Open-Q™ 2100 Development Kit
BSP Programmer's Guide

[Document: ICS-01IMP1181SPPG-001 Version: 1.1]

IDENTIFICATION

Document Title	Open-Q™ 2100 Development Kit BSP Programmer's Guide
Document Number	ITC-01IMP1181SPPG-001
Document Version	1.2
Date	Oct 17, 2017

Table of Contents

1. INTRODUCTION

1.1 Purpose

The purpose of this BSP Programmers Guide is to provide primary user information for programming of software intended for Android Board Support Package.

For technical questions related to the Open-Q™ 2100 Development Kit go to:
https://helpdesk.intrinsyc.com

For more Android-related device information, see the Qualcomm Developer Network page at:
http://developer.qualcomm.com/dev/android.

If you are looking for information on developing applications only please visit:
http://developer.android.com/tools/index.html

1.2 Scope

This document describes the following for the Open-Q™ 2100 Development Kit:

- Software Licensing and version information
- Accessing Android software for the kit
- Setting up your PC development environment used to build/install software on the kit
- Building the software binaries from source code
- Methods to download/install Android software binaries from your PC onto the kit
- Debug/ADB usage.

1.3 Intended Audience

This document is intended for developers who have purchased an Open-Q™ 2100 Development Kit and are interested in Android BSP customization / Linux Device driver development / modification.

1.4 Organization

This document is organized as follows:

- **Section 1. Introduction:** This section describes the purpose, scope and structure of this document.
- **Section 2. Documents:** This section lists other documents that are parents of or supplement this document.
- **Section 3. Software Licensing:** This section identifies the Android Software licensing for the software supplied for use on your Open-Q™ 2100 Development Kit.
- **Section 4. Software Version Tracking:** This section identifies Android Software version information for the software supplied for use on your Open-Q™ 2100 Development Kit.
- **Section 5. Source Code Access:** This section describes where and how to access the Android BSP including the kernel source code that runs on the Open-Q™ 2100 Development Kit.
- **Section 6. Building an Android Software Image:** This section describes how to setup your host PC software development environment and build software binaries from source code for use with your Open-Q™ 2100 Development Kit.
- **Section 7. Installing an Android Software Image:** This section describes how to install Android software binaries onto your Open-Q™ 2100 Development Kit
- **Section 8. Advanced Development and Debugging Tips:** This section describes how to configure and control the various subsystems that are part of your Open-Q™ 2100 Development Kit.
- **Section 9. Troubleshooting:** This section describes some known problems and suggested solutions.

1.5 Acronyms

TERM AND ACRONYMS	DEFINITION
RF	Radio Frequency
SOM	System on Module
QHD	Quarter High Definition
SPMI	System Power Management Interface (Qualcomm PMIC / baseband proprietary protocol)
UIM	User Identity module
NFC	Near Field Communication
EMMC	Embedded Multimedia Card
USB HS	USB High Speed
USB SS	USB Super Speed
DSI	MIPI Display Serial Interface
MIPI	Mobile Industry processor interface
SATA	Serial ATA
ANC	Audio Noise Cancellation
AMIC	Analog Microphone
JTAG	Joint Test Action Group
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
BLSP	Bus access manager Low Speed Peripheral (Serial interfaces like UART / SPI / I2C/ UIM)
SLIMBUS	Serial Low-power Inter-chip Media Bus
MPP	Multi-Purpose Pin
CSI	Camera Serial Interface
DP	Display Port
HDMI	High Definition Media Interface
GPS	Global Positioning system
EEPROM	Electrically Erasable Programmable Read only memory
SSBI	Single wire serial bus interface (Qualcomm proprietary mostly PMIC / Companion chip and baseband processor protocol)
LNA	Low Noise Amplifier
HSIC	High Speed Inter Connect Bus

1.6 Resources

The following resources were used in the creation of this document:

- http://source.android.com/source/initializing.html
- http://source.android.com/source/version-control.html
- http://developer.android.com/sdk/index.html
- http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html

2. DOCUMENTS

This section lists any parent and supplementary documents for the Open-Q™ 2100 Development Kit User Guide. Unless stated otherwise, applicable documents supersede this document and reference documents provide background and supplementary information.

2.1 Applicable Documents

REFERENCE	AUTHOR	TITLE
A-1	Intrinsyc	Intrinsyc Purchase and Software License Agreement for the Open-Q™ 2100 Development Kit

2.2 Reference Documents

REFERENCE	DOCUMENT NUMBER	AUTHOR	TITLE
R-1	01RND1260-QSG-001	Intrinsyc	Open-Q™ 2100 Development Kit Quick Start Guide
R-2	01RND1260-UG-001	Intrinsyc	Open-Q™ 2100 Development Kit User Guide
R-3	ICS-01IMP1181SPPG-001	Intrinsyc	Release Notes for The Open-Q™ 2100 Development Kit

3. SOFTWARE LICENSING

3.1 Introduction

As a part of purchasing the Open-Q™ 2100 Development Kit, you are provided access to

- Android BSP binary images
- Android BSP source code
- Qualcomm Proprietary binary blob for Hardware Acceleration.

3.2 Software Licensing

Your use of this document is subject to and governed by those terms and conditions in the Intrinsyc Purchase and Software License Agreement for the Snapdragon 2100 based Open-Q™ 2100 Development Kit, which you or the legal entity you represent, as the case may be, accepted and agreed to when purchasing an Open-Q™ 2100 Development Kit from Intrinsyc Technologies Corporation ("**Agreement**"). You may use this document, which shall be considered part of the defined term "Documentation" for purposes of the Agreement, solely in support of your permitted use of the Open-Q™ 2100 Development Kit under the Agreement. Distribution of this document is strictly prohibited without the express written permission of Intrinsyc Technologies Corporation and its respective licensors, which they can withhold, condition or delay in its sole discretion.

This document contains technical data that 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.

4. SOFTWARE VERSION TRACKING

4.1 Introduction

Software releases from Intrinsyc use a release version based on an underlying Linux Foundation owned and Qualcomm maintained software baseline from CAF.

4.2 Software Version Number Convention

The BSP software version of the Intrinsyc Open- Q™ 2100 Development Kit is a 2-digit version number signifying the major and minor software release (e.g., 1.1). This version number convention is used for Intrinsyc internal tracking purposes and maintenance of the bugs reported based on the version number information.

4.3 Determining your Software's Version Number

You can check your BSP software version from Android, "Settings->About"

The system will show the following version number:

msm8909w-userdebug 7.1.1 OpenQ2100-v1.1 eng.git.20170802.013256 test-keys

5. SOURCE CODE ACCESS

5.1 Introduction

As mentioned in section 4.1, software releases from Intrinsyc use a release version based on an underlying Qualcomm software baseline from CAF.

Users can build their own BSP software images or they can use pre-built software images from Intrinsyc to program new images on their device.

If you wish to program pre-built software images from Intrinsyc, you can skip the information in this section and go directly to section 7, “Android Software Image”.

5.2 Downloading the Board Support Package

Users can download the Board Support Package from the same Intrinsyc website where this document was obtained.

Download the “8009w_Open-Q_2100_Android_BSP-N-1.1.zip” to your Linux build machine to build from source. Extracting the zip file, the contents of the package would be as below:

```
8009w_Open-Q_2100_Android_BSP-N-1.1.zip
├── Binaries                : Precompiled Binaries
│   ├── boot.img            : Linux Kernel + Ramdisk boot image
│   ├── cache.img           : Android Cache partition ext4 image
│   ├── flashall.sh         : Script to flash images over fastboot
│   ├── flashall.bat        : Script to flash images over fastboot for Windows
│   ├── persist.img         : Android persist partition ext4 image
│   ├── recovery.img        : Recovery image
│   ├── system.img          : Android system partition ext4 image
│   └── userdata.img        : Android data partition ext4 image
├── README.txt              : A very quick README for the release
├── Source_Package          : Source Package
│   ├── getSource_and_build.sh   : Script to download and build from source
│   ├── patches             : Patches for Open-Q 2100
│   └── propritary.tar.gz   : Qualcomm Proprietary Libraries
└── usb_driver              : Windows Driver for ADB for Open-Q 2100
```

After copying the above files to your Linux PC, follow the steps in section 6, "Building an Android BSP” to extract and build the source code.

5.3 Code Aurora Forum (CAF)

The Source Package mentioned in the previous section includes a script file that will pull software from CAF to your Linux PC. This script file will automatically download the correct branch of code from CAF using the specific manifest needed for building the appropriate Android software image for the development kit.

Note:

1. Please don’t post your board specific or BSP specific technical questions on CAF, as CAF contains many other projects for different families. It won’t be answered at all.

6. BUILDING AN ANDROID BSP

6.1 Introduction

This section describes how to set up your development workstation (host PC), including the software tools required to build software from source code for your Open-Q™ 2100 Development Kit.

A description is also provided for the steps to build the software for your Open-Q™ 2100 Development Kit.

6.2 Development Environment Setup

6.2.1 Introduction

A PC running Ubuntu Linux is required for use as the development environment to build the Android BSP

The following table identifies the specific hardware, software and other equipment needed for a developer to install and run the software:

Item #	Item description	Version	Source/vendor	Purpose
1	Linux development workstation exceeding minimum desktop system requirements for running Ubuntu 64-bit OS	—		Android build machine
2	Ubuntu 14.04.x Linux distribution for 64-bit architecture	14.04 LTS	Ubuntu Community/ Canonical, Ltd.	Android build host OS
3	Open JDK for Linux x64	8	Open Source	Required for building Android
4	Repo	--	Android Open Source Project	Android source management tool
5	Internet connectivity for your Linux PC	--		The Linux PC will download packages from CAF to be built.

Note:

1. BSP release as build environment is tested on Ubuntu 14.04 (64-bit). Other build environments might work but are not tested.
   NOTE: Android 7.1.1 takes long to build so it is recommended at least to have 16GB RAM available on Quad core (i7) CPU
2. Please check your build machine before proceeding below steps.
3. The open source Linux Foundation hosted `codeaurora.org` contains a huge amount of code that is approximately 40GB when downloaded. A full build will consume approximately 60 GB of disk space. Please account for this, otherwise you will potentially cause errors during the build process.

6.2.2 Initializing Build Environment

6.2.2.1 Setup Build Machine

1) The BSP currently is tested and supported with Ubuntu 14.04 – 64bit LTS. Building on a virtual machine running Ubuntu is also tested and supported. The current supported release for building the BSP is: Ubuntu 14.04 – 64bit LTS.

2) Install the packages needed for the building the BSP

```
sudo apt-get install git gnupg flex bison gperf build-essential
sudo apt-get install tofrodos lib32z1-dev lib32stdc++6 libxml2-utils xsltproc
```

6.2.2.2 Install JDK

The Android build system for Open-Q™ 2100 Nougat onwards requires Open JDK 1.8. to install the JDK version of Java 8, use below commands

```
sudo apt-get install default-jre
sudo apt-get install default-jdk
```

6.2.3 Repo Installation

Repo is a source code configuration management tool used by the Android project. It is a front end to git written in Python. Repo uses a manifest file to aid downloading code bases organized as a set of projects stored in different git repositories.

To install repo:

1. Create a ~/bin directory in your home directory, or, if you have root or sudo access, install for all system users under a common location, such as /usr/local/bin or somewhere under /opt
2. Download the repo script:

```
   $ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo >
~/bin/repo
```

3. Set the repo script’s attributes to executable.

```
$ chmod a+x ~/bin/repo
```

4. Make sure that the installed directory location for repo is included in your PATH.

```
$ export PATH=~/bin:$PATH
```

5. Try running “repo –help” to verify installation; you should see a message similar to the following:

```
$ repo –help
```

usage: repo COMMAND [ARGS] repo is not yet installed. Use "repo init" to install it here.

The most commonly used repo commands are:

init – Install repo in the current working directory

help – Display detailed help on a command

For access to the full online help, install repo (“repo init”).

6.3 Downloading and Building Android BSP Images from Source

6.3.1 Introduction

This section describes how to build software for your Open-Q™ 2100 Development Kit from source code.

This section assumes your development workstation has already been set up according to section 6.2.

6.3.2 Build Instructions

Following contains Source_Package structure of the release.

```
Source_Package  ---------------------------------Source code patches
    |-- getSource_and_build.sh  -----------------Main download and build script
    |-- patches  --------------------------------Patches for Open-Q 2100 DevKit support
    |-- proprietary.tar.gz  ---------------------Qualcomm-specific binaries required for Android
```

Source build is depending on the following CAF project.

https://www.codeaurora.org/projects/all-active-projects/android-msm

For git / code browsing you can visit below

https://www.codeaurora.org/cgit/quic/la

The current release is based on CAF “LAW.BR.2.0-01710-8x09w.0” manifest / TAG.

Note: This TAG may change for future software upgrades.

After setting up the build machine and downloading described in section 5.3, create one workdir in your /home/username/ or /opt/ folder a.k.a. use following commands to start the build

```
build@ubuntu$ cd /home/username/

build@ubuntu$ mkdir workdir

build@ubuntu$ cd workdir

build@ubuntu$ cp -ar /home/username/Downloads/8009w_Open-Q_2100_Android_BSP-N-
1.1/Source_Package/ /home/username/workdir/

build@ubuntu$ cd Source_Package

build@ubuntu$ fromdos getSource_and_build.sh

Note: Above command converts from dos file to linux as if you download / extract
source package on windows, you might need above command.

build@ubuntu$ chmod 777 getSource_and_build.sh
```

The above script downloads from CAF, applies patches to respective Android projects, extracts propritetary.tar.gz and creates android build root named:

```
APQ8009W_LAW.BR.2.0-01710-8x09w.0_N_v1.0
```

Note: Before you run this script make sure your internet connection is fine to download code from Code Aurora, and you have necessary hard-disk space, enough CPU power and necessary RAM.

1) The script uses 4 threads concurrently to build the Android Image for board, open getSource_and_build.sh script and search following line where it starts build

```
make -j4 BUILD_ID=OPENQ_$ITCVER
```

Above –j4 is using 4 threads – change that to 8 or 2 depending upon your CPU power.

2) Minimum 8 GB RAM is recommended for build process.

```
build@ubuntu$ ./getSource_and_build.sh
```

After you run above script, it will start the code download, apply the patches required for Open-Q™ 2100, and then start the build process to compile the source code automatically.
The script downloads code to `/home/username/workdir/Source_Package/8009w_Open-Q_2100_Android_BSP-N-1.1` where you will find your Android working build tree with root Makefile.

After a successful automated build, if you wish to build yourself, a.k.a. after making some changes to the Android BSP, use the below commands to build subsequently.

```
build@ubuntu$ cd /home/username/workdir/Source_Package/8009w_Open-Q_2100_Android_BSP-N-1.1/
build@ubuntu$ source build/envsetup.sh
build@ubuntu$ lunch msm8909w-userdebug
build@ubuntu$ make –j8
```

7. ANDROID SOFTWARE IMAGE INSTALLATION

7.1 Introduction

This section describes how to install Android software binaries you built / prebuilt onto your Open-Q™ 2100 Development Kit.

Note: In any following section if mentioned <android-source-tree> should be mapped to the following path:

/home/username/workdir/8009w_Open-Q_2100_Android_BSP-N-1.1

7.2 Using JFLash to upgrade previous BSP version or after SD card recovery

For Open-Q 2100 SOMs running earlier software versions, including 1.0, the method of upgrading to software version 1.1 is to use the JFash utility. This will update all binary software components on the device, not only those programmed using fastboot.

JFlash is a Java tool used on both Linux and Windows. Before upgrading to this new BSP release, one will need to execute this one-time upgrade. Please visit http://support.intrinsyc.com to download the latest 8009w_Open-Q_2100_Android_BSP-N-1.1_JFlash_Upgrade.zip file for your board.

You will need the following additional files included in the BSP release zip file:

- Flashall.sh – Linux script to autoload/program the images via Android Fastboot tools
- Flashall.bat – Windows batch script to autoload/program the images via Android Fastboot tools
- Windows USB drivers/installation files.

Notes:

1. The Windows USB drivers/installations files are needed only if you intend to use a Windows PC instead of a Linux PC to program/download binary images to your Development Kit.
2. You should uninstall any Windows PC drivers previously installed for older software releases before installing the new Windows USB drivers for this release.
3. Any pre-existing adb server on the PC should be terminated using the "**adb kill-server**" command.
4. Use of the scripts above will re-program the user data section of the flash memory. Accordingly, any existing user files and settings will be lost. Settings (for example the HDMI Out configuration set via the LCD display) must be re-entered after the programming of the software is complete.

7.3 Fastboot and ADB

7.3.1 Introduction

Fastboot is the used to install an Android image from a Linux (Ubuntu) development PC over a USB connection to the Open-Q™ 2100 Development Kit. Fastboot can also been used to install an android image from Windows machine.

Android Debug Bridge (ADB) is a debug interface over USB between your PC and the development kit. ADB is not required for installing a software image, but its configuration on a PC is similar to that of Fastboot and therefore ADB configuration included in this document for convenience of PC configuration.

Fastboot and ADB for Linux are available from the following folder after you follow steps in section 6.3 to build an image successfully:

<android-source-tree>/out/host/linux-x86/bin/.

Alternatively, if you have not configured a full Linux development environment and built an image, you can use the Fastboot and ADB binaries supplied by Google's Android SDK (http://developer.android.com/sdk/index.html). In either case, ensure Fastboot and ADB are in your PATH for your Linux PC.

Before you can use Fastboot and ADB, you must ensure your PC is configured to recognize the Open-Q™ 2100 by configuring the USB VID/PID. To configure the USB VID/PID for the Open-Q™ 2100 for use with Fastboot and ADB on your PC, follow the instructions in section 7.3.2 before you connect your PC to your OpenQ™ 2100.

7.3.2 USB Driver Configuration for Fastboot and ADB

As described in Google's instructions for setting up a hardware device (http://developer.android.com/tools/device.html), your Linux development workstation USB driver configuration must be modified to recognize the development kit when you use ADB or Fastboot from Google's Android SDK.

Here are the configuration settings required for using ADB and Fastboot with the kit:

1. Log in as root and navigate to the following directory:

```
build@ubuntu$ cd /etc/udev/rules.d/
```

2. Create this file or edit it if already existing:

   51-android.rules

   Add the following lines to the end of the file to use platforms with software:

```
#Fastboot low-level bootloader
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", MODE="0777", GROUP="adm"
# adb composite interface device 9025
SUBSYSTEM=="usb", ATTR{idVendor}=="05C6", MODE="0777", GROUP="adm"
```

   Now execute:

```
build@ubuntu$ sudo chmod a+r /etc/udev/rules.d/51-android.rules
Unplug your device if connected and run below command and then
re-connect your device.
build@ubuntu$ sudo service udev restart
```

3. After step 2, you need to connect the USB micro connector (DEBUG UART) via a USB cable with Minicom, Putty or TeraTerm on your host machine. On your Host PC set your UART settings to following

   BaudRate: 115200

   Parity: None

   Data Bits: 8

   Hardware Flow Control: None

   Software Flow Control: None

   Stop Bits: 1

4. After step 2, you need to put your development kit into Fastboot Mode before connecting via USB to the PC to communicate via Fastboot or ADB. The following are 2 methods to put your development kit into Fastboot mode:
   a. Press the Volume - button on the carrier board while powering the board on
   b. Power on the board and type “su” and “reboot bootloader” on the serial debug UART

      In either case above, the board will boot into fastboot mode, showing the following debug output on the serial debug UART:

      […]

      [160] fastboot_init()

      [160] udc_start()

      This will result in the dev kit rebooting into Fastboot mode

5. To confirm successful communication between the PC and dev kit in Fastboot Mode, you can type “fastboot devices” or “lsusb” on the PC to detect the dev kit. You should see a 18d1:d00d – Unnamed device

7.3.3 Programming System Images using Fastboot

These steps assume your Open-Q™ 2100 Development Kit eMMC has a pre-existing Android image configured (note kits supplied by Intrinsyc will have a pre-existing Android image installed). At a minimum, a boot partition with the Android boot loader LK image is required for using Fastboot to program a new image.

Steps to programming using Fastboot:

1. Ensure you have followed the steps in sections 7.3.1 and 7.3.2 to have Fastboot in your PATH, functional on your PC, and your dev kit booted into Fastboot Mode.
2. If you are programming pre-built binary images downloaded from Intrinsyc website:
   a. You should have the following files:
      i. Programming script: flashall.sh

Note: flashall.sh assumes that adb and Fastboot android utilities are already in your path, if not update your .bashrc accordingly.

ii. image files: boot.img, cache.img, persist.img, system.img, userdata.img, recovery.img

b. Copy all the above files to a folder on your Linux PC

c. Make sure the above flashall.sh programming script is executable.

d. Execute the flashall.sh script

e. If the dev kit does not reboot automatically after executing the script, turn off/on your board to reboot your dev kit.

3. If you are programming your own built binary images follow the instructions from section 6.3:

a. Obtain the flash programming script "flashall.sh" from the pre-built binary image folder from the BSP, you can find it under "8009w_Open-Q_2100_Android_BSP-N-1.1.zip/Binaries"

b. Copy the flash programming script to the following directory:

< android-source-tree >/out/target/product/msm8909w

c. Navigate to the following directory:

```
cd < android-source-tree >/out/target/product/msm8909w
```

Copy flashall.sh from release to < android-source-tree >/out/target/product/msm8909w

d. Execute the `flashall.sh` script[1]

e. If the dev kit does not reboot automatically after executing the script, turn reboots your board to fastboot updated the image and reboot your dev kit with newly programmed images

4. Following is the eMMC partition structure you can use to program your images

Number	Start (Sector)	End (Sector)	Size	Code	Name
4			1024 KiB	FFFF	aboot
20			32 MiB	FFFF	boot
21			1.2 GiB	FFFF	system
23			67 MiB	FFFF	cache
22			21 MiB	FFFF	persist
33			2 GiB	FFFF	userdata

Warning! Don't program or try to go beyond above partition size and sector limit, otherwise it might damage your board into an un-recoverable mode.

[1] Note this script will erase user data, as part of the new image installation.

You can follow the above partition structure and program your partition using Fastboot.

Example, to program the recovery partition:

```
build@ubuntu$fastboot flash recovery recovery.img
```

Syntax:

```
build@ubuntu$fastboot flash <emmc_partition> <emmc_parititon image from build>
```

This will program specified partition with image provided.

Example, to erase recovery image:

```
build@ubuntu$fastboot erase recovery
```

syntax is:

```
build@ubuntu$fastboot erase <emmc_parition>
```

This will erase specific partition using fastboot

7.3.4 Fastboot and ADB use on a Windows PC

It is also possible to use Fastboot and ADB from a Windows PC for software image programming and debugging. You will need to:

1. Install the Android SDK (http://developer.android.com/sdk/index.html) on your Windows PC.
2. Edit Environment variable to add adb and fastboot to path.
3. Install Windows drivers for the dev kit on your Windows PC. These drivers are available with the release from the Intrinsyc support website, in folder "usb_driver" of Intrinsyc-OpenQ-2100-AndroidBSP-N-v1.0 package.

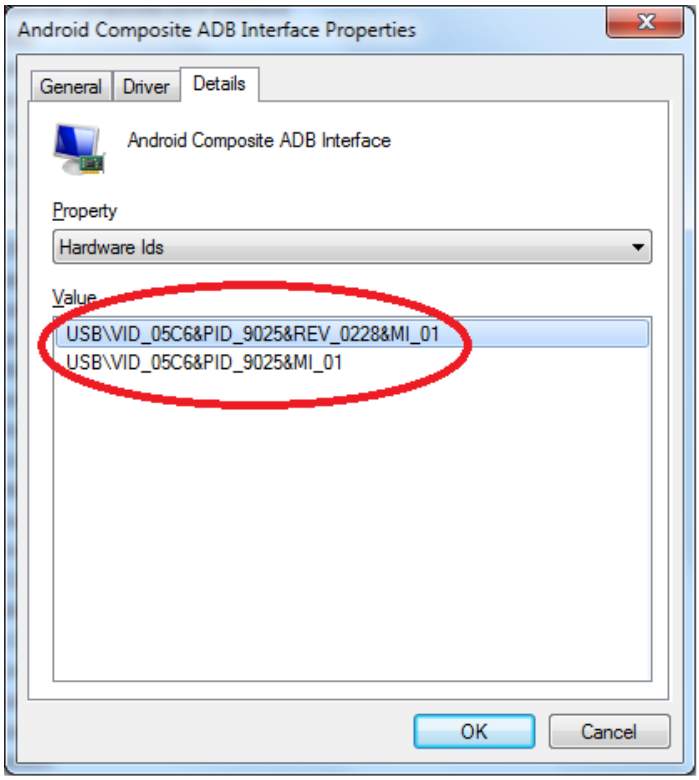
Notes:

a) If you have installed a previous version of PC drivers from an older release of the dev kit software or from other products, you must first uninstall those older drivers first.

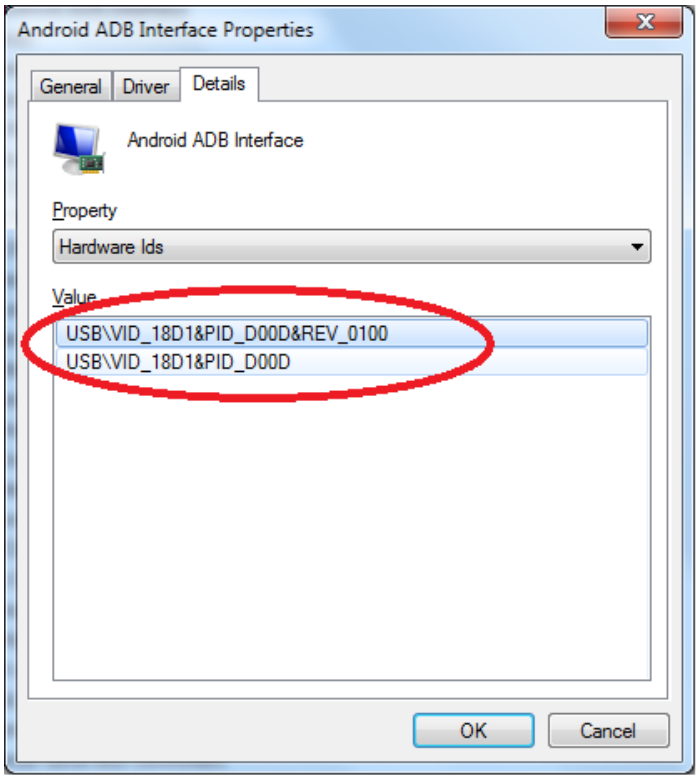
b) Please remember any pre-existing adb server on the PC should be terminated using the "adb kill-server" command

c) The device creates 4 different endpoints out of which only one endpoint will work; the other USB unknown-devices should be ignored.

d) While the USB drivers are installed you should make sure to use the following VID\PID for the ADB interface:



When device is in fastboot, then you will see the following VID\PID:



4. To install prebuilt binaries to a windows machine just unzip release downloaded from Intrinsyc web site. The image files to be flashed are located in “Intrinsyc-OpenQ-2100-AndroidBSP-N-v1.0/Binaries”

The files are:

boot.img, cache.img, persist.img, system.img, userdata.img, recovery.img

Once the above steps are complete, you can put the dev kit into Fastboot Mode (refer to section 7.3.2, step 4), open a Windows command prompt from “IntrinsycOpen-Q2100AndroidBSP-Lollipop-1.2/Binaries” directory and issue the following fastboot commands (or simply run the batch file if supplied)[2]:

OR

4. If you have compiled your own images from the BSP source, as mentioned in Section 6.3.

Then copy boot.img, cache.img, persist.img, system.img, userdata.img, recovery.img files to your windows machine and use the following commands from the Windows command prompt:

fastboot flash system system.img

fastboot flash persist persist.img

[2] Note this will erase user data, as part of the userdata programming.

```
fastboot flash cache cache.img
fastboot flash userdata userdata.img
fastboot flash boot boot.img
fastboot flash recovery recovery.img

fastboot boot boot.img
fastboot reboot
```

Or

Copy “flashall.bat” from “Intrinsyc-OpenQ-2100-AndroidBSP-N-v1.0/Binaries” to the place where you copied all the above files, and then runt the batch file.

Note:

flashall.bat file assumes you have adb and fastboot android utilities are in your command prompt path or system wide path.

Device will fully boot to downloaded image.

8. ADVANCED BUILDING TIPS

8.1 Introduction

This section describes how to configure the kernel / recompile kernel only.

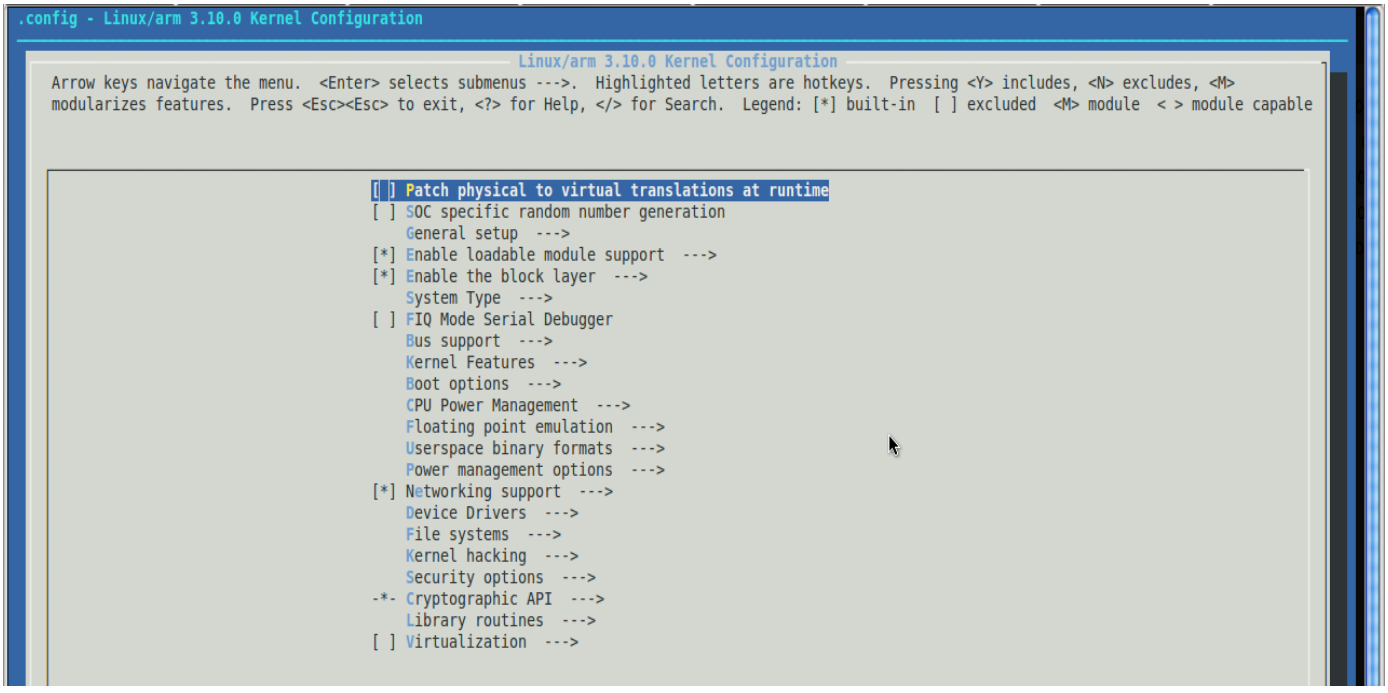
8.2 Reconfiguring / Recompiling and Updating Kernel Image on Device

Use below command to use kernel menuconfig options for changing kernel configuration as per your need.

Note : If you don't have the `ncurses` development libraries installed on your build PC, you might get errors while doing kernel menuconfig.

```
build@ubuntu$ sudo apt-get install libncurses5-dev

build@ubuntu$ cd <android-source-tree>

build@ubuntu$ source build/envsetup.sh

build@ubuntu$ lunch msm8909w-userdebug

build@ubuntu$ make ARCH=arm CROSS_COMPILE=arm-linux-androideabi- msm8909w_defconfig

build@ubuntu$ make ARCH=arm CROSS_COMPILE=arm-linux-androideabi- menuconfig
```

This will start the following screen with kernel menuconfig options shown in screen below.



Change your kernel config options as per your need

```
build@ubuntu$ cp –v .config arch/arm/configs/msm8909w_defconfig
```

Use below command to recompile only the kernel and then boot image.

```
build@ubuntu$ cd ../; make –j8 bootimage
```

This will build the boot image with updated kernel configuration you change just as above, then reprogram the boot image as mentioned in Section 7.2.3.

8.3 FAQS for Open-Q™ 2100 BSP

8.3.1 ADB Root permission / Root fs remount in read / write mode doesn't work with build.

To get full root permission on Open-Q™ 2100 BSP

Open File

1) `<android-source-tree>/out/target/product/msm8909w/root/default.prop`

2) set `ro.secure=0` instead of default `ro.secure=1`, due to user-debug build it sets default.prop to 1 which disables the any root permission.

3) Compile to kernel + ramdisk = bootimage (Keeping you have the build environment in the console setup)

```
build@ubuntu$ make –j8 bootimage
```

This will generate new image.

4) Program the new boot.img with following commands and then reboot the board.

```
fastboot flash boot <android-source-tree>/out/target/product/msm8909w/boot.img
fastboot reboot
```

5) Perform remount of system partition or use adb shell, you should get full permission / root file system mounted in read / write mode.

OR

1) Build engineering debug build.

2) Qualcomm recommends using user-debug build for apq8009w(msm8909w), which is default for Open-Q™ BSP but for gaining root access you can compile engineering build with following build commands.

```
build@ubuntu$ cd <android-source-tree>
build@ubuntu$ . build/envsetup.sh
build@ubuntu$ choosecombo 1 msm8909w 3
```

3) Compile the entire android.

```
build@ubuntu$ make –j4
```

Warning!! This will erase your previous build as variant for build changes if you have, and start from fresh, program all images, with full access to everything.

4) Program the newly generated images

OR

Use following adb commands

1) adb root
2) adb remount

Above command should succeed

3) adb shell should give you root permission

EXTERNAL REFERENCES

Some Useful Links for Qualcomm Technologies and Accelerated Application Development and debugging tools.

Augmented Reality with Snapdragon Family Processors

Vuforia: https://developer.qualcomm.com/mobile-development/mobile-technologies/augmented-reality

Accelerated Computer Vision Processing for Snapdragon Family

FastCV: https://developer.qualcomm.com/mobile-development/mobile-technologies/computer-vision-fastcv

Gaming and Adreno GPU:

https://developer.qualcomm.com/mobile-development/mobile-technologies/gaming-graphics-optimization-adreno

Snapdragon SDK:

https://developer.qualcomm.com/mobile-development/mobile-technologies/snapdragon-sdk-android

Debugging Tools:

https://developer.qualcomm.com/mobile-development/performance-tools/trepn-profiler

Trepn Profiler for profiling applications

Multimedia and Qualcomm Hexagon DSP Program:

https://developer.qualcomm.com/mobile-development/mobile-technologies/multimedia-optimization/hexagon-dsp-access-program