NXP Semiconductors
User's Guide

Document Number: AUG
Rev. P9.0.0_2.2.0-ga, 07/2019

Android™ User's Guide

Contents

1 Overview

This document describes how to build Android Pie 9.0 platform for the i.MX 6 and i.MX 7 series devices. It provides instructions for:

- Configuring a Linux® OS build machine.
- Downloading, patching, and building the software components that create the Android™ system image.
- Building from sources and using pre-built images.
- Copying the images to boot media.
- Hardware/software configurations for programming the boot media and running the images.

For more information about building the Android platform, see source.android.com/source/building.html.

2 Preparation

The minimum recommended system requirements are as follows:

- 16 GB RAM
- 300 GB hard disk

2.1 Setting up your computer

To build the Android source files, use a computer running the Linux OS. The Ubuntu 16.04 64bit version and openjdk-8-jdk is the most tested environment for the Android Pie 9.0 build.

After installing the computer running Linux OS, check whether all the necessary packages are installed for an Android build. See "Setting up your machine" on the Android website source.android.com/source/initializing.html.

In addition to the packages requested on the Android website, the following packages are also needed:

```
$ sudo apt-get install uuid uuid-dev
$ sudo apt-get install zlib1g-dev liblz-dev
$ sudo apt-get install liblzo2-2 liblzo2-dev
$ sudo apt-get install lzop
$ sudo apt-get install git-core curl
$ sudo apt-get install u-boot-tools
$ sudo apt-get install mtd-utils
$ sudo apt-get install android-tools-fsutils
$ sudo apt-get install openjdk-8-jdk
$ sudo apt-get install device-tree-compiler
$ sudo apt-get install gdisk
$ sudo apt-get install m4
$ sudo apt-get install libz-dev
```

NOTE

If you have trouble installing the JDK in Ubuntu, see How to install misc JDK in Ubuntu for Android build.

Configure git before use. Set the name and email as follows:

- git config --global user.name "First Last"
- git config --global user.email "first.last@company.com"

2.2 Unpacking the Android release package

After you set up a computer running Linux OS, unpack the Android release package by using the following commands:

```
$ cd ~ # or any other directory you like
$ tar xzvf imx-p9.0.0_2.2.0-ga.tar.gz
```

3 Building the Android platform for i.MX

3.1 Getting Android source code (Android/kernel/U-Boot)

The i.MX Android release source code consists of three parts:

- NXP i.MX public source code, which is maintained in CodeAurora Forum repository.
- AOSP Android public source code, which is maintained in android.googlesource.com.
- NXP i.MX Android proprietary source code package, which is maintained in www.NXP.com.

To generate the i.MX Android release source code build environment, follow the steps below. Assume you had i.MX Android proprietary source code package imx-p9.0.0_2.2.0-ga.tar.gz under the ~/. directory.

```
$ mkdir ~/bin
$ curl https://storage.googleapis.com/git-repo-downloads/repo  > ~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=${PATH}:~/bin
$ source ~/imx-p9.0.0_2.2.0-ga/imx_android_setup.sh
```

```
# By default, the imx_android_setup.sh script will create the source code build environment
in the folder ~/android_build
# ${MY_ANDROID} will be refered as the i.MX Android source code root directory in all i.MX
Android release documentation.
$ export MY_ANDROID=~/android_build
```

3.2 Building Android images

Building the Android image is performed when the source code has been downloaded (Section 3.1) and patched (Section 3.2).

Commands **lunch <buildName-buildType>** to set up the build configuration and **make** to start the build process are executed.

The build configuration command **lunch** can be issued with an argument <Build name>-<Build type> string, such as **lunch sabresd_6dq-userdebug**, or can be issued without the argument presenting a menu of selection.

The Build Name is the Android device name found in the directory ${MY_ANDROID}/device/fsl/. The following table lists the i.MX build names.

Table 1. Build names

Build name	Description
sabreauto_6q	i.MX 6Quad/6DualLite/6QuadPlus SABRE-AI Board
sabresd_6dq	i.MX 6Quad/6DualLite/6QuadPlus SABRE-SD Board and SABRE Platform
sabresd_6sx	i.MX 6SoloX SABRE-SD Board
sabresd_7d	i.MX 7Dual SABRE-SD Board
evk_7ulp	i.MX 7ULP EVKB-SD Board

The build type is used to specify what debug options are provided in the final image. The following table lists the build types.

Table 2. Build types

Build type	Description
user	Production ready image, no debug
userdebug	Production ready image similar with "user" but with root access and debug tools
eng	Development image with debug tools

Android build steps are as follows:

1. Change to the top level build directory.

   ```
   $ cd ${MY_ANDROID}
   ```

2. Set up the environment for building. This only configures the current terminal.

   ```
   $ source build/envsetup.sh
   ```

3. Execute the Android **lunch** command. In this example, the setup is for the production image of i.MX 6Quad SABRE Board/Platform device with user type.

   ```
   $ lunch sabresd_6dq-userdebug
   ```

4. Execute the **make** command to generate the image.

   ```
   $ make 2>&1 | tee build-log.txt
   ```

When the **make** command is complete, the build-log.txt file contains the execution output. Check for any errors.

For BUILD_ID & BUILD_NUMBER changing, update build_id.mk in your ${MY_ANDROID} directory. For details, see the i.MX Android Frequently Asked Questions.

For i.MX 6DualLite SABRE-SD, i.MX 6Quad SABRE-SD, and i.MX 6QuadPlus SABRE-SD boards, the same build configuration is used. They share the same kernel/system/recovery images with the exception of the U-Boot image. The following outputs are generated by default in ${MY_ANDROID}/out/target/product/sabresd_6dq:

- root/: root file system (including init, init.rc). Mounted at /.
- system/: Android system binary/libraries. Mounted at /system.
- data/: Android data area. Mounted at /data.
- recovery/: root file system when booting in "recovery" mode. Not used directly.
- dtbo-imx6q.img: Board's device tree binary for i.MX 6Quad 1 GHz SABRE-SD.
- dtbo-imx6q-ldo.img: Board's device tree binary for i.MX 6Quad 1.2 GHz SABRE-SD.
- dtbo-imx6qp.img: Board's device tree binary for i.MX 6QuadPlus 1 GHz SABRE-SD.
- dtbo-imx6qp-ldo.img: Board's device tree binary for i.MX 6QuadPlus 1.2 GHz SABRE-SD.
- dtbo-imx6dl.img: Board's device tree binary for i.MX 6DualLite SABRE-SD.
- vbmeta-imx6q.img: Android Verify boot metadata image for dtbo-imx6q.img.
- vbmeta-imx6q-ldo.img: Android Verify boot metadata image for dtbo-imx6q-ldo.img.
- vbmeta-imx6qp.img: Android Verify boot metadata image for dtbo-imx6qp.img.
- vbmeta-imx6qp-ldo.img: Android Verify boot metadata image for dtbo-imx6qp-ldo.img.
- vbmeta-imx6dl.img: Android Verify boot metadata image for dtbo-imx6dl.img.
- ramdisk.img: ramdisk image generated from "root/". Not used directly.
- system.img: EXT4 image generated from "system/". It can be programmed to "SYSTEM" partition on SD/eMMC card with "dd".
- recovery-imx6q.img: EXT4 image for i.MX 6Quad 1 GHz SABRE-SD, which is generated from "recovery/". Can be programmed to the "RECOVERY" partition on SD/eMMC card with "dd".
- recovery-imx6q-ldo.img: EXT4 image for i.MX 6Quad 1.2 GHz SABRE-SD, which is generated from "recovery/". Can be programmed to the "RECOVERY" partition on SD/eMMC card with "dd".
- recovery-imx6qp.img: EXT4 image for i.MX 6QuadPlus 1 GHz SABRE-SD, which is generated from "recovery/". Can be programmed to the "RECOVERY" partition on SD/eMMC card with "dd".
- recovery-imx6qp-ldo.img: EXT4 image for i.MX 6QuadPlus 1.2 GHZ SABRE-SD, which is generated from "recovery/". Can be programmed to the "RECOVERY" partition on SD/eMMC card with "dd".
- recovery-imx6dl.img: EXT4 image for i.MX 6DualLite SABRE-SD, which is generated from "recovery/". Can be programmed to the "RECOVERY" partition on SD/eMMC card with "dd".
- partition-table.img: GPT partition table image, used for 8 GB SD card.
- partition-table-14GB.img: GPT partition table image, used for 16 GB SD card.
- partition-table-28GB.img: GPT partition table image, used for 32 GB SD card.
- u-boot-imx6q.imx: U-Boot image with no padding for i.MX 6Quad 0.8/1 GHz SABRE-SD.
- u-boot-imx6q-ldo.imx: U-Boot image with no padding for i.MX 6Quad 1.2 GHz SABRE-SD.
- u-boot-imx6qp.imx: U-Boot image with no padding for i.MX 6QuadPlus 1 GHz SABRE-SD.
- u-boot-imx6qp-ldo.imx: U-Boot image with no padding for i.MX 6QuadPlus 1.2 GHz SABRE-SD.
- u-boot-imx6dl.imx: U-Boot image with no padding for i.MX 6DualLite SABRE-SD.
- u-boot-imx6dl-sabresd-uuu.imx: U-Boot image used by UUU for i.MX 6DualLite SABRE-SD. It is not flashed to MMC.
- u-boot-imx6q-ldo-sabresd-uuu.imx: U-Boot image used by UUU for i.MX 6Quad 1.2 GHz SABRE-SD. It is not flashed to MMC.
- u-boot-imx6q-sabresd-uuu.imx: U-Boot image used by UUU i.MX 6Quad 0.8/1 GHz SABRE-SD. It is not flashed to MMC.
- u-boot-imx6qp-ldo-sabresd-uuu.imx: U-Boot image used by UUU for i.MX 6QuadPlus 1.2 GHz SABRE-SD. It is not flashed to MMC.
- u-boot-imx6qp-sabresd-uuu.imx: U-Boot image used by UUU for i.MX 6QuadPlus 1 GHz SABRE-SD. It is not flashed to MMC.
- vendor.img: Vendor image, which holds platform binaries, mounted at /vendor.
- boot.img: A composite image, which includes the kernel Image, ramdisk, and boot parameters.

NOTE

- To build the U-Boot image separately, see Section Building U-Boot images.
- To build the kernel uImage separately, see Section Building a kernel image.
- To build boot.img, see Section Building boot.img.
- To build dtbo.img, see Section Building dtbo.img.

3.2.1 Configuration examples of building i.MX devices

The following table shows examples of using the `lunch` command to set up different i.MX devices. After the desired i.MX device is set up, the `make` command is used to start the build.

Table 3. i.MX device lunch examples

Build name	Description
i.MX 6DualLite/Quad/QuadPlus SABRE-SD Board and Platform	$ lunch sabresd_6dq-userdebug
i.MX 6Quad/DualLite/QuadPlus SABRE-AI Board	$ lunch sabreauto_6q-userdebug
i.MX 6SoloX SABRE-SD Board	$ lunch sabresd_6sx-userdebug
i.MX 7Dual SABRE-SD Board	$ lunch sabresd_7d-userdebug
i.MX 7ULP EVKB Board	$ lunch evk_7ulp-userdebug

After the `lunch` command is executed, the **make** command is issued:

```
$ make 2>&1 | tee build-log.txt
```

3.2.2 Build mode selection

There are three types of build mode to select: eng, user, and userdebug.

NOTE

To pass CTS, use **user** build mode.

The userdebug build mode should behave the same as the user build mode, with the ability to enable additional debugging that normally violates the security model of the platform. This makes the userdebug build mode good for user test with greater diagnosis capabilities.

The eng build mode prioritizes engineering productivity for engineers who work on the platform. The eng build mode turns off various optimizations used to provide a good user experience. Otherwise, the eng build mode behaves similar to the user and userdebug build modes, so that device developers can see how the code behaves in those environments.

In a module definition, the module can specify tags with `$(LOCAL_MODULE_TAGS)`, which can be one or more values of optional (default), debug, eng.

If a module does not specify a tag with `$(LOCAL_MODULE_TAGS)`, its tag defaults to optional. An optional module is installed only if it is required by product configuration with `PRODUCT_PACKAGES`.

The main differences among the three modes are as follows:

- eng (development configuration with additional debugging tools)
  - Installs modules tagged with: eng and/or debug.
  - Installs modules according to the product definition files, in addition to tagged modules.
  - ro.secure=0
  - ro.debuggable=1

  - ro.kernel.android.checkjni=1
  - adb is enabled by default.
- user (limited access, suitable for production)
  - Installs modules tagged with user.
  - Installs modules according to the product definition files, in addition to tagged modules.
  - ro.secure=1
  - ro.debuggable=0
  - adb is disabled by default.
- userdebug (like user but with root access and debuggability, preferred for debugging)
  - Installs modules tagged with debug.
  - ro.debuggable=1
  - adb is enabled by default.

There are two methods for the build of Android image.

Method 1 : Set the environment first and then execute the make command:

```
$ cd ${MY_ANDROID}
$ source build/envsetup.sh   #set env
$ make -j4 PRODUCT-XXX userdebug 2>&1 | tee build-log.txt
```

XXX depends on different boards. See the following table.

Table 4. Android system image production build method 1

NXP development tool	Description	Image build command
SABRE Board/Platform for Smart Devices	i.MX 6Quad/6QuadPlus/6DualLite	`$ make -j4 PRODUCT-sabresd_6dq-user`
SABRE Board/Platform for Smart Devices	i.MX 6SoloX	`$ make -j4 PRODUCT-sabresd_6sx-user`
SABRE Board/Platform for Smart Devices	i.MX 7Dual	`$ make -j4 PRODUCT-sabresd_7d-user`
SABRE for Automotive Infotainment	i.MX 6Quad/6QuadPlus/6DualLite	`$ make -j4 PRODUCT-sabreauto_6q-user`
Evaluation Kit	i.MX 7ULP	`$ make -j4 PRODUCT-evk_7ulp-userdebug`

Method 2: Set environment first, use lunch command to configure argument (see the following table), and then execute the make command.

An example for the SABRE Board for Smart Devices i.MX 6Dual/Quad is:

```
$ cd ${MY_ANDROID}
$ source build/envsetup.sh
$ lunch sabresd_6dq-user
$ make -j4
```

Table 5. Android system image production build method 2

NXP development tool	Description	Lunch configuration
SABRE Board/Platform for Smart Devices	i.MX 6Quad	sabresd_6dq-user
SABRE Board/Platform for Smart Devices	i.MX 6SoloX	sabresd_6sx-user

Table continues on the next page...

Table 5. Android system image production build method 2 (continued)

SABRE Board/Platform for Smart Devices	i.MX 7Dual	sabresd_7d-user
SABRE for Automotive Infotainment	i.MX 6Quad /6DualLite/6QuadPlus	sabreauto_6q-user
Evaluation Kit	i.MX 7ULP	evk_7ulp-userdebug

To create Android over-the-air, OTA, and package, specify the following make target:

```
$ make otapackage -j4
```

For more Android building information, see source.android.com/source/building.html.

3.3 Building U-Boot images

You can use the following command to generate u-boot.imx under the Android environment.

An example for the i.MX 6Quad/6DualLite and i.MX 6QuadPlus SABRE-AI is as follows:

```
$ cd ${MY_ANDROID}
$ source build/envsetup.sh
$ lunch sabreauto_6q-userdebug
$ make bootloader -j4
```

For other platforms, use "lunch " to set up the build configuration.

For detailed build configuration, see Section 3.2 Building Android images.

3.4 Building a kernel image

Kernel image is automatically built when building the Android root file system.

The following are the default Android build commands to build the kernel image:

```
$ cd ${MY_ANDROID}/vendor/nxp-opensource/kernel_imx
$ echo $ARCH && echo $CROSS_COMPILE

# Make sure you have those 2 environment variables set
# If the two variables have not set, please set the as:
$ export ARCH=arm
$ export CROSS_COMPILE=${MY_ANDROID}/prebuilts/gcc/linux-x86/arm/arm-linux-
androideabi-4.9/bin/arm-linux-androideabi-

# Generate ".config" according to default config file under arch/arm/configs.
# to build the kernel Image for i.MX 6Quad, 6QuadPlus, 6DualLite, 6Solo, 6SoloLite,
6SoloX ,7Dual and 7ULP
$ make imx_v7_android_defconfig
$ make KCFLAGS=-mno-android
```

With a successful build of the above case, the generated kernel images are:

```
${MY_ANDROID}/out/target/product/${buildName}/obj/KERNEL_OBJ/arch/arm/boot/Image
```

3.5 Building boot.img

boot.img and boota are default booting command and image.

As outlined in Running the Android Platform with a Prebuilt Image, we use boota as the default image to boot, not the uramdisk and uImage we used before.

Use this command to generate boot.img under Android environment:

```
# Boot image for the i.MX 6DualLite/6Quad/6QuadPlus SABRE-SD board
$ cd ${MY_ANDROID}
$ source build/envsetup.sh
$ lunch sabresd_6dq-userdebug
$ make bootimage

# Boot image for the i.MX 6DualLite/6Quad/6QuadPlus SABRE-AI board
$ source build/envsetup.sh
$ lunch sabreauto_6q-userdebug
$ make bootimage

# Boot image for the i.MX 6SoloX SABRE-SD board
$ source build/envsetup.sh
$ lunch sabresd_6sx-userdebug
$ make bootimage

# Boot image for i.MX 7Dual SABRE-SD board
$ source build/envsetup.sh
$ lunch sabresd_7d-userdebug
$ make bootimage

# Boot image for i.MX 7ULP EVKB-SD board
$ cd ${MY_ANDROID}
$ source build/envsetup.sh
$ lunch evk_7ulp-userdebug
$ make bootimage -j4
```

3.6 Building dtbo.img

The dtbo image holds the board's device tree binary.

An example for the i.MX 6Dual/6Quad/6QuadPlus SABRE-SD is as follows:

```
$ source build/envsetup.sh
$ lunch sabresd_6dq-userdebug
$ make dtboimage -j4
```

For other platforms, use "lunch <buildName-buildType>" to set up the build configuration.

For detailed build configuration, see Section 3.2 Building Android images.

4 Running the Android Platform with a Prebuilt Image

To test the Android platform before building any code, use the prebuilt images from the following packages and go to "Download Images" and "Boot".

Table 6. Image packages

Image package	Description

Table continues on the next page...

Table 6. Image packages (continued)

android_P9.0.0_2.2.0-ga_image_6qsabresd.tar.gz	Prebuilt image for i.MX 6Quad, i.MX 6QuadPlus, and i.MX 6DualLite SABRE-SD board, which includes NXP extended features.
android_P9.0.0_2.2.0-ga_image_6qsabreauto.tar.gz	Prebuilt image for i.MX 6Quad, i.MX 6QuadPlus, and i.MX 6DualLite SABRE-AI board, which includes NXP extended features.
android_P9.0.0_2.2.0-ga_image_6sxsabresd.tar.gz	Prebuilt image for the i.MX 6SoloX SABRE-SD board, which includes NXP extended features.
android_P9.0.0_2.2.0-ga_image_7dsabresd.tar.gz	Prebuilt image for i.MX 7Dual SABRE-SD board, which includes NXP extended features.
android_p9.0.0_2.2.0-ga_image_7ulpevk.tar.gz	Prebuilt image for i.MX 7ULP EVKB-SD board, which includes NXP extended features.

The following tables list the detailed contents of android_P9.0.0_2.2.0-ga_image_6qsabresd.tar.gz image packages.

The table below shows the prebuilt images to support the system boot from eMMC and SD cards on the i.MX 6Quad SABRE-SD board and platform and i.MX 6DualLite SABRE-SD platform.

Table 7. Images for i.MX 6 SABRE-SD board and platform eMMC boot

SABRE-SD eMMC image	Description
/u-boot-imx6q.imx	The bootloader (with padding) for 800 MHz or 1 GHz i.MX 6Quad SABRE-SD board
/u-boot-imx6dl.imx	The bootloader (with padding) for i.MX 6DualLite SABRE-SD board
/u-boot-imx6q-ldo.imx	The bootloader (with padding) for 1.2 GHZ i.MX 6Quad SABRE-SD board
/u-boot-imx6qp.imx	The bootloader (with padding) for i.MX 6QuadPlus SABRE-SD board
/u-boot-imx6qp-ldo.imx	The bootloader (with padding) for 1.2 GHZ i.MX6QuadPlus SABRE-SD board
/u-boot-imx6dl-sabresd-uuu.imx	U-Boot image used by UUU for i.MX 6DualLite SABRE-SD board
/u-boot-imx6q-ldo-sabresd-uuu.imx	U-Boot image used by UUU for 1.2 GHz i.MX 6Quad SABRE-SD board
/u-boot-imx6q-sabresd-uuu.imx	U-Boot image used by UUU for 800 MHz/1 GHz i.MX 6Quad SABRE-SD board
u-boot-imx6qp-ldo-sabresd-uuu.imx	U-Boot image used by UUU for 1.2 GHz i.MX 6QuadPlus SABRE-SD board
u-boot-imx6qp-sabresd-uuu.imx	U-Boot image used by UUU for 1 GHz i.MX 6QuadPlus SABRE-SD board
/partition-table.img	GPT table image for 8 GB eMMC/SD card
/partition-table-14GB.img	GPT table image for 16 GB eMMC/SD card
/partition-table-28GB.img	GPT table image for 32 GB eMMC/SD card
/dtbo-imx6dl.img	Device tree image for i.MX 6DualLite SABRE-SD board
/dtbo-imx6q-ldo.img	Device tree image for 1.2 GHz i.MX6Quad SABRE-SD board
/dtbo-imx6q.img	Device tree image for 800 MHz/1 GHz i.MX6Quad SABRE-SD board

Table continues on the next page...

Table 7. Images for i.MX 6 SABRE-SD board and platform eMMC boot (continued)

/dtbo-imx6qp-ldo.img	Device tree image for 1.2 GHz i.MX 6QuadPlus SABRE-SD board
/dtbo-imx6qp.img	Device tree image for 1 GHz i.MX 6QuadPlus SABRE-SD board
/boot.img	Boot image
/system.img	System Boot Image
/recovery-imx6dl.img	Recovery Image for i.MX 6Solo/6DualLite SABRE-SD board
/recovery-imx6q.img	Recovery Image for 800 MHz or 1 GHz i.MX 6Quad SABRE-SD board
/recovery-imx6q-ldo.img	Recovery Image for 1.2 GHZ i.MX 6Quad SABRE-SD board
/recovery-imx6qp.img	Recovery Image for i.MX 6QuadPlus SABRE-SD board
/recovery-imx6qp-ldo.img	Recovery Image for 1.2 GHz i.MX 6QuadPlus SABRE-SD board
/vendor.img	Vendor image
/vbmeta-imx6dl.img	vbmeta image for i.MX 6DualLite SABRE-SD board
/vbmeta-imx6q-ldo.img	vbmeta image for 1.2 GHz i.MX 6Quad SABRE-SD board
/vbmeta-imx6q.img	vbmeta image for 800 MHz/1 GHz i.MX 6Quad SABRE-SD board
/vbmeta-imx6qp-ldo.img	vbmeta image for 1.2 GHz i.MX 6QuadPlus SABRE-SD board
/vbmeta-imx6qp.img	vbmeta image for 1 GHz i.MX6QuadPlus SABRE-SD board

The following tables list the detailed contents of android_P9.0.0_2.2.0-ga_image_6qsabreauto.tar.gz image packages.

The table below shows the prebuilt images to support the system boot from SD on the i.MX 6Quad, i.MX 6QuadPlus, and i.MX 6Solo/6DualLite SABRE-AI boards.

Table 8. Images for i.MX 6 SABRE-AI SD boot

SABRE-AI SD image	Description
/u-boot-imx6q.imx	Bootloader (with padding) for i.MX 6Quad SABRE-AI SD boot
/u-boot-imx6dl.imx	Bootloader (with padding) for i.MX 6Solo/6DualLite SABRE-AI SD boot
/u-boot-imx6qp.imx	Bootloader (with padding) for i.MX 6QuadPlus SABRE-AI SD boot
/u-boot-imx6dl-sabreauto-uuu.imx	U-Boot image used by UUU for i.MX 6DualLite SABRE-AI
/u-boot-imx6q-sabreauto-uuu.imx	U-Boot image used by UUU for i.MX 6Quad SABRE-AI
/u-boot-imx6qp-sabreauto-uuu.imx	U-Boot image used by UUU for i.MX 6QuadPlus SABRE-AI
/partition-table.img	GPT table image for 8 GB SD card
/partition-table-14GB.img	GPT table Image for 16 GB SD card
/partition-table-28GB.img	GPT table Image for 32 GB SD card
/boot.img	Boot image
/dtbo-imx6q.img	Device tree image for i.MX 6Quad SABRE-AI SD boot
/dtbo-imx6dl.img	Device tree image for i.MX 6DualLite SABRE-AI SD boot
/dtbo-imx6qp.img	Device tree image for i.MX 6QuadPlus SABRE-AI SD boot

Table continues on the next page...

Table 8. Images for i.MX 6 SABRE-AI SD boot (continued)

/system.img	System Boot Image
/recovery-imx6q.img	Recovery image for i.MX 6Quad SABRE-AI SD boot
/recovery-imx6dl.img	Recovery image for i.MX 6DualLite SABRE-AI SD boot
/recovery-imx6qp.img	Recovery image for i.MX 6QuadPlus SABRE-AI SD boot
/vendor.img	Vendor Image
/vbmeta-imx6q.img	vbmeta image for i.MX 6Quad SABRE-AI SD boot
/vbmeta-imx6dl.img	vbmeta image for i.MX 6DualLite SABRE-AI SD boot
/vbmeta-imx6qp.img	vbmeta image for i.MX 6QuadPlus SABRE-AI SD boot

The following tables list the detailed contents of the android_P9.0.0_2.2.0-ga_image_6sxsabresd.tar.gz image package.

The table below shows the prebuilt images to support the system boot from SD on the i.MX 6SoloX SABRE-SD board.

Table 9. Images for i.MX 6SoloX SABRE-SD SD boot

i.MX 6SoloX SABRE-SD SD image	Description
/u-boot-imx6sx.imx	Bootloader (with padding) for the i.MX 6SoloX SABRE-SD board
/u-boot-imx6sx-sabresd-uuu.imx	U-Boot image used by UUU for i.MX 6SoloX SABRE-SD board
/partition-table.img	GPT table image for 8 GB SD card
/partition-table-14GB.img	GPT table image for 16 GB SD card
/partition-table-28GB.img	GPT table Image for 32 GB SD card
boot.img	Boot image
/system.img	System boot image
/recovery-imx6sx.img	Recovery image for the i.MX 6SoloX SABRE-SD board
/vendor.img	Vendor image
/dtbo-imx6sx.img	Device tree image for i.MX 6SoloX SABRE-SD board
/vbmeta-imx6sx.img	vbmeta image

The following table lists the detailed contents of android_P9.0.0_2.2.0-ga_image_7dsabresd.tar.gz image package.

The table below shows the prebuilt images to support the system boot from SD on i.MX 7Daul SABRE-SD boards.

Table 10. Images for i.MX 7Dual SABRE-SD SD boot

i.MX 7Dual SABRE-SD SD image	Description
/u-boot-imx7d.imx	Bootloader (with padding) for the i.MX 7Dual SABRE-SD board
u-boot-imx7d-sabresd-uuu.imx	U-Boot image used by UUUfor i.MX 7Dual SABRE-SD board
/partition-table.img	GPT table image for 8 GB SD card
/partition-table-14GB.img	GPT table Image for 16 GB SD card
/partition-table-28GB.img	GPT table Image for 32 GB SD card
boot.img	Boot image
/system.img	System boot image

Table continues on the next page...

Table 10. Images for i.MX 7Dual SABRE-SD SD boot (continued)

/recovery-imx7d.img	Recovery image for the i.MX 7Dual SABRE-SD board
/vendor.img	Vendor image
/dtbo-imx7d.img	Device tree image for i.MX 7Dual SABRE-SD board
/vbmeta-imx7d.img	vbmeta image

The following tables list the detailed contents of android_p9.0.0_2.2.0-ga_image_7ulpevk.tar.gz image package.

The table below shows the prebuilt images to support the system boot from SD on i.MX 7ULP EVKB Rev. A boards.

Table 11. Images for i.MX 7ULP EVKB SD boot

i.MX 7ULP EVKB-SD images	Descriptions
/u-boot-imx7ulp.imx	Bootloader (with padding) for i.MX 7ULP EVKB-SD board
/u-boot-imx7ulp-evk-uuu.imx	U-Boot image used by UUU for i.MX 7ULP EVKB-SD board
/imx7ulp_m4_demo.img	Prebuilt image for Cortex-M4 core on SoC
/partition-table.img	GPT table image for 8 GB SD card
/partition-table-14GB.img	GPT table image for 16 GB SD card
/partition-table-28GB.img	GPT table image for 32 GB SD card
/boot.img	Boot Image for i.MX 7ULP EVKB-SD board
/dtbo-imx7ulp.img	Device tree image for i.MX 7ULP EVKB-SD board
/system.img	System Boot image
/recovery-imx7d.img	Recovery Image for i.MX 7ULP EVKB-SD board
/vendor.img	Vendor image
/vbmeta.img	Vbmeta image

NOTE

boot.img is an Android image that stores zImage and ramdisk together. It also stores other information such as the kernel boot command line, machine name.

This information can be configured in android.mk. It is used to override any bootloader default boot arguments without changing it in the bootloader source code.

5 Programming Images

The images from the prebuilt release package or created from source code contain the U-Boot boot loader, system image, and recovery image. At a minium, the storage devices on the development system (MMC/SD) must be programmed with the U-Boot boot loader. The i.MX 6 series boot process determines what storage device to access based on the switch settings. When the boot loader is loaded and begins execution, the U-Boot environment space is then read to determine how to proceed with the boot process. For U-Boot environment settings, see Section Booting.

The following download methods can be used to write the Android System Image:

- UUU to download all images to the eMMC/SD card.
- fsl-sdcard-partition.sh to download all images to the SD card.
- fastboot_imx_flashall script to download all images to the eMMC/SD storage.

5.1 System on MMC/SD

The images needed to create an Android system on MMC/SD can either be obtained from the release package or be built from source.

The images needed to create an Android system on MMC/SD are listed below:

- U-Boot image: u-boot.imx
- boot image: boot.img
- dtbo image: dtbo.img
- Android verify image: vbmeta.img
- Android system image: system.img
- Recovery image: recovery.img
- GPT table image: partition-table.img
- Vendor image: vendor.img

5.1.1 Storage partitions

The layout of the MMC/SD/TF card for Android system is shown below:

- [Partition type/index] which is defined in the GPT.
- [Name] is only meaningful in the Android platform. Ignore it when creating these partitions.
- [Start Offset] shows where partition is started, unit in MB.

The SYSTEM partition is used to put the built Android system image. The DATA is used to put applications' unpacked codes/data, system configuration database, etc. In normal boot mode, the root file system is mounted from uramdisk. In recovery mode, the root file system is mounted from the RECOVERY partition.

Table 12. Storage partitions

Partition type/ index	Name	Start offset	Size	File system	Content
N/A	bootloader	1 KB	4 MB	N/A	bootloader
1	dtbo	2 MB	4 MB	N/A	dtbo.img
2	boot	Follow dtbo	48 MB	boot.img format, a kernel + ramdisk	boot.img
3	recovery	Follow Boot	48 MB	boot.img format, a kernel + ramdisk	recovery.img
4	system	Follow Recovery	1536 MB	EXT4. Mount as / system	Android system files under `/system/ dir`
5	cache	Follow SYSTEM	512 MB	EXT4. Mount as / cache	Android cache for image store of OTA
6	misc	Follow CACHE	4 MB	N/A	For recovery storage bootloader message, reserve
7	datafooter	Follow Misc	2 MB	N/A	For crypto footer of DATA partition encryption
8	metadata	Follow Dtafootor	2 MB	N/A	For system slide show

Table continues on the next page...

Table 12. Storage partitions (continued)

9	presister	Follow Metadata	1 MB	N/A	Option to operate unlock \unlock
10	vendor	Follow PRESISTER	112 MB	EXT4. Mount as / vendor	vendor.img
11	data	Follow VENDOR	Total-Other images	EXT4. Mount at / data	Application data storage for system application. And for internal media partition, in `/mnt/sdcard/ dir`
12	fbmisc	Follow Data	1 MB	N/A	To store the state of lock \unlock

To create these partitions, use UUU described in the *Android Quick Start Guide* (AQSUG), or use script fsl-sdcard-partition.sh in the source code directory.

The script below can be used to partition an SD card as shown in the partition table above:

```
$ cd ${MY_ANDROID}/
$ sudo ./device/fsl/common/tools/fsl-sdcard-partition.sh -f <soc_name> /dev/sdX
# <soc_name> can be as imx6qp, imx6q, imx6dl, imx6sx, imx7d, and imx7ulp
```

NOTE

- The minimum size of the SD card is 8 GB.
- If the SD card is 8 GB, use `sudo ./device/fsl/common/tools/fsl-sdcard-partition.sh -f <soc_name> /dev/sdX` to flash images.
- If the SD card is 16 GB, use `sudo ./device/fsl/common/tools/fsl-sdcard-partition.sh -f <soc_name> -c 14 /dev/sdX` to flash images.
- If the SD card is 32 GB, use `sudo ./device/fsl/common/tools/fsl-sdcard-partition.sh -f <soc_name> -c 28 /dev/sdX` to flash images.
- /dev/sdX, the X is the disk index from 'a' to 'z'. That may be different on each computer running Linux OS.
- Unmount all the SD card partitions before running the script.
- Put the related bootloader, boot image, system image, recovery image, GPT table image, and vendor image in your current directory. This script requires to install the simg2img tool on the computer. simg2img is a tool that converts the sparse system image to raw system image on the Linux OS host computer. The android-tools-fsutils package includes the simg2img command for Ubuntu Linux OS.
- The Cortex-M4 image of i.MX 7ULP EVKBSD should be in serial flash on board, so this script cannot flash the Cortex-M4 image of i.MX 7ULP EVKB-SD.

5.1.2 Downloading images with UUU

UUU can be used to download all images into a target device. It is a quick and easy tool for downloading images. See *Android™ Quick Start Guide* (AQSUG) for a detailed description of UUU.

5.1.3 Download images with fastboot_imx_flashall script

UUU can be used to flash the Android System Image into the board, but you need to make the board enter serial down mode first, and make the board enter boot mode when flashing is finished.

There is another tool of fastboot_imx_flashall script, which uses fastboot to flash the Android System Image into the board. It requires the target board be able to enter fastboot mode and the device is unlocked. It does not need to change the boot mode with this fastboot_imx_flashall script.

The table below describes the fastboot_imx_flashall script.

Table 13. fastboot_imx_flashall script

Name	Host system to execute the script
fastboot_imx_flashall.sh	Linux OS
fastboot_imx_flashall.bat	Windows OS

With the help of fasboot_imx_flashall related scripts, you do not need to use fastboot to flash Android images manually one by one. These scripts will automatically flash all images with only one line of command.

fastboot can be built with the Android build system. Based on Section 3, which describes how to build Android images, perform the following steps to build fastboot:

```
$ cd ${MY_ANDROID}
$ make -j4 fastboot
```

After the build process finishes building fastboot, the directory to find the fastboot is as follows:

- Linux version binary file: `${MY_ANDROID}/host/linux-x86/bin/`
- Windows version binary file: `${MY_ANDROID}/host/windows-x86/bin/`

The way to use these scripts is as follows:

- Linux shell script usage: `sudo fastboot_imx_flashall.sh <option>`
- Windows batch script usage: fastboot_imx_flashall.bat <option>

Table 14. Script options

Option	Decription
-h	Displays the help message.
-f soc_name	Flashes Android image file with soc_name.
-c card_size	Optional setting: 7 / 14 / 28 • If it is not set, use partition-table.img (default). • If it is set to 7, use partition-table-7GB.img for 8 GB SD card. • If it is set to 14, use partition-table-14GB.img for 16 GB SD card. • If it is set to 28, use partition-table-28GB.img for 32 GB SD card. Make sure the corresponding file exists for your platform.
-m	Flashes the Cortex-M4 image.
-d dev	Flashes dtbo, vbmeta, and recovery image file with dev. If it is not set, use the default dtbo, vbmeta, and recovery image
-e	Erases the user data after all image files are flashed.
-l	Locks the device after all image files are flashed.
-D directory	Directory of images. If this script is executed in the directory of the images, it does not need to use this option.
-s ser_num	Serial number of board. If only one board is connected to computer, it does not need to use this option

NOTE

- -f option is mandatory, soc_name can be imx6qp, imx6q, imx6dl, imx6sx, imx7d, and imx7ulp.
- Boot the device to U-Boot fastboot mode, and then execute these scripts. Device should be unlocked first.

Example and option explanations:

```
sudo ./fastboot_imx_flashall.sh -f imx7ulp -m -D /imx_pi9.0/evk_7ulp/
```

- -f imx7ulp: flashes images for i.MX 7ULP EVKB Rev. A Board.
- -m: Cortex-M4 image "imx7ulp_m4_demo.img" is flashed.
- -D /imx_pi9.0/evk_7ulp/: images to be flashed are in the directory of /imx_pi9.0/evk_7ulp/.

6 Booting

6.1 Booting from eMMC/SD

6.1.1 Booting from MMC/SD on the i.MX 6QuadPlus/6Quad/6DualLite SABRE-SD board

This section contains boot switch information and steps needed to bootup from MMC/SD.

The following table lists the boot switch settings for different boot methods.

Table 15. Boot switch settings

Download mode (UUU mode)	(SW6) 00001100 (from 1-8 bit)
eMMC 4-bit (MMC2) boot	(SW6) 11100110 (from 1-8 bit)
eMMC 8-bit (MMC2) boot	(SW6) 11010110 (from 1-8 bit)
SD2 boot	(SW6) 10000010 (from 1-8 bit)
SD3 boot	(SW6) 01000010 (from 1-8 bit)

To boot from eMMC, change the board boot switch to (SW6) 11100110 (from 1-8 bit).

To boot from SD card, change the board boot switch to (SW6) 01000010 (from 1-8 bit).

The default environment is in boot.img. To use the default environment in boot.img, do not set bootargs environment in U-Boot.

To clear the bootargs environment being set and saved previously, use the following commands:

```
U-Boot > setenv bootargs
U-Boot > saveenv        #Save the environments
```

NOTE

bootargs env is an optional setting for boota. The boot.img includes a default bootargs, which is used if there is no definition of the bootargs environment.

Some SoCs on SABRE-SD boards do not have MAC address fused. Therefore, to use FEC in U-Boot, set the following environment:

```
U-Boot > setenv ethaddr 00:04:9f:00:ea:d3       #set up the MAC
address
U-Boot > setenv fec_addr 00:04:9f:00:ea:d3      #set up the MAC
address
```

6.1.2 Booting from SD on the the i.MX 6Quad/6DualLite/6QuadPlus SABRE-AI board

This section contains boot switch information and steps needed to bootup from SD.

The following table lists the boot switch settings for different boot methods on i.MX 6 series SABRE-AI boards.

Table 16. Boot switch settings

Download mode (UUU mode)	(S3) 0101 (from 1-4 bit)
SD on CPU Board	(S1) 0100100000 (from 1-10 bit) (S2) 0010 (from 1-4 bit) (S3) 0010 (from 1-4 bit)

To boot from SD, perform the following operations:

Change the board boot switch to (S3, S2, S1) 0010, 0010,0100100000 (from 1 bit).

The default environment is in boot.img. To use the default environment in boot.img, do not set bootargs environment in U-Boot.

To clear the bootargs environment being set and saved previously, use the following command:

```
U-Boot > setenv bootargs
U-Boot > saveenv   #[Save the environments]
```

NOTE

bootargs environment is an optional setting for Android boot. The boot.img includes a default bootargs, which is used if there is no definition of the bootargs environment.

Some SoCs on SABRE-AI boards do not have MAC address fused. Therefore, to use FEC in U-Boot, set the following environment:

```
U-Boot > setenv ethaddr 00:04:9f:00:ea:d3       #set up the MAC
address
U-Boot > setenv fec_addr 00:04:9f:00:ea:d3      #set up the MAC
address
```

6.1.3 Booting from SD on the i.MX 6SoloX SABRE-SD board

This section contains boot switch information and steps needed to bootup from SD.

The following table lists the boot switch settings used to control the boot storage.

Table 17. Boot switch settings

Download mode (UUU mode)	SW10: 00000000 (from 1-8 bit)

Table continues on the next page...

Table 17. Boot switch settings (continued)

	SW11: 00111000 (from 1-8 bit) SW12: 01000000 (from 1-8 bit) Boot_Mode: 10 (from 1-2 bit)
SD boot	SW3: 00000000 (from 1-8 bit) SW4: 00111000 (from 1-8 bit) SW5: 01000000 (from 1-8 bit) Boot_Mode: 01 (from 1-2 bit)

To boot from SD, perform the following operations:

Change the board Boot_Mode switch to 01 (from 1-2 bit) and (SW10, 11, 12) 00000000 00111000 01000000 (from 1-8 bit).

The default environment is in boot.img. To use the default environment in boot.img, do not set bootargs environment in U-Boot.

To clear the bootargs environment being set and saved previously, use the following command:

```
U-Boot > setenv bootargs
U-Boot > saveenv                          [Save the environments]
```

NOTE

bootargs environment is an optional setting for Android boot. The boot.img file includes a default bootargs, which is used if there is no definition about the bootargs env.

Due to some SoCs on the SABRE-SD boards, do not fuse MAC addres. Set the following environment to use FEC in U-Boot:

```
U-Boot > setenv ethaddr 00:04:9f:00:ea:d3       #set up the MAC
address
U-Boot > setenv fec_addr 00:04:9f:00:ea:d3      #set up the MAC
address
```

6.1.4 Booting from SD on the i.MX 7Dual SABRE-SD board

The following table lists the boot switch settings used to control the boot storage.

Table 18. Boot switch settings

Download mode (UUU mode)	SW4: 00100000 (from 1-8 bit) Boot_Mode: 01 (from 1-2 bit)
SD boot	SW4: 00111000 (from 1-8 bit) Boot_Mode: 10 (from 1-2 bit)

To boot from SD, perform the following operations:

Change the board Boot_Mode switch to 10 (from 1-2 bit) and SW4 00100000 (from 1-8 bit).

The default environment is in boot.img. To use the default environment in boot.img, do not set bootargs environment in U-Boot.

To clear the bootargs environment being set and saved previously, use the following command:

```
U-Boot > setenv bootargs
U-Boot > saveenv                          [Save the environments]
```

NOTE

bootargs environment is an optional setting for Android boot. The boot.img file includes a default bootargs, which is used if there is no definition about the bootargs env.

Due to some SoCs on the SABRE-SD boards, do not fuse MAC addres. Set the following environment to use FEC in U-Boot:

```
U-Boot > setenv ethaddr 00:04:9f:00:ea:d3       #set up the MAC
address
U-Boot > setenv fec_addr 00:04:9f:00:ea:d3      #set up the MAC
address
```

6.1.5 Booting from SD on the i.MX 7ULP EVKB-SD board

The following table lists the boot switch settings used to control the boot storage.

Table 19. Boot switch settings

Boot switch	Download mode (UUU mode)	SD boot
SW4 (from 1-4 bit)	01xx	1001

To boot from SD, perform the following operations:

Change the board SW1 switch to 1001 (from 1-4bit).

The default environment is in boot.img. To use the default environment in boot.img, do not set bootargs environment in U-Boot.

To clear the bootargs environment being set and saved previously, use the following command:

```
U-Boot > setenv bootargs
U-Boot > saveenv                          [Save the environments]
```

NOTE

bootargs environment is an optional setting for Android boot. The boot.img file includes a default bootargs, which is used if there is no definition about the bootargs env.

Due to some SoCs on the SABRE-SD boards, do not fuse MAC addres. Set the following environment to use FEC in U-Boot:

```
U-Boot > setenv ethaddr 00:04:9f:00:ea:d3       #set up the MAC
address
U-Boot > setenv fec_addr 00:04:9f:00:ea:d3      #set up the MAC
address
```

6.2 Boot-up configurations

This section explains the common U-Boot like U-Boot environments, kernel command line, and DM-verity configuartions.

6.2.1 U-Boot environment

If you do not define the bootargs environment, it uses the default bootargs inside the image.

- bootcmd is the first variable to run after U-Boot boot.
- bootargs is the kernel command line, which the bootloader passes to the kernel. As described in Kernel command line (bootargs), bootargs environment is optional for boota. boot.img already has bootargs. If you do not define the bootargs environment, it uses the default bootargs inside the image.

  To use the default environment in boot.img, use the following command to clear the bootargs environment.

  ```
  > setenv bootargs
  ```
- dhcp: get the IP address by BOOTP protocol, and load the kernel image ($bootfile env) from the TFTP server.
- boota:

  boota command parses the boot.img header to get the zImage and ramdisk. It also passes the bootargs as needed (it only passes bootargs in boot.img when it cannot find "bootargs" var in your U-Boot environment). To boot the system, do the following:

  ```
  > boota
  ```

  To boot into recovery mode, execute the following command:

  ```
  > boota recovery
  ```

  If you have read the boot.img into memory, use this command to boot from

  ```
  > boota 0xXXXXXXXX
  ```

6.2.2 Kernel command line (bootargs)

Depending on the different booting/usage scenarios, you may need different kernel boot parameters set for bootargs.

Table 20. Kernel boot parameters

Kernel parameter	Description	Typical value	Used when
console	Where to output kernel log by printk.	console=ttymxc0,115200	All use cases.
init	Tells kernel where the init file is located.	init=/init	All use cases. "init" in the Android platform is located in "/" instead of in "/sbin".
video	Tells kernel/driver which resolution/ depth and refresh rate should be used, or tells kernel/driver not to register a framebuffer device for a display device.	video=mxcfb0:dev=ldb,LDB-XGA,if=RGB666,bpp=32 or video=mxcfb1:dev=hdmi, 1920x1080M@60,if=RGB24,bpp=32 or video=mxcfb2:off	To specify a display framebuffer with: video=mxcfb<0,1,2>:dev=<ldb,hdmi>,<LDB-XGA,xres x yresM@fps>,if=<RGB666,RGB24>,bpp=<16,32> or To disable a display device's framebuffer register with: video=mxcfb<0,1,2>:off
vmalloc	vmalloc virtual range size for kernel.	vmalloc=128M	vmalloc=<size>

Table continues on the next page...

Table 20. Kernel boot parameters (continued)

Kernel parameter	Description	Typical value	Used when
androidboot.console	The Android shell console. It should be the same as console=.	androidboot.console=ttymxc0	To use the default shell job control, such as Ctrl+C to terminate a running process, set this for the kernel.
cma	CMA memory size for GPU/VPU physical memory allocation.	cma=320M	It is 320 MB by default.
androidboot.selinux	Argument to disable selinux check and enable serial input when connecting a host computer to the target board's USB UART port. For details about selinux, see Security-Enhanced Linux in Android.	androidboot.selinux=permissive	Android Pie 9.0 CTS requirement: The serial input should be disabled by default. Setting this argument enables console serial input, which violates the CTS requirement.
loop.max_part	Defines how many partitions that each loop device can manage.	loop.max_part=7	-

6.2.3 DM-verity configuration

DM-verity (device-mapper-verity) provides transparent integrity checking of block devices. It can prevent device from running unauthorized images. This feature is enabled by default. Replacing one or more partitions (boot, vendor, system, vbmeta) will make the board unbootable. Disabling DM-verity provides convience for developers, but the device is unprotected.

To disable DM-verity, perform the following steps:

1. Unlock the device.
   a. Boot up the device.
   b. Choose **Settings** -> **Developer Options** -> **OEM Unlocking** to enable OEM unlocking.
   c. Execute the following command on the target side to make the board enter fastboot mode:

      ```
      reboot bootloader
      ```

   d. Unlock the device. Execute the following command on the host side:

      ```
      fastboot oem unlock
      ```

   e. Wait until the unlock process is complete.
2. Disable DM-verity.
   a. Boot up the device.
   b. Disable the DM-verity feature. Execute the following command on the host side:

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

7 Android Platform Update

The following is an example for the i.MX 7ULP board to build and implement OTA update. For other platform, use "lunch " to set up the build configuration.

For detailed build configuration, see Section 3.2 Building Android images.

7.1 Building Android update package

Android build system supports auto generation of the `update.zip` function. It generates the `updater_script` and all `system.img` files.

Take i.MX 7ULP EVKB-SD as an example. Use the following command to generate an OTA package under the Android environment:

```
$ cd ${MY_ANDROID}
$ source build/envsetup.sh
$ lunch evk_7ulp-userdebug
$ make otapackage -j4
```

After the build is finished, you can find OTA packages in the following path:

```
out/target/proudct/evk_7ulp/evk_7ulp-ota-${date}-${soc}.zip
```

7.2 Updating the Android platform

7.2.1 Using ADB to update the Android platform

To use the ADB to update the Android platform, peform the following steps:

1. Enter recovery mode.
   There are two methods to enter recovery mode:
   - If the board supports the physical **Power/VOLUME DOWN/VOLUME UP** keys, press **VOLUME DOWN** and **Power** to enter recovery mode when the system is powered on.
   - Execute the following command on the board's console:

     ```
     $ reboot recovery
     ```

     - When the system completes boot-up into recovery mode, an Android Robot Logo is displayed.
     - Move the menu option by the **VOLUME UP** and **VOLUME DOWN** button.
     - Select the **Apply update from ADB** option by **Power Key**.
2. Download the OTA package.

   You can build the OTA package by following the steps in Building Android update package. Make sure your host has the ADB driver, and then connect the board with your host.

   Execute the following command:

   ```
   $ adb sideload $YOUR_UPDATE_PACKAGE.zip
   ```

   After the package is sent, the system starts updating the firmware with the update file.

7.2.2 Using the application to update the Android platform

Perform the following steps to use this application:

1. Set up an HTTP server (such as lighttpd, apache).
   You need one HTTP server to hold OTA packages. Put the following files to ${http_root}/evk_7ulp_${ota_folder_suffix}_${version}:

   ```
   ${MY_ANDROID}/out/target/product/evk_7ulp/system/build.prop
   ${MY_ANDROID}/out/target/product/evk_7ulp/evk_7ulp-ota-${date}-${soc}.zip
   ```

   - `evk_7ulp-ota-${date}-${soc}.zip` is built from Section 7.1 Building Android update package.
     - There may be two or more OTA packages as follows:
       - `evk_7ulp-ota-20180529-imx7ulp-mipi.zip`
       - `evk_7ulp-ota-20180529-imx7ulp.zip`
     - The OTA application only supports updating `evk_7ulp-ota-20180529-imx7ulp.zip`.
     - To update `evk_7ulp-ota-20180529-imx7ulp-mipi.zip`, change its name to `evk_7ulp-ota-20180529-imx7ulp.zip`.
   - `${ota_folder_suffix}` is stored at board's `/vendor/etc/ota.conf`.
   - `${version}` can be obtained by the following command on the board's console:

     ```
     $getprop | grep "ro.build.version.release"
     ```

2. Configure the IP address and port number of the OTA server.
   The content of the OTA configuration file (`/vendor/etc/ota.conf`) is as follows:

   ```
   server=192.168.1.100
   port=10888
   ota_folder_suffix=pie
   ```

   Modify it to fit your environment.

3. Open the OTA application.
   a. The reference application is a dialog activity, and can be enabled in the **Settings->About tablet->Additional system Update** menu. The application downloads this build.prop, parses it to get the build time, and compares it with its own build time.
   b. If the server build is newer, it shows the version information and package size and prompts the user for upgrade. Click the **Update** button to update the Android platform. Then the downloading starts.
   c. After the downloading is finished, the title changes to "Verify package". During this time, it is actually doing the `RecoverySystem.verifyPackage()` API to verify whether the package is complete, such as a MD5 checksum checking. It also checks whether the key chain in the package aligns with the key chain in the device.
   d. After the verification is finished, it calls the `RecoverySystem.installPackage()` API to install the package. It writes a recovery command and stores it into `/cache/recovery/command`, and then reboots the system. After rebooting, the system boots to recovery mode.
   e. After it installs the update package, the "Android Robot" is spinning on the screen. If an error occurs, "Error Robot" is displayed, and it stops spinning. Press the **MENU** button to view the log output on the screen.

8 Customized Configuration

8.1 How to change boot command line in boot.img

After boot.img is used, the default kernel boot command line is stored inside the image. It packages together during android build.

You can change this by changing BOARD_KERNEL_CMDLINE's definition in the `BoardConfig.mk` file under `${MY_ANDROID}/device/fsl/{product}/BoardConfig.mk`.

8.2 How to configure the logical display density

The Android UI framework defines a set of standard logical densities to help application developers target application resources.

Device implementations must report one of the following logical Android framework densities:

- 120 dpi, known as 'ldpi'
- 160 dpi, known as 'mdpi'
- 213 dpi, known as 'tvdpi'
- 240 dpi, known as 'hdpi'
- 320 dpi, known as 'xhdpi'
- 480 dpi, known as 'xxhdpi'

Device implementations should define the standard Android framework density that is numerically closest to the physical density of the screen, unless that logical density pushes the reported screen size below the minimum supported.

To configure the logical display density for framework, you must define the following line in `init.rc` under `${MY_ANDROID}/device/fsl/{product}`:

```
setprop ro.sf.lcd_density <density>
```

8.3 How to use SPDIF-in to record audio

By default, the SABRE-AI board supports audio record through the onboard audio input port.

To use the SPDIF -in device to record audio, execute the following commands on the target side:

```
cd /vendor/etc
cp audio_policy_configuration.xml audio_policy_configuration.xml_backup
cp audio_policy_configuration_spdif.xml audio_policy_configuration.xml
pkill audioserver
```

NOTE

Before this operation, see Section 6.2.3 DM-verity configuration to disable DM-verity.

9 Revision History

Table 21. Revision history

Revision number	Date	Substantive changes
O8.0.0_1.0.0	02/2018	Initial release
O8.0.0_1.0.0	10/2018	Updated the Graphic - HW 3D acceleration feature for the i.MX 7Dual to N/A.
P9.0.0_2.2.0-ga	07/2019	i.MX 6 and i.MX 7 GA release.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number AUG
Revision P9.0.0_2.2.0-ga, 07/2019