Document Number:AUG
Rev. 13.5.0_GA, 11/2012

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 use the 10.10 or 11.04 64bit version of Ubuntu which are 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-....tar.gz file)
$ tar xzvf imx-android-13.5.0-ga.tar.gz
$ cd imx-android-13.5.0-ga/code
$ tar xzvf 13.5.0-ga.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 a directory that is uncompressed from the release package:

- U-Boot image
  - bootloader
    - image_imx-android-13.5.0-ga_6slevkeink/u-boot-mx6sl.bin (with padding) supporting i.MX 6SoloLite
- Boot image
  - image_imx-android-13.5.0-ga_6slevkeink/SD/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 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 SD card:
  - System root: image_imx-android-13.5.0-ga_6slevkeink/SD/system.img
  - Data: image_imx-android-13.5.0-ga_6slevkeink/SD/userdata.img. Not provided since it's empty.
  - Boot image: image_imx-android-13.5.0-ga_6slevkeink/SD/boot.img
  - Recovery image: image_imx-android-13.5.0-ga_6slevkeink/SD/recovery.img
- Android system images for TFTP/NFS:
  - image_imx-android-13.5.0-ga_6slevkeink/NFS/android_fs.tar.gz
- Kernel image for TFTP/NFS
  - image_imx-android-13.5.0-ga_6slevkeink/NFS/uImage

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

The Android source code is maintained at 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-13.5.0-ga/.
$ cd ~
$ mkdir myandroid
$ mkdir bin
$ 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-r13.5.0-ga/code/r13.5.0-ga/default.xml .repo/manifests/default.xml
```

```
(To avoid loading unnecessary gits from Google repo, meanwhile load some gits from Google
repo which is 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:
```

NOTE

Git 1.8.0 or newer is required. Otherwise, gc error would occur during the sync process. The corresponding error is "error: Exited sync due to gc errors."

```
$ 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 13.5.0-GA kernel source code from Freescale 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-13.5.0-ga
```

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-13.5.0-ga
```

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-13.5.0-ga/.
$ cd ~/myandroid
$ source /opt/imx-android-13.5.0-ga/code/13.5.0-ga/and_patch.sh
$ help
Now you should see that the "c_patch" function is available
$ c_patch /opt/imx-android-13.5.0-ga/code/13.5.0-ga imx_13.5.0-ga
Here "/opt/imx-android-13.5.0-ga/code/13.5.0-ga" is the location of the patches (i.e.
directory
created when you unzip release package)
"imx_13.5.0-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_13.5.0-ga".
If everything is OK, "c_patch" will generate the following output to indicate successful
patch:

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

NOTE

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

6 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-
$ make distclean
$ make mx6sl_evk_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 execute 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 build 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 loaded by U-Boot, is outlined in the subsequent sections of this document.

7 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 m as follows:

```
$ export ARCH=arm
$ export CROSS_COMPILE=~/myandroid/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin/arm-eabi-
$ make imx6s_android_defconfig
```

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

```
$ make uImage
```

With a successful build the generated kernel image is ~/myandroid/kernel_imx/arch/arm/boot/uImage.

8 Build Android Image

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

```
$ cd ~/myandroid
$ source build/envsetup.sh
$ lunch evk_6sl_eink-user
$ make
"evk_6sl_eink" 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.

The following outputs are generated by default in myandroid/out/target/product/evk_6sl_eink:

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-6s.bin is an U-Boot image with padding for i.MX 6SoloLite EVK.

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.

8.1 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-evk_6sl_eink-user 2>&1 | tee build_evk_6sl_eink_android.log
```

Or you can use the following commands:

```
$ source build/envsetup.sh
$ lunch evk_6sl_eink-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

8.2 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 evk_6sl_eink-user
$ make bootimage
```

9 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 6SoloLite EVK board:

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

9.1 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.

9.1.1 Storage Partitions

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

- [Name] is only meaningful in Android. You can ignore it when creating these partitions.
- [Start Offset] shows where the partition is started, unit in MB.

The SYSTEM partition is used to put the 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.

Table 1. Storage Partitions

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 these partitions, you can use MFGTool as described in the Android Quick Start Guide, or you can use format tools in prebuilt dir.

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

```
$ cd ~/myandroid/
$sudo sh ./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.

9.1.2 Download Images with MFG Tool Version 2.0

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.

For more information about the difference between version 1 and version 2, please refer to MfgTool2 Release Notes and MfgTool2 User Guide. Please note that MFGTool v1.0 is no longer supported.

Please make sure that the key “name” in the [LIST] section of cfg.ini is Android-EVK-SD in order to select Android downloading profile. The corresponding download operation list is defined in the Android-EVK-SD list of ucl2.xml in the Linux Update profile. Also, all images (u-boot.bin, boot.img, system.img, and recovery.img) should be copied to Profiles \MX6SL Linux Update\OS Firmware\files\android folder.

9.1.3 Download Images with dd Utility

The linux utility "dd" on Linux PC can be used to download the images into the MMC/SD card. Before downloading, make sure your MMC/SD 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
```

9.2 System on NFS

Android support runs the system on NFS root file system. We can put the total 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.

9.2.1 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 building your own image, make sure you have generated an uImage (see Build Kernel Image).

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

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

Setup the Android file system:

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

```
$ cd /opt/imx-android-13.5.0-ga/image/evk_6sl_eink/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/evk_6sl_eink/root/* /opt/nfsroot/
$ cp -r out/target/product/evk_6sl_eink/system/* /opt/nfsroot/system/
```

NOTE

Since the NFS uses system and cache folder under /opt/nfsroot/, comment out the mount system and cache lines in /opt/nfsroot/init.freescale.rc, since framework will clear eth0's IP when suspend which will cause resume failure. Therefore, system property ethernet.clear.ip should be set to "no" in /opt/nfsroot/init.freescale.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
```

10 Bootup from MMC/SD

i.MX 6SoloLite EVK board boot from SD.

Table 2. Boot Switch

download Mode (MFGTool mode)	S1: BOOT_MODE[0:1]=10 (from 1-2 bits on S1) 1 means switching to ON
SD boot from SD2 slot	SW3,4,5 01000000 00101100 00000000 (from 1-8 bit) S1: BOOT_MODE[0:1]=01 (from 1-2 bits on S1)

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 mmc1
U-Boot > setenv bootargs console=ttymxc0,115200 init=/init androidboot.console=ttymxc0
max17135:pass=2,vcom=-2030000 video=mxcepdcfb:E060SCM,bpp=16
video=mxc_elcdif_fb:off#[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 i.MX 6SoloLite EVK 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]
```

10.1 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:

EVK board

```
U-Boot > setenv loadaddr 0x80800000
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 rw nfsroot=${serverip}:/$
{nfsroot} androidboot.console=ttymxc0 max17135:pass=2,vcom=-2030000 video

=mxcepdcfb:E060SCM,bpp=16 video=mxc_elcdif_fb:off

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.

10.2 Boot Up Configurations

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

10.2.1 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, your definition 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 parser 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 6SoloLite EVK board. SD is mmc1.

  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 for the NFS) starts running the kernel. For other cases, use booti command.

10.2.2 Kernel Command Line (bootargs)

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

Table 3. Kernel Boot Parameters

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.

Table continues on the next page...

Table 3. Kernel Boot Parameters (continued)

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=video=mxcepdcfb:E060 SCM,bpp=16 video=mxc_elcdif_fb:off (for eink)	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.
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 i.MX 6SoloLite EVK board, the SoC does not have MAC address fused in. If you want to use FEC, please assign this parameter to the kernel.
max17135	Configure Maxim17135 EPD PMIC pass number and VCOM voltage.	max17235:pass=[pass_num],vcom=[vcom_uV]	Used when enabling EPDC.pass_num. It should equal 2 for all IMXEBOOKDC2 cards. vcom_uV should be equal to the value printed on the cable connector that is attached the E-Ink panel being used.

Table 4. Keypad Mapping

EVK Key	Add-on Board Key	Description
SW2		Reset
SW1		POWER
SW6	SELECT	Enter
SW7	BACK	BACK
SW8	F1	F1
	F2	F2
SW9	F3	F3
SW10	F4	Voulme Down
SW11	F5	Volume Up
	MENU	Menu

Table continues on the next page...

Table 4. Keypad Mapping (continued)

SW12	PREV	Previous
SW13	NEX1	Next
	HOME	Home
	NEX2	Next
	SW16 (Arrow Up)	Up
	SW18 (Arrow Left)	Left
	SW17 (Arrow Right)	Right
	SW19 (Arrow Down)	Down

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AUG
Rev. 13.5.0_GA
11/2012