Document Number: AUG
Rev KK4.4.3_2.0.0-ga, 02/2015

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. In addition, it provides the information on how to get Android source from Google, and what the device storage usage and boot options are. For more information, see source.android.com/source/building.html.

2 Preparation

2.1 Setting up your computer

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

You also need to use the 10.10, 11.04, or 12.04 bit version of Ubuntu which are the most tested OS for the Android kk4.4.3 build.

After installing the Linux OS 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 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 add-apt-repository ppa:git-core/ppa
$ sudo apt-get install lzop
$ sudo apt-get update
$ sudo apt-get install git-core curl
$ sudo apt-get install u-boot-tools
```

NOTE

If you have trouble in installing the Sun's JDK in Ubuntu, refer to community.freescale.com/docs/DOC-98441.

2.2 Unpacking 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_KK4.4.3_2.0.0-
ga_core_source.tar.gz file)

$ tar xzvf android_KK4.4.3_2.0.0-ga_core_source.tar.gz

$ cd android_KK4.4.3_2.0.0-ga_core_source/code/

$ tar xzvf KK4.4.3_2.0.0-ga.tar.gz
```

2.3 Running Android with a prebuilt image

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

Table 1. Image packages

Image package	Description
android_KK4.4.3_2.0.0-ga_core_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, which has basic Android features.
android_KK4.4.3_2.0.0-ga_core_image_6qsabreauto.tar.gz	The table below shows the prebuilt image for i.MX 6Dual/ 6Quad and i.MX 6Solo/6DualLite SABRE-AI platform, which has basic Android features.
android_KK4.4.3_2.0.0-ga_core_image_6slevk.tar.gz	The table below shows the prebuilt image for the i.MX 6SoloLite EVK board, which has basic Android features.
android_KK4.4.3_2.0.0-ga_core_image_6sxsabresd.tar.gz	Prebuilt image for the i.MX 6SoloX Sabre-SD board, which has basic Android features.
android_KK4.4.3_2.0.0-ga_full_image_6qsabresd.tar.gz	The table below shows the prebuilt image for i.MX 6Dual/ Quad and i.MX 6DualLite SABRE-SD board, which includes Freescale extended features.

Table continues on the next page...

Table 1. Image packages (continued)

	For more information and details about the Freescale Extended Feature Packages available for this release, refer to the *Android Release Notes*.
android_KK4.4.3_2.0.0-ga_core_image_6sxsabreauto.tar.gz	Prebuilt image for i.MX 6SoloX SABRE-AI board, which has basic Android features.
android_KK4.4.3_2.0.0-ga_full_image_6qsabreauto.tar.gz	The table below shows the prebuilt image for i.MX 6Dual/Quad and i.MX 6DualLite/Solo SabreAI board, which includes Freescale extended features. For more information and details about the Freescale Extended Feature Packages available for this release, refer to the *Android Release Notes*.
android_KK4.4.3_2.0.0-ga_full_image_6slevk.tar.gz	Prebuilt image for the i.MX 6SoloLite EVK board, which includes Freescale extended features.
android_KK4.4.3_2.0.0-ga_full_image_6sxsabresd.tar.gz	Prebuilt image for the i.MX 6SoloX SABRE-SD board, which includes Freescale extended features.
android_KK4.4.3_2.0.0-ga_full_image_6sxsabreauto.tar.gz	Prebuilt image for i.MX 6SoloX SABRE-AI board, which includes Freescale extended features.

The following tables list the detailed contents of android_KK4.4.3_2.0.0-ga_core_image_6qsabresd.tar.gz and android_KK4.4.3_2.0.0-ga_full_image_6qsabresd.tar.gz image packages.

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

Table 2. 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 i.MX 6Dual/6Quad SABRE-SD board and platform
u-boot-imx6dl.imx	The bootloader (with padding) for i.MX 6Solo/6DualLite SABRE-SD platform
eMMC/boot-imx6dl.img eMMC/boot-imx6q.img eMMC/boot-imx6q-ldo.img	Boot Image for eMMC
eMMC/system.img	System Boot Image
eMMC/recovery-imx6dl.img eMMC/recovery-imx6q.img eMMC/recovery-imx6q-ldo.img	Recovery Image

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

Table continues on the next page...

Table 3. Images for SABRE-SD SD (continued)

u-boot-6dl.bin	The bootloader (with padding) for the i.MX 6DualLite SABRE-SD board
SD/boot-imx6dl.img SD/boot-imx6q.img SD/boot-imx6q-ldo.img	Boot image for SD
SD/system.img	System boot image
SD/recovery-imx6dl.img SD/recovery-imx6q.img SD/recovery-imx6q-ldo.img	Recovery image

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

Table 4. Images for i.MX 6 SABRE-SD TFTP and NFS

SABRE-SD TFTP/NFS image	Description
u-boot-imx6q.imx	The bootloader (with padding) for the i.MX 6Dual/6Quad SABRE-SD board and platform
u-boot-imx6dl.imx	The bootloader (with padding) for the i.MX 6Solo/6DualLite SABRE-SD platform
NFS/android_fs.tar.gz	NFS rootfs
NFS/uImage	Kernel image for TFTP
NFS/imx6q-sabresd.dtb NFS/imx6q-sabresd-ldo.dtb	Kernel device tree binary for the i.MX 6Dual/Quad SABRE-SD board
NFS/imx6dl-sabresd.dtb	Kernel device tree binary for the i.MX 6DualLite SABRE-SD board

The following tables list the detailed contents of android_KK4.4.3_2.0.0-ga_core_image_6qsabreauto.tar.gz and android_KK4.4.3_2.0.0-ga_full_image_6qsabreauto.tar.gz image packages:

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

Table 5. Images for i.MX 6 SABRE-AI SD boot

SABRE-AI SD image	Description
u-boot-imx6q.imx	The bootloader (with padding) for i.MX 6Dual/6Quad SABRE-AI SD boot
u-boot-imx6dl.imx	The bootloader (with padding) for i.MX 6DualLite SABRE-AI SD boot
SD/boot-imx6q.img SD/boot-imx6dl.img	Boot Image for SD
SD/system.img	System Boot Image
SD/recovery-imx6q.img SD/recovery-imx6dl.img	Recovery Image

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

Table 6. Images for i.MX 6 SABRE-AI NAND boot

SABRE-AI NAND image	Description
u-boot-imx6q-nand.imx	The bootloader (with padding) for i.MX 6Dual/6Quad SABRE-AI NAND boot
u-boot-imx6dl-nand.imx	The bootloader (with padding) for i.MX 6DualLite SABRE-AI NAND boot
NAND/boot-imx6q-nand.img NAND/boot-imx6dl-nand.img	Boot Image for NAND
NAND/android_root.img	System Boot Image
NAND/recovery-imx6q-nand.img NAND/recovery-imx6dl-nand.img	Recovery Image

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

Table 7. Images for i.MX 6 SABRE-AI TFTP and NFS

SABRE-AI TFTP/NFS image	Description
u-boot-imx6q.imx	The bootloader (with padding) for i.MX 6Dual/6Quad SABRE-AI SD boot
u-boot-imx6dl.imx	The bootloader (with padding) for i.MX 6DualLite SABRE-AI boot
NFS/android_fs.tar.gz	NFS rootfs
NFS/uImage	Kernel image for TFTP
NFS/imx6q-sabreauto.dtb	Kernel device tree binary for the i.MX 6Dual/Quad SABRE-AI board
NFS/imx6dl-sabreauto.dtb	Kernel device tree binary for the i.MX 6DualLite SABRE-AI board

The following tables list the detailed contents of android_KK4.4.3_2.0.0-ga_core_image_6slevk.tar.gz image package.

The table below shows the prebuilt images to support the system boot from SD on the i.MX 6SoloLite EVK board.

Table 8. Images for i.MX 6SoloLite EVK SD

EVK SD image	Description
u-boot-imx6sl.imx	Bootloader (with padding) for the i.MX 6SoloLite EVK board
SD/boot-imx6sl.img	Boot image for SD
SD/system.img	System boot image
SD/recovery-imx6sl.img	Recovery image

The table below shows the prebuilt images to support the system boot from TFTP server and NFS rootfs on the i.MX 6SoloLite EVK board.

Table 9. Images for i.MX 6SoloLite EVK TFTP and NFS

EVK TFTP/NFS image	Description
u-boot-imx6sl.imx	Bootloader (with padding) for the i.MX 6SoloLite EVK board
NFS/android_fs.tar.gz	NFS rootfs
NFS/uImage	Kernel image for TFTP
NFS/imx6sl-evk-csi.dtb	Kernel device tree binary for the i.MX 6SoloLite EVK board

The following tables list the detailed contents of the android_KK4.4.3_2.0.0-ga_core_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 10. Images for i.MX 6SoloX SABRE-SD SD

i.MX 6SoloX SABRE-SD SD image	Description
u-boot-imx6sx.imx	Bootloader (with padding) for the i.MX 6SoloX SABRE-SD board
SD/boot-imx6sx.img	Boot image for the i.MX 6SoloX SABRE-SD board
SD/system.img	System boot image
SD/recovery-imx6sx.img	Recovery image for the i.MX6SoloX SABRE-SD board

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

Table 11. Images for i.MX 6SoloX SABRE-SD TFTP and NFS

i.MX 6SoloX SABRE-SD TFTP/NFS image	Description
u-boot-imx6sx.imx	Bootloader (with padding) for the i.MX 6SoloLite EVK board
NFS/android_fs.tar.gz	NFS rootfs
NFS/uImage	Kernel image for TFTP
NFS/imx6sx-sdb.dtb	Kernel device tree binary for the i.MX 6SoloX SABRE-SD board

The following tables list the detailed contents of android_KK4.4.3_2.0.0-ga_core_image_6sxsabreauto.tar.gz and android_KK4.4.3_2.0.0-ga_full_image_6sxsabreauto.tar.gz image packages.

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

Table 12. Images for i.MX 6SoloX SABRE-AI SD

i.MX 6SoloX SABRE-AI SD image	Description
u-boot-imx6sx.imx	Bootloader (with padding) for i.MX 6SoloX SABRE-AI SD boot.
SD/boot-imx6sx.img	Boot image for SD.
SD/system.img	System boot Image.
SD/recovery-imx6sx.img	Recovery image.

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

Table 13. Images for i.MX 6SoloX SABRE-AI NAND

i.MX 6SoloX SABRE-AI NAND image	Description
u-boot-imx6sx-nand.imx	Bootloader (with padding) for i.MX 6SoloX NAND boot
NAND/boot-imx6sx.img	Boot image for NAND
NAND/android_root.img	System Boot image
NAND/recovery-imx6sx.img	Recovery image

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

Table 14. Images for i.MX 6SoloX SABRE-AI TFTP and NFS

i.MX 6SoloX SABRE-AI TFTP and NFS image	Description
u-boot-imx6sx.imx	Bootloader (with padding) for i.MX6SoloX SABRE-AI SD boot
NFS/android_fs.tar.gz	NFS rootfs
NFS/zImage	Kernel image for TFTP
NFS/imx6sx-sabreauto.dtb	Kernel device tree binary for the i.MX 6SoloX SABRE-AI board

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. This information can be configured in corresponding board's BoardConfig.mk. If you use boot.img, you do not need uImage and uramdisk.img.

3 Building Android for i.MX

3.1 Getting 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:

```
$ cd ~
$ mkdir myandroid
$ mkdir bin
$ cd myandroid
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ ~/bin/repo init -u https://android.googlesource.com/platform/manifest -b android-4.4.3_r1
$ ~/bin/repo sync # this command loads most needed repos. Therefore, it can take several
hours to load.
```

Get KK4.4.3_2.0.0-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 kk4.4.3_2.0.0-ga
```

NOTE

If you are behind a 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 kk4.4.3_2.0.0-ga
```

3.2 Patch code for i.MX

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

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

1. Assume you have unzipped the i.MX Android release package to `/opt/android_KK4.4.3_2.0.0-ga_source`.

   ```
   $ cd ~/myandroid
   $ source /opt/android_KK4.4.3_2.0.0-ga_core_source/code/KK4.4.3_2.0.0-ga/and_patch.sh
   $ help
   ```
2. You should see that the "c_patch" function is available.

   ```
   $ c_patch /opt/android_KK4.4.3_2.0.0-ga_core_source/code/KK4.4.3_2.0.0-ga
   imx_KK4.4.3_2.0.0-ga
   ```

   Here, "/opt/android_KK4.4.3_2.0.0-ga_source/code/KK4.4.3_2.0.0-ga" is the location of the patches, which is the directory created when you unzip the release package.

   "imx_KK4.4.3_2.0.0-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 instead of "imx_KK4.4.3_2.0.0-ga".
3. If everything is OK, "c_patch" will generate the following output to indicate the 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 Patch Freescale extended features code

Besides the core features code package, Freescale also provides extended features code packages. These extended features code packages bring more features. The following tables list the extended features code packages.

Table 15. Extended features code package

Package name	Package Description
android_KK4.4.3_2.0.0-ga_omxplayer_source.tar.gz	Source code package, binaries and scripts for using omx player
android_KK4.4.3_2.0.0-ga_wfdsink_source.tar.gz	Source code package to add Freescale WIFI Sink support into the core features code package

To apply android_KK4.4.3_2.0.0-ga_omxplayer_source.tar.gz patches, carry out the following commands:

```
$ cp android_KK4.4.3_2.0.0-ga_omxplayer_source.tar.gz ~/myandroid
$ cd ~/myandroid
$ tar xzvf android_KK4.4.3_2.0.0-ga_omxplayer_source.tar.gz
```

After unpacking this package, the current build mode is “full”, which means omxplayer feature will be built into Android.

Use switch_build_to.sh to switch between “core” mode and “full” mode.

```
$ ./switch_build_to.sh          (display current mode)
$ ./switch_build_to.sh mode     (mode shall be "core" or "full")
$ ./clean_obj_before_building.sh (clean related binary in the out directory of previous
android build)
```

NOTE

Run this script after running “lunch” and before running “make”.

To apply android_KK4.4.3_2.0.0-ga_wfdsink_source.tar.gz patches, carry out the following commands:

```
$ cd /opt (or any other directory you like)
$ tar xzvf android_KK4.4.3_2.0.0-ga_wfdsink_source.tar.gz
$ cp -a android-KK4.4.3_2.0.0-ga_wfdsink_source/wfd-proprietary ~/myandroid/device/
```

3.4 Building Android images

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

```
$ cd ~/myandroid
$ source build/envsetup.sh
$ lunch sabresd_6dq-user
$ make 2>&1 | tee build_sabresd_6dq_android.log
"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.MX 6 SABRE-SD boards
$ lunch sabresd_6dq-user
$ make 2>&1 | tee build_sabresd_6dq_android.log
#Build Android images for i.MX 6 SABRE-AI boards
$ lunch sabreauto_6dq-user
$ make 2>&1 | tee build_sabreauto_6dq_android.log
#Build Android images for i.MX 6SoloLite EVK boards
$ lunch evk_6sl-user
$ make 2>&1 | tee build_evk_6sl_android.log
#Build Android images for i.MX6SoloX SABRE-SD boards
$ lunch sabresd_6sx-user
$ make 2>&1 | tee build_sabresd_android.log
#Build Android images for the i.MX 6SoloX SABRE-AI boards
```

```
$ lunch sabreauto_6sx-user
$ make 2>&1 | tee build_sabreauto_android.log
```

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/: root file system (including init, init.rc, etc). 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-imx6q.img: composite image for i.MX 6Dual/6Quad SABRE-SD. which includes the kernel zImage, ramdisk, board's device tree binary, and boot parameters.
- boot-imx6dl.img: composite image for i.MX 6DualLite SABRE-SD, which includes the kernel zImage, ramdisk, board's device tree binary, and boot parameters.
- boot-imx6q-ldo.img: a composite image for i.MX 6DualQuad 1.2GHZ SABRE-SD, which includes the kernel zImage, ramdisk, board's device tree binary, and boot parameters.
- 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 6Dual/6Quad 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 6Dual/6Quad 1.2GHZ SABRE-SD, which is generated from "recovery/". Can be programmed to "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".
- u-boot-imx6q.imx: U-Boot image with no padding for i.MX 6Dual/6Quad SABRE-SD.
- u-boot-imx6dl.imx: U-Boot image with no padding for i.MX 6DualLite SABRE-SD.

NOTE

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

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

- 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.

3.4.1 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.

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

- Set ro.secure=1, and ro.debuggable=0. adb is disabled by default.

```
# Build images for the i.MX 6Dual/Quad and i.MX 6DualLite SABRE-SD board
$ make PRODUCT-sabresd_6dq-user 2>&1 | tee build_sabresd_6dq_android.log
```

```
# Build Images for the i.MX 6Dual/Quad and i.MX 6DualLite SABRE-AI board
$ make PRODUCT-sabreauto_6q-user 2>&1 | tee build_sabreauto_6dq_android.log

# Build Images for the i.MX 6SoloLite EVK board
$ make PRODUCTevk_6sluser 2>&1 | tee build_evk_6sl_android.log

# Build images for the i.MX 6SoloX SABRE-SD board
$ make PRODUCT-sabresd_6sx-user 2>&1 | tee build_sabresd_6sx_android.log

# Build images for the i.MX 6SoloX SABRE-AI board
$ make PRODUCT-sabreauto_6sx-user 2>&1 | tee build_sabreauto_6sx_android.log
```

Or you can use the following commands:

```
# Build images for i.MX 6Dual/Quad and i.MX 6DualLite SABRE-SD board
$ source build/envsetup.sh
$ lunch sabresd_6dq-user
$ make
$ make otapackage # you can generate the OTA package with this command.

# Build images for i.MX 6Dual/Quad and i.MX 6DualLite SABRE-AI board
$ source build/envsetup.sh
$ lunch sabreauto_6q-user
$ make
$ make otapackage # you can generate the OTA package with this command.

# Build images for i.MX 6SoloLite EVK board
$ source build/envsetup.sh
$ lunch evk_6sluser
$ make
$ make otapackage # you can generate the OTA package with this command.

# Build images for the i.MX 6SoloX SABRE-SD board
$ source build/envsetup.sh
$ lunch sabresd_6sx-user
$ make
$ make otapackage # you can generate the OTA package with this command.

# Build images for the i.MX 6SoloX SABRE-AI board
$ source build/envsetup.sh
$ lunch sabreauto_6sx-user
$ make
$ make otapackage # you can generate ota package with this command.
```

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

3.4.2 Building Android image for the SD card on the SABRE-SD Board

The default configuration in the source code package takes internal eMMC as the boot storage. It can be changed to make the SD card in SD Slot 3 as the boot storage as follows:

```
remove out/target/product/sabresd_6dq/root directory and boot*.img
remove /out/target/product/sabresd_6dq/recovery directory and recovery*.img
make bootimage BUILD_TARGET_DEVIC=sd
```

Follow Section 3.4 to build the images.

3.4.3 Building Android images for NAND on the SABRE-AI board

i.MX 6Quad/DualLite/Solo SABRE-AI platform:

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
```

i.MX 6SoloX SABRE-AI platform:

The default configuration in the source code package takes SD as the boot storage for SABRE-AI. The default setting can be changed to make the NAND Flash in U19 to be the boot storage as shown below:

```
make PRODUCT-sabreauto_6sx-user BUILD_TARGET_FS=ubifs
```

3.5 Building 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-
$ make distclean
```

For i.MX 6Quad SABRE-SD:

```
$ make mx6qsabresdandroid_config
$ make
```

For i.MX 6DualLite SABRE-SD:

```
$ make mx6dlsabresdandroid_config
$ make
```

For i.MX 6Solo SABRE-SD:

```
$ make mx6solosabresdandroid_config
```

For i.MX 6Quad SABRE-AI SD:

```
$ make mx6qsabreautoandroid_config
$ make
```

For i.MX 6Quad SABRE-AI NAND:

```
$ make mx6qsabreautoandroid_nand_config
$ make
```

For i.MX 6DualLite SABRE-AI SD:

```
$ make mx6dlsabreautoandroid_config
$ make
```

For i.MX 6DualLite SABRE-AI NAND:

```
$ make mx6dlsabreautoandroid_nand_config
$ make
```

For i.MX 6Solo SABRE-AI SD:

```
$ make mx6solosabreautoandroid_config
$ make
```

For i.MX 6Solo SABRE-AI NAND:

```
$ make mx6solosabresdandroid_nand_config
```

For i.MX 6SoloLite EVK SD:

```
$ make mx6slevkandroid_config
```

For i.MX 6SoloX SABRE-SD SD:

```
$ make mx6sxsabresdandroid_config
```

For i.MX 6SoloX SABRE-AI SD:

```
$ make mx6sxsabreautoandroid_config
```

For i.MX 6SoloX SABRE-AI NAND:

```
$ make mx6sxsabreautoandroid_nand_config
$ make
```

"u-boot.imx" is generated if you have a successful build.

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.

3.6 Building a kernel image

Kernel image will be built 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-
# Generate ".config" according to default config file under arch/arm/configs.
# To build the kernel image for i.MX 6Dual/Quad, 6DualLite, 6Solo, 6SoloLite, and 6SoloX
$ make imx_v7_android_defconfig

# To build the kernel image for i.MX6Dual/Quad, 6DualLite, and 6Solo
$ make uImage LOADADDR=0x10008000

# To build the kernel image for i.MX 6SoloLite
$ make uImage LOADADDR=0x80008000

# To build the kernel image for i.MX 6SoloX
$ make uImage LOADADDR=0x80008000
```

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

3.7 Building boot.img

boot.img and booti are default booting commands.

As outlined in Running Android with a 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 the SABRE-SD board
$ cd ~/myandroid
$ source build/envsetup.sh
$ lunch sabresd_6dq-user
$ make bootimage

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

# Boot image for the i.MX 6SoloLite EVK board
$ source build/envsetup.sh
$ lunch evk_6sl-user
$ make bootimage

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

# Boot image for the i.MX 6SoloX SABRE-AI board
$ source build/envsetup.sh
$ lunch sabreauto_6sx-user
$ make bootimage
```

4 Downloading 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 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
- Android system root image: system.img
- Recovery root image: recovery.img

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 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
Primary 1	Boot	8 MB	8 MB	boot.img format, kernel + ramdisk	boot.img
Primary 2	Recovery	Follow Boot	8 MB	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	4 MB	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\Linux\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 -f <soc_name> /dev/sdX
# <soc_name> can be as imx6q, imx6dl, imx6sl, and imx6sx.
```

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.
- Unmount all the SD card partitions before running the script.

4.1.2 Downloading 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 Downloading 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.imx 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 supports running the system on the 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 Setting up the 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_KK4.4.3_2.0.0-ga_image_6qsabresd/NFS

$ tar xzvf ./android_fs.tar.gz

$ cd android_fs

$ rm -rf /opt/nfsroot/*

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

- If you built 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 the SABRE-AI board

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

- U-Boot image: u-boot-imx6q-nand.imx, u-boot-imx6dl-nand.imx, u-boot-imx6solo-nand.imx
- 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 from source.

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 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	64 MB	N/A	bootloader
MTD1	Boot	64 MB	16 MB	boot.img format, kernel + ramdisk	boot.img
MTD2	Recovery	Follow Boot	16 MB	boot.img format, kernel + ramdisk	recovery.img
MTD3(ubi0:system)	SYSTEM	Follow Recovery	360 MB	ubifs. Mount as / system	Android system files under / system/ dir.
MTD3(ubi0:cache)	CACHE	Follow SYSTEM.	200 MB	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	300 MB	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 Downloading 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 Booting

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

5.1 Booting from MMC/SD

This section describes booting from MMC/SD on the SABRE-SD board, on the SABRE-AI board, and on the EVK board.

5.1.1 Booting from MMC/SD on the 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 (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, perform the following operations:

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 environment, you can use the following commands:

```
U-Boot > setenv fastboot_dev mmc2 [eMMC as fastboot deivce]
U-Boot > setenv bootcmd booti mmc2      [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 video=mxcfb3:off vmalloc=400M androidboot.console=ttymxc0
consoleblank=0  androidboot.hardware=freescale cma=384M [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 environment.

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]
```

To boot from the SD card, perform the following operations:

Change the board boot switch to 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 mmc1  [eMMC as fastboot deivce]
U-Boot > setenv bootcmd booti mmc1         [Load the boot.img from SD card]
U-Boot > setenv bootargs console=ttymxc3,115200 init=/init video=mxcfb0:dev=ldb,bpp=32
video=mxcfb1:off video=mxcfb2:off video=mxcfb3:off vmalloc=400M androidboot.console=ttymxc3
consoleblank=0 androidboot.hardware=freescale cma=384M  mtdparts=gpmi-nand:16m(bootloader),
16m(bootimg),16m(recovery),-(root) ubi.mtd=4   #[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 environment.

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 Booting from MMC/SD on the the 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)

To boot from SD, perform the following operations:

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 environment, you can use the following command:

```
U-Boot > setenv bootcmd booti mmc1      #[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 video=mxcfb3:off vmalloc=400M androidboot.console=ttymxc3
consoleblank=0 androidboot.hardware=freescale cma=384M  #[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 environment.

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]
```

5.1.3 Booting from SD on the i.MX 6SoloLite EVK board

Below are the Boot Switch settings to control the boot storage:

Download mode (MFGTool mode)	SW3: 01000000(from 1-8 bit) SW4: 00101100 (from 1-8 bit) SW5: 00000000(from 1-8 bit) Boot_Mode: 10 (from 1-2 bit)
SD boot	SW3: 01000000(from 1-8 bit) SW4: 00101100 (from 1-8 bit) SW5: 00000000(from 1-8 bit) Boot_Mode: 01 (from 1-2 bit)

To boot from SD, perform the following operations:

Chang the board Boot_Mode switch to 01 (from 1-2 bit) and (SW3,4,5) 01000000 0010110000000000 (from 1-8 bit).

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

```
UBoot > setenv bootargs
```

To clear the bootargs environment, you can use the following command:

```
UBoot > setenv bootcmd booti mmc1
UBoot > setenv bootargs console=ttymxc0,115200 init=/init androidboot.console=ttymxc0
consoleblank=0 androidboot.hardware=freescale   #[Optional]
UBoot > saveenv                       [Save the environments]
```

NOTE

bootargs env is an optional setting for booti. 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 EVK boards, do not fuse MAC address. You need to set the following environment if you want to use FEC in U-Boot:

```
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]
```

5.1.4 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 to control the boot storage:

Download mode (MFGTool mode)	SW10: 00000000 (from 1-8 bit) SW11: 00111000 (from 1-8 bit)

Table continues on the next page...

	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:

Chang 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 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 environment, you can use the following command:

```
U-Boot > setenv bootcmd booti mmc2
U-Boot > setenv bootargs console=ttymxc0,115200 init=/init androidboot.console=ttymxc0
consoleblank=0 androidboot.hardware=freescale cma=384M      [Optional]
U-Boot > saveenv                          [Save the environments]
```

NOTE

bootargs env is an optional setting for booti. 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. You need to set the following environment if you want to use FEC in U-Boot:

```
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 Booting from NAND for the SABRE-AI board

5.2.1 Booting from NAND on the i.MX 6DualQuad/6DualLite SABRE-AI board

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

Download mode (MFGTool mode)	(S3): 0101 (from 1-4 bit)
NAND (Micro 29F8G08ABACA)	(S3): 0010 (from 1-4 bit) (S2): 0001 (from 1-4 bit) (S1): 0001000000 (from 1-10 bit)

Boot from NAND

Change the board boot switch to (S3, S2,S1) 0010, 0001,0001000000 (from 1 bit) 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:

```
UBoot > setenv bootcmd 'nand read 0x12000000 0x4000000 0x1000000;booti 0x12000000'
#[Load the boot.img from NAND]
UBoot > setenv bootargs console=ttymxc3,115200 init=/init video=mxcfb0:dev=ldb,bpp=32
video=mxcfb1:off video=mxcfb2:off video=mxcfb3:off vmalloc=400M androidboot.console=ttymxc3
consoleblank=0 androidboot.hardware=freescale cma=384M mtdparts=gpminand: 64m(bootloader),
16m(bootimg),16m(recovery),(root) ubi.mtd=4   [Optional]
UBoot > 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 environment.

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]
```

5.2.2 Booting from NAND on the i.MX 6SoloX SABRE-AI board

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

Download mode (MFGTool mode)	S1 (Boot_Mode): 0101 (from 1-4 bit)
NAND (Micro 29F8G08ABACA)	S1 (Boot_Mode): 0010 (from 1-4 bit) SW3: 00000000 (from 1-8bit) SW4: 00000001 (from 1-8bit)

Boot from NAND

Change the board boot switch to (S1, S3,S4) 0010, 00000000,00000001 (from 1bit) on an i.MX 6SoloX 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:

```
UBoot > setenv bootcmd 'nand read 0x80800000 0x4000000 0x1000000;booti0x12000000' #[Load the
boot.img from NAND]
UBoot > setenv bootargs console=ttymxc0,115200 init=/init video=mxcfb0:dev=ldb,bpp=32
video=mxcfb1:off video=mxcfb2:off video=mxcfb3:off vmalloc=400M androidboot.console=ttymxc0
consoleblank=0 androidboot.hardware=freescale cma=384M mtdparts=gpminand: 64m(bootloader),
16m(bootimg),16m(recovery),(root)
ubi.mtd=4 [Optional]
UBoot > 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 environment.

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:

```
UBoot > setenv ethaddr 00:04:9f:00:ea:d3   [setup the MAC address]
UBoot > setenv fec_addr 00:04:9f:00:ea:d3  [setup the MAC address]
```

5.3 Booting 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.

5.3.1 i.MX 6Quad and i.MX 6DualLite SABRE-SD platforms

```
U-Boot > setenv get_cmd dhcp
U-Boot > setenv loadaddr 0x12000000
U-Boot > setenv bootfile zImage
U-Boot > setenv serverip <your server ip>       [Your TFTP/NFS server ip]
U-Boot > setenv nfsroot <your rootfs>        [Your rootfs]
U-Boot > setenv bootcmd 'run netboot'         [load kernel from TFTP and boot]
U-Boot > setenv fdt_file <your dtb file>      [Your DTB file]
U-Boot > setenv fdt_addr 0x18000000
U-Boot > setenv netargs 'setenv bootargs console=ttymxc0,115200 consoleblank=0 init=/init
root=/dev/nfs rw ip=dhcp nfsroot=${serverip}:${nfsroot},v3,tcp console=ttymxc0,115200 init=/
init androidboot.console=ttymxc0 androidboot.hardware=freescale cma=384M'
U-Boot > setenv netboot 'echo Booting from net ...; run netargs;${get_cmd} ${bootfile };$
{get_cmd} ${fdt_addr} ${fdt_file};bootz ${loadaddr} - ${fdt_addr};'
U-Boot > saveenv                      [Save the environments]
```

NOTE

If there is no execute permission for rootfs/system/bin/*, you can use the command in PC "chmod 777 /system/bin/*" and "chmod 777 /data" to update the file/directory access permission.

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

For the first-time boot, it takes some time to finish and get to the Android UI.

5.3.2 i.MX 6Quad and i.MX 6DualLite SABRE-AI platforms

```
U-Boot > setenv get_cmd dhcp
U-Boot > setenv loadaddr 0x12000000
U-Boot > setenv bootfile zImage
U-Boot > setenv serverip <your server ip>       [Your TFTP/NFS server ip]
U-Boot > setenv nfsroot <your rootfs>        [Your rootfs]
U-Boot > setenv bootcmd 'run netboot'         [load kernel from TFTP and boot]
U-Boot > setenv fdt_file <your dtb file>      [Your DTB file]
U-Boot > setenv fdt_addr 0x18000000
U-Boot > setenv netargs 'setenv bootargs console=ttymxc3,115200 init=/init root=/dev/nfs rw
ip=dhcp nfsroot=${serverip}:${nfsroot},v3,tcp video=mxcfb0:dev=ldb,bpp=32 video=mxcfb1:off
video=mxcfb2:off video=mxcfb3:off vmalloc=400M androidboot.console=ttymxc3 consoleblank=0
androidboot.hardware=freescale cma=384M'
U-Boot > setenv netboot 'echo Booting from net ...; run netargs;${get_cmd} ${bootfile };$
{get_cmd} ${fdt_addr} ${fdt_file};bootz ${loadaddr} - ${fdt_addr};'
U-Boot > saveenv                      [Save the environments]
```

Note: If there is no execute permission for rootfs/system/bin/*, you can use the command in PC "chmod 777 /system/bin/*" and "chmod 777 /data" to update the file or directory access permission.

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

For the first-time boot, it takes some time to finish and get to the Android UI.

5.3.3 i.MX 6SoloLite EVK platform

```
U-Boot > setenv get_cmd dhcp
U-Boot > setenv loadaddr 0x80800000
U-Boot > setenv bootfile zImage
U-Boot > setenv serverip <your server ip>       [Your TFTP/NFS server ip]
U-Boot > setenv nfsroot <your rootfs>        [Your rootfs]
U-Boot > setenv bootcmd 'run netboot'         [load kernel from TFTP and boot]
U-Boot > setenv fdt_file <your dtb file>      [Your DTB file]
U-Boot > setenv fdt_addr 0x81700000
U-Boot > setenv netargs 'setenv bootargs console=ttymxc0,115200 init=/init root=/dev/nfs rw
ip=dhcp nfsroot=${serverip}:${nfsroot},v3,tcp console=ttymxc0,115200 init=/init
androidboot.console=ttymxc0 consoleblank=0 androidboot.hardware=freescale'
U-Boot > setenv netboot 'echo Booting from net ...; run netargs;${get_cmd} ${bootfile };$
{get_cmd} ${fdt_addr} ${fdt_file};bootz ${loadaddr} - ${fdt_addr};'
U-Boot > saveenv                        [Save the environments]
```

Note: If there is no execute permission for rootfs/system/bin/*, you can use the command in PC "chmod 777 /system/bin/*" and "chmod 777 /data" to update the file or directory access permission.

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

For the first-time boot, it takes some time to finish and get to the Android UI.

5.3.4 i.MX 6SoloX SABRE-SD and SABRE-AI platforms

```
U-Boot > setenv get_cmd dhcp
U-Boot > setenv loadaddr 0x80800000
U-Boot > setenv bootfile zImage
U-Boot > setenv serverip <your server ip>       [Your TFTP/NFS server ip]
U-Boot > setenv nfsroot <your rootfs>        [Your rootfs]
U-Boot > setenv bootcmd 'run netboot'         [load kernel from TFTP and boot]
U-Boot > setenv fdt_file <your dtb file>      [Your DTB file]
U-Boot > setenv fdt_addr 0x81700000
U-Boot > setenv netargs 'setenv bootargs console=ttymxc0,115200 consoleblank=0 init=/init
root=/dev/nfs rw ip=dhcp nfsroot=${serverip}:${nfsroot},v3,tcp console=ttymxc0,115200 init=/
init androidboot.console=ttymxc0 androidboot.hardware=freescale cma=384M'
U-Boot > setenv netboot 'echo Booting from net ...; run netargs;${get_cmd} ${bootfile };$
{get_cmd} ${fdt_addr} ${fdt_file};bootz ${loadaddr} - ${fdt_addr};'
U-Boot > saveenv                        [Save the environments]
```

If there is no execute permission for rootfs/system/bin/*, you can use the command in PC "chmod 777 /system/bin/*" and "chmod 777 /data" to update the file or directory access permission.

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

For the first-time boot, it takes some time to finish and get to the Android UI.

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 environment, 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 mmc2.

  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.
init	Tells kernel where the init file is located.	init=/init	All cases. "init" in Android is located in "/" instead of in "/sbin".

Table continues on the next page...

Kernel parameter	Description	Typical value	Used when
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
vmalloc	vmalloc virtual range size for kernel.	vmalloc=400M	vmalloc=<size>
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.
cma	CMA memory size for GPU/VPU physical memory allocation.	cma=384M	It is 256 MB by default.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AUG
Rev. KK4.4.3_2.0.0-ga
02/2015