Devboards for Android

Release 0.1

Sumit Semwal

May 03, 2024

CONTENTS

CHAPTER

ONE

INTRODUCTION

There are many dev-boards available in the market, and a great number of interested folks that would like to have Android Open Source Project (AOSP) running on these devices, for various reasons.

The **dev-boards for Android** community initiative is to enable a collaborative space for developers desirous of keeping AOSP running with relevant kernels. The dev-boards can then be used for new feature development for Android, testing and validating them with Android tests like CTS and VTS. They can also be instrumental in sharing and co-developing features like HALs across multiple devices.

We would aim for the dev-boards to boot to UI with AOSP, be testable and usable as development platforms.

It would be great to have device-owner(s) for each device that is enabled in this collaboration.

In the future, we would like to enable as many dev-boards as possible with CI via LAVA, which can be then used to track board status and quality on a rolling basis.

1.1 Goals

- Be an umbrella project for collaboration on dev-boards for Android.
- Help foster a healthy community around AOSP.
- Provide a space for feature-sharing across devices.
- Enable CI testing via LAVA, increasing level of confidence in these devices.

1.2 Software Components

For each supported device, links are made available to the kernel source, local manifests, device specific files and binaries (like bootloader, firmware, HALs) and documentation.

1.3 Community Maintainer(s)

- Sumit Semwal <sumit.semwal@linaro.org>

CHAPTER

TWO

DEVICES SUPPORTED

2.1 Hikey960

Hikey960 until recently was a devboard listed on the AOSP reference boards page. It is now in maintenance phase mostly used to test legacy builds. No further development activity is planned for this devboard.

We are hosting it here as an example of how the devboardsforandroid infrastructure can be used to support devboards.

More details about the devboard can be found at the 96boards page.

- Documentation
- Kernel source code - will be shared soon

Artifacts hosted here

- **device config**
  - This also includes vendor binaries for Hikey960.
- prebuilt kernels
- Local manifest

Build Instructions

```
$ repo init -u https://android.googlesource.com/platform/manifest -b main
$ git clone https://gerrit.devboardsforandroid.linaro.org/platform/manifest .repo/local_
↪manifests/
$ repo sync -j`nproc`
$ source build/envsetup.sh
$ lunch linaro_hikey960-userdebug
$ make -j$(nproc)
```

Artifacts hosted at AOSP (available as of 24 Oct, 2023)

- device config at AOSP
- prebuilt kernels at AOSP
- **Vendor binary blobs**
  - This is required as a binary package when using the AOSP based artifacts.

Build instructions

```
$ repo init -u https://android.googlesource.com/platform/manifest -b main
$ repo sync -j`nproc`
$ ./device/linaro/hikey/fetch-vendor-package.sh
$ source build/envsetup.sh
$ lunch hikey960-userdebug
$ make -j`nproc`
```

2.1.1 Device Maintainer(s)

- Yongqin Liu - <liuyq at #aosp-developers on OFTC IRC>

2.2 RB5

RB5 is one of the supported dev-boards listed on source.android.com.

The vendor-packages required to build RB5 images with AOSP are listed here

2.2.1 Getting started with RB5

Learn about your RB5 as well as how to prepare and set up for basic use from the 96boards RB5 getting started page.

Make sure you are running AOSP (ptable compatible) bootloader on RB5. Latest bootloader binaries (build #29 and above) are hosted here.

For flashing instructions checkout 96boards RB5 recovery page.

Note: You can also update bootloader binaries by running flashall script, which is a part of the vendor package of RB5 AOSP build target. Boot in fastboot mode and run following command from your HOST machine:

```
git clone https://gerrit.devboardsforandroid.linaro.org/linaro-vendor-package
cd src/rb5/rb5-bootloader-ufs-aosp/
./flashall
```

2.2.2 Install pre-built AOSP images on RB5

Linaro creates daily AOSP builds for RB5 that user can download, flash and boot from. If you are interested in prebuilt AOSP images for RB5 and want to avoid compiling your own, please download and flash boot.img, vendor_boot.img, super.img and userdata.img from the snapshot here.

Flash downloaded AOSP images by running following commands, while booted in fastboot mode:

```
fastboot flash userdata userdata.img
fastboot flash super super.img
fastboot flash vendor_boot vendor_boot.img
fastboot flash boot boot.img
```

2.2.3 Compile AOSP from sources for RB5

1. Download the AOSP source tree and build db845c-trunk_staging-userdebug build target:

```
repo init -u https://android.googlesource.com/platform/manifest -b master
repo sync -j`nproc`
./device/linaro/dragonboard/fetch-vendor-package.sh
source ./build/envsetup.sh
lunch db845c-trunk_staging-userdebug #DB845c builds boot on RB5 as well.
make -j`nproc`
```

1. Install: Boot RB5 into fastboot mode and run following command:

```
./device/linaro/dragonboard/installer/rb5/flash-all-aosp.sh
```

You can also perform QDL board recovery by running following script after booting RB5 in USB flashing mode:

```
./device/linaro/dragonboard/installer/rb5/recovery.sh
```

2.2.4 Building the kernel for RB5

The **Preferred** option is to build RB5 Android GKI kernel artifacts using Bazel build. Run the following commands to clone the kernel source, prebuilt Android toolchains and build scripts.

```
mkdir repo-rb5
cd repo-rb5
repo init -u https://android.googlesource.com/kernel/manifest -b common-android-mainline
repo sync -j`nproc`
tools/bazel clean
tools/bazel run //common:db845c_dist
```

Now delete all the objects in $(AOSP_TOPDIR)device/linaro/dragonboard-kernel/android-mainline/, then copy build artifacts from out/db845c/dist/ to $(AOSP_TOPDIR)/device/linaro/dragonboard-kernel/android-mainline/

If you want to properly test the GKI kernel, you should

- grab the latest kernel_aarch64 build from https://ci.android.com/builds/branches/aosp_kernel-common-android-mainline/grid?
- under artifacts, download the Image.gz and copy it to $(AOSP_TOPDIR)/device/linaro/dragonboard-kernel/android-mainline/

Then rebuild AOSP using:

```
make TARGET_KERNEL_USE=mainline -j`nproc`'
```

2.2.5 ToDo / Known Issues

- Factory version of RB5 comes with older version of lt9611uxc firmware flashed on it, so in case you do not see display up and running then please upgrade the lt9611uxc firmware version to v43 or newer.
- WiFi-BT drivers are WIP and not upstreamed yet. You can find the patches here https://git.linaro.org/people/amit.pundir/linux.git/log/?h=rbX-mainline

Device Maintainer(s)

- Amit Pundir <pundir at #aosp-developers on OFTC IRC>

2.3 RB3

RB3 is one of the supported dev-boards listed on source.android.com.

The vendor-packages required to build RB3 images with AOSP are listed here.

2.3.1 Getting started with RB3 (also known as DB845c)

Learn about your RB3 board as well as how to prepare and set up for basic use from the 96boards DB845c getting started page.

Make sure you are running AOSP (ptable compatible) bootloader on DB845c. Latest bootloader binaries (build #97 and above) are hosted here.

For flashing instructions checkout 96boards DB845c recovery page.

Note: You can also update bootloader binaries by running **flashall** script, which is a part of the vendor package of the RB3 AOSP build target. Boot in fastboot mode and run following command from your HOST machine:

```
git clone https://gerrit.devboardsforandroid.linaro.org/linaro-vendor-package
cd src/db845c/dragonboard-845c-bootloader-ufs-aosp/
./flashall
```

2.3.2 Install pre-built AOSP images on RB3

Linaro create daily AOSP builds for DB845c that user can download, flash and boot from. If you are interested in prebuilt AOSP images for DB845c and want to avoid compiling your own, please download and flash boot.img, vendor_boot.img, super.img and userdata.img from the snapshot here.

Flash downloaded AOSP images by running following commands, while booted in fastboot mode:

```
fastboot flash userdata userdata.img
fastboot flash super super.img
fastboot flash vendor_boot vendor_boot.img
fastboot flash boot boot.img
```

2.3.3 Compile AOSP from sources for RB3

1. Download the AOSP source tree and build db845c-trunk_staging-userdebug build target:

```
repo init -u https://android.googlesource.com/platform/manifest -b master
repo sync -j`nproc`
./device/linaro/dragonboard/fetch-vendor-package.sh
source ./build/envsetup.sh
lunch db845c-trunk_staging-userdebug
make -j`nproc`
```

1. Install: Boot DB845c into fastboot mode and run following command:

```
./device/linaro/dragonboard/installer/db845c/flash-all-aosp.sh
```

You can also perform QDL board recovery by running following script after booting DB845c in USB flashing mode:

```
./device/linaro/dragonboard/installer/db845c/recovery.sh
```

2.3.4 Building the kernel for RB3

The **Preferred** option is to build DB845c Android GKI kernel artifacts using official Bazel build. Run the following commands to clone the kernel source, prebuilt Android toolchains and build scripts.

```
mkdir repo-db845c
cd repo-db845c
repo init -u https://android.googlesource.com/kernel/manifest -b common-android-mainline
repo sync -j`nproc`
tools/bazel clean
tools/bazel run //common:db845c_dist
```

Now delete all the objects in $(AOSP_TOPDIR)device/linaro/dragonboard-kernel/android-mainline/, then copy build artifacts from out/db845c/dist/ to $(AOSP_TOPDIR)/device/linaro/dragonboard-kernel/android-mainline/

If you want to properly test the GKI kernel, you should

- grab the latest kernel_aarch64 build from https://ci.android.com/builds/branches/aosp_kernel-common-android-mainline/grid?
- under artifacts, download the Image.gz and copy it to $(AOSP_TOPDIR)/device/linaro/dragonboard-kernel/android-mainline/

Then rebuild AOSP using:

```
make TARGET_KERNEL_USE=mainline -j`nproc`'
```

2.3.5 Booting AOSP from MMC Sdcard

Booting AOSP on DB845c from a mmc sdcard is an experimental build configuration and is only intended to be used in the LKFT lab. Regular users should not enable this build flag and should flash and boot from the UFS instead.

Booting from external sdcards will help prevent the internal emmc/ufs wear off in the long run and extend the lab-life of most of our devboards. To avoid flashing anything on internal UFS and boot solely from a sdcard, we are switching to chainloading U-Boot from ABL bootloader. For now we are using a WIP upstream u-boot fork.

Note: In the long run we plan to switch to AOSP/external/u-boot project to catch up with the Android bootloader features.

Set `TARGET_SDCARD_BOOT=true` at build time to build and boot AOSP from a mmc sdcard. This device configuration need atleast 8GB sdcard to boot from. Here are the instructions to prepare and flash AOSP images on a MMC sdcard:

Note: Following commands that are listed with **=>** prompt means those commands need to be run from the u-boot prompt on the device and commands with **$** means they need to be run from the HOST machine.

- Boot DB845c in the fastboot mode as mentioned above and erase the boot partition to make sure that every reboot/reset lands at the ABL fastboot mode prompt.

```
$ fastboot erase boot
```

- Build U-Boot for DB845c and boot with it:

```
$ git clone https://source.devboardsforandroid.linaro.org/platform/external/u-boot
$ cd u-boot
$ source envsetup.sh
$ mu qcom_defconfig
$ budt dragonboard845c
$ fastboot boot /tmp/u-boot.img   # this will boot U-Boot on DB845c
```

- Prepare AOSP partition layout on the sdcard from the U-Boot prompt. Make sure that a 8GB+ MMC sdcard is plugged into the board:

```
=> run gpt_mmc_aosp
=> reset                          # this will reboot in ABL fastboot mode
$ fastboot boot /tmp/u-boot.img
=> run fastboot                   # starting U-Boot's fastboot command
$ fastboot erase boot erase init_boot erase vendor_boot erase modemst1 erase modemst2␣
↪erase fsg erase fsc erase misc erase metadata erase super erase userdata
$ fastboot reboot                 # rebooting in ABL fastboot mode
$ fastboot boot /tmp/u-boot.img
=> run fastboot
```

- Build AOSP target db845c-userdebug with MMC sdcard support and flash images on the MMC sdcard. Make sure we run U-Boot's fastboot command on the device before running the flash commands:

```
$ make TARGET_SDCARD_BOOT=true -j`nproc`
$ cd out/target/product/db845c
$ fastboot flash super ./super.img flash userdata ./userdata.img format:ext4 metadata␣
↪reboot
$ fastboot boot ./boot.img
```

Note: We do not flash **boot.img** on the sdcard; instead, we load the boot image from device RAM by running `fastboot boot boot.img`.

Device Maintainer(s)

- Amit Pundir <pundir at #aosp-developers on OFTC IRC>

2.4 VIM3

VIM3 is one of the few supported dev-boards listed on source.android.com.

The VIM3 wiki provides links to the documentation for using these devices.

2.4.1 Board status

The VIM3 is build-tested every week:

- CI builds are available at concourse.baylibre.com
- Prebuilt binaries are available at binaries.baylibre.com

These prebuilts are regularly boot-tested to home screen using HDMI.

VIM3 is also used for U-Boot development, to test `fastboot` and Android boot flow changes. The VIM3 board is fully upstream in U-Boot. Documentation can be found on U-Boot's documentation To build for Android, use `configs/khadas-vim3_android_defconfig`.

2.4.2 Device Maintainer(s)

VIM3 is co-maintained by Mattijs and Guillaume from BayLibre.

To reach out to them by email:

- Mattijs Korpershoek <mkorpershoek@baylibre.com>
- Guillaume La Roque <glaroque@baylibre.com>

A mailing list for support is also available at aosp@baylibre.com

Mattijs is also available on IRC: `mkorpershoek` at `#aosp-developers` on OFTC IRC.

2.5 SM8550-HDK

Note: sm8x50-userdebug is an AOSP build target for Snapdragon 8 Gen devboards. It is supported only on v6.8+ kernel versions. Primarily supported and tested on sm8550-hdk. Other devboards (sm8550-qrd and sm8650-qrd) are supported on the best efforts basis.

SM8550-HDK is a Snapdragon 8 Gen 2 Mobile Hardware Development Kit.

2.5.1 Getting started with SM8550-HDK

The power-up sequence for sm8550-hdk is similar to that of RB3 and RB5. The wall power need to be switched ON first and then connect the USB-C to boot automatically without pressing any physical button on the board. Make sure that switch <7, 8> of S5702 is ON before powering ON the device to enable HDMI out.

sm8x50-userdebug boots to UI with v6.8+ android-mainline and upstream kernel using software rendering and linux-firmware binaries for now. *lunch* target is not added yet and will be added as soon the firmware binaries land in linaro-vendor / linux-firmware repo to make sure that the device has more features enabled and is in a more usable state. Boot image header v2 is used for now, while we are working on releasing ABL with header v4 support. Also make sure you are running upstream kernel friendly ABL on SM8550-HDK.

2.5.2 Download and Build Kernel and AOSP images from source for SM8x50 devices

1. Download and build the AOSP source tree for sm8x50-userdebug build target:

```
mkdir $AOSP
cd $AOSP
repo init -u https://android.googlesource.com/platform/manifest -b master
repo sync -j`nproc`
./device/linaro/dragonboard/fetch-vendor-package.sh
source ./build/envsetup.sh
lunch sm8x50-trunk_staging-userdebug
make -j`nproc`
```

1. Flash sm8x50-userdebug AOSP images:

Reboot sm8x50 in fastboot mode and run following commands to flash and boot AOSP:

Disable verity and and erase AOSP partitions for any stale images:

```
$AOSP/external/avb/avbtool.py make_vbmeta_image --flag 2 --padding_size 4096 --output ./
↪vbmeta_disabled.img
fastboot --disable-verity --disable-verification flash vbmeta ./vbmeta_disabled.img
fastboot erase boot dtbo init_boot vendor_boot
fastboot reboot bootloader
```

Flash AOSP images:

```
cd $AOSP/out/target/product/sm8x50/
fastboot flash super ./super.img flash userdata ./userdata.img format:ext4 metadata boot␣
↪./boot.img
```

It should boot sm8x50 devices to UI with software rendering support.

1. Download and build upstream tracking kernel for SM8x50 devices.

```
git clone https://git.linaro.org/people/amit.pundir/linux -b rbX-mainline
cd linux
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- rbX_aosp_defconfig
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- DTC_FLAGS=-@ -j`nproc`
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- INSTALL_MOD_PATH=./modules/ INSTALL_MOD_
↪STRIP=1 modules_install -j`nproc`
cp arch/arm64/boot/Image.gz arch/arm64/boot/dts/qcom/qrb5165-rb5.dtb arch/arm64/boot/dts/
↪qcom/sdm845-db845c.dtb arch/arm64/boot/dts/qcom/sm8550-hdk.dtb arch/arm64/boot/dts/
```

(continues on next page)

(continued from previous page)

```
↪qcom/sm8550-qrd.dtb arch/arm64/boot/dts/qcom/sm8650-qrd.dtb $AOSP/device/linaro/
↪dragonboard-kernel/android-upstream/
find ./modules/lib/ -iname \*.ko -exec cp {} $AOSP/device/linaro/dragonboard-kernel/
↪android-upstream/ \;
```

1. Rebuild sm8x50-userdebug AOSP images with local kernel build:

```
cd $AOSP
source ./build/envsetup.sh
lunch sm8x50-trunk_staging-userdebug
make TARGET_KERNEL_USE=upstream -j`nproc`
```

2.5.3 Known issues and Troubleshooting on sm8550-hdk

1. UFS is not stable. There are a series of known issues that are currently being worked upon.

   For example: Probability of UFS probe running into a hard crash during boot time is very high and it needs a power reset to recover.

   And then there is a run time UFS crash which leaves the device unusable. v6.9-rc1 kernel will hopefully be more stable.

2. At times *fastboot boot* run into the following failure:

```
$ fastboot boot ./boot.img
Sending 'boot.img' (19988 KB)                          OKAY [  0.460s]
Booting                                                FAILED (remote: 'Failed to load/
↪authenticate boot image: Load Error')
fastboot: error: Command failed
```

Run the following set of commands to recover from the above error:

```
$ fastboot reboot bootloader
$ fastboot set_active a boot ./boot.img
```

Device Maintainer(s)

- Amit Pundir <pundir at #aosp-developers on OFTC IRC>

CHAPTER

THREE

CONTRIBUTING

For familiarity and uniformity with the AOSP development process, Gerrit-based review and merge process is followed for dev-boards that have their source code hosted here.

To submit contributions for projects hosted here on devboardsforandroid, please use devboardsforandroid Gerrit server.

For communities around existing devices that already have a process for their contributions, reviewing and merging, we can continue to follow the same.

3.1 Requirements for adding Devboards

We are excited to add Devboards here, but it is important to put some rules for adding and keeping the Devboards in sync.

1. Devboard should have active maintainer(s) that are responsible for them.
2. AOSP/main should be built and boot tested to UI with the Devboard within a 30 day cycle.
3. Mandatory availability of a mechanism that shows the health of the Devboard. It could be CI loop, or something similar.
4. Source code should be available - same as would be expected if the Devboard was in AOSP.
5. Device deprecation: If there is no update or the Devboard is broken with AOSP/main for > 3 months, it would be deprecated. Orphaned Devboards are not useful for the community.

3.2 Interactions

We have a couple of IRC channels on OFTC

- **#linaro-android** to interact with the team at Linaro that works on Android
- **#aosp-developers** as a community go-to place

3.3 Google Groups

A few Google groups around various topics of interest related to AOSP are also a good source of information and troubleshooting:

- Building AOSP
- AOSP Internals
- Porting AOSP
- Contributing to AOSP