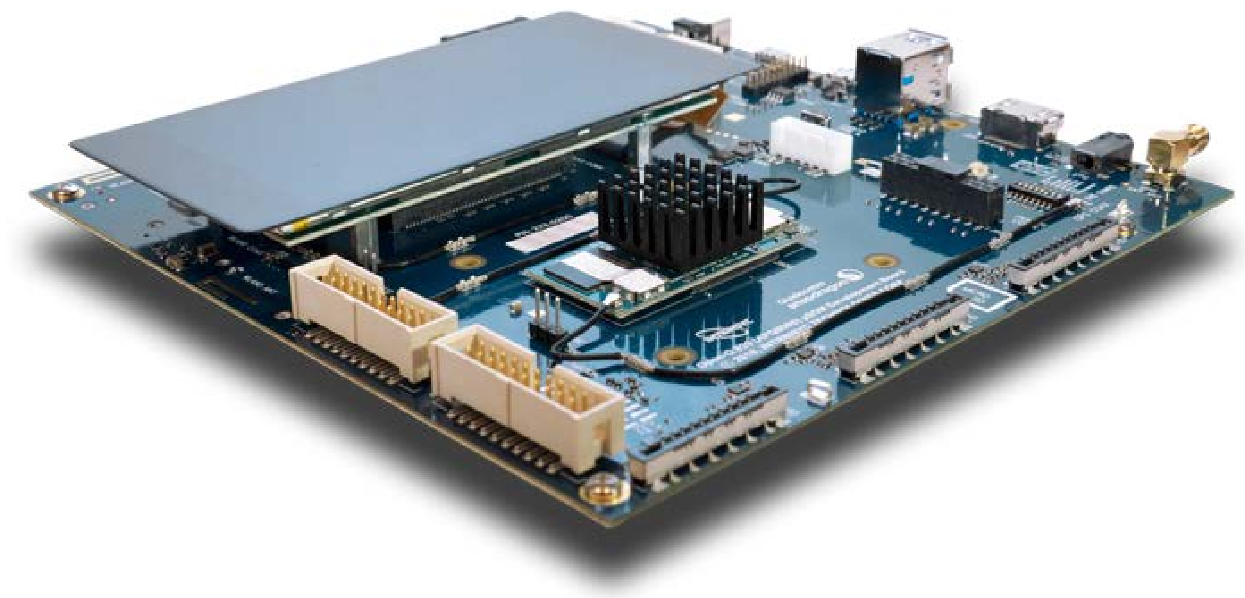


Open-Q™ 820 µSOM Development Kit (APQ8096) BSP Programmer Guide for Android 9 v5.1

[Document: ITC-01RND1239-BSPPG-001 revision: 1.0]

Identification

Document Title	Open-Q™ 820 µSOM Development Kit (APQ8096) BSP Programmer Guide for Android 9 v5.1
Document Number	ITC-01RND1239-BSPPG-001
Document revision	1.0
Date	Dec. 9, 2019

Table of Contents

1. INTRODUCTION..4
1.1 Purpose 4
1.2 Scope.. 4
1.3 Intended Audience 4
1.4 Organization 5
1.5 Acronyms.. 6
1.6 Resources.. 7
2. DOCUMENTS..8
2.1 Applicable Documents.. 8
2.2 Reference Documents.. 8
3. Software Version Tracking..9
3.1 Introduction.. 9
3.2 Software Version Number Convention 9
3.3 Determining your Software's Version Number.. 9
4. Source Code Access 10
4.1 Introduction.. 10
4.2 Downloading the Board Support Package.. 10
4.3 Code Aurora Forum (CAF) 11
5. Building an Android BSP 12
5.1 Introduction.. 12
5.2 Development Environment Setup.. 12
5.2.1 Introduction.. 12
5.2.2 Initializing Build Environment.. 13
5.2.3 repo Installation.. 14
5.3 Downloading and Building Android BSP Images from Source.............. 15
5.3.1 Introduction.. 15
5.3.2 Build Instructions.. 15
6. Installing an Android Software Image..17
6.1 Introduction.. 17
6.2 Fastboot and ADB 17
6.2.1 Introduction.. 17
6.2.2 USB Driver Configuration for fastboot and adb on Linux (Ubuntu) Machine 17
6.2.3 Programming System Images using fastboot.. 19
6.2.4 Fastboot and adb use on a Windows PC 21
7. Advanced Building Tips 23
7.1 Introduction.. 23
7.2 Reconfiguring / Recompiling and Updating Kernel Image on Device..... 23
7.3 FAQS for the 820 uSOM BSP.. 24
7.3.1 Build configuration 24
7.3.2 Programming newly generated Android images.. 24
7.3.3 Remounting system partition over adb.. 24

8. External references..25

1. INTRODUCTION

1.1 Purpose

The purpose of this BSP Programmer Guide is to provide information on how to access the SW, how to set up your own build environment, build the BSP SW from source, and load the resulting binary image onto the dev kit. You can get more information about your Open-Q 820 uSOM Development Kit at http://tech.intrinsyc.com (dev kit registration required).

Please also refer to the Release Notes document (R-3) for more information on this software release.

For more Android-related device information, see the Qualcomm Developer Network page at https://developer.qualcomm.com/get-started/android-development

If you are looking for information on developing applications only, visit https://developer.android.com/studio/index.html

1.2 Scope

This document describes the following for the Open-Q 820 uSOM Development Kit:

- 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 on the kit
- Debug/ADB Usage.

1.3 Intended Audience

This document is intended for end users who have purchased an Open-Q 820 uSOM Development Kit and are interested in Android BSP customization or Linux device driver development.

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 Version Tracking:** This section identifies Android Software version information for the software supplied for use on your Open-Q 820 µSOM Development Kit.
- **Section 4. 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 820 µSOM Development Kit.
- **Section 5. 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 820 µSOM Development Kit.
- **Section 6. Installing an Android Software Image:** This section describes how to install Android software binaries onto your Open-Q 820 µSOM Development Kit
- **Section 7. Advanced Development and Debugging Tips:** This section describes how to configure and control the various subsystems that are part of your Open-Q 820 µSOM Development Kit.
- **Section 8. External References:** This section provides some links to useful references.

1.5 Acronyms

TERM AND ACRONYMS	DEFINITION
ADB	Android Debug Bridge
AMIC	Analog Microphone
ANC	Audio Noise Cancellation
B2B	Board to Board
BLSP	Bus access manager Low Speed Peripheral(Serial interfaces like UART / SPI / I2C/ UIM)
BSP	Board Support Package
CAF	Code Aurora Forum
CSI	Camera Serial Interface
DP	Display Port
DSI	MIPI Display Serial Interface
EEPROM	Electrically Erasable Programmable Read only memory
EMMC	Embedded Multimedia Card
GPS	Global Positioning system
HDMI	High Definition Media Interface
HSIC	High Speed Inter Connect Bus
JTAG	Joint Test Action Group
LNA	Low Noise Amplifier
MIPI	Mobile Industry processor interface
MPP	Multi-Purpose Pin
NFC	Near Field Communication
PID	(USB) Product ID
QHD	Quarter High Definition
RF	Radio Frequency
SATA	Serial ATA
SDK	Software Development Kit
SLIMBUS	Serial Low-power Inter-chip Media Bus
SOM	System On Module
SPMI	System Power Management Interface (Qualcomm PMIC / baseband proprietary protocol)
SSBI	Single wire serial bus interface (Qualcomm proprietary mostly PMIC / Companion chip and baseband processor protocol)
UART	Universal Asynchronous Receiver Transmitter

TERM AND ACRONYMS	DEFINITION
UIM	User Identity module
USB	Universal Serial Bus
USB HS	USB High Speed
USB SS	USB Super Speed
VID	(USB) Vendor ID

1.6 Resources

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

- https://source.android.com/source/initializing.html
- https://developer.android.com/tools/publishing/versioning.html
- https://developer.android.com/studio/releases/platform-tools.html

2. DOCUMENTS

This section lists any parent and supplementary documents for the Open-Q 820 µSOM Development Kit BSP Programmer 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 µSOM 820 Development Kit

2.2 Reference Documents

These documents are available on the Open-Q 820 Technical Portal: http://tech.intrinsyc.com (Dev Kit registration required).

REFERENCE	DOCUMENT NUMBER	AUTHOR	TITLE
R-1	01RND1239-QSG-001	Intrinsyc	Intrinsyc Open-Q µSOM 820 APQ8096 Development Kit Quick Start Guide
R-2	01RND1239-UG-001	Intrinsyc	Intrinsyc Open-Q µSOM 820 APQ8096 Development Kit User Guide
R-3	ITC-01RND 1239-RN-003	Intrinsyc	Intrinsyc Open-Q 820 µSOM Dev Kit Release Notes for Android 9 v5.1

3. SOFTWARE VERSION TRACKING

3.1 Introduction

Software releases from Intrinsyc use a release version based on an underlying Linux Foundation-owned and Qualcomm-maintained software baseline from Code Aurora Forum (CAF), as well as binary images of Qualcomm proprietary software. The Intrinsyc SW packages include additional changes to support specific hardware features of the Open-Q 820 uSOM Development Kit as well as other Intrinsyc-specific SW features.

3.2 Software Version Number Convention

The BSP software version of the Intrinsyc Open-Q 820 uSOM Development Kit is a 2 or 3 digit version number signifying the major and minor software release (e.g., 1.1). This version is for Intrinsyc Internal tracking purposes, and for maintaining bug reports. Based on the version information Intrinsyc can provide better technical support to customers.

3.3 Determining your Software's Version Number

You can check your BSP software version from the display under "Settings->About Phone->Build number" as follows:

```
msm8996-userdebug 9.0 OpenQ820_P_vX.Y
```

[where X= major, Y= minor (bugfix)]

All users who received a board with any previous version of the BSP will have to upgrade the board using the Jflash tool (runs on both Linux and Windows) before upgrading to this new BSP release. This is a one-time upgrade. Please visit https://tech.intrinsyc.com/ to download the latest Open-Q_u820_APQ8096_P_vX.Y.Z-ITC-JFlash.zip file for your board

You will have the following device version info after downloading V5.1 images on your development kit

```
    msm8996-userdebug 9.0 OpenQ820_P_v5.1
```

4. SOURCE CODE ACCESS

4.1 Introduction

This section and the following describe how to download the BSP source code and how to build the Android BSP from source.

Users can build their own BSP software images or they can use pre-built software images from Intrinsyc to program new images on the Open-Q 820 µSOM. If you only wish to program pre-built software images, you can skip the information in this section and go directly to section 6, “Installing an Android Software Image”.

4.2 Downloading the Board Support Package

Users can download the BSP source package from the Files tab of the Open-Q 820 Tech Portal – SW Source Release section.

Download the “8096_Open-Q_u820__Android_BSP-P_vX.Y.Z.zip” to Linux build machine to build from source. Extract the zip file the content of the package would be as below:

```
8096_Open-Q_820_Android_BSP-P_vX.Y.Z
├── 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 image
│   ├── recovery.img      : Recovery image
│   ├── system.img        : Android system partition ext4 image
│   ├── vendor.img        : Android vendor 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 µSOM 820
│   └── propritary.tar.gz   : Qualcomm Proprietary Libraries
├── usb_driver            : Windows Driver for ADB for Open-Q µSOM 820
├── Licence               : Licence Information for Dev platform software update
├── AdbFastboot           : Adb and fastboot binaries
```

After copying the above files to your Linux PC, follow the steps in section 5, "Building an Android BSP” to extract and build the source code.

4.3 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. Intrinsyc-provided patches for fixes and features on the Open-Q development kit are applied on top of this specific CAF baseline via the installation script.

Note:

1. Do not post your board specific or BSP specific technical questions on CAF, as CAF contains many other projects for different families. They will not be answered.
2. For technical support contact https://www.intrinsyc.com/contact-support

5. BUILDING AN ANDROID BSP

5.1 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 820 µSOM Development Kit.

5.2 Development Environment Setup

5.2.1 Introduction

A PC running Ubuntu Linux is required to setup the development environment for building 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 Or Ubuntu 16.04	14.04.x LTS OR 16.04	Ubuntu Community/ Canonical, Ltd.	Android build host OS
3	Open JDK for Linux x64	1.8	Oracle	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. Currently the BSP release supports

   “Ubuntu 14.04 – 64bit” as build environment.
2. Other versions of Ubuntu Linux are not tested against the release build and are unsupported. If you are running any other Ubuntu distribution other than the recommended, you may encounter build errors.
3. If at any point of time, if you encounter errors/issues, report these to https://www.intrinsyc.com/contact-support along with supporting logs.
4. The source code download from codeaurora.org takes up almost 60GB disk space. Full build takes up approx. 140 GB of disk space, so make sure you have necessary disk space before running the build script from the BSP package.

5.2.2 Initializing Build Environment

5.2.2.1 Installing JDK

The Android 9 build requires Open JDK 8. Install the JDK version of Java 8 for Ubuntu 14.04, use below commands

```
$ sudo add-apt-repository ppa:openjdk-r/ppa
$ sudo apt-get update
$ sudo apt-get install openjdk-8-jdk
```

Once installation is complete, select the jdk-8 option

```
$ sudo update-alternatives --config java
```

Select the jdk-8 option out of the alternatives

To confirm you have installed/switched to Java 8, check the version using the command:

```
$ java -version

openjdk version “1.8.0_XX”

To install java for Ubuntu 16.04, use below commands
$sudo apt-get update
$sudo apt-get install openjdk-8-jdk
```

Note: https://source.android.com/setup/initializing contains up-to-date information about the build environment setup. Please refer to this link if you are facing setup issues.

5.2.2.1 Installing Ubuntu packages

The following packages need to be installed for Ubuntu 14.04

sudo apt-get install git-core gnupg flex bison gperf build-essential zip
curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 lib32ncurses5-dev
x11proto-core-dev libx11-dev lib32z-dev libgl1-mesa-dev libxml2-utils xsltproc unzip

Along with the above 14.04 packages, the following additional packages need to be installed for Ubuntu 16.04

sudo apt-get install ccache automake lzop python-networkx bzip2 libbz2-dev libbz2-1.0 libghc-bzlib-dev squashfs-tools pngcrush schedtool dpkg-dev liblz4-tool optipng maven libc6-dev linux-libc-dev g++-5-multilib libssl-dev

5.2.3 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 3 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 https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
   ```
3. Set the repo script's attributes to executable.
   ```
   $ chmod +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”).
   ```

5.3 Downloading and Building Android BSP Images from Source

5.3.1 Introduction

This section describes how to build software for your Open-Q 820 Development Kit from source code and assumes your development workstation has already been setup according to section 5.2.

5.3.2 Build Instructions

Following shows the structure of the Source_Package release.

```
Source_Package  -----------------------------Source code patches
    |-- getSource_and_build.sh  -------------Main download and build script
    |-- patches  ---------------------------Patches for Open-Q 820 support
    |-- intrinsyc.tar.gz  ------------------Intrinsyc Package(Stub at the moment)
    `-- proprietary.tar.gz  ----------------Qualcomm-specific binaries required for Android
```

The source build depends on the following CAF project:

https://www.codeaurora.org/project/android-for-msm

For git / code browsing you can visit below

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

Following are the release table matching the CAF manifest / TAG.

Intrinsyc Release	CAF TAG
8096_microSOM_820_Android_BSP-P-5.1.zip	LA.UM.7.5.r1-03100-8x96.0

To initiate the build, extract the BSP zip file and execute the `getSource_and_build.sh` script located in the Source_Package folder.

```
$ unzip 8096_microSOM_820_Android_BSP-P-5.1.zip
$ cd 8096_microSOM_820_Android_BSP-P-5.1/Source_Package
$ chmod +x getSource_and_build.sh
```

Before running `getSource_and_build.sh`, make sure the build environment meets below requirements:

1) Access internet to download code from Code Aurora
2) Minimum 32 GB RAM recommended for build process
3) **Drive must have at least 140GB free space for complete build.**

Run below command to download the source code and build the BSP:

```
$ ./getSource_and_build.sh
```

```
If USER build is required, the following commands should be run.
$ export BV=user
$ ./getSource_and_build.sh –v ${BV}
```

The script will automatically download source code from CAF, apply patches to respective Android projects, extract propritetary.tar.gz, and build Android.

After the build is complete, for subsequent builds, the following commands should be run from the root of the source (i.e. `APQ8096_LA.UM.7.5.r1-03100-8x96.0_P_v5.1` folder)

```
$ ./build.sh msm8996 –j $(nproc)
```

6. INSTALLING AN ANDROID SOFTWARE IMAGE

6.1 Introduction

This section describes how to install Android software binaries you built / prebuilt onto your Open-Q 820 uSOM Development Kit.

Note: In any following sections `<android-source-tree>` refers to the root Android directory (i.e. `APQ8096_LA.UM.7.5.r1-03100-8x96.0_P_v5.1` folder).

6.2 Fastboot and ADB

6.2.1 Introduction

"fastboot" is the tool used to install an Android image from a Linux (Ubuntu) development PC over a USB connection to the Open-Q µSOM 820 Development Kit. fastboot can also be used to install an android image from Windows machine. Within the SnapDragon 820 embedded software, fastboot is implemented in the LK bootloader.

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 5.3 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 Platform Tools (https://developer.android.com/studio/releases/platform-tools.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 µSOM 820 by configuring the USB VID/PID. To configure the USB VID/PID for the Open-Q µSOM 820 for use with fastboot and adb on your PC, follow the instructions in section 6.2.2 before you connect your PC to your Open-Q µSOM 820.

The development board has J22 (Debug UART) and J24 USB 2.0 connector (adb/fastboot USB) installed. J24 is used for programming with fastboot and for interfacing with adb.

6.2.2 USB Driver Configuration for fastboot and adb on Linux (Ubuntu) Machine

As described in Google's instructions for setting up a hardware device (https://developer.android.com/studio/run/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 Platform Tools.

Here is the configuration required for using adb and fastboot with the kit:

1. Create this file or edit this file as root in the folder `/etc/udev/rules.d/` in your PC:

   ```
   51-android.rules
   ```

   Add the following lines to the end of the file to use platforms with software v3.0:

   ```
   #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"
   ```

   Restart the udev service on your PC using:

   ```
   $ sudo service udev restart
   ```

2. After step 2, you need to connect UART FTDI serial cable (Refer to user guide for connectors) with Minicom or 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
   ```

3. 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 you power the board on
   b. Power on the board and type “`su`” and then “`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:

   ```
   […]
   [170] fastboot_init()
   ```

   This will result in the dev kit rebooting into Fastboot mode

4. To confirm successful communication between the PC and dev kit in Fastboot mode, you can type “`fastboot devices`” to list the devices connected in Fastboot mode or “`lsusb`” shows:

   ```
   18d1:d00d – Unnamed device
   ```

6.2.3 Programming System Images using fastboot

These steps assume your Open-Q µSOM 820 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 6.2.1 and 6.2.2 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 the path on your PC. If not update your .bashrc accordingly.
      ii. Image files: `boot.img, cache.img, persist.img, system.img, userdata.img, recovery.img, vendor.img`
   b. Copy all the above files to a folder on your Linux PC
   c. Make sure the `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 5.3.
4. Obtain the flash programming script "`flashall.sh`" from the pre-built binary image folder from the BSP, you can find it under "8096_microSOM_820_Android_BSP-P-5.1/Binaries"

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

      `<android-source-tree>/out/target/product/msm8996`
   b. Execute the `flashall.sh` script[1]
   c. If the dev kit does not reboot automatically after executing the script, turn off/on your board to reboot your dev kit with the newly programmed images
5. Programming partitions using fastboot:

   To program a partition on the device, use the fastboot command as follows:

   ```
   $ fastboot flash <partition_name> <path to the partition image>
   ```

   This will program partition with image provided.

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

For example, to flash the kernel image,

```
$ fastboot flash boot boot.img
```

To erase a partition, the command is:

```
$ fastboot erase <partition_name>
```

6.2.4 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 Platform Tools (https://developer.android.com/studio/releases/platform-tools.html) on your Windows PC.
2. Edit Environment variable to add adb and fastboot to path.

Install Windows drivers for the dev kit on your Windows PC. These drivers are available with the release from Intrinsyc website, in folder "usb_driver" of 8096_microSOM_820_Android_BSP-P-X.Y.zip package.

Notes:

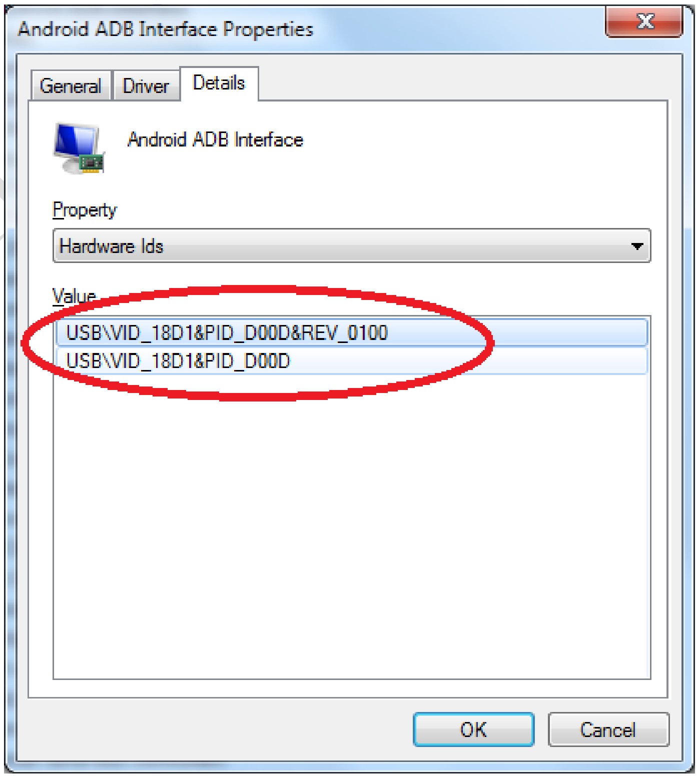
a) If you have installed a previous version of PC drivers from an older release of the dev kit software, you must first uninstall those older drivers first.

b) Terminate any pre-existing adb server on the PC 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:



Android ADB Interface

Property: Hardware Ids

Value:

```
USB\VID_18D1&PID_D00D&REV_0100
USB\VID_18D1&PID_D00D
```

3. To install prebuilt binaries via windows machine just unzip release downloaded from Intrinsyc web site. The image files to be flashed are located in “8096_Open-Q_820_Android_BSP-P_vX.Y.Z\Binaries”

The files are:

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

Once the above steps are complete, you can open place the dev kit into Fastboot mode (refer to section 6.2.2, step 3), open a Windows command prompt from “8096_Open-Q_820_Android_BSP-P_vX.Y.Z\Binaries”

Directory and issue following fastboot commands (or simply run the batch file if supplied)[2]:

OR

3. If you have compiled your own images from the BSP source, as mentioned in Section 5.3, then copy `boot.img, cache.img, persist.img, system.img, userdata.img, recovery.img,` vendor.img files to your windows machine and use following commands from Windows Command prompt.

```
fastboot flash system system.img
fastboot flash cache cache.img
fastboot flash persist persist.img
fastboot flash userdata userdata.img
fastboot flash boot boot.img
fastboot flash vendor vendor.img
fastboot flash recovery recovery.img
fastboot reboot
```

OR

Copy “`flashall.bat`” from “8096_Open-Q_820_Android_BSP-P_vX.Y.Z\Binaries” to the folder where you copied all the above files, and then run the batch file.

Note:

`flashall.bat` file assumes you have adb and fastboot utilities in the path.

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

7. ADVANCED BUILDING TIPS

7.1 Introduction

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

7.2 Reconfiguring / Recompiling and Updating Kernel Image on Device

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

If you don't have ncurses development libraries installed on your build PC, you might get errors while running kernel menuconfig.

To install ncurses:

```
$ sudo apt-get install libncurses5-dev

$ cd <android-source-tree>

$ source build/envsetup.sh

$ lunch msm8996-userdebug

$ env KCONFIG_NOTIMESTAMP=true make -j4 -C kernel/msm-3.18
O=../../out/target/product/msm8996/obj/kernel/msm-3.18 ARCH=arm64
CROSS_COMPILE=aarch64-linux-android- menuconfig

$ env KCONFIG_NOTIMESTAMP=true make -j4 -C kernel/msm-3.18
O=../../out/target/product/msm8996/obj/kernel/msm-3.18 ARCH=arm64
CROSS_COMPILE=aarch64-linux-android- savedefconfig

$ cp out/target/product/msm8996/obj/kernel/msm-3.18/defconfig kernel/msm-
3.18/arch/arm64/configs/msm_defconfig
```

Change your kernel configuration options as needed, and use following command to recompile just the kernel:

```
$ make –j4 bootimage
```

This will build the boot image (kernel) with updated kernel configuration. Flash the boot image as mentioned in Section 6.2.3.

To build system image

```
$ make –j4 systemimage
```

To build userdata image

```
$ make –j4 userdataimage
```

To build vendor image

```
$ make –j4 vendorimage
```

7.3 FAQS for the 820 uSOM BSP

7.3.1 Build configuration

As a development kit, the Open-Q 820 uses the "userdebug" build configuration, and this is the default configuration. This build provides you with "root" access.

Should you wish to rebuild the BSP with other configurations, it is recommended to perform a clean build before a complete recompile.

To clean the Android build, use the command:

1. :

   $ cd <android-source-tree>

   $ make clobber

   ```
   $ ./build.sh msm8996 –j $(nproc)
   ```

7.3.2 Programming newly generated Android images

You can use fastboot to program each newly generated binary, or you can use the provided *flashall.sh* script.

```
$ flashall.sh
```

The flashall script programs the built binary images related to the Android release.

The SOM also contains other proprietary binary images which must agree with your Android release. These other proprietary components are installed using the JFlash tool, and only need to be reprogrammed should you be switching Android baseline versions.

7.3.3 Remounting system partition over adb

Remounting the system partition over adb requires the "userdebug" build configuration

To remount the system partition as writable, use following adb commands on the host PC:

```
$ adb root
$ adb remount
```

8. EXTERNAL REFERENCES

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

Accelerated Computer Vision Processing for Snapdragon Family - Fast CV:
https://developer.qualcomm.com/software/fastcv-sdk

Gaming and Adreno GPU:

https://developer.qualcomm.com/software/adreno-gpu-sdk

Debugging Tools:

https://developer.qualcomm.com/software/trepn-power-profiler

Trepn Profiler for profiling applications

Multimedia and Qualcomm Hexagon DSP SDK:

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