
Execution ended after 0:00:10.204220673
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Freeing pipeline ...

#ls /data/output
1920_1080_60.h264
```

4.12 OpenGL ES

1. Get the test file from:

   https://www.thundercomm.com/app_en/product/1544580412842651#doc



testResource	Audio
	FastCV
	OpenGLES
	Video

2. Extract the downloaded file:

```
$ unzip Robotics-opengles-testbin.zip
$ tree Robotics-opengles-testbin/
Robotics-opengles-testbin/
└── es11
    ├── es11_32
    │   ├── conform_cl
    │   ├── conform_cm
    │   ├── covegl
    │   ├── covgl_cl
    │   ├── covgl_cm
    │   ├── primtest_cl
    │   └── primtest_cm
    ├── es11_64
    │   ├── conform_cl
    │   ├── conform_cm
    │   ├── covegl
    │   ├── covgl_cl
    │   ├── covgl_cm
    │   ├── primtest_cl
    │   └── primtest_cm
└── push.sh
```

3. Upload test commands to the device:

```
$ adb root
$ adb remount
$ adb shell mount -o remount rw /
$ adb shell mount -o remount rw /data
$ adb shell mkdir -p /data/testApp/es11
$ adb push es11_64 /data/testApp/es11
$ adb shell chmod 777 /data/testApp/es11/*
```

4. OpenGL ES conformance test:

```
$ adb shell
# cd /data/testApp/es11
# ./conform_cl -h
Options:
    -1 <test>  Single test using "test" id.
    -c <id>         Use config id.
    -C [1-1]        Use predefined config.
    -f <file>  Use test set in "file".
    -g <file>  Generate test set in "file".
    -h         Print this help screen.
    -l <file>  Generate logfile.
    -p [1-4]   Set path level.
    -r <seed>  Set random seed.
    -s         Skip state check.
    -v [0-2]   Verbose level.
-x        Force fail for config tests.
```

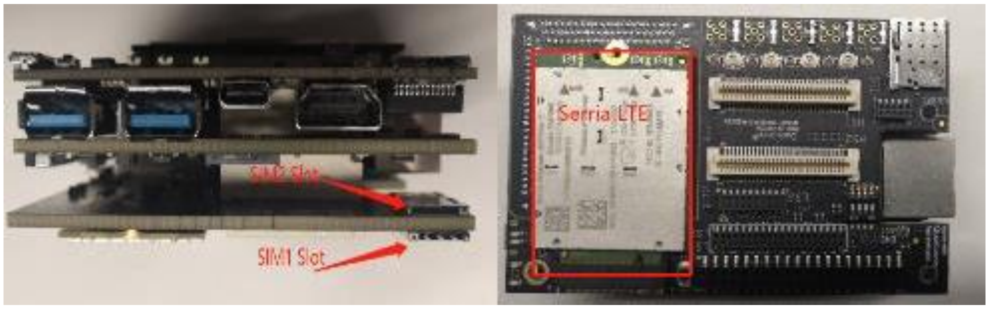
5. Execute the example as follows(testing process can take up to five hours):

```
# ./conform_cl -r 32555 -l mustpass.log
OpenGL ES Conformance Test
Version CONFORM_VERSION (CONFORM_DATE)

Setup Report.
    Verbose level = 1.
    Random number seed = 32555.
    Path inactive.

Config Report.
    Config ID = 1.
    RGBA (5, 6, 5, 0).

Default State test passed.
Must Pass test passed.
… …
Config Report.
    Config ID = 63.
    RGBA (8, 8, 8, 8).
    Stencil (8).
    Depth (24).

  Config is identified as NON_CONFORMANT.
  It may be rendered to using an OpenGL ES context.

******************** SUMMARY OF 'conform' RESULTS ********************
SUMMARY: NO tests failed at any path level.
SUMMARY: 48 conformant configs, 12 non-conformant configs.
****************** END SUMMARY OF 'conform' RESULTS ******************
```

4.13 LTEModule

NOTE: This section explains the how to validate the Serria LTE module on Qualcomm cellular mezzanine board.

1. Please look at the picture,it shows the Serria LTE module,SIM1 Slot,SIM2 Slot.



2. Select active SIM interface

   Before using SIM interface,you need to select it.

   □ For selectingfirst SIM interface:

```
# echo -e "AT!UIMS=0\r\n" > /dev/ttyUSB2
# cat /dev/ttyUSB2
AT!UIMS=0?
OK
```

   □ For selectingsecond SIM interface:

```
# echo -e "AT!UIMS=1\r\n" > /dev/ttyUSB2
# cat /dev/ttyUSB2
AT!UIMS=1?
OK
```

3. Check if the SIM card exists

   □ The following is the case offailure.

```
# echo -e "AT+CPIN?\r\n" > /dev/ttyUSB2
# cat /dev/ttyUSB2
+CME ERROR: SIM not inserted
```

   □ The following is the case of success.

```
# echo -e "AT+CPIN?\r\n" > /dev/ttyUSB2
# cat /dev/ttyUSB2
AT+CPIN?
+CPIN: READY
OK
```

Please refer GetWirelessLLC.com for more information about Sierra LTE module.We recommend looking into the Connection Manager Sample Application within the Sierra Linux QMI SDK.

4.14 Software tools and libraries

4.14.1 ROS

- ROS configuration :

```
# adb shell
#cd /opt/ros/indigo/
/opt/ros/indigo # bash
bash-4.4#source ./ros-env.sh
bash-4.4# roscore &
[1] 4065
bash-4.4# ... logging to /home/root/.ros/log/7ce256f6-2def-11e9-8312-
4962d813cee8/roslaunch-sda845-4065.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:46837/
ros_comm version 1.11.21


SUMMARY
========

PARAMETERS
 * /rosdistro: indigo
 * /rosversion: 1.11.21

NODES

auto-starting new master
process[master]: started with pid [4077]
ROS_MASTER_URI=http://localhost:11311/

setting /run_id to 7ce256f6-2def-11e9-8312-4962d813cee8
process[rosout-1]: started with pid [4090]
started core service [/rosout]
```

- Run hello_world:

```
bash-4.4# cd bin/
bash-4.4# ./hello_world
[ INFO] [1549016659.810443307]: hello_world
[ INFO] [1549016660.813989869]: hello world
```

  Log painting "[INFO] [TimeTag]: hello_world" and there should be no failure to return.

- Please refer following link for more details:

  https://github.com/ander-ansuategi/ros-hello-world/archive/master.tar.gz

4.14.2 Qualcomm Hexagon Vector eXtensions(Qualcomm® Hexagon™ Vector eXtensions(HVX))

Hexagon DSP SDK supports the RB3 Platform. For details see:

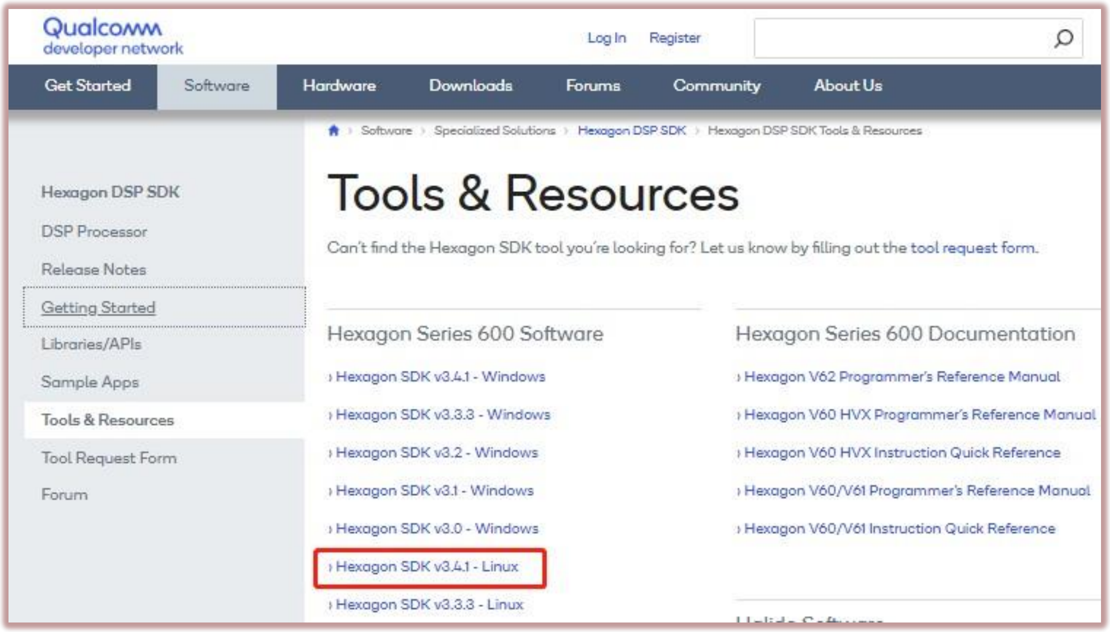
https://developer.qualcomm.com/software/hexagon-dsp-sdk

This section provides a step-by-step guide to build, load, and execute the calculator example on Robotics in an LU environment.

4.14.2.1 Hexagon SDK environment construction

1. Hexagon SDK download:

   https://developer.qualcomm.com/software/hexagon-dsp-sdk/tools



Tools & Resources

Can't find the Hexagon SDK tool you're looking for? Let us know by filling out the tool request form.

Hexagon Series 600 Software

- › Hexagon SDK v3.4.1 - Windows
- › Hexagon SDK v3.3.3 - Windows
- › Hexagon SDK v3.2 - Windows
- › Hexagon SDK v3.1 - Windows
- › Hexagon SDK v3.0 - Windows
- › Hexagon SDK v3.4.1 - Linux
- › Hexagon SDK v3.3.3 - Linux

Hexagon Series 600 Documentation

- › Hexagon V62 Programmer's Reference Manual
- › Hexagon V60 HVX Programmer's Reference Manual
- › Hexagon V60 HVX Instruction Quick Reference
- › Hexagon V60/V61 Programmer's Reference Manual
- › Hexagon V60/V61 Instruction Quick Reference

   a. The installer takes care of downloading and installing all Hexagon SDK dependencies.

   b. If you want, you can install at your desired location eg: /local/mnt/RB3/Qualcomm/Hexagon_SDK

   c. To get started with the Hexagon SDK, open a new terminal and run setup_sdk_env.source.

      This script configures the local environment. These changes are not persistent in the terminal instance, so you must run setup_sdk_env.source on each terminal you want to develop in.

```
$ cd < Hexagon SDK root directory, default is:
~/Qualcomm/Hexagon_SDK/<version>>
$ source setup sdk env.source
```

   d. Verify env path setting : echo $HEXAGON TOOLS_ROOT

2. Hexagon tool (Hexagon.LLVM_linux_installer_8.1.05.bin) needs to go to the website below to apply. https://developer.Qualcomm.com/software/hexagon-dsp-sdk/tool-request

When Qualcomm approves your application, you will receive an email with a download link. Download the tool and install it.

a. During installation you can opt for default location or change it according to your desired path eg: /local/mnt/RB3/Qualcomm/HEXAGON_Tools

b. If the tools are installed anywhere but the default location, then set the environment variable “HEXAGON_TOOLS_ROOT” to point to the installed location

On Linux:

```
$export HEXAGON_TOOLS_ROOT=/local/mnt/RB3/Qualcomm/HEXAGON_Tools
RB3@rb3-linux:/local/mnt/workspace/RB3/Qualcomm$ ls -al
total 16
drwxrwxr-x 4 rb3-linux users 4096 Feb 13 17:29 .
drwxr-xr-x 5 rb3-linux users 4096 Feb 13 16:52 ..
drwxrwxr-x 3 rb3-linux users 4096 Feb 13 16:32 Hexagon_SDK
drwxr-xr-x 6 rb3-linux users 4096 Feb 13 17:29 HEXAGON_Tools
```

3. Linux cross-compilation tool

This version of Hexagon SDK supports Android and certain versions of Linux distributions (Yocto and Linaro). Android is supported by variants starting with Android (e.g., Android_Debug). Linux is supported by variants starting with LU (e.g., LU_Debug). The binaries (executables and libs) are provided for both HLOS.

The cross compilation tools for Linux are not provided with the Hexagon SDK. You need to download these separately and install them under <SDK_ROOT>/tools/linaro. If you do this, then you will be able to build LU variants for the examples. The examples are tested with gcc-linaro-4.9.

- Steps to build 64 bit LU binaries on Linux:

a. Download gcc-linaro-4.9-2014.11-x86_64_aarch64-linux-gnu.tar.xz from:

http://releases.linaro.org/archive/14.11/components/toolchain/binaries/aarch64-linux-gnu

b. Extract the tar file and copy folder gcc-linaro-4.9-2014.11-x86_64_aarch64-linux-gnu to <Hexagon_SDK_ROOT>/tools/ folder.Rename gcc-linaro-4.9-2014.11-x86_64_aarch64-linux-gnu folder to linaro.

- Steps to build 32 bit LU binaries on Linuxs:

a. Download gcc-linaro-4.9-2014.11-x86_64_arm-linux-gnueabi.tar.xz from:

http://releases.linaro.org/archive/14.11/components/toolchain/binaries/arm-linux-gnueabi

b. Extract the tar file and copy folder gcc-linaro-4.9-2014.11-x86_64_arm-linux-gnueabi to <Hexagon_SDK_ROOT>/tools/ folder.Rename gcc-linaro-4.9-2014.11-x86_64_arm-linux-gnueabi folder to linaro .

4. Compiling example code:

When building the calculator example, both the stub and skeleton must be compiled and linked. This can be done by compiling both for the variant desired on the DSP as well as the application processor.

For example, to create a stub/skel pair for Linux and Hexagon, the following commands must be executed:

```
$ cd ~/Qualcomm/Hexagon_SDK/3.4.1
$ source setup_sdk_env.source
$ cd examples/common/calculator
$ make tree V=hexagon_Debug_dynamic_toolv81_v65 CDSP_FLAG=1
$ make tree V=UbuntuARM_Debug_aarch64CDSP_FLAG=1
```

Compiled file:

```
$ tree UbuntuARM_Debug_aarch64/ship/
UbuntuARM_Debug_aarch64/ship/
├── calculator
├── calculator_test.so
└── libcalculator.so
$ tree hexagon_Debug_dynamic_toolv81_v65/ship/
hexagon_Debug_dynamic_toolv81_v65/ship/
├── libcalculator_skel.a
└── libcalculator skel.so
```

NOTE: Please refer to help file “Qualcomm/Hexagon_SDK/3.4.1/docs/Dependencies_Common.html” in case you face compilation issue

4.14.2.2 On-target testing

- To execute the calculator test on Robotics perform the following steps:

```
$ cd ~/Qualcomm/Hexagon_SDK/3.4.1/examples/common/calculator
$ adb push hexagon_Debug_dynamic_toolv81_v65/ship/libcalculator_skel.so
/usr/lib/rfsa/adsp/
$ adb push UbuntuARM_Debug_aarch64/ship/libcalculator.so /usr/lib64/
$ adb push UbuntuARM Debug aarch64/ship/calculator /usr/bin/
```

- To install the TestSig on device:

1. First discover the device serial number. The following steps print out the device serial number.

```
$ cd ~/Qualcomm/Hexagon_SDK/3.4.1
$ adb push
tools/elfsigner/getserial/UbuntuARM_Release_aarch64/getserial
/usr/bin/
$ adb shell /usr/bin/getserial
/usr/bin/getserial: /usr/lib64/libadsprpc.so: no version information
available (required by /usr/bin/getserial)
###################Serial number (see below)#########################

    Serial Num : 0x2f5800d6

###################Serial number (see above)#########################
```

- Next generate a test signature based on that serial number and push it to the device. The test signature is discovered on boot so a reboot is required.

```
$ cd ~/Qualcomm/Hexagon_SDK/3.4.1
$ source setup_sdk_env.source
$ python tools/elfsigner/elfsigner.py -t 0x2f5800d6
Logging to ~/Qualcomm/Hexagon_SDK/3.4.1/output/Elfsigner_log.txt
```

```
    Attention:
    Use of this tool is conditioned upon your compliance with
QualcommTechnologies'(and its affiliates') license terms and
conditions;including, without limitations, such terms and
conditions addressingthe use of such tools with open source
software.

    Agree? [y/n]:
    y
Signing a file may take up to 3 minutes due to network connectivity.
Please wait patiently.
------------------------------------------------------------------------------------------------------------------------------------------
Signing complete! Output saved at
/home/lizc/Qualcomm/Hexagon_SDK/3.4.1/output/testsig-0x2f5800d6.so

$ adb push output/testsig-0x2f5800d6.so /usr/lib/rfsa/adsp/
$ adb reboot
```

- Execute the example as follows:

```
$ adb shell
# calculator 1 1000       // Run Calculator Example Locally on CPU

- starting calculator test
- allocate 4000 bytes from ION heap
- creating sequence of numbers from 0 to 999
- compute sum locally
- sum = 499500
- success

# calculator 0 1000       //Run Calculator Example on DSP

- starting calculator test
- allocate 4000 bytes from ION heap
- creating sequence of numbers from 0 to 999
- compute sum on the DSP
- sum = 499500
- success
```

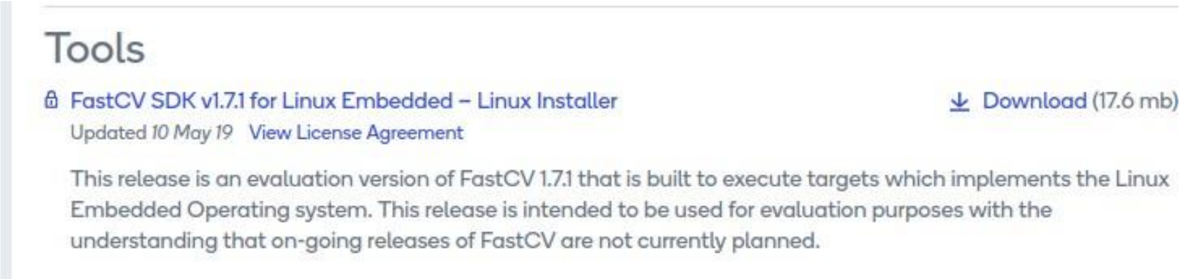
4.14.3 FastCV™

FastCV is designed for efficiency on all ARM-based processors, but is tuned to take advantage of the Qualcomm® Snapdragon™ processor (S2 and above). This gives you the most widely used, computationally intensive vision processing APIs, with hardware acceleration and better performance on mobile devices.

4.14.3.1 FastCV SDK environment construction

1. FastCV SDK download

https://developer.qualcomm.com/software/fast-cv-sdk/tools



Tools

FastCV SDK v1.7.1 for Linux Embedded – Linux Installer

Updated 10 May 19 View License Agreement

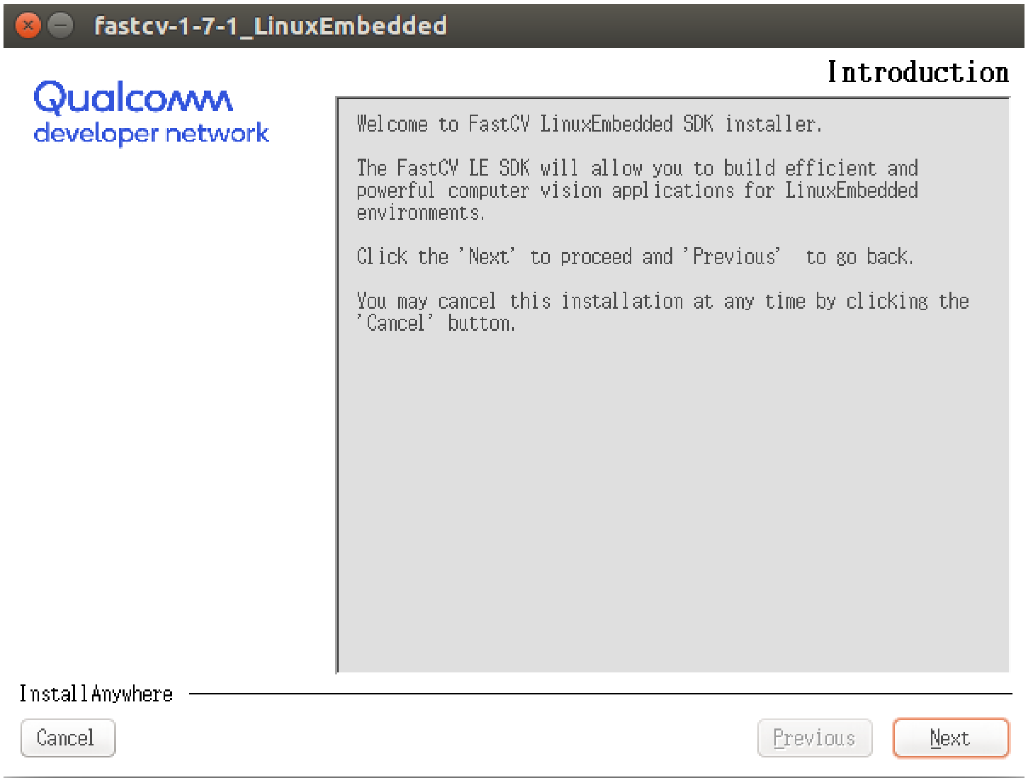
This release is an evaluation version of FastCV 1.7.1 that is built to execute targets which implements the Linux Embedded Operating system. This release is intended to be used for evaluation purposes with the understanding that on-going releases of FastCV are not currently planned.

You could download v1.7.1 for Linux Embedded.

2. FastCV installation and compilation

The FastCV binary file is fastcv-installer-linuxembedded-1-7-1.bin. You can install it.

```
chmod 755 fastcv-installer-linuxembedded-1-7-1.bin
./fastcv-installer-linuxembedded-1-7-1.bin
```



Welcome to FastCV LinuxEmbedded SDK installer.

The FastCV LE SDK will allow you to build efficient and powerful computer vision applications for LinuxEmbedded environments.

Click the 'Next' to proceed and 'Previous' to go back.

You may cancel this installation at any time by clicking the 'Cancel' button.

You could refer to the README.txt in <FastCV_SDK_ROOT>/samples/README.txt

Copy fastcvSimpleTest folder and libfastcv.a to <Hexagon_SDK_ROOT>/...

Run the setup_sdk_env.cmd as mentioned in <Hexagon_SDK_ROOT>\docs\readme.html.

```
cd examples/common/fastcvSimpleTest
make tree V=UbuntuARM Debug aarch64
```

4.14.3.2 FastCV SDK Test

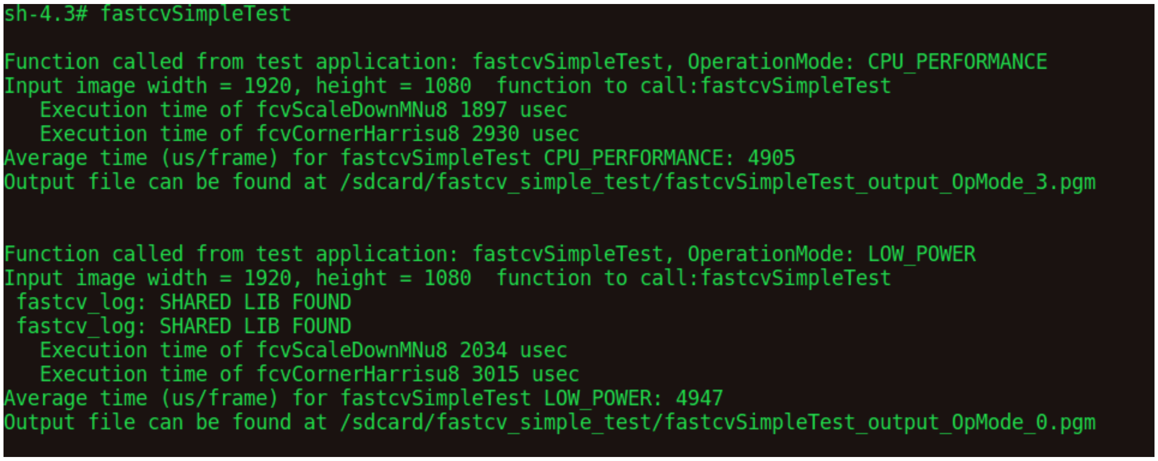
Push the fastcvSimpleTest binary file in the target:

```
adb push
<Hexagon_SDK_ROOT>\examples\common\fastcvSimpleTest\UbuntuARM_Debug_aarc
h64\ship\fastcvSimpleTest /usr/bin/
```

Change bin permissions and execute the fastcvSimpleTest:

```
adb shell chmod 777 /usr/bin/fastcvSimpleTest
adb shell /usr/bin/fastcvSimpleTest
```

Test result

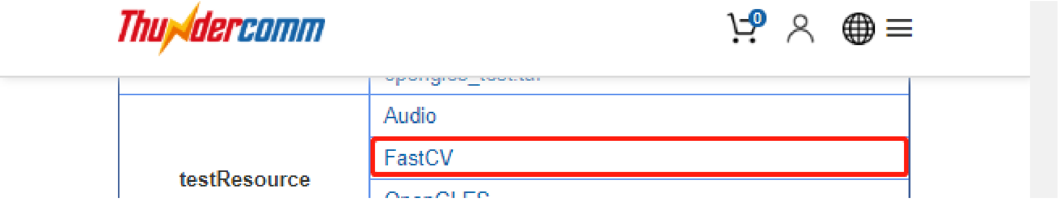


```
sh-4.3# fastcvSimpleTest

Function called from test application: fastcvSimpleTest, OperationMode: CPU_PERFORMANCE
Input image width = 1920, height = 1080  function to call:fastcvSimpleTest
   Execution time of fcvScaleDownMNu8 1897 usec
   Execution time of fcvCornerHarrisu8 2930 usec
Average time (us/frame) for fastcvSimpleTest CPU_PERFORMANCE: 4905
Output file can be found at /sdcard/fastcv_simple_test/fastcvSimpleTest_output_OpMode_3.pgm


Function called from test application: fastcvSimpleTest, OperationMode: LOW_POWER
Input image width = 1920, height = 1080  function to call:fastcvSimpleTest
 fastcv_log: SHARED LIB FOUND
 fastcv_log: SHARED LIB FOUND
   Execution time of fcvScaleDownMNu8 2034 usec
   Execution time of fcvCornerHarrisu8 3015 usec
Average time (us/frame) for fastcvSimpleTest LOW_POWER: 4947
Output file can be found at /sdcard/fastcv_simple_test/fastcvSimpleTest_output_OpMode_0.pgm
```

4.14.3.3 FastCV Other Test

1. Download “FastCV_test_data” test resources from:

   https://www.thundercomm.com/app_en/product/1544580412842651#doc



2. Push the test resource to the device:

```
$ adb push fastcv_test_data /data/
```

3. The test commands are divided into 64bit and 32bit. The following is introduced with 64bit:

```
$ adb shell
# lib64_fastcv_test
USAGE: lib64_fastcv_test test_data_directory [-l loops] [-m module_name]
[-p power_level][-E] [-S resolution#] [-f func_name] [-t target] [-M
operation_mode] [-OPT] [-s see] [-nbp] [-o] [-U] [-P]


       OPTIONS
       ------------------------------------------------------------------------------------------------------------------------------------------------------------
       -nbp
       Uses mallocs instead of internal buffer pool for
scratch/temporary buffers.

       -psb nPreAllocBytes
       Enable Preallocate Scratch Buffers
       nPreAllocBytes is the number of bytes for the pre-allocated
buffer.

       -l +integer
       0 = [default] do not profile
       +integer = loops to use when profiling.

       -m string
       String name of module to limit to.  Valid strings in order are:
       HW, MEM, DEPTH, DOT, SSD, IIMG, IIMGYUV, IDIFF, TRNS, WARP,
3CHANNELWARP,COLORYUVRGB, COLORYUV, COLORRGB, SCALE, BLUR, EDGES,
SCHARR, SAD, FAST10, DESCRIPTOR,, THRESH, COPY, VEC, KMEANS, AFFINEEVAL,
AFFINEFIT, HOMOGRAPHYEVAL, HOMOGRAPHYFIT, POSEEVAL, POSEREFINE,
3POINTPOSEEST, KDTREE, LINEARSEARCH, BITCNT, BITWISEOP, OFBM, BOUNDRECT,
UPSAMPLE, IPP, IPPTRANSFORM, CONTOUR, SOLVE, PERSPTRANSFORM, SET,
ADAPTTHRESH, SFGMASK, ABSDIFF, QUAD, AVERAGE, SHIFT, FLOODFILL, MOTION,
SVD, POLYGON, BGCODEBOOK, DRAWCONTOUR, HOUGHCIRCLE, HOUGHLINE, CALIBRATE,
REMAP, PYRAMID, IMGSEGMENTATION, LBP, FFT, CORNERSUBPIX, CHANNEL, STATS,
NCC, FIR, FAST, IMGINTENSITY, EDGE, KLT, MINMAX, KMEANSTREESEARCH,
SMOOTH, ARITHM, SVM, HARRIS, MSER

       -f string
       String name of function to limit to.  Make sure to specify
corresponding module using -m. Valid strings are:
       fcvNCCPatchOnCircle8x8u8, fcvNCCPatchOnSquare8x8u8,
       … …
       fcvGLBPu8, fcvFFTu8, fcvIFFTf32, fcvCornerRefineSubPixu8
       Note: Not all modules have function limit functionality yet.
       Please use this option with module limit option to limit to
       a particular function.

       -e string
       String name of exhaustive test data directory.
       -p integer
       Power mode to run QDSP in.  Valid values are:
       0 = minimum power mode
       1 = normal power mode
       2 = [default] maximum power mode

       -t integer
       Integer value indicating target.  Valid values are:
```

```
       FASTCV_ALL_TARGETS = 0; FASTCV_UNIT_ARM  = 2; FASTCV_UNIT_VENUM =
4;
       FASTCV_UNIT_QDSP = 8; FASTCV_UNIT_FPGA = 16; FASTCV_UNIT_GPU = 32
       FASTCV_UNIT_C2D = 64; FASTCV_UNIT_VFP = 128; FASTCV_UNIT_ARMv7 =
256
       FASTCV_UNIT_DMA = 512; FASTCV_UNIT_QDSP_TEST = 1024;

       -M integer
       Integer value indicating operation mode.  Valid values are:
       Skip operation mode test = 0
       FASTCV_OP_LOW_POWER = 1;  FASTCV_OP_PERFORMANCE = 2;
FASTCV_OP_CPU_OFFLOAD = 4
       Combination  FASTCV_OP_LOW_POWER & FASTCV_OP_PERFORMANCE = 3
       Combination  FASTCV_OP_LOW_POWER & FASTCV_OP_CPU_OFFLOAD = 5
       Combination  FASTCV_OP_PERFORMANCE & FASTCV_OP_CPU_OFFLOAD = 6
       All Combination  FASTCV_OP_LOW_POWER & FASTCV_OP_PERFORMANCE &
FASTCV_OP_CPU_OFFLOAD = 7

       -OPT
       Use this option with [-M operation_mode] to skip unit testing on
individual processing units and test only operation tables

       -s seed
       Seed for random number generator.

       -o
       FLAG to enable opencv benchmark profiling Results

       -L +integer
       Tells the number of iterations for profiling a function on QDSP
with only single remoting overhead

       -U
        Enables only the Unit Tests while disabling the Performance
Tests

       -P
       Enables only profiling while disabling the unit Tests

       -DF
       Disables fuzzing check

       -E
       Enables exhaustive testing to validate profiling vectors

       -S Resolution#
       Resolution for profiling.. Deafult is VGA
       -H
       Enables QDSP test vectors to be allocated on ARM Heap instead of
ION

       -AC
       Bump up ARM clocks

       -AL
       ION allocation buffers are aligned to no more than element size.
No effect on heap allocated buffers.

       -TWOp
       Test a couple fastcv API without calling any fcvSetOperationMode
to run default C reference code.

       NOTE: Fast running functions will automatically have a loop
multiplier and the results will be normalized accordingly.
```

4. Execute the example as follows:

```
$ adb shell
# lib64_fastcv_test /data/fastcv_test_data/ -t 0 -l 1 -L 1 -m HW
FASTCV_PROFILE, CLOCKS: ARM cpu0: 1228800, cpu1: 1516800, cpu2: 1766400,
cpu3: 1766400
FASTCV_PROFILE,             cpu4: 825600, cpu5: 825600, cpu6: 825600,
cpu7: 825600
FASTCV_PROFILE,         QDSP clk : 939205026, bimc clk: 0, snoc clk: 0
FASTCV_PROFILE, PROFILING ORDER : REF, Venum, QDSP, QDSPw/oRem, GPU,
LowPower, Performance, CPUOffload, CPUPerformance,
FASTCV_PROFILE, FIT:(FeatureName=>FASTCV, Overall=>PASS)
```

4.14.4 Snapdragon$^{\text{TM}}$ Neural Processing Engine(SNPE)

The Snapdragon Neural Processing Engine (SNPE) is a Qualcomm Snapdragon software accelerated runtime for the execution of deep neural networks.

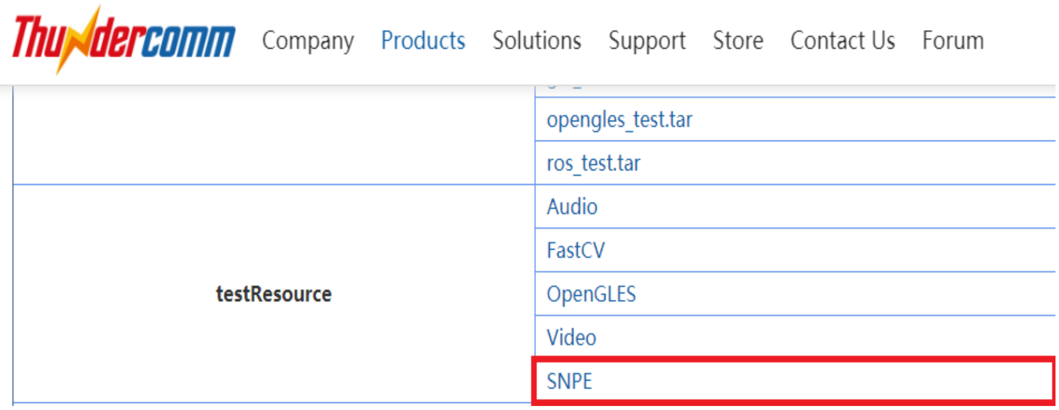
This section will tell you if snpe is available in your device.

When you build your deep neural networks, you can refer to https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk.

Notes: The following test resource is based on the validation results from QDN(Qualcomm Developer Network) SNPE User Guide.

1. Download snpe test resources from:

   https://www.thundercomm.com/app_en/product/1544580412842651#doc



	opengles_test.tar
	ros_test.tar
testResource	Audio
	FastCV
	OpenGLES
	Video
	SNPE

2. Extract the downloaded file:

```
$tar xzvf snpe_test.tar.gz
```

3. Push the test resource to the device:

```
$adb push snpebm /home/
```

4. Test command for cpu 64bit

```
$adb shell
#/home/snpebm/artifacts/aarch64-linux-gcc4.9/bin/snpe-bench_cmds_cpu.sh
```

- The correct log is as follows:

```
------------------------------------------------------------------------------------------------
Model String: N/A
SNPE v1.24.0.256
------------------------------------------------------------------------------------------------
Processing DNN input(s):
cropped/handicap_sign.raw
Processing DNN input(s):
cropped/trash_bin.raw
Processing DNN input(s):
cropped/chairs.raw
Processing DNN input(s):
cropped/notice_sign.raw
Processing DNN input(s):
cropped/plastic_cup.raw
```

5. Test command for gpu 64bit

```
$adb shell
#/home/snpebm/artifacts/aarch64-linux-gcc4.9/bin/snpe-bench_cmds_gpu.sh
```

- The correct log reference 4.

6. Test command for dsp 64bit

```
$adb shell
#/home/snpebm/artifacts/aarch64-linux-gcc4.9/bin/snpe-bench_cmds_dsp.sh
```

- The correct log reference 4.

7. Test command for cpu 32bit

```
$adb shell
```

```
#/home/snpebm/artifacts/arm-linux-gcc4.9sf/bin/snpe-bench_cmds_cpu.sh
```

- The correct log reference 4.

8. Test command for gpu 32bit

```
$adb shell
#/home/snpebm/artifacts/arm-linux-gcc4.9sf/bin/snpe-bench_cmds_gpu.sh
```

- The correct log reference 4.

9. Test command for dsp 32bit

```
$adb shell
#/home/snpebm/artifacts/arm-linux-gcc4.9sf/bin/snpe-bench_cmds_dsp.sh
```

- The correct log reference 4.

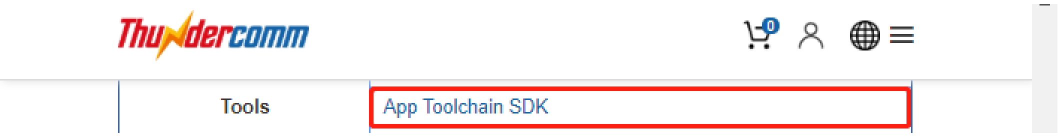
5 Develop an application

This section explains how to compile the test commands used in the Section 4.

5.1 Install the Application SDK

1. Download the Application SDK:

   https://www.thundercomm.com/app_en/product/1544580412842651#doc



2. Change to the sdk directory:

```
$ cd ROBOTICS-SDA845-LE-APP-SDK
$ ls
oecore-x86_64-aarch64-toolchain-nodistro.0.sh
```

3. Execute the oecore-x86_64-aarch64-toolchain-nodistro.0.sh command:

```
$ ./oecore-x86_ 64-aarch64-toolchain-nodistro.0.sh
```

4. To choose the default target directory, pressEnter and type Y.

```
robot SDK installer version nodistro.0
======================================
Enter target directory for SDK (default: /usr/local/oecore-x86_64):
You are about to install the SDK to "/usr/local/oecore-x86_64".
Proceed[Y/n]? Y
Extracting SDK.............................................done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to
source the environment setup script e.g.
 $  .  /usr/local/oecore-x86_64/environment-setup-aarch64-oe-linux
 $ . /usr/local/oecore-x86_64/environment-setup-armv7a-neon-oemllib32-
linux-gnueabi
```

5. Configure the system environment:

```
$  source  /usr/local/oecore-x86_64/environment-setup-aarch64-oe-linux
```

6. Compile the command:

```
$ aarch64-oe-linux-gcc --sysroot=/usr/local/oecore-
x86_64/sysroots/aarch64-oe-linux -O2 -fexpensive-optimizations -frename-
registers -fomit-frame-pointer -Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-
needed  test.c -o test
```

5.2 Hello RB3

Once the Application SDK is installed, the first RB3 Hello application can be installed..

1. Create a source file and edit:

```
$ vi Hello.c
#include <stdio.h>
#include <stdlib.h>
int main(void){
       printf("Hello RB3 !!!\n");
       return 0;
}
```

2. Build and transfer the application

   a. Build the application:

```
$  source  /usr/local/oecore-x86_64/environment-setup-aarch64-oe-linux
$ aarch64-oe-linux-gcc --sysroot=/usr/local/oecore-
x86_64/sysroots/aarch64-oe-linux -O2 -fexpensive-optimizations -
frename-registers -fomit-frame-pointer  -Wl,-O1 -Wl,--hash-style=gnu
-Wl,--as-needed  Hello.c   -o Hello
```

   b. Connect to the PC via TYPE-C and ensure that the ADB port can be used. Transfer the application:

```
$ adb push Hello /bin/
```

3. Execute the application:

```
$ adb shell
# chmod u+x /bin/Hello
# Hello
```

4. The terminal outputs the expected print information:

```
/ # Hello
Hello RB3!!!
```

5.3 Button Samples app

This example shows how to get the key value of a button through the corresponding file node.

1. Create a source file and edit:

```
$ vi button_test.c
#include <stdio.h>
#include <unistd.h>
#include <linux/input.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#define DEV_PATH0 "/dev/input/event0"    //VOL-, POWER
```

```
int main(int argc, char **argv)
{
    int i = 0;
    int times = 20; //defalt time
    int keys_fd = 0;
    struct input_event t;

    if (argc == 2) {
        times = (int)strtoul(argv[1], NULL, 10);
    }

    keys_fd=open(DEV_PATH0, O_RDONLY | O_NDELAY);
    if(keys_fd <= 0){
        printf("open /dev/input/event0 device error!\n");
        return -1;
    }

    for(i=0;i<times*10;i++)
    {
        if(read(keys_fd, &t, sizeof(t)) ==
            sizeof(t)){ if(t.type == EV_KEY && t.value ==
            1)
                printf("Event0: type:%d, value:%d, code:%d\n",
                        t.type, t.value, t.code);
        }
        usleep(100*1000);
    }
    close(keys_fd);

    return 0;
```

2. Build and transfer the application

   a. Edit the Makefile:

```
OS.exec = uname -s
OS ?= $(shell $(OS.exec))$(OS.exec:sh)
OS := $(OS)

PROG = button_test

CFLAGS ?= -O2 -fexpensive-optimizations -frename-registers -fomit-
frame-pointer
CFLAGS += -Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-needed

# Common library includes
LDLIBS = -lsensors.ssc -lpthread

$(PROG): $(PROG).o
```

```
$(PROG).o: $(PROG).c $(HDR)

clean:
  $(RM) $(PROG).o $(PROG)

sparse: $(PROG).c
  $(SPARSE) $(CPPFLAGS) $(CFLAGS) $(SPARSEFLAGS) $^

.PHONY: clean
```

b. Build the application:

```
$  source  /usr/local/oecore-x86_64/environment-setup-aarch64-oe-linux
$ make
```

c. Connect the board to the host PC via Type C and ensure that the ADB port can be used. Transfer the application:

```
$ adb push button_test /bin/
```

3. Execute the application:

```
$ adb shell
# button_test
```

4. The terminal outputs the expected print information:

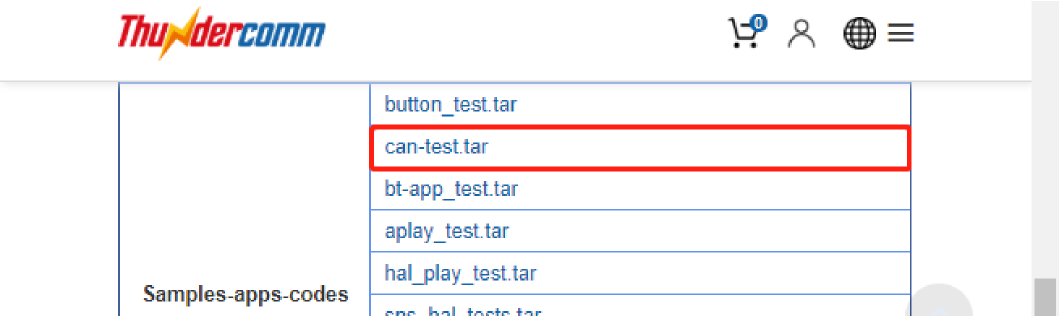
```
Event0: type:1, value:1, code:114
Event0: type:1, value:1, code:114
```

5.4 CAN

1. Download source code

   As mentioned in Secton 4.4, the source code for testing CAN can be downloaded here.

   https://www.thundercomm.com/app_en/product/1544580412842651#doc



The directory after decompression is as follows:

```
$ tree can-test/ -L 1
can-test/
├── can-test-build.sh
├── canutils-4.0.6
├── iproute2-4.9.0
└── libsocketcan-0.0.11
```

2. Build and transfer the application

   a. The compiled script has been integrated into the code.

```
$ cat can-test/can-test-build.sh
#!/bin/bash
set -e

source /usr/local/oecore-x86_64/environment-setup-aarch64-oe-linux

WORKSPACE=`pwd`

mkdir ${WORKSPACE}/libs
mkdir ${WORKSPACE}/bin

#build libsocketcan-0.0.11
cd  ${WORKSPACE}/libsocketcan-0.0.11/
./configure --host=aarch64-oe-linux-gnueabi --
prefix=${WORKSPACE}/libs
make
make install

#build canutils-4.0.6
cd ${WORKSPACE}/canutils-4.0.6
 ./configure --host=arm-linux libsocketcan_LIBS="-
L${WORKSPACE}/libs/lib -lsocketcan" libsocketcan_CFLAGS="-
I${WORKSPACE}/libs/include" CFLAGS="-I${WORKSPACE}/libs/include"
make

#build iproute2-4.9.0
cd ${WORKSPACE}/iproute2-4.9.0
./configure
make

cp -rf ${WORKSPACE}/canutils-4.0.6/src/canconfig ${WORKSPACE}/bin
cp -rf ${WORKSPACE}/canutils-4.0.6/src/candump ${WORKSPACE}/bin
cp -rf ${WORKSPACE}/canutils-4.0.6/src/canecho ${WORKSPACE}/bin
cp -rf ${WORKSPACE}/canutils-4.0.6/src/cansequence ${WORKSPACE}/bin
cp -rf ${WORKSPACE}/iproute2-4.9.0/ip/ip ${WORKSPACE}/bin

echo
echo "Done ..."
```

   b. Build the application:

```
$ ./can-test-build.sh
```

   c. Connect the board to the host PC via Type C and ensure that the ADB port can be used. Transfer the application:

```
$ adb push bin/ /usr/bin/
```

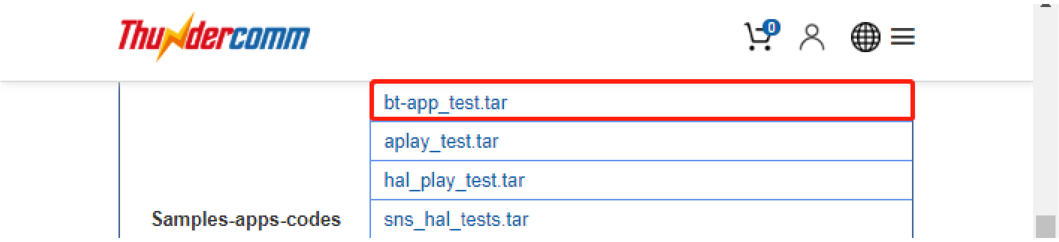
See Section 4.4 for how to use CAN command.

5.5 Bluetooth

1. Download source code

   As mentioned in Secton 4.5.2, the source code for testing bluetooth can be downloadedhere.

   https://www.thundercomm.com/app_en/product/1544580412842651#doc



	bt-app_test.tar
	aplay_test.tar
	hal_play_test.tar
Samples-apps-codes	sns_hal_tests.tar

   The directory after decompression is as follows:

```
$ tree btapp-samplecode -L 2
btapp-samplecode
├── bt
│   ├── bt-app
│   ├── gatt
│   ├── obex_profiles
│   └── README
├── hardware
│   └── qcom
├── include
│   └── hardware
├── lib
│   ├── libbtobex.a
│   └── libgengatt.a
└── vendor
└── qcom
```

2. Build and transfer the application

   a. The compiled script has been integrated into the code.

```
$ cat btapp-samplecode/bt/bt-app/configure.sh
#!/bin/sh

aclocal
libtoolize --copy --force

autoconf
autoheader
automake --add-missing

./configure  --build=x86_64-linux --host=arm-oemllib32-linux-gnueabi
--target=arm-oemllib32-linux-gnueabi --prefix=/usr    --
exec_prefix=/usr --bindir=/usr/bin --sbindir=/usr/sbin --
libexecdir=/usr/libexec --datadir=/usr/share --sysconfdir=/etc --
sharedstatedir=/com --localstatedir=/var --libdir=/usr/lib --
includedir=/usr/include --oldincludedir=/usr/include --
```

```
infodir=/usr/share/info --mandir=/usr/share/man   --disable-silent-
rules --disable-dependency-tracking --with-libtool-
sysroot=/usr/local/oecore-x86_64/sysroots/armv7a-vfp-neon-oe-linux-
gnueabi --with-common-includes=$Path/include/ --with-glib --with-lib-
path=/usr/local/oecore-x86_64/sysroots/armv7a-vfp-neon-oe-linux-
gnueabi/usr/lib --with-btobex --enable-target=sda845
```

b. Build the application:

```
$ cd btapp-samplecode/bt/bt-app/
$ source /usr/local/oecore-x86_64/environment-setup-armv7a-neon-
oemllib32-linux-gnueabi
$ ./configure.sh
$ make
```

c. Connect the board to the host PC via Type C and ensure that the ADB port can be used. Transfer the application:

```
$ adb push main/btapp /usr/bin/
```

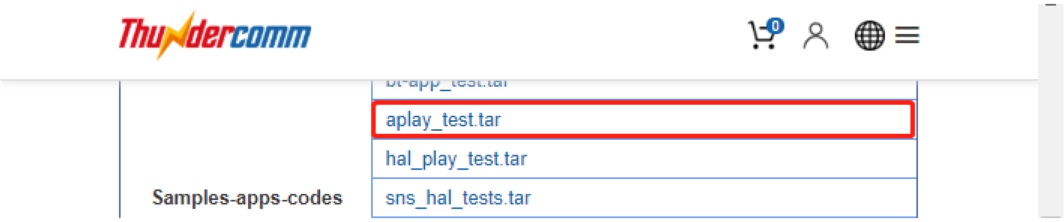
See Section 4.5.2for how to use bluetoothcommand.

5.6 Audio

5.6.1 aplay

1. Download source code

   As mentioned in Secton 4.8.1, the source code for testing audio can be downloaded here.



https://www.thundercomm.com/app_en/product/1544580412842651#doc

The directory after decompression is as follows:

```
$ tree alsa-utils-build -L 1
alsa-utils-build
├── alsa-utils-1.1.4
└── alsa-utils-build.sh
```

2. Build and transfer the application

   a. The compiled script has been integrated into the code.

```
$ cat alsa-utils-build/alsa-utils-build.sh
#!/bin/bash

ALSAWORKSPACE=`pwd`
mkdir -p ${ALSAWORKSPACE}/build
cd ${ALSAWORKSPACE}/build
```

```
../alsa-utils-1.1.4/configure --build=x86_64-linux    --host=aarch64-
oe-linux   --target=aarch64-oe-linux   --prefix=/usr   --
exec_prefix=/usr   --bindir=/usr/bin   --sbindir=/usr/sbin   --
libexecdir=/usr/libexec   --datadir=/usr/share   --sysconfdir=/etc --
sharedstatedir=/com --localstatedir=/var   --libdir=/usr/lib64    --
includedir=/usr/include   --oldincludedir=/usr/include    --
infodir=/usr/share/info   --mandir=/usr/share/man    --disable-silent-
rules   --disable-dependency-tracking --with-libtool-
sysroot=${ALSAWORKSPACE}/recipe-sysroot --disable-rst2man --disable-
bat --disable-xmlto --with-udev-rules-dir=/lib/udev/rules.d --
disable-nls

Make
```

b. Build the application:

```
$  source  /usr/local/oecore-x86_64/environment-setup-aarch64-oe-linux
$ ./alsa-utils-build.sh
```

c. Connect the board to the host PC via Type C and ensure that the ADB port can be used. Transfer the application:

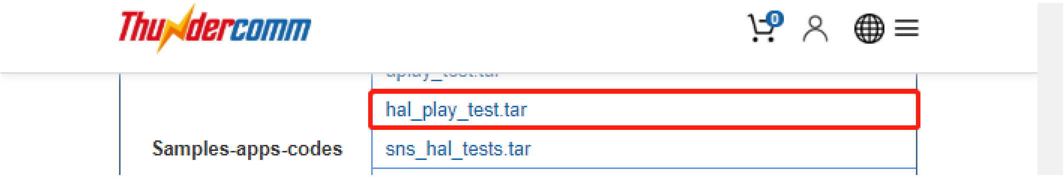
```
$ adb push alsa-utils-build/build/aplay/aplay /usr/bin/
```

See Section 4.8.1 for how to use aplay command.

5.6.2 hal_play_test

1. Download source code

As mentioned in Secton 4.8.2, the source code for testing audio can be downloadedhere.



aplay_test.tar

Samples-apps-codes

hal_play_test.tar

sns_hal_tests.tar

https://www.thundercomm.com/app_en/product/1544580412842651#doc

The directory after decompression is as follows:

```
$ tree hal_play_test -L 1
hal_play_test
├── Android.mk
├── build.sh
├── configure.ac
├── inc
├── lib
├── Makefile.am
├── qahw_effect_test.c
├── qahw_effect_test.h
├── qahw_multi_record_test.c
├── qahw_playback_test.c
├── qahw_playback_test.h
├── qap_wrapper_extn.c
└── trans loopback test.c
```

2. Build and transfer the application

   a. The compiled script has been integrated into the code.

```
$ cat hal_play_test/build.sh
#!/bin/bash

aclocal
autoconf
autoheader
touch NEWS README AUTHORS ChangeLog
automake --add-missing

./configure --build=x86_64-linux --host=aarch64-oe-linux --
target=aarch64-oe-linux --prefix=/usr --exec_prefix=/usr --
bindir=/usr/bin --sbindir=/usr/sbin --libexecdir=/usr/libexec --
datadir=/usr/share --sysconfdir=/etc --sharedstatedir=/com --
localstatedir=/var --libdir=/usr/lib64 --includedir=/usr/include --
oldincludedir=/usr/include --infodir=/usr/share/info --
mandir=/usr/share/man

make
```

   b. Build the application:

```
$  source  /usr/local/oecore-x86_64/environment-setup-aarch64-oe-linux
$ ./build.sh
```

   c. Connect the board to the host PC via Type C and ensure that the ADB port can be used. Transfer the application:

```
$ adb push hal play test /usr/bin/
```

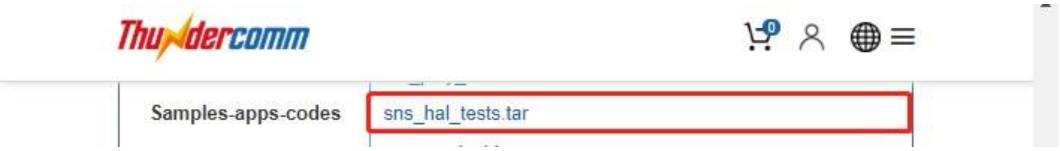
See Section 4.8.2 for how to use aplay command.

5.7 Sensors

1. Download source code

   As mentioned in Secton 4.9, the source code for testing sensors can be downloaded here.

   https://www.thundercomm.com/app_en/product/1544580412842651#doc



   The directory after decompression is as follows:

```
$ tree sns_hal_tests
sns_hal_tests/
├── Makefile
└── sns hal batch.c
```

2. Build and transfer the application

   a. The compiled script has been integrated into the code.

```
$ cat sns_hal_tests/Makefile
OS.exec = uname -s
OS ?= $(shell $(OS.exec))$(OS.exec:sh)
OS := $(OS)

PROG = sns_hal_batch

CFLAGS ?= -O2 -fexpensive-optimizations -frename-registers -fomit-
frame-pointer
CFLAGS += -Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-needed
CFLAGS += -D SNS_LE_QCS605

# Common library includes
LDLIBS = -lsensors.ssc -lpthread

$(PROG): $(PROG).o

$(PROG).o: $(PROG).c $(HDR)

clean:
  $(RM) $(PROG).o $(PROG)

sparse: $(PROG).c
  $(SPARSE) $(CPPFLAGS) $(CFLAGS) $(SPARSEFLAGS) $^

.PHONY: clean
```

   b. Build the application:

```
$  source  /usr/local/oecore-x86_64/environment-setup-aarch64-oe-linux
$ make
```

   c. Connect the board to the host PC via Type C and ensure that the ADB port can be used. Transfer the application:

```
$ adb push sns_hal_batch /usr/bin/
```

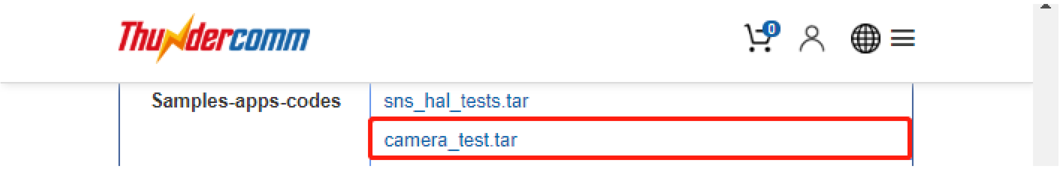
See Section 4.9 for how to use sns_hal_batch command.

5.8 Camera

1. Download source code

   As mentioned in Secton 4.10, the source code for testing camera can be downloaded here.

   https://www.thundercomm.com/app_en/product/1544580412842651#doc



Samples-apps-codes

sns_hal_tests.tar

camera_test.tar

   The directory after decompression is as follows:

```
$tree -L 1 camera_test
camera_test/
├── common.mk
├── count.sh
├── hal3_test
├── include
├── Makefile
└── src
```

2. Build and transfer the application

   a. Edit the Makefile:

```
##### Make sure all is the first target.
all:

CXX ?= g++
CC  ?= gcc
CFLAGS += -I./src/ -DLINUX_ENABLED  -DUSE_GRALLOC1  -
DDISABLE_META_MODE=1
CFLAGS += -DCAMERA_STORAGE_DIR="\"/data/misc/camera/\"" -I./include
CFLAGS += -g -pthread -Wall

 LDFLAGS += -lcutils -lutils -llog -lhardware -lcamera_metadata \
          -lglib-2.0 -ldl -lstdc++ -latomic -lpthread -lOmxCore \
         -lcamera_client -lbinder

C_SRC=
CXX_SRC=
OBJ=
DEP=

CXX_SRC+=src/BufferManager.cpp \
        src/QCameraHAL3Device.cpp \
        src/QCameraHAL3Base.cpp \
        src/QCameraHAL3TestSnapshot.cpp \
        src/QCameraHAL3TestVideo.cpp \
        src/QCameraTestVideoEncoder.cpp   \
        src/QCameraHAL3TestConfig.cpp \
        src/QCameraHAL3TestPreview.cpp \
        src/QCameraHAL3TestTOF.cpp \
        src/TestLog.cpp \
        src/OMX_Encoder.cpp \
        src/BufferManager.cpp

OBJ_CE = src/QCameraHALTestMain.o

hal3_test: $(OBJ_CE)
TARGET_OBJ += $(OBJ_CE)
TARGETS     += hal3_test
include ./common.mk
```

b. Build the application:

```
$ source /usr/local/oecore-x86_64/environment-setup-armv7a-neon-
oemllib32-linux-gnueabi
$ make
```

c. Connect the board to the host PC via Type C and ensure that the ADB port can be used. Transfer the application:

```
$ adb push hal3_test /bin/
```

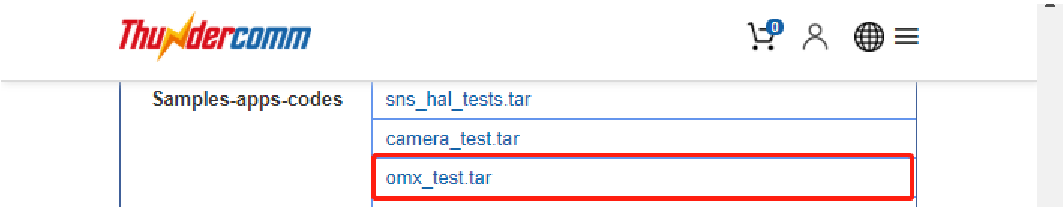
See Section 4.10 for how to use hal3_test command.

5.9 Video

5.9.1 OMX

1. Download source code

   As mentioned in Secton4.11.1, the source code for testing video can be downloaded here.



Samples-apps-codes

sns_hal_tests.tar

camera_test.tar

omx_test.tar

https://www.thundercomm.com/app_en/product/1544580412842651#doc

The directory after decompression is as follows:

```
$ tree mm-video-test -L 1
mm-video-test
├──── build.sh
├──── common
├──── decode
├──── encode
└──── include
```

2. Build and transfer the application

a. The compiled script has been integrated into the code.

```
$ cat mm-video-test/build.sh
#!/bin/bash
mkdir image

cd decode
./build.sh
cd -

cd encode
./build.sh
cd -

cp decode/vtest_decode image/
cp encode/vtest_encode image/
```

b. Build the application:

```
$  source  /usr/local/oecore-x86_64/environment-setup-aarch64-oe-linux
$ ./build.sh
```

c. Connect the board to the host PC via Type C and ensure that the ADB port can be used. Transfer the application:

```
$ adb push image/vtest_decode /usr/bin
$ adb push image/vtest_encode /usr/bin
```

3. Execute the application:

a. Test Video Encode

```
$ adb shell mkdir /data/input
$ adb push 1920_1080.yuv /data/input
$ adb shell
# cd /data/input/
# vtest_encode -o file.264 -i 1920_1080.yuv -W 1920 -H 1080
# vtest_encode -o file.265 -i 1920_1080.yuv -W 1920 -H 1080
```

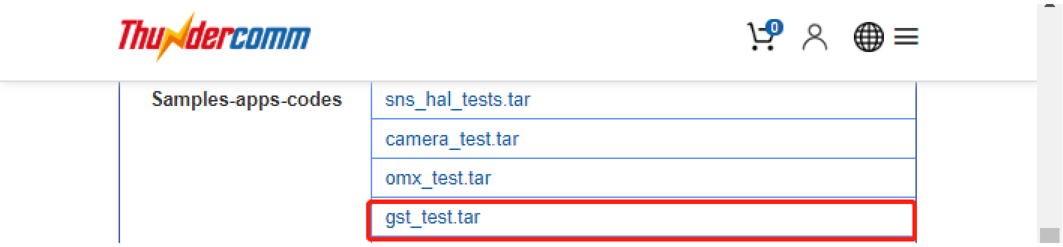
b. Test Video Decode:

```
$ adb shell mkdir /data/input
$ adb push H264_3840_2160_60fps /data/input
$ adb shell
# cd /data/input/
# vtest_decode -o file.yuv -i H264_3840_2160_60fps.264 -W 3840 -H
2160
# vtest_decode -o test.yuv -i H265_3840_2160_60fps.265 -W 3840 -H
2160
```

5.9.2 GST

1. Download source code

   As mentioned in Secton 4.11.2, the source code for testing video can be downloaded here.



Samples-apps-codes	sns_hal_tests.tar
	camera_test.tar
	omx_test.tar
	gst_test.tar

   https://www.thundercomm.com/app_en/product/1544580412842651#doc

   The directory after decompression is as follows:

```
$ tree gst_test -L 1
gst_test
├── build.sh
├── gst-inspect
├── gst-launch
├── gst-stats
├── gst-typefind
└── include
```

2. Build and transfer the application

   d. The compiled script has been integrated into the code.

```
$ cat gst_test/build.sh
#!/bin/bash

GSTWORKSPACE=`pwd`

mkdir -p ${GSTWORKSPACE}/image/usr/bin

#build gst-inspect
cd ${GSTWORKSPACE}/gst-inspect
make
cp gst-inspect ${GSTWORKSPACE}/image/usr/bin

#build gst-launch
cd ${GSTWORKSPACE}/gst-launch
make
cp gst-launch ${GSTWORKSPACE}/image/usr/bin

#build gst-stats
cd ${GSTWORKSPACE}/gst-stats
make
cp gst-stats ${GSTWORKSPACE}/image/usr/bin
```

```
#build gst-typefind
cd ${GSTWORKSPACE}/gst-typefind
make
cp gst-typefind ${GSTWORKSPACE}/image/usr/bin
```

e. Build the application:

```
$  source  /usr/local/oecore-x86_64/environment-setup-aarch64-oe-linux
$ ./build.sh
```

f. Connect the board to the host PC via Type C and ensure that the ADB port can be used. Transfer the application:

```
$ adb push image/usr/bin /usr/bin/
```

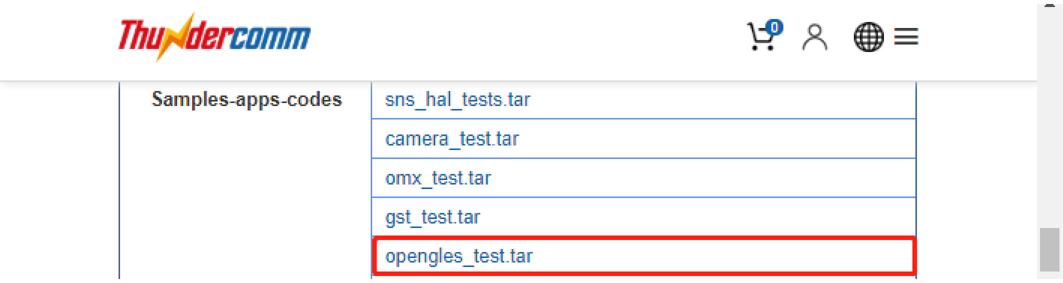
See Section 4.11.2 for how to use gst command.

5.10 OpenGL-ES

1. Download source code

As mentioned in Secton 4.12 the source code for testing OpenGL ES can be downloaded here.

https://www.thundercomm.com/app_en/product/1544580412842651#doc



Samples-apps-codes	sns_hal_tests.tar
	camera_test.tar
	omx_test.tar
	gst_test.tar
	opengles_test.tar

The directory after decompression is as follows:

```
$ tree opengles_test -L 2
opengles_test
└── es11
    ├── Android.mk
    ├── commondefs
    ├── commonrules
    ├── conform
    ├── covgl.txt
    ├── ctk
    ├── doc
    ├── fixed
    ├── GNUmakefile
    ├── include
    ├── platform
    └── subdirrules
```

2. Build and transfer the application

   a. Build the application:

```
$ cd opengles_test/es11
$  source  /usr/local/oecore-x86_64/environment-setup-aarch64-oe-linux
$ export CONFORMES=`pwd`
$ make
```

   b. Connect the board to the host PC via Type C and ensure that the ADB port can be used. Transfer the application:

```
$ adb push conform/conform/conform /bin/conform
```

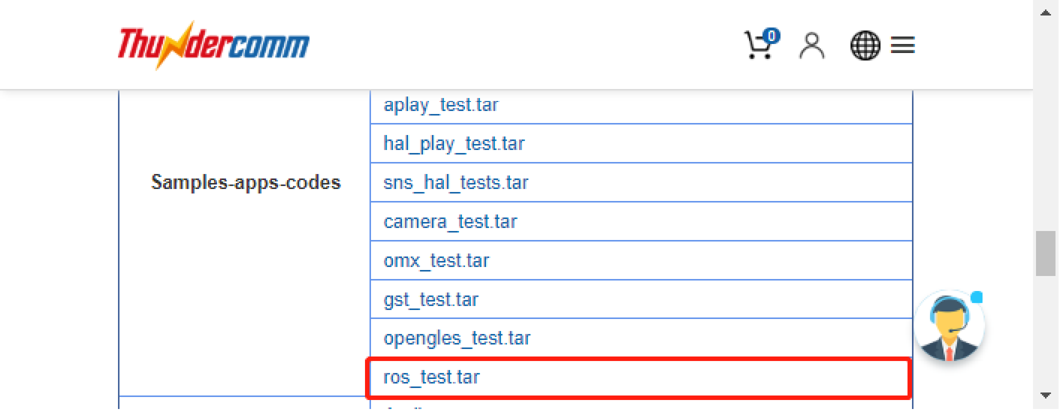
See Section 4.12for how to use conform command.

5.11 ROS

1. Download source code

   As mentioned in Secton 4.14.1 the source code for testing ROS can be downloaded here:

   https://www.thundercomm.com/app_en/product/1544580412842651#doc



Samples-apps-codes	aplay_test.tar
	hal_play_test.tar
	sns_hal_tests.tar
	camera_test.tar
	omx_test.tar
	gst_test.tar
	opengles_test.tar
	ros_test.tar

The directory after decompression is as follows:

```
$ tree ros_test -L 2
ros_test
├── ros-hello-world-master
│    ├── hello_world
│    └── README.md
└──  ros-publisher_subscriber-test
     ├── listener
     ├── Readme.txt
     └── talker
```

2. Build and transfer the application

   a. Build the application:

   - Configure environment

```
$ source /usr/local/oecore-x86_64/environment-setup-armv7a-neon-
oemllib32-linux-gnueabi
```

```
$ export PYTHONPATH=/usr/local/oecore-x86_64/sysroots/aarch64-oe-
linux/usr/lib/python2.7/site-packages
```

- Build Hello world

```
$ cd ros_test/ros-hello-world-master
$ mkdir build ; cd build

$ cmake ../hello_world/ '-DCMAKE_PREFIX_PATH=/usr/local/oecore-
x86_64/sysroots/aarch64-oe-linux/opt/ros/indigo; /usr/local/oecore-
x86_64/sysroots/aarch64-oe-linux/usr' -
DCMAKE_INSTALL_PREFIX:PATH=/opt/ros/indigo -
DCATKIN_BUILD_BINARY_PACKAGE=ON -DSETUPTOOLS_DEB_LAYOUT=OFF -
DCATKIN_ENABLE_TESTING=0 -Wno-dev
```

- Build Listener

```
$ cd ros_test/ros-publisher_subscriber-test
$ mkdir build_listener ; cd build_listener

$ cmake ../listener/ '-DCMAKE_PREFIX_PATH=/usr/local/oecore-
x86_64/sysroots/aarch64-oe-linux/opt/ros/indigo; /usr/local/oecore-
x86_64/sysroots/aarch64-oe-linux/usr' -
DCMAKE_INSTALL_PREFIX:PATH=/opt/ros/indigo -
DCATKIN_BUILD_BINARY_PACKAGE=ON -DSETUPTOOLS_DEB_LAYOUT=OFF -
DCATKIN_ENABLE_TESTING=0  -Wno-dev
```

- Build Talker

```
$ cd ros_test/ros-publisher_subscriber-test
$ mkdir build_talker ; cd build_talker

$ cmake ../taker/ '-DCMAKE_PREFIX_PATH=/usr/local/oecore-
x86_64/sysroots/aarch64-oe-linux/opt/ros/indigo; /usr/local/oecore-
x86_64/sysroots/aarch64-oe-linux/usr' -
DCMAKE_INSTALL_PREFIX:PATH=/opt/ros/indigo -
DCATKIN_BUILD_BINARY_PACKAGE=ON -DSETUPTOOLS_DEB_LAYOUT=OFF -
DCATKIN_ENABLE_TESTING=0 -Wno-dev
```

b. Connect the board to the host PC via Type C and ensure that the ADB port can be used. Transfer the application:

```
$ adb push ros-hello-world-
master/build/devel/lib/hello_world/hello_world /opt/ros/indigo/bin

$ adb push ros-publisher_subscriber-
test/build_listener/devel/lib/listener/listener /opt/ros/indigo/bin

$ adb push ros-publisher_subscriber-
test/build_talker/devel/lib/talker/talker /opt/ros/indigo/bin
```

3. Execute the application:

```
$ adb shell
#cd /opt/ros/indigo/
# bash
```

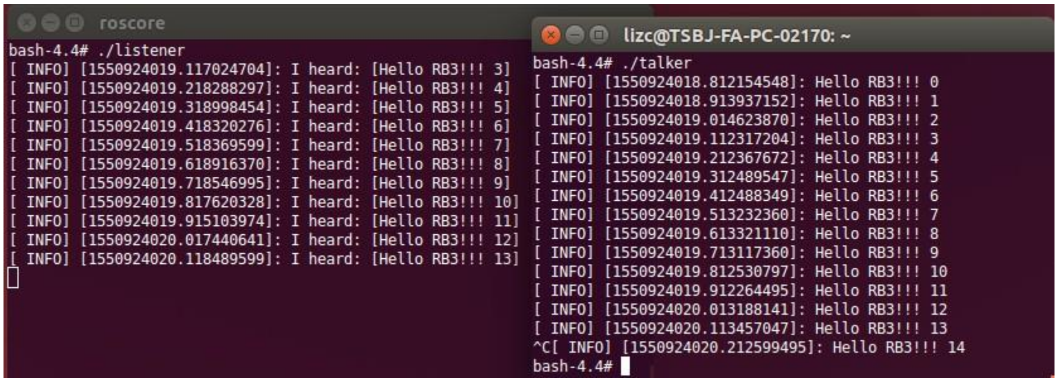
```
bash-4.4#source ./ros-env.sh
bash-4.4# roscore &
bash-4.4# cd bin
bash-4.4# ./listener
```

Open another terminal, send data:

```
$ adb shell
#cd /opt/ros/indigo/
# bash
bash-4.4#source ./ros-env.sh
bash-4.4# cd bin
bash-4.4# ./talker
```

See Section 4.14.1 for how to use hello_world command.

4. The terminal outputs the expected print information:



```
bash-4.4# ./listener
[ INFO] [1550924019.117024704]: I heard: [Hello RB3!!! 3]
[ INFO] [1550924019.218288297]: I heard: [Hello RB3!!! 4]
[ INFO] [1550924019.318998454]: I heard: [Hello RB3!!! 5]
[ INFO] [1550924019.418320276]: I heard: [Hello RB3!!! 6]
[ INFO] [1550924019.518369599]: I heard: [Hello RB3!!! 7]
[ INFO] [1550924019.618916370]: I heard: [Hello RB3!!! 8]
[ INFO] [1550924019.718546995]: I heard: [Hello RB3!!! 9]
[ INFO] [1550924019.817620328]: I heard: [Hello RB3!!! 10]
[ INFO] [1550924019.915103974]: I heard: [Hello RB3!!! 11]
[ INFO] [1550924020.017440641]: I heard: [Hello RB3!!! 12]
[ INFO] [1550924020.118489599]: I heard: [Hello RB3!!! 13]
```

```
bash-4.4# ./talker
[ INFO] [1550924018.812154548]: Hello RB3!!! 0
[ INFO] [1550924018.913937152]: Hello RB3!!! 1
[ INFO] [1550924019.014623870]: Hello RB3!!! 2
[ INFO] [1550924019.112317204]: Hello RB3!!! 3
[ INFO] [1550924019.212367672]: Hello RB3!!! 4
[ INFO] [1550924019.312489547]: Hello RB3!!! 5
[ INFO] [1550924019.412488349]: Hello RB3!!! 6
[ INFO] [1550924019.513232360]: Hello RB3!!! 7
[ INFO] [1550924019.613321110]: Hello RB3!!! 8
[ INFO] [1550924019.713117360]: Hello RB3!!! 9
[ INFO] [1550924019.812530797]: Hello RB3!!! 10
[ INFO] [1550924019.912264495]: Hello RB3!!! 11
[ INFO] [1550924020.013188141]: Hello RB3!!! 12
[ INFO] [1550924020.113457047]: Hello RB3!!! 13
^C[ INFO] [1550924020.212599495]: Hello RB3!!! 14
bash-4.4#
```