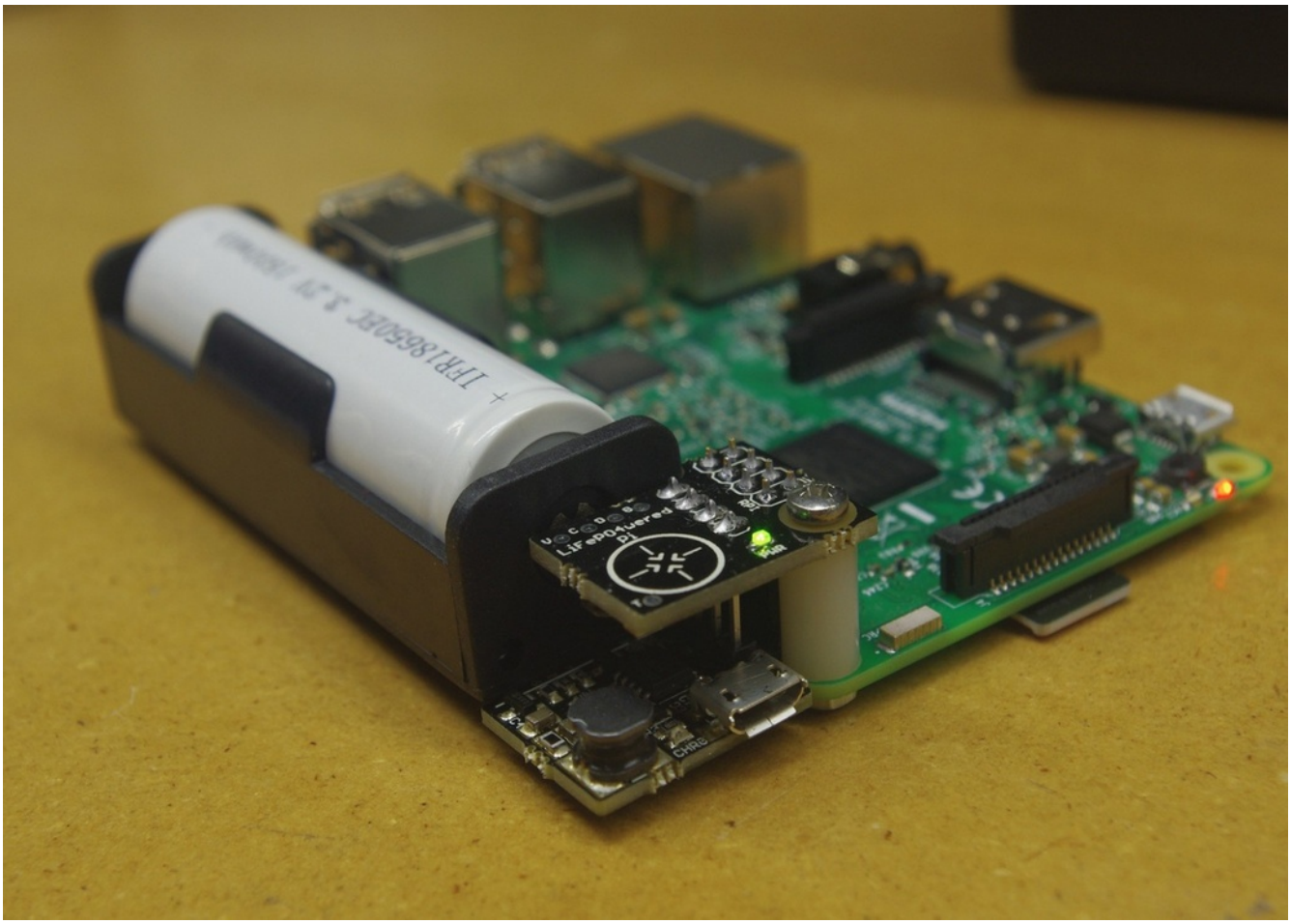


Introduction

The LiFePO4wered/Pi3™ is a high performance battery power system for the Raspberry Pi. It can power a Raspberry Pi for 1 to 9 hours from the battery (depending on Raspberry Pi model, attached peripherals and system load) and can be left plugged in continuously. It features:

- A single 3.2 V, 1500 mAh $LiFePO_4$ (lithium iron phosphate) cell, providing high power density, extended cycle life (2000+ cycles), and safety from fire and explosions.
- A smart USB charge controller with over-charge protection, allowing the device to stay plugged in and provide UPS (Uninterruptible Power Supply) functionality.
- Smart auto-adjusting charge current that allows the charger to draw up to 1.33 A when used with high power chargers, but automatically reduce the current to not overload lower power

sources such as a PC USB port.

- MPPT functionality that makes it possible to charge the LiFePO4wered/Pi3™ from 5 – 7 V nominal solar cells.
- Two-way communication between the power system and the Raspberry Pi over I^2C bus (to monitor voltages and customize settings) and Raspberry Pi shutdown detection.
- A smart power manager controller and open source daemon that work in tandem to provide over-discharge protection and clean shutdown functionality.
- Continuous measurement of USB input voltage, battery voltage and load voltage with smart user programmable thresholds for boot, shutdown and hard power down.
- A touch pad with programmable parameters that provides clean boot/shutdown capability even in headless setups.
- Press-and-hold touch pad for protection against accidental activation and with the ability to also monitor the touch pad in user software.
- A green PWR LED that indicates the Raspberry Pi power state, gives user feedback and can even be controlled by user code.
- A separate red CHRG LED that indicates charging state.
- A wake timer allowing the Raspberry Pi to be off most of the time for low duty cycle applications.
- An auto-boot feature to maximize uptime by making the Raspberry Pi run whenever there is sufficient battery power or when the USB input voltage is present.
- An auto shutdown feature that makes it possible to automatically shut down the Raspberry Pi when the USB input voltage has been removed for a programmable amount of time.
- Out of the box compatibility with Raspberry Pi Model A+, Model B+, Raspberry Pi 2, Raspberry Pi 3 and Raspberry Pi Zero.
- Compatibility with original Raspberry Pi Model A and Model B with additional wiring.

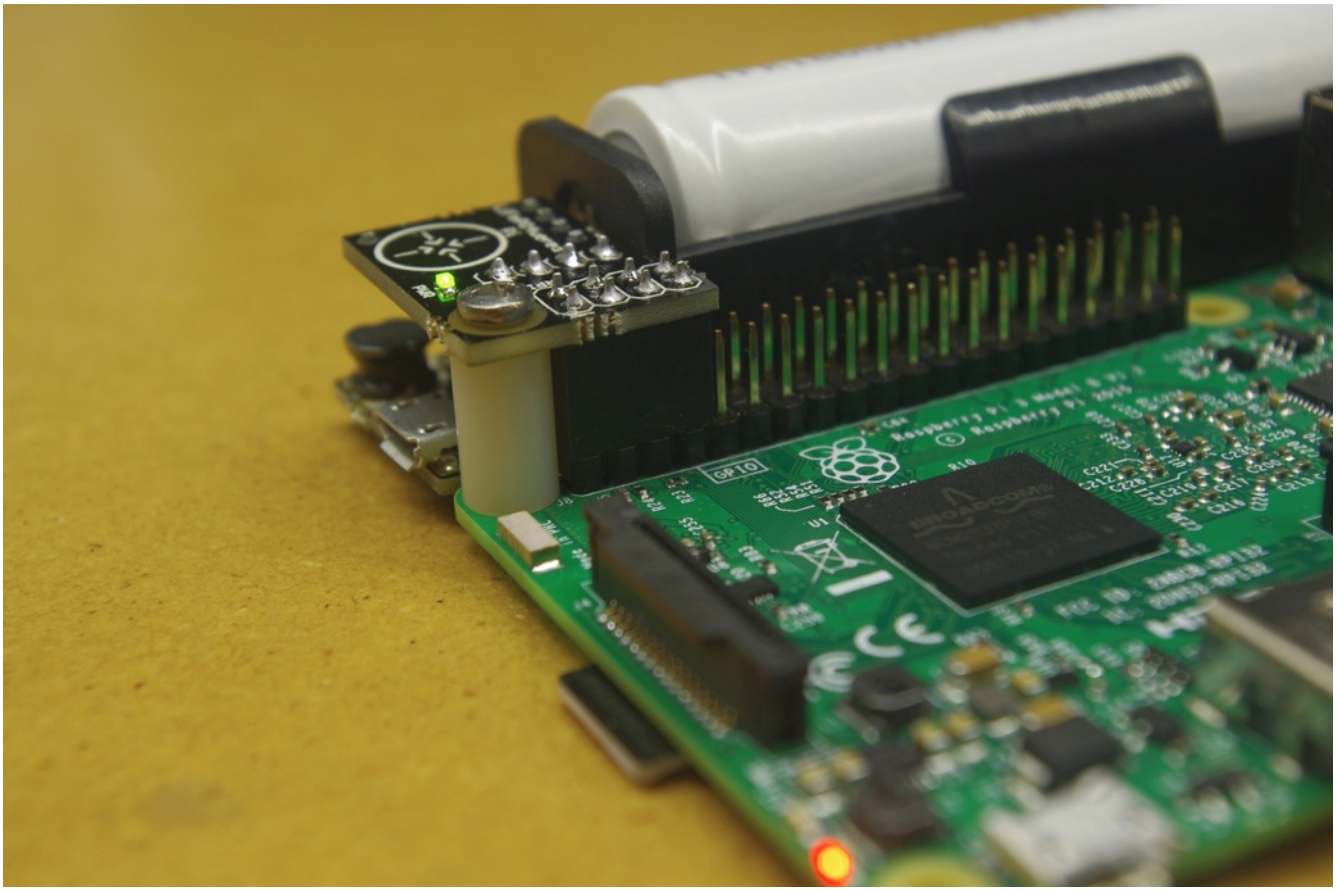
Hardware installation

The LiFePO4wered/Pi3™ is designed to connect to the first 8 pins of the Raspberry Pi GPIO header. In case of the Pi Zero (which doesn't come with a header populated), it is necessary to first install a header

for at least the first 8 GPIO pin locations.

The LiFePO4wered/Pi3™ has a single mounting hole which lines up with the mounting hole on the Raspberry Pi. For mechanical stability, it is recommended to mount the LiFePO4wered/Pi3™ to the Raspberry Pi using a 16 mm minimum length M2.5 machine screw, M2.5 nut and a 7/16” length, number 4 screw size nylon spacer which maintains the correct distance without putting stress on the GPIO header connections.

The image below shows a correctly installed LiFePO4wered/Pi3™.



Software installation

The LiFePO4wered/Pi3™ requires software to be running on the Raspberry Pi to operate correctly. This software provides a daemon that automatically manages the power state and shutdown of the Raspberry Pi, a shared library that allows integration of LiFePO4wered/Pi3™ functionality in user programs, and a CLI (command line interface) program that allows the user to easily configure the

LiFePO4wered/Pi3™ or control it from shell scripts.

After the LiFePO4wered/Pi3™ hardware is connected as described in the hardware installation section, the Raspberry Pi can be turned on by pressing and holding the touch button until the green PWR LED starts to pulse. The Raspberry Pi will boot up, but the LED will keep pulsing instead of going to the steady on state. This is normal and happens because the LiFePO4wered/Pi3™ software has not yet been installed on the Raspberry Pi.

The user can find the LiFePO4wered/Pi3™ software package on Github at:

https://github.com/xorbit/LiFePO4wered-Pi

The "Clone or download" button provides the option to download a ZIP file or clone the software using Git. It is recommended to use Git since this makes updating the software easier. This can be done by opening a terminal window on the Raspberry Pi (using a local interface or over SSH), and running the following command to first ensure the build tools and Git are installed:

```
sudo apt-get -y install build-essential git
```

Then clone the software package to a location where you keep source software packages:

```
git clone https://github.com/xorbit/LiFePO4wered-Pi.git
```

Now go into the newly created LiFePO4wered-Pi directory by running:

```
cd LiFePO4wered-Pi
```

In the source project directory, you can now build the software by running:

```
python build.py
```

This will create the binaries to be installed on the system. These can be installed by running:

```
sudo ./INSTALL.sh
```

This will not only install the software to the Raspberry Pi system, but also perform any necessary configuration changes such as enabling the I^2C bus and enabling the GPIO UART.

If the I^2C bus had already been enabled before, the LiFePO4wered/Pi3™ PWR LED should now go on solid, because the install script also starts the daemon. The LiFePO4wered/Pi3™ is now fully operational.

If the PWR LED does not yet go on solid, it is likely that the I^2C was not yet enabled before the installer was run, and a reboot is required to enable all LiFePO4wered/Pi3™ functionality. This can be done by running:

```
sudo reboot
```

After reboot, the LiFePO4wered/Pi3™ daemon should be started and the PWR LED should go on solid to indicate the system is on.

Basic usage

In the basic use case, the user does not need to interact with the LiFePO4wered/Pi3™ software on the Raspberry Pi at all once it is installed. The only necessary user interaction is with the touch button, with feedback provided by the green PWR LED.

To use the system as a basic power manager, just keep a 5 V USB charger connected to the LiFePO4wered/Pi3™ micro USB, like you normally would have it connected to the Raspberry Pi's own micro USB. The Raspberry Pi's micro USB should remain unconnected (no damage will occur if you connect it, but the LiFePO4wered/Pi3™ will not be able to control the Raspberry Pi's power).

The LiFePO4wered/Pi3™ touch button can be used to turn the Raspberry Pi on and off. The touch button needs to be pressed and held to take effect. During this press-and-hold delay, the PWR LED glow will ramp up. The press-and-hold delay is implemented to prevent accidental activation when handling the system.

Once the system is booting or shutting down, the LiFePO4wered/Pi3™ cannot respond to more touch input until the Raspberry Pi reaches the desired state (on or off). The changing of state (booting or shutting down) is indicated by the slow pulsing of the PWR LED, which indicates the system is “busy”. If the user touches the button during this time, the PWR LED will do a quick flashing sequence to indicate it cannot comply with the user request at that time. Once the Raspberry Pi reaches a steady state (on or off), the user can interact with the touch button again.

If the power going to the LiFePO4wered/Pi3™ is disconnected or fails, the system will keep running from the battery for 1 to 8 hours, depending on the Raspberry Pi model, attached peripherals and system load. If the power returns during that time, the battery will be recharged and the system will not experience any down time. If the battery power runs out before the power returns, the LiFePO4wered/Pi3™ will instruct the Raspberry Pi to do a shutdown, and once the system is shut down properly, the power will be turned off.

In the default configuration, the system will not be automatically booted when power returns, but the user is expected to turn the system back on using the touch button. It is possible to make the Raspberry Pi boot again automatically when power returns by configuring the AUTO_BOOT setting in the configuration. This is ideal for unattended systems that need to provide maximum uptime.

If the user attempts to turn on the Raspberry Pi by touching the button when the battery is depleted, the

PWR LED will do a quick flashing sequence to inform the user that the system cannot comply with the request. The same happened if the LiFePO4wered/Pi3™ detects that Raspberry Pi is already powered from another source.

Since the LiFePO4wered/Pi3™ ensures that the Raspberry Pi is always shut down in a proper way before power is removed, no matter what the reason for the shutdown is, the file systems are always properly unmounted and left in a clean state. This will go a long way in preserving reliable system operation and preventing SD card corruption, which often is a result of removing power while the system is running.

Limitations

Power source dependence

The LiFePO4wered/Pi3™ was designed specifically to provide continuous UPS functionality for a Raspberry Pi 3 under high load. The design was tested to be able to charge the battery while powering a Pi 3 with 100% CPU load on all 4 cores, with an active Ethernet connection and a USB flash drive plugged in. All tests were done with a high quality 2 A power source and high quality USB cable.

As was mentioned, the charger automatically adjusts its charge current to the power available from the connected source. This means you can power the LiFePO4wered/Pi3™ from a PC USB port for instance, but the charge current will be reduced to 0.5 A typically and the system won't be able to keep the battery charged under the high load described above. It is the user's responsibility to provide a power source sufficient for the expected system load.

Another matter of concern is the USB cable used. There are unfortunately many low-grade micro USB cables on the market, often using very thin gauge wire. When drawing high currents through such a cable, a substantial amount of voltage will be dropped and the LiFePO4wered/Pi3™ will again limit the current to not make the voltage drop below the MPPT threshold. This can severely limit the charge current even when a good power supply is used.

Battery run time

The run time on battery power depends on many factors, so only general guidelines can be given to help set expectations. In general a $LiFePO_4$ cell will have more capacity when discharged at a lower rate. A cell will also lose some capacity as it ages, but this effect is small in $LiFePO_4$ cells compared to most other lithium chemistries. The cell manufacturer specifies that the cell should still have 80% of its original capacity after 2000 cycles. Also note that the cell will have increased self-discharge at higher temperatures. When deployed in high temperature environments, make sure the cell is charged

regularly so it doesn't discharge far enough to cause permanent damage (< 2 V).

The following scenarios can be used as a reference to estimate battery run times for the LiFePO4wered/Pi3™:

- Raspberry Pi 3 with 4 cores @ 100% load + Ethernet: 1 hour
- Raspberry Pi 3 idle + WiFi: 3 hours
- Pi Zero idle: 9 hours

Electrostatic discharge

In dry conditions, electrostatic charge can build up in the human body and this charge will be discharged into conductive systems such as the LiFePO4wered/Pi3™ when the user touches them.

While no reports of permanent damage due to electrostatic discharge have been received, it is possible that such a discharge will reset the microcontroller on the LiFePO4wered/Pi3™, cutting power to the Raspberry Pi abruptly without doing a proper shutdown first. In dry climates and during dry seasons, it is therefor recommended that the user first discharge before interacting with the LiFePO4wered/Pi3™.

Bidirectional load switch

As was mentioned, the LiFePO4wered/Pi3™ incorporates a bidirectional load switch which will protect the LiFePO4wered/Pi3™ from damage in case the Raspberry Pi is powered from another source such as its own micro USB power connector. However, this switch only works correctly if the LiFePO4wered/Pi3™ is powered (the battery is present). Applying power to the Raspberry Pi with the LiFePO4wered/Pi3™ connected but the battery removed will expose the LiFePO4wered/Pi3™ to voltages that can cause permanent damage.

Software interface

The LiFePO4wered/Pi3™ exposes a set of registers that can be accessed from the Raspberry Pi through the I2C bus. By default, the 7-bit device address is 0x43. This can be changed in case of a conflict, but keep in mind that the library, daemon and CLI will need to be adjusted and recompiled to access the LiFePO4wered/Pi3™ at any other address.

Low level I2C register specification

The following I2C registers are available in the LiFePO4wered/Pi3™:

I2C_REG_VER

1 byte, register address 0x00, read only access

Value: 0x04

This value specifies an I^2C register set version. It allows the client to choose the correct register addresses.

I2C_ADDRESS

1 byte, register address 0x01, read/write access, saved to flash

Default value: 0x43

7-bit bus address of the LiFePO4wered/Pi3™ device. If this is changed, the software on the Raspberry Pi needs to be changed and recompiled to match the new value.

LED_STATE

1 byte, register address 0x02, read/write access, saved to flash

Default value: 0x01

This byte can be used to set the PWR LED state when the Raspberry Pi is on. The LED is under control of the LiFePO4wered/Pi3™ when the system is off (LED off), booting (LED pulsing) or shutting down (LED pulsing). When the Raspberry Pi is on, by default the LED is on solid, but this can be changed. Possible reasons to do so are to save power for maximum run time or to indicate the state of a user program. Possible values are: 0x00 (LED off), 0x01 (LED on), 0x02 (LED pulsing) or 0x03 (LED fast flash).

TOUCH_STATE

1 byte, register address 0x23, read only

Value is 0 if touch pad is not currently touched. Value is nonzero when indicating touch states. Press and hold of the touch button will make the Raspberry Pi turn off, but short touch events can be interpreted by user code. The 4 lowest bits of this byte indicate the last 4 touch button samples, shifting from the low to the high bit. For instance, a value of 0x01 indicates the user just started touching the touch pad, while 0x0E indicates the touch pad was just released after it had been held for at least 3 system ticks.

TOUCH_CAP_CYCLES

1 byte, register address 0x03, read/write access, saved to flash

Default value: 20

The total number of charge and discharge cycles generated and measured by the touch detection subsystem. This is one of the touch parameters that can be customized in case sensitivity needs to be adjusted.

TOUCH_THRESHOLD

1 byte, register address 0x04, read/write access, saved to flash

Default value: 12

Internally, a low pass filtered baseline is maintained that follows the average touch reading level. For a touch to be detected, the current touch reading has to exceed the baseline plus the touch threshold plus the touch hysteresis (see below). For the touch detection to become inactive, the current touch reading has to fall below the baseline plus the touch threshold minus the touch hysteresis. This is one of the touch parameters that can be customized in case sensitivity needs to be adjusted.

TOUCH_HYSTERESIS

1 byte, register address 0x05, read/write access, saved to flash

Default value: 2

The touch detection system has a hysteresis to ensure reliable touch detection performance. The hysteresis is added to and subtracted from the touch threshold, depending on whether an active touch is detected. This is one of the touch parameters that can be customized in case sensitivity needs to be adjusted.

DCO_RSEL

1 byte, register address 0x06, read/write, saved to flash

Default value: factory calibrated

This value is factory calibrated so the microcontroller clock runs at 12 MHz. Refer to the MSP430G2231 datasheet for more details. The user should not need to change this value.

DCO_DCOMOD

1 byte, register address 0x07, read/write, saved to flash

Default value: factory calibrated

This value is factory calibrated so the microcontroller clock runs at 12 MHz. Refer to the

MSP430G2231 datasheet for more details. The user should not need to change this value.

VIN

2 bytes little endian, register address 0x21, read only

Value: input (USB) voltage, 10-bit value, 9.67 V full scale, 9.44 mV per LSB

This value represents the input (USB) voltage. The Raspberry Pi software package contains scaling code so the value is converted to mV for convenience.

VBAT

2 bytes little endian, register address 0x1D, read only

Value: battery voltage, 10-bit value, 5 V full scale, 4.88 mV per LSB

This value represents the battery voltage. The Raspberry Pi software package contains scaling code so the value is converted to mV for convenience.

VOUT

2 bytes little endian, register address 0x1F, read only

Value: output (Raspberry Pi supply) voltage, 10-bit value, 5.548 V full scale, 5.42 mV per LSB

This value represents the output (Raspberry Pi supply) voltage. The Raspberry Pi software package contains scaling code so the value is converted to mV for convenience.

VBAT_MIN

2 bytes little endian, register address 0x08, read/write, saved to flash

Default value: 584 (corresponding to 2.85 V, 4.88 mV per LSB)

This value determines the minimum battery voltage. If the input voltage falls below this value, the LiFePO4wered/Pi3™ will immediately shut the Raspberry Pi power off so no damage occurs to the battery. Note that this is an emergency procedure which normally doesn't occur, the Raspberry Pi should have been given a command to shut down at a higher battery voltage, but in case the Raspberry Pi fails to shut down, this is provided as a safety feature.

The Raspberry Pi software package contains scaling code so the value can be read and set in mV for convenience.

VBAT_SHDN

2 bytes little endian, register address 0x0A, read/write, saved to flash

Default value: 604 (corresponding to 2.95 V, 4.88 mV per LSB)

This value determines the battery voltage at which the Raspberry Pi will be instructed to shut down. The Raspberry Pi software package contains scaling code so the value can be read and set in mV for convenience.

VBAT_BOOT

2 bytes little endian, register address 0x0C, read/write, saved to flash

Default value: 645 (corresponding to 3.15 V, 4.88 mV per LSB)

This value determines the battery voltage level at which the Raspberry Pi is allowed to boot. Note that this value is higher than VBAT_SHDN to provide hysteresis. This will ensure that the system will not oscillate between boot and shutdown when the battery is nearly empty, but then the voltage recovers when the load is turned off. Under heavy load, it may be necessary to increase this value to prevent continuous boot / shutdown cycling.

The Raspberry Pi software package contains scaling code so the value can be read and set in mV for convenience.

VIN_THRESHOLD

2 bytes little endian, register address 0x10, read/write, saved to flash

Default value: 503 (corresponding to 4.75 V, 9.44 mV per LSB)

This value determines the input (USB) voltage level at which the Raspberry Pi will be booted, if the battery voltage is also high enough (VBAT_BOOT threshold), and the AUTO_BOOT register is set to option 3 or 4. It also determines the input voltage level at which the input is considered to be missing and the Raspberry Pi will be shut down after AUTO_SHDN_TIME minutes.

The Raspberry Pi software package contains scaling code so the value can be read and set in mV for convenience.

VOUT_MAX

2 bytes little endian, register address 0x0E, read/write, saved to flash

Default value: 717 (corresponding to 3.88 V, 5.42 mV per LSB)

This value determines the minimum output voltage present for which the LiFePO4wered/Pi3™ will

refuse to boot the Raspberry Pi when it is supposed to be off (according to the LiFePO4wered/Pi3™). The LiFePO4wered/Pi3™ power supply employs a bidirectional load switch that makes it possible to power the Raspberry Pi from a different source (such as its own micro USB power connector) with the LiFePO4wered/Pi3™ attached without causing any damage (NOTE: this only works if the battery is present, damage *will* occur if the Raspberry Pi is powered from another power source and the battery has been removed from the LiFePO4wered/Pi3™). Because the LiFePO4wered/Pi3™ should not be allowed to turn on when the Raspberry Pi is powered from a different source, this voltage check provides a safety feature that prevents this from happening.

The Raspberry Pi software package contains scaling code so the value can be read and set in mV for convenience.

VOFFSET_ADC

2 bytes little endian, register address 0x12, read/write, saved to flash

Default value: 0 (corresponding to 0 V, 4.88 mV per LSB)

This register provides a calibration value for the ADC voltage measurements. It is a simple 1 point offset calibration that provides compensation for inaccuracy of the internal reference voltage.

The Raspberry Pi software package contains scaling code so the value can be read and set in mV for convenience.

AUTO_BOOT

1 byte, register address 0x18, read/write, saved to flash

Default value: 0x00

When this register is 0 (AUTO_BOOT_OFF), the LiFePO4wered/Pi3™ will stay off until the user touches the on/off touch pad to turn the Raspberry Pi on.

Setting this register to 1 (AUTO_BOOT_VBAT) will make the Raspberry Pi boot immediately when sufficient battery voltage is available (VBAT >= VBAT_BOOT threshold). This is useful when using the LiFePO4wered/Pi3™ as a UPS to maximize uptime.

Setting this register to 2 (AUTO_BOOT_VBAT_SMART) will make the Raspberry Pi boot immediately when sufficient battery voltage is available, but only if the unit was previously shut down due to a low voltage condition. This makes it so the user can still choose to turn the Raspberry Pi off with the touch button or from a user program.

Setting this register to 3 (AUTO_BOOT_VIN) will make the Raspberry Pi boot if sufficient battery voltage is available (VBAT >= VBAT_BOOT threshold) and the USB voltage is also present (VIN >=

VIN_THRESHOLD). Setting the register to 4 (AUTO_BOOT_VIN_SMART) enables the smart version of this setting, which allows the user to shut down the unit manually as described above.

WAKE_TIME

2 bytes little endian, register address 0x1A, read/write, not saved to flash

Default value: 0

This register allows the user to set a time in minutes that determines how long the Raspberry Pi will stay off before the LiFePO4wered/Pi3™ will automatically boot it again. It is implemented using RC oscillators in the microcontroller and as such has limited accuracy (expect +/- 10%). If the value is 0, the wake timer is off.

This value cannot be saved in flash, but needs to be set by a user program every time before the Raspberry Pi shuts down. It allows extended run time on battery power for tasks that have low duty cycles. The LiFePO4wered/Pi3™ will still respond to touch button presses and AUTO_BOOT as usual when the wake timer is set.

A user program can check this value after boot, the value will reflect the number of minutes remaining in the wake timer when the system was booted. If there is still time remaining, this indicates the system boot was not triggered by the wake timer, but from another source.

SHDN_DELAY

2 bytes little endian, register address 0x14, read/write, saved to flash

Default value: 65

This sets the number of LiFePO4wered/Pi3™ system ticks that elapse between when the Raspberry Pi is shut down (detected by the UART TX line going low) and when the power to it is turned off. The system ticks are not at all accurate (they are implemented with a low power oscillator), and can vary from 2.6 to 13 ticks per second. The default value is chosen to allow plenty of time between shutdown and power off, even with the fastest system tick. To attain maximum run time on battery power in low duty cycle systems using the wake timer, the user can reduce this value.

Another possible use for changing this value is when the Raspberry Pi 3 is configured to disable the UART on the GPIO header. This is actually the default state on the Raspberry Pi 3, however the LiFePO4wered/Pi3™ software installer will change the system configuration to turn the UART back on. If this conflicts with what the user wants to do, it is possible to keep the UART disabled and set this register to a large value. This can make the system work correctly for shutdown and reboot even when UART TX line detection is not available. The delay in that case has to be long enough to last

through a reboot from the time the LiFePO4wered/Pi3™ daemon is unloaded until it's loaded again.

AUTO_SHDN_TIME

2 bytes little endian, register address 0x16, read/write, saved to flash

Default value: 0xFFFF

Auto shutdown is a feature that will shut down the Raspberry Pi if the input (USB) voltage falls below the VIN_THRESHOLD level. The shutdown will happen after a delay in minutes specified in this register. By default, the time of very long, longer than the run time of the system on battery power, effectively disabling auto shutdown. When this value is set to 0, immediate shutdown is triggered when the input power is removed. Since it may be better to keep the system running through minor power interruptions, it is generally better to allow at least a minute before shutting down.

PI_RUNNING

1 byte, register address 0x1C, read/write, not saved to flash

Default value: 1 once the Raspberry Pi is booted

This is an important register that determines the state of the Raspberry Pi power. It is normally managed by the LiFePO4wered/Pi3™ itself and the LiFePO4wered/Pi3™ daemon. When the power to the Raspberry Pi is off or when the Raspberry Pi is booting, the value of this flag is 0. When the LiFePO4wered/Pi3™ daemon starts, it sets this flag to 1 to indicate the Raspberry Pi has booted. This will change the state of the LiFePO4wered/Pi3™, the PWR LED will go from pulsing to on state, and will be ready for user input (touching the button to turn the Raspberry Pi off again). This flag can be cleared by various sources, such as a user pressing the touch button, the battery voltage falling below VBAT_SHDN, or the daemon being shut down when a user shuts down the Raspberry Pi. On the other hand, the daemon will also *trigger* a shutdown if this flag goes low from another source. In either case, the system will be shutting down, the LiFePO4wered/Pi3™ will show this by pulsating the PWR LED and the power will be turned off.

As mentioned, the user does not need to worry about manually controlling this flag, the LiFePO4wered/Pi3™ daemon takes care of it. If the user sets this flag to 0, this will trigger a system shutdown.

CFG_WRITE

1 byte, register address 0x19, read/write

Default value: 0

This register makes it possible to make configuration changes permanent by writing the values to flash memory (only those marked by "saved to flash"). It may not be necessary to use this, since the LiFePO4wered/Pi3™ microcontroller stays powered even when the Raspberry Pi is off. However, when the battery is removed, configuration changes will be lost. This can be a good thing, it allows the user to experiment with changing configuration values, and if they cause a problem, they can be undone by removing the battery and putting it back, with power disconnected. Only if the user is very sure about their configuration changes should they be written to flash. Writing bad configurations to flash can **MAKE THE DEVICE UNUSABLE**.

To write the current configuration to flash, the user has to write the "magic value" 0x46 (70) to the CFG_WRITE register. Any other value is ignored, and the register is always read as 0.

Command line tool specification

To make it convenient to interact with the LiFePO4wered/Pi3™, the software package installed on the Raspberry Pi provides a command line tool. Help is provided when you run it without parameters:

```
lifepo4wered-cli
```

The tool can be used to get and set the values of the LiFePO4wered/Pi3™ I²C registers described in the previous section without having to know implementation details such as register addresses and voltage scaling. For instance, to get the current battery voltage, run:

```
lifepo4wered-cli get vbat
```

This will return the battery voltage converted to millivolts. To set the wake time to an hour, run:

```
lifepo4wered-cli set wake_time 60
```

When you shut down the Raspberry Pi, it will wake up again in about 60 minutes. Or if you want the Raspberry Pi to always run whenever the USB input voltage is present, run:

```
lifepo4wered-cli set auto_boot 3
```

Please refer to the I²C register specification for a complete reference of available options.

Electrical characteristics

Unless otherwise indicated, all characteristics apply for V_{USB} = 4.75 V to 5.25 V and T_A = 0 ºC to 50 ºC. Typical values are at 25 ºC and V_{USB} = 5 V.

Parameter	Sym	Min	Typ	Max	Unit	Conditions
USB charge voltage	V_{USB}	4.75	5.0	9.0	V	

Battery leakage current	$I_{DISCHARGE}$		5		µA	USB voltage absent, Raspberry Pi powered off
Battery charge current	I_{CHARGE}		1.33		A	
Load current	I_{LOAD}			2	A	
Output voltage	V_{OUT}	4.8	5.0	5.2	V	
Wake timer accuracy	t_{WAKE}	-10		+10	%	
Default minimum battery voltage (power forced off)	$VBAT_{MIN}$		2.85		V	
Default shutdown battery voltage (Pi shutdown triggered)	$VBAT_{SHDN}$		2.95		V	
Default minimum boot battery voltage	$VBAT_{BOOT}$		3.15		V	
Default output voltage preventing boot	$VOUT_{MAX}$		3.88		V	Raspberry Pi is powered from another source
Default input (USB) threshold voltage	VIN_{THRS}		4.75		V	
Maximum Power Point Tracking threshold voltage	VIN_{MPPT}		4.66		V	Charge current will be limited to prevent VIN from dropping below this threshold

Sales and support

To buy the LiFePO4wered/Pi3™, please visit http://lifepo4wered.com. To order in quantity and for volume discounts, please contact sales@lifepo4wered.com.

For technical support, please contact support@lifepo4wered.com.