AXIOMTEK

Q7M120-120-EVK

(Q7M120 and Q7B120)

BSP

Android User's Manual

Disclaimers

This manual has been carefully checked and believed to contain accurate information. Axiomtek Co., Ltd. assumes no responsibility for any infringements of patents or any third party's rights, and any liability arising from such use.

Axiomtek does not warrant or assume any legal liability or responsibility for the accuracy, completeness or usefulness of any information in this document. Axiomtek does not make any commitment to update the information in this manual.

Axiomtek reserves the right to change or revise this document and/or product at any time without notice.

No part of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Axiomtek Co., Ltd.

CAUTION

If you replace wrong batteries, it causes the danger of explosion. It is recommended by the manufacturer that you follow the manufacturer's instructions to only replace the same or equivalent type of battery, and dispose of used ones.

April 2014, Version A1

Printed in Taiwan

ESD Precautions

Computer boards have integrated circuits sensitive to static electricity. To prevent chipsets from electrostatic discharge damage, please take care of the following jobs with precautions:

- Do not remove boards or integrated circuits from their anti-static packaging until you are ready to install them.

- Before holding the board or integrated circuit, touch an unpainted portion of the system unit chassis for a few seconds. It discharges static electricity from your body.

- Wear a wrist-grounding strap, available from most electronic component stores, when handling boards and components.

Trademarks Acknowledgments

Table of Contents

Chapter 1
Preparation

1.1 Setting Up Your Computer

To build the Android source files, you will need to use a Linux computer.
You also need to use the 12.04 64 bit version of Ubuntu which are the most build OS for the Android JB4.3.
After installing the Linux computer, you need to check whether you have all the necessary packages installed for an Android. 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
$ sudo add-apt-repository ppa:git-core/ppa
$ sudo apt-get update
$ sudo apt-get install git-core curl
```

1.2 Unpacking 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_jb4.3_1.1.0-ga_source.tar.gz file)

```
$ tar xzvf android_jb4.3_1.1.0-ga_source.tar.gz
$ cd android_jb4.3_1.1.0-ga_source/code
$ tar xzvf jb4.3_1.1.0-ga.tar.gz
```

This page is intentionally left blank.

Chapter 2
Building Android for Q7M120-120-EVK

2.1 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-4.3_r2.1
$ ~/bin/repo sync # this command loads most needed repos. Therefore, it can take several hours to load.
```

Get jb4.3_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 jb4.3_1.1.0-ga
$ cd myandroid/bootable
$ cd bootloader
$ git clone git://git.freescale.com/imx/uboot-imx.git uboot-imx
$ cd uboot-imx
$ git checkout jb4.3_1.1.0-ga
```

2.2 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 have unzipped i.MX Android release package to
/opt/android_jb4.3_1.1.0-ga_source.
$ cd ~/myandroid
$ source /opt/android_jb4.3_1.1.0-ga_source/code/jb4.3_1.1.0-ga/and_patch.sh
$ help
Now you should see that the "c_patch" function is available
$ c_patch /opt/android_jb4.3_1.1.0-ga_source/code/jb4.3_1.1.0-ga imx_jb4.3_1.1.0-ga
Here "/opt/android_jb4.3_1.1.0-ga_source/code/jb4.3_1.1.0-ga" is the location of the patches (i.e. directory created when you unzip release package) "imx_jb4.3_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_jb4.3_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
****
$ source
/opt/android_jb4.3_1.1.0-ga_source/code/Q7M120_patches/patch-q7m120-android-4.3-1-1-0.sh

2.3 Building 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 by using the steps below:

```
$ cd ~/myandroid
$ source build/envsetup.sh
$ lunch sabresd_6dq-user
$ make
```

"sabresd_6dq" is the freescale product name (see ~/myandroid/device/fsl/product) After build, check build_*_android.log to make sure no build error.
#Build Android images for Q7M120 boards

```
$ lunch sabresd_6dq-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 Q7M120 boards, we use the same build configuration. The board 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".
- recovery.img is an EXT4 image generated from "recovery/". It can be programmed to "RECOVERY" partition on SD/eMMC card with "dd".

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

u-boot-6q.bin is an U-Boot image with padding for Q7M120.

- u-boot-6dl.bin is an U-Boot image with padding for Q7M120.
- To build the U-Boot image separately, see Building U-Boot Images.
- To build the kernel uImage separately, see Building Kernel Image.
- To build boot.img, see Building boot.img.

2.3.1 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 Q7M120 board
$ make PRODUCT-sabresd_6dq-user 2>&1 | tee build_sabresd_6dq_android.log
Or you can use the following commands:
# Build images for Q7M120 board
$ source build/envsetup.sh
$ lunch sabresd_6dq-user
$ make
For more Android building information, see source.android.com/source/building.html.

2.3.2 Building 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 Q7M120.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 configure files in device/fsl.git by the following patch, which make sure partitions are mounted for SD:

```
diff git a/sabresd_6dq/fstab.freescale b/sabresd_6dq/fstab.freescale
index c2b737a..d2e59a3 100644
--- a/sabresd_6dq/fstab.freescale
+++ b/sabresd_6dq/fstab.freescale
@@ 3,12 +3,12 @@
# The filesystem that contains the filesystem checker binary (typically /system) can not
# specify MF_CHECK, and must come before any filesystems that do specify MF_CHECK
-/devices/platform/sdhciesdhcimx.2/mmc_host/mmc1 /mnt/extsd vfat defaults voldmanaged=sdcard:auto
+/devices/platform/sdhciesdhcimx.1/mmc_host/mmc2 /mnt/extsd vfat defaults voldmanaged=sdcard:auto
/devices/platform/fslehci/mnt/udisk vfat defaults voldmanaged=sdcard:auto
-/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,encryptable=footer
-/dev/block/mmcblk0p6 /cache ext4 nosuid,nodev,nomblk_io_submit
wait
-/dev/block/mmcblk0p7 /device ext4 ro,nosuid,nodev
wait
-/dev/block/mmcblk0p1 /boot emmc defaults defaults
-/dev/block/mmcblk0p2 /recovery emmc defaults defaults
-/dev/block/mmcblk0p8 /misc emmc defaults defaults
+/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,encryptable=footer
+/dev/block/mmcblk1p6 /cache ext4 nosuid,nodev,nomblk_io_submit
wait
+/dev/block/mmcblk1p7 /device ext4 ro,nosuid,nodev
wait
+/dev/block/mmcblk1p1 /boot emmc defaults defaults
+/dev/block/mmcblk1p2 /recovery emmc defaults defaults
+/dev/block/mmcblk1p8 /misc emmc defaults defaults
```

2. Follow the 3.3.1 to build the images.

2.4 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-
```

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

For i.MX 6Quad SABRE-SD:

```
$ make mx6q_sabresd_android_config
$ make
```

For i.MX 6DualLite SABRE-SD:

```
$ make mx6dl_sabresd_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.

2.5 Building 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-
# Generate ".config" according to default config file under arch/arm/configs.
# to build the kernel image for Q7M120
$ make imx6_android_defconfig
$ make uImage
```

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

2.6 Building boot.img

boot.img and booti are default booting commands. As outlined in Running 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 Q7M120 board
$ cd ~/myandroid
$ source build/envsetup.sh
$ lunch sabresd_6dq-user
$ make bootimage
```

This page is intentionally left blank.

Chapter 3
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)

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.

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.

3.1 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.

3.1.1 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
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.	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.

3.1.2 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.

3.1.3 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.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
```

3.2 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.

3.2.1 Setting 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_jb4.3_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
```

This page is intentionally left blank.

Chapter 4
Booting

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

4.1 Booting from MMC/SD

This section describes booting from MMC/SD on the Q7M120 board.

4.1.1 Booting 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)
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 cardChange 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 mmc2 [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]
```

4.2 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.

4.2.1 i.MX 6Quad and i.MX 6Dual/Lite SABRE-SD Platform

```
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.

4.3 Boot-Up Configurations

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

4.3.1 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, youneed 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 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.