Configuring the Raspberry Pi for the UPS

To enable to Raspberry Pi to detect the presence of the UPS and to enable it to shut itself and the UPS down in an orderly fashion changes have to be made to the operating system. The information below will enable you to make these changes yourself. The operating system is assumed to be Raspbian.

This is just one example of how this functionality can be implemented. We have purposefully refrained from using tools like python and relied on the bash shell and cron as these two tools will always be present.

The following scripts assume that the signalling input (from the UPS) is pin GPIO4 and the output (to the UPS) is pin GPIO14 (see the diagram below).

Note that you have to be root to create or modify these files. The files can be created or modified with your favourite text editor like *vi*, *vim*, *emacs* or *nano*.

You can learn more about the GPIO by visiting:

`http://elinux.org/RPi_Low-level_peripherals#General_Purpose_Input.2FOutput_.28GPIO.29`

1. Create file /usr/local/sbin/sdc.sh

Note: you might have to create /usr/local/sbin if it doesn't yet exist. Alternatively, the scripts can be placed in any other directory that is convenient and the paths in the following scripts updated accordingly.

```
#!/bin/bash
SDC=/var/run/sdc; export SDC
# this is the number of minutes of continuously running on battery power before
# the RPi will shut itself down. Feel free to change this. On a fully charged
# battery the processor should run for about 2 hours
SDCT=10; export SDCT
LOG=/var/log/ups.log; export LOG
```

Then make this shell script executable with the following command:

```
chmod +x /usr/local/sbin/sdc.sh
```

2. Modify the existing /etc/rc.local

Add the following lines right before the final line with 'exit 0'

```
#!/bin/bash
# monitor GPIO pin 4 (wiringPi pin 7) for shutdown signal
# export GPIO pin 4 and set to floating input
echo "4" > /sys/class/gpio/export
echo "in" > /sys/class/gpio/gpio4/direction
echo "off" > /sys/class/gpio/gpio4/direction
echo "14" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio14/direction
#
#
. /usr/local/sbin/sdc.sh
echo "Creating countdown timer" >> $LOG
echo $SDCT > $SDC
```

3. Create file /usr/local/sbin/high14.sh

```
#!/bin/bash
echo "1" > /sys/class/gpio/gpio14/value
```

Then make this shell script executable with the following command:

```
chmod +x /usr/local/sbin/high14.sh
```

4. Create file /usr/local/sbin/low14.sh

```
#!/bin/bash
echo "0" > /sys/class/gpio/gpio14/value
```

Then make this shell script executable with the following command:

```
chmod +x /usr/local/sbin/low14.sh
```

5. Create file /usr/local/sbin/shutdown_timer.sh

This file is executed from the root cronjob every minute and monitors whether the mains power to the UPS is still present

```
#!/bin/bash
# set some variables
. /usr/local/sbin/sdc.sh
# $SDC is the count-down timer. If it does not exist we have just been rebooted, so
create it and initiate the monitoring pins on the GPIO
if [ ! -s $SDC ]
then
        # monitor GPIO pin 4 for shutdown signal from the UPS
        # export GPIO pin 4 and set to floating input
        echo "4" > /sys/class/gpio/export
        echo "in" > /sys/class/gpio/gpio4/direction
        echo "off" > /sys/class/gpio/gpio4/direction
        echo "14" > /sys/class/gpio/export
```

```
        echo "out" > /sys/class/gpio/gpio14/direction
        echo "Creating countdown timer" >> $LOG
        echo $SDCT > $SDC
fi
# wait for pin to go low
if [ `cat /sys/class/gpio/gpio4/value` -eq '0' ]
then
        SDTIME=`cat $SDC`; export SDTIME
        echo "On battery power. Shutting down in $SDTIME minutes" >> $LOG
        if [ $SDTIME -eq '0' ]
        then
                echo "SHUTTING DOWN NOW *****" >> $LOG
                rm $SDC
                /sbin/shutdown -h now &
                exit 1
        else
                echo $((SDTIME-1)) > $SDC
        fi
else
        /usr/local/sbin/high14.sh
        echo $SDCT > $SDC
fi
```

Then make this shell script executable with the following command:

```
chmod +x /usr/local/sbin/shutdown_timer.sh
```

6. Edit the crontab file for root

Edit the crontab file by issuing the command

```
crontab -e
```

and add this entry

```
*       *       *       *       *     /usr/local/sbin/shutdown_timer.sh > /dev/null
```

7. Edit /etc/init.d/umountfs

Change the following lines:

```
if [ "$REG_MTPTS" ]
then
        if [ "$VERBOSE" = no ]
        then
                log_action_begin_msg "Unmounting local filesystems"
                fstab-decode umount -f -r -d $REG_MTPTS
                log_action_end_msg $?
        else
                log_daemon_msg "Will now unmount local filesystems"
                fstab-decode umount -f -v -r -d $REG_MTPTS
                log_end_msg $?
        fi
fi
```

To (adding the commands in blue):

```
if [ "$REG_MTPTS" ]
then
        # Last check to see whether power has returned at the last second.
        # Our only option is to reboot if that happens
        . /usr/local/sbin/sdc.sh
        if [ "`cat /sys/class/gpio/gpio4/value`" -eq '1' ]
        then
                echo "Power has returned. Re-booting" >> $LOG
                echo "Power has returned. Re-booting"
                sleep 1
                /sbin/reboot
        fi
        # setting the pin low will cause the UPS to shut itself down in
        # approximately 3 seconds
```

```
        /usr/local/sbin/low14.sh
        if [ "$VERBOSE" = no ]
        then
                log_action_begin_msg "Unmounting local filesystems"
                fstab-decode umount -f -r -d $REG_MTPTS
                log_action_end_msg $?
        else
                log_daemon_msg "Will now unmount local filesystems"
                fstab-decode umount -f -v -r -d $REG_MTPTS
                log_end_msg $?
        fi
fi
```