Document Number:AUG
Rev jb4.2.2_1.1.0-GA, 07/2013

Android User's Guide

Contents

1 Overview

This document explains how to use the Android release package.

It describes how to set up the environment, how to apply i.MX Android patches, and how to build the Android system. It also describes how to download prebuilt images to a target storage device and set up the correct hardware/software boot configurations to boot the system. Meanwhile, it provides the information on how to get Android source from Google, and what the device storage usage and boot option are. For more information, see http://source.android.com/source/building.html.

2 Preparation

2.1 Set Up Your Computer

To build the Android source files, you will need to use Linux computer.

You also need to use the 10.10 or 11.04 64 bit version of Ubuntu which are the most tested OS for the Android JB4.2.2 build.

Preparation

After installing the Linux computer, you need to check whether you have all the necessary packages installed for an Android build. See "Setting up your machine" on the Android website http://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
```

2.2 Unpack Android Release Package

After you have set up a Linux computer, unpack the Android release package.

After you have setup a Linux computer, unpack the Android release package by using the following commands:

```
$ cd /opt (or any other directory where you placed the android_jb4.2.2_1.1.0-ga_source.tar.gz file)

$ tar xzvf android_jb4.2.2_1.1.0-ga_source.tar.gz

$ cd android_jb4.2.2_1.1.0-ga_source/code

$ tar xzvf jb4.2.2_1.1.0-ga.tar.gz
```

2.3 Run Android with Prebuilt Image

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

Table 1. Image Packages

Image Package	Description
android_jb4.2.2_1.1.0-ga_image_6qsabresd.tar.gz	The table below shows the prebuilt image for i.MX 6Dual/6Quad SABRE-SD board and platform and i.MX 6Solo/6DualLite SABRE-SD platform.
android_jb4.2.2_1.1.0-ga_image_6qsabreauto.tar.gz	The table below shows the prebuilt image for i.MX 6Dual/6Quad and i.MX 6Solo/6DualLite SABRE-AI platform.

The following tables list the detailed contents of android_jb4.2.2_1.1.0-ga_image_6qsabresd.tar.gz image package.

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

Table 2. Images for SABRE-SD Board and Platform eMMC Boot

SABRE-SD eMMC Image	Description
u-boot-6q.bin	The bootloader (with padding) for i.MX 6Dual/6Quad SABRE-SD board and platform
u-boot-6dl.bin	The bootloader (with padding) for i.MX 6Solo/6DualLite SABRE-SD platform
eMMC/boot.img	Boot Image for eMMC
eMMC/system.img	System Boot Image

Table continues on the next page...

Table 2. Images for SABRE-SD Board and Platform eMMC Boot (continued)

eMMC/recovery.img	Recovery Image

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

Table 3. Images for SABRE-SD SD

SABRE-SD SD Image	Description
u-boot-6q.bin	The bootloader (with padding) for i.MX 6Dual/6Quad SABRE-SD board and platform
u-boot-6dl.bin	The bootloader (with padding) for i.MX 6Solo/6DualLite SABRE-SD platform
SD/boot.img	Boot Image for SD
SD/system.img	System Boot Image
SD/recovery.img	Recovery Image

The table below shows the prebuilt images to support the system boot from TFTP server and NFS rootfs on i.MX 6Dual/6Quad and i.MX 6Solo/6DualLite SABRE-SD boards.

Table 4. Images for SABRE-SD TFTP and NFS

SABRE-SD TFTP/NFS Image	Description
u-boot-6q.bin	The bootloader (with padding) for i.MX 6Dual/6Quad SABRE-SD board and platform
u-boot-6dl.bin	The bootloader (with padding) for i.MX 6Solo/6DualLite SABRE-SD platform
NFS/android_fs.tar.gz	NFS rootfs
NFS/uImage	Kernel image for TFTP

The following tables list the detailed contents of android_jb4.2.2_1.1.0-ga_image_6qsabreauto.tar.gz image package:

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

Table 5. Images for SABRE-AI SD Boot

SABRE-AI SD Image	Description
u-boot-mx6q.bin	The bootloader (with padding) for i.MX 6Dual/6Quad SABRE-AI SD boot
u-boot-6dl.bin	The bootloader (with padding) for i.MX 6DualLite SABRE-AI SD boot
u-boot-mx6solo.bin	The bootloader (with padding) for i.MX 6Solo SABRE-AI SD boot
SD/boot.img	Boot Image for SD
SD/system.img	System Boot Image
SD/recovery.img	Recovery Image

The table below shows the prebuilt images to support the system boot from NAND on i.MX 6 series SABRE-AI boards.

Table 6. Images for SABRE-AI NAND Boot

SABRE-AI NAND Image	Description
u-boot-mx6q-nand.bin	The bootloader (with padding) for i.MX 6Dual/6Quad SABRE-AI NAND boot
u-boot-mx6dl-nand.bin	The bootloader (with padding) for i.MX 6DualLite SABRE-AI NAND boot
u-boot-mx6solo-nand.bin	The bootloader (with padding) for i.MX 6Solo SABRE-AI NAND boot
NAND/boot.img	Boot Image for NAND
NAND/android_root.img	System Boot Image
NAND/recovery.img	Recovery Image

The table below shows the prebuilt images to support the system boot from TFTP server and NFS rootfs on i.MX 6 series SABRE-AI boards.

Table 7. Images for SABRE-AI TFTP and NFS

SABRE-AI TFTP/NFS Image	Description
u-boot-6q.bin	The bootloader (with padding) for i.MX 6Dual/6Quad SABRE-AI SD boot
u-boot-6dl.bin	The bootloader (with padding) for i.MX 6DualLite SABRE-AI boot
u-boot-mx6solo.bin	The bootloader (with padding) for i.MX 6Solo SABRE-AI SD boot
NFS/android_fs.tar.gz	NFS rootfs
NFS/uImage	Kernel image for TFTP

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, e.g. These information can be configured in corresponding board's BoardConfig.mk. If you use boot.img, you do not need uImage and uramdisk.img. The SD card images only boot up during the sanity test, since the only differences between eMMC and SD images are the boot.img.

3 Build Android for i.MX

3.1 Get Android Source Code (Android/Kernel/U-Boot)

The Android source code is maintained as more than 100 gits in the Android repository (android.googlesource.com).

To get the Android source code from Google repo, follow the steps below:

```
Assume you had unzipped i.MX Android release package to /opt/android_jb4.2.2_1.1.0-
ga_source/.
```

```
$ cd ~
$ mkdir myandroid
$ cd myandroid
$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ./repo
$ chmod a+x ./repo
$ ./repo init -u https://android.googlesource.com/platform/manifest -b android-4.2.2_r1
$ ./repo sync # this command loads most needed repos. Therefore, it can take a while to load.
```

Get jb4.2.2_1.1.0-ga kernel source code from Freescale open source git:

```
$ cd myandroid
$ git clone git://git.freescale.com/imx/linux-2.6-imx.git kernel_imx # the kernel repo is
heavy. Therefore, this process can take a while.
$ cd kernel_imx
$ git checkout jb4.2.2_1.1.0-ga
```

NOTE

If you are behind proxy, use socksify to set socks proxy for git protocol.

If you use U-Boot as your bootloader, then you can clone the U-Boot git repository from Freescale open source git:

```
$ cd myandroid/bootable
$ cd bootloader
$ git clone git://git.freescale.com/imx/uboot-imx.git uboot-imx
$ cd uboot-imx
$ git checkout jb4.2.2_1.1.0-ga
```

3.2 Patch Code for i.MX

The patch script (and_patch.sh) requires some basic utilities like awk/sed.

Apply all i.MX Android patches by using the following steps:

```
Assume you had unzipped i.MX Android release package to /opt/android_jb4.2.2_1.1.0-ga_source.
$ cd ~/myandroid
$ source /opt/android_jb4.2.2_1.1.0-ga_source/code/jb4.2.2_1.1.0-ga/and_patch.sh
$ help
Now you should see that the "c_patch" function is available
$ c_patch /opt/android_jb4.2.2_1.1.0-ga_source/code/jb4.2.2_1.1.0-ga imx_jb4.2.2_1.1.0-ga
Here "/opt/android_jb4.2.2_1.1.0-ga_source/code/jb4.2.2_1.1.0-ga" is the location of the
patches (i.e. directory created when you unzip release package)
"imx_jb4.2.2_1.1.0-ga" is the branch which will be created automatically for you to hold all
patches (only in those existing Google gits).
You can choose any branch name you like instead of "imx_jb4.2.2_1.1.0-ga".
If everything is OK, "c_patch" will generate the following output to indicate successful
patch:
```

```
************************************************************
Success: Now you can build the Android code for FSL i.MX platform
************************************************************
```

NOTE

The patch script (and_patch.sh) requires some basic utilities like awk/sed. If they are not available on your Linux computer, install them first.

3.3 Build Android Image

After applying all i.MX patches, build the U-Boot, kernel, and Android image.

After applying all i.MX patches, build the U-Boot, kernel, and Android image using the steps below:

```
# Build Android images for i.MX6 SABRE-SD boards
$ cd ~/myandroid
$ source build/envsetup.sh
$ lunch sabresd_6dq-user
$ make
"sabresd_6dq" is the product name (see ~/myandroid/device/fsl/product)
After build, check build_*_android.log to make sure no build error.

#Build Android images for i.MX6 SABRE-AI boards
$ lunch sabreauto_6q-user
$ make
```

For BUILD_ID & BUILD_NUMBER, add a buildspec.mk in your ~/myandroid directory. For details, see the Android Frequently Asked Questions document.

For i.MX 6Dual/6Quad SABRE-SD and i.MX 6DualLite SABRE-SD boards, we use the same build configuration. The two boards share the same kernel/system/recovery images with the exception of the U-Boot image. The following outputs are generated by default in myandroid/out/target/product/sabresd_6dq:

- root/ is a root file system (including init, init.rc, etc). Mounted at /
- system/ is an Android system binary/libraries. Mounted at /system.
- data/ is an Android data area. Mounted at /data.
- recovery/ is a root file system when booting in "recovery" mode. Not used directly.
- boot.img is a composite image which includes the kernel zImage, ramdisk, and boot parameters.
- ramdisk.img is a ramdisk image generated from "root/". Not used directly.
- system.img is an EXT4 image generated from "system/". It can be programmed to "SYSTEM" partition on SD/eMMC card with "dd".
- userdata.img is an EXT4 image generated from "data/".
- recovery.img is an EXT4 image generated from "recovery/". It can be programmed to "RECOVERY" partition on SD/eMMC card with "dd".
- u-boot-6q.bin is an U-Boot image with padding for i.MX 6Dual/6Quad SABRE-SD.
- u-boot-6dl.bin is an U-Boot image with padding for i.MX 6DualLite SABRE-SD.

NOTE

Make sure the mkimage is a valid command in your build machine. If not, use the command below to have it installed:

```
$sudo apt-get install uboot-mkimage
```

- To build the U-Boot image separately, see Build U-Boot Images.
- To build the kernel uImage separately, see Build Kernel Image.
- To build boot.img, see Build boot.img.

3.3.1 User Build Mode

For a production release, the Android image should be built in the user mode.

When compared to eng mode, it will have the following differences:

- It will have limited access due to security reasons, and it will lack certain debug tools.
- It will install modules tagged with user, and APKs& tools according to product definition files, which are in PRODUCT_PACKAGES in device/fsl/imx6/imx6.mk.
- Set ro.secure=1, and ro.debuggable=0. adb is disabled by default.

If you need to add your customized package, add the package MODULE_NAME or PACKAGE_NAME to this list.

```
# Build images for i.MX6 SABRE-SD board
$ make PRODUCT-sabresd_6dq-user 2>&1 | tee build_sabresd_6dq_android.log

# Build Images for i.MX6 SABRE-AI board
$ make PRODUCT-sabreauto_6q-user 2>&1 | tee build_sabreauto_6dq_android.log
```

Or you can use the following commands:

```
# Build images for i.MX6 SABRE-SD board
$ source build/envsetup.sh
$ lunch sabresd_6dq-user
$ make
$ make dist # you can generate ota package with this command.

# Build images for i.MX6 SABRE-AI board
$ source build/envsetup.sh
$ lunch sabreauto_6q-user
$ make
$ make dist # you can generate ota package with this command.
```

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

3.3.2 Build Android Image for SD Card on SABRE-SD Board

The default configuration in the source code package takes internal eMMC as the boot storage for i.MX 6Dual/6Quad and i.MX 6Solo/6DualLite SABRE-SD boards.

The default setting can be changed to make the SD card in SD Slot 3 be the boot storage as shown below:

1. Change the fstab.freescale and recovery.fstab configuration files in device/fsl.git by using the following patch to make sure that the partitions are mounted for SD:

```
diff --git a/imx6/etc/fstab.freescale b/imx6/etc/fstab.freescale
index 685b4ea..084adb3 100644
--- a/imx6/etc/fstab.freescale
+++ b/imx6/etc/fstab.freescale
@@ -4,8 +4,8 @@
 # specify MF_CHECK, and must come before any filesystems that do specify MF_CHECK

-/dev/block/mmcblk0p5    /system  ext4
ro                                                                          wait
-/dev/block/mmcblk0p4    /data    ext4
nosuid,nodev,nodiratime,noatime,nomblk_io_submit,noauto_da_alloc,errors=panic    wait
-/dev/block/mmcblk0p6    /cache   ext4
nosuid,nodev,nomblk_io_submit                                              wait
-/dev/block/mmcblk0p7    /device  ext4
ro,nosuid,nodev                                                            wait
+/dev/block/mmcblk1p5    /system  ext4
```

```
ro                                                                                  wait
+/dev/block/mmcblk1p4    /data    ext4
nosuid,nodev,nodiratime,noatime,nomblk_io_submit,noauto_da_alloc,errors=panic    wait
+/dev/block/mmcblk1p6    /cache   ext4
nosuid,nodev,nomblk_io_submit                                                   wait
+/dev/block/mmcblk1p7    /device  ext4    ro,nosuid,nodev wait
diff --git a/sabresd_6dq/recovery.fstab b/sabresd_6dq/recovery.fstab
index 84e242b..d72c782 100644
--- a/sabresd_6dq/recovery.fstab
+++ b/sabresd_6dq/recovery.fstab
@@ -1,9 +1,9 @@
-/boot   emmc /dev/block/mmcblk0p1
-/recovery emmc /dev/block/mmcblk0p2
-/system ext4 /dev/block/mmcblk0p5
-/cache ext4 /dev/block/mmcblk0p6
-/data ext4 /dev/block/mmcblk0p4 reserved=32768
-/misc  emmc  /dev/block/mmcblk0p8
+/boot   emmc /dev/block/mmcblk1p1
+/recovery emmc /dev/block/mmcblk1p2
+/system ext4 /dev/block/mmcblk1p5
+/cache ext4 /dev/block/mmcblk1p6
+/data ext4 /dev/block/mmcblk1p4 reserved=32768
+/misc  emmc  /dev/block/mmcblk1p8
 /sdcard vfat /dev/block/mmcblk1p1

 # Use below lines if you use NAND:
```

2. Change the vold configuration file in device/fsl.git by using the following patch to make sure the primary media storage is correctly loaded onto the SD:

```
diff --git a/sabresd_6dq/vold.fstab b/sabresd_6dq/vold.fstab
index db230c5..4f1615b 100644
--- a/sabresd_6dq/vold.fstab
+++ b/sabresd_6dq/vold.fstab
@@ -26,7 +26,7 @@
 #used for all usb host
 dev_mount udisk /mnt/udisk auto /devices/platform/fsl-ehci
 #mount SDHC4 SD card /mnt/sdcard as primary storage for MX6Q SABER_LITE RevC
-dev_mount extsd /mnt/extsd auto /devices/platform/sdhci-esdhc-imx.2/mmc_host/mmc1
+dev_mount extsd /mnt/extsd auto /devices/platform/sdhci-esdhc-imx.1/mmc_host/mmc2
 #mount SDHC3 TF card to /mnt/extsd as external storage forMX6Q SABER_LITE RevC
 #dev_mount sdcard /mnt/sdcard 4 /devices/platform/sdhci-esdhc-imx.3/mmc_host/mmc0
```

3. Follow User Build Mode to build the images.

3.3.3 Build Android Image for NAND on SABRE-AI Board

The default configuration in the source code package takes SD card as the boot storage for i.MX 6 series SABRE-AI boards.

The default setting can be changed to make the NAND Flash in U19 be the boot storage as shown below:

```
make PRODUCT-sabreauto_6q-user BUILD_TARGET_FS=ubifs
```

NOTE

Make sure the uuid-dev is installed in your build machine (/usr/lib/x86_64-linux-gnu/uuid.so). If not, please follow the command below to have it installed:

```
$sudo apt-get install uuid-dev
$sudo apt-get install liblzo2-dev
```

3.4 Build U-Boot Images

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

```
$ cd ~/myandroid/bootable/bootloader/uboot-imx
$ export ARCH=arm
$ export CROSS_COMPILE=~/myandroid/prebuilts/gcc/linux-x86/arm/arm-eabi-4.6/bin/arm-eabi-
```

Command to build for i.MX 6Dual/6Quad SABRE-SD or i.MX 6DualLite SABRE-SD board is:

```
$ make distclean
```

For i.MX 6Quad SABRE-SD:

```
$ make mx6q_sabresd_android_config
```

For i.MX 6DualLite SABRE-SD:

```
$ make mx6dl_sabresd_android_config
$ make
```

Command to build for i.MX 6 series SABRE-AI boards is:

```
$ make distclean
```

For i.MX 6Quad SABRE-AI SD:

```
$ make mx6q_sabreauto_android_config
$ make
```

For i.MX 6Quad SABRE-AI NAND:

```
$ make mx6q_sabreauto_nand_android_config
$ make
```

For i.MX 6DualLite SABRE-AI SD:

```
$ make mx6dl_sabreauto_android_config
$ make
```

For i.MX 6DualLite SABRE-AI NAND:

```
$ make mx6dl_sabreauto_nand_android_config
$ make
```

For i.MX 6Solo SABRE-AI SD:

```
$ make mx6solo_sabreauto_android_config
$ make
```

For i.MX 6Solo SABRE-AI NAND:

```
$ make mx6solo_sabreauto_nand_android_config
$ make
```

"u-boot.bin" is generated if you have a successful build. The above u-boot.bin has 1KB padding at the head of the file, for example: first executable instruction is at the offset 1KB. If you want to generate a no-padding image, you need to implement the dd command specified below in host.

```
$ sudo dd if=./u-boot.bin of=./u-boot-no-padding.bin bs=1024 skip=1; sync
```

Usually the no-padding U-Boot image is used in the SD card, for example, program the no-padding U-Boot image into 1KB offset of SD card so that you do not overwrite the MBR (including partition table) within first 512B on the SD card.

NOTE

Any image that should be loaded by U-Boot must have an unique image head. For example, data must be added at the head of the loaded image to tell U-Boot about the image (i.e., it's a kernel, or ramfs, etc) and how to load the image (i.e., load/execute address). Before you can load any image into RAM by U-Boot, you need a tool to add this information and generate a new image which can be recognized by U-Boot. The tool is delivered together with U-Boot. After you set up U-Boot using the steps outlined above, you can find the tool (mkimage) under tools/. The process of using mkimage to generate an image (for example, kernel image and ramfs image), which is to be loaded by U-Boot, is outlined in the subsequent sections of this document.

3.5 Build Kernel Image

Kernel image will be built out while building the Android root file system.

If you do not need to build the kernel image, you can skip this section.

To run Android using NFS, or from SD, build the kernel with the default configuration as described below:

Assume you had already built U-Boot. mkimage was generated under myandroid/bootable/bootloader/uboot-imx/tools/ and it is in your PATH.

```
$ export PATH=~/myandroid/bootable/bootloader/uboot-imx/tools:$PATH
$ cd ~/myandroid/kernel_imx
$ echo $ARCH && echo $CROSS_COMPILE
```

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

```
$ export ARCH=arm
$ export CROSS_COMPILE=~/myandroid/prebuilts/gcc/linux-x86/arm/arm-eabi-4.6/bin/arm-eabi-
$ make imx6_android_defconfig
```

Generate ".config" according to default config file under arch/arm/configs.

```
$ make uImage
```

With a successful build in either of the above case, the generated kernel image is ~/myandroid/kernel_imx/arch/arm/boot/uImage.

3.6 Build boot.img

boot.img and booti are default booting commands.

As outlined in Run Android with Prebuilt Image, we use boot.img and booti as default commands to boot, not the uramdisk and uImage we used before.

You can use this command to generate boot.img under Android environment:

```
# Boot image for SABRE-SD board
$ cd ~/myandroid
$ source build/envsetup.sh
$ lunch sabresd_6dq-user
$ make bootimage

# Boot image for SABRE-AI board
$ source build/envsetup.sh
$ lunch sabreauto_6q-user
$ make bootimage
```

4 Download Images

i.MX Android can be booted up from MMC/SD and NFS (networking).

i.MX Android can be booted up in the following ways:

1. Boot from MMC/SD
2. Boot from NFS (networking)
3. Boot from NAND for SABRE-AI board

Before boot, you should program the bootloader, kernel, ramdisk, and rootfs images into the main storage device (MMC/SD, TF or NAND), or unpack the NFS root filesystem into the NFS server root.

The following download methods are supported for i.MX 6Dual/6Quad SABRE-SD and i.MX 6Solo/6DualLite SABRE-SD boards:

- MFGTool to download all images to MMC/SD card.
- Using dd command to download all images to MMC/SD card.

4.1 System on MMC/SD

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

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

- U-Boot image: u-boot.bin or u-boot-no-padding.bin
- boot image: boot.img
- Android system root image: system.img
- Recovery root image: recovery.img

The images can either be obtained from the release package or they can be built out.

4.1.1 Storage Partitions

To create storage partitions, you can use MFGTool as described in the Android Quick Start Guide, or you can use format tools in prebuild dir.

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

- [Partition type/index] is which defined in the MBR.
- [Name] is only meaningful in Android. You can 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 out 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.

Partition Type/Index	Name	Start Offset	Size	File System	Content
N/A	BOOT Loader	1 KB	1 MB	N/A	bootloader

Table continues on the next page...

Primary 1	Boot	8 MB	8 MB	boot.img format, kernel + ramdisk	boot.img
Primary 2	Recovery	Follow Boot	8MB	boot.img format, kernel + ramdisk	recovery.img
Logic 5 (Extended 3)	SYSTEM	Follow Recovery	512 MB	EXT4. Mount as / system	Android system files under / system/ dir.
Logic 6 (Extended 3)	CACHE	Follow SYSTEM.	512 MB	EXT4. Mount as / cache.	Android cache for image store of OTA.
Logic 7 (Extended 3)	Device	follow CACHE	8 MB	Ext4. Mount at /vender.	To Store MAC address files.
Logic 8 (Extended 3)	Misc	Follow Device	4M	N/A	For recovery store bootloader message, reserve.
Primary 4	DATA	Follow Misc.	Total - Other images	EXT4. Mount at / data.	Application data storage for the system application and for internal media partition in /mnt/sdcard/ dir.

There is a shell script mksdcard.sh.tar in MFGTool-Dir\Profiles\MX6DL Linux Update\OS Firmware.

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

```
$ cd ~/myandroid/
$ sudo chmod +x ./device/fsl/common/tools//fsl-sdcard-partition.sh
$ sudo ./device/fsl/common/tools/fsl-sdcard-partition.sh /dev/sdX
```

NOTE

- The minimum size of SD card is 2GB.
- /dev/sdxN, the x is the disk index from 'a' to 'z'. That may be different on each Linux computer.
- The default images and source code in this release only support boot from eMMC device for i.MX 6Dual/6Quad SABRE-SD and i.MX 6Solo/6DualLite SABRE-SD. If you want to boot it from the external SD card, see the Android Frequently Asked Questions document included in the release package.
- Unmount all the SD card partitions before running the script.

4.1.2 Download Images with MFGTool

MFGTool 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 for detailed description of MFGTool.

4.1.3 Download Images with dd Utility

The Linux utility "dd" on Linux computer can be used to download the images into the MMC/SD/TF card.

Before downloading, ensure that your MMC/SD/TF card partitions are created as described in Storage Partitions.

All partitions can be recognized by the Linux computer. To download all images into the card, use the commands below:

```
Download the U-Boot image:
# sudo dd if=u-boot.bin of=/dev/sdx bs=1K skip=1 seek=1; sync
Or If you're using no padding uboot image:
```

```
# sudo dd if=u-boot-no-padding.bin of=/dev/sdx bs=1K seek=1; sync
Download the boot image:
# sudo dd if=boot.img of=/dev/sdx1; sync
Download the android system root image:
# sudo dd if=system.img of=/dev/sdx5; sync
Download the android recovery image:
# sudo dd if=recovery.img of=/dev/sdx2; sync
```

4.2 System on NFS

Android support runs the system on NFS root file system.

We can put the entire Android root system in NFS, and we can load kernel image from TFTP server.

You must have a computer that has NFS and TFTP server, with their root directory set up correctly, for example, /opt/tftproot for TFTP root, and /opt/nfsroot for NFS root.

4.2.1 Set Up TFTP and NFS Root

After you set up the TFTP/NFS server, put the kernel image into the TFTP server root directory and the Android file system files into the NFS server root directory.

For kernel image, use uImage instead of zImage.

- If you are using a prebuilt image, ensure that you pick up the correct uImage (see "Prebuilt image for using U-Boot").
- If you are building your own image, ensure that you have generated a uImage (see "Generate uImage to be loaded by U-Boot").

Copy uImage to the TFTP server root directory. For example:

```
$ cp your_uImage /opt/tftproot/
```

Set up the Android file system (take i.MX 6Dual/6Quad SABRE-SD or i.MX 6Solo/6DualLite SABRE-SD as an example):

- If you are using a prebuilt image, unzip the Android zip file (see "Prebuilt image for using U-Boot") to the NFS server root. For example:

  ```
  $ cd /opt/android_jb4.2.2_1.1.0-ga_image_6qsabresd/NFS

  $ tar xzvf ./android_fs.tar.gz

  $ cd android_fs

  $ rm -rf /opt/nfsroot/*

  $ cp -r * /opt/nfsroot/*
  ```

- If you built out your own Android image, copy the generated Android files to the NFS root manually. For example:

  ```
  $ cd ~/myandroid
  $ rm -rf /opt/nfsroot/*
  $ cp -r out/target/product/sabresd_6dq/root/* /opt/nfsroot/
  $ cp -r out/target/product/sabresd_6dq/system/* /opt/nfsroot/system/
  ```

NOTE

Since the NFS uses system, data, and cache folders under /opt/nfsroot/, we have to change some settings and command sequence in /opt/nfsroot/init.rc and /opt/nfsroot/init.freescale.rc. Since the framework will clear ethernet's IP when suspended, which causes resume failure, the system property ethernet.clear.ip should be set to "no" in /opt/nfsroot/init.rc. For example:

```
--- a/opt/nfsroot/init.rc
+++ b/opt/nfsroot/init.rc
@@ -144,7 +144,6 @@ loglevel 3

on post-fs
# once everything is setup, no need to modify /
-mount rootfs rootfs / ro remount
# We chown/chmod /cache again so because mount is run as root + defaults
chown system cache /cache
chmod 0770 /cache
@@ -370,6 +369,7 @@ on boot
class_start core
class_start main
+class_start late_start
on property:sys.boot_completed=1
# Set default CPU frequency governor
@@ -400,8 +400,6 @@ on property:sys.interactive="active"
chmod 0660 /sys/devices/system/cpu/cpufreq/interactive/input_boost
-on nonencrypted
-class_start late_start
on charger
class_start charger

--- a/opt/nfsroot/init.freescale.rc
+++ b/opt/nfsroot/init.freescale.rc
@@ -93,6 +93,7 @@ on boot
# No bluetooth hardware present
setprop hw.bluetooth 0
setprop wlan.interface wlan0
+setprop ro.nfs.mode yes
# mount the debugfs
mount debugfs none /sys/kernel/debug/
@@ -126,6 +127,6 @@ service iprenew_wlan0 /system/bin/dhcpcd -n
disabled
oneshot
-on fs
+#on fs
# mount ext4 partitions
-mount_all /fstab.freescale
+#mount_all /fstab.freescale
```

4.3 System on NAND for SABRE-AI Board

The images needed to create an Android system on NAND are listed below:

- U-Boot image: u-boot-mx6q-nand.bin, u-boot-mx6dl-nand.bin, u-boot-mx6solo-nand.bin
- Boot image: boot.img
- Android system root image: android_root.img
- Recovery root image: recovery.img

The images can either be obtained from the release package or they can be built out.

4.3.1 Storage Partitions on NAND

The layout of the NAND for Android system is shown below:

- [Partition type/index] is which defined in the MBR.
- [Name] is only meaningful in Android. You can 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 out Android system image. The DATA is used to put the application's 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.

Partition Type/Index	Name	Start Offset	Size	File System	Content
MTD0	BOOT Loader	0	16MB	N/A	bootloader
MTD1	Boot	16M	16MB	boot.img format, kernel + ramdisk	boot.img
MTD2	Recovery	Follow Boot	128MB	boot.img format, kernel + ramdisk	recovery.img
MTD3(ubi0:system)	SYSTEM	Follow Recovery	300MB	ubifs. Mount as / system	Android system files under / system/ dir.
MTD3(ubi0:cache)	CACHE	Follow SYSTEM.	200MB	ubifs. Mount as / cache	Android cache for image store of OTA.
MTD3(ubi0:device)	Vendor	Follow CACHE.	10 MB	ubifs. Mount at / vender	For Store MAC address files.
MTD3(ubi0:data)	Data	follow Vendor	300MB	ubifs Mount at /vender.	Application data storage for system application. And for internal media partition, in / mnt/sdcard/ dir.

To create storage partitions, you can use MFGTool as described in the Android Quick Start Guide.

4.3.2 Download Images with MFGTool for NAND

MFGTool 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 for detailed description of MFGTool for NAND.

5 Boot

5.1 Boot from MMC/SD

5.1.1 Boot from MMC/SD on 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:

download Mode(MFGTool mode)	(SW6) 00001100 (from 1-8 bit)
eMMC 4-bit (MMC3) boot	(SW6) 11100110 (from 1-8 bit)
eMMC 8-bit (MMC3) boot	(SW6) 11010110 (from 1-8 bit)

Table continues on the next page...

MMC4 (SD2) boot	(SW6) 10000010 (from 1-8 bit)
MMC2 (SD3) boot	(SW6) 01000010 (from 1-8 bit)

Boot from eMMC

Change the board boot switch to eMMC 4-bit mode and make (SW6) 11100110 (from 1-8 bit). Or change (SW6) 11100110 (from 1-8 bit) for 8-bit boot mode.

The default environment in boot.img is booting from eMMC. If you want to use the default environment in boot.img, you can use the following command:

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

To clear the bootargs env, you can use the following commands:

```
U-Boot > setenv fastboot_dev mmc3 [eMMC as fastboot deivce]
U-Boot > setenv bootcmd booti mmc3      [Load the boot.img from eMMC]
U-Boot > setenv bootargs console=ttymxc0,115200 init=/init video=mxcfb0:dev=ldb,bpp=32
video=mxcfb1:off video=mxcfb2:off fbmem=10M fb0base=0x27b00000 vmalloc=400M
androidboot.console=ttymxc0 androidboot.hardware=freescale  #[Optional]
U-Boot > saveenv        #[Save the environments]
```

NOTE

The mmcX value changes depending on the boot mode. These are the correct values:

- eMMC --> mmc3
- MMC4 (SD2) --> mmc1
- MMC2 (SD3) --> mmc2

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

Some SoCs on SABRE-SD boards do not have MAC address fused.

Therefore, if you want to use FEC in U-Boot, set the following environment:

```
U-Boot > setenv ethaddr 00:04:9f:00:ea:d3            #[setup the
MAC address]
U-Boot > setenv fec_addr 00:04:9f:00:ea:d3           $[setup the
MAC address]
```

Boot from SD card

Change the board boot switch to MMC2 (SD3) boot: (SW6) 01000010 (from 1-8 bit). To clear the bootargs env and set up the booting from SD card in SD slot 3, you can use the following command:

```
U-Boot > setenv fastboot_dev  [eMMC as fastboot deivce]
U-Boot > setenv bootcmd booti mmc2         [Load the boot.img from SD card]
U-Boot > setenv bootargs console=ttymxc0,115200 init=/init video=mxcfb0:dev=ldb,bpp=32
video=mxcfb1:off video=mxcfb2:off fbmem=10M fb0base=0x27b00000 vmalloc=400M
androidboot.console=ttymxc0 androidboot.hardware=freescale  #[Optional]
U-Boot > saveenv    #[Save the environments]
```

NOTE

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

Some SoCs on SABRE-SD boards do not have MAC address fused.

Therefore, if you want to use FEC in U-Boot, set the following environment:

```
U-Boot > setenv ethaddr 00:04:9f:00:ea:d3            #[setup the MAC address]
U-Boot > setenv fec_addr 00:04:9f:00:ea:d3           $[setup the MAC address]
```

5.1.2 Boot from MMC/SD on SABRE-AI 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 on i.MX 6 series SABRE-AI boards.

download Mode(MFGTool 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)

Boot from SD

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

The default environment in boot.img is booting from SD. If you want to use the default environment in boot.img, you can use the following command:

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

To clear the bootargs env, you can use the following commands:

```
U-Boot > setenv bootcmd booti mmc2      #[Load the boot.img from SD]
U-Boot > setenv bootargs console=ttymxc3,115200 init=/init video=mxcfb0:dev=ldb,bpp=32
video=mxcfb1:off video=mxcfb2:off fbmem=10M vmalloc=400M androidboot.console=ttymxc3
androidboot.hardware=freescale  #[Optional]
U-Boot > saveenv   #[Save the environments]
```

NOTE

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

Some SoCs on SABRE-AI boards do not have MAC address fused.

Therefore, if you want to use FEC in U-Boot, set the following environment:

```
U-Boot > setenv ethaddr 00:04:9f:00:ea:d3            #[setup the
MAC address]
U-Boot > setenv fec_addr 00:04:9f:00:ea:d3           $[setup the
MAC address]
```

5.2 Boot from NAND

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

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

download Mode (MFGTool mode)	(S3) 0101 (from 1-4 bit)
NAND (Micro 29F8G08ABABA)	(S3) 0010 (from 1-4 bit) (S2) 0001 (from 1-4bit) (S1) 0001000000 (from 1-10 bit)

Boot from NAND

Change the board boot switch to (S3, S2,S1) 0010, 0001,0001000000 (from 1bit) on an i.MX 6 series SABRE-AI board.

The default environment in boot.img is booting from NAND. If you want to use the default environment in boot.img, you can use the following command:

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

To clear the bootargs env, you can use the following commands:

```
U-Boot > setenv bootcmd nand read 0x12800000 0x1000000 0x500000;booti 0x12800000
#[Load the boot.img from NAND]
U-Boot > setenv bootargs console=ttymxc3,115200 init=/init video=mxcfb0:dev=ldb,bpp=32
video=mxcfb1:off video=mxcfb2:off fbmem=10M vmalloc=400M androidboot.console=ttymxc3
androidboot.hardware=freescale mtdparts=gmi-nand:16m(bootloader),16m(bootimg),
128m(recovery),-(root) ubi.mtd3 #[Optional]
U-Boot > saveenv    $[Save the environments]
```

NOTE

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

Some SoCs on SABRE-AI boards do not have MAC address fused.

Therefore, if you want to use FEC in U-Boot, set the following environment:

```
U-Boot > setenv ethaddr 00:04:9f:00:ea:d3            #[setup the
MAC address]
U-Boot > setenv fec_addr 00:04:9f:00:ea:d3           $[setup the
MAC address]
```

5.3 Boot from TFTP and NFS

Set up the U-Boot environment for loading kernel from TFTP and mounting NFS as root file system after the U-Boot shell.

SABRE SD board

```
U-Boot > setenv loadaddr 0x10800000
U-Boot > setenv bootfile uImage
U-Boot > setenv serverip <your server ip>   #[Your TFTP/NFS server ip]
U-Boot > setenv nfsroot <your rootfs>       #[Your rootfs]
U-Boot > setenv bootcmd 'dhcp;bootm'        #[load kernel from TFTP and boot]
U-Boot > setenv bootargs console=ttymxc0,115200 init=/init ip=dhcp nfsroot=${serverip}:/$
{nfsroot} video=mxcfb0:dev=ldb,bpp=32 video=mxcfb1:off video=mxcfb2:off fbmem=10M
fb0base=0x27b00000 vmalloc=400M androidboot.console=ttymxc0 androidboot.hardware=freescale
U-Boot > saveenv   #[Save the environments]
```

After you have configured these settings, reboot the board, let U-Boot run the bootcmd environment to load kernel and run.

For the first time boot, finishing and getting to the Android UI takes some time.

5.4 Boot Up Configurations

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

5.4.1 U-Boot Environment

If you do not define the bootargs env, it will use the default bootargs inside the image.

- ethaddr/fec_addr is a MAC address of the board.
- serverip is an IP address of the TFTP/NFS server.
- loadaddr/rd_loadaddr is the kernel/initramfs image load address in memory.
- bootfile is the name of the image file loaded by "dhcp" command, when you are using TFTP to load kernel.
- 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 env is optional for booti. boot.img already has bootargs. If you do not define the bootargs env, it will use the default bootargs inside the image. If you have the env, it will be used.

  If you want to use default env in boot.img, you can use the following command to clear the bootargs env.

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

  booti command will parse the boot.img header to get the zImage and ramdisk. It will also pass the bootargs as needed (it will only pass bootargs in boot.img when it can't find "bootargs" var in your U-Boot env). To boot from mmcX, you need to do the following:

  ```
  > booti mmcX
  ```

  To read the boot partition (the partition store boot.img, in this case, mmcblk0p1), the X was the MMC bus number, which is the hardware MMC bus number, in i.MX 6Dual/6Quad SABRE-SD and i.MX 6Solo/6DualLite SABRE-SD boards. eMMC is mmc3.

  You can also add partition ID after mmcX.

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

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

  ```
  > booti 0xXXXXXXXX
  ```
- bootm (only works in for the NFS) starts running the kernel. For other cases, use booti command.
- splashimage is the virtual or physical address of bmp file in memory. If MMU is enabled in board configuration file, the address is virtual. Otherwise, it is physical. See README in U-Boot code root tree for details.
- splashpos sets the splash image to a free position, 'x,y', on the screen. x and y should be a positive number, which is used as number of pixel from the left/top. Note that the left and top should not make the image exceed the screen size. You can specify 'm,m' for centering the image. Usually, for example, '10,20', '20,m', 'm,m' are all valid settings. See README in U-Boot code root tree for details.
- lvds_num chooses which LVDS interface, 0 or 1, is used to show the splash image. Note that we only support boot splash on LVDS panel. We do not support HDMI or any other display device.

5.4.2 Kernel Command Line (bootargs)

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

Kernel Parameter	Description	Typical Value	Used When
console	Where to output kernel log by printk.	console=ttymxc0,115200	All cases.

Table continues on the next page...

init	Tells kernel where the init file is located.	init=/init	All cases. "init" in Android is located in "/" instead of in "/sbin".
ip	Tells kernel how/ whether to get an IP address.	ip=none or ip=dhcp or ip=static_ip_address	"ip=dhcp" or "ip=static_ip_address" is mandatory in "boot from TFTP/NFS".
nfsroot	Where the NFS server/directory is located.	rootfs=ip_address:/opt/ nfsroot,v3,tcp	Used in "boot from tftp/NFS" together with "root=/dev/nfs".
root	Indicates the location of the root file system.	root=/dev/nfs or root=/dev/mmcblk0p2	Used in "boot from tftp/NFS" (i.e. root=/dev/ nfs). Used in "boot from SD" (i.e. root=/dev/ mmcblk0p2) if no ramdisk is used for root fs.
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
fbmem	framebuffer reservation size configuration	fbmem=5M,10M	fbmem=<fb0 size>,<fb2 size>,<fb4 size>,<fb5 size>
vmalloc	vmalloc virtual range size for kernel	vmalloc=400M	vmalloc=<size>
enable_wait_mode	Enables or disable the i.MX 6 WAIT mode.	enable_wait_mode=on	enable_wait_mode=<on/off> Wait mode is enabled by default for this release.
arm_freq	Lets CPU work at special frequency(MHz).	arm_freq=800	If you want to specify CPU working frequency
androidboot.console	The Android shell console. It should be the same as console=.	androidboot.console=ttymxc0	If you want to use the default shell job control, such as Ctrl+C to terminate a running process, you must 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 MAC address fused in. If you want to use FEC, assign this parameter to the kernel.
fb0base	Tells kernel the framebuffer base address that bootloader uses	fb0base=0x27b00000	This is only for Hannstar XGA LVDS panel. To choose to support smooth UI transition from bootloader to Kernel, you need to set this. When this is set, you have to set

Table continues on the next page...

	for splashscreen. Kernel will use the address for framebuffer.		'fbmem' for fb0 to reserve fb memory as well.
gpumem	Sets up the reserved memory size for GPU driver	gpumem=192 M	It is 192 M by default on the SABRE-SD platform. This kernel parameter can only be used when limiting GPU/VPU usage.
Bluetooth	Tells kernel to enable Bluetooth	Bluetooth	This is only for Sabre-SD Rev C board. It will use UART5 to enable BT support.
ldo_active	Enable or disable LDO bypass.	ldo_active=off ldo_active=on	By defaut, LDO bypass is enabled. If you want to use internal LDO, specify "ldo_active=on" to the kernel command line. LDO bypass can only be enabled on the board that mounted with external PMIC to supply VDDARM_IN/VDDSOC_IN power rail
caam	Enable/disable CAAM module.	caam	By default, CAAM is disabled. If you want to use CAAM module, specify "caam" to the kernel command line. CAAM uses ALT7 mode of pad GPIO_0, which conflicts with any other module that using pad GPIO_0 on the board. On an i.MX 6 series SABRE-SD board, CAAM conflicts with audio codec (WM8962) and camera(ov5642) module.
bluetooth	Tells kernel to enable bluetooth.	bluetooth	This is only for SABRE-SD Rev C board. It uses UART 5 to enable the BT support.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AUG
Rev jb4.2.2_1.1.0-GA
07/2013