SMARC-FiMX6-ANDROID-M6.0.1-2.0.1

On this page:

Building Freescale/Embedian’s Android M6.0.1_2.0.1 BSP Distribution

Eric Lee

version 1.0a, 1/15/2017

Introduction

This document describes how to build and deploy Android Marshmallow on the SMARC-FiMX6. It is based on NXP's IMX6_M6.0.1_2.1.0-ga ANDROID release.

Generating SSH Keys

In order to download u-boot and kernel from Embedian. We recommend you use SSH keys to establish a secure connection between your computer and Embedian Gitlab server. The steps below will walk you through generating an SSH key and then adding the public key to our Gitlab account.

Step 1. Check for SSH keys

First, we need to check for existing ssh keys on your computer. Open up Git Bash and run:

```
$ cd ~/.ssh
$ ls
# Lists the files in your .ssh directory
```

Check the directory listing to see if you have a file named either `id_rsa.pub` or `id_dsa.pub`. If you don't have either of those files go to **step 2**. Otherwise, you already have an existing keypair, and you can skip to **step 3**.

Step 2. Generate a new SSH key

To generate a new SSH key, enter the code below. We want the default settings so when asked to enter a file in which to save the key, just press enter.

```
$ ssh-keygen -t rsa -C "your_email@example.com"
# Creates a new ssh key, using the provided email as a label
# Generating public/private rsa key pair.
# Enter file in which to save the key (/c/Users/you/.ssh/id_rsa): [Press enter]
$ ssh-add id_rsa
```

Now you need to enter a passphrase.

```
Enter passphrase (empty for no passphrase): [Type a passphrase]
Enter same passphrase again: [Type passphrase again]
```

Which should give you something like this:

```
Your identification has been saved in /c/Users/you/.ssh/id_rsa.
Your public key has been saved in /c/Users/you/.ssh/id_rsa.pub.
The key fingerprint is:
01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your_email@example.com
```

Step 3. Add your SSH key to Embedian Gitlab Server

Copy the key to your clipboard.

```
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAABDAQABAAABAQDQUEnh8uGpfxaZVU6+uE4bsDrs/tEE5/BPW7jMAxak
6qgOh6nUrQGBWS+VxMM2un3KzwvLRJSj8G4TnTK2CSmlBvR+X8ZeXNTyAdaDxULs/StVhH+QRtFEGy4o
iMIzvIlTyORY89jzhIsgZzwr01nqoSeWWASd+59JWtFjVy0nwVNVtbek7NfuIGGAPaijO5Wnshr2uChB
Pk8ScGjQ3z4VqNXP6CWhCXTqIk7EQl7yX2GKd6FgEFrzae+5Jf63Xm8g6abbE3ytCrMT/jYy5OOj2XSg
6jlxSFnKcONAcfMTWkTXeG/OgeGeG5kZdtqryRtOlGmOeuQe1dd3I+Zz3JyT your_email@example.c
om
```

Go to Embedian Git Server. At Profile Setting --> SSH Keys --> Add SSH Key

Paste your public key and press "Add Key" and your are done.

Overview of this document

The objective of this document is to guide SMARC-FiMX6 Android developers to obtain Android Marshmallow sources, setting up host environment, compilation and deployment.

This document contains instructions for:

- Hardware and software requirements.
- Setup the hardware.
- Setup the toolchain.
- Download & build the sources.
- Install the binaries on the SMARC-FiMX6 SOM.

Hardware Requirement

EVK-STD-CARRIER and SMARC-FiMX6.

Host (PC) setup requirements

The host development environment for Android is based on Ubuntu and Debian, please install Ubuntu version 14.04 64bit LTS http://www.ubuntu.com/download/desktop or Debian 8.4 64bit https://www.debian.org/releases

⚠ Do not use other Ubuntu or Debian releases, than recommended above.

Install required packages on host PC

```
$ sudo apt-get -y install git-core gnupg flex bison gperf build-essential zip curl
zlib1g-dev gcc-multilib g++-multilib
$ sudo apt-get -y install libc6-dev-i386 lib32ncurses5-dev x11proto-core-dev
libx11-dev lib32z-dev ccache libgl1-mesa-dev libxml2-utils
$ sudo apt-get -y install xsltproc unzip mtd-utils u-boot-tools lzop liblzo2-2
liblzo2-dev zlib1g-dev liblz-dev uuid uuid-dev android-tools-fsutils
```

Install the OpenJDK

```
$ sudo apt-get update
$ sudo apt-get install openjdk-7-jdk
```

Update the default Java version by running:

```
$ sudo update-alternatives --config java
$ sudo update-alternatives --config javac
```

⚠ The build machine should have at least 50GB of free space to complete the build process.

Obtain Source Code

Get NXP's Android Release Package

Go to NXP's website, download IMX6_M6.0.1_2.1.0_ANDROID_SOURCE_BSP (filename: android_M6.0.1_2.1.0_source.tar.gz) and put into your ~/downloads directory.

```
$ cd ~/downloads
$ tar xvfz android_M6.0.1_2.1.0_source.tar.gz
```

Download Google Android M6.0.1

```
$ mkdir -p ~/android/smarcfimx6/m_601_210_build
$ cd ~/android/smarcfimx6/m_601_210_build
$ mkdir ~/bin
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=~/bin:$PATH
$ repo init -u https://android.googlesource.com/platform/manifest -b android-6.0.1_r22

$ repo sync -j4
```

Clone Embedian's U-Boot and Linux kernel sources

```
$ mkdir -p ~/android/smarcfimx6/m_601_210_build/bootable/bootloader
$ cd ~/android/smarcfimx6/m_601_210_build/bootable/bootloader
$ git clone git@git.embedian.com:developer/smarc-t335x-uboot.git uboot-imx
$ cd uboot-imx
$ git checkout smarc-m6.0.1_2.1.0-ga
$ cd ~/android/smarcfimx6/m_601_210_build
$ git clone git@git.embedian.com:developer/smarc-fsl-linux-kernel.git kernel_imx
$ cd kernel_imx
$ git checkout smarc-m6.0.1_2.1.0-ga
```

Apply all the i.MX Android patches with Freescale i.MX6 support

```
$ cd ~/android/smarcfimx6/m_601_210_build
$ source ~/downloads/android_M6.0.1_2.1.0_source/code/M6.0.1_2.1.0/and_patch.sh
**** Invoke ". and_patch.sh" from your shell to add following functions to your
environment:
****-- c_gotop:     Changes directory to the top of the tree
****-- c_patch:     Recover working tree to base version and then applying FSL android
patch

$ c_patch  ~/downloads/android_M6.0.1_2.1.0_source/code/M6.0.1_2.1.0 imx_M6.0.1_2.1.0
If everything is OK, "c_patch" generates the following output to indicate the
successful patch:
************************************************************
Success: Now you can build android code for FSL i.MX platform
************************************************************
```

Apply Embedian's i.MX6 platforms' patches

```
$ cd ~/android/smarcfimx6
$ git clone http://git.embedian.com/developer/smarc-fimx6-android.git embedian
$ cd embedian
$ git checkout smarc-m6.0.1_2.1.0-ga
$ cd ../
$ embedian/install
```

Build Android Images

Change to Android top level directory.

```
$ cd ~/android/smarcfimx6/m_601_210_build
$ source build/envsetup.sh
$ export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
$ export PATH=$JAVA_HOME/bin/:$PATH
$ lunch smarc_mx6-eng
or
$ lunch smarc_mx6-user
```

⚠ smarc_mx6-user creates a production version of the Android Marshmallow. smarc_mx6-eng creates an engineering version of the Android Marshmallow. Development mode enable and development tools are available on target.

Switching from eMMC build to SD card build and vice versa

When you switch your target MMC device you need to remove the fstab file. This will guarantee that the make system will copy the right one into target.

```
$ rm out/target/product/smarc_mx6/recovery/root/fstab*
out/target/product/smarc_mx6/root/fstab*
```

Build Android for SD card

```
$ make -j4 BUILD_TARGET_DEVICE=sd 2>&1 | tee build1-1.log
```

⚠ When running Android from an SD card, the eMMC will be detected and presented as an SD card storage.

Build Android for on-SMARC eMMC

```
$ make -j4 BUILD_TARGET_DEVICE=emmc 2>&1 | tee build1-1.log
```

Images created by the Android build for Embedian SMARC-FiMX6 system

All images will be created under *out/target/product/smarc_mx6* directory.

Image	Description
boot-<*name1*>-<*name2*>.img	Boot image that contains zImage, device tree blob and ramdisk
recovery-<*name1*>-<*name2*>.img	Recovery image that contains zImage, device tree blob and ramdisk
system.img	Android system image file.
u-boot-<*defconfig*>.img	Bootloader

1. <*name1*>: *smarcfimx6dl* is for solo and dual lite core and *smarcfimx6dq* is for dual and quad core *i.MX6* processor.
2. <*name2*>: If display output is HDMI or parallel RGB, this field is not necessary. If display output is LVDS LCD, this field stands for the LVDS LCD resolutions (*wvga, wxga, xga, 1080p*, etc...).
3. <*defconfig*>: This is the u-boot defconfig file. If you use quad core and 1GB DDR3L memory, SER3 as your console output port, the u-boot file that you should use is *u-boot-smarcfimx6_quad_1g_ser3_android_defconfig.imx*. If you use dual lite core and use SER3 as console output, the u-boot file that you should use is *u-boot-smarcfimx6_dl_1g_ser3_android_defconfig.imx*.

Setup Bootloader

U-Boot boots from on-module SPI NOR flash, and the rest of the Android images will be loaded from SD card or on-module eMMC.

To flash u-boot into on-module SPI NOR flash. First, you need to prepare for a new SD card and insert into your Linux host PC. The *u-boot-<defconfig>.imx* is pre-installed in SPI NOR flash at factory default. SMARC-FiMX6 is designed to always boot up from SPI NOR flash and to load other Android images based on the setting of *BOOT_SEL*. If users need to fuse their own u-boot or perform u-boot upgrade. This section will instruct you how to do that.

For these instruction, we are assuming: DISK=/dev/mmcblk0, "lsblk" is very useful for determining the device id.

```
$ export DISK=/dev/mmcblk0
```

Erase SD card:

```
$ sudo dd if=/dev/zero of=${DISK} bs=1M count=16
```

Create Partition Layout:

With util-linux v2.26, sfdisk was rewritten and is now based on libfdisk.

sfdisk
```
$ sudo sfdisk --version
sfdisk from util-linux 2.17.1
```

Create Partitions:

sfdisk >=2.26.x
```
$ sudo sfdisk ${DISK} <<-__EOF__
1M,48M,0x83,*
,,,-
__EOF__
```

sfdisk <=2.25
```
$ sudo sfdisk --in-order --Linux --unit M ${DISK} <<-__EOF__
1,48,0x83,*
,,,-
__EOF__
```

Format Partitions:

```
for: DISK=/dev/mmcblk0
$ sudo mkfs.vfat -F 16 ${DISK}p1 -n boot

for: DISK=/dev/sdX
$ sudo mkfs.vfat -F 16 ${DISK}1 -n boot
```

Mount Partitions:

On some systems, these partitions may be auto-mounted...

```
$ sudo mkdir -p /media/boot/

for: DISK=/dev/mmcblk0
$ sudo mount ${DISK}p1 /media/boot/

for: DISK=/dev/sdX
$ sudo mount ${DISK}1 /media/boot/
```

Copy u-boot-<defconfig>.imx to the boot partition.

~/android/smarcfimx6/m_601_210_build/out/target/product/smarc_mx6

```
$ sudo cp -v u-boot-<defconfig>.imx /media/boot/u-boot.imx
```

Install Bootloader

If SPI NOR Flash is not empty (Factory Default)

Fuse u-boot.imx to the SPI NOR flash.

Insert the SD card that you just made into EVK-STD-CARRIER SD card slot. Stop at U-Boot command prompt (Press any key when booting up). Copy and Paste the following script under u-boot command prompt.

u-boot command prompt

```
U-Boot# mmc rescan; mmc dev; load mmc 0:1 0x10800000 u-boot.imx; sf probe; sleep 2; sf erase 0 0xc0000;
sf write 0x10800000 0x400 80000
```

If SPI NOR Flash is empty

In some cases, when SPI NOR flash is erased or the u-boot is under development, we need a way to boot from SD card first. Users need to shunt cross the ***TEST#*** pin to ground. In this way, *SMARC-FiMX6* will always boot up from SD card.

Insert the same SD card into your host Linux PC.

Copy u-boot.imx to the SD card.

~/android/smarcfimx6/m_601_210_build/out/target/product/smarc_mx6

```
$ sudo dd if=u-boot-<defconfig>.imx of=${DISK} bs=512 seek=2
```

Insert the SD card into EVK-STD-CARRIER. Stop at U-Boot command prompt (Press any key when booting up). Copy and Paste the following script under u-boot command prompt.

u-boot command prompt

```
U-Boot# mmc rescan; mmc dev; load mmc 0:1 0x10800000 u-boot.imx; sf probe; sleep 2; sf erase 0 0xc0000;
sf write 0x10800000 0x400 80000
```

Setup SD card

Prepare for the othe SD card that is differmt from the one for bootloader. Insert into your Linux host PC

```
$ cd ~/android/smarcfimx6/m_601_210_build/out/target/product/smarc_mx6/
$ cp ~/android/smarcfimx6/m_601_210_build/emb-mksdcard.sh .
$ sudo ./emb-mksdcard.sh -f <name1>-<name2> /dev/sdX;sync
```


1. *<name1>*: *smarcfimx6dl* is for solo and dual lite core and *smarcfimx6dq* is for dual and quad core *i.MX6* processor.
2. *<name2>*: If display output is HDMI or parallel RGB, this field is not necessary. If display output is LVDS LCD, this field stands for the LVDS LCD resolutions (*wvga, wxga, xga, 1080p*, etc...).

Setup eMMC

First, make sure the images that you built is for eMMC (use *BUILD_TARGET_DEVICE=emmc* when built Android).

Setup eMMC for Android is a bit complex, but trivial. There are a couple of ways to achieve it.

Use MFG Tools v2

NXP/Freescale provides with a way to boot up, partition, format, and program images into eMMC. User can go to NXP's website to download mfgtool and follow their guide to achieve it. We will leave it to users if you would like to use this method to set up your eMMC. Make sure that the *FORCE_RECOV#* pin has to be shunt to Ground when using this tool.

Use a Ubuntu 14.04 Bootable SD card

The second way that we also recommend is to make a bootable Ubuntu 14.04 SD card for SMARC-FiMX6. User go to our Linux Development Site to learn how to make a bootable Ubuntu 14.04 SD card. An prr-built images can be downloaded from our ftp site. Users can download those images and follow the "Setup SD card" section from our Linux development site. Once it done, you can copy the *emb-mksdcard.sh* scipt and all Android images into your home directory. Follow exactly what you did for SD card, but now, eMMC device will be emulated as /dev/mmcblk3.

```
$ sudo ./emb-mksdcard.sh -f <name1>-<name2> /dev/mmcblk3;sync
```


1. *<name1>*: *smarcfimx6dl* is for solo and dual lite core and *smarcfimx6dq* is for dual and quad core *i.MX6* processor.
2. *<name2>*: If display output is HDMI or parallel RGB, this field is not necessary. If display output is LVDS LCD, this field stands for the LVDS LCD resolutions (*wvga, wxga, xga, 1080p*, etc...).

Power off and set BOOT_SEL to OFF ON ON and you will be able to boot up your Android from on-module eMMC.

Use USB Fastboot

The third way is to use Android USB fast boot. This way will work only when the on-module eMMC is partitioned and formatted. To partition and format the on-module eMMC. You can use the SD card mentioned above.

```
$ sudo ./emb-mksdcard.sh -np /dev/mmcblk3
```

Once you have your on-module eMMC partitioned and formated.

On your Linux host PC, you need to install Android tools.

```
$ sudo apt-get install android-tools-adb android-tools-fastboot
```

Connect the device with host PC at fastboot mode:

1. Connect a USB OTG cable from the target board OTG port to a your host machine USB HOST port.
2. Power up the board and hit return/space to stop the boot at U-Boot.
3. type **fastboot** in the U-Boot command line.

On the Host PC:

```
$ sudo fastboot flash boot out/target/product/smarc_mx6/boot-<name1>-<name2>.img
$ sudo fastboot flash recovery
out/target/product/smarc_mx6/recovery-<name1>-<name2>.img
$ sudo fastboot flash system out/target/product/smarc_mx6/system.img
$ sudo fastboot reboot
```

Android Recovery Mode

Enter board in Android Recovery mode

Shunt LID# pin to ground will enter Android Recovery mode.

Update Android Firmware

Generate OTA Packages

For generating "OTA" packages, use the following commands:

```
$ cd ~/android/smarcfimx6/m_601_210_build/
# if Android for SD card
$ make -j4 BUILD_TARGET_DEVICE=sd otapackage 2>&1 | tee build1-1.log
# if Android for eMMC
$ make -j4 BUILD_TARGET_DEVICE=emmc otapackage 2>&1 | tee build1-1.log
```

Install OTA Packages to device

1. Enter to Android Recovery mode
2. Select menu item "apply update from ADB"
3. To the host system, perform the following command:

```
$ out/host/linux-x86/bin/adb sideload
out/target/product/smarc_mx6/smarc_mx6-ota-<data>-<image-id>.zip
```

Reboot the device.

Real example name for OTA package: *out/target/product/smarc_mx6/smarc_mx6-ota-20170114-smarcfimx6dq-wvga.zip*

Manual Operations

Build boot.img

When you perform changes to the kernel, you may build boot.img solely instead of building the whole Android.

```
$ cd ~/android/smarcfimx6/m_601_210_build/
$ source build/envsetup.sh
$ lunch smarc_mx6-user (or smarc_mx6-eng)
$ make bootimage
```

Toolchain setup for manual build kernel and U-Boot

Setup the toolchain path to point to arm-eabi- tools in prebuilts/gcc/linux-x86/arm/arm-eabi-4.8/bin

```
$ export ARCH=arm
$ export
CROSS_COMPILE=~/android/smarcfimx6/m_601_210_build/prebuilts/gcc/linux-x86/arm/arm-eab
i-4.8/bin/arm-eabi-
```

Manual build Bootloader

Change directory to U-Boot

```
$ cd ~/android/smarcfimx6/m_601_210_build/bootable/bootloader/uboot-imx
```

Execute following commands:

```
$ make distclean
$ make smarcfimx6_quad_1g_ser3_android_defconfig
$ make -j4
```

Note1:

If the board is SMARC-FiMX6-Q-2G or SMARC-FiMX6-D-2G, use
$ make ARCH=arm CROSS_COMPILE=${CC} smarcfimx6_quad_2g_ser3_android_defconfig

If the board is SMARC-FiMX6-Q-1G or SMARC-FiMX6-D-1G, use
$ make ARCH=arm CROSS_COMPILE=${CC} smarcfimx6_quad_1g_ser3_android_defconfig

If the board is SMARC-FiMX6-U-1G, use
$ make ARCH=arm CROSS_COMPILE=${CC} smarcfimx6_dl_1g_ser3_android_defconfig

If the board is SMARC-FiMX6-S, use
$ make ARCH=arm CROSS_COMPILE=${CC} smarcfimx6_solo_ser3_android_defconfig

Note 2:

"*ser3*" stands for console debug port. In this example, we uses SER3 as debug port. If user uses SER0 as your debug port, make change to "*ser0*" instead. Same as SER1 and SER2.

Note 3:

The *SMARC-FiMX6* module always boot up from the onboard *SPI NOR* flash. The factory default will be *u-boot.imx* pre-installed. In some cases when the *SPI NOR* flash is empty or needs to be upgraded. Users can shunt crossed the *TEST#* to ground. In this way, the *SMARC-FiMX6* module will boot up to carrier SD card, if *TEST#* pin is shunt crossed. The *u-boot.imx* image are the same, the difference is how you flash *u-boot.imx*. This will be explained in the "Setup Bootloader" and "Setup SD card" section.

It will generate u-boot.imx file.

Manual build Android Linux Kernel and modules

```
$ cd ~/android/smarcfimx6/m_601_210_build/kernel_imx
$ make distclean
$ make smarcfimx6_android_defconfig
$ make -j4 uImage LOADADDR=0x10008000
$ make -j4 modules
```

This will generate the uImage (kernel image) in the kernel/arch/arm/boot folder

---

version 1.0a,1/15/2017

Last updated 2017-01-25