SMARC-iMX8M-ANDROID-Q10.0.0_2.6.0

On this page:

Building Freescale/Embedian’s Android Q10.0.0_2.6.0 BSP Distribution

Eric Lee

version 1.0a, 11/24/2021

Introduction

This document describes how to build and deploy Android Oreo on the SMARC-iMX8M. It is based on NXP's IMX8M_Q10.0.0_2.6.0 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-iMX8M Android developers to obtain Android Q10.0.0_1.0.0 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-iMX8M SOM.

Hardware Requirement

EVK-STD-CARRIER-S20 and SMARC-iMX8M.

Host (PC) setup requirements

The host development environment for Android is based on Ubuntu and Debian, please install Ubuntu version 16.04 64bit LTS http://www.ubuntu.com/download/desktop or Debian 9.6 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-8-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 Q10.0.0_2.6.0_ANDROID_SOURCE (filename: `imx-android-10.0.0_2.6.0.tar.gz` and put into your

~/downloads directory.

```
$ cd ~/downloads
$ tar xvfz imx-android-10.0.0_2.6.0.tar.gz
```

Obtain Google Android Q10.0.0_2.6.0 source code and Apply NXP's patch

```
$ mkdir -p ~/android/smarcimx8mq/q_1000_260
$ cd ~/android/smarcimx8mq/q_1000_260
$ mkdir ~/bin
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=~/bin:$PATH
$ mv ~/download/imx-android-10.0.0_2.6.0 .
$ source imx-android-10.0.0_2.6.0/imx_android_setup.sh
```

After done, it will create an android_build directory. Go to this directory.

Clone Embedian's U-Boot (2020.04) and Linux kernel (5.4.70) sources

```
$ cd ~/android/smarcimx8mq/q_1000_260/android_build
$ mkdir -p vendor/embedian
$ cd vendor/embedian
$ git clone git@git.embedian.com:developer/smarc-t335x-uboot.git uboot-imx -b
smarc-8m-android-10.0.0_2.6.0
$ git clone git@git.embedian.com:developer/smarc-fsl-linux-kernel.git kernel_imx -b
smarc-8m-android-10.0.0_2.6.0
```

Apply Embedian's patches for i.MX8M platforms

```
$ cd ~/android/smarcimx8mq/q_1000_260/android_build/device
$ git clone git@git.embedian.com:developer/smarc-imx8m-android.git embedian -b
smarc-8mq-q10.0.0_2.6.0
$ embedian/scripts/install.sh
```

Build Android Images

Change to Android top level directory.

```
$ export MY_ANDROID=~/android/smarcimx8mq/q_1000_260/android_build
$ cd ${MY_ANDROID}
$ source build/envsetup.sh
$ export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
$ export PATH=$JAVA_HOME/bin/:$PATH
$ lunch smarc_mx8mq-eng
or
$ lunch smarc_mx8mq-userdebug
$ ./imx-make.sh -j4 2>&1 | tee build-1.log
```

userdebug build creates a debuggable version of Android. eng build creates an engineering version of Android. Development mode enable and development tools are available on target.

The commands below can achieve the same result as "`./imx-make.sh -j4 2>&1 | tee build-1.log`":

```
$ ./imx-make.sh bootloader kernel -j4 2>&1 | tee build-log.txt
# Build U-Boot/kernel with imx-make.sh first, but not to build Android images.
$ make -j4 2>&1 | tee -a build-log.txt
# Start the process of build Android images with "make" function.
```

Images created by the Android build for SMARC-iMX8M system

The images created are located at out/target/product/smarc_mx8m/ directory.

Image	Description
u-boot-imx8mq-smarc_2g.imx	Bootloader for eMMC/SD card boot for SMARC-iMX8M-D/Q/L-2G(-I) Other SMARC variants could be defined at device/embedian/imx8m/smarc_mx8mq/BoardConfig.mk
u-boot-imx8mq-smarc_2g-uuu.imx	Bootloader used by uuu for eMMC/SD card boot for SMARC-iMX8M-D/Q/L-2G(-I) Other SMARC variants could be defined at device/embedian/imx8m/smarc_mx8mq/BoardConfig.mk
u-boot-imx8mq-trusty-smarc_2g.imx	Trusty Bootloader for eMMC/SD card boot for SMARC-iMX8M-Q/L-2G(-I) Other SMARC variants could be defined at device/embedian/imx8m/smarc_mx8mq/BoardConfig.mk
u-boot-imx8mq-trusty-secure-unlock-smarc_2g.imx	Trusty Bootloader for eMMC/SD card boot for SMARC-iMX8M-Q/L-2G(-I) Other SMARC variants could be defined at device/embedian/imx8m/smarc_mx8mq/BoardConfig.mk It is a demonstration of secure unlock mechanism.
u-boot-imx8mq-smarc_4g.imx	Bootloader for eMMC/SD card boot for SMARC-iMX8M-D/Q/L-4G(-I) Other SMARC variants could be defined at device/embedian/imx8m/smarc_mx8mq/BoardConfig.mk
u-boot-imx8mq-smarc_4g-uuu.imx	Bootloader used by uuu for eMMC/SD card boot for SMARC-iMX8M-D/Q/L-4G(-I) Other SMARC variants could be defined at device/embedian/imx8m/smarc_mx8mq/BoardConfig.mk

u-boot-imx8mq-trusty-smarc_4g.imx	Trusty Bootloader for eMMC/SD card boot for SMARC-iMX8M-Q/L-4G(-I) Other SMARC variants could be defined at device/embedian/imx8m/smarc_mx8mq/BoardConfig.mk
u-boot-imx8mq-trusty-secure-unlock-smarc_4g.imx	Trusty Bootloader for eMMC/SD card boot for SMARC-iMX8M-Q/L-4G(-I) Other SMARC variants could be defined at device/embedian/imx8m/smarc_mx8mq/BoardConfig.mk It is a demonstration of secure unlock mechanism.
boot.img	Boot image for SMARC-iMX8M. It contains kernel, a part of ramdisk, and default kernel command line.
partition-table.img	GPT table image for 16GB SD card and eMMC.
partition-table-7GB.img	GPT table image for 8GB SD card and eMMC.
partition-table-28GB.img	GPT table image for 28GB SD card and eMMC.
system.img	System image
vbmeta-imx8mq-<dtb_feature>.img	Android Verified Boot metadata Image for SMARC-iMX8M to support different display output
vendor.img	Vendor image
dtbo-<dtb_feature>.img	Device Tree image for SMARC-iMX8M to support different display output
product.img	Product image for SMARC-iMX8M
super.img	Super image generated from system.img, system_ext.img, vendor.img, and product.img.
<dtb_feature>	*<blank>* Support no display configuration. *hdmi* Support HDMI display configuration (DCSS). *dp* Support Display Port (DP) display configuration (DCSS). *lcdif-lvds* Support LCDIF LVDS display configuration. *dcss-lvds* Support DCSS LVDS display configuration. *dual-display* Support dual LVDS+HDMI display configuration.

DCSS vs LCDIF

i.MX8M comes with 2 display controllers: DCSS and LCDIF.

DCSS can be connected to either HDMI or MIPI-DSI (to LVDS bridge) and supports resolutions up to 4K.

LCDIF can be connected only to MIPI-DSI and supports resolutions up to 1080p.

UUU Scripts

The table below describes the UUU scripts in android_q10.0.0_2.6.0. They are used with the UUU binary file to download the images above into SMARC-iMX8M SMARC modules.

UUU Script name	Function
uuu-android-smarc-mx8mq-emmc.lst	Used with the UUU binary file to download image files into eMMC.

uuu-android-smarc-mx8mq-sd.lst	Used with the UUU binary file to download image files into SD card.
`uuu_android_smarc_flash.bat`	Script to generate lst file on Windows OS
`uuu_android_smarc_flash.sh`	Script to generate lst file on Linux OS

Setup SD card

Prepare for an SD card and insert into your Linux host PC

```
$ cp smarc-mksdcard.sh out/target/product/smarc_mx8mq/
$ cd out/target/product/smarc_mx8mq/
$ chmod a+x smarc-mksdcard.sh
$ sudo ./smarc-mksdcard.sh -f imx8mq -d <dtb_feature> -u <uboot_feature> /dev/sdX;
sync
```

`<dtb_feature>`	***`N/A`*** **Support no display configuration.** ***`hdmi`*** **Support HDMI display configuration (DCSS).** ***`dp`*** **Support Display Port (DP) display configuration (DCSS).** ***`lcdif-lvds`*** **Support LCDIF LVDS display configuration.** ***`dcss-lvds`*** **Support DCSS LVDS display configuration.** ***`dual-display`*** **Support dual LVDS+HDMI display configuration.**
`<uboot_feature>`	***`smarc_2g`*** **If the LPDDR4 memory capacity on your module is 2GB.** ***`smarc_4g`*** **If the LPDDR4 memory capacity on your module is 4GB.**

1. The minimum size of the SD card is 8 GB.
2. /dev/sdX, the X is the disk index from 'a' to 'z'. That may be different on each computer running Linux OS.
3. If the SD card is 16 GB, use "sudo ./smarc-mksdcard.sh -f smarcimx8mq -d <name> -u <uboot_feature> /dev/sdX" to flash images.
4. If the SD card is 8 GB, use "sudo ./smarc-mksdcard.sh -f smarcimx8mq -d <name> -u <uboot_feature> -c 7 /dev/sdX" to flash images.
5. If the SD card is 32 GB, use "sudo ./smarc-mksdcard.sh -f smarcimx8mq -d <name> -u <uboot_feature> -c 28 /dev/sdX" to flash images.

If primary display set to HDMI, a VESA standard HDMI display need to be connected before booting up.

Insert the SD card into your device, set the *BOOT_SEL* to "ON OFF OFF" and shut cross the *TEST#* pin to Ground. You will be able to see Android booting up. For eMMC boot, leave the TEST# floating and the *BOOT_SEL* should be set as "OFF ON ON". The next section will instruct you to flash eMMC.

Setup eMMC

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

Use UUU

UUU can be used to download all images into a target device. It is a quick and easy tool for downloading images. See the *AndroidTM Quick Start Guide (AQSUG)* for detailed description of UUU.

Make sure that the *FORCE_RECOV#* pin has to be shunt to Ground to enter into CPU serial download mode when using this tool. Connect USB0 min-B port of the device to your host computer.

Copy *uuu.exe, u-boot-imx8mq-<u-boot_feature>.imx, u-boot-imx8mq-smarc_2g-uuu.imx, u-boot-imx8mq-smarc_4g-uuu.imx, partition-table.img, boot.img, vbmeta-<dtb_feature>.img, product.img, system.img, vendor.img, lpmake.exe, dtbo-<dtb_feature>* and `uuu_android_smarc_flash.bat` into the same folder.

```
$ .\uuu_android_smarc_flash.bat -f imx8mq -d <dtb_feature> -u <uboot_feature> -e
```

<dtb_feature>	***hdmi*** **Support HDMI display configuration (DCSS).** ***dp*** **Support Display Port (DP) display configuration (DCSS).** ***lcdif-lvds*** **Support LCDIF LVDS display configuration.** ***dcss-lvds*** **Support DCSS LVDS display configuration.** ***dual-display*** **Support dual LVDS+HDMI display configuration.**

<uboot_feature>	***smarc_2g*** **If the LPDDR4 memory capacity on your module is 2GB.** ***smarc_4g*** **If the LPDDR4 memory capacity on your module is 4GB.** ***trusty-smarc_2g*** **Trusty OS, if the LPDDR4 memory capacity on your module is 2GB.** ***trusty-smarc_4g*** **Trusty OS, if the LPDDR4 memory capacity on your module is 4GB.**

After done, make sure that *FORCE_RECOV#* and *TEST#* pins are floating. Change the *BOOT_SEL* to OFF ON ON and the module will boot up from eMMC.

Use a Ubuntu 18.04 Bootable SD card

The second way that we also recommend is to make a bootable Ubuntu 16.04 SD card for SMARC-iMX8M. User go to our Linux Development Site to learn how to make a bootable Ubuntu 16.04 SD card. An pre-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 smarc-*mksdcard.sh* script and all Android images (u-boot-imx8mq-smarcimx8mq-2g.imx, u-boot-imx8mq-smarcimx8mq-4g.imx, partition-table.img, boot.img, vbmeta-<name>.img, system_raw.img, vendor_raw.img, dtbo-<name>, smarc-mkemmccard.sh) into your home directory. Follow exactly what you did for SD card, but now, eMMC device will be emulated as /dev/mmcblk0.

```
$ sudo ./smarc-mkemmccard.sh -f imx8mq -d <dtb_feature> -u <uboot_feature>
/dev/mmcblk0; sync
```

`<dtb_feature>`	***N/A*** **Support no display configuration.** ***hdmi*** **Support HDMI display configuration (DCSS).** ***dp*** **Support Display Port (DP) display configuration (DCSS).** ***lcdif-lvds*** **Support LCDIF LVDS display configuration.** ***dcss-lvds*** **Support DCSS LVDS display configuration.** ***dual-display*** **Support dual LVDS+HDMI display configuration.**

`<uboot_feature>`	***smarc_2g*** **If the LPDDR4 memory capacity on your module is 2GB.** ***smarc_4g*** **If the LPDDR4 memory capacity on your module is 4GB.** ***trusty-smarc_2g*** **Trusty OS, if the LPDDR4 memory capacity on your module is 2GB.** ***trusty-smarc_4g*** **Trusty OS, if the LPDDR4 memory capacity on your module is 4GB.**

Power off and set *BOOT_SEL* to "OFF ON ON", leave the TEST# pin floating and you will be able to boot up your Android from on-module eMMC.

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 ${MY_ANDROID}
$ make PRODUCT=smarc_mx8mq-userdebug -j4 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_mx8mq/smarc_mx8mq-ota-<data>-<name>.zip
```

smarc_mx8mq-ota-${date}.zip includes payload.bin and payload_properties.txt . The two files are used for full update.

Reboot the device.

⚠ Real example name for OTA package: *out/target/product/smarc_mx8mq/smarc_mx8mq-ota-20180416-smarcimx8mq-hdmi.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 ${MY_ANDROID}
$ source build/envsetup.sh
$ export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
$ export PATH=$JAVA_HOME/bin/:$PATH
$ lunch smarc_mx8mq-eng
or
$ lunch smarc_mx8mq-userdebug
$ make bootimage
```

Manual build Bootloader

```
$ cd ${MY_ANDROID}
$ source build/envsetup.sh
$ export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
$ export PATH=$JAVA_HOME/bin/:$PATH
$ lunch smarc_mx8mq-eng
or
$ lunch smarc_mx8mq-userdebug
$ ./imx-make.sh bootloader -j4
```

It will generate `u-boot-imx8mq-smarc_2g.imx`, `u-boot-imx8mq-smarc_2g-uuu.imx`, `u-boot-imx8mq-smarc_2g-trusty.imx`, `u-boot-imx8mq-smarc_4g.imx`, `u-boot-imx8mq-smarc_4g-uuu.imx`, and `u-boot-imx8mq-smarc_4g-trusty.imx` files.

Manual build Android Linux Kernel and modules

```
$ cd ${MY_ANDROID}
$ source build/envsetup.sh
$ export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
$ export PATH=$JAVA_HOME/bin/:$PATH
$ lunch smarc_mx8mq-eng
or
$ lunch smarc_mx8mq-userdebug
$ ./imx-make.sh kernel -j4
```

The kernel images are found in ${MY_ANDROID}/out/target/product/smarc_mx8mq/obj/KERNEL_OBJ/arch/arm64/boot/Image

version 1.0a, 11/24/2021

Last updated 2021-12-05

`<name>`

`smarcimx8mq` Support no display configuration.

`smarcimx8mq-hdmi` Support HDMI display configuration (DCSS).

`smarcimx8mq-dp` Support Display Port (DP) display configuration (DCSS).

`smarcimx8mq-edp` Support Embedded Display Port (eDP) display configuration (DCSS).

`smarcimx8mq-lcdif-lvds` Support LCDIF LVDS display configuration.

`smarcimx8mq-dcss-lvds` Support DCSS LVDS display configuration.

`smarcimx8mq-dual-display` Support dual LVDS+HDMI display configuration.

```
Trusty Bootloader for eMMC/SD card boot for SMARC-iMX8M-Q/L-2G(-I)

Other SMARC variants could be defined at device/embedian/imx8m/smarc_mx8mq/BoardConfig.mk
```