Freescale Semiconductor

Document Number:AUG
Rev. 13.4.1, 12/2012

Android User Guide

Contents

1 Overview

This document explains how to use the Android release package.

It describes how to setup the environment, how to apply i.MX Android patches, and how to build the Android system. It also describes how to download built out images into storage device and setup the correct hardware/software boot configurations to boot up the system. Meanwhile, it provides the information on how to get Android source from Google, and what the device storage usage and boot option is. Please see http://source.android.com/source/building.html.

2 PC Setup

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

You also need to use the 11.04 64bit version of Ubuntu which is the most tested OS for the Android 4.0.4 build.

After installing the Linux PC, you need to check whether you have all the necessary packages installed for an Android build. Refer to "Setting up your machine" on the Android web site http://source.android.com/source/initializing.html.

3 Unpack i.MX Android Release Package

After you setup a Linux PC, unpack the FSL i.MX Android release package using the following commands:

```
$ cd /opt (or any other directory where you placed the imx-android-r....tar.gz file)
$ tar xzvf imx-android-13.4.1.tar.gz
$ cd imx-android-13.4.1/code
$ tar xzvf 13.4.1.tar.gz
$ tar xzvf 13.4.1-delta.tar.gz
```

4 Run Android with Prebuilt Image

To test Android before building any code, use the prebuilt images under image/ and go to "Download Images and Bootup".

The prebuilt image is under each platform directory that is uncompressed from the release package:

- U-Boot image
    - bootloader
        - sabresd_6dq/u-boot-6q.bin (with padding) supporting i.MX 6Dual/6Quad
        - sabresd_6dq/u-boot-6dl.bin (with padding) supporting i.MX 6DualLite
- Boot image
    - sabresd_6dq/eMMC/boot.img: boot.img is an Android image that stores zImage and ramdisk together. It can also store other information such as the kernel boot command line, machine name, e.g. This information can be configured in android.mk. It can avoid touching boot loader code to change any default boot arguments.

      Notes:

      If you use boot.img, you don't need uImage and uramdisk.img
- Android system images for eMMC card:
    - System root: sabresd_6dq/eMMC/system.img
    - Data: sabresd_6dq/eMMC/userdata.img. Not provided since it's empty.
    - Boot image:sabresd_6dq/eMMC/boot.img
    - Recovery image: sabresd_6dq/eMMC/recovery.img
- Android system images for SD card:
    - System root: sabresd_6dq/SD/system.img
    - Data: sabresd_6dq/SD/userdata.img. Not provided since it's empty.
    - Boot image:sabresd_6dq/SD/boot.img
    - Recovery image: sabresd_6dq/SD/recovery.img
- Android system images for TFTP/NFS:
    - sabresd_6dq/NFS/android_fs.tar.gz
- Kernel image for TFTP/NFS
    - sabresd_6dq/NFS/uImage

NOTE

The SD card images only bootup during the sanity test, since the only differences between eMMC and SD images are the boot.img.

5 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/imx-android-13.4.1/.
$ 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-4.0.4_r1.1
$ cp /opt/imx-android-13.4.1/code/13.4.1/default.xml .repo/manifests/default.xml
(It's used to avoid loading unnecessary gits from Google repo. Meanwhile it can load some
gits from Google repo
which are not included in default manifest)
$ ./repo sync # this command loads most needed repos, it can take a while

after sync code successfully, do the following:
$ cd myandroid/external
$ git clone git://android.git.linaro.org/platform/external/alsa-lib.git
$ cd myandroid/external
$ git clone git://android.git.linaro.org/platform/external/alsa-utils.git
$ cd myandroid/hardware
$ git clone git://android.git.linaro.org/platform/hardware/alsa_sound.git
```

Get R13.4-GA kernel source code from freescale's opensource git:

```
$ cd myandroid
$ git clone git://git.freescale.com/imx/linux-2.6-imx.git kernel_imx # the kernel repo is
heavy, so this can take a while
$ cd kernel_imx
$ git checkout imx-android-13.4.1
```

NOTE: If you're behind proxy, please 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's opensource git:

```
$ cd myandroid/bootable
$ mkdir bootloader
$ cd bootloader
$ git clone git://git.freescale.com/imx/uboot-imx.git uboot-imx
$ cd uboot-imx
$ git checkout imx-android-13.4.1
```

5.1 Patch Code for i.MX

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

```
Assume you had unzipped i.MX Android release package to /opt/imx-android-13.4.1/.
$ cd ~/myandroid
$ source /opt/imx-android-13.4.1/code/13.4.1/and_patch.sh
$ help
Now you should see that the "c_patch" function is available
$ c_patch /opt/imx-android-13.4.1/code/13.4.1 imx_13.4.1
Here "/opt/imx-android-13.4.1/code/13.4.1" is the location of the patches (i.e. directory
created when you unzip release package)
"imx_13.4.1" 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_13.4.1".
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 PC, install them in advance.

If your software has upgraded to R13.4-GA and you would only like to apply the additional patches between R13.4-GA and 13.4.1, please use the following steps:

```
Take framwork/base.git as an example. Assume you had unzipped i.MX Android release package
to /opt/imx-android-13.4.1/ .
$ cd /opt/imx-android-13.4.1/code/13.4.1-delta/platform/frameworks/base.git
$ ls *.patch > series
$ cd ~/myandroid/frameworks/base
$ git quiltimport --patches   /opt/imx-android-13.4.1/code/13.4.1-delta/platform/frameworks/
base.git
If the error is reported when applying some patches, please use "git apply" to patch one by
one.
The same way can be applied to the other gits.
```

NOTE

device/fsl-proprietary.git includes all prebuild binary files. Therefore, it is not the delta patch based on R13.4-GA. Please clear the contents in this folder and apply the new patch to regenerate. Please pay close attention to platform\hardware\imx.git in this release. The name for this file in R13.4-GA is platform\hardware\mx5x.git. Pay attention, 13.4.1 delta patches already include R13.4-ga add-on patches. So please drop R13.4-ga add-on patches then apply 13.4.1 delta patches. Otherwis, you may meet conflict caused by duplicated patch

5.2 Build U-Boot Images

```
$ cd ~/myandroid/bootable/bootloader/uboot-imx
$ export ARCH=arm
$ export CROSS_COMPILE=~/myandroid/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/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
```

"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 which 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.

5.3 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's 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 2 environment variables set
If the two variables have not set, please set the as:
$ export ARCH=arm
$ export CROSS_COMPILE=~/myandroid/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/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.

5.4 Build boot.img

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:

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

5.5 Build Android Image

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

```
$ 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.
```

For BUILD_ID & BUILD_NUMBER, please add a buildspec.mk in your ~/myandroid directory. For details, please check FAQs.

For i.MX 6Quad and i.MX 6DualLite SABRE SD board, 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/ 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, please refer to the Build U-Boot Images.

To build the kernel uImage separately, please refer to the Build Kernel Image.

To build boot.img, please refer to the Build boot.img.

5.5.1 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, please add the package MODULE_NAME or PACKAGE_NAME to this list.

Change "eng" to "user" to generate user mode image:

eg:

```
$ make PRODUCT-sabresd_6dq-user 2>&1 | tee build_sabresd_6dq_android.log
```

Or you can use the following commands:

```
$ source build/envsetup.sh
$ lunch sabresd_6dq-user
$ make
$ make dist # you can generate ota package with this command.
```

For more Android building information please refer to: http://source.android.com/source/building.html

5.5.2 Build Android image for SD card

The default configuration in the source code package takes internal eMMC as the boot storage.

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

- Change the init.rc configure files in device/fsl.git by using the following patch to make sure that the partitions are mounted for SD:

```
diff --git a/imx6/sabresd/init.rc b/imx6/sabresd/init.rc
index 3ba950d..b980b54 100755
--- a/imx6/sabresd/init.rc
+++ b/imx6/sabresd/init.rc
@@ -101,8 +101,8 @@ service iprenew_wlan0 /system/bin/dhcpcd -n
on fs
# mount ext4 partitions
-mount ext4 /dev/block/mmcblk0p5 /system
-mount ext4 /dev/block/mmcblk0p5 /system ro remount
-mount ext4 /dev/block/mmcblk0p7 /data nosuid nodev nodiratime noatime noauto_da_alloc
-mount ext4 /dev/block/mmcblk0p6 /cache nosuid nodev
-mount ext4 /dev/block/mmcblk0p8 /vendor nosuid nodev
+mount ext4 /dev/block/mmcblk1p5 /system
+mount ext4 /dev/block/mmcblk1p5 /system ro remount
+mount ext4 /dev/block/mmcblk1p7 /data nosuid nodev nodiratime noatime noauto_da_alloc
+mount ext4 /dev/block/mmcblk1p6 /cache nosuid nodev
+mount ext4 /dev/block/mmcblk1p8 /vendor nosuid nodev
```

- Change the vold configure 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/imx6/sabresd/vold.fstab b/imx6/sabresd/vold.fstab
index 8a2546b..0f549fc 100644
--- a/imx6/sabresd/vold.fstab
+++ b/imx6/sabresd/vold.fstab
@@ -24,9 +24,9 @@ dev_mount udisk /mnt/udisk auto /devices/platform/fsl-ehci.0/usb1
#used for usb otg host
#dev_mount udisk /mnt/sdcard/udisk auto /devices/platform/fsl-ehci.0/usb
#mount SDHC4 SD card /mnt/sdcard as primary storage forMX6Q 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
+dev_mount sdcard /mnt/sdcard 4 /devices/platform/sdhci-esdhc-imx.2/mmc_host/mmc1
```

- Follow the section "User Build mode" to build the Images.

6 Download Images

i.MX Android can be booted up in two 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 or TF), 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 6DualLite SABRE-SD boards:

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

6.1 System on MMC/SD

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.

6.1.1 Storage Partitions

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	0	1MB	N/A	bootloader
Primary 1	Boot	8M	8MB	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	512MB	EXT4. Mount as / system	Android system files under / system/ dir.
Logic 6 (Extended 3)	CACHE	Follow SYSTEM.	256MB	EXT4. Mount as / cache.	Android cache for image store of OTA.
Logic 7 (Extended 3)	DATA	Follow CACHE.	> 1024MB	EXT4. Mount at / data.	Application data storage for the system application.
Logic 8 (Extended 3)	Vendor	follow DATA	8MB	Ext4. Mount at /vender.	For Store MAC address files.
Logic 9 (Extended 3)	Misc	Follow DATA	4M	N/A	For recovery store bootloader message, reserve.
Primary 4	MEDIA	Follow Misc	Total - Other images	VFAT	For internal media partition, in /mnt/sdcard/ dir.

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

```
$ cd ~/myandroid/
$ sudo chmod +x ./prebuilt/linux-x86/fsl/fsl-sdcard-partition.sh
$ sudo ./prebuilt/linux-x86/fsl/fsl-sdcard-partition.sh /dev/sdX
```

NOTE:

- The minimum size of SD card is 2G bytes.
- /dev/sdxN, the x is the disk index from 'a' to 'z'. That may be different on each Linux PC.
- 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 6DualLite SABRE-SD. If you want to boot it from the external SD card, please refer to FAQs for more information.

6.1.2 Download Images with MFG Tool

MFGTool can be used to download all images into a target device.

It's a quick and easy tool for downloading images. Please refer to Android Quick Start User Guide for detailed description of MFGTool.

6.1.3 Download Images with dd Utility

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

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

All partitions can be recognized by the Linux PC. To download all images into the card, please 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
```

NOTE

X is supposed to be changed depending on the letter assigned to the SD by Linux.

6.2 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 PC which has NFS and TFTP server, with their root directory setup correctly, e.g /opt/tftproot for TFTP root, and /opt/nfsroot for NFS root.

6.2.1 Setup the TFTP and NFS Root

After you setup 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, make sure you pick up the correct uImage (see "Prebuilt image for using U-Boot").
- If you are building your own image, make sure you have generated an 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/
```

Setup the Android file system: (take i.MX 6Dual/6Quad SABRE-SD or i.MX 6DualLite SABRE-SDas an example)

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

  ```
  $ cd /opt/imx-android-r13.4-ga/image/sabresd_6dq/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 the system and cache folder under /opt/nfsroot/, you need to comment out the mount system and cache lines in /opt/nfsroot/init.rc. Since the framework will clear eth0'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:

```
# mount rootfs rootfs / ro remount
setprop ethernet.clear.ip no
# on fs
# mount ext4 partitions
#    mount ext4 /dev/block/mmcblk0p5 /system
#    mount ext4 /dev/block/mmcblk0p5 /system ro remount
#    mount ext4 /dev/block/mmcblk0p7 /data nosuid nodev nodiratime noatime noauto_da_alloc
#    mount ext4 /dev/block/mmcblk0p6 /cache nosuid nodev
#    mount ext4 /dev/block/mmcblk0p8 /vendor nosuid nodev
```

7 Bootup from MMC/SD

i.MX 6Dual/6Quad SABRE-SD board U-Boot default setting is to boot from eMMC.

Boot Switch:

download Mode(MFGTool mode)	(SW6) 00001100 (from 1-8 bit)
eMMC(MMC3) boot	(SW6) 11100110 (from 1-8 bit)
MMC4 (SD2) boot	(SW6) 10000010 (from 1-8 bit)
MMC2 (SD3) boot	(SW6) 01000010 (from 1-8 bit)

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

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

To clear the bootargs env.

```
U-Boot > setenv bootcmd booti mmc3
U-Boot > setenv bootargs console=ttymxc0,115200 androidboot.console=ttymxc0 vmalloc=400M
init=/init video=mxcfb0:dev=ldb,LDB-XGA,if=RGB666,bpp=16 video=mxcfb1:off video=mxcfb2:off
fbmem=10M fb0base=0x27b00000 #[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, please 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 (SW6) 01000010 (from 1-8 bit). To clear the bootargs env and setup the booting from SD card in SD slot 3, you can use the following command:

```
U-Boot > setenv bootcmd booti mmc2         [Load the boot.img from SD card]
U-Boot > setenv bootargs console=ttymxc0,115200 init=/init video=mxcfb0 video=mxcfb1:off
video=mxcfb2:off fbmem=10M fb0base=0x27b00000 vmalloc=400M androidboot.console=ttymxc0
[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, please 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]
```

7.1 Boot from TFTP and NFS

Setup the U-Boot environment for loading kernel from TFTP and mounting NFS as root filesystem 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 rw video=mxcfb0 ip=dhcp fbmem=10M
fb0base=0x27b00000 nfsroot=${serverip}:/${nfsroot} vmalloc=400M androidboot.console=ttymxc0
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.

7.2 Boot Up Configurations

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

7.2.1 U-Boot Environment

- 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:

  > setenv bootargs

  To clear the bootargs env.

  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 6DualLite SABRE-SDboard. 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 configure file, the address is virtual. Otherwise, it's 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.

7.2.2 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	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".

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 configure	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 the i.MX 6 WAIT mode.	enable_wait_mode=off	enable_wait_mode=<on/off>
arm_freq	Lets CPU work at special frequency(MHz).	arm_freq=800	Just for those boards which do not rework for 1GHz.

Table continues on the next page...

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 set this for the kernel.
fec_mac	Setup of 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, please assign this parameter to the kernel.
fb0base	Tell kernel the framebuffer base address which bootloader uses for splashscreen. Kernel will go on to use the address for framebuffer.	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 ‘fbmem’ for fb0 to reserve fb memory as well.
gpumem	Setup the reserved memory size for GPU driver	gpumem=196M	It’s 196M by default on the SabreSD platform. This kernel parameter can only be used when limiting GPU/VPU usage.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AUG
Rev. 13.4.1
12/2012