NXP Semiconductors
User's Guide

Document Number: AUG
Rev. O8.0.0_1.1.0_8QM-PRC1, 12/2017

Android™ User's Guide

Contents

1 Overview

This document describes how to build Android Oreo 8.0 platform for the i.MX 8 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

For any problems on the building process related to the jack server, see the Android website source.android.com/source/jack.html.

2.1 Setting up your computer

To build the Android source files, use a computer running the Linux OS. The Ubuntu 14.04 64bit version and openjdk-8-jdk is the most tested environment for the Android Oreo 8.0 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
```

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

```
# cd /opt (or any other directory where the imx-o8.0.0_1.1.0_8qm-prc1.tar.gz file is placed)

$ tar xzvf imx-o8.0.0_1.1.0_8qm-prc1.tar.gz
```

3 Building the Android platform for i.MX

3.1 Getting i.MX Android release source code

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

- NXP i.MX public source code, which is maintained in the 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

Assume you had i.MX Android proprietary source code package imx-o8.0.0_1.1.0_8qm-prc1.tar.gz under ~/. directory. To generate the i.MX Android release source code build environment, execute the following commands:

```
$ mkdir ~/bin
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo  > ~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=${PATH}:~/bin
```

```
$ source ~/imx-o8.0.0_1.1.0_8qm-prc1/imx_android_setup.sh
# By default, the imx_android_setup.sh script will create the source code build environemnt
in the folder ~/android_build
# ${MY_ANDROID} will be refered as the i.MX Android source code root directory in all i.MX
Andorid 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 arm2_8q-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
arm2_8qm	i.MX 8QuadMax Validation 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	Provides image with root access and debug, similar to "user"
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 8QuadMax Validation Board/Platform device with user type.

   ```
   $ lunch arm2_8q-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 *Android™ Frequently Asked Questions* (AFAQ).

The following outputs are generated by default in ${MY_ANDROID}/out/target/product/arm2_8q:

- 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.
- boot-imx8qm.img: a composite image for i.MX 8QuadMax Validation, which includes the kernel zImage, ramdisk, board's device tree binary, and boot parameters.
- ramdisk.img: Ramdisk image generated from "root/". Not directly used.
- system.img: EXT4 image generated from "system/". Can be programmed to "SYSTEM" partition on SD/eMMC card with "dd".
- userdata.img: EXT4 image generated from "data/".
- vbmeta-imx8qm.img: Android Verify boot metadata image for boot-imx8qm.img.
- partition-table.img: GPT partition table image. Used for 16 GB SD card.
- partition-table-7GB.img: GPT partition table image. Used for 8 GB SD card.
- partition-table-28GB.img: GPT partition table image. Used for 32 GB SD card.
- u-boot-imx8qm.imx: U-Boot image without padding for i.MX 8QuadMax Validation.
- vendor.img: vendor image, which holds platform binaries. Mounted at **/vendor**

NOTE

- To build the U-Boot image separately, see Building U-Boot images.
- To build the kernel uImage separately, see Building a kernel image.
- To build boot.img, see Building boot.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 8QuadMax Validation Board	$ lunch arm2_8q-userdebug

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

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

3.2.2 User build mode

A production release Android system image is created by using the **userdebug** Build Type. For configuration options, see Table "Build types" in Section Building Android images.

The notable differences between the **user** and **eng** build types are as follows:

- Limited Android System image access for security reasons.
- Lack of debugging tools.
- Installation modules tagged with user.
- APKs and tools according to product definition files, which are found in PRODUCT_PACKAGES in the sources folder ${MY_ANDROID}/device/fsl/imx8/imx8.mk. To add customized packages, add the package MODULE_NAME or PACKAGE_NAME to this list.
- The properties are set as: `ro.secure=1` and `ro.debuggable=0`.
- `adb` is disabled by default.

There are two methods for the build of Android image.

Method 1: Set the environment first and then issue the `make` command:

```
$ cd ${MY_ANDROID}
$ source build/envsetup.sh   #set env
$ make PRODUCT-XXX        #XXX depends on different board, see table below
```

Table 4. Android system image production build method 1

i.MX development tool	Description	Image build command
Evaluation Kit	i.MX 8QuadMax Validation	$ make PRODUCT-arm2_8q-userdebug>&1 | tee buildlog.txt

Method 2: Set the environment and then use `lunch` command to configure argument. See table below. An example for the i.MX 8QuadMax Validation board is as follows:

```
$ cd ${MY_ANDROID}
$ source build/envsetup.sh
$ lunch arm2_8q-userdebug
$ make
```

Table 5. Android system image production build method 2

i.MX development tool	Description	Lunch configuration
Evaluation Kit	i.MX 8QM Validation	arm2_8q-userdebug

To create Android platform over-the-air, OTA, and package, the following make target is specified:

```
$ make otapackage
```

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

3.3 Building U-Boot images

After you set up U-Boot using the steps outlined above, you can find the tool (mkimage) under tools/.

```
$ cd ${MY_ANDROID}/bootable/bootloader/uboot-imx
$ export ARCH=arm64
$ export CROSS_COMPILE=${MY_ANDROID}/prebuilts/gcc/linux-x86/aarch64/aarch64-linux-
android-4.9/bin/aarch64-linux-android-
$ make distclean
```

For i.MX 8QuadMax Validation board:

```
# To build uboot.imx that is used on the Android platform:
$ make mx8qm_lpddr4_arm2_android_defconfig
$ make
$ cp out/target/product/arm2_8q/obj/BOOTLOADER_OBJ/u-boot.bin ${MY_ANDROID}/external/imx-
mkimage/iMX8QM/u-boot.bin
$ cp ${MY_ANDROID}/device/fsl-proprietary/uboot-firmware/mx8qm-scfw-tcm.bin ${MY_ANDROID}/
external/imx-mkimage/iMX8QM/scfw_tcm.bin
$ cp ${MY_ANDROID}/device/fsl-proprietary/uboot-firmware/imx8qm_dcd.cfg.tmp ${MY_ANDROID}/
external/imx-mkimage/iMX8QM/.
$ cp ${MY_ANDROID}/device/fsl-proprietary/uboot-firmware/bl31-imx8qm.bin ${MY_ANDROID}/
external/imx-mkimage/iMX8QM/.
$ make -C ${MY_ANDROID}/external/imx-mkimage/ clean
$ make -C ${MY_ANDROID}/external/imx-mkimage/ SOC=iMX8QM flash
```

"external/imx-mkimage/iMX8QM/flash.bin" is generated if the build is successful. Flash.bin is taken as uboot-imx8qm.imx.

NOTE

imx-mkimage is a tool that bundles uboot.bin, spl, and firmware 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}/kernel_imx
$ echo $ARCH && echo $CROSS_COMPILE
```

Make sure that you have those two environment variables set. If the two variables are not set, set them as follows:

```
$ export ARCH=arm64
$ export CROSS_COMPILE=${MY_ANDROID}/prebuilts/gcc/linux-x86/aarch64/aarch64-linux-
android-4.9/bin/aarch64-linux-android-

# Generate ".config" according to default config file under arch/arm64/configs/
android_defconfig.
# To build the kernel zImage for i.MX 8QuadMax
$ make android_defconfig
$ make KCFLAGS=-mno-android

# to build the zImage which is used in MFGTOOL
# zImage is under mfgtools\Profiles\Linux\OS Firmware\firmware\
$ make android_defconfig
$ make KCFLAGS=-mno-android -j4
```

The kernel images are found in ${MY_ANDROID}/kernel_imx/arch/arm64/boot/Image.

3.5 Building boot.img

boot.img and boota are default booting commands.

As outlined in Running the Android Platform with a Prebuilt Image, we use boot.img and boota as default commands to boot instead of the uramdisk and zImage we used before.

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

```
# Boot image for i.MX 8QuadMax Validation board
$ source build/envsetup.sh
$ lunch arm2_8q-userdebug
$ make bootimage
```

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_O8.0.0_1.1.0_8QM-PRC1_image_8qmarm2.tar.gz	Prebuilt-image for i.MX 8QuadMax Validation board, which includes NXP extended features.

The following tables list the detailed contents of android_O8.0.0_1.1.0_8QM-PRC1_image_8qmarm2.tar.gz image package.

The table below shows the prebuilt images to support the system boot from SD on i.MX 8QuadMax Validation boards.

Table 7. Images for i.MX 8QuadMax Validation

i.MX 8QuadMax Validation image	Description
u-boot-imx8qm.imx	Bootloader (with padding) for i.MX 8QuadMax Validation board
partition-table.img	GPT table image for 16 GB SD card
partition-table-7GB.img	GPT table image for 8 GB SD card
partition-table-28GB.img	GPT table image for 32 GB SD card
boot-imx8qm.img	Boot image for i.MX 8QuadMax Validation board
system.img	System Boot image
vbmeta-imx8qm.img	Android Verify Boot metadata Image for i.MX 8QuadMax Validation board
vendor.img	Vendor image for i.MX 8QuadMax Validation board

NOTE

boot.img is an Android image that stores kernel Image 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 can avoid touching the boot loader code to change any default boot arguments.

5 Programming Images

The images from the prebuilt release package or created from source code contain the U-Boot boot loader, system image, GPT image, vendor image, and vbmeta image. At a minium, the storage devices on the development system (MMC/SD or NAND) must be programmed with the U-Boot boot loader. The i.MX 8 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:

- fsl-sdcard-partition.sh to flash images to the SD card.

5.1 System on eMMC/SD

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

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

- U-Boot image: u-boot.imx

- boot image: boot.img
- Android system root image: system.img
- Android verify boot metadata image: vbmeta.img
- GPT table image: partition-table.img
- Vendor image: vendor.img

5.1.1 Storage partitions

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

- [Partition type/index] which is defined in the GPT.
- [Start Offset] shows where partition is started, unit in MB.

The system partition is used to put the built-out Android system image. The userdata parition is used to put the unpacked codes/data of the applications, system configuration database, etc. In normal boot mode, the root file system is mounted from the system partition. In recovery mode, the root file system is mounted from the boot partition.

Table 8. Storage partitions

Partition type/index	Name	Start offset	Size	File system	Content
N/A	bootloader	33 KB	8 MB - 33 KB	N/A	bootloader
1	boot_a	8 MB	32 MB	boot.img format, a kernel + recovery ramdisk	boot.img
2	boot_b	Follow boot_a	32 MB	boot.img format, a kernel + recovery ramdisk	boot.img
3	system_a	Follow boot_b	1536 MB	EXT4. Mount as / system	Android system files under / system/dir
4	system_b	Follow system_a	1536MB	EXT4. Mount as / system	Android system files under / system/dir
5	misc	Follow system_b	4 MB	N/A	For recovery store bootloader message, reserve
6	datafooter	Follow misc	2 MB	N/A	For crypto footer of DATA partition encryption
7	metadata	Follow datafootor	2 MB	N/A	For system slide show
8	presistdata	Follow metadata	1 MB	N/A	Option to operate unlock \unlock
9	vendor_a	Follow persistdata	112 MB	EXT4. Mount at / vendor	vendor.img
10	vendor_b	Follow vendor_a	112 MB	EXT4. Mount at / vendor	vendor.img
11	userdata	Follow vendor_b	Remained space	EXT4. Mount at /data	Application data storage for system application, and for internal media partition, in /mnt/sdcard/ dir.
12	fbmisc	Follow userdata	1 MB	N/A	For storing the state of lock \unlock

Table continues on the next page...

Table 8. Storage partitions (continued)

Partition type/index	Name	Start offset	Size	File system	Content
13	vbmeta_a	Follow fbmisc	1 MB	N/A	For storing the verify boot's metadata
14	vbmeta_b	Follow vbmeta_a	1 MB	N/A	For storing the verify boot's metadata

To create these partitions, use fsl-sdcard-partition.sh described in the *Android™ Quick Start Guide* (AQSUG), or use format tools in the prebuilt 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 imx8mq.
```

NOTE

- The minimum size of the SD card is 8 GB.
- /dev/sdX, the X is the disk index from 'a' to 'z'. That may be different on each computer running Linux OS.
- If the SD card is 8 GB, use "sudo fsl-sdcard-partition.sh -f imx8qm -c 7 /dev/sdX" to flash images.
- If the SD card is 16 GB, use "sudo fsl-sdcard-partition.sh -f imx8qm /dev/sdX" to flash images.
- If the SD card is 32 GB, use "sudo fsl-sdcard-partition.sh -f imx8qm -c 28 /dev/sdX" to flash images.
- Unmount all the SD card partitions before running the script.
- Put the related bootloader, boot image, system image, recovery 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.
- If the SD card is 8 GB, copy partition-table-7GB.img and rename it to partition-table.img. If the SD card is 16 GB, use the default partition-table.img. If the SD card is 32 GB, copy partition-table-28GB.img and rename it to partition-table.img.

6 Booting

This chapter describes booting from MMC/SD, NAND, TFTP and NFS.

6.1 Booting from eMMC/SD/NAND

6.1.1 Booting from SD on the i.MX 8QuadMax Validation board

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

Table 9. Boot switch settings

boot switch	download Mode (MFGTool mode)	SD boot
SW500 Boot_Mode (from 0 bit to 5 bit)	001000	001100

Boot from SD

Chang the board Boot_Mode switch to 001100 (from 0 bit to 5 bit).

The default environment in boot.img is booting from SD. To use the default environment in boot.img, use the following command:

```
U-Boot > setenv bootargs
```

To clear the bootargs environment, use the following command:

```
U-Boot > setenv bootcmd boota mmc1
U-Boot > setenv bootargs console=ttyLP0,115200 earlycon=lpuart32,0x5a060000,115200,115200
init=/init video=imxdpufb1:off video=imxdpufb2:off video=imxdpufb3:off video=imxdpufb4:off
androidboot.console=ttyLP0 consoleblank=0 androidboot.hardware=freescale cma=800M [Optional]
U-Boot > saveenv                          [Save the environments]
```

NOTE

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

6.2 Boot-up configurations

This section explains the common U-Boot environments used for NFS, MMC/SD boot, and kernel command line.

6.2.1 U-Boot environment

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

  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 from mmcX, do the following:

  ```
  > booti mmcX
  ```

  To read the boot partition (the partition store boot.img, in this instance, mmcblk0p1), the X is the eMMC bus number, which is the hardware eMMC bus number, in SABRE-SD boards. eMMC is mmc2 or you can add the partition ID after mmcX.

  Add partition ID after mmcX.

```
> boota mmcX boot   #  boot is default
> boota mmcX recovery      # boot from the recovery partition
```

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 10. 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".
androidboot.hardware	Specifies hardware name for this product. Android init process loads the configuration file init.$ (androidboot.hardware).rc in the root directory.	androidboot.hardware=freescale	All use cases.
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=imxdpufb1:off video=imxdpufb2:off video=imxdpufb3:off video=imxdpufb4:off	-
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.
fec_mac	Sets up the FEC MAC address.	fec_mac=00:04:9f:00:ea:d3	On SABRE-SD board, the SoC does not have a MAC address fused in. To use FEC, assign this parameter to the kernel.
cma	CMA memory size for GPU/VPU physical memory allocation.	cma=800M	It is 800 MB by default.
androidboot.selinux	Argument to disable selinux check and enable	androidboot.selinux=permissive	Android Nougat 7.1 CTS requirement: The serial input should be disabled by default. Setting this argument enables console

Table 10. Kernel boot parameters

Kernel parameter	Description	Typical value	Used when
	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.		serial input, which violates the CTS requirement.

7 Revision History

Table 11. Revision history

Revision number	Date	Substantive changes
O8.0.0_1.1.0_8QM-EAR	10/2017	Initial release
O8.0.0_1.1.0_8QM-PRC1	12/2017	i.MX 8QuadMax PRC1 release

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AUG
Rev. O8.0.0_1.1.0_8QM-PRC1
12/2017