MyPi Industrial IoT Edge Gateway

User Guide

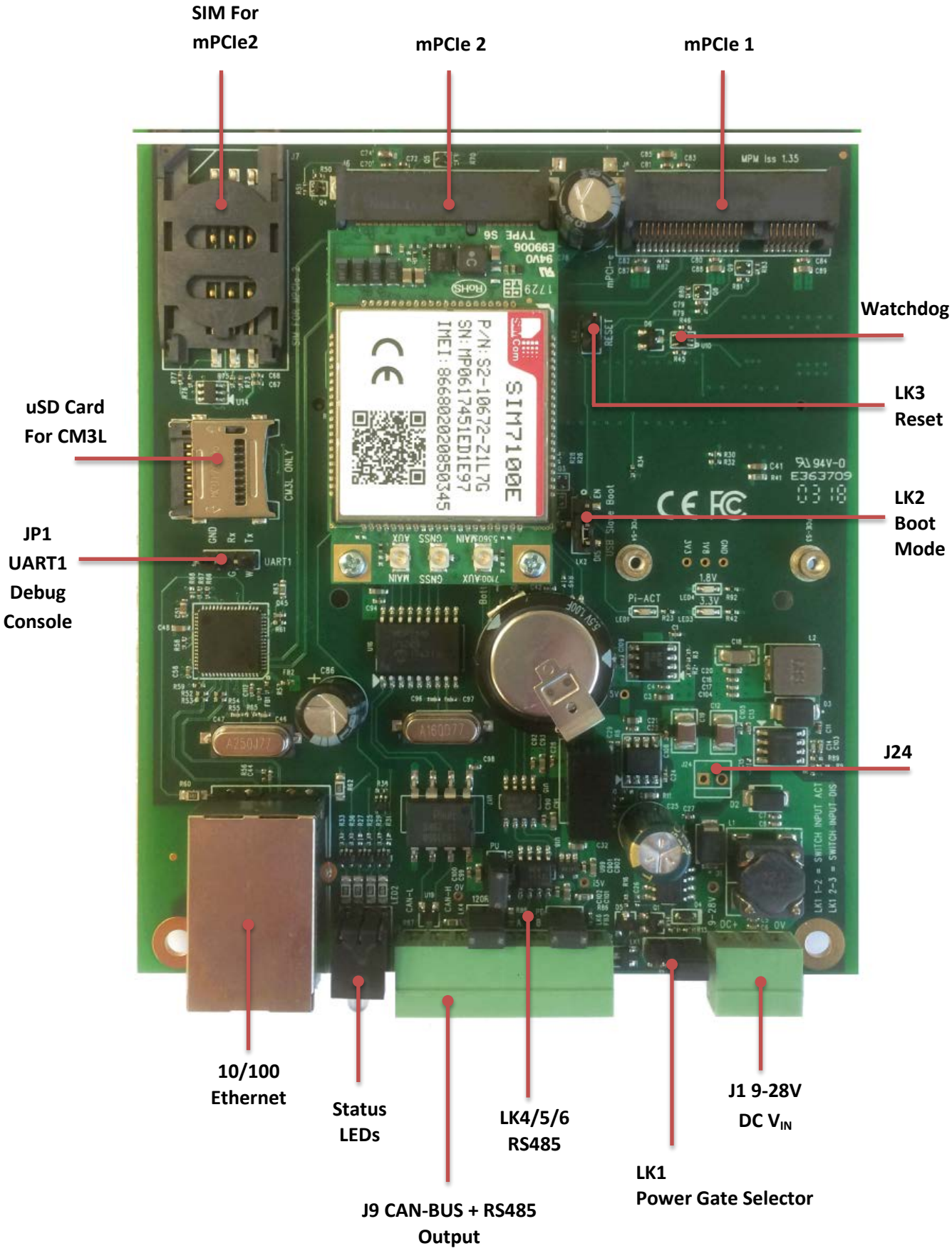
Issue : 1.1

Dated : March 2018

Prepared By : Andrew O'Connell

Features

- 10/100 Ethernet
- Isolated CAN Interface with on-board termination resistor
- Isolated 2-Wire RS485 Interface with transparent hardware flow control and on-board bus pull and termination resistors
- 2 x mPCIe Interface supporting 3G/4G Modems (1 x on-board SIM slot) and USB WiFi N/AC cards as well as Bluetooth/LoRa/Zigbee RF IO modules
- External 1.6s Watchdog
- Switched power input with 'Power Good' input line allowing for controlled power up and safe shutdown operation – ideal for automotive environments
- Supports Compute Module 1/3/3L (on-board SD card slot for CM3L) providing a low power or high performance system
- Wide 7-28V DC power input range, withstanding transient dips to 6V (e.g,. Engine Crank)
- FCC/CE Class A approval
- Wide fan-less operational ambient temperature range -25 to +60 deg C
- Small, Rugged aluminium enclosure 11 x 13 x 3cm with flexible mounting Kit


SIM For
mPCIe2
mPCIe 2
mPCIe 1
Watchdog
LK3
Reset
uSD Card
For CM3L
LK2
Boot
Mode
JP1
UART1
Debug
Console
J24
10/100
Ethernet
Status
LEDs
LK4/5/6
RS485
J1 9-28V
DC V$_{IN}$
LK1
Power Gate Selector
J9 CAN-BUS + RS485
Output

Connector Pin out

Power In (J1)

1 7-28V Vin
2 'Power Good' signal (6-30V Input)
3 0V

IO Plug (J9)

1 CAN L
2 CAN H
3 CAN-BUS 120R Termination Enable A*
4 CAN-BUS 120R Termination Enable B*
5 RS485 – A (D+)
6 RS485 – B (D-)
7 CAN/RS485 Isolated Ground/0V
8 No Connect

*** Connect these 2 pins together to enable the internal CAN termination resistor**

UART1 (JP1) – Serial Console

1 Tx
2 Rx
3 GND

The silkscreen on the PCB shows the Tx/Rx lines as well as "G" (Green) and "W" (White) which corresponds to the wire colours on the USB-TTL UART cable on the dev kit pack.

AUX Power Feed Connector (J24)

1 Switched VIN (after power switch FET)
2 0V

SD Card

This connects only to the RPi Compute Module and can only be used with a CM3L

SIM Card

This connects only to MPCIE2 (Left hand side, nearest to SIM/SD Card)

Link Operation

LK1	**Power Switch FET operation**
1-2	Enable switchable power input
2-3	Disable switchable power input (Power always enabled)*

LK2	**RPi Compute Module eMMC USB programming mode**
1-2	Enabled
2-3	Disabled*

LK3	**System Reset**
1	0V
2	/RESET

LK4 **120R RS485 Termination Resistor Enable***

LK5 **640R RS485 Bus Pull Up Enable***

LK6 **640R RS485 Buss Pull Down Enable***

*Default fitted position as shipped.

For information on how to use the RPi CM programming mode see this link :

https://www.raspberrypi.org/documentation/hardware/computemodule/cm-emmc-flashing.md

Front Panel LEDS

The top Red/Green LEDs are driven by UART0 RX/TX lines to indicate serial comms activity

As shipped the Lower Green Status LED is controlled by the LED output of mPCIe-2 (giving an indication on WiFi/Modem Activity) and GPIO36 controls the front panel lower RED status LED.

A custom build standard changing the top LEDs to being driven by GPIO34/35 is possible, please contact sales for quotation.

GPIO Usage Table

The following table illustrates the system usage of the Pi Module GPIO Lines

All IO line signals pulled to inactive setting as default, in some cases the board is fitted with a matching resistor pull in the same direction as the default internal 50k pull.

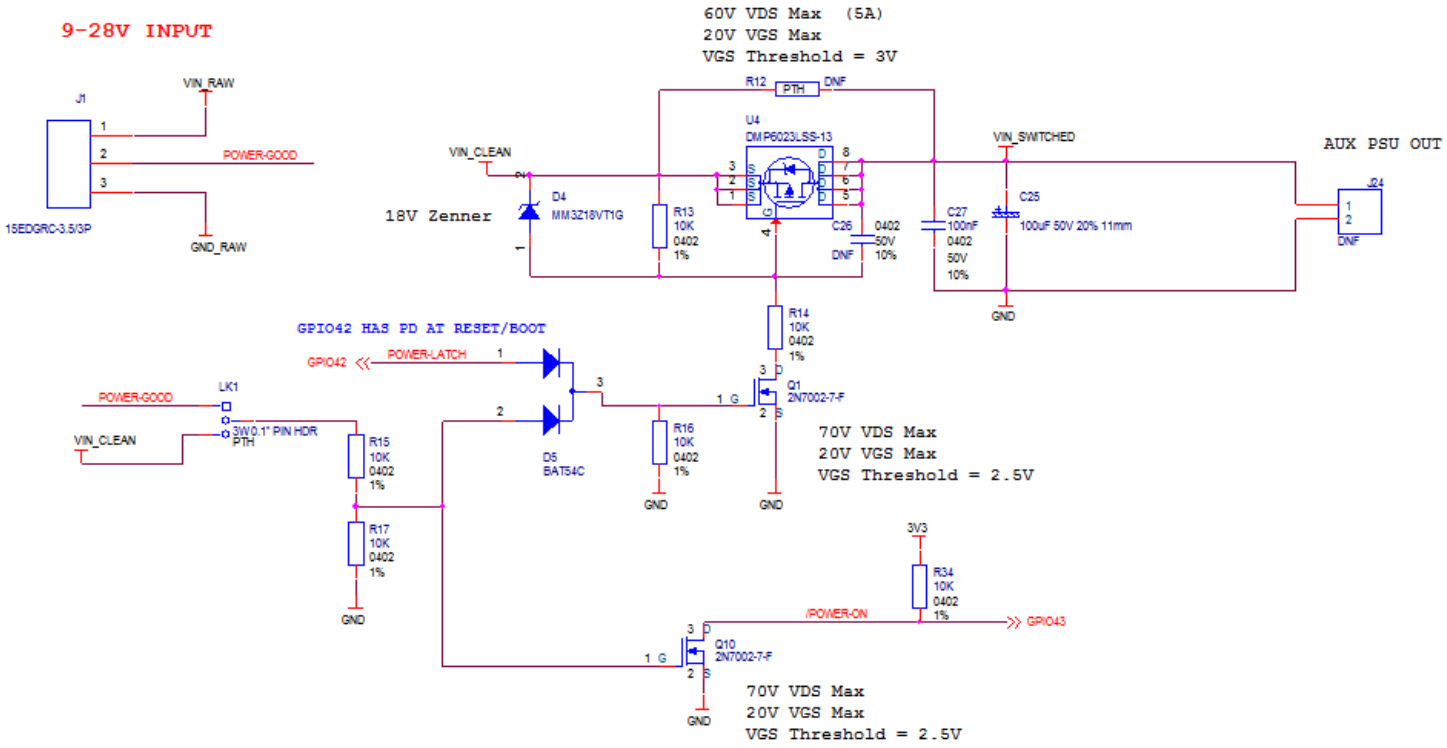
GPIO	Pi PU/PD	Signal
2		I2C-SDA
3		I2C-SCL
5	PU	/CAN RESET
7		SPI0-CE1
8		SPI0-CE0
9		SPI0-MISO
10		SPI0-MOSI
11		SPI0-SCLK
14		UART-0 (ttyAMA0) TX – RS485
15		UART-0 (ttyAMA0) RX – RS485
21	PD	SIERRA RESET**
22	PD	MPCIE 2 RESET
23	PD	MPCIE 2 WDIS
25		CANIRQ
26	PU	WD ENABLE
27	PU	WD INPUT
32		UART-1 (ttyS0) TX – Console
33		UART-1 (ttyS0) RX – Console
34	PU	LED2-RED (Option)
35	PU	LED2-GREEN (Option)
36	PU	LED1-RED
40	PD	MPCIE 1 WDIS
41	PD	MPCIE 1 RESET
42	PD	POWER LATCH
43	PU	/POWER-ON
44	(PU)*	/LAN_RESET

* This line has an external pull up resistor only.

** Some Sierra Wireless models use an alternate reset pin to standard

Switched Power Input Operation

The 'Power Good ' input line on the main power in connector provides a mechanism to allow the system to control power up and graceful power down, this line can accept a wide voltage range from 6-30V DC making it ideal for use in an automotive environment where the unit can run from the raw battery power but be switched on by the usage of the accessory power line which is switched from the main ignition key.



With link LK1 set in position 1-2

The 'Power-Good' input from the main DC power in plug is fed, via the diode wired-OR gate created by D5, through to Q1 which controls whether the power FET is on or OFF.

A DC Input of 6-30V on 'Power-Good' input will enable the main power FET (U4) and the board will power up.

As part of the power up boot sequence the system can be configured using a custom device tree file (as per the demo OS image), or using the latest Raspbian firmware image (as of 28th March 2018) the GPIO overlay command, to condition the initial state of GPIO lines very early in the boot sequence, and well before the Linux kernel is loaded. Using this facility we can effectively latch the power FET on by driving GPIO42 High shortly after power up (usually within 5-10 seconds from power on).

Using Q10/R34 the state of the 'Power Good' line can also be monitored via GPIO43.

As this line is normally pulled high by R34 if GPIO43 is reading a 0 (logic low) then it can be determined that the 'Power Good' input voltage is present, as Q10 is pulling R34 Low.

If GPIO43 reads 1 (logic high) then it can be determined that the ‘Power Good’ input voltage has been disconnected or switched off.

So by a combination of using the ‘Power Latch’ (GPIO42) line to latch the power supply switch FET at start-up and monitoring ‘Power Good’ via (GPIO43) we can check for a power down/switch off event (e.g. Ignition key removed from car) and the system can be put into a controlled shutdown sequence (e.g. Linux command ‘halt’).

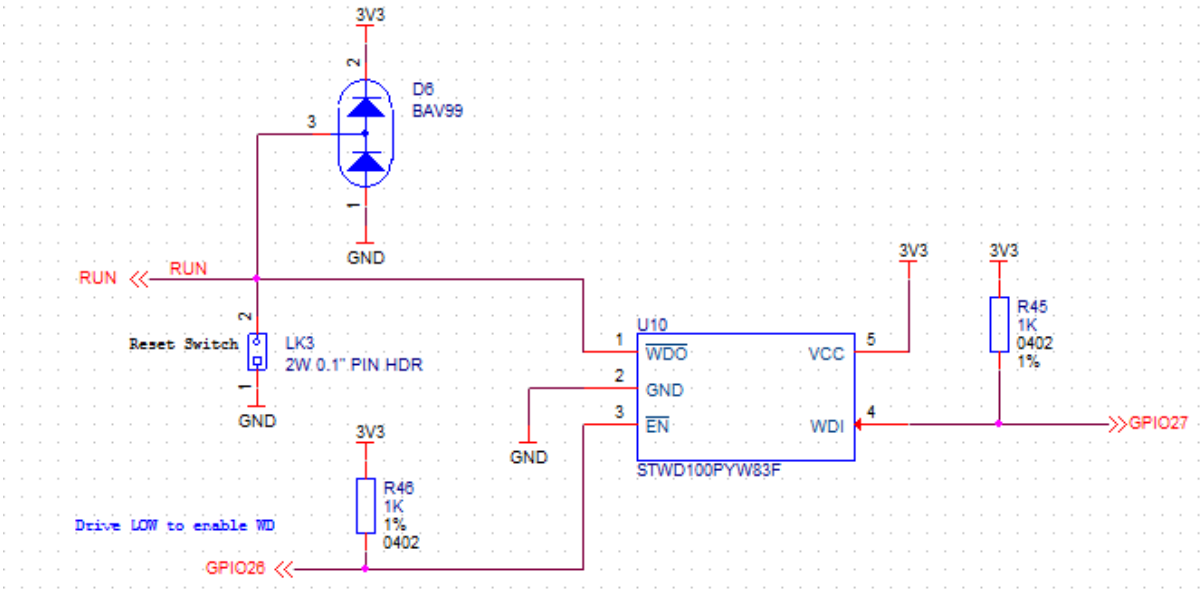
As the default state for the Power Latch (GPIO42) line is logic low the board will power off at the end of the standard Linux shutdown sequence in the same manner as a Desktop PC with an ATX power supply (assuming that Power-Good line remains low for the latter part of the shutdown sequence)

With LK1 set to positions 2-3

The ‘Power Good’ voltage input line is ignored and Q1 is instead fed from the main DC input power, as a result of this the system will only switch off once the main system power feed is removed**.**

If the Linux command halt is used in this case the system will hang at the end of the shutdown sequence as it cannot cut off its own power feed due to ‘Power-Good’ being held at >3V.

Watchdog



The on-board external 1.6 second watchdog is a single chip part provided by **ST STWD100PYW83F**, see this link for part datasheet for full operational specifications.

www.st.com/resource/en/datasheet/stwd100.pdf

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 driven by GPIO26 (/WD Enable) and GPIO27 (/WD Input), by default the watchdog is disabled.

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 (1.6 seconds) and the low level pulse period must be >1uS.

If the device sees a valid low-to-high transition on the input pin the internal 1.6 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, which will also cause GPIO26 (/WD Enable) to be pulled high (as the Pi CPU resets)and so disable the watchdog allowing the system to boot.

With this in mind if the external watchdog is not used a hard reset of the Pi module can be effected by setting WD enable line high and then not toggling the watchdog input line.

The Reset lines for all other devices (including mPCIe) are available via separate, independent GPIO lines.

When the system hard resets in this manner all GPIO lines will revert back to their default state, which will have implications if the gated power input facility is in use.

mPCIe Options & Compatibility

The mPCIe sockets 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	Reserved	40	GND
41	Reserved	42	LED_WWAN#
43	Reserved	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

For mPCIe-2 (Left hand side nearest SIM Socket) pins 42/43 are connected to the bottom Green LED driver.

For mPCIe-1 (Right hand side) pins 8/10/12/14/16 (SIM) are not connected

Pins 20/22 on both sockets are connected to GPIOs to provide software controllable access to Reset and Wireless Disable lines.

For mPCIe-2 Pin 33 is connected to GPIO21 for newer Sierra Wireless Modems (modem reset line)

Modems

See the below link to pages from the main modem documentation section

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

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)
- ZIGBEE/802.15.4 (Silicon Labs/Telegesis RX357 Module L.R. UFL)
- XBEE (no RF module included)

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/**

```
SUBSYSTEMS=="usb",ATTRS{busnum}=="1",ATTRS{product}=="FT230X Basic UART",ATTRS{idProduct}=="6015",ATTRS{manufacturer}=="FTDI",SYMLINK+="ttyS1"
```

This one line creates a symlink for the FTDI serial port called **/dev/ttyS1**

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

Configuration Files

There are a handful of files needed to configure the MyPi-Mini from the base Jessie install, these are available from this link :

https://drive.google.com/open?id=0B8gkYppHOJNMbTk1dFo4ODBGVTg

Filename	File location	Description
config.txt	/boot	System Configuration File
cmdline.txt	/boot	Kernel boot command line options
dt-blob.bin	/boot	Custom Device tree File to latch GPIO42 high at startup
mypi.sh	/etc/init.d	Bash Script to configure Linux OS GPIO exports for command line use & create /dev shortcuts.

Config.txt controls which on-board peripherals are enabled, there should be no need to alter this unless the user wants to alter CPU speed for optimising thermal/power operation.

cmdline.txt this controls what kernel command line options are enabled at boot, again there should be no need to alter this from the default unless you wish to disable the serial console.

dt-blob.bin by placing this in /boot/ this overrides the default device tree file for the CM3, the only difference being that GPIO42 is pulled high immediately at start-up (and before the OS is loaded) to latch the power switch FET on.

mypi.sh creates the below exports and shortcuts, the demo image has this enabled to run during the startup sequence

```
/dev/can-reset              -> /sys/class/gpio/gpio5/value
/dev/lan-disable            -> /sys/class/gpio/gpio44/value
/dev/mpcie1-reset           -> /sys/class/gpio/gpio41/value
/dev/mpcie1-wdisble         -> /sys/class/gpio/gpio40/value
/dev/mpcie2-reset           -> /sys/class/gpio/gpio22/value
/dev/mpcie2-sierra-reset    -> /sys/class/gpio/gpio21/value
/dev/mpcie2-wdisble         -> /sys/class/gpio/gpio23/value
/dev/power-good             -> /sys/class/gpio/gpio43/value
/dev/power-latch            -> /sys/class/gpio/gpio42/value
/dev/wd-enable              -> /sys/class/gpio/gpio26/value
/dev/wd-input               -> /sys/class/gpio/gpio27/value
```

Example usage :

```
echo 1 >/dev/lan-disable

echo 1 >/dev/mpcie1-reset
echo 0 >/dev/mpcie1-reset

cat /dev/power-good
0
```

For more information load the **mypi.sh** file into a text editor.

For **resin.os** you should place **dtblob.bin** into the FAT boot area on the host file system, the other config.txt variables can then be set inside the docker image along with the mypi.sh script.