Debian_Stretch_SMARC-iMX8M

Build and Install Debian Stretch for *SMARC-iMX8M* (Dual, Quad Lite and Quad Core)

This document provides instructions for advanced users how Embedian offers patches and builds Debian Stretch for Embedian's *SMARC-iMX8M* product platform and how to install the images to bring the evaluation board up and running.

Our aim is to fully support our hardware through device drivers. We also provide unit tests so that testing a board is easy and custom development can start precisely. The recommended host environment is Ubuntu 16.04.

Availability

SMARC-iMX8M from Embedian

Carrier Board

EVK-STD-CARRIER-S20 (universal carrier board for all SMARC 2.0 modules) from Embedian

Basic Resources

- AArch64 Cross Compiler
    - Linaro: https://launchpad.net/linaro-toolchain-binaries
- Bootloader
    - Das U-Boot – the Universal Boot Loader http://www.denx.de/wiki/U-Boot

    - Source – http://git.denx.de/?p=u-boot.git;a=summary
- Linux Kernel
    - Linus's Mainline tree: http://git.kernel.org/?p=linux/kernel/git/torvalds/linux.git;a=summary
    - Freescale Linux source tree: git://git.freescale.com/imx/linux-imx.git
    - Freescale BSP meta layer: git://git.freescale.com/imx/meta-fsl-bsp-release
    - OpenEmbedded/Yocto BSP layer for Freescale's ARM platform git://git.yoctoproject.org/meta-fsl-arm
    - Embedian SMARC-iMX8M kernel source tree for linux smarc-8m_imx_4.14.78_1.0.0_ga: git@git.embedian.com:developer/smarc-fsl-linux-kernel.git
- ARM based rootfs
    - Debian Squeeze: http://www.debian.org/

Generating SSH Keys

We recommend you use SSH keys to establish a secure connection between your computer and Embedian Gitlab server. To download u-boot and kernel source codes from Embedian server. You need to register from Embedian's Gitlab server and put your ssh public key there. 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.

Create build environment

Install required packages

On Ubuntu machine:

```
$ sudo apt-get install binfmt-support qemu qemu-user-static debootstrap kpartx \

lvm2 dosfstools gpart binutils git lib32ncurses5-dev python-m2crypto gawk wget \
git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat libsdl1.2-dev \
autoconf libtool libglib2.0-dev libarchive-dev python-git xterm sed cvs subversion \
coreutils texi2html bc docbook-utils python-pysqlite2 help2man make gcc g++ \
desktop-file-utils libgl1-mesa-dev libglu1-mesa-dev mercurial automake groff curl \
lzop asciidoc u-boot-tools mtd-utils device-tree-compiler
```

Deploy Sources

Download archive containing the build script and support files for building Debian Stretch

```
$ cd ~/

$ git clone git@git.embedian.com:developer/smarc_mx8m_debian_stretch.git -b debian_stretch_mx8m_smarc01

$ cd ~/smarc_mx8m_debian_stretch

$ ./make_smarc_mx8m_debian.sh -c deploy
```

This environment prepared to build.

Make Debian

Build all

The internet connection in your host PC has to be available.

```
$ cd ~/smarc_mx8m_debian_stretch
```

```
$ sudo ./make_smarc_mx8m_debian.sh -c all | tee 2.log
```

Build by parts

Build bootloader

```
$ cd ~/smarc_mx8m_debian_stretch
$ sudo ./make_smarc_mx8m_debian.sh -c bootloader
```

Build kernel, dtb files and kernel modules

```
$ cd ~/smarc_mx8m_debian_stretch
$ sudo ./make_smarc_mx8m_debian.sh -c kernel
$ sudo ./make_smarc_mx8m_debian.sh -c modules
```

Build rootfs

```
$ cd ~/smarc_mx8m_debian_stretch
$ sudo ./make_smarc_mx8m_debian.sh -c rootfs
```

Pack rootfs

```
$ cd ~/smarc_mx8m_debian_stretch
$ sudo ./make_smarc_mx8m_debian.sh -c rtar
```

Setup SD card Automatically

```
$ cd ~/smarc_mx8m_debian_stretch
$ sudo ./make_smarc_mx8m_debian.sh -c sdcard -d /dev/sdX
```

where "/dev/sdX" is the SD block device in your host system. Shunt the "TEST#" pin of your device to Ground. Insert the SD card and you will see SMARC-iMX8M booting with Debian Stretch. Console port is defined on *SER3*.

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.

Selecting display configuration is a matter of selecting an appropriate DTB file.

All available DTB files are listed in the table below.

DTB File Name	Description
fsl-smarcimx8mq.dtb	Device tree blob for no display configuration.
fsl-smarcimx8mq-hdmi.dtb	Device tree blob for HDMI display configuration (DCSS).
fsl-smarcimx8mq-dp.dtb	Device tree blob for Display Port (DP) display configuration (DCSS).
fsl-smarcimx8mq-lcdif-lvds.dtb	Device tree blob for LCDIF LVDS display configuration.
fsl-smarcimx8mq-dcss-lvds.dtb	Device tree blob for DCSS LVDS display configuration.
fsl-smarcimx8mq-dual-display.dtb	Device tree blob for dual LVDS+HDMI display configuration.
fsl-smarcimx8mq-edp.dtb	Device tree blob for Embedded Display Port (eDP) display configuration (DCSS).

1. The default display output is HDMI. If you would like to change to default display output to different interfaces, make chages the file make_smarc_mx8m_debian.sh and find readonly DISPLAY="-hdmi"
   No Display: readonly DISPLAY=""
   LVDS (DCSS): *readonly DISPLAY="-dcss-lvds"*
   LVDS (LCDIF): *readonly DISPLAY="lcdif-lvds"*
   Dual Display: *readonly DISPLAY="-dual-display"*
   Display Port: *readonly DISPLAY="-dp"*
   Embedded Display Port: *readonly DISPLAY="-edp"*

Setup SD Card Manually

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: Leave 2MB offset for flash.bin.

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

sfdisk
```
$ sudo sfdisk --version
sfdisk from util-linux 2.27.1
```

Create Partitions:

sfdisk >=2.26.x
```
$ sudo sfdisk ${DISK} <<-__EOF__
2M,48M,0x83,*
50M,,,
__EOF__
```

sfdisk <=2.25

```
$ sudo sfdisk --in-order --Linux --unit M ${DISK} <<-__EOF__
2,48,0x83,*
,,,-
__EOF__
```

Format Partitions:

```
for: DISK=/dev/mmcblk0
$ sudo mkfs.vfat -F 16 ${DISK}p1 -n boot
$ sudo mkfs.ext4 ${DISK}p2 -L rootfs

for: DISK=/dev/sdX
$ sudo mkfs.vfat -F 16 ${DISK}1 -n boot
$ sudo mkfs.ext4 ${DISK}2 -L rootfs
```

Mount Partitions:

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

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

for: DISK=/dev/mmcblk0
$ sudo mount ${DISK}p1 /media/boot/
$ sudo mount ${DISK}p2 /media/rootfs/

for: DISK=/dev/sdX
$ sudo mount ${DISK}1 /media/boot/
$ sudo mount ${DISK}2 /media/rootfs/
```

Install Boot File

Users need to shunt cross the ***TEST#*** pin to ground. In this way, *SMARC-iMX8M* will always boot up from SD card.

Fuse flash.bin to the SD card.

~/

```
$ cd ~/smarc_mx8m_debian_stretch
$ sudo dd if=output/imx-boot-sd.bin of=${DISK} bs=1024 seek=33
```

uEnv.txt based bootscript

Copy uEnv.txt to the boot partition:

~/smarc_mx8m_debian_stretch

```
$ sudo cp -v embedian/uEnv.txt /media/boot/
```

Install Kernel Image

Copy Image to the boot partition:

~/smarc_mx8m_debian_stretch

```
$ sudo cp -v output/Image /media/boot
```

Install Kernel Device Tree Binary

```
$ sudo mkdir -p /media/boot/dtbs
$ sudo cp -v output/<device tree name> /media/boot/dtbs/fsl-smarcimx8mq.dtb
```

Selecting display configuration is a matter of selecting an appropriate DTB file.

All available DTB files are listed in the table below.

DTB File Name	Description
fsl-smarcimx8mq.dtb	Device tree blob for no display configuration.
fsl-smarcimx8mq-hdmi.dtb	Device tree blob for HDMI display configuration (DCSS).
fsl-smarcimx8mq-dp.dtb	Device tree blob for Display Port (DP) display configuration (DCSS).
fsl-smarcimx8mq-lcdif-lvds.dtb	Device tree blob for LCDIF LVDS display configuration.
fsl-smarcimx8mq-dcss-lvds.dtb	Device tree blob for DCSS LVDS display configuration.
fsl-smarcimx8mq-dual-display.dtb	Device tree blob for dual LVDS+HDMI display configuration.
fsl-smarcimx8mq-edp.dtb	Device tree blob for Embedded Display Port (eDP) display configuration (DCSS).

The device tree name in your SD card has be to fsl-smarcimx8mq.dtb

Install Root File System and Kernel Modules

Extract Root File System:

directory where your root file system is

```
$ sudo tar xvfz output/rootfs.tar.gx -C /media/rootfs
```

Note

1. MAC address is factory pre-installed at on board I2C EEPROM at offset 60 bytes). It starts with Embedian's vendor code *10:0D :32*. u-boot will read it and pass this parameter to kernel.
2. Kernel module is pre-built in debian rootfs
3. The default console port is *SER3*.

Remove SD card:

```
$ sync
$ sudo umount /media/boot
$ sudo umount /media/rootfs
```

Build Results

The resulted images are locared in *~/smarc_mx8m_debian_stretch/output* directory:

Image Name	Description
rootfs.tar.gz	Root filesystem tarball for installation on SD card and eMMC

Image	Linux Kernel Image
imx-boot-sd.bin	Boot file for SD card and eMMC

DTB File Name	Description
fsl-smarcimx8mq.dtb	Device tree blob for no display configuration.
fsl-smarcimx8mq-hdmi.dtb	Device tree blob for HDMI display configuration (DCSS).
fsl-smarcimx8mq-dp.dtb	Device tree blob for Display Port (DP) display configuration (DCSS).
fsl-smarcimx8mq-lcdif-lvds.dtb	Device tree blob for LCDIF LVDS display configuration.
fsl-smarcimx8mq-dcss-lvds.dtb	Device tree blob for DCSS LVDS display configuration.
fsl-smarcimx8mq-dual-display.dtb	Device tree blob for dual LVDS+HDMI display configuration.
fsl-smarcimx8mq-edp.dtb	Device tree blob for Embedded Display Port (eDP) display configuration (DCSS).

Linux Console Access

User Name	User Password	User Descriptor
root	root	system administrator
user	user	local user
x_user		used for X session access

Setup eMMC

Shunt TEST# pin to Ground and boot up from your SD card. Run the following command as ***root*** user.

```
$ debian-emmc.sh -d <-hdmi/-dp/-lcdif-lvds/-dcss-lvds/-dual-display/-edp>
```

1. The "-d" parameter specifies which device tree blob that you would like copy to eMMC.
   No Display: readonly DISPLAY=""
   LVDS (DCSS): *-d -dcss-lvds*
   LVDS (LCDIF): *-d lcdif-lvds*
   Dual Display: *-d -dual-display*
   Display Port: *-d -dp*
   Embedded Display *-d -edp*

Modify the kernel configuration

To modify the kernel configuration (add/remove features and drivers). Please follow the step below.

```
$ cd ~/smarc_mx8m_debian_stretch/src/kernel

$ sudo make arch=arm64 mrproper
```

```
$ sudo make arch=arm64 smarcimx8m_defconfig

$ sudo make arch=arm64 menuconfig


Navigate the menu and select the desired kernel functionality Exit the menu and answer "Yes" when asked "Do you wish to save your new configuration?" $ sudo make ARCH=arm64 savedefconfig $ sudo cp arch/arm64/configs/smarcimx8m_defconfig arch/arm64/configs/smarcimx8m_defconfig.orig $ sudo cp .config arch/arm64/configs/smarcimx8m_defconfig Follow the instructions above to rebuild kernel and modules, repack rootfs images and recreate SD card
```

Video Decoding

For playing video, we can use three solutions to support it.

a) `# gplay-1.0 <video file>`

b) `# gst-launch-1.0 playbin uri=file://<video absolute path>`

c) (i) if video container on .mp4 format

```
    # gst-launch-1.0 filesrc location=<file name.mp4> typefind=true ! video/quicktime ! qtdemux ! queue max-size-time=0 ! vpudec ! queue max-size-time=0 ! kmssink force-hantrope=true sync=false &
```

(ii) if video container on .ts format

```
    # gst-launch-1.0 filesrc location=<file name.ts> typefind=true ! video/mpegts ! tsdemux ! queue max-size-time=0 ! vpudec ! queue max-size-time=0 ! waylandsink
```

---

version 1.0a, 5/13/2019

Last updated 2019-05-13