pITX-iMX8MP_lf-5.10.y_1.0.0

Build and Install Linux System for *pITX-MX8M-PLUS* (Dual and Quad Core)

This document provides instructions for advanced users how Embedian offers patches and builds a customized version of u-boot and linux kernel for Embedian's *pITX-MX8M-PLUS* 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 or 18.04 LTS.

Availability

pITX-MX8M-PLUS from Embedian

Carrier Board

pITX-IOB-2201 (Expansion IO board for pITX-MX8M-PLUS) at 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 pITX-MX8M-PLUS kernel source tree for linux pitx-8mp_lf-5.10.y: git@git.embedian.com:developer/pitx-fsl-linux-kernel.git
- ARM based rootfs
    - Debian Squeeze: http://www.debian.org/

ARM Cross Compiler: GCC

To build Embedian's *pITX-MX8M-PLUS* u-boot and linux kernel, you will need to install the following ARM GNU aarch64 compiler first:

For **u-boot 2020.04 and Linux 5.10.9**, you need to use the following ARM GNU aarch64 coss compilier.

```
$ wget -c https://developer.arm.com/-/media/Files/downloads/gnu-a/8.3-2019.03/binrel/gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu.tar.xz

$ sudo tar -C /opt -xJf gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu.tar.xz

$ export CC=/opt/gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu/bin/aarch64-linux-gnu-
```

Test:

If this test fails, verify that you have the 32bit libraries installed on your development system.

```
$ ${CC}gcc --version

aarch64-linux-gnu-gcc (GNU Toolchain for the A-profile Architecture 8.3-2019.03 (arm-rel-8.36)) 8.3.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Generating SSH Keys

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.

Boot File: flash.bin

The boot file is called flash.bin. It is made up of some pieces of programs. This section instruct you how to generate flash.bin.

1. Download the *imx-mkimage* tool and apply Embedian's patch to accept Embedian's device tree blob.

```
$ git clone https://source.codeaurora.org/external/imx/imx-mkimage -b lf-5.10.y_1.0.0
$ cd imx-mkimage
$ wget -c ftp://ftp.embedian.com/public/pitximx8mp/0001-change-uboot-device-tree-name.patch
$ patch -p1 <0001-change-uboot-device-tree-name.patch
$ cd ../
```

2. Get and Build the ARM Trusted firmware and copy bl31.bin to imx-mkimage/iMX8M directory.

```
$ git clone https://source.codeaurora.org/external/imx/imx-atf -b lf_v2.4

$ cd imx-atf

$ wget -c ftp://ftp.embedian.com/public/pitximx8mp/0001-imx-atf-change-console-to-ttymxc3.patch

$ patch -p1 <0001-imx-atf-change-console-to-ttymxc3.patch

$ make CROSS_COMPILE=${CC} PLAT=imx8mp bl31

$ cp build/imx8mp/release/bl31.bin ../imx-mkimage/iMX8M/

$ cd ../
```

3. Get the DDR firmware and copy to imx-mkimage/iMX8M/ directory.

```
$ wget https://www.nxp.com/lgfiles/NMG/MAD/YOCTO/firmware-imx-8.11.bin

 $ chmod a+x firmware-imx-8.11.bin

$ ./firmware-imx-8.11.bin
enter "y" to accept EULA

$ cd firmware-imx-8.11.bin

$ cp firmware/ddr/synopsys/lpddr4_pmu_train_1d_dmem_202006.bin ../imx-mkimage/iMX8M/
$ cp firmware/ddr/synopsys/lpddr4_pmu_train_1d_imem_202006.bin ../imx-mkimage/iMX8M/
$ cp firmware/ddr/synopsys/lpddr4_pmu_train_2d_dmem_202006.bin ../imx-mkimage/iMX8M/
$ cp firmware/ddr/synopsys/lpddr4_pmu_train_2d_imem_202006.bin ../imx-mkimage/iMX8M/

$ cp firmware/hdmi/cadence/signed_hdmi_imx8m.bin ../imx-mkimage/iMX8M/

$ cd ../
```

4. Clone the U-Boot source code from Embedian Git Server and copy related files to *imx-mkimage/iMX8M/* directory.

Download:

For u-boot v2020.04:

```
$ git clone git@git.embedian.com:developer/smarc-t335x-uboot.git u-boot_v2020.04 -b pitx_8mp_lf_v2020.04

$ cd u-boot_v2020.04
```

Configure and Build:

```
$ make ARCH=arm CROSS_COMPILE=${CC} distclean

$ make ARCH=arm CROSS_COMPILE=${CC} pitximx8mp_4g_defconfig

$ make ARCH=arm CROSS_COMPILE=${CC}
```

Note 1:

If the board is 2GB LPDDR4, use
`$ make ARCH=arm CROSS_COMPILE=${CC} pitximx8mp_2g_defconfig`

If the board is 4GB LPDDR4, use
`$ make ARCH=arm CROSS_COMPILE=${CC} pitximx8mp_4g_defconfig`

If the board is 6GB LPDDR4, use
`$ make ARCH=arm CROSS_COMPILE=${CC} pitximx8mp_6g_defconfig`

Note 2:

The *pITX-MX8M-PLUS* module always boot up from the on-module *eMMC* flash. The factory default will be *flash.bin* pre-installed. In some cases when the *eMMC* flash is empty or needs to be upgraded. Users can set SW2 port 1-3 as (ON ON ON). In this way, the *pITX-MX8M-PLUS* module will boot up to carrier microSD card. The *flash.bin* image for eMMC and microSD card is the same, the difference is how you flash *flash.bin* and how SW2 port 1-3 is set. This will be explained in the "Setup microSD card" section.

Copy `u-boot-nodtb.bin spl/u-boot-spl.bin arch/arm/dts/imx8mp-pitx.dtb` to *`imx-mkimage/iMX8M`* directory and copy *`tools/mkimage`* to *`imx-mkimage/iMX8M/mkimage_uboot`*

```
$ cp u-boot-nodtb.bin spl/u-boot-spl.bin arch/arm/dts/imx8mp-pitx.dtb ../imx-mkimage/iMX8M/

$ cp tools/mkimage ../imx-mkimage/iMX8M/mkimage_uboot
```

5. Generate flash.bin file.

```
$ cd ../imx-mkimage
$ make CROSS_COMPILE=${CC} SOC=iMX8MP clean

$ make CROSS_COMPILE=${CC} SOC=iMX8MP flash_evk
```

The *flash.bin* file will be located at *imx-mkimage/iMX8M* directory. Go to "Setup microSD Card" section to instruct you how to flash this file into microSD card.

Linux Kernel

Download:

For 5.10.9 (Based on NXP lf-5.10.y official release):

```
$ git clone git@git.embedian.com:developer/smarc-fsl-linux-kernel.git linux_v5.10.9 -b
pitx_8mp_lf-5.10.y

$ cd linux_v5.10.9
```

Configure and Build:

```
$ make ARCH=arm64 CROSS_COMPILE=${CC} distclean

$ make ARCH=arm64 CROSS_COMPILE=${CC} pitxmx8mplus_defconfig

$ make ARCH=arm64 CROSS_COMPILE=${CC} Image modules dtbs
```

Selecting display configuration is a matter of selecting an appropriate DTB file under `arch/arm64/boot/dts/embedian`.

All available DTB files are listed in the table below.

DTB File Name	Description
`imx8mp-pitx.dtb`	Device tree blob for no display configuration.
`imx8mp-pitx-hdmi.dtb`	Device tree blob for HDMI display configuration.
`imx8mp-pitx-lvds.dtb`	Device tree blob for LVDS display configuration.
`imx8mp-pitx-m7.dtb`	Device tree blob for Cortex-M7 co-processor configuration.

Root File System

Debian 11 Bullseys:

User	Password
root	root

Debian 11 Bullseye Download:

```
$ wget -c ftp://ftp.embedian.com/public/dev/minfs/ubuntu/bionic/ubuntu-minimal-imx8m-18.04-arm64.tar.gz
```

Verify:

```
$ md5sum ubuntu-minimal-imx8m-18.04-arm64.tar.gz
50afb3331998473d505693eabd031b9e  ubuntu-minimal-imx8m-18.04-arm64.tar.gz
```

Yocto Build Root File System:

User	Password
root	N/A

Find the yocto pre-built root file systems here at Embedian's ftp site based on your module CPU variants.

Setup microSD Card

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

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

Erase microSD card:

```
$ sudo dd if=/dev/zero of=${DISK} bs=1M count=160
```

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

If on-module eMMC Flash is empty

In some cases, when eMMC flash is erased or the u-boot is under development, we need a way to boot from microSD card first. Users need to ser SW2 port 1-3 and (ON ON ON). In this way, *pITX-iMX8M-PLUS* will always boot up from microSD card.

Fuse flash.bin to the microSD card.

~/imx-mkimage

```
$ sudo dd if=iMX8M/flash.bin of=${DISK} bs=1024 seek=32
```

If on-module eMMC Flash is not empty

The *flash.bin* is pre-installed in on-module eMMC flash at factory default. *pITX-MX8M-PLUS* is designed to always boot up from on-module eMMC flash and to load Image, device tree blob and root file systems based on the setting of SW2 port 1-3. If users need to fuse your own flash.bin or perform u-boot upgrade. This section will instruct you how to do that.

Copy flash.bin to the second partition home directory of your microSD card and boot into microSD card. Go to home directory and you should see flash.bin file (The flash.bin file is located at *imx-mkimage/iMX8M/* directory).

```
$ sudo cp -v imx-mkimage/iMX8M/flash.bin /media/rootfs/home/root/
```

Fuse flash.bin to the on-module eMMC flash. (The eMMC flash is emulated as /dev/mmcblk2 in *pITX-MX8M-PLUS*)

home directory

```
$ sudo dd if=flash.bin of=/dev/mmcblk2 bs=1024 seek=32
```


1. If your u-boot hasn't been finalized and still under development, it is recommended to set SW2 port 1-3 as (ON ON ON) and boot directly from microSD card first. Once your u-boot is fully tested and finalized, you can fuse your flash.bin to eMMC flash.
2. When SW2 port1-3 on *pITX-MX8M-PLUS* is set as (OFF ON OFF), it will always boot up from on-module eMMC flash. U-boot will read the SW2 port 1-3 configuration and determine where it should load Image and device tree blob. When SW2 port 1-3 is set (ON ON ON), it will always boot up from microSD card.

uEnv.txt based bootscript

Create "uEnv.txt" boot script: ($ vim uEnv.txt)

~/uEnv.txt

```
optargs="video=HDMI-A-1:1920x1080-32@60 consoleblank=0"
#optargs="video=HDMI-A-1:3840x2160-32@30 consoleblank=0"
#optargs="video=HDMI-A-1:3840x2160-32@60 consoleblank=0"
console=ttymxc3,115200 earlycon=ec_imx6q,0x30a60000,115200
mmcdev=1
mmcpart=1
image=Image
loadaddr=0x40480000
fdt_addr=0x43000000
mmcroot=/dev/mmcblk1p2 rw
usbroot=/dev/sda2 rw
mmcrootfstype=ext4 rootwait fixrtc
netdev=eth0
ethact=FEC0
ipaddr=192.168.1.150
serverip=192.168.1.53
gatewayip=192.168.1.254
mmcargs=setenv bootargs console=${console} root=${mmcroot} rootfstype=${mmcrootfstype} ${optargs}
uenvcmd=run loadimage; run loadfdt; run mmcboot
# USB Boot
#usbargs=setenv bootargs console=${console} root=${usbroot} rootfstype=${mmcrootfstype} ${optargs}
#uenvcmd=run loadusbimage; run loadusbfdt; run usbboot
```

Copy uEnv.txt to the boot partition:

~/

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

Install Kernel Image

Copy Image to the boot partition:

~/linux_v5.10.9

```
$ sudo cp -v arch/arm64/boot/Image /media/boot
```

Install Kernel Device Tree Binary

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

$ sudo cp -v arch/arm64/boot/dts/embedian/<device tree name> /media/boot/dtbs/imx8mp-pitx.dtb
```

All available DTB files are listed in the table below.

DTB File Name	Description
imx8mp-pitx.dtb	Device tree blob for no display configuration.
imx8mp-pitx-hdmi.dtb	Device tree blob for HDMI display configuration.
imx8mp-pitx-lvds.dtb	Device tree blob for LVDS display configuration.
imx8mp-pitx-m7.dtb	Device tree blob for Cortex-M7 co-processor configuration.

The device tree name in your microSD card has be to *imx8mp-pitx.dtb*

Install Root File System and Kernel Modules

Copy Root File System:

Yocto Pre-Built Rootfs:

directory where your root file system is

```
$ sudo tar jxvf <filename.tar.gz> -C /media/rootfs
```

Debian 11 Bullseys:

directory where your root file system is

```
$ sudo tar xvfz ubuntu-minimal-imx8m-18.04-arm64.tar.gz -C /media/rootfs
```

Copy Kernel Modules:

~/linux_v5.4.24

```
$ sudo make ARCH=arm64 CROSS_COMPILE=${CC} INSTALL_MOD_PATH=/media/rootfs modules_install
```

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. If your rootfs is yocto built, the kernel modules will be included in the rootfs. No need to copy again.

Networking:

Edit: /etc/network/interfaces

```
$ sudo vim /media/rootfs/etc/network/interfaces
```

Add:

/media/rootfs/etc/network/interfaces

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
```

Remove microSD card:

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

Setup eMMC

Setting up eMMC usually is the last step at development stage after the development work is done at your microSD card or NFS environments. From software point of view, eMMC is nothing but a non-removable SD card on board. For *pITX-MX8M-PLUS*, the microSD card is always emulated as /dev/mmcblk1 and on-module eMMC is always emulated as /dev/mmcblk2. Setting up eMMC now is nothing but changing the device descriptor.

This section gives a step-by-step procedure to setup eMMC flash. Users can write a shell script your own at production to simplify the steps.

First, we need to backup the final firmware from your microSD card or NFS.

Prepare for eMMC binaries from microSD card (or NFS):

Insert SD card into your Linux PC. For these instructions, we are assuming: DISK=/dev/mmcblk0, "lsblk" is very useful for determining the device id.

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

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

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/
```

Copy Image to rootfs partition:

```
$ sudo cp -v /media/boot/Image /media/rootfs/home/root
```

> **Note**
>
> 1. If your rootfs is Debian Bullseye, copy to */media/rootfs/root/* instead of /media/rootfs/root

Copy uEnv.txt to rootfs partition:

Copy and paste the following contents to /media/rootfs/home/root ($ sudo vim /media/rootfs/home/root/uEnv.txt)

```
optargs="video=HDMI-A-1:1920x1080-32@60 consoleblank=0"
#optargs="video=HDMI-A-1:3840x2160-32@30 consoleblank=0"
#optargs="video=HDMI-A-1:3840x2160-32@60 consoleblank=0"
console=ttymxc3,115200 earlycon=ec_imx6q,0x30a60000,115200
mmcdev=2
mmcpart=1
image=Image
loadaddr=0x40480000
fdt_addr=0x43000000
mmcroot=/dev/mmcblk2p2 rw
usbroot=/dev/sda2 rw
mmcrootfstype=ext4 rootwait fixrtc
netdev=eth0
ethact=FEC0
ipaddr=192.168.1.150
serverip=192.168.1.53
gatewayip=192.168.1.254
mmcargs=setenv bootargs console=${console} root=${mmcroot} rootfstype=${mmcrootfstype} ${optargs}
uenvcmd=run loadimage; run loadfdt; run mmcboot
# USB Boot
#usbargs=setenv bootargs console=${console} root=${usbroot} rootfstype=${mmcrootfstype} ${optargs}
#uenvcmd=run loadusbimage; run loadusbfdt; run usbboot
```

Copy device tree blob to rootfs partition:

```
$ sudo cp -v /media/boot/dtbs/imx8mp-pitx.dtb /media/rootfs/home/root/imx8mp-pitx.dtb
```

Copy real rootfs to rootfs partition:

Yocto Built Root File Systems

```
$ pushd /media/rootfs
$ sudo tar cvfz ~/pitximx8mp-emmc-rootfs.tar.gz .
$ sudo mv ~/pitximx8mp-emmc-rootfs.tar.gz /media/rootfs/home/root
$ popd
```

Debian 11 Bullseye Root File Systems

```
$ sudo vim /media/rootfs/etc/udev/rules.d/70-persistent-net.rules
```

Delete all contents starting with "*SUBSYSTEM==*"

```
$ pushd /media/rootfs
$ sudo tar cvfz ~/pitximx8mp-emmc-rootfs.tar.gz .
$ sudo mv ~/pitximx8mp-emmc-rootfs.tar.gz /media/rootfs/root/
$ popd
```

Remove microSD card:

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

Copy Binaries to eMMC from microSD card:

Insert this microSD card into your *pITX-MX8M-PLUS* device and boot into microSD card.

Now it will be almost the same as you did when setup your microSD card, but the eMMC device descriptor is */dev/mmcblk2* now.

```
$ export DISK=/dev/mmcblk2
```

Erase eMMC:

```
$ sudo dd if=/dev/zero of=${DISK} bs=2M count=16
```

Create Partition Layout:

```
$ sudo sfdisk ${DISK} <<-__EOF__
 2M,48M,0x83,*
 50M,,,
__EOF__
```

Format Partitions:

```
$ sudo mkfs.vfat -F 16 ${DISK}p1 -n boot
$ sudo mkfs.ext4 ${DISK}p2 -L rootfs
```

Mount Partitions:

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

```
$ sudo mount ${DISK}p1 /media/boot/
$ sudo mount ${DISK}p2 /media/rootfs/
```

Install binaries for partition 1

Copy uEnv.txt/Image/*.dtb to the boot partition

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

Install Kernel Device Tree Binary

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

$ sudo cp -v imx8mp-pitx.dtb /media/boot/dtbs/
```

Setup eMMC

```
$ sudo tar -zxvf pitximx8mp-emmc-rootfs.tar.gz -C /media/rootfs
```

Unmount eMMC:

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

Switch your Boot Select to eMMC and you will be able to boot up from eMMC now.

---

version 1.0a, 08/18/2022

Last updated 2022-08-18