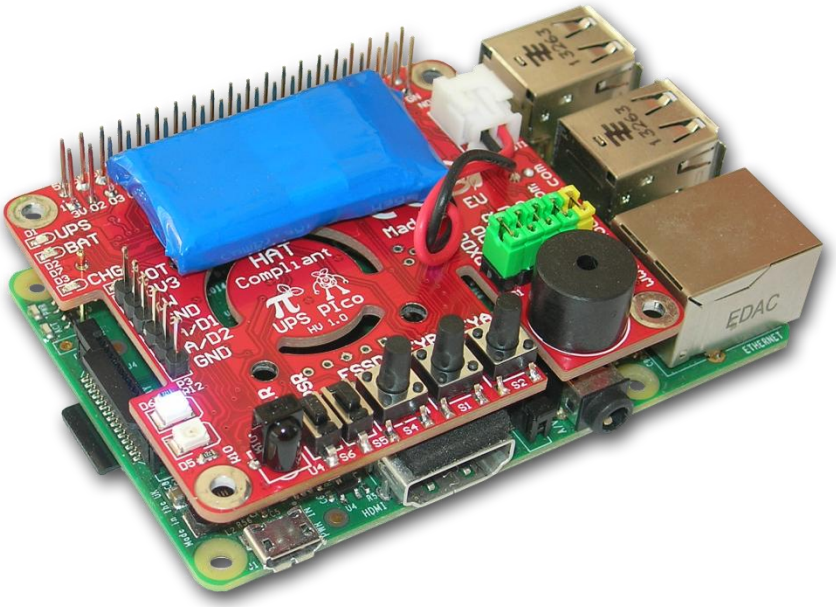
UPS PIco

Uninterruptible Power Supply

with Peripherals and I²C control Interface

for use with

Raspberry Pi® B+, A+, B, and A



HAT Compliant

Short User Guide

Preliminary Version 0.98

Intelligent Modules for your Raspberry Pi®

Document Revisions

Version	Date	Modified Pages	Modified Sections	Comments
1.0	28/01/2015	none	none	First Public Document Release

Table of Figures

Credits

System Overview

Introduction

The **UPS PIco** is an advanced uninterruptible power supply for the Raspberry Pi® that adds a wealth of innovative power back-up functionality and development features to the innovative microcomputer!

The standard **UPS PIco** is equipped with a 300mAh LiPO battery specially designed to enable safe shutdown during a power cut. Additionally, this can be easily upgraded to the extended 3000mAh version, which enables prolonged use of a Raspberry Pi for **up to 8 hours** without a power supply connected!

The **UPS PIco** features an embedded measurement system that continuously checks the powering voltage of the Raspberry Pi®. When the cable power on the Raspberry Pi® is absent, insufficient, or the device detects a power failure, the **UPS PIco** automatically switches to the unit's battery source. The module then continues to check the voltage on the Pi and switches automatically back to the regular cable supply when power becomes once again available.

The **UPS PIco** is powered and the battery pack intelligently charged via the GPIO pins on the Raspberry Pi®, so no additional cabling or power supply is required.

The **UPS PIco** is designed to be 100% compliant with HAT standards for the Raspberry Pi® B+ and A+, and is mechanically compatible with the original Raspberry Pi® models A and B when an extension header is used. In addition to this, because the **UPS PIco** requires no external powering and fits within the footprint of the Raspberry Pi®, it is compatible with most cases.

The **UPS PIco** can also be equipped with an optional **Infra-Red Receiver** which is routed directly to GPIO18 via the PCB. This opens the door for remote operation of the Raspberry Pi®and **UPS PIco**!

Finally, the **UPS PIco** features an implemented Automatic Temperature Control **PWM fan controller**, and can be equipped with a micro fan kit, which enables the use of the Raspberry Pi® in extreme conditions including very high temperature environments.

Applications

UPS PIco is equipped with plenty of features which make it an extremely useful tool for Raspberry Pi® project development. It not only provides powering continuity, but also offers extra user programmable LEDs, sensors, buttons and I/O's. The unit also features a dedicated **10-bit analogue to digital converter** with two channels making it the perfect board for remote and unmanned sensor deployment. These extra features result in the **UPS PIco** being a superior all-in-one device, perfect for many innovative projects and embedded applications.

Features

The list of features of the **UPS PIco** is as follows:

- Raspberry Pi B+ **HAT Compliant**
- **Plug and Play**
- **Smart Uninterruptible Power Supply (UPS)**
- **Integrated LiPO Battery** (8-10 Minutes of Power Back-Up)
- **Intelligent Automatic Charger**
- **No Additional External Power Required**
- **Optional 3000 mAh** Battery for 8 Hours Run-Time (Not Included)
- **5V 2A Power Backup (Peak Output 5V 3A)**
- Integrated Software Simulated **Real Time Clock (RTC)** with Battery Back-Up
- **File Safe Shutdown** Functionality
- Raspberry Pi B+ **Activity Pin**
- **PWM fan control** (Fan Not Included)
- **2 User Defined LEDs**
- **2 User Defined Buttons**
- **Integrated Buzzer** for UPS and User Applications
- **Status Monitoring** - Powering Voltage, UPS Battery Voltage and Temperature
- **I^2C PICo Interface** for Control and Monitoring
- **RS232 Raspberry Pi** Interface for Control and Monitoring
- **XTEA Based** Cryptography User Software Protection
- 2 Level **Watch-dog Functionality** with **FSSD and Hardware Reset**
- **Raspberry Pi B+ Hardware Reset Button via Spring Test Pin** (Not Included)
- **Jumpers for Raspberry Pi B+ Pin** Functionality Selection
- **Stackable Header** for Add-On Boards
- **Boot Loader** for Live Firmware Update
- Compatible with **Intelligent IR Remote Power ON/OFF (PowerMyPi)**
- **Integrated ESD-Protected 2 Channel A/D 10 Bit Converters 0-5.2V**
- **Integrated ESD-Protected 1-Wire Interface**
- **Labeled J8 Raspberry Pi B+ GPIO Pins** for Easy Plug & Play
- **Infra Red Receiver** Sensor Interface (IR Not Included)
- **Upgradable** with **PIco Add-on Boards**
- **Fits Inside Most Existing Cases**

General Information

The **UPS PIco** Module uses the **5 VDC** and **GND** pins for powering, and interacts with the Raspberry Pi® through GPIO_GEN22 and GPIO_GEN27. Simple Python script must be running on the Raspberry Pi® that interacts with the **UPS PIco** Module. The GPIO_GEN22 is used to force the File Safe Shutdown (FFSD) procedure when the **UPS PIco** Module make it low, and read by the Raspberry Pi®. The GPIO_GEN27 is used by the same Python script and generate a pulse train that is recognized by the UPS PIco Module and interpreted if the Raspberry Pi® is running or not. This approach allows simplifying the design, and cutting the needs to have current measure system on the **UPS PIco** Module. If the Raspberry Pi® is running (so the pulse train is generated) user can see that the UPS LED on the **UPS PIco** Module is flashing (fast if cable powered, and slow if battery powered). If the UPS LED is not flashing, it means that the Raspberry Pi® is not running (the Python script is not running). Therefore it is mandatory for a proper use of the **UPS PIco** Module to have the Python script installed. Due to this implemented feature, the UPS LED flashing functionality, user can by a single view have a knowledge if the Raspberry Pi® is running or not (Hang-up, or Shutdown).

It is important to notice that for the UPS PIco Module operation it is mandatory to have connected GPIO_GEN27 to the UPS PIco Module trough the FSSDU Jumper, as also installed the picofssd.py script. The GPIO_GEN27 is generating pulse train, that is interpreted by the UPS PIco Module and allows it to recognize different states of Raspberry Pi® powering and running.

UPS PIco Jumpers

For the Basic operation, **UPS PIco** Module uses only the **5 VDC** and **GND** pins and interacts with the Raspberry Pi® through GPIO_GEN22 and GPIO_GEN27. Some specialized features require in addition RS232 and I^2C. In addition if the 1-wire device and IR Receiver are used, there are directly routed to GPIO_GEN04 and GPIO_GEN18 pins respectively. However these pins if not used with above features (the 1-wire not connected and IR Receiver not soldered) can be freely used as a standard GPIO pins. The RS232, GPIO_GEN22, GPIO_GEN27 and HAT EEPROM are going to the **UPS PIco** Module trough Jumpers set. Therefore, if user not needs some of them can remove the jumpers and free that pins for other applications.

Jumper	Descrition	Usage	Connection
RXD0	RS232 Receive	RX for RS232 @command for the UPS PIco	GPIO_GEN15

TXD0	RS232 Transmit	TX for RS232 @command for the UPS PIco	GPIO_GEN14
FSSDR	FSSD Raspberry Pi	File Safe Shutdown Pin for the Raspberry Pi	GPIO_GEN22
FSSDU	FSSD UPS PIco	RUN Pin for the UPS PIco	GPIO_GEN27
HATWP	HAT EEPROM WP	HAT EEPROM Write Protection	EEPROM WP

Figure 1 UPS PIco Jumpers Usage Table

UPS PIco Buttons

The **UPS PIco** Module is equipped with 5 buttons that can be used in various ways. Two of them are dedicated for user applications and can be handled by user through the **PICo** (I^2C) interface or **@commands** (RS232), all other are specific for various **UPS PIco** Module functionalities. All of them can be used for some start-up functionalities when **UPS PIco** Module is reset. A detailed description of all buttons and their usage is provided on below table.

Button	Description	Usage	Additional Functionalities
RPiR	**R**aspberry **Pi** Hardware **R**eset	Make Raspberry Pi Hardware Reset when pressed. To be used need installed (soldered) the Gold Plated Reset Pin. **NOTE1**: Resetting of the Raspberry Pi®, can corrupt files on the SD card if used **NOTE2**: Resetting of the Raspberry Pi®, does **not** affect the **UPS PIco** (including PIco RTC)	NONE
UPSR	**UPS** PIco Hardware **R**eset	Make UPS PIco Hardware Reset when pressed. **NOTE1**: Resetting of the **UPS PIco** does not reset the Raspberry Pi®. **NOTE2**: Resetting of the **UPS PIco** does **reset** the simulated RTC to default values.	When pressed with combination with other buttons activate various start-up functionalities. The procedure is to press first the **UPSR** button, and then another one, then release the **UPSR** button and then release the other button (valid for **FSSD, KEYA, KEYB)**.
FSSD	**File Safe Shutdown**	When pressed initiate the File Safe Shutdown Procedure. If used need to have **FSSDR Jumper** short. If Raspberry Pi®+UPS PIco system battery powered, after FSSD finished UPS PIco will cut the power. Pressed again (need to have installed the Gold Plated Reset Pin for the restart option), start the Raspberry Pi®+UPS PIco system again.	When used with **UPSR** button, make factory self test, used during boards testing. Not useful for user, as a special test board with spring test pins need to be connected.
KEYA	**User Key A**	Can be used for User Application – Read the status via PICo (I^2C) or RS232 interface	When used with **UPSR** button, makes the factory default, and reset the RTC to startup values.
KEYB	**User Key B**	Can be used for User Application – Read the status via PICo (I^2C) or RS232 interface	When used with **UPSR** button, invokes the bootloader (light the Red User LED). The bootloader can be invoked also from the PICo interface.

Figure 2 UPS PIco Buttons Usage Table

UPS PIco LEDs

The **UPS PIco** Module is equipped with 6 LEDs that offers information about the **UPS PIco** Module system status. Two of them are dedicated for user applications and can be handled by the **PICo** (I^2C) interface or **@commands** (RS232). One of them is **Red** and the second one is **Blue**. A detailed description of all LEDs and their usage is provided on below table.

LED	Description	Usage
UPS LED - Green	Provide information about **UPS PIco** Module status	• Flashes Normally when system is cable powered and Raspberry Pi® is running (100 ms ON, 500ms OFF) • Flashes Slow when system is battery powered and Raspberry Pi® is running (100ms ON, 2000ms OFF) • Flashes Fast when FSSD is executed (100ms ON, 200ms OFF) • Not light when **UPS PIco** Module is in Low Powering Mode (LPR) and the Raspberry Pi® is not running.
BAT LED - Orange	Provide information about **UPS PIco** Module Battery Level when system is powered from it	• Battery Level > 4.0 V BAT LED not Flash • 4.0V <= Battery Level > 3.7 V BAT LED Flashes 100ms ON, 2000ms OFF • 3.7V <= Battery Level > 3.5 V BAT LED Flashes 100ms ON, 1000ms OFF • 3.4V <= Battery Level > 3.5 V BAT LED Flashes 100ms ON, 500ms OFF • 3.3V <= Battery Level > 3.4 V BAT LED Flashes 100ms ON, 200ms OFF FSSD is immediately initiated
CHG LED - Green	Provide information about **UPS PIco** Module Battery Charger Status	It is valid only when cable powering is present. When battery is charged the CHG LED Lights.
HOT LED - Orange	Provide information about **UPS PIco** Module Temperature. Read both sensors (the embedded SOT23 and the PIco FAN Kit TO092) and shows the higher one.	When system temperature is higher than threshold the HOT LED lights. The temperature is measured on both sensors the SMD placed on the top as also on the TO-92 is PIco FAN Kit is used. The higher temperature on one of them activates the HOT LED, and remains until both sensors have lower temperature than the threshold. The default value is 42 Degs Celsius.
LED RED - PLCC2 size	Provided for **UPS PIco** Module User Application	Available for user application. Handled by the PICo and @commands interface. During Boot Functionalities support interaction with various lightings
LED BLUE - PLCC2 size	Provided for **UPS PIco** Module User Application	Available for user application. Handled by the PICo and @commands interface. During Boot Functionalities support interaction with various lightings

Figure 3 UPS PIco LEDs Usage Table

UPS PIco I/Os

The **UPS PIco** Module is equipped with 4 I/Os, there are:

- 1-wire interface
- 10 bits A/D converter 1 (0 - 5.2V)
- 10 bits A/D converter 2 (0 - 5.2V)
- IR Receiver

The 1-wire interface is supported with supply of 3.3V (which is independent from the Raspberry Pi® powering) and GND on the same connector. It simplifies the connection making when the 1-wire is used. In addition the 1-wire interface is ESD protected. Is is directly routed to the GPIO_GEN04. If the 1-wire sensor is not used the GPIO can be used for any other application on the J8 pins. It contains just the required 4K7 resistor connected to the 3.3V.

The UPS PIco Module supports also pre-calibrated 2 x 10 bits A/D converters. Their readings can be easy accessed easy via **PICo** (I^2C) interface or **@commands** (RS232). Read values are in mV. Those inputs are also ESD protected.

The IR Receiver if assembled (soldered on the PCB) is directly routed to the GPIO_GEN18. A very good tutorial how to use it can be found on below link:

https://www.modmypi.com/blog/raspberry-pis-remotes-ir-receivers

User does not need to add any other component in order to use the IR Receiver. If the IR Receiver is not used (not soldered to the PCB) the GPIO_GEN18 can be used for any other application.

The PICo (I^2C) Interface - Peripherals I^2C Control Interface

The **P**eripherals **I**^{2}C **Co**ntrol – The **PICo Interface** – is an implementation of **I^{2}C** interface adapted to easy control of the peripheral connected to the Raspberry Pi® via command line. By using human understandable simple commands, control of peripherals is made extremely simple. Control at programming language level is also possible and easy. The core concept of the **PICo interface** is that all peripheral device control and data exchange between it and Raspberry Pi® variables are common for the **I^{2}C interface** as also for the peripheral itself. Therefore any change of them by either party, Raspberry Pi® and the peripheral, causes immediate update and action.

Two types of variables are available:

- **Common**, where data are stored in the same place and any change on it will cause action on the **UPS PIco** Module

- **Mirror**, where are copy of data stored on internal variables of the **UPS PIco** Module, they are protected, so changes on it will not implies the **UPS PIco** Module functionality and will be overwritten immediately when **UPS PIco** Module recognized changes on them

There have been implemented 3 **PICo** addresses assigned to the following entities:

0x69 -> UPS PIco RTC Registers Direct Access Specification

Address	Name	Size	Type	R/W	Explanation
0 or **0x00**	Seconds	Byte	Mirror	Read	Seconds in BCD
1 or **0x01**	Minutes	Byte	Mirror	Read	Minutes in BCD
2 or **0x02**	Hours	Byte	Mirror	Read	Hours in BCD
3 or **0x03**	DOW	Byte	Mirror	Read	DOW in BCD
4 or **0x04**	Days	Byte	Mirror	Read	Days in BCD
5 or **0x05**	Month	Byte	Mirror	Read	Month in BCD
6 or **0x06**	Year	Byte	Mirror	Read	Year in BCD
7 or **0x07**	RTCCF	Byte	Common	R/W	Real Time Clock Correction Factor in HEX Change the RTC timer for multiples of 1 tick per second Timers tick is 1/32768 Hz= 0,000030517578125 sec **Write:** 0x00 or 0x80 not change the RTC tick **Write:** 0x01 – 0x79 change the RTC tick by subtract tick multiplication from the standard timer values, therefore it will decrease the "duration" of each second by multiple value of timer ticks – counted second will be shorter, so RTC will be running faster **Write:** 0x81 – 0xFF change the RTC tick by adding tick multiplication to the standard timer values, therefore it will increase the "duration" of each second by multiple value of timer ticks – counted second will be longer, so RTC will be running slower Adding or subtract of one tick change the 24 hours RTC by 86400 * 0,000030517578125 sec = 2,63671875 seconds **Read:** Checks actual RTCCF Value

Figure 4 0x69 -> UPS PIco RTC Registers Direct Access Table

Access to these variables can be done indepened if the RTC is used by the Raspberry Pi(R) system. The last variable is **Common** changes of it trim the RTC accuracy according to specifications provided in above table. Therefore writting to this register cause immediate action and trimming of the RTC.

Accessing 0x69 Varibales

In order to access the **0x69** variavbles the following commands need to be executed form the OS command line

sudo i2cget -y 1 0x69 0

The result wil be the seconds counter of the implemented RTC. There are a very good python script and “C” software that showing how to access them, on the products forum.

sudo i2cget -y 1 0x69 1

The result wil be the minutes counter of the implemented RTC.

sudo i2cset -y 1 0x69 7 0x01

The result will set the RTCCF trimming RTC factor to 1.

UPS PIco @commands RS232 interface

There are plenty of commands for full system control. Thanks to the implemented bootloader, the set of commands can be constantly enhanced with new ones, as we release more of them. We are open to customer suggestions about new commands to implement. Customers can propose new commands by e-mail or on our forum: if we find them generally useful, then we will implement them for free and distribute them via our bootloader system. Customized versions of the firmware featuring customer-specific commands are also possible.