Flashing the Compute Module eMMC

The Compute Module has an on-board eMMC device connected to the primary SD card interface. This guide explains how to write data to the eMMC storage using a Compute Module IO board.

Please also read the section in the Compute Module Datasheets

Steps to flash the eMMC on a Compute Module

To flash the Compute Module eMMC, you either need a Linux system (a Raspberry Pi is recommended, or Ubuntu on a PC) or a Windows system (Windows 10 is recommended). For BCM2837 (CM3), a bug which affected the Mac has been fixed, so this will also work.

Note There is a bug in the BCM2835 (CM1) bootloader which returns a slightly incorrect USB packet to the host. Most USB hosts seem to ignore this benign bug and work fine; we do, however, see some USB ports that don't work due to this bug. We don't quite understand why some ports fail, as it doesn't seem to be correlated with whether they are USB2 or USB3 (we have seen both types working), but it's likely to be specific to the host controller and driver. This bug has been fixed in BCM2837.

For Windows users

Under Windows, an installer is available to install the required drivers and boot tool automatically. Alternatively, a user can compile and run it using Cygwin and/or install the drivers manually.

Windows installer

For those who just want to enable the Compute Module eMMC as a mass storage device under Windows, the stand-alone installer is the recommended option. This installer has been tested on Windows 10 32-bit and 64-bit, and Windows XP 32-bit.

Please ensure you are not writing to any USB devices whilst the installer is running.

1. Download and run the Windows installer to install the drivers and boot tool.
2. Plug your host PC USB into the CMIO USB SLAVE port, making sure J4 is set to the EN position.

3. Apply power to the CMIO board; Windows should now find the hardware and install the driver.
4. Once the driver installation is complete, run the `RPiBoot.exe` tool that was previously installed.
5. After a few seconds, the Compute Module eMMC will pop up under Windows as a disk (USB mass storage device).

Setting up the Compute Module IO board

Compute Module 4

Ensure the Compute Module is fitted correctly installed on the IO board. It should lie flat on the IO board.

- Make sure that `nRPI_BOOT` which is J2 (`disable eMMC Boot`) on the IO board is set to the 'EN' position.
- Use a micro USB cable to connect the micro USB slave port J11 on IO board to the host device.
- Do not power up yet.

Compute Module 1..3

Ensure the Compute Module itself is correctly installed on the IO board. It should lie parallel with the board, with the engagement clips clicked into place.

- Make sure that J4 (USB SLAVE BOOT ENABLE) is set to the 'EN' position.
- Use a micro USB cable to connect the micro USB slave port J15 on IO board to the host device.
- Do not power up yet.

Building rpiboot on your host system (Cygwin/Linux)

We will be using Git to get the rpiboot source code, so ensure Git is installed. In Cygwin, use the Cygwin installer. On a Pi or other Debian-based Linux machine, use the following command:

```
sudo apt install git
```

Git may produce an error if the date is not set correctly. On a Raspberry Pi, enter the following to correct this:

```
sudo date MMDDhhmm
```

where `MM` is the month, `DD` is the date, and `hh` and `mm` are hours and minutes respectively.

Clone the `usbboot` tool repository:

```
git clone --depth=1 https://github.com/raspberrypi/usbboot
cd usbboot
```

`libusb` must be installed. If you are using Cygwin, please make sure `libusb` is installed as previously described. On Raspberry Pi OS or other Debian-based Linux, enter the following command:

```
sudo apt install libusb-1.0-0-dev
```

Now build and install the `usbboot` tool:

```
make
```

Run the `usbboot` tool and it will wait for a connection:

```
sudo ./rpiboot
```

Now plug the host machine into the Compute Module IO board USB slave port and power the CMIO board on. The `rpiboot` tool will discover the Compute Module and send boot code to allow access to the eMMC.

For more information run

```
./rpiboot -h
```

Writing to the eMMC - Windows

After `rpiboot` completes, a new USB mass storage drive will appear in Windows. We recommend following this guide and using Win32DiskImager to write images to the drive, rather than trying to use `/dev/sda` etc. from Cygwin.

Make sure J4 (USB SLAVE BOOT ENABLE) / J2 (nRPI_BOOT) is set to the disabled position and/or nothing is plugged into the USB slave port. Power cycling the IO board should now result in the Compute Module booting from eMMC.

Writing to the eMMC - Linux

After `rpiboot` completes, you will see a new device appear; this is commonly `/dev/sda` on a Pi but it could be another location such as `/dev/sdb`, so check in `/dev/` or run `lsblk` before running `rpiboot` so you can see what changes.

You now need to write a raw OS image (such as Raspberry Pi OS) to the device. Note the following command may take some time to complete, depending on the size of the image: (Change `/dev/sdX` to the appropriate device.)

```
sudo dd if=raw_os_image_of_your_choice.img of=/dev/sdX bs=4MiB
```

Once the image has been written, unplug and re-plug the USB; you should see two partitions appear (for Raspberry Pi OS) in `/dev`. In total, you should see something similar to this:

```
/dev/sdX    <- Device
/dev/sdX1   <- First partition (FAT)
/dev/sdX2   <- Second partition (Linux filesystem)
```

The `/dev/sdX1` and `/dev/sdX2` partitions can now be mounted normally.

Make sure J4 (USB SLAVE BOOT ENABLE) / J2 (nRPI_BOOT) is set to the disabled position and/or nothing is plugged into the USB slave port. Power cycling the IO board should now result in the Compute Module booting from eMMC.

Flashing the bootloader EEPROM - Compute Module 4

The `rpiboot` tool is the recommended method for updating the bootloader EEPROM on Compute Module 4. After following the initial EMMC flashing setup steps run the following command to run the `recovery` image instead of the EMMC image.

```
./rpiboot -d recovery
```

The `recovery` directory of the `rpiboot` tool contains a default `pieeprom.bin` bootloader EEPROM image. See the boot EEPROM and bootloader configuration pages for more information about how to change the embedded configuration file.

The SHA256 checksum file must match the `pieeprom.bin` image. To generate the `.sig` file run

```
sha256sum pieeprom.bin | awk '{print $2}' > pieeprom.sig
```

The bootloader image in the `recovery` directory is the latest manufacturing image with default settings. It is intended for use on a Compute Module 4 IO board with Raspberry Pi OS booting from SD/EMMC as a Compute Module 4 development platform.

Troubleshooting

For a small percentage of Raspberry Pi Compute Module 3s, booting problems have been reported. We have traced these back to the method used to create the FAT32 partition; we believe the problem is due to a difference in timing between the BCM2835/6/7 and the newer eMMC devices. The following method of creating the partition is a reliable solution in our hands.

```
$ sudo parted /dev/<device>
(parted) mkpart primary fat32 4MiB 64MiB
(parted) q
$ sudo mkfs.vfat -F32 /dev/<device>
$ sudo cp -r <files>/* <mountpoint>
```