Raspberry Pi

COMPUTE MODULE OPERATING INSTRUCTIONS

POWERING THE MODULE

- The Compute Module has 6 separate supplies that must be present at all times (i.e. they must all be powered, you cannot leave any of them unpowered even if a specific interface or GPIO bank is unused).

1. VBAT is used to power the BCM2835 processor core (it feeds the SMPS that generates the chip core voltage)
2. 3V3 powers various BCM2835 PHYs, IO and the eMMC Flash
3. 1V8 powers various BCM2835 PHYs, IO and SDRAM
4. VDAC powers the composite (TV out) DAC
5. GPIO0-27_VREF powers the GPIO 0-27 IO bank
6. GPIO28-45_VREF powers the GPIO 28-45 IO bank

SUPPLY	VOLTAGE / VOLTAGE RANGE	TOLERANCE
VBAT	2.3-5V [1]	+/- 5%
3V3	3.3V	+/- 5%
1V8	1.8V	+/- 5%
VDAC	2.5-2.8V (can connect to 3V3 if unused)	+/- 5%
GPIO0-27_VREF	1.8-3.3V	+/- 5%
GPIO28-45_VREF	1.8-3.3V	+/- 5%

[1] ****NOTE** that the voltage range for best SMPS efficiency is ~3.3-4.3V

POWER SEQUENCING

- Supplies must be synchronised to come up at exactly the same time, or staggered such that the highest voltage comes up first, then the next highest etc. This is to avoid forward biasing internal (on-chip) diodes between supplies and causing latch-up.

POWER REQUIREMENTS

- Exact power requirements will be heavily dependent upon the individual use case. If an on-chip subsystem is unused it is usually in a low power state or completely turned off. For instance if your application does not use 3D graphics then a large part of the core digital logic will never turn on and need power. Similarly for camera / display interfaces, HDMI, USB interfaces, video encode/decode etc.

- Powerchain design is critical for stable and reliable operation of the Compute Module. We strongly recommend that designers spend time measuring and verifying power requirements for their particular use case and application, as well as paying careful attention to power supply sequencing and maximum supply voltage tolerance.

- The following table gives a rough guide as to minimum supply requirements. **However, the user is responsible for verifying that their power-chain is designed sufficient for their application. In some more pathological use cases these minimum requirements may well be too low!**

SUPPLY	MINIMUM REQUIREMENT (MA OR MW)
VBAT	2000mW [1]
3V3	250mA
1V8	250mA
VDAC	25mA
GPIO0-27_VREF	See note [2]
GPIO28-45_VREF	See note [2]

[1] ****NOTE** VBAT is heavily dependent upon the application. E.g. with video encode, 3D and camera all running the power requirements can be substantial.

[2] ****NOTE** that each GPIO bank will only need a few mW if unused, however when in use the requirements will vary depending on number of IOs in use, and the load on each. The designer is responsible for calculating or measuring this themselves based on their particular use case.

MODULE BOOTING / FLASHING THE EMMC

- The 4GByte eMMC Flash device on the Compute module is directly connected to the primary BCM28235 SD/eMMC interface. These connections are not accessible on the module pins.

- When initially powered on (or after the RUN pin has been held low and then released) the BCM2835 will try and access the eMMC device and look for a file called bootcode.bin on the primary partition (which must be FAT) to start booting the system. If it cannot access the eMMC device or the boot code cannot be found, it will fall back to waiting for boot code to be written to it over USB (i.e. its USB port is in slave mode waiting to accept boot code from a suitable host).

- A USB boot tool is available (https://github.com/raspberrypi/tools/tree/master/usbboot) which allows a host PC running Linux to write the BCM2835 boot code over USB to the module. That boot code then runs and provides access to the eMMC as a USB mass storage device, which can then be read/written using the host PC. (Note that a Raspberry Pi can be used as the host machine).

- The compute module has a pin called EMMC_DISABLE_N which when shorted to GND will disable the eMMC, forcing BCM2835 to boot from USB. Note that when the eMMC is disabled like this it takes a few seconds from power up for the processor to give up trying to talk to the eMMC device and fall back to booting from USB.

****NOTE** that once booted over USB, BCM2835 needs to re-enable the eMMC device (by releasing EMMC_DISABLE_N) to allow access to it as mass storage. It expects to be able to do this by driving the GPIO47_1V8 pin LOW (which at boot is initially an input with a pull up to 1V8). If an end user wishes to add the ability to access the eMMC over USB in their product, similar circuitry to that used on the Compute Module IO Board to enable/disable the USB boot and eMMC must be used (EMMC_DISABLE_N pulled low via MOSFET(s) and released again by MOSFET with gate controlled by GPIO47_1V8.) Be careful to use MOSFETs suitable for switching at 1.8V (i.e. use a device with Vt << 1.8V).**

STEPS TO FLASH THE EMMC ON A COMPUTE MODULE USING A COMPUTE MODULE IO BOARD

You need a host Linux system (a Raspberry Pi will do!).

****ON YOUR COMPUTE MODULE IO BOARD:****
Make sure that J4 (USB SLAVE BOOT ENABLE) is set to the 'EN' position.

****ON YOUR HOST SYSTEM:****
Git can get upset if the date is not set correctly so on a Raspberry Pi you can do a:

```
sudo date MMDDhhmm
```

Where MM is month, DD day and hh mm hours and minutes respectively.

- Clone the usbboot tool repository and grab libusb:

```
sudo git clone --depth=1 https://github.com/raspberrypi/tools
cd tools/usbboot
sudo apt-get install libusb-1.0-0-dev
```

- Build the usbboot tool:

```
sudo make
```

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

```
sudo ./rpiboot
```

- Now plug the host machine into Compute Module IO Board USB slave port (J15) and power on the CMIO board. The usbboot tool will see the compute module and send it boot code to allow access to the eMMC. Once complete you will see a new device appear, commonly /dev/sda
(but it could be /dev/sdb etc. so have a look in /dev/ before running rpiboot so you can see what changes).

- You now need to write a raw OS image (e.g. [Raspian](http://downloads.raspberrypi.org/raspbian_latest) to the device. Note the following command may take some time to complete (depends on the size of the image):

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

- Once the image has been written, unplug and re-plug the USB, and you should see 2 partitions appear (for Raspian) in /dev
In total you should see something like:

```
/dev/sda  ← Device
/dev/sda1 ← First partition (FAT)
/dev/sda2 ← Second partition (Linux filesystem)
```

The **/dev/sda1** and **/dev/sda2** partitions can now be mounted normally.

Make sure J4 (USB SLAVE BOOT ENABLE) 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.