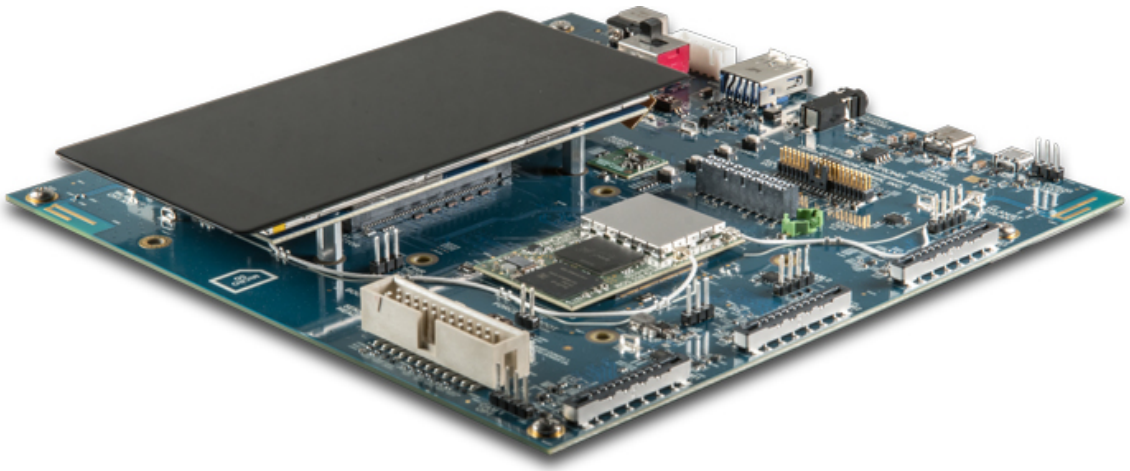
LANTRONIX® Formerly INTRINSYC



Open-Q™ 865XR SOM Development Kit
BSP Programmer Guide
for Android 10 v1.0

Part Number PMD-00078
Revision A August 2020

Contacts

Lantronix, Inc.
7535 Irvine Center Drive, Suite 100
Irvine, CA 92618, USA
Toll Free: 800-526-8766
Phone: 949-453-3990
Fax: 949-453-3995

IES Support
Support: https://helpdesk.intrinsyc.com

Lantronix Technical Support
Online: http://www.lantronix.com/support

Sales Offices
For a current list of our domestic and international sales offices, go to the Lantronix web site at http://www.lantronix.com/about-us/contact/

Revision History

Date	Rev.	Comments
August 2020	A	Initial release

For the latest revision of this product document, please go to: http://tech.intrinsyc.com.

Contents

Contacts 2
Revision History 3

1 Introduction 6
1.1 Purpose 6
1.2 Scope 6
1.3 Intended Audience 6
1.4 Organization 7
1.5 Acronyms 8
1.6 Resources 9

2 Documents 10
2.1 Applicable Documents 10
2.2 Reference Documents 10

3 Software Version Tracking 11
3.1 Introduction 11
3.2 Software Version Number Convention 11
3.3 Determining your Software's Version Number 11

4 Source Code Access 12
4.1 Introduction 12
4.2 Downloading the Board Support Package 12
4.3 Code Aurora Forum (CAF) 13

5 Building an Android BSP 14
5.1 Introduction 14
5.2 Development Environment Setup 14
5.2.1 Introduction 14
5.2.2 Initializing Build Environment 15
5.2.3 repo Installation 16
5.3 Downloading and Building Android BSP Images from Source 17
5.3.1 Introduction 17
5.3.2 Build Instructions 17

6 Installing an Android Software Image 19
6.1 Introduction 19
6.2 Fastboot and ADB 19
6.2.1 Introduction 19
6.2.2 USB Driver Configuration for fastboot and adb on Linux (Ubuntu) Machine 19
6.2.3 Programming System Images using fastboot 20

6.2.4 Fastboot and adb use on a Windows 10 PC 23

7 Advanced Building Tips 24

7.1 Introduction 24
7.2 Reconfiguring / Recompiling and Updating Kernel Image on Device 24
7.3 FAQS for SXR2130P BSP 25
7.3.1 Build Configuration 25
7.3.2 Programming Newly Generated Android Images 25
7.3.3 Remounting System Partition over adb 25
7.3.4 User Data Partition Formatting on First Boot 26
7.3.5 Change Virtual Display Resolution 26
7.3.6 Assign Static IP Address to Ethernet Interface 26

8 External References 27

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 865XR SOM at http://tech.intrinsyc.com (dev kit registration required).

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 developing applications only, visit

https://developer.android.com/studio/index.html

1.2 Scope

This document describes the following for the Open-Q 865XR SOM Development Kit:

- 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 865XR SOM Development Kit and who 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 Version Tracking:** This section identifies Android Software version information for the software supplied for use on your Open-Q 865XR SOM 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 865XR SOM 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 865XR SOM Development Kit
- **Section 6. Installing an Android Software Image:** This section describes how to install Android software binaries onto your Open-Q 865XR SOM 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 865XR SOM Development Kit.
- **Section 8. External References:** This section describes some known problems and suggested solutions.

1.5 Acronyms

TERM AND ACRONYMS	DEFINITION
ADB	Android Debug Bridge
AMIC	Analog Microphone
ANC	Audio Noise Cancellation
B2B	Board to Board
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
QUP	Qualcomm Universal Peripheral (Serial interfaces such as UART/I2C/SPI)
RF	Radio Frequency

TERM AND ACRONYMS	DEFINITION
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
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 865XR SOM Development Kit 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	Lantronix	Purchase and Software License Agreement for the Open-Q 865XR SOM Development Kit

2.2 Reference Documents

REFERENCE	DOCUMENT NUMBER	AUTHOR	TITLE
R-1	N/A	Lantronix	Open-Q 865XR SOM Release Notes
R-2	895-0040-00	Lantronix	Open-Q 865XR SOM Development Kit Quick Start Guide
R-3	PMD-00028	Lantronix	Open-Q 865XR SOM Development Kit User Guide

3 Software Version Tracking

3.1 Introduction

Software releases from Lantronix use a release version based on an underlying Linux Foundation-owned and Qualcomm-maintained software baseline from Code Aurora Forum (CAF).

3.2 Software Version Number Convention

The BSP software version of the Lantronix Open-Q 865XR SOM Development Kit is a 2-digit version number signifying the major and minor software release (e.g., 1.1). This version is for Lantronix Internal tracking purposes, and for maintaining bug reports. Based on the version information Lantronix can provide better support.

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:

```
kona-userdebug 10 QKQ1.200531.002
Open-Q_865_Q_vX.Y test-keys
```

[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.

Visit http://tech.intrinsyc.com to download the latest Open-Q_865_Android-Q_vX.Y_JFlash.zip file for your board.

After upgrading your board, you will have the following device version info:

```
kona-userdebug 10 QKQ1.200531.002
Open-Q_865_Q_v1.0 test-keys
```

Note: *Jflash was last verified on Java version 13. Minimum Java version for Jflash is 1.7.*

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 Lantronix to program new software on the Open-Q 865XR SOM. If you only wish to program pre-built software images, you can skip 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 865XR SOM Tech Portal - Source Package section.

Download the Open-Q_865_Android-Q_vX.Y.zipto a Linux build machine to build from source. After extracting the zip file, the contents of the package should look like:

```
Open-Q_865_Android-Q_vX.Y.zip
├── Binaries                : Precompiled Binaries
│   ├── boot.img            : Linux Kernel + Ramdisk boot image
│   ├── dtbo.img            : Android dtbo partition image
│   ├── flashall.bat        : Script to flash images over fastboot for Windows
│   ├── flashall.sh         : Script to flash images over fastboot
│   ├── metadata.img        : Android metadata encription partition
│   ├── persist.img         : Android persist partition image
│   ├── recovery.img        : Recovery boot partition
│   ├── super.img           : Android super partition, dynamic partitions habitat
│   ├── vbmeta.img          : Android verified boot partition
│   └── vbmeta_system.img  : Android verified system partition
├── 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 865
│   └── proprietary.tar.gz   : Qualcomm Proprietary Libraries
└── Licence                 : Licence Information for Dev platform software update
```

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. Lantronix-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 setup your development workstation (host PC), including the software tools required to build software from source code for your Open-Q 865XR SOM 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 or 16.04 Linux distribution for 64-bit architecture	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	Python	3.6+	https://www.python.org	Python Programming Language
6	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" and Ubuntu 16.04 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 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 250 GB disk space. Full build takes up approx. 400 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 Upgrading Python

To use latest repo needed for this build, python 3.6+ is mandatory. Ubuntu 14.04 and 16.04 are coming with python version less than 3.6. Please upgrade it to any version 3.6 or above.

5.2.2.2 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.*

5.2.2.3 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 openssl libssl-dev
```

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
```

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 865XR SOM Development Kit from source code and assumes your development workstation has already been set up according to section 5.2 , and that you have access to the source code package as described in section 4

5.3.2 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 865XR support
    `-- proprietary.tar.gz  -----------------Qualcomm-specific binaries required for
Android
```

The source build depends on the following CAF project:

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

For git /code browsing you can visit below

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

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

Lantronix Release	CAF TAG
Open-Q_865_Android-Q_v1.0	LA.UM.8.12.r1-11900-sm8250.0

Open-Q_865_Android-Q_v1.0

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

```
$ unzip Open-Q_865_Android-Q_v1.0.zip
$ cd Open-Q_865_Android-Q_v1.0/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 400GB free space for complete build.**

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

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

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. Open-Q_865_Android-Q_v1.0 folder)

```
$ source build/envsetup.sh
$ lunch kona-userdebug
$ ./build.sh dist –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 865XR SOM Development Kit.

Note: *In any following sections* `<android-source-tree>` *refers to the root Android directory (i.e. Open-Q_865_Android-Q_v1.0 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 865XR SOM Development Kit. fastboot can also be used to install an android image from Windows 10 machine. Within the embedded software, fastboot is implemented in the EDK2 UEFI 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 (Ubuntu) and Windows 10 are supplied by Google's Android SDK Platform Tools (https://developer.android.com/studio/releases/platform-tools.html).

Note: *Do not use "apt-get install" to install adb or fastboot on your Linux PC*

Ensure fastboot and adb are in your PATH for your PC.

After that your Windows 10 PC is ready, but you will need one additional step for your Linux PC.

Before you can use fastboot and adb, you must ensure your PC is configured to recognize the Open-Q 865XR SOM Development Kit by configuring the USB VID/PID. To configure the USB VID/PID for the Open-Q 865XR SOM Development Kit 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 865XR SOM Development Kit.

The development board has J1500/J1501 (Debug UART) and J2200 USB 3.1 connector (adb/fastboot USB) installed. J2200 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 v1.0:

```
#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 865XR SOM Development Kit internal storage has a pre-existing Android image configured (kits supplied by Lantronix 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 the 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, dtbo.img, metadata.img, persist.img, recovery.img, super.img, vbmeta.img, vbmeta_system.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 "Open-Q_865_Android-Q_v1.0/Binaries"

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

    `<android-source-tree>/out/target/product/kona`

    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.

    For all partitions that support A/B update use "--slot" option of "fastboot" command to specify which slot should be flashed.

    By default, the active slot will be flashed with provided partition image.

    "—slot all" will flash all partition slots with provided partition image.

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

For example, commands to flash the kernel image could be,

```
$ fastboot flash boot boot.img
$ fastboot --slot all flash boot boot.img
$ fastboot --slot a flash boot boot.img
```

6.2.4 Fastboot and adb use on a Windows 10 PC

It is also possible to use fastboot and adb from a Windows 10 PC for software image programming and debugging. You will need to:

1. Download the Android SDK Platform Tools on your Windows PC. https://developer.android.com/studio/releases/platform-tools)
2. Unzip it somewhere to local drive. It will create platform-tools directory
3. Edit Environment variable to add adb and fastboot to path. .(In enviroment variables set path to <local path>/platform-tools)
4. Place the dev kit into Fastboot mode (refer to section 6.2.2, step 3)
5. Obtain pre-built binaries from release, downloaded from the web site.

   They are located in "Open-Q_865_Android-Q_vX.Y\Binaries" and include

   a. Programming batch utility: `flashall.bat`

      Note: *`flashall.sh` assumes that adb and fastboot android utilities are already in the path on your PC. If not update your .bashrc accordingly*

   b. Image files: `boot.img, dtbo.img, metadata.img, persist.img, recovery.img, super.img, vbmeta.img, vbmeta_system.img`
6. If programming binaries, provided in the release, open a Windows command prompt from "Open-Q_865_Android-Q_vX.Y \Binaries and run the batch file.
7. If you have compiled your own images from the BSP source, as mentioned in Section 5.3

   a. Obtain image files `(boot.img, dtbo.img, metadata.img, persist.img, recovery.img, super.img, vbmeta.img, vbmeta_system.img)` from `<android-source-tree>/out/target/product/kona`

   b. Copy batch flash utility "`flashall.bat`" from Lantronix release "`Open-Q_865_Android-Q_vX.Y`\Binaries" folder `and place it along with your image files`

      Note: *flashall.bat file assumes you have adb and fastboot utilities in the path*

   c. Open Windows command prompt from the same folder and run the batch file.

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

- Change kernel configuration files as needed:

Files are in <android-source-tree>/kernel/msm-4.19/arch/arm64/configs/vendor

`kona_defconfig` – kernel configuration for debug builds

`kona-perf_defconfig` – kernel configuration for release builds

- Use following command to recompile just the kernel image:

```
$ cd <android-source-tree>
$ source build/envsetup.sh
$ lunch kona-userdebug
$ make –j4 bootimage
```

This will build the boot.img (kernel) with updated kernel configuration.

- Flash the boot image as mentioned in Section 6.2.3.

7.3 FAQS for SXR2130P BSP

7.3.1 Build Configuration

Qualcomm recommends using “userdebug” build for the SXR2130P. 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 and build the Android build, use the following command:

```
$ cd <android-source-tree>
$ source build/envsetup.sh
$ lunch kona-userdebug
$ make clobber
$ ./build.sh dist -j $(nproc)
```

Program the new generated images.

7.3.2 Programming Newly Generated Android Images

Android 8.0 and higher includes a reference implementation of Verified Boot called Android Verified Boot (AVB). It verifies image integrity and prevent any unsinged images from execution. Therefore, you need to issue following commands to disable AVB to replace images with your custom-made ones.

```
$ adb root
$ adb disable-verity
$ adb reboot
```

After rebooting, 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
```

7.3.4 User Data Partition Formatting on First Boot

The internal storage device on Open-Q 865XR SOM has been provisioned during manufacture process. After reserving essential system space, the reset of storage is allocated to LUN#0 for user data. The user data partition is not formatted until device first boots up after initial image flashing. File system format is determined by Android during the initialization process. This formatting takes place through a reboot through Android recovery mode.

7.3.5 Change Virtual Display Resolution

We recommend using vysor, a free tool which allows user to access GUI through ADB connection, when there is no LCD mounted. Sometimes vysor cannot determine correct display resolution and results in connection failure. In such case, you can assign the display size by following example command.

```
$ adb shell wm size 1960x1640
```

7.3.6 Assign Static IP Address to Ethernet Interface

The network configuration can be statically assigned to Ethernet interface instead of determining by DHCP. To achieve this, we must manipulate the 'ipconfig.txt' file in Android. Here is an example.

1. Get source code from https://github.com/jhswartz/ipconfigstore and follow the instructions on webpage to build 'ipconfigstore' tool.
2. Edit the configuration file to meet your network environment and save it as 'static.conf'

```
ipAssignment: STATIC
linkAddress: 172.31.0.254/24
gateway: 172.31.0.1
dns: 172.31.0.7
dns: 172.31.0.8
proxySettings: NONE
id: eth0
```

3. Use following commands to transform the configuration for Android.

```
$ ./ipconfigstore -p 3 < static.conf > ipconfig.txt
```

4. Next, update the configuration to your device.

```
$ adb root
$ adb push ipconfig.txt /data/misc/ethernet/
```

5. Reboot your device to apply new settings.
6. To reset the configuration, use following commands:

```
$ adb root
$ adb shell rm /data/misc/ethernet/ipconfig.txt
```

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 - Trepn Profiler for profiling applications:

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

Multimedia and Qualcomm Hexagon DSP SDK:

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