NXP Semiconductors
User's Guide

Document Number: SCMIMX6AUG
M6.0.1_2.1.0, May 15th, 2017

SCM i.MX 6 Series Android User's Guide

Contents

Chapter 1
Overview

This document provides the technical information related to the SCM i.MX6DQ QWKS and i.MX6SX EVB devices:

- Instructions for building from sources or using pre-built images
- Copying the images to a boot media
- Hardware/software configurations for programming the boot media and running the images

This document describes how to configure a Linux build machine and provides the steps to download, patch, and build the software components that create the Android system image when working with the sources.

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

Chapter 2
Preparation

2.1 Setup your computer

To build the Android source files, you will need to use a computer running Linux OS. The 14.04 64bit version and openjdk-7-jdk of Ubuntu are the most tested environment for the Android Marshmallow 6.0 build.

After installing 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 .

In addition to the packages requested on the Android website, the following packages are also needed:

```
$ sudo apt-get install uuid uuid-dev
$ sudo apt-get install zlib1g-dev liblz-dev
$ sudo apt-get install liblzo2-2 liblzo2-dev
$ sudo apt-get install lzop
$ sudo apt-get install git-core curl
$ sudo apt-get install u-boot-tools
$ sudo apt-get install mtd-utils
$ sudo apt-get install android-tools-fsutils
$ sudo apt-get install openjdk-7-jdk
```

NOTE

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

2.2 Unpack the SCM i.MX Android Release Package

Note that SCM packages is deployed as a separate tarball, identifiable by the **mx6scm** in-text .

After you setup a Linux PC, unpack the SCM Android Release Package by using the following commands.

```
$ cd /opt (or any other directory you like)
$ tar xzvf android_M6.0.1_2.1.0_mx6scm_source.tar.gz
```

Chapter 3
Build Android for SCM i.MX

3.1 Get Android Source Code (Android/Kernel/uboot)

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

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

```
Assume you had unzipped i.MX Android release package to /opt/
android_M6.0.1_2.1.0_mx6scm_source/.

$ cd ~
$ mkdir myandroid
$ mkdir bin
$ cd myandroid
$ curl https://storage.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-6.0.1_r22
$ ~/bin/repo sync
```

Get M6.0.1_2.1.0 kernel source code from freescale's opensource git:

```
$ cd myandroid
$ git clone git://git.freescale.com/imx/linux-2.6-imx.git kernel_imx
$ cd kernel_imx
$ git checkout m6.0.1_2.1.0-ga
```

NOTE

If you're behind proxy, please use socksify to set socks proxy for git protocol. If you use uboot as your bootloader, then you can clone the uboot git repository from freescale's opensource git:

```
$ cd myandroid/bootable
$ cd bootloader
$ git clone git://git.freescale.com/imx/uboot-imx.git uboot-imx
$ cd uboot-imx
$ git checkout m6.0.1_2.1.0-ga
```

3.2 Patch NXP standard code package

Apply all i.MX Android patches (**including SCM patches**) using the following steps:

```
Assume you had unzipped i.MX Android release package to /opt/
android_M6.0.1_2.1.0_mx6scm_source/.

$ cd ~/myandroid
$ source /opt/android_M6.0.1_2.1.0_mx6scm_source/code/M6.0.1_2.1.0/and_patch.sh
$ help

Now you should see "c_patch" function is available for you
$ c_patch /opt/android_M6.0.1_2.1.0_mx6scm_source/code/M6.0.1_2.1.0 imx_M6.0.1_2.1.0-ga

Here "/opt/android_M6.0.1_2.1.0_mx6scm_source/code/M6.0.1_2.1.0" is the location of the
patches (i.e. directory created when you unzip release package)
"imx_M6.0.1_2.1.0" 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_M6.0.1_2.1.0".
If everything is OK, "c_patch" will generate the following output to indicate successful
patch:
```

```
************************************************************
Success: Now you can build 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.

3.3 Build Android Image

Building the Android image is performed when the source code has been downloaded (Section 3.1) and patched (Section 3.2). Two commands are executed to build: one is lunch <buildName-buildType> to set up the build configuration, and the other is make to start the build process. The build configuration command lunch can be issued with an argument Build Name – Build Type string, such as lunch qwks_6scm-user, or can be issued without the argument presenting a menu of selection.

The Build Name is the Android device name found directory ~/myandroid/device/fsl/. The following table lists the Freescale build names.

Build name	Description
qwks_6scm	SCM i.MX6DQ Quick Start Board (QWKS)
evb_6sxscm	SCM i.MX6SX Evaluation Board (EVB)

The build type is used to specify what debug options are provided in the final image. The following table lists the build types.

Build type	Description
user	Production ready image, no debug
userdebug	Provides image with root access and debug, similar to "user"
eng	Development image with debug tools

Android build steps are as follows:

1. Change to the top level build directory.

$ cd ~/myandroid

2. Set up the environment for building. This only configures the current terminal.

$ source build/envsetup.sh

3. Execute the Android lunch command. In this example, the setup is for the production image of SCM i.MX6DQ QWKS Board/Platform device.

$ lunch qwks_6scm -user

4. Execute the make command to generate the image.

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

When the make command is complete, the build-log.txt file contains the execution output. Please check for any errors.

For BUILD_ID & BUILD_NUMBER changing, please update build_id.mk in your ~/myandroid directory, detail step, please check Android Frequently Asked Questions document.

For both SCM i.MX6DQ QWKS and SCM i.MX6SX EVB platforms, we use the same build configuration. They share the same system images, except the boot, recovery and uboot image. The following outputs are generated in the directory myandroid/out/target/product/ qwks_6scm:

- **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-imx6qdqscm-1gb-qwks-rev3.img**: a composite image for SCM i.MX6DQ 1GB QWKS REVC (rev3) which includes the kernel zImage, ramdisk, board's device tree binary, and boot parameters.
  - **boot-imx6qdqscm-1gb-qwks-rev2.img**: a composite image for SCM i.MX6DQ 1GB QWKS REVB (rev2) which includes the kernel zImage, ramdisk, board's device tree binary, and boot parameters.
  - **ramdisk.img**: Ramdisk image generated from "root/". Not directly used.
  - **system.img:** EXT4 image generated from "system/". Can be programmed to "SYSTEM" partition on SD/eMMC card with "dd"
  - **userdata.img:** EXT4 image generated from "data/".
  - **recovery-imx6dqscm-1gb-qwks-rev3.img:** EXT4 image for SCM i.MX6DQ 1GB QWKS REVC (rev3), which is generated from "recovery/". Can be programmed to "RECOVERY" partition on SD/eMMC card with "dd"
  - **recovery-imx6dqscm-1gb-qwks-rev2.img:** EXT4 image for SCM i.MX6DQ 1GB QWKS REVB (rev2), which is generated from "recovery/". Can be programmed to "RECOVERY" partition on SD/eMMC card with "dd"
  - **u-boot-imx6dqscm-1gb-qwks-rev3.imx:** uboot image with no padding for SCM i.MX6DQ 1GB QWKS REVC (rev3).
  - **u-boot-imx6dqscm-1gb-qwks-rev2.imx:** uboot image with no padding for SCM i.MX6DQ 1GB QWKS REVB (rev2).

NOTE

• To build the uboot image separately, please refer to the section 3.5 • To build the kernel uImage separately, please refer to the section 3.6 • To build boot.img, please refer to the section 3.7

3.3.1 Configuration examples of building NXP devices

The following table shows examples of using the lunch command to set up different NXP devices. After the desired NXP device is set up, the make command is used to start the build.

Build name	Description
SCM i.MX6DQ QWKS Board	$ lunch qwks_6scm -user
SCM i.MX6SX EVB Board	$ lunch evb_6sxscm-user

3.3.2 User Build mode

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

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

- Limited Android System image access for security reasons.
- Lack of debugging tools.
- Installation modules tagged with user.
- APK's and tools according to product definition files, which are found in PRODUCT_PACKAGES in the sources folder ~/myandroid/device/fsl/imx6/imx6.mk. To add customized packages, add the package MODULE_NAME or PACKAGE_NAME to this list.
- The properties are set as: ro.secure=1 and ro.debuggable=0.
- adb is disabled by default.

  For building the SCM Android image use the next method:

  Set environment first, use lunch command to config argument(see table below), then make.

  An example for the QWKS board for SCM-i.MX 6D/Q is:

```
$ cd ~/myandroid
$ source build/envsetup.sh
$ lunch qwks_6scm-user
$ make
```

Table 1. Android system image production build method 2

Description	Lunch configuration
SCM i.MX6DQ QWKS	qwks_6scm -user
SCM i.MX6SX EVB	evb_6sxscm -user

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

```
$ make otapackage
```

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

3.3.3 Build Android image for SD2 card on QWKS Board

The default configuration in the source code package takes SD3 (micro SD slot) as the boot storage. It can be changed to make the SD in Slot 2 as the boot storage as below steps:

1. remove /out/target/product/ qwks_6scm /root directory and boot*.img
2. remove /out/target/product/ qwks_6scm /recovery directory and recovery*.img
3. build the boot.img and recovery.img as below commands

```
make bootimage BUILD_TARGET_DEVICE=sd2
make recoveryimage BUILD_TARGET_DEVICE=sd2
```

3.4 Build U-Boot Images

```
$ cd ~/myandroid/bootable/bootloader/uboot-imx
$ export ARCH=arm
$ export CROSS_COMPILE=~/myandroid/prebuilts/gcc/linux-x86/arm/arm-linux-
androideabi-4.9/bin/arm-linux-androideabi-
Command to build are:
$ make distclean

For SCM i.MX6DQ 1GB QWKS  REVC (rev3):
# to build uboot.imx which is used in android
```

```
$ make mx6dqscm_1gb_interleaving_qwks_rev3_android_defconfig
# to build uboot.img which is used in MFGTOOL
$ make mx6dqscm_1gb_fix_qwks_rev3_defconfig

For SCM i.MX6DQ 1GB QWKS  REVB (rev2):
# to build uboot.imx which is used in android
$ make mx6dqscm_1gb_interleaving_qwks_rev2_android_defconfig
# to build uboot.img which is used in MFGTOOL
$ make mx6dqscm_1gb_fix_qwks_rev2_defconfig

For SCM i.MX6SX 1GB EVB:
# to build uboot.imx which is used in android
$ make mx6sxscm_1gb_evb_android_defconfig
# to build uboot.img which is used in MFGTOOL
$ make mx6sxscm_1gb_evb_defconfig


$ make

"u-boot.imx" is generated if you have a successful build.
uboot.img in android should put in mfgtools\Profiles\Linux\OS Firmware\files\android\board
\.
uboot.img in MFGTOOL should put in mfgtools\Profiles\Linux\OS Firmware\firmware\.
```

NOTE

Any image which should be loaded by uboot must have an unique image head, for example, some data must be added at the head of the loaded image to tell uboot 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 uboot, you need a tool to add this information and generate a new image which can be recognized by uboot. Fortunately, this tool is delivered together with uboot. After you set up uboot using the steps above, you can find the tool (mkimage) under tools/. The process of how to use mkimage for generating the image (for example kernel image, ramfs image), which is to be loaded by uboot, is outlined in the subsequent sections of this document.

3.5 Build Kernel Image

Kernel image will be automatically built out when building the android root file system. There are below ways to help build out the kernel image independent default android build command.

```
$ 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/prebuilts/gcc/linux-x86/arm/arm-linux-
androideabi-4.9/bin/arm-linux-androideabi-

# Generate ".config" according to default config file under arch/arm/configs.
# to build the kernel zImage for both SCM i.MX 6DQ and 6SX
$ make imx_v7_android_defconfig
$ make KCFLAGS=-mno-android


# to build the kernel uImage for SCM i.MX 6DQ QWKS board
$ make uImage LOADADDR=0x10008000 KCFLAGS=-mno-android

# to build the kernel uImage for SCM i.MX6SX EVB board
$ make uImage LOADADDR=0x80008000 KCFLAGS=-mno-android

# to build the zImage which is used in MFGTOOL
# zImage is under mfgtools\Profiles\Linux\OS Firmware\firmware\
```

```
$ make imx_v7_mfg_defconfig
$ make KCFLAGS=-mno-android -j4
```

With a successful build in either of the above case, the generated kernel images are

~/myandroid/kernel_imx/arch/arm/boot/zImage and

~/myandroid/kernel_imx/arch/arm/boot/uImage.

3.6 Build boot.img

As outlined in Section 2.3, we use boot.img and boota as default commands to boot rather than the uramdisk and zImage we used before.

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

```
# Boot image for SCM i.MX 6DQ QWKS board
$ cd ~/myandroid
$ source build/envsetup.sh
$ lunch qwks_6scm-user
$ make bootimage

# Boot image for SCM i.MX6SX EVB board
$ source build/envsetup.sh
$ lunch evb_6sxscm-user
$ make bootimage
```

Chapter 4
Run Android with Prebuilt Image

To test Android before building any code, use the prebuilt images into the release package and go to "Download Images" and "Boot":

Image Package	Descriptions
android_M6.0.1_2.1.0_full_image_mx6scm.tar.gz --- qwks_rev3	Prebuilt-Image for SCM i.MX6DQ 1GB QWKS REVC (rev3) board
android_M6.0.1_2.1.0_full_image_mx6scm.tar.gz --- qwks_rev2	Prebuilt-Image for SCM i.MX6DQ 1GB QWKS REVB (rev2) board
android_M6.0.1_2.1.0_full_image_mx6scm.tar.gz --- evb_6sxscm	Prebuilt-Image for SCM i.MX6SX 1GB EVB board.

The following tables list the detailed contents of android_M6.0.1_2.1.0_full_image_mx6scm.tar.gz (qwks_rev3 directory) image package. The table below shows the prebuilt images to support the system boot from SD3 on for SCM i.MX6DQ QWKS REVC (rev3) board.

SCM i.MX6DQ QWKSREVC (rev3) Images	Descriptions
u-boot-imx6dqscm-1gb-qwks-rev3.imx	The bootloader (with padding) for SCM i.MX6DQ QWKS REVC (rev3) board
boot-imx6dqscm-1gb-qwks-rev3.img	Boot Image for SD3
system.img	System Boot Image
recovery-imx6dqscm-1gb-qwks-rev3.img	Recovery Image

The following tables list the detailed contents of android_M6.0.1_2.1.0_full_image_mx6scm.tar.gz (qwks_rev2 directory) image package.

The table below shows the prebuilt images to support the system boot from SD3 on for SCM i.MX6DQ QWKS REVB (rev2) board.

SCM i.MX6DQ QWKSREVB (rev2) Images	Descriptions
u-boot-imx6dqscm-1gb-qwks-rev2.imx	The bootloader (with padding) for SCM i.MX6DQ QWKS REVC (rev2) board
boot-imx6dqscm-1gb-qwks-rev2.img	Boot Image for SD3
system.img	System Boot Image
recovery-imx6dqscm-1gb-qwks-rev2.img	Recovery Image

The following tables list the detailed contents of android_M6.0.1_2.1.0_full_image_mx6scm.tar.gz (evb_6sxscm directory) image package. The table below shows the prebuilt images to support the system boot from SD3 on for SCM i.MX6SX EVB board.

SCM i.MX6SX EVB Images	Descriptions
u-boot-imx6sxscm-1gb-evb.imx	The bootloader (with padding) for SCM SCM i.MX6SX EVB
boot-imx6sxscm-1gb-evb.img	Boot Image for SD3
system.img	System Boot Image
recovery-imx6sxscm-1gb-evb.img	Recovery Image

NOTE

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.

Chapter 5
Programming Images

The images from the NXP prebuilt release package or created from source code contain the U-Boot bootloader, system image, and recovery image. At a minimum, the storage devices (SD) on the NXP development system must be programmed with the U-Boot bootloader. The i.MX 6 series boot process determines what storage device to access based on the switch settings. When the boot loader is loaded and begins execution, the U-Boot environment space is then read to determine how to proceed with the boot process. For U-Boot environment settings, see Section Booting.

The following download methods can be used to write the Android System Image:

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

5.1 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

5.1.1 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	1K	1MB	N/A	bootloader
Primary 1	Boot	8M	16MB	boot.img format, a kernel + ramdisk	boot.img
Primary 2	Recovery	Follow Boot	16MB	boot.img format, a kernel + ramdisk	recovery.img

Table continues on the next page...

Table continued from the previous page...

Logic 5 (Extended 3)	SYSTEM	Follow Recovery	800MB	EXT4. Mount as / system	Android system files under / system/ dir
Logic 6 (Extended 3)	CACHE	Follow SYSTEM	512MB	EXT4. Mount as / cache	Android cache, for image store for OTA
Logic 7 (Extended 3)	Device	Follow CACHE	8MB	Ext4. Mount at /device	For Store MAC address files.
Logic 8 (Extended 3)	Misc	Follow Device	6MB	N/A	For recovery storage bootloader message, reserve.
Logic 9 (Extended 3)	DATAFOOTER	Follow Misc	2MB	N/A	For crypto footer of DATA partition encryption
Primary 4	DATA	Follow Misc	Total - Other images	EXT4. Mount at / data	Application data storage for system application. And for internal media partition, in /mnt/ sdcard/ dir.

To create these partitions, you can use MFGTool, or use format tools in prebuilt directory.

The script below can be used to partition a sdcard and download image to them as shown in the partition table above:

```
$ cd ~/myandroid/
$ sudo ./device/fsl/common/tools/fsl-sdcard-partition.sh -f <soc_name> /dev/sdX
# <soc_name> can be as imx6q or imx6sx.
```

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. • Unmount all the SD card partitions before running the script. • Put related bootloader, boot image, system image, recovery image in your current directory. • This script need simg2img tool to be installed in your PC. simg2img is a tool which convert sparse system image to raw system image on linux host PC. The android-tools-fsutils package includes the simg2img command for Ubuntu Linux.

5.1.2 Download Images with MFG Tool

MFGTool can be used to download all the images into target device. It's a quick and easy tool to download images.

5.1.3 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 section 4.1.1.

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

```
Download the uboot 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 simg2img system.img system_raw.img
# sudo dd if=system_raw.img of=/dev/sdx5; sync
Download the android recovery image:
# sudo dd if=recovery.img of=/dev/sdx2; sync
```

NOTE

simg2img is a tool which convert sparse system image to raw system image on linux host PC. It will be built out as myandroid/out/host/linux-x86/bin/simg2img. The android-tools-fsutils package also includes the simg2img command for Ubuntu Linux.

Chapter 6
Boot

6.1 Boot from SD3

6.1.1 Boot from SD3 on SCM i.MX6DQ QWKS REVC (rev3)/REV B (rev2) Board

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

download Mode (MFGTool mode)	(SW3)/(SW1) 11001001 (from 1-8 bit)
SD3 boot	(SW3)/(SW1) 10101001 (from 1-8 bit)

HDMI is the default display on QWKS. To change to ldb as the main display you can use the following settings on uboot

```
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=256M androidboot.console=ttymxc0 consoleblank=0
androidboot.hardware=freescale cma=384M [Optional]

U-Boot > saveenv        [Save the environments]
```

6.1.2 Boot from SD3 on SCM i.MX6SX EVB Board

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

download Mode(MFGTool mode)	(S1) 10 (from 1-2 bit)
SD3 boot	(S1) 01 (from 1-2 bit) (SW2) 00000000 (from 1-10 bit) (SW3) 00110000 (from 1-4 bit) (SW4) 01000010 (from 1-4 bit)

6.2 Boot Up configurations

This section explains the common u-boot environments we used for NFS, MMC/SD boot above, and also kernel command line we may have changed for different usage scenarios.

6.2.1 U-Boot environment

- ethaddr/fec_addr: MAC address of your board
- serverip: IP address of your TFTP/NFS server
- loadaddr/rd_loadaddr: the kernel/initramfs image load address in memory
- bootfile: the name of image file loaded by "dhcp" command, when you using TFTP to load kernel.
- bootcmd: the first variable to run after uboot boot

- bootargs: the kernel command line, bootloader passed to kernel. Described in 4.3.2 section, bootargs env is a optional for boota. There is a default bootargs stored in boot.img. If the bootargs env isn't been manually set in uboot, the default on in boot.img will be used. If you want to use default env in boot.img, you can use command below to clear the bootargs env.

```
> setenv bootargs
```

- dhcp: get ip address by BOOTP protocol, and load the kernel image ($bootfile env) from TFTP server.
- boota: boota command will parse the boot.img 's header to get the zImage, and ramdisk, also will pass the bootargs as needed (it will only pass bootargs in boot.img when it can't find "bootargs" var in your uboot env). To boot from mmcX, you only need to do the following:

```
> boota 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.

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

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

```
> boota 0xXXXXXXXX
```

• bootm: (only work in NFS case) start to run the kernel, for other case, we use boota command. • splashimage: 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 uboot code root tree for detail. • splashpos: this option sets the splash image to a free position 'x,y' on the screen. x and y should be positive number, which is used as number of pixel from left/top. Note that 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 uboot code root tree for detail. • lvds_num: choose which lvds interface 0 or 1 to show splash image. Note that we only support boot splash on LVDS panel, but not support HDMI or other display device.

6.2.2 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 or console=ttymxc2	QWKS use console=ttymxc0 EVB use console=ttymxc2
init	tell kernel where is the init file	init=/init	All case for Android. "init" in Android is located in "/" instead of in "/sbin"

Table continues on the next page...

Table continued from the previous page...

video	tell kernel/driver which resolution/depth and refresh rate should be used or tell 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=256M	vmalloc=<size>
androidboot.console	The android shell console, should be same as console=	androidboot.console=ttymxc0	To use the default shell's job control, like Ctrl-C to terminate a running process, you need to set this for kernel.
fec_mac	Setup the FEC mac address	fec_mac=00:04:9f:00:ea:d3	On some QWKS/EVB board, the SoC does not have MAC address fused in, so if you want to use FEC, please assign this parameter to kernel.
cma	CMA memory size for GPU/VPU physical memory allocation	cma=384M	It is 256M by default
androidboot.selinux	Argument to disable selinux check and enable serial input when connecting a host computer to the target board's USB UART port. Please refer below link for detail information about selinux. http://source.android.com/devices/tech/security/selinux/	androidboot.selinux=disabled	As since Android Lollipop 5.1 CTS requirement: serial input should be disabled by default. Setting this argument enables console serial input, which will violate the CTS requirement. Setting this argument will also bypass all the selinux rules defined in Android system. Recommend to set this argument for internal developer

Table continues on the next page...

Table continued from the previous page...

androidboot.dm_verity	Argument to disable the verified boot, which to make sure binary in boot partition can only load the certain binary on system partition. https://source.android.com/devices/tech/security/verifiedboot/index.html	androidboot.dm_verity=disabled	Setting this argument will bypass integrity checking on the system partition Recommend to set this argument for internal developer, or the case binary in system partition need to be changed in developing.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support