MyPi Industrial CM4 Integrator Board

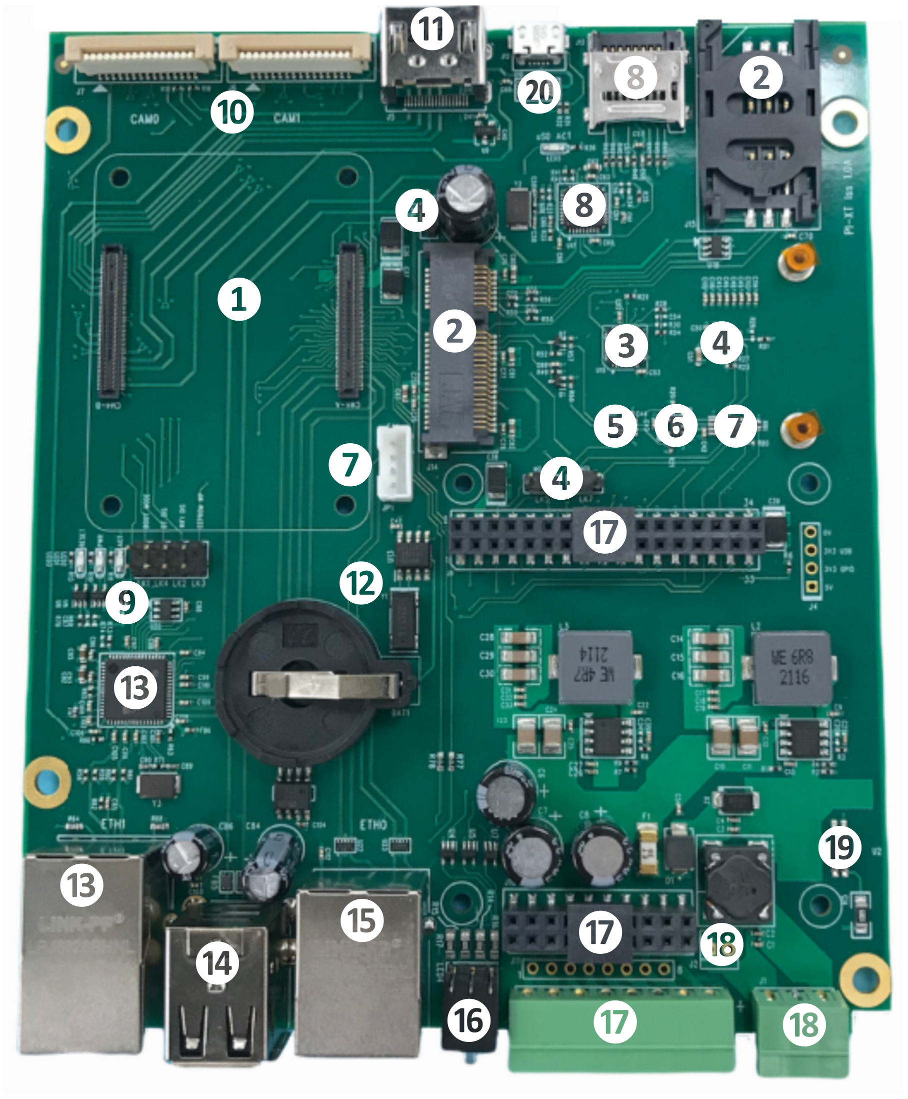
User Guide

Issue	:	2.1
Dated	:	June 2023
Prepared By	:	Andrew O'Connell

FEATURES

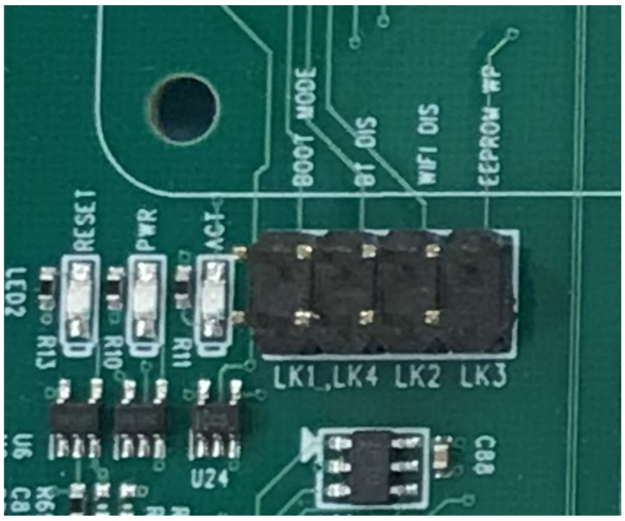
- Supports All Raspberry Pi Compute Module 4 variants
- 1 x 10/100/1000 LAN
- 1 x 10/100 LAN
- 2 x USB 2.0 (external access)
- 1 x uSD Card Storage (USB Interfaced)
- 1 x mPCIe Interface (USB Interfaced) + SIM
- 1 x SPI Infineon SLB9670 TPM 2.0
- 1 x Battery Backed RTC
- 1 x Board ID EEPROM (Preprogrammed)
- 2 x Camera Interfaces
- 1 x HDMI
- 1 x Opto-Isolated Digital Input
- 1 x Modular IO slot with 28 GPIO Pins
  - 3 x SPI
  - 5 x I2C
  - 4 x UART
  - SDIO Interfaces
  - 3 x GPCLK
  - 2 x PWM Channels
- 1 x Optional 60second watchdog (active from power up/boot)
- 1 x Temperature activated fan control (same as Pi CMIO4 board)
- 2 x Bi-colour user LEDs
- 9-28V Input
- Wide -20°C to +80°C Ambient operating temperature

BOARD IO FEATURES



❶	Compute Module 4 Socket	⓫	HDMI Out
❷	mPCIe Socket + Modem SIM Socket	⓬	I2C DS1338Z RTC + Coin Cell Backup Battery
❸	SPI Interfaced SLB9670 TPM 2.0	⓭	USB LAN9514 10/100 LAN + USB Interface
❹	60 Second Watchdog + Enable Links	⓮	2 x USB 2.0 Ports
❺	I2C GPIO Expander	⓯	Pi Gigabit 10/100/1000 Interface
❻	I2C Serial EEPROM	⓰	Dual Bi-colour LED
❼	I2C EMC2301 Fan Controller	⓱	GPIO IO Card interface + Front Connector
❽	USB µSD Card Interface + Socket	⓲	Power In (9-28V DC)
❾	Pi Status LEDs + Mode Links	⓳	Power In Digital Input
❿	Dual RPi Camera Sockets	⓴	µUSB CM4 Programming port

HARDWARE CONFIGURATION LINKS



LED - RESET

This LED indicates when the Pi unit is in reset condition and has asserted an external reset signal, which is routed to parts 3, 8 & 13

LED - PWR

This GPIO driven LED indicates 'power' functionality on a Raspberry Pi and can be repurposed for general usage, signal also connected the bottom red LED.

LED - ACT

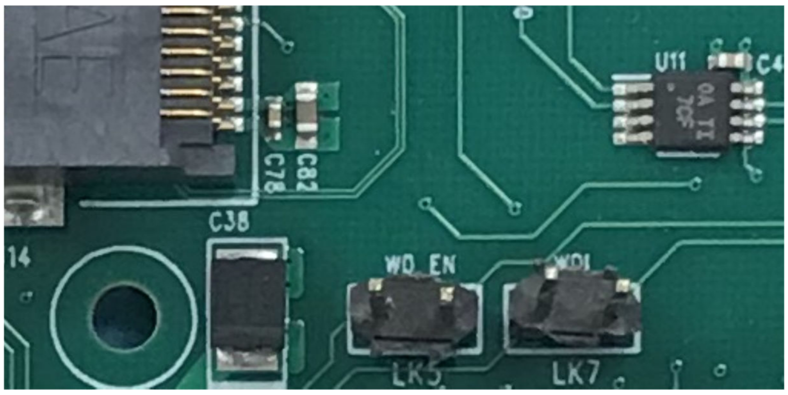
This LED indicates 'Activity' functionality on the Pi unit, by default this indicates eMMC flash access on the module, but can be reassigned to indicate other status signals.

LK1 -	**Boot Mode**
Fitted	Forces CM4 module into eMMC programming or EEPROM Firmware update mode
Open	Default, boots as normal according to EEPROM settings

LK4 -	**BT DIS**
Fitted	Forces CM4 module to disable Bluetooth RF Output
Open	Default

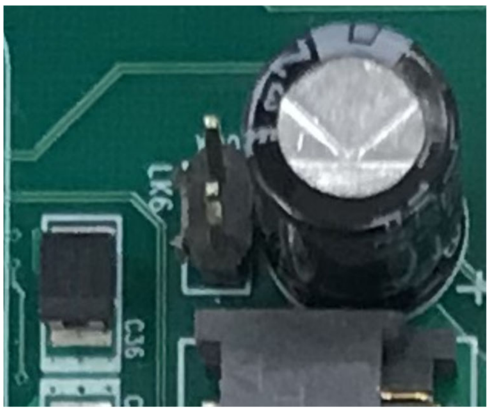
LK2 -	**WIFI DIS**
Fitted	Forces CM4 module to disable WiFi RF Output
Open	Default

LK3 -	**EEPROM DIS**
Fitted	Indicates to CM4 module to disable EEPROM *(Facility not enabled in firmware by default)*
Open	Default



LK5 -	**WD EN**
Fitted	Connect External Watchdog Enable Line to GPIO16
Open	Default

LK7 -	**WDI**
Fitted	Connect External Watchdog Input Line to GPIO17
Open	Default



LK6 -	**WATCHDOG RESET OUT**
Fitted	Connect External Watchdog Reset Out to CM4 RUN/RESET Line
Open	Default

RASPBERRY PI COMPUTE MODULE PROGRAMMING

The unit as shipped is configured to allow the eMMC flash on the compute module to be re-programmed without removing the PCB from the enclosure.

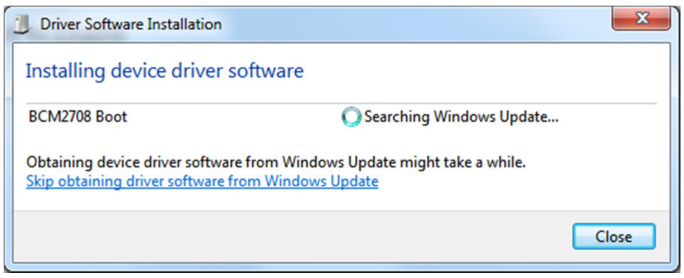
Units come pre-programmed with the demo Raspbian OS pre-installed, this section describes how to write a new disk image to the Compute Module.

First of all download the windows USB boot installer, this will install the device drivers as well as a program we'll use later called RPi-Boot

Raspberry Pi RPI-BOOT Software Download Link

Connect the mini USB connector to the Windows PC using the supplied USB A to micro USB B data cable, fit the boot mode jumper link (LK1) and then power up the unit.

Windows will then show the following stages as it configures the OS :

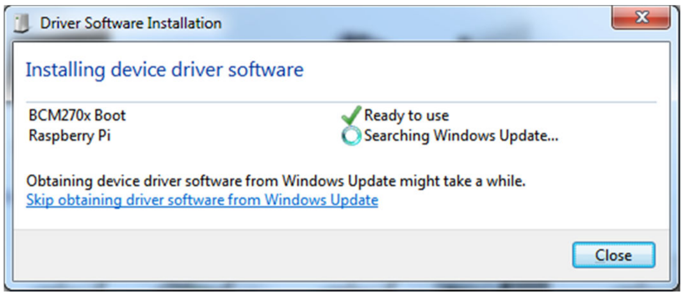


Driver Software Installation

Installing device driver software

BCM2708 Boot   Searching Windows Update...

Obtaining device driver software from Windows Update might take a while.
Skip obtaining driver software from Windows Update

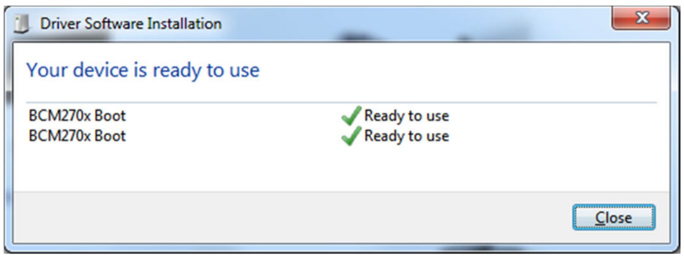


Driver Software Installation

Installing device driver software

BCM270x Boot   Ready to use
Raspberry Pi   Searching Windows Update...

Obtaining device driver software from Windows Update might take a while.
Skip obtaining driver software from Windows Update



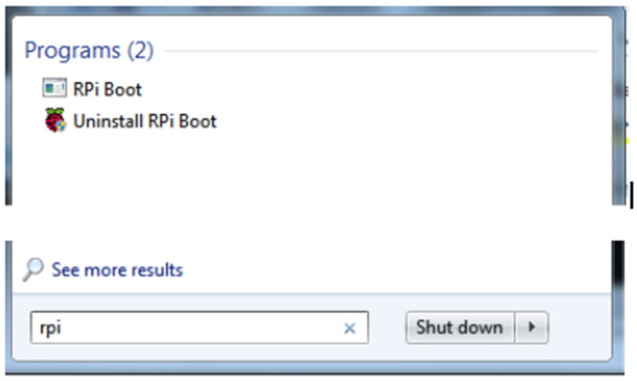
Driver Software Installation

Your device is ready to use

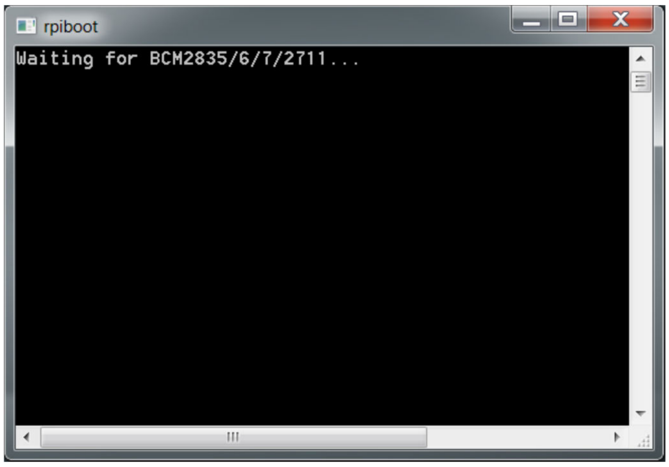
BCM270x Boot   Ready to use
BCM270x Boot   Ready to use

Once that sequence has finished Windows has now installed the required drivers and you can power off the unit for a moment whilst we get the PC side ready for the next step.

Making sure you have the unit powered off start up RPi Boot, this is easiest done via the start menu, we have found this needs to be run as ‘Administrator’ privilege mode for correct operation

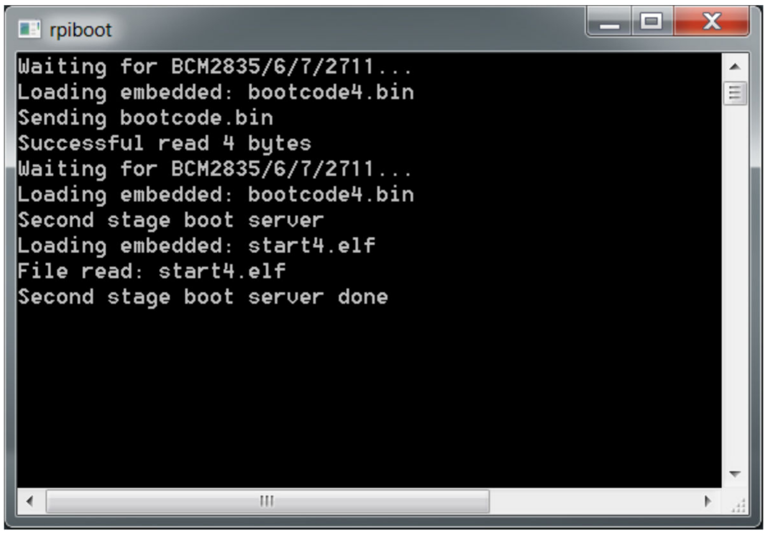


When the RPi-Boot starts up it’ll sit and wait for the attached board to boot up :



```
Waiting for BCM2835/6/7/2711...
```

Power up the unit and RPi-Boot will configure the unit to appear as a flash drive :

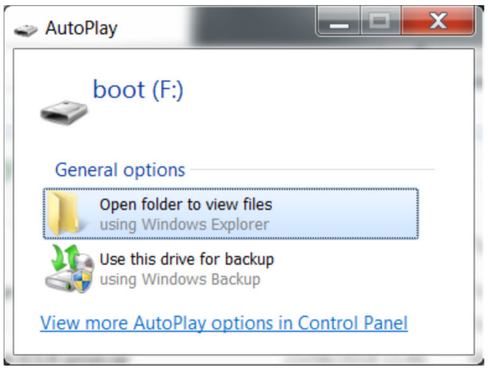


```
Waiting for BCM2835/6/7/2711...
Loading embedded: bootcode4.bin
Sending bootcode.bin
Successful read 4 bytes
Waiting for BCM2835/6/7/2711...
Loading embedded: bootcode4.bin
Second stage boot server
Loading embedded: start4.elf
File read: start4.elf
Second stage boot server done
```

When done the compute module will alternate into mass storage mode (so behaving just as though it's a USB memory stick) and windows will then recognise the module as an external drive.

If the compute module eMMC already contains an OS Windows will recognise the FAT partition and assign that (at least) a drive letter, this is useful in the event that a configuration error with the boot files is made (e.g. dt-blob.bin or config.txt) and needs recovery actions to be performed.

After drive letter assignment Windows may indicate that partitions need scanning or fixing, these can be ignored/cancelled.


AutoPlay
boot (F:)
General options
Open folder to view files
using Windows Explorer
Use this drive for backup
using Windows Backup
View more AutoPlay options in Control Panel

There are a few different ways we can load on the OS, for simplicity we'll cover using the recommended OS writing software and process from the main Raspberry Pi website

This process writes a disk image, containing the partition table as well as both FAT boot partition and Linux EXT partitions, ***over the entire disk.***

The basic sequence we're following is :

1. Download the Win32DiskImager utility from this **Download Link**
2. Install and run the Win32DiskImager utility (You will need to run the utility as administrator, right-click on the shortcut or exe file and select **Run as administrator**)
3. Select the OS image file you wish to write
4. Select the drive letter of the compute module in the device box (in our case F:) - Again note that the disk image is a 1:1 of the entire disk (containing the partition table, FAT & EXT partitions)

   Be careful to select the correct drive; if you get the wrong one you can destroy your data on the computer's hard disk!

5. Click **Write** and wait for the write to complete

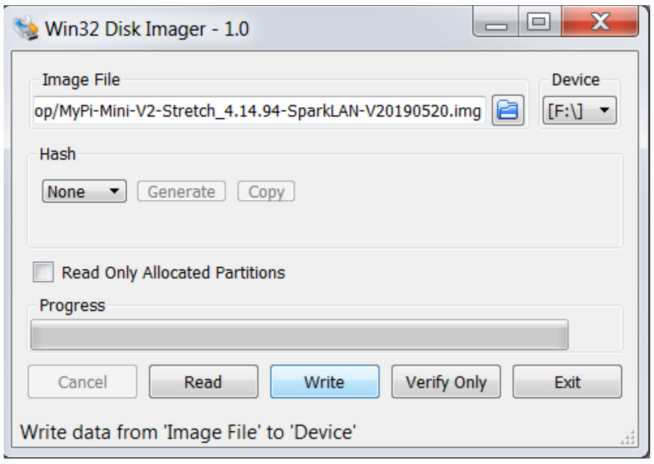

Win32 Disk Imager - 1.0

Image File: op/MyPi-Mini-V2-Stretch_4.14.94-SparkLAN-V20190520.img

Device: [F:\]

Hash: None

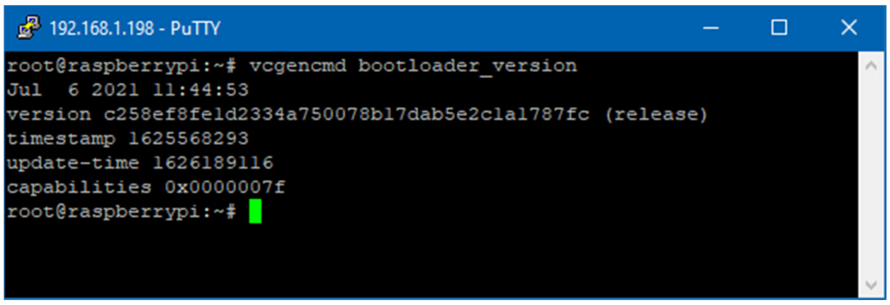
Write data from 'Image File' to 'Device'

CM4 BOOT EEPROM FIRMWARE UPDATE

On the CM4 it is not possible to update the boot firmware EEPROM from the command line

To find the current boot loader version run **vcgencmd bootloader_version**

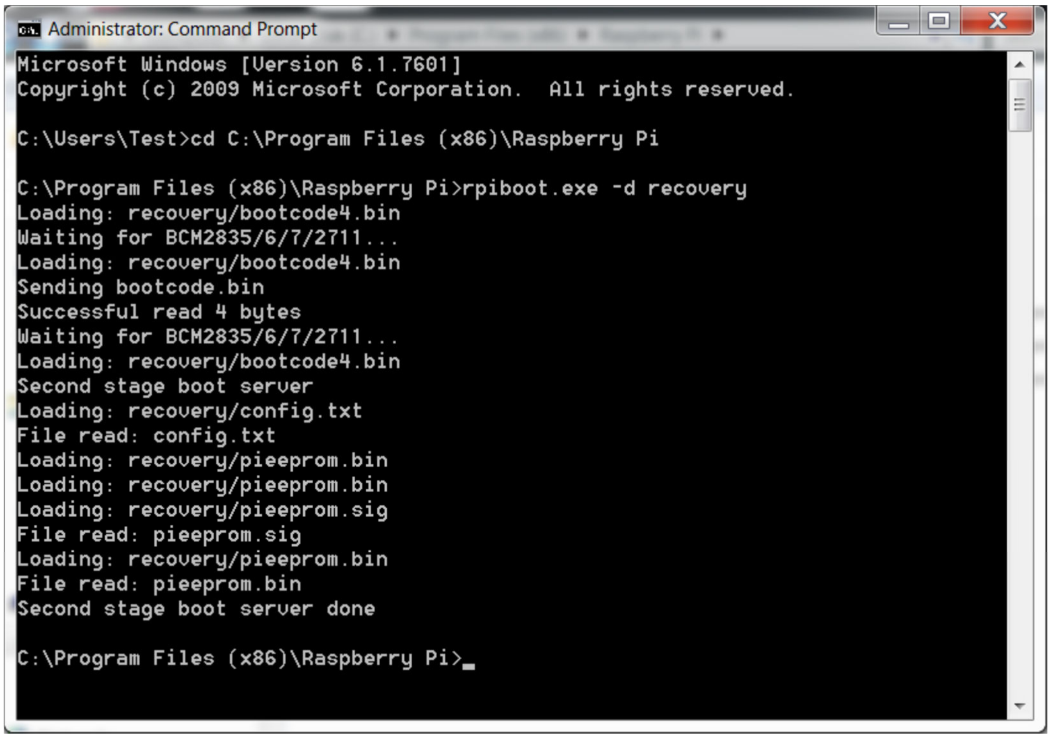
For best USB and Camera support we recommend installing version July 6th 2021 or later



```
root@raspberrypi:~# vcgencmd bootloader_version
Jul  6 2021 11:44:53
version c258ef8fe1d2334a750078b17dab5e2c1a1787fc (release)
timestamp 1625568293
update-time 1626189116
capabilities 0x0000007f
root@raspberrypi:~#
```

To update the firmware on a CM4 device the same **rpiboot** program is used, but with a different syntax to usual. This directs the system to push a different set of files to the CM4 device containing the firmware update files.

Fit the boot mode link and connect up the microUSB programming cable and run rpiboot as shown in the screenshot below. This process takes a short amount of time and once completed the HDMI output will indicate a green screen with rapidly blinking status LED to show success.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Test>cd C:\Program Files (x86)\Raspberry Pi

C:\Program Files (x86)\Raspberry Pi>rpiboot.exe -d recovery
Loading: recovery/bootcode4.bin
Waiting for BCM2835/6/7/2711...
Loading: recovery/bootcode4.bin
Sending bootcode.bin
Successful read 4 bytes
Waiting for BCM2835/6/7/2711...
Loading: recovery/bootcode4.bin
Second stage boot server
Loading: recovery/config.txt
File read: config.txt
Loading: recovery/pieeprom.bin
Loading: recovery/pieeprom.bin
Loading: recovery/pieeprom.sig
File read: pieeprom.sig
Loading: recovery/pieeprom.bin
File read: pieeprom.bin
Second stage boot server done

C:\Program Files (x86)\Raspberry Pi>_
```

To change/update the firmware version files of your install replace the 'recovery' folder on your local machine with that of the 'recovery' folder from the main github repo below

https://github.com/raspberrypi/usbboot

CM4 SECURE BOOT

Secure boot facilities on the CM4 are currently in beta release, see below website link for examples and notes on how to create a signed boot image

https://github.com/raspberrypi/usbboot#secure-boot---image-creation

See also TPM Hardware Security Module notes following

SYSTEM GPIO

In order to minimise CM4 GPIO line usage an I2C interfaced PCA9536 GPIO expander has been included

These present as **gpio500-503** on the OS and are configured via the system device tree overlays :

```
# I2C Bus (Note : Also need to add i2c-dev to /etc/modules)
dtparam=i2c_arm=on
dtoverlay=pca953x,addr=0x41,pca9536
```

A bash script **/etc/init.d/mypi-init.sh** which is called from **/etc/rc.local** during boot-up which creates the below symbolic links for quick access in **/dev**

```
mpcie-wdisble    -> /sys/class/gpio/gpio500/value
mpcie-reset      -> /sys/class/gpio/gpio501/value
led2-red         -> /sys/class/gpio/gpio502/value
led2-green       -> /sys/class/gpio/gpio503/value
led1-red         -> /sys/class/leds/led1/brightness
rtc_nvram        -> /sys/class/rtc/rtc0/device/nvram
wd-enable        -> /sys/class/gpio/gpio16/value
wd-input         -> /sys/class/gpio/gpio17/value
```

Example usage :

```
$ echo 1 >/dev/led1-red
$ echo 0 >/dev/led1-red

$ echo 1 >/dev/led2-green
$ echo 0 >/dev/led2-green


$ echo 1 >/dev/mpcie-reset
$ echo 0 >/dev/mpcie-reset

# echo 'battery backed up ram' > /dev/rtc_nvram
cat /dev/rtc_nvram
battery backed up ram
```

Board OS Configuration

The sample OS image provided has been produced by overlaying a series of files over a standard Raspberry Pi Lite OS Image. The configuration files can be downloaded using the tar file linked to below

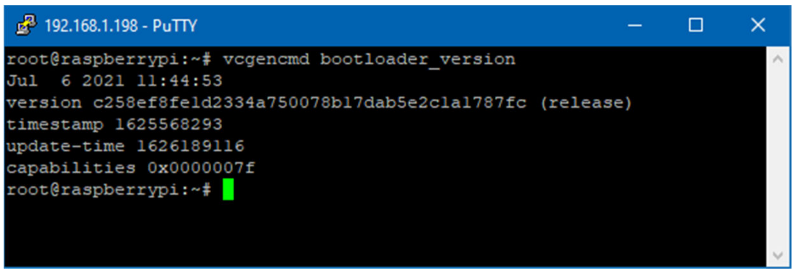
https://drive.google.com/file/d/1b8HxL6FkDFo-N_3SfSyCkyBcHYBUnngL/view?usp=sharing

USB INTERFACE

There are two critical settings that determine the USB controller active on the CM4, without these steps the system will either disable the USB port or enable the low bandwidth port controller.

1. Firmware version

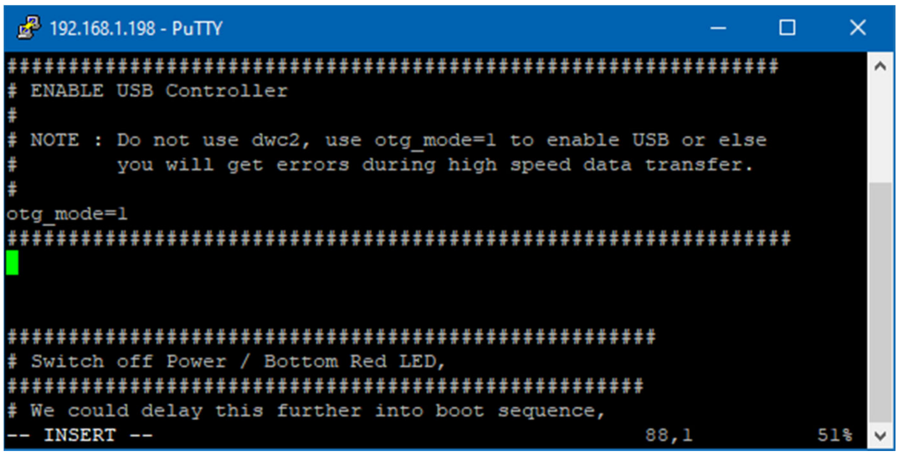
This should be version dated **July 6th 2021** or later, needed to enable the internal controller



```
root@raspberrypi:~# vcgencmd bootloader_version
Jul  6 2021 11:44:53
version c258ef8fe1d2334a750078b17dab5e2c1a1787fc (release)
timestamp 1625568293
update-time 1626189116
capabilities 0x0000007f
root@raspberrypi:~#
```

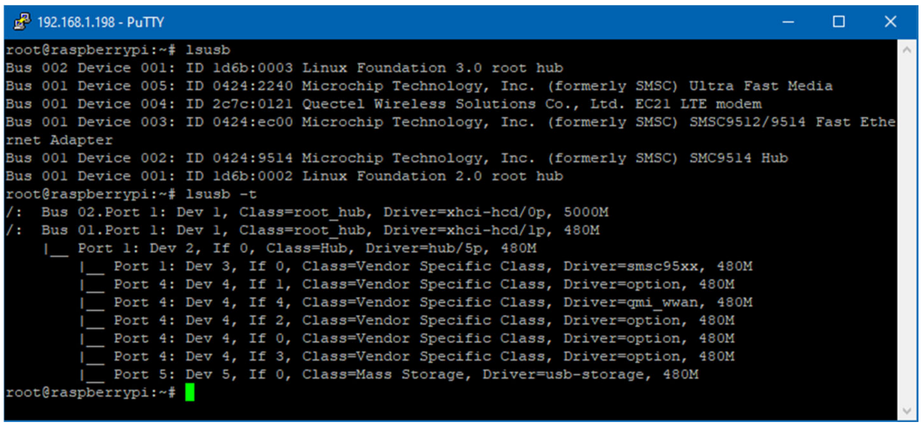
2. Config.txt setting

otg_mode=1 should be included in the configuration file to select the **xhci-hcd** controller



```
################################################################
# ENABLE USB Controller
#
# NOTE : Do not use dwc2, use otg_mode=1 to enable USB or else
#        you will get errors during high speed data transfer.
#
otg_mode=1
#################################################################



######################################################
# Switch off Power / Bottom Red LED,
#####################################################
# We could delay this further into boot sequence,
-- INSERT --                                        88,1          51%
```

Correctly setup the system will report the root hub as being an **xhci-hcd** device as shown below

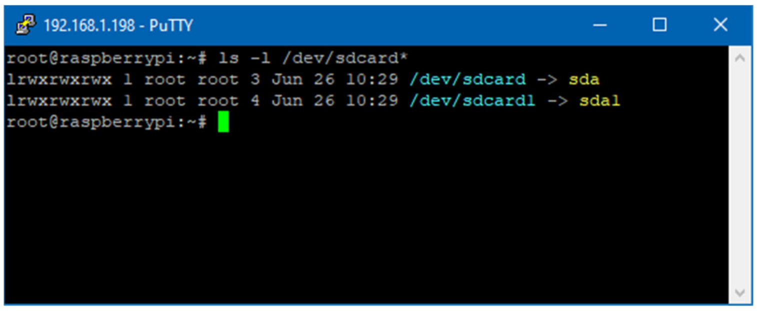


```
root@raspberrypi:~# lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 005: ID 0424:2240 Microchip Technology, Inc. (formerly SMSC) Ultra Fast Media
Bus 001 Device 004: ID 2c7c:0121 Quectel Wireless Solutions Co., Ltd. EC21 LTE modem
Bus 001 Device 003: ID 0424:ec00 Microchip Technology, Inc. (formerly SMSC) SMSC9512/9514 Fast Ethe
rnet Adapter
Bus 001 Device 002: ID 0424:9514 Microchip Technology, Inc. (formerly SMSC) SMC9514 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
root@raspberrypi:~# lsusb -t
/:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=xhci-hcd/0p, 5000M
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=xhci-hcd/1p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/5p, 480M
        |__ Port 1: Dev 3, If 0, Class=Vendor Specific Class, Driver=smsc95xx, 480M
        |__ Port 4: Dev 4, If 1, Class=Vendor Specific Class, Driver=option, 480M
        |__ Port 4: Dev 4, If 4, Class=Vendor Specific Class, Driver=qmi_wwan, 480M
        |__ Port 4: Dev 4, If 2, Class=Vendor Specific Class, Driver=option, 480M
        |__ Port 4: Dev 4, If 0, Class=Vendor Specific Class, Driver=option, 480M
        |__ Port 4: Dev 4, If 3, Class=Vendor Specific Class, Driver=option, 480M
        |__ Port 5: Dev 5, If 0, Class=Mass Storage, Driver=usb-storage, 480M
root@raspberrypi:~#
```

USB SD CARD INTERFACE

The on-board micro SD Card is interfaced to the Raspberry Pi Compute Module using on-board Microchip USB2240 SD card interface controller, this provides fast access to secondary storage for datalogging.

configuration file **/etc/udev/rules.d/8-sdcard.rules** creates the below **/dev** shortcuts for the main SD Card and any partitions contained



```
root@raspberrypi:~# ls -l /dev/sdcard*
lrwxrwxrwx 1 root root 3 Jun 26 10:29 /dev/sdcard -> sda
lrwxrwxrwx 1 root root 4 Jun 26 10:29 /dev/sdcard1 -> sda1
root@raspberrypi:~# 
```

This SD card cannot be booted from however can be auto mounted at boot (via **/etc/fstab**) so offers a low cost method of expanding the core eMMC filesystem

We recommend the use of industrial grade SD cards, which whist more expensive have greater operating temperature range, on-device wear-levelling and generally greater endurance than commercial grade parts.

For more information please see our knowledgebase article below

https://embeddedpi.com/documentation/sd-card-interface/raspberry-pi-industrial-micro-sd-cards

A hardware reset of the USB2240 device is asserted at reboot/power up by the Compute Module

USB 10/100 LAN + USB CONTROLLER

Integrated on-board is an Microchip LAN9514 device, this is connected to the Raspberry Pi via USB port and provides 4 additional downstream USB ports, two of which are used for the on-board mPCIe interface and USB2240 SD Card controller and the remaining two are brought out to the front face USB ports.

There are two scripts that are helpful:

/usr/local/bin/resetbyauthorized.sh

This script allows you to issue a software reset command to a USB peripheral by supplying the **vendorid** & **productid** identifiers

/usr/local/bin/usbpwrctl.sh

This script allows you to switch the power off/on to either/both of the front USB ports

A hardware reset of the LAN9514 device is asserted at reboot/power up by the Compute Module

USB MINI-PCIE INTERFACE

The Integrated mPCIe socket installed on the base board are wired to the below standard :

Pin	Signal	Pin	Signal
1	WAKE#	2	3.3V
3	Reserved	4	GND
5	Reserved	6	1.5V
7	CLKREQ#	8	SIM_VCC
9	GND	10	SIM_I/O
11	REFCLK-	12	SIM_CLK
13	REFCLK+	14	SIM_RST
15	N/C or GND	16	SIM_VPP
	Mechanical Key		
17	Reserved	18	GND
19	Reserved	20	W_DIS#
21	GND	22	PERST#
23	PERn0	24	+3.3Vaux
25	PERp0	26	GND
27	GND	28	+1.5V
29	GND	30	SMB_CLK
31	PETn0	32	SMB_DATA
33	PETp0	34	GND
35	GND	36	USB_D-
37	Reserved	38	USB_D+
39	+3.3V	40	GND
41	+3.3V	42	LED_WWAN#
43	GND	44	LED_WLAN#
45	Reserved	46	LED_WPAN#
47	Reserved	48	+1.5V
49	Reserved	50	GND
51	Reserved	52	+3.3V

Signals in RED are not available

The below GPIO connections are made to the connector

```
mpcie-reset      -> /sys/class/gpio/gpio500/value

mpcie-wdisble    -> /sys/class/gpio/gpio501/value
```

The WWAN LED is connected to the front top green bi-colour LED to indicate modem network registration/data transmission status.

Modem Compatibility/Operation

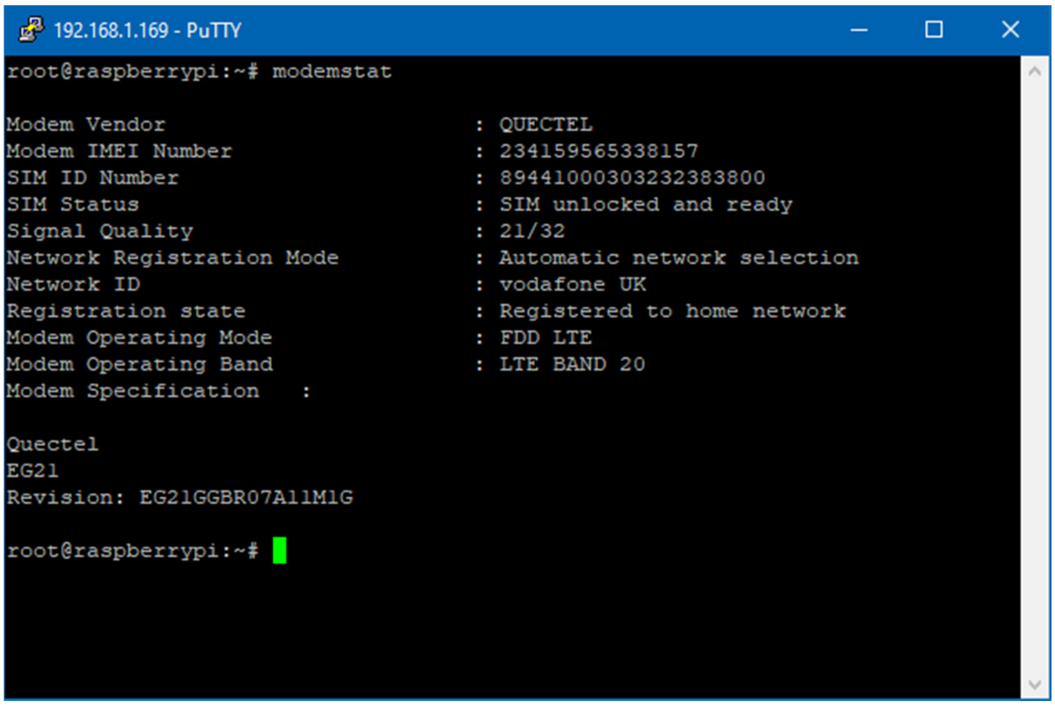
See the below link to pages from the main modem documentation section for details on how to operate modems :

http://www.embeddedpi.com/documentation/3g-4g-modems

The system has been pre-installed with modem helper status script **modemstat** which supports Sierra Wireless, Quectel and Simcom

See web page below for more details

https://embeddedpi.com/documentation/3g-4g-modems/mypi-industrial-raspberry-pi-3g-4g-modem-status

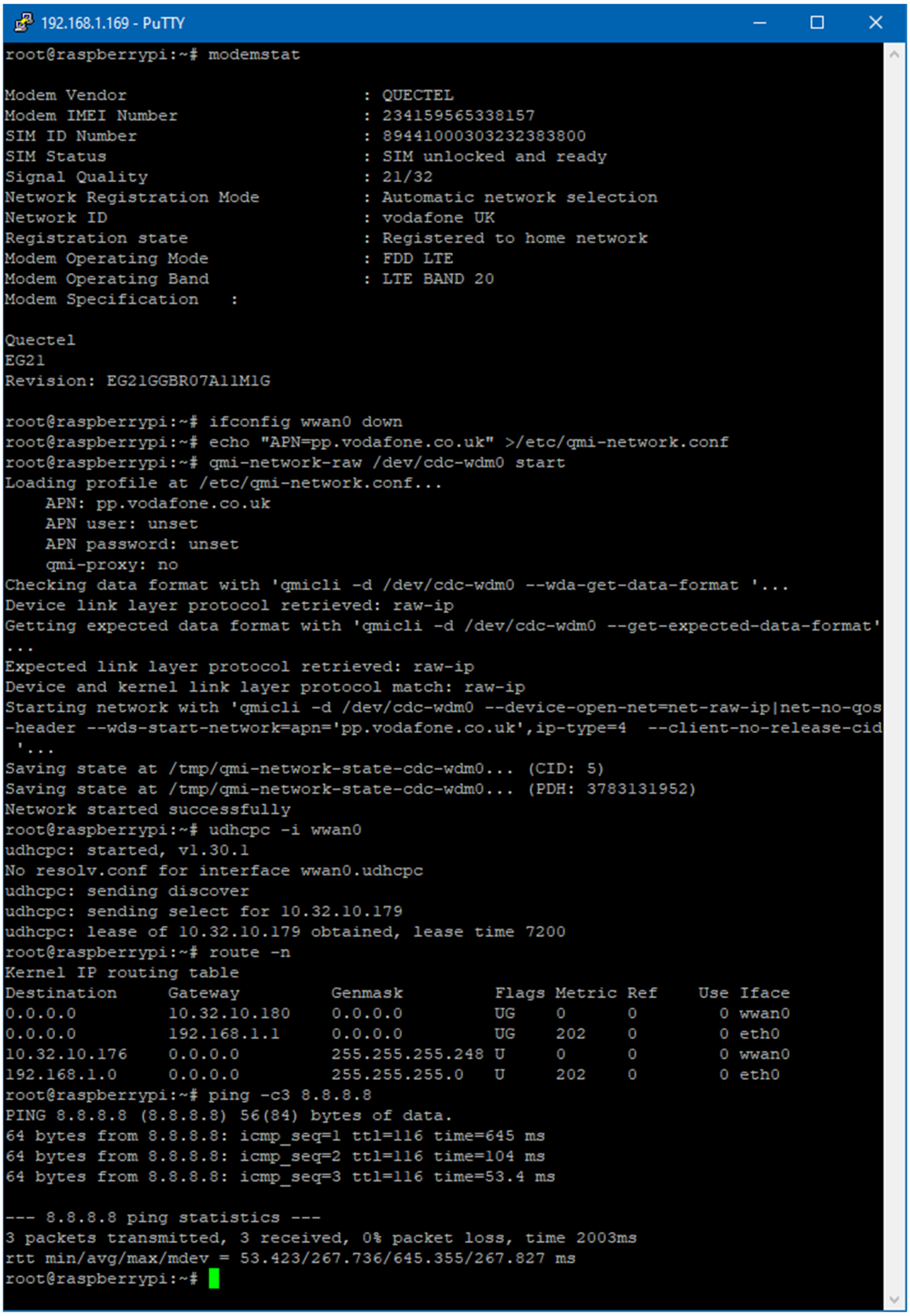


```
root@raspberrypi:~# modemstat

Modem Vendor                                : QUECTEL
Modem IMEI Number                           : 234159565338157
SIM ID Number                               : 89441000303232383800
SIM Status                                  : SIM unlocked and ready
Signal Quality                              : 21/32
Network Registration Mode                   : Automatic network selection
Network ID                                  : vodafone UK
Registration state                          : Registered to home network
Modem Operating Mode                        : FDD LTE
Modem Operating Band                        : LTE BAND 20
Modem Specification     :

Quectel
EG21
Revision: EG21GGBR07A11M1G

root@raspberrypi:~#
```

A number of udev rules have been added to provide consistent shortcut symbolic links for easy identification of the various ttyUSB interfaces created by the modem. These udev rule files are contained in the **/etc/udev/rules.d/modem-rules** folder.

Note that increasingly modems are requiring **raw ip** connection method to be implemented, to this end we have added **qmi-network-raw in /usr/local/bin** which makes this connection type easier along with **udhcp** which supports raw ip mode for obtaining an IP address once connection has been made.

QMI Network Connection example :



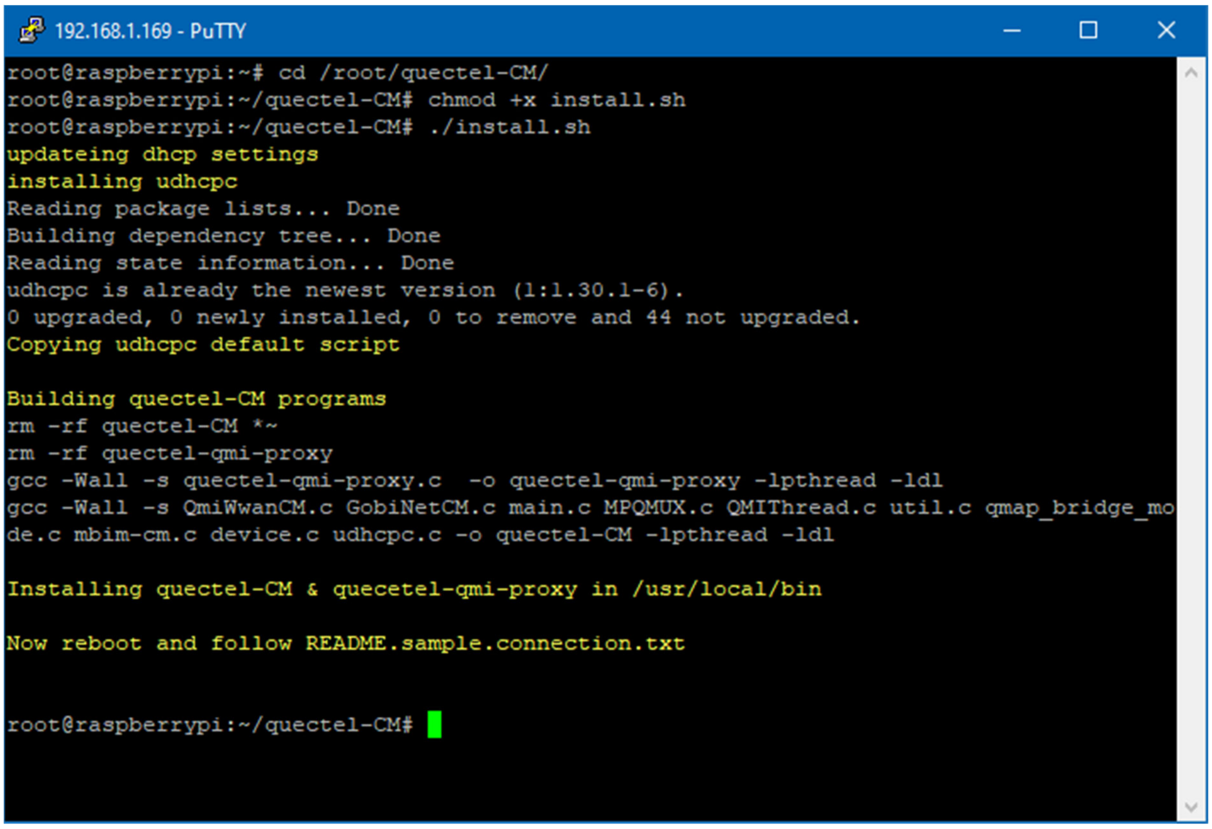
```
root@raspberrypi:~# modemstat

Modem Vendor                      : QUECTEL
Modem IMEI Number                 : 234159565338157
SIM ID Number                     : 89441000303232383800
SIM Status                        : SIM unlocked and ready
Signal Quality                    : 21/32
Network Registration Mode         : Automatic network selection
Network ID                        : vodafone UK
Registration state                : Registered to home network
Modem Operating Mode              : FDD LTE
Modem Operating Band              : LTE BAND 20
Modem Specification    :

Quectel
EG21
Revision: EG21GGBR07A11M1G

root@raspberrypi:~# ifconfig wwan0 down
root@raspberrypi:~# echo "APN=pp.vodafone.co.uk" >/etc/qmi-network.conf
root@raspberrypi:~# qmi-network-raw /dev/cdc-wdm0 start
Loading profile at /etc/qmi-network.conf...
    APN: pp.vodafone.co.uk
    APN user: unset
    APN password: unset
    qmi-proxy: no
Checking data format with 'qmicli -d /dev/cdc-wdm0 --wda-get-data-format '...
Device link layer protocol retrieved: raw-ip
Getting expected data format with 'qmicli -d /dev/cdc-wdm0 --get-expected-data-format'
...
Expected link layer protocol retrieved: raw-ip
Device and kernel link layer protocol match: raw-ip
Starting network with 'qmicli -d /dev/cdc-wdm0 --device-open-net=net-raw-ip|net-no-qos
-header --wds-start-network=apn='pp.vodafone.co.uk',ip-type=4  --client-no-release-cid
 '...
Saving state at /tmp/qmi-network-state-cdc-wdm0... (CID: 5)
Saving state at /tmp/qmi-network-state-cdc-wdm0... (PDH: 3783131952)
Network started successfully
root@raspberrypi:~# udhcpc -i wwan0
udhcpc: started, v1.30.1
No resolv.conf for interface wwan0.udhcpc
udhcpc: sending discover
udhcpc: sending select for 10.32.10.179
udhcpc: lease of 10.32.10.179 obtained, lease time 7200
root@raspberrypi:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.32.10.180    0.0.0.0         UG    0      0        0 wwan0
0.0.0.0         192.168.1.1     0.0.0.0         UG    202    0        0 eth0
10.32.10.176    0.0.0.0         255.255.255.248 U     0      0        0 wwan0
192.168.1.0     0.0.0.0         255.255.255.0   U     202    0        0 eth0
root@raspberrypi:~# ping -c3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=645 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=104 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=53.4 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 53.423/267.736/645.355/267.827 ms
root@raspberrypi:~#
```

QUECTEL-CM example

Quectel Modems have a utility provided by Quectel to manage the connection process and which will automatically configure any raw-ip settings

First install the all-in-one quectel-cm connection helper program; this will automatically configure any raw-ip settings

https://github.com/mypiandrew/quectel-cm/releases/download/V1.6.0.12/quectel-CM.tar.gz



```
root@raspberrypi:~# cd /root/quectel-CM/
root@raspberrypi:~/quectel-CM# chmod +x install.sh
root@raspberrypi:~/quectel-CM# ./install.sh
updateing dhcp settings
installing udhcpc
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
udhcpc is already the newest version (1:1.30.1-6).
0 upgraded, 0 newly installed, 0 to remove and 44 not upgraded.
Copying udhcpc default script

Building quectel-CM programs
rm -rf quectel-CM *~
rm -rf quectel-qmi-proxy
gcc -Wall -s quectel-qmi-proxy.c  -o quectel-qmi-proxy -lpthread -ldl
gcc -Wall -s QmiWwanCM.c GobiNetCM.c main.c MPQMUX.c QMIThread.c util.c qmap_bridge_mo
de.c mbim-cm.c device.c udhcpc.c -o quectel-CM -lpthread -ldl

Installing quectel-CM & quecetel-qmi-proxy in /usr/local/bin

Now reboot and follow README.sample.connection.txt


root@raspberrypi:~/quectel-CM#
```

The command has the below syntax

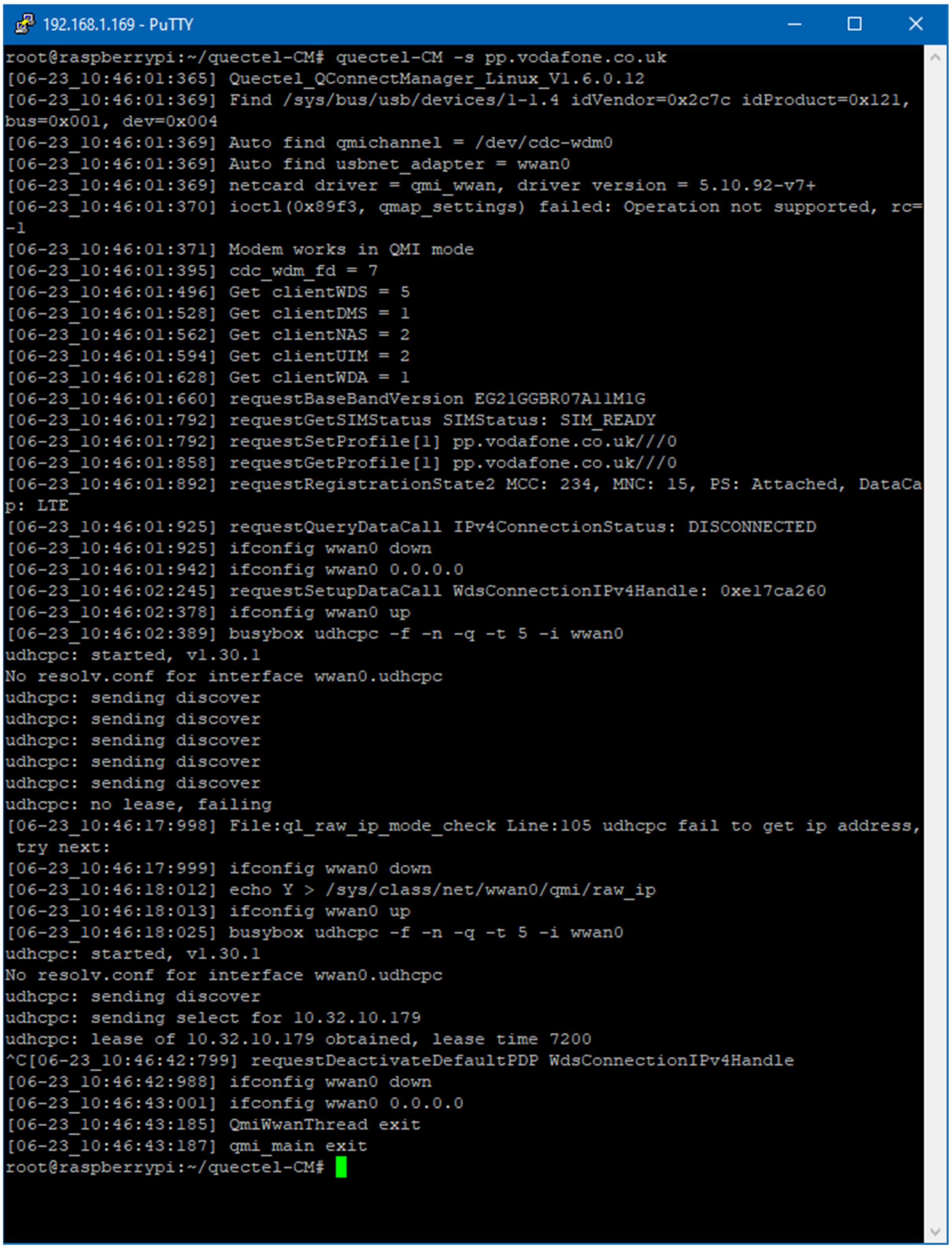
```
quectel-CM [-s [apn [user password auth]]]

              [-p pincode] [-f logfilename] -s [apn [user password auth]]
```

Example 1: ./quectel-CM
Example 2: ./quectel-CM -s pp.vodafone.co.uk
Example 3: ./quectel-CM -s internet web web 0 -p 1234 -f modemconnect.log

Note that this is a non-exiting process so will not automatically fork and run in the background

Sample Connection output, note the fall back to raw-ip is automatic.

Killing the process or issuing Ctrl-C results in the connection to be disconnected and interface disabled.



```
root@raspberrypi:~/quectel-CM# quectel-CM -s pp.vodafone.co.uk
[06-23_10:46:01:365] Quectel_QConnectManager_Linux_V1.6.0.12
[06-23_10:46:01:369] Find /sys/bus/usb/devices/1-1.4 idVendor=0x2c7c idProduct=0x121,
bus=0x001, dev=0x004
[06-23_10:46:01:369] Auto find qmichannel = /dev/cdc-wdm0
[06-23_10:46:01:369] Auto find usbnet_adapter = wwan0
[06-23_10:46:01:369] netcard driver = qmi_wwan, driver version = 5.10.92-v7+
[06-23_10:46:01:370] ioctl(0x89f3, qmap_settings) failed: Operation not supported, rc=
-1
[06-23_10:46:01:371] Modem works in QMI mode
[06-23_10:46:01:395] cdc_wdm_fd = 7
[06-23_10:46:01:496] Get clientWDS = 5
[06-23_10:46:01:528] Get clientDMS = 1
[06-23_10:46:01:562] Get clientNAS = 2
[06-23_10:46:01:594] Get clientUIM = 2
[06-23_10:46:01:628] Get clientWDA = 1
[06-23_10:46:01:660] requestBaseBandVersion EG21GGBR07A11M1G
[06-23_10:46:01:792] requestGetSIMStatus SIMStatus: SIM_READY
[06-23_10:46:01:792] requestSetProfile[1] pp.vodafone.co.uk///0
[06-23_10:46:01:858] requestGetProfile[1] pp.vodafone.co.uk///0
[06-23_10:46:01:892] requestRegistrationState2 MCC: 234, MNC: 15, PS: Attached, DataCa
p: LTE
[06-23_10:46:01:925] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[06-23_10:46:01:925] ifconfig wwan0 down
[06-23_10:46:01:942] ifconfig wwan0 0.0.0.0
[06-23_10:46:02:245] requestSetupDataCall WdsConnectionIPv4Handle: 0xe17ca260
[06-23_10:46:02:378] ifconfig wwan0 up
[06-23_10:46:02:389] busybox udhcpc -f -n -q -t 5 -i wwan0
udhcpc: started, v1.30.1
No resolv.conf for interface wwan0.udhcpc
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending discover
udhcpc: no lease, failing
[06-23_10:46:17:998] File:ql_raw_ip_mode_check Line:105 udhcpc fail to get ip address,
 try next:
[06-23_10:46:17:999] ifconfig wwan0 down
[06-23_10:46:18:012] echo Y > /sys/class/net/wwan0/qmi/raw_ip
[06-23_10:46:18:013] ifconfig wwan0 up
[06-23_10:46:18:025] busybox udhcpc -f -n -q -t 5 -i wwan0
udhcpc: started, v1.30.1
No resolv.conf for interface wwan0.udhcpc
udhcpc: sending discover
udhcpc: sending select for 10.32.10.179
udhcpc: lease of 10.32.10.179 obtained, lease time 7200
^C[06-23_10:46:42:799] requestDeactivateDefaultPDP WdsConnectionIPv4Handle
[06-23_10:46:42:988] ifconfig wwan0 down
[06-23_10:46:43:001] ifconfig wwan0 0.0.0.0
[06-23_10:46:43:185] QmiWwanThread exit
[06-23_10:46:43:187] qmi_main exit
root@raspberrypi:~/quectel-CM#
```

mPCIe IO Cards

Also available are our range of pre-certified RF modules :

- LoRa (Microchip RN2483/RN2903)
- Bluetooth 4.0 BLE (Silicon Labs/BlueGiga BLE112)
- Bluetooth 5 (Laird BL652)
- enOcean TCM310
- ZIGBEE/802.15.4 (Silicon Labs/Telegesis RX357 Module L.R. UFL)
- XBEE

These all feature an FTDI230X USB to UART chip and so appear automatically as a standard serial port ready to run with minimal configuration needed, so offer a fast development cycle.

In order to make the **ttyUSBx** serial port for the mPCIe cards above constantly easy to identify we use a udev rule to help us, this is called **10-ftdi-usbserial.rules** and is located **/etc/udev/rules.d/**

This udev rule creates a symlink for the FTDI ttyUSBx serial port called **/dev/ttyS1**

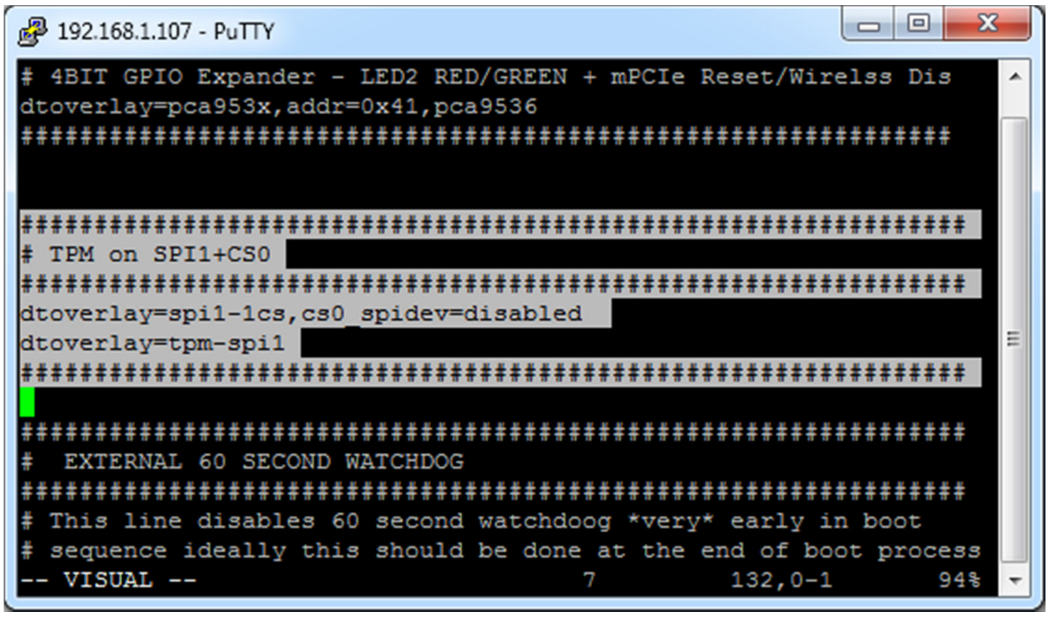
For more information on how each card works please see the respective documentation page on the website.

SPI TRUSTED PLATFORM MODULE

Integrated on-board is an Infineon SLB9670 TPM 2.0 device, this is connected to the Raspberry Pi via the SPI-1 bus.

Support for this device was included in mainline Kernel 4.14.85 and the device is configured via the files below

/boot/config.txt
/boot/overlays/tpm-spi1.dtbo



192.168.1.107 - PuTTY

```
# 4BIT GPIO Expander - LED2 RED/GREEN + mPCIe Reset/Wirelss Dis
dtoverlay=pca953x,addr=0x41,pca9536
################################################################


#################################################################
# TPM on SPI1+CS0
#################################################################
dtoverlay=spi1-1cs,cs0_spidev=disabled
dtoverlay=tpm-spi1
#################################################################

#################################################################
#  EXTERNAL 60 SECOND WATCHDOG
#################################################################
# This line disables 60 second watchdoog *very* early in boot
# sequence ideally this should be done at the end of boot process
-- VISUAL --                                7          132,0-1         94%
```

This configures SPI-1 interface and the TPM, when the system has booted it will create a **/dev/tpm0** node.

We have pre-installed **tpmtool** and **tpmupdate** software utilities to allow for administration of the device

/usr/local/bin/tpminfo.sh

/usr/local/bin/tpmtool
/root/tpm-toolkit/

/usr/local/bin/tpmupdate

For more information see the github repository below

https://github.com/Infineon/eltt2

A hardware reset of the SLB9670 device is asserted at reboot/power up by the Compute Module

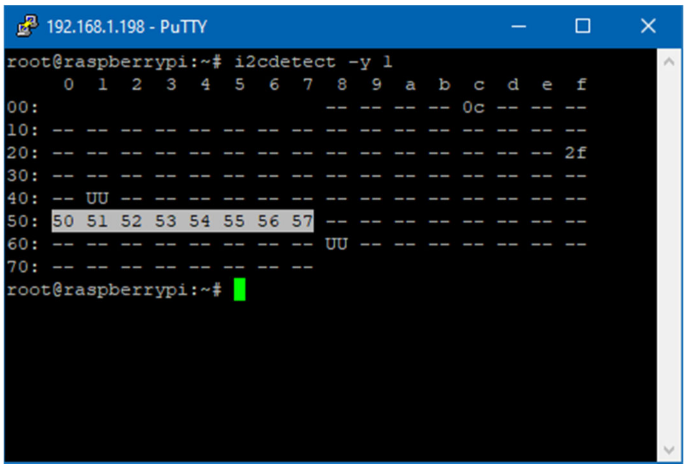
I2C USER EEPROM

A 256Byte EEPROM for user ID storage

The lower 128Byte has read/write access for user storage, the first 4 hex bytes have been programmed with an ID code visible on the barcoded sticker affixed to the PCB.

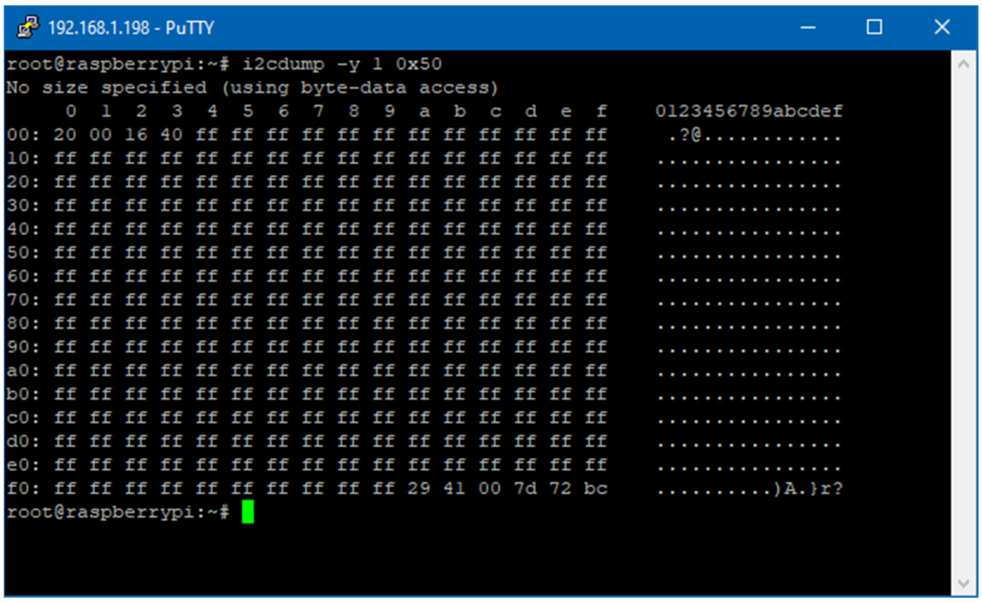
The upper 128byte is read only with the last 32bits (6 hex bytes) containing a unique ID code.

The EEPROM's id is 0x50 with shadow addresses at 0x51-0x57



```
root@raspberrypi:~# i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- 0c -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- 2f
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- UU -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 51 52 53 54 55 56 57 -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- UU -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
root@raspberrypi:~#
```

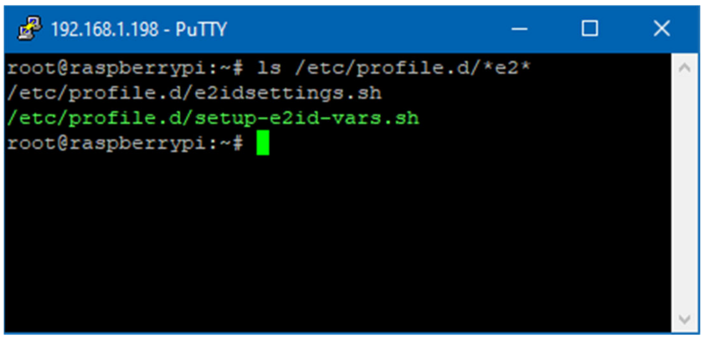
The EEPROM can be accessed for read/write operations using **i2c-tools** utilities, such as **i2cdump**



```
root@raspberrypi:~# i2cdump -y 1 0x50
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 20 00 16 40 ff ff ff ff ff ff ff ff ff ff ff ff     .?@...........
10: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
20: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
30: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
40: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
50: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
60: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
70: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
80: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
90: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
a0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
b0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
c0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
d0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
e0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
f0: ff ff ff ff ff ff ff ff ff ff 29 41 00 7d 72 bc    ..........)A.}r?
root@raspberrypi:~#
```

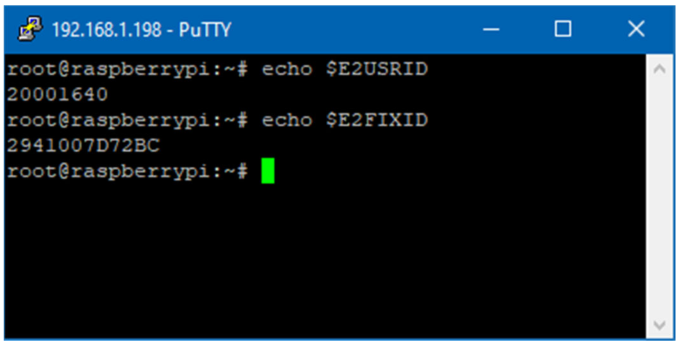
For convenience a script to create two bash environment variables has been created in **/etc/profile.d**

setup-e2id-vars.sh creates **e2idsettings.sh** on first run

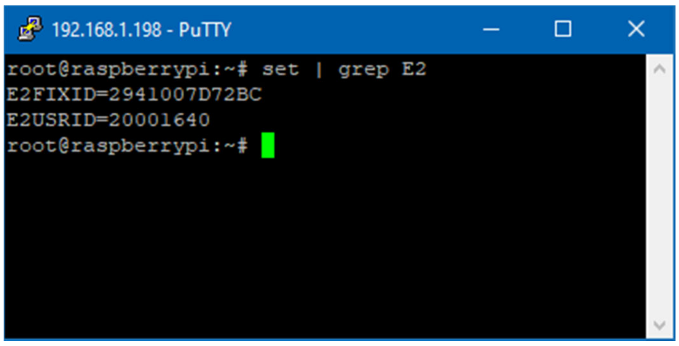


```
root@raspberrypi:~# ls /etc/profile.d/*e2*
/etc/profile.d/e2idsettings.sh
/etc/profile.d/setup-e2id-vars.sh
root@raspberrypi:~#
```

These environment variables can be used in scripting by any user

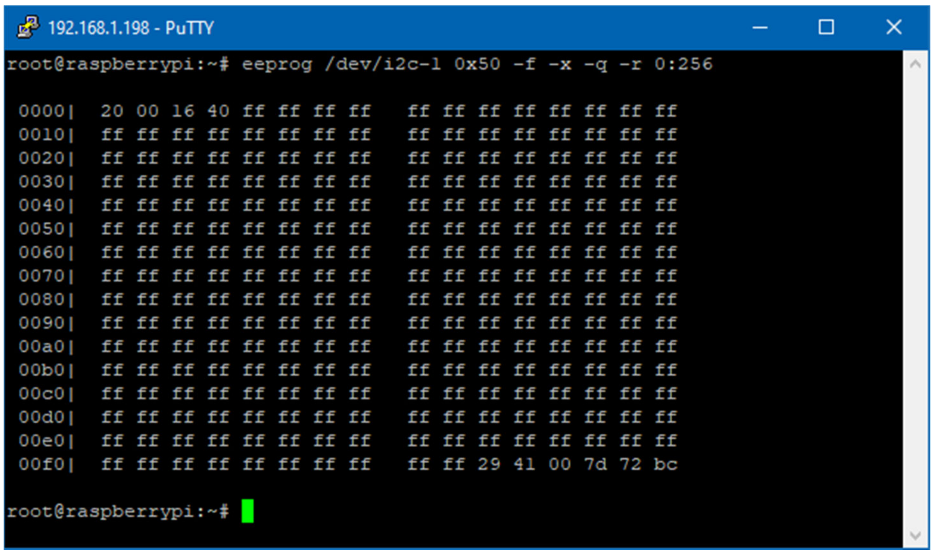


```
root@raspberrypi:~# echo $E2USRID
20001640
root@raspberrypi:~# echo $E2FIXID
2941007D72BC
root@raspberrypi:~#
```



```
root@raspberrypi:~# set | grep E2
E2FIXID=2941007D72BC
E2USRID=20001640
root@raspberrypi:~#
```

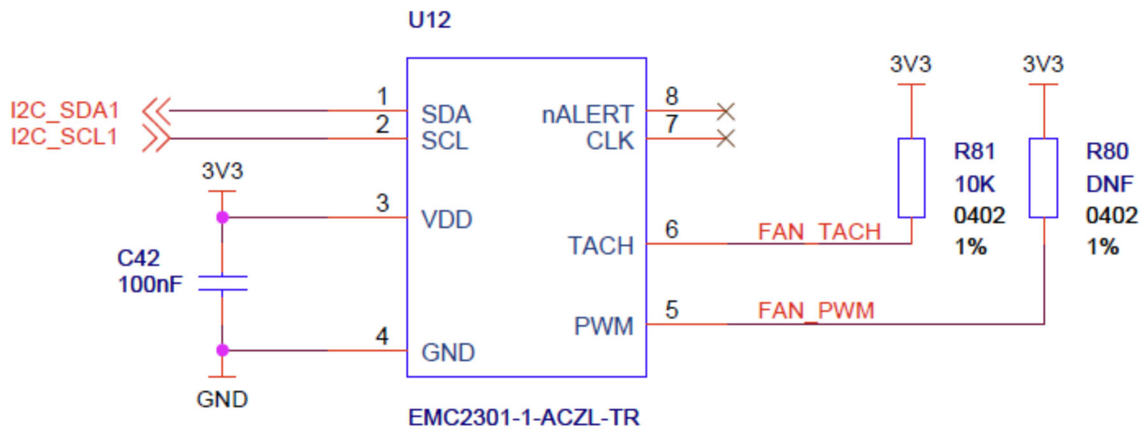
Also included on the factory Raspbian OS image is the **eeprog** command line utility that can also be used to read/write the EEPROM (source code in **/root/eeprom**)

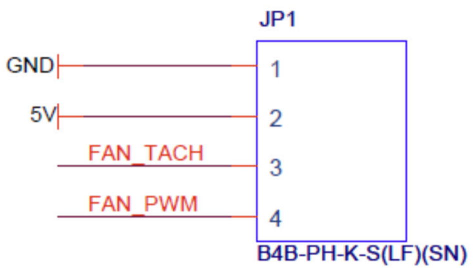


```
root@raspberrypi:~# eeprog /dev/i2c-1 0x50 -f -x -q -r 0:256

 0000|  20 00 16 40 ff ff ff ff   ff ff ff ff ff ff ff ff
 0010|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 0020|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 0030|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 0040|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 0050|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 0060|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 0070|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 0080|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 0090|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 00a0|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 00b0|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 00c0|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 00d0|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 00e0|  ff ff ff ff ff ff ff ff   ff ff ff ff ff ff ff ff
 00f0|  ff ff ff ff ff ff ff ff   ff ff 29 41 00 7d 72 bc

root@raspberrypi:~#
```

I2C FAN CONTROLLER

Integrated onto the main board is a Microchip EMC2301 PWM Fan controller





This device occupies address 0x2F and can be operated by either basic I2C commands or a more comprehensive kernel driver

I2C Method

Turn fan off:

```
i2cset -y 1 0x2f 0x30 0x00
```

Turn fan on 100%:

```
i2cset -y 1 0x2f 0x30 0xff
```

We have created a bash script to operate this and get RPM information, see link below :

https://github.com/mypiandrew/fanctrl

Kernel Driver Method

A 3rd party driver here can be installed to allow more control over the fan operation

https://github.com/neg2led/cm4io-fan

I2C REAL TIME CLOCK

A DS1338Z-33+ Real Time Clock with battery backup cell is integrated onto the board, this is configured by the below device tree overlay

```
dtoverlay=i2c-rtc,ds1307,addr=0x68
```

Further OS integration to remove the **fake-hwclock** functionality, and ensure the system reads/writes to the hwclock, has also been done.

A good primer on this can be found here :

https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi/set-rtc-time

A symbolic link is also setup to allow quick access to the devices battery backed NVRAM

```
rtc_nvram       -> /sys/class/rtc/rtc0/device/nvram
```

Example usage :

```
# echo ‘battery backed up ram’ > /dev/rtc_nvram

# cat /dev/rtc_nvram
battery backed up ram
```

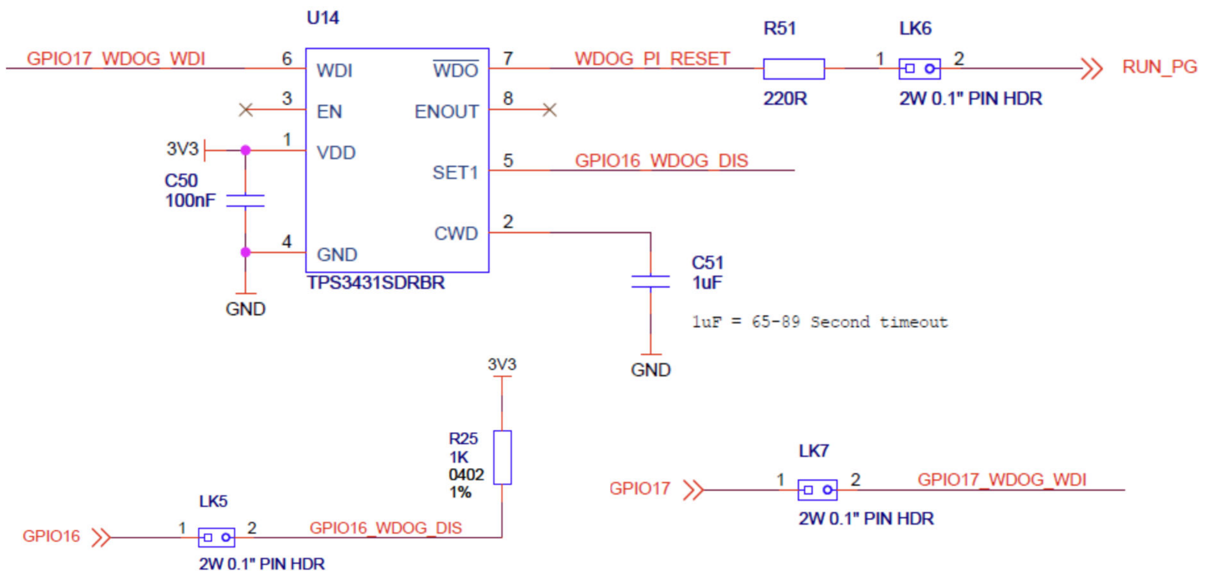
WATCHDOG

The on-board external watchdog with a 60 Second delay timer, provided by a Texas Instruments TPS3431 part, the reset output of this is connected to the Raspberry Pi Compute module's reset pin.

This is provided to give an extra layer of resilience over a system lockup in the event that the user considers the RPi on-chip watchdog is unsuitable for their application.

The external watchdog device is enabled by push on links LK5, LK7 & LK7 and driven by GPIO16 (WD Enable) and GPIO17 (WD Input).

Once these push on links are fitted the watchdog is enabled by default covering the whole boot cycle.



Once the watchdog is enabled the WD Input pin on the device must be togged H-L-H at least once per watchdog time-out period (60 seconds) and the low level pulse period must be >1uS long for the transition to be valid.

If the device sees a valid low-to-high transition on the input pin the internal 60 second countdown timer is reset and restarted. If the device does not see a valid input pulse within the watchdog time out period it will pull the RPi CPU module reset line low and hard reset the system.

The watchdog can be disabled completely by either physically removing LK6 (optionally removing LK5&7 additionally) or by driving GPIO16 low.

/etc/init.d/mypi-init.sh sets up symbolic links for these GPIO Lines, see this file for more details.

Watchdog integration can be directly done inside application code by writing directly to the GPIO lines or can be done via a kernel GPIO watchdog process via the watchdog package which provides more varied sources for monitoring.

The OS files for the CM4 setup have been placed in the **/root/watchdog** folder on the demo OS image.

External Watchdog OS Integration

Integrated into the Raspbian OS there are pre-built utilities for configuring and managing watchdogs, in this example we will show how to configure the OS to use the on board external watch dog such that a file's last update timestamp can trigger a watchdog time out.

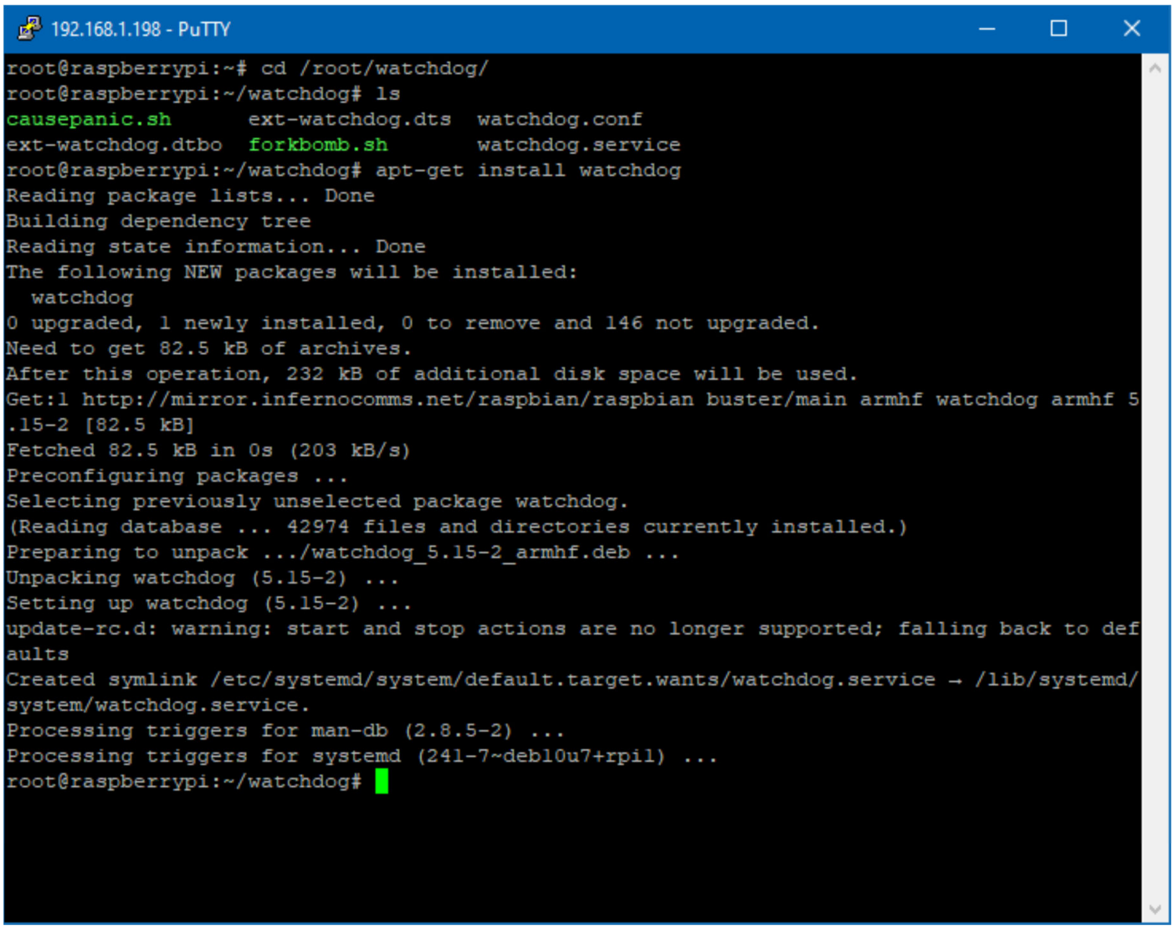
In this configuration if the target file is not updated the system will attempt an "orderly" reset as it performs some basic "clean-up" tasks prior to finally stopping the watchdog input line toggling, and so causing the Raspberry Pi Compute Module's reset line (aka RUN pin) to be momentarily pulled low by the watchdog device resulting in a hard reset.

The watchdog system is configured by 3 main files

- A device tree configuration file to enable the GPIO Watchdog timer **/dev/watchdog1**
- A systemd service file **/lib/systemd/system/watchdog.service**
- The conditional check options specified in **/etc/watchdog.conf**

The configuration files for these are stored in **/root/watchdog** on the demo image

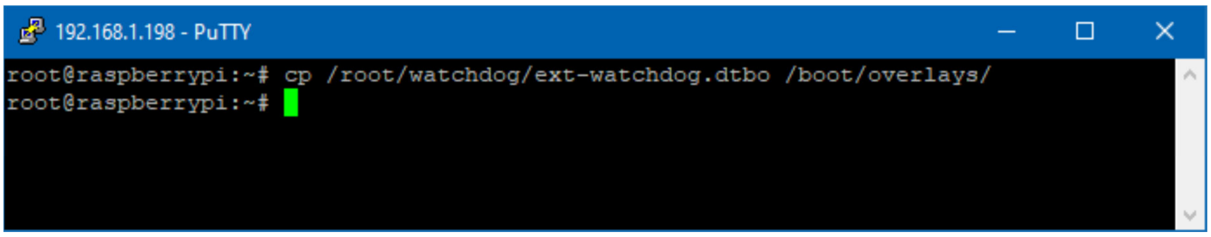
The watchdog OS package needs to be installed



```
root@raspberrypi:~# cd /root/watchdog/
root@raspberrypi:~/watchdog# ls
causepanic.sh      ext-watchdog.dts  watchdog.conf
ext-watchdog.dtbo  forkbomb.sh       watchdog.service
root@raspberrypi:~/watchdog# apt-get install watchdog
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  watchdog
0 upgraded, 1 newly installed, 0 to remove and 146 not upgraded.
Need to get 82.5 kB of archives.
After this operation, 232 kB of additional disk space will be used.
Get:1 http://mirror.infernocomms.net/raspbian/raspbian buster/main armhf watchdog armhf 5
.15-2 [82.5 kB]
Fetched 82.5 kB in 0s (203 kB/s)
Preconfiguring packages ...
Selecting previously unselected package watchdog.
(Reading database ... 42974 files and directories currently installed.)
Preparing to unpack .../watchdog_5.15-2_armhf.deb ...
Unpacking watchdog (5.15-2) ...
Setting up watchdog (5.15-2) ...
update-rc.d: warning: start and stop actions are no longer supported; falling back to def
aults
Created symlink /etc/systemd/system/default.target.wants/watchdog.service → /lib/systemd/
system/watchdog.service.
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for systemd (241-7~deb10u7+rpi1) ...
root@raspberrypi:~/watchdog#
```

```
# apt-get install watchdog
```

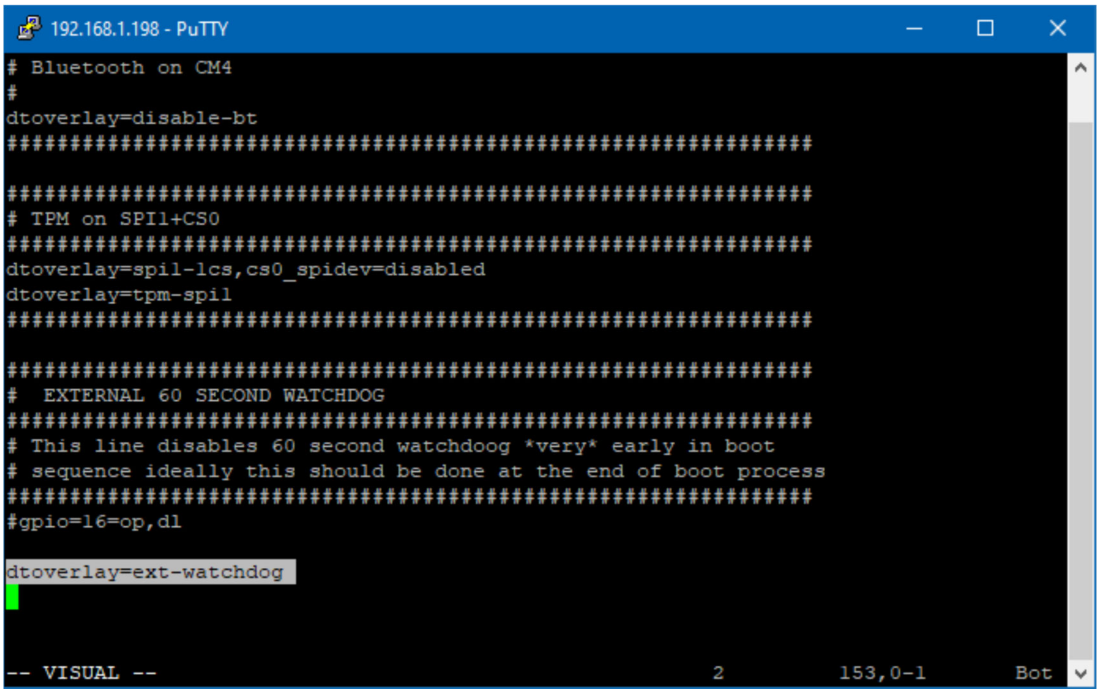
Copy the device tree overlay for this module to the /boot/overlays folder if not already done so to configure the kernel to add a GPIO watchdog to the system


192.168.1.198 - PuTTY

```
root@raspberrypi:~# cp /root/watchdog/ext-watchdog.dtbo /boot/overlays/
root@raspberrypi:~#
```

Add the below line to the end of **/boot/config.txt** to enable the overlay

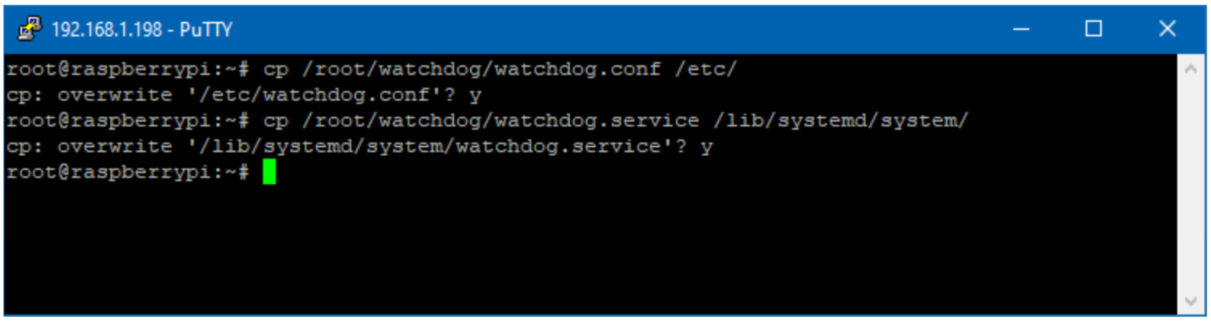
```
dtoverlay=ext-watchdog
```


192.168.1.198 - PuTTY

```
# Bluetooth on CM4
#
dtoverlay=disable-bt
################################################################

################################################################
# TPM on SPI1+CS0
################################################################
dtoverlay=spi1-1cs,cs0_spidev=disabled
dtoverlay=tpm-spi1
################################################################

################################################################
#  EXTERNAL 60 SECOND WATCHDOG
################################################################
# This line disables 60 second watchdoog *very* early in boot
# sequence ideally this should be done at the end of boot process
################################################################
#gpio=16=op,dl

dtoverlay=ext-watchdog


-- VISUAL --                                        2         153,0-1       Bot
```

Install the configuration files for the watchdog service and the service file that starts the watchdog OS service then enables the watchdog.

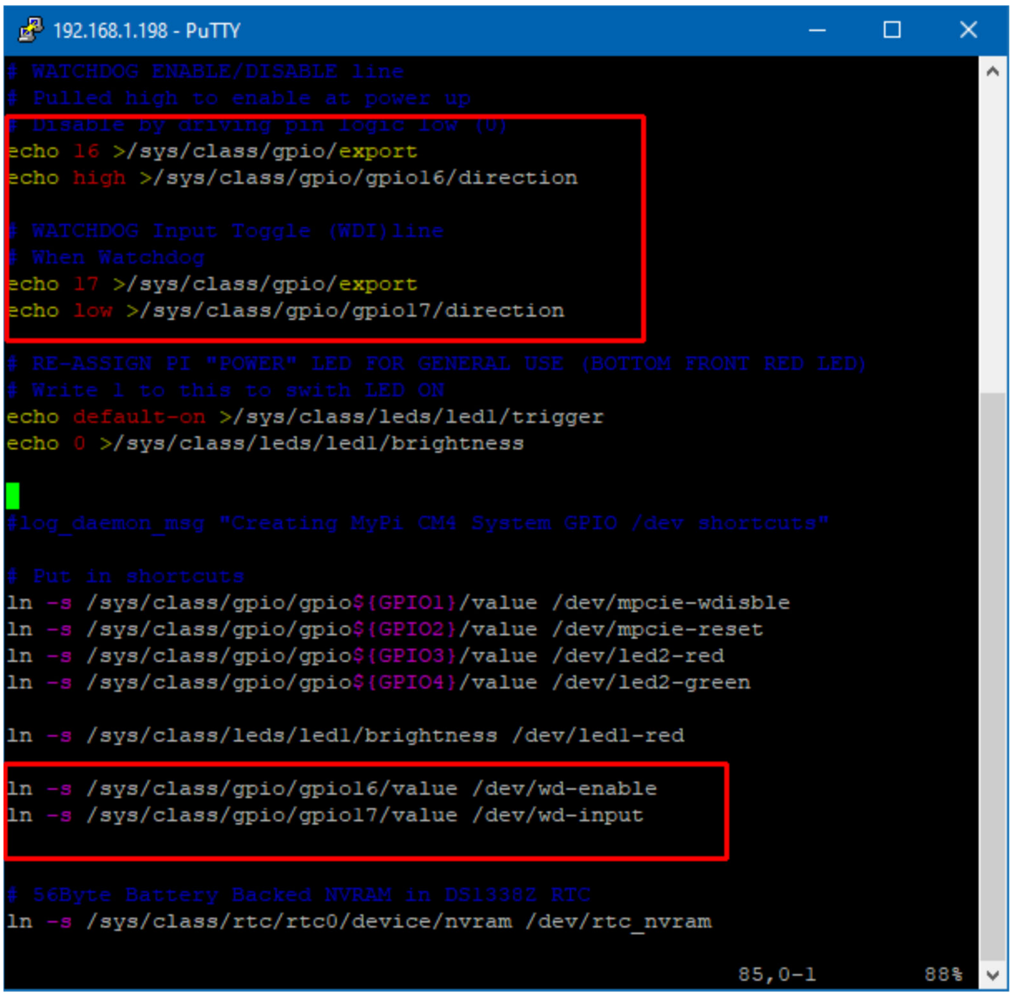
```
cp /root/watchdog/watchdog.conf /etc/
cp /root/watchdog/watchdog.service /lib/systemd/system/watchdog.service
```


192.168.1.198 - PuTTY

```
root@raspberrypi:~# cp /root/watchdog/watchdog.conf /etc/
cp: overwrite '/etc/watchdog.conf'? y
root@raspberrypi:~# cp /root/watchdog/watchdog.service /lib/systemd/system/
cp: overwrite '/lib/systemd/system/watchdog.service'? y
root@raspberrypi:~#
```

Finally we need to remove the reference to the watchdog from **/etc/init.d/mypi-init.sh** to allow the watchdog and service file to claim the relevant IO lines.

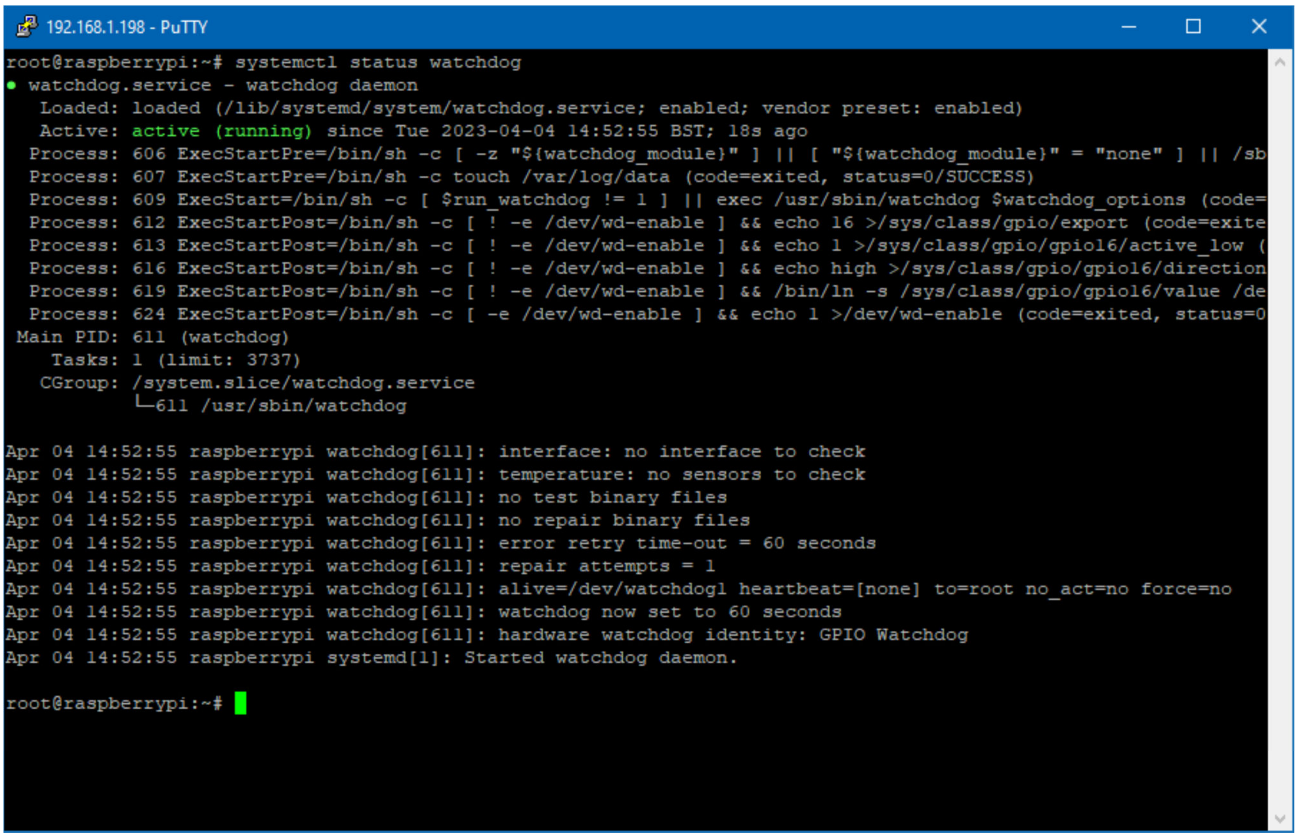
Do this by adding a **#** to the start of of each of the lines highlighted below



```
# WATCHDOG ENABLE/DISABLE line
# Pulled high to enable at power up
# Disable by driving pin logic low (0)
echo 16 >/sys/class/gpio/export
echo high >/sys/class/gpio/gpio16/direction

# WATCHDOG Input Toggle (WDI)line
# When Watchdog
echo 17 >/sys/class/gpio/export
echo low >/sys/class/gpio/gpio17/direction

# RE-ASSIGN PI "POWER" LED FOR GENERAL USE (BOTTOM FRONT RED LED)
# Write 1 to this to swith LED ON
echo default-on >/sys/class/leds/led1/trigger
echo 0 >/sys/class/leds/led1/brightness


#log_daemon_msg "Creating MyPi CM4 System GPIO /dev shortcuts"

# Put in shortcuts
ln -s /sys/class/gpio/gpio${GPIO1}/value /dev/mpcie-wdisble
ln -s /sys/class/gpio/gpio${GPIO2}/value /dev/mpcie-reset
ln -s /sys/class/gpio/gpio${GPIO3}/value /dev/led2-red
ln -s /sys/class/gpio/gpio${GPIO4}/value /dev/led2-green

ln -s /sys/class/leds/led1/brightness /dev/led1-red

ln -s /sys/class/gpio/gpio16/value /dev/wd-enable
ln -s /sys/class/gpio/gpio17/value /dev/wd-input


# 56Byte Battery Backed NVRAM in DS1338Z RTC
ln -s /sys/class/rtc/rtc0/device/nvram /dev/rtc_nvram

                                                        85,0-1        88%
```

On reboot you should be able to issue the command shown below to check the services have started correctly.

```
# systemctl status watchdog
```

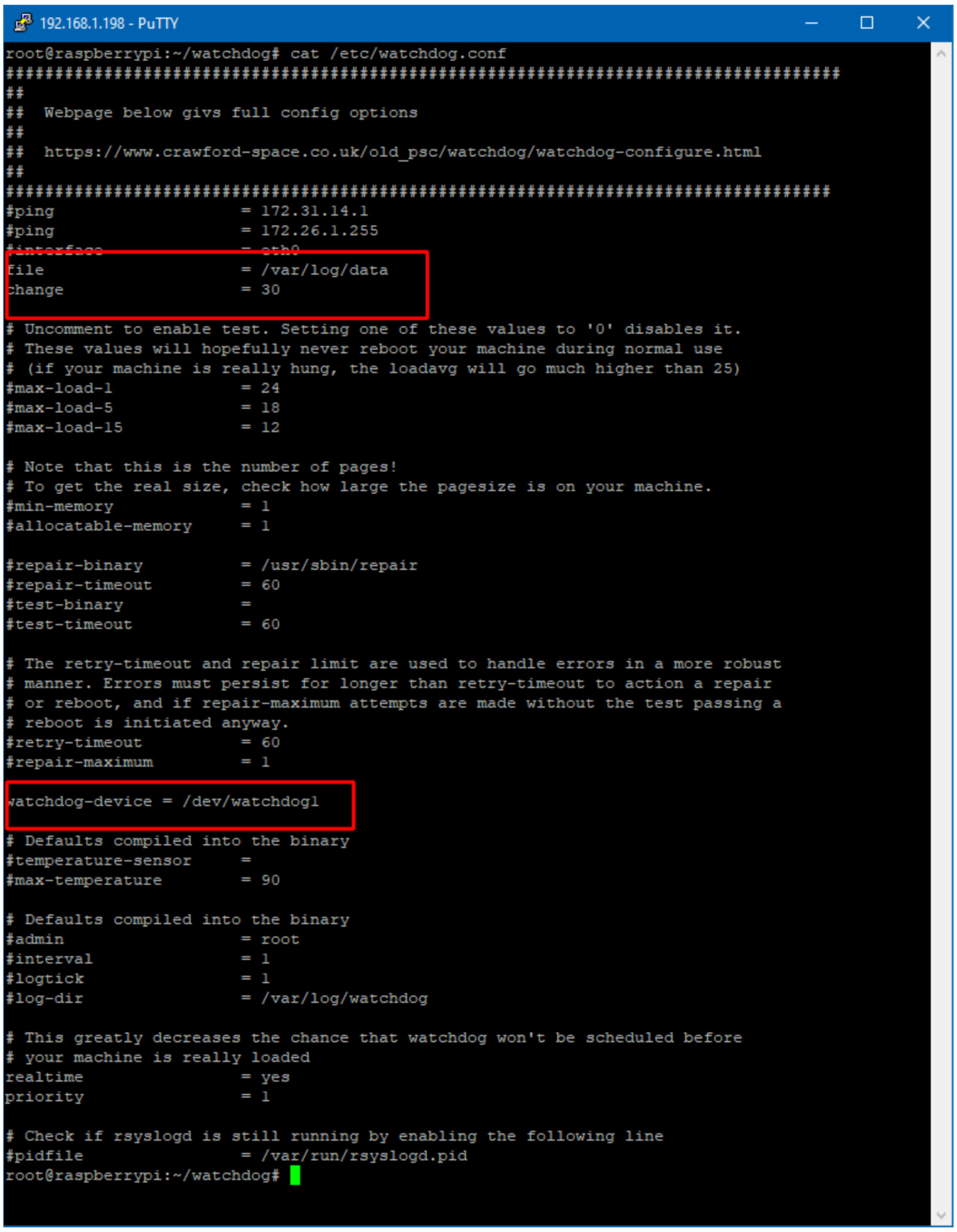


```
root@raspberrypi:~# systemctl status watchdog
● watchdog.service - watchdog daemon
   Loaded: loaded (/lib/systemd/system/watchdog.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2023-04-04 14:52:55 BST; 18s ago
  Process: 606 ExecStartPre=/bin/sh -c [ -z "${watchdog_module}" ] || [ "${watchdog_module}" = "none" ] || /sb
  Process: 607 ExecStartPre=/bin/sh -c touch /var/log/data (code=exited, status=0/SUCCESS)
  Process: 609 ExecStart=/bin/sh -c [ $run_watchdog != 1 ] || exec /usr/sbin/watchdog $watchdog_options (code=
  Process: 612 ExecStartPost=/bin/sh -c [ ! -e /dev/wd-enable ] && echo 16 >/sys/class/gpio/export (code=exite
  Process: 613 ExecStartPost=/bin/sh -c [ ! -e /dev/wd-enable ] && echo 1 >/sys/class/gpio/gpio16/active_low (
  Process: 616 ExecStartPost=/bin/sh -c [ ! -e /dev/wd-enable ] && echo high >/sys/class/gpio/gpio16/direction
  Process: 619 ExecStartPost=/bin/sh -c [ ! -e /dev/wd-enable ] && /bin/ln -s /sys/class/gpio/gpio16/value /de
  Process: 624 ExecStartPost=/bin/sh -c [ -e /dev/wd-enable ] && echo 1 >/dev/wd-enable (code=exited, status=0
 Main PID: 611 (watchdog)
    Tasks: 1 (limit: 3737)
   CGroup: /system.slice/watchdog.service
           └─611 /usr/sbin/watchdog

Apr 04 14:52:55 raspberrypi watchdog[611]: interface: no interface to check
Apr 04 14:52:55 raspberrypi watchdog[611]: temperature: no sensors to check
Apr 04 14:52:55 raspberrypi watchdog[611]: no test binary files
Apr 04 14:52:55 raspberrypi watchdog[611]: no repair binary files
Apr 04 14:52:55 raspberrypi watchdog[611]: error retry time-out = 60 seconds
Apr 04 14:52:55 raspberrypi watchdog[611]: repair attempts = 1
Apr 04 14:52:55 raspberrypi watchdog[611]: alive=/dev/watchdog1 heartbeat=[none] to=root no_act=no force=no
Apr 04 14:52:55 raspberrypi watchdog[611]: watchdog now set to 60 seconds
Apr 04 14:52:55 raspberrypi watchdog[611]: hardware watchdog identity: GPIO Watchdog
Apr 04 14:52:55 raspberrypi systemd[1]: Started watchdog daemon.

root@raspberrypi:~#
```

/etc/watchdog.conf contains the configuration for the external watchdog process, this has been configured in this example to monitor the file **/var/log/data** and check to see its timestamp updates at least once every 30 seconds

Other options are available :

https://manpages.debian.org/testing/watchdog/watchdog.8.en.html
https://manpages.debian.org/testing/watchdog/watchdog.conf.5.en.html

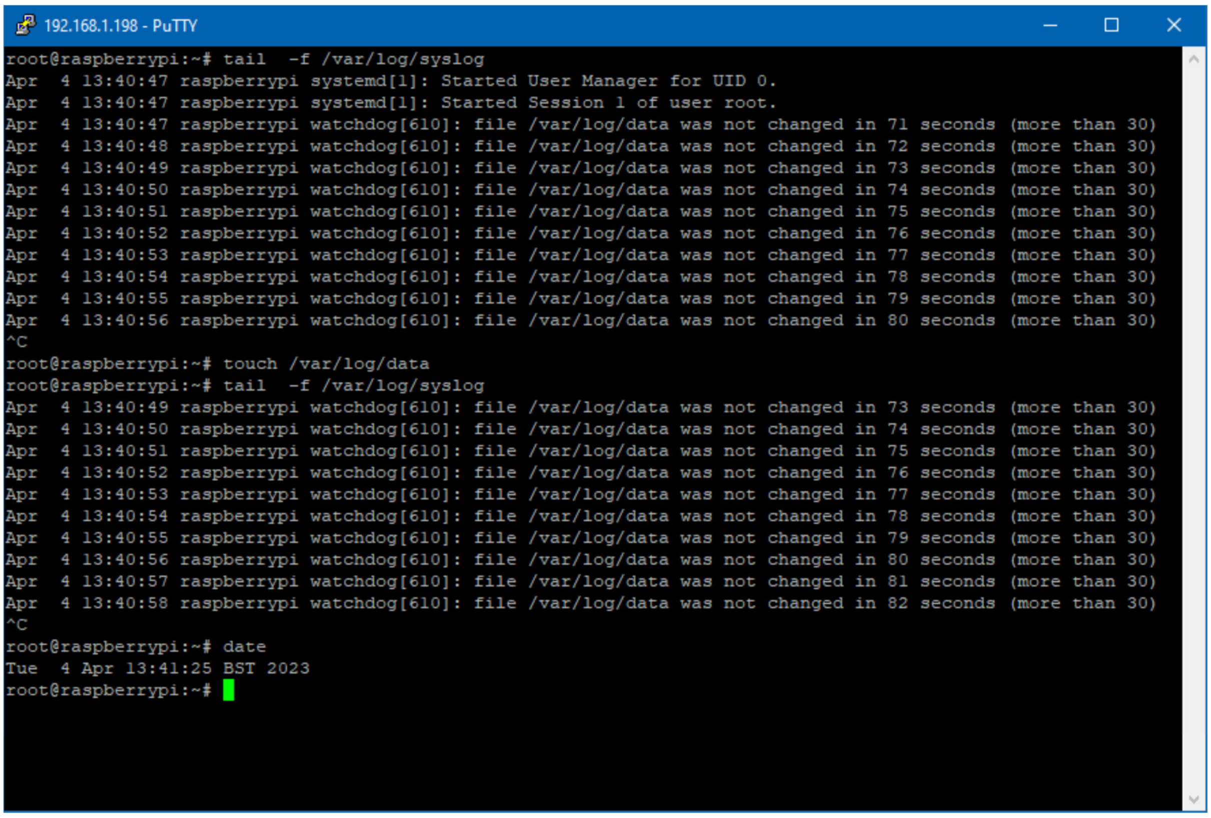


```
root@raspberrypi:~/watchdog# cat /etc/watchdog.conf
#####################################################################################
##
##  Webpage below givs full config options
##
##  https://www.crawford-space.co.uk/old_psc/watchdog/watchdog-configure.html
##
####################################################################################
#ping                    = 172.31.14.1
#ping                    = 172.26.1.255
#interface               = eth0
file                     = /var/log/data
change                   = 30
# Uncomment to enable test. Setting one of these values to '0' disables it.
# These values will hopefully never reboot your machine during normal use
# (if your machine is really hung, the loadavg will go much higher than 25)
#max-load-1              = 24
#max-load-5              = 18
#max-load-15             = 12

# Note that this is the number of pages!
# To get the real size, check how large the pagesize is on your machine.
#min-memory              = 1
#allocatable-memory      = 1

#repair-binary           = /usr/sbin/repair
#repair-timeout          = 60
#test-binary             =
#test-timeout            = 60

# The retry-timeout and repair limit are used to handle errors in a more robust
# manner. Errors must persist for longer than retry-timeout to action a repair
# or reboot, and if repair-maximum attempts are made without the test passing a
# reboot is initiated anyway.
#retry-timeout           = 60
#repair-maximum          = 1

watchdog-device = /dev/watchdog1

# Defaults compiled into the binary
#temperature-sensor      =
#max-temperature         = 90

# Defaults compiled into the binary
#admin                   = root
#interval                = 1
#logtick                 = 1
#log-dir                 = /var/log/watchdog

# This greatly decreases the chance that watchdog won't be scheduled before
# your machine is really loaded
realtime                 = yes
priority                 = 1

# Check if rsyslogd is still running by enabling the following line
#pidfile                 = /var/run/rsyslogd.pid
root@raspberrypi:~/watchdog#
```

Note that the watchdog device we have configured is **/dev/watchdog1** and not **/dev/watchdog0** which is the internal Pi CPU watchdog)

If the file **/var/log/data** we have configured as the test for watchdog time out is not written to for a period of 3 x the change value (in seconds) then the system will attempt a managed restart, by shutting as many services down as possible etc and then stopping the watchdog timer, causing a hard reset

At any point up to this final time out writing/touching the file will reset the counter.



```
root@raspberrypi:~# tail  -f /var/log/syslog
Apr  4 13:40:47 raspberrypi systemd[1]: Started User Manager for UID 0.
Apr  4 13:40:47 raspberrypi systemd[1]: Started Session 1 of user root.
Apr  4 13:40:47 raspberrypi watchdog[610]: file /var/log/data was not changed in 71 seconds (more than 30)
Apr  4 13:40:48 raspberrypi watchdog[610]: file /var/log/data was not changed in 72 seconds (more than 30)
Apr  4 13:40:49 raspberrypi watchdog[610]: file /var/log/data was not changed in 73 seconds (more than 30)
Apr  4 13:40:50 raspberrypi watchdog[610]: file /var/log/data was not changed in 74 seconds (more than 30)
Apr  4 13:40:51 raspberrypi watchdog[610]: file /var/log/data was not changed in 75 seconds (more than 30)
Apr  4 13:40:52 raspberrypi watchdog[610]: file /var/log/data was not changed in 76 seconds (more than 30)
Apr  4 13:40:53 raspberrypi watchdog[610]: file /var/log/data was not changed in 77 seconds (more than 30)
Apr  4 13:40:54 raspberrypi watchdog[610]: file /var/log/data was not changed in 78 seconds (more than 30)
Apr  4 13:40:55 raspberrypi watchdog[610]: file /var/log/data was not changed in 79 seconds (more than 30)
Apr  4 13:40:56 raspberrypi watchdog[610]: file /var/log/data was not changed in 80 seconds (more than 30)
^C
root@raspberrypi:~# touch /var/log/data
root@raspberrypi:~# tail  -f /var/log/syslog
Apr  4 13:40:49 raspberrypi watchdog[610]: file /var/log/data was not changed in 73 seconds (more than 30)
Apr  4 13:40:50 raspberrypi watchdog[610]: file /var/log/data was not changed in 74 seconds (more than 30)
Apr  4 13:40:51 raspberrypi watchdog[610]: file /var/log/data was not changed in 75 seconds (more than 30)
Apr  4 13:40:52 raspberrypi watchdog[610]: file /var/log/data was not changed in 76 seconds (more than 30)
Apr  4 13:40:53 raspberrypi watchdog[610]: file /var/log/data was not changed in 77 seconds (more than 30)
Apr  4 13:40:54 raspberrypi watchdog[610]: file /var/log/data was not changed in 78 seconds (more than 30)
Apr  4 13:40:55 raspberrypi watchdog[610]: file /var/log/data was not changed in 79 seconds (more than 30)
Apr  4 13:40:56 raspberrypi watchdog[610]: file /var/log/data was not changed in 80 seconds (more than 30)
Apr  4 13:40:57 raspberrypi watchdog[610]: file /var/log/data was not changed in 81 seconds (more than 30)
Apr  4 13:40:58 raspberrypi watchdog[610]: file /var/log/data was not changed in 82 seconds (more than 30)
^C
root@raspberrypi:~# date
Tue  4 Apr 13:41:25 BST 2023
root@raspberrypi:~#
```

To test the system operation in the event of a kernel fault run the below to provoke a kernel panic

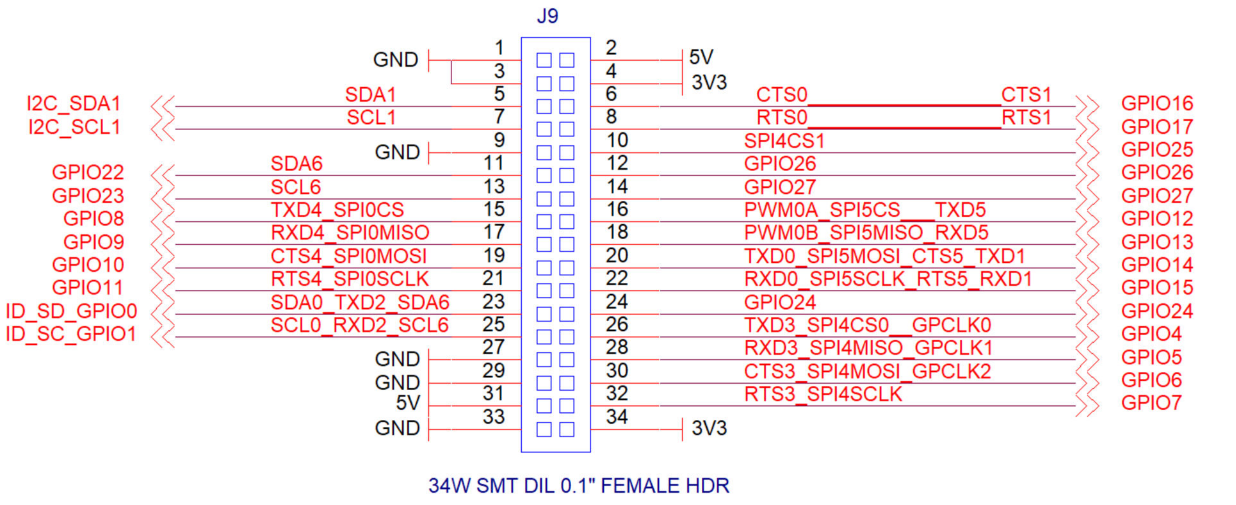
```
# echo c > /proc/sysrq-trigger
```

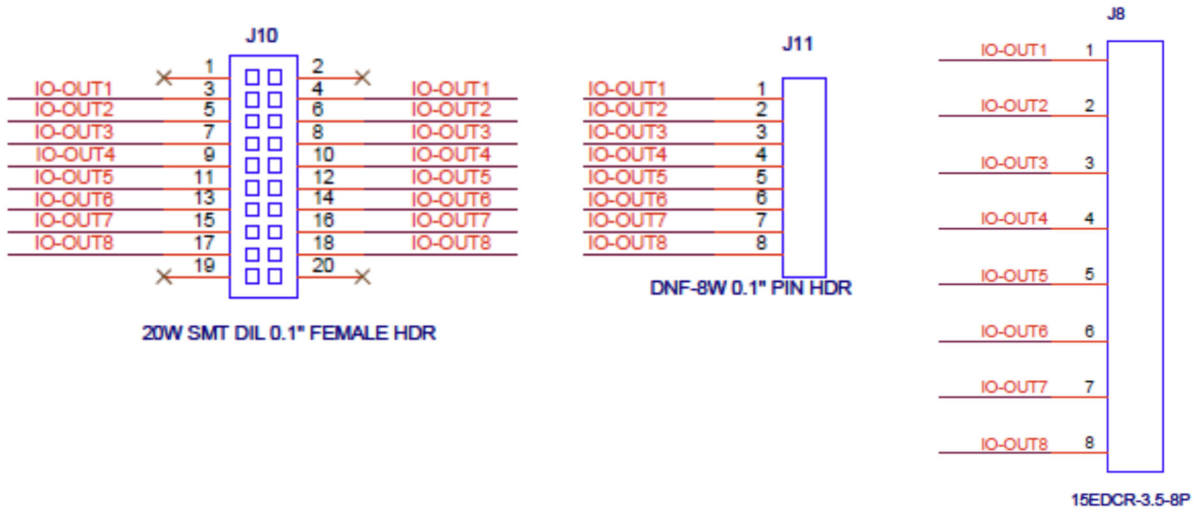
Alternately a recursive "fork bomb" which causes all CPU resources to be used can be provoked using the command below

```
# :(){ :|:& };:
```

GPIO CARD SLOT

The IO Card slot on the board supports a variety of interface cards


J9
GND 1 — 2 5V
3 — 4 3V3
I2C_SDA1 SDA1 5 — 6 CTS0 CTS1 GPIO16
I2C_SCL1 SCL1 7 — 8 RTS0 RTS1 GPIO17
GND 9 — 10 SPI4CS1 GPIO25
GPIO22 SDA6 11 — 12 GPIO26 GPIO26
GPIO23 SCL6 13 — 14 GPIO27 GPIO27
GPIO8 TXD4_SPI0CS 15 — 16 PWM0A_SPI5CS__TXD5 GPIO12
GPIO9 RXD4_SPI0MISO 17 — 18 PWM0B_SPI5MISO_RXD5 GPIO13
GPIO10 CTS4_SPI0MOSI 19 — 20 TXD0_SPI5MOSI_CTS5_TXD1 GPIO14
GPIO11 RTS4_SPI0SCLK 21 — 22 RXD0_SPI5SCLK_RTS5_RXD1 GPIO15
ID_SD_GPIO0 SDA0_TXD2_SDA6 23 — 24 GPIO24 GPIO24
ID_SC_GPIO1 SCL0_RXD2_SCL6 25 — 26 TXD3_SPI4CS0__GPCLK0 GPIO4
GND 27 — 28 RXD3_SPI4MISO_GPCLK1 GPIO5
GND 29 — 30 CTS3_SPI4MOSI_GPCLK2 GPIO6
5V 31 — 32 RTS3_SPI4SCLK GPIO7
GND 33 — 34 3V3
34W SMT DIL 0.1" FEMALE HDR



NOTE : Use of CAM 0 and GPIO0/1 (TX2/RX2) are mutually exclusive

Note that the green 8 way plug in screw terminal connector is uncommitted and is defined by the signals connected to IO-OUT on the 20way connector giving rise to a truly flexible IO interface solution.

Template files for this card can be downloaded from the website and is the same form factor as the CM3 based integrator board.

Note that 'double height' IO cards require 21mm headers minimum to clear the LAN port

Note that the GPIO 3V has a recommended maximum capacity of 500mA (600mA absolute maximum)

GPIO Card Slot Pin Functions

Pin	Signal	ALT0	ALT3	ALT4	ALT5	Pin	Signal	ALT0	ALT3	ALT4	ALT5
1	GND					2	+5V				
3	GND					4	+3.3V				
5	GPIO2	SDA1			SDA3	6	GPIO16		CTS-0		CTS-1
7	GPIO3	SCL1			SCL3	8	GPIO17		RTS-0		RTS-1
9	GND					10	GPIO25				SPI4-CE1
11	GPIO22				SDA6	12	GPIO26				SPI5-CE1
13	GPIO23				SCL6	14	GPIO27				
15	GPIO8	SPI0-CE0	I2CSL CE	TXD-4	SDA4	16	GPIO12	PWM0_0	SPI5-CE0	TXD-5	SDA5
17	GPIO9	SPI0-MISO	I2CSL SDI	RXD-4	SCL4	18	GPIO13	PWM0_1	SPI5-MISO	RXD-5	SCL5
19	GPIO10	SPI0-MOSI	I2CSL SDA	CTS-4	SDA5	20	GPIO14	TXD-0	SPI5-MOSI	CTS-5	TXD-1
21	GPIO11	SPI0-SCLK	I2CSL SCL	RTS-4	SCL5	22	GPIO15	RXD-0	SPI5-SCLK	RTS-5	RXD-1
23	GPIO0	SDA0		TXD-2	SDA6	24	GPIO24				
25	GPIO1	SCL0		RXD-2	SCL6	26	GPIO5	GPCLK1	SPI4-MISO	RXD-3	SCL3
27	GND					28	GPIO6	GPCLK2	SPI4-MOSI	CTS-3	SDA4
29	GND					30	GPIO7	SPI0-CE1	SPI4-SCLK	RTS-3	SCL4
31	+5V					32	GPIO4	GPCLK0	SPI4-CE0	TXD-3	SDA3
33	GND					34	+3.3V				

NOTE : Use of CAM 0 and GPIO0/1 (TX2/RX2) are mutually exclusive

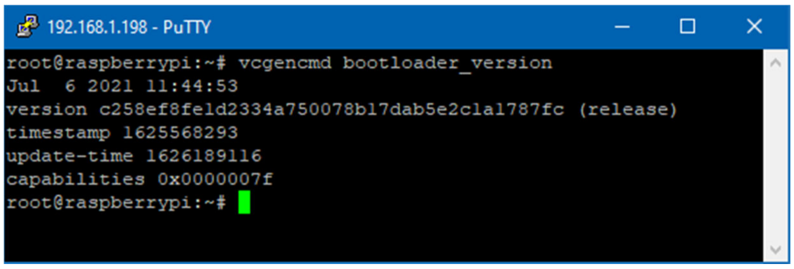
Full Pi GPIO Function Listing

Notes	GPIO	Pull	ALT0	ALT3	ALT4	ALT5
CAMERA 0 | IO SLOT	GPIO0	High	SDA0		TXD-2	SDA6
CAMERA 0 | IO SLOT	GPIO1	High	SCL0		RXD-2	SCL6
I2C1 (IO SLOT)	GPIO2	High	SDA1			SDA3
I2C1 (IO SLOT)	GPIO3	High	SCL1			SCL3
IO SLOT	GPIO4	High	GPCLK0	SPI4-CE0	TXD-3	SDA3
IO SLOT	GPIO5	High	GPCLK1	SPI4-MISO	RXD-3	SCL3
IO SLOT	GPIO6	High	GPCLK2	SPI4-MOSI	CTS-3	SDA4
IO SLOT	GPIO7	High	SPI0-CE1	SPI4-SCLK	RTS-3	SCL4
IO SLOT	GPIO8	High	SPI0-CE0	I2CSL CE_N	TXD-4	SDA4
IO SLOT	GPIO9	Low	SPI0-MISO	I2CSL SDI	RXD-4	SCL4
IO SLOT	GPIO10	Low	SPI0-MOSI	I2CSL SDA	CTS-4	SDA5
IO SLOT	GPIO11	Low	SPI0-SCLK	I2CSL SCL	RTS-4	SCL5
IO SLOT	GPIO12	Low	PWM0_0	SPI5_CE0_N	TXD-5	SDA5
IO SLOT	GPIO13	Low	PWM0_1	SPI5_MISO	RXD-5	SCL5
IO SLOT	GPIO14	Low	TXD-0	SPI5_MOSI	CTS-5	TXD-1
IO SLOT	GPIO15	Low	RXD-0	SPI5_SCLK	RTS-5	RXD-1
WDOG-EN | IO SLOT	GPIO16	Low		CTS-0		CTS-1
WDOG-IN | IO SLOT	GPIO17	Low		RTS-0		RTS-1
On Board TPM	GPIO18	Low			SPI1-CE0	
On Board TPM	GPIO19	Low			SPI1-MISO	
On Board TPM	GPIO20	Low			SPI1-MOSI	
On Board TPM	GPIO21	Low			SPI1-SCLK	
IO SLOT	GPIO22	Low	SD0_CLK	SD1_CLK		SDA6
IO SLOT	GPIO23	Low	SD0_CMD	SD1_CMD		SCL6
IO SLOT	GPIO24	Low	SD0_DAT0	SD1_DAT0		
IO SLOT	GPIO25	Low	SD0_DAT1	SD1_DAT1		SPI4-CE1
IO SLOT	GPIO26	Low	SD0_DAT2	SD1_DAT2		SPI5-CE1
IO SLOT	GPIO27	Low	SD0_DAT3	SD1_DAT3		

DUAL CAMERA

Dual Camera support has the below pre-requisites

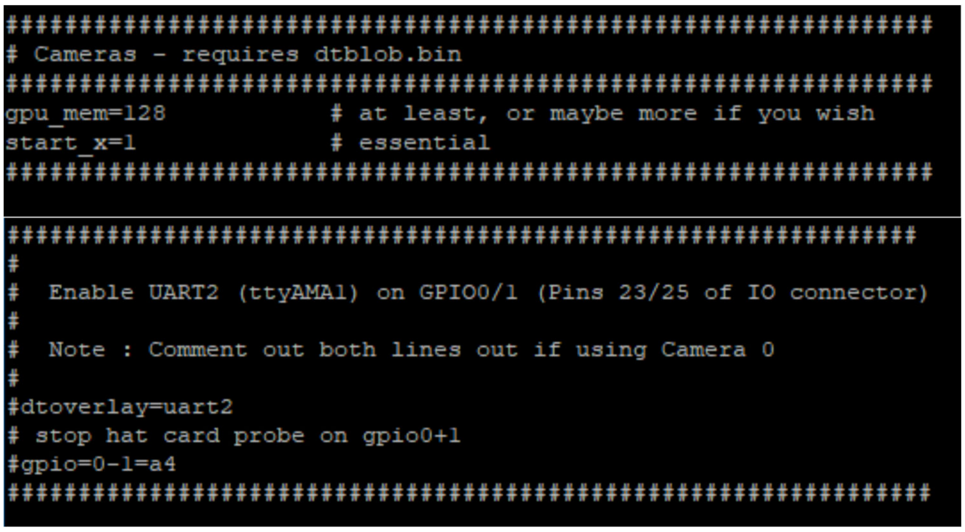
1. **Boot EEPROM Firmware version should be 6th July or later**



192.168.1.198 - PuTTY

```
root@raspberrypi:~# vcgencmd bootloader_version
Jul  6 2021 11:44:53
version c258ef8fe1d2334a750078b17dab5e2c1a1787fc (release)
timestamp 1625568293
update-time 1626189116
capabilities 0x0000007f
root@raspberrypi:~#
```

2. **System config.txt configuration settings for camera usage**

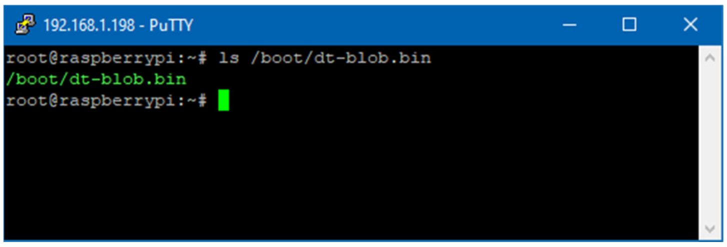


```
################################################################
# Cameras - requires dtblob.bin
################################################################
gpu_mem=128              # at least, or maybe more if you wish
start_x=1                # essential
################################################################
```

```
################################################################
#
#  Enable UART2 (ttyAMA1) on GPIO0/1 (Pins 23/25 of IO connector)
#
#  Note : Comment out both lines out if using Camera 0
#
#dtoverlay=uart2
# stop hat card probe on gpio0+1
#gpio=0-1=a4
################################################################
```

3. **System dt-blob.bin file configuring the camera setup**

This file is located in **/boot** and configures the control lines and interfaces used for camera setup

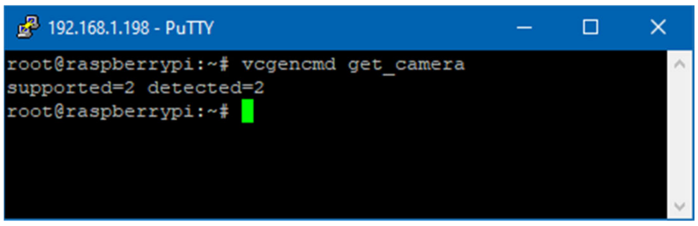


192.168.1.198 - PuTTY

```
root@raspberrypi:~# ls /boot/dt-blob.bin
/boot/dt-blob.bin
root@raspberrypi:~#
```

Reboot the system to complete the changes

With this in place the command below should report back accordingly after the system has rebooted.



```
root@raspberrypi:~# vcgencmd get_camera
supported=2 detected=2
root@raspberrypi:~# 
```

Note : If 2 cameras are configured (as per default image) but only 1 camera is connected it will always be detected as camera 0 regardless of which physical port the camera is plugged into.

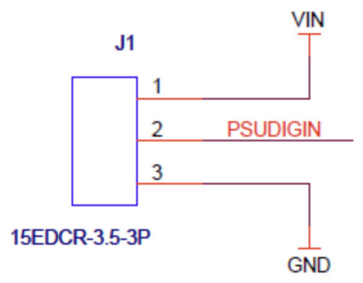
10/100/1000 ETHERNET INTERFACE

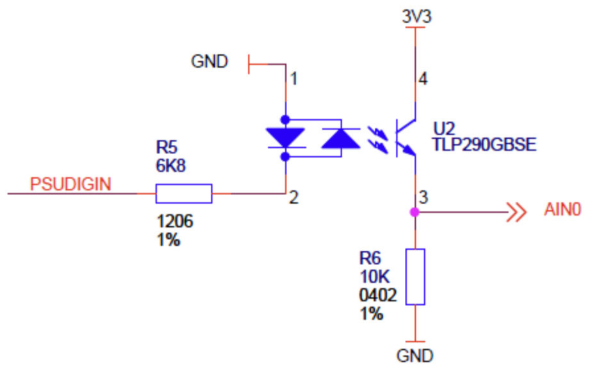
The integrated Gigabit has been brought out to the main face of the card, this takes the place of the RJ45 Serial port on the CM3 Integrator board, so as to allow easy hardware migration.

Note that the Gigabit interface takes its MAC address from the CM4’s serial number.

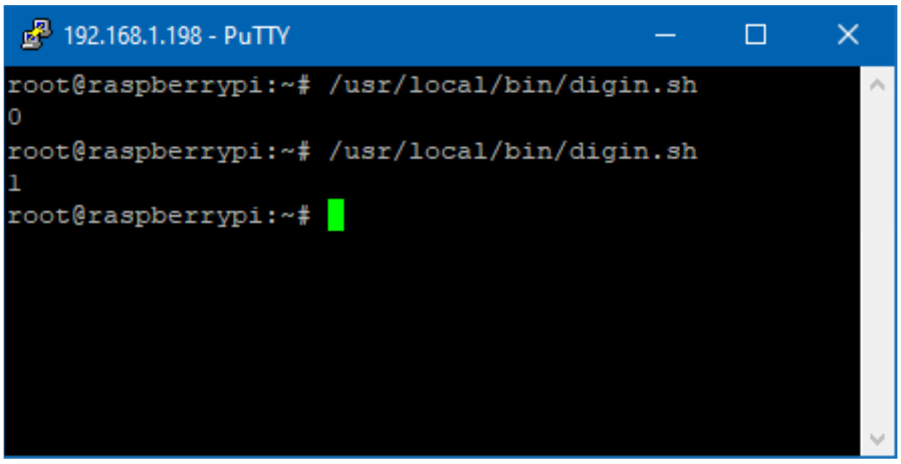
POWER IN DIGITAL INPUT

The Power input connector has an integrated opto-isolated digital input allowing any voltage from 5V up to 48V to be registered as a 3V digital signal on the CPU side. Here we are re-purposing the Analogue In 0 pin of the CM4s Power Management Controller IC





We have created a simple bash script to make the process of reading this input easy and report the logical input state



```
root@raspberrypi:~# /usr/local/bin/digin.sh
0
root@raspberrypi:~# /usr/local/bin/digin.sh
1
root@raspberrypi:~#
```

OS CONFIGURATION FILES

This is a list of the files altered from a base install to create the demo OS image

```
/boot/dt-blob.bin
/boot/config.txt
/boot/cmdline.txt
/boot/overlays/tpm-spi1.dtbo
/boot/overlays/disablepcie.dtbo
/boot/overlays/ext-watchdog.dtbo

/usr/local/bin/camerastat
/usr/local/bin/deleteallsmsmessages
/usr/local/bin/deletemessage
/usr/local/bin/digin.sh
/usr/local/bin/eeprog
/usr/local/bin/fanctl.sh
/usr/local/bin/mbpoll
/usr/local/bin/modemstat
/usr/local/bin/openopc -> /opt/OpenOPC-1.3.1/src/openopc.py
/usr/local/bin/qmi-network-raw
/usr/local/bin/resetbyauthorized.sh
/usr/local/bin/sendsms
/usr/local/bin/smscmd
/usr/local/bin/smscmddemo.sh
/usr/local/bin/smslist
/usr/local/bin/takephoto
/usr/local/bin/tpminfo.sh
/usr/local/bin/tpmtool
/usr/local/bin/tpmupdate
/usr/local/bin/uhubctl
/usr/local/bin/usbpwrctl.sh

/etc/udev/rules.d/8-sdcard.rules
/etc/udev/rules.d/20-modem-ec2x.rules
/etc/udev/rules.d/20-modem-7xxxx.rules
/etc/udev/rules.d/modem-rules/*

/etc/rc.local
/etc/issue
/etc/modules

/etc/init.d/mypi-init.sh
/etc/init.d/hwclock.sh

/opt/OpenOPC-1.3.1/

/root/backups/*
/root/image-create/*
/root/overlays/*
/root/stresscpu.sh
/root/tpm-toolkit/*
/root/watchdog/*
/root/quectel-CM/*
```

RASPBERRY PI DOCUMENTATION

Raspberry Pi have produced a comprehensive knowledge base on how to configure and control various aspects of the Compute Module and it's OS.

https://www.raspberrypi.com/documentation

SCHEMATICS

A reduced schematic set can be provided on request, please contact your technical support representative for more details.

FCC Class A Statement

This equipment has been tested and complies with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

Properly shielded and grounded cables and connectors must be used in order to meet FCC emission limits. Embedded Micro Technology is not responsible for any radio or television interference caused by using other than recommended cables and connectors or by unauthorized changes or modifications to this equipment. Unauthorized changes or modifications could void the user's authority to operate the equipment.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation