UPiS

Uninterruptible Power intelligent Supply

for use with

RaspberryPi®

User Manual

Version 1.10

Preliminary Version

"Raspberry Pi" is a trademark of the RaspberryPi Foundation

Firmware Release 1.00

Hardware Release VCO2

Table of Contents

Table of Figures

Credits

Our Company would like to thank the following people that reviewed and, many times, commented and corrected this document before we released it to the public domain.

Marcello Antonucci, Rome, Italy

Georgios Karachos, Siegen, Germany

Mathew E. Rivers of the Univeristy of Alabama at Bimingham - USA

James M. Olson of the Univeristy of Alabama at Bimingham - USA

Document Revisions

TBC

Firmware Updates Road Map

The **UPiS** is a hardware platform that covers many features. Most of them are supported by the Firmware Version Release 1.00. However, some of them are planned features still in the testing or under development phase and will be released soon in the next versions of firmware. The following table describes the Implementation and Version Release Road Map.

Firmware Version	Features Added
1.00	UPiS Official Firmware Version Release
1.10	1. USB – RS232 interface bridge option (powering matters)
1.20	1. Original micro USB RaspberryPi® powering (the RPI mode) full implementation 2. Onboard ESD Protected I/O pin, controlled via RS232 3. Low Powering Mode when cabling supply is available
1.30	1. iButton interface algorithm implementation for the Scripting Language
1.40	1. Extended Tiny Encryption Algorithm (*XTEA*) cryptographic Customer Software Protection System (with custom defined protection keys) 2. RTC based Alarms for RaspberryPi® wake-up and sleep
1.50	3. Event Driven Advanced Scripting Language

Figure 1 Firmware Updates Road Map

System Overview

Introduction

The **UPiS** is an Advanced Powering add-on Module for the RaspberryPi® that adds a wealth of additional features to the powering functionality. It is equipped with a LiPO battery (1150 or 2600 mAh) and features a buck/boost switching power converter. There is no need for any additional cabling or power supply, as the **UPiS** is powered by the same power supply of your original RaspberryPi®; you just insert the **UPiS** on the top of the **P1** connector of your RaspberryPi®. The **UPiS** has an embedded measurement system that continuously checks the powering voltage and current consumption, and when the cable power is absent or insufficient, it automatically switches to the battery source. Then, it keeps checking the input voltage on all power sources, and when cable power is available again, it switches to it automatically, turning the battery source off. The **UPiS** uses exactly the same micro USB Power Supply that you are using to supply your RaspberryPi®, however it also has an extended external voltage input[1] for other non-standard powering sources.

Applications

The **UPiS** as an add-on Module is addressed to all users that need a power back-up and/or sensing features for applications running on the RaspberryPi®. All applications running on the RaspberryPi® can take advantage of the *uninterruptible power supply* feature of the **UPiS** (ranging from RaspberryPi®-based fan-less servers to solar-powered applications), but in addition, the **UPiS** provides a wealth of sensors and features, all cumulated in a single all-in-one unit, that can enable writing many innovative applications.

Features

The features of the **UPiS Module** can be categorized as follows:

- Powering functionalities
- I/O and control functionalities
- RTC functionalities
- Interfaces functionalities
- Software Protection functionalities
- Environment supervising functionalities

In detail, the list of **UPiS** features is below:

1. Supervised and Protected Powering from all cable sources
   a. RaspberryPi ® micro USB (5 VDC) – available from firmware release V1.20
   b. Additional micro USB (5V DC)
   c. Extended External Powering Input (7V DC – 18V DC) [Advanced version only]
2. Battery Power Backup on each cable powering source (including original RaspberryPi® micro USB – optional after firmware activation) – the UPS feature
3. Onboard Rechargeable LiPO Battery (1150/2600 mAh) – battery working time is from 2 to 5 hours, depending on the version, system load and configuration

[1] The extended external voltage input is available only in the advanced version of the **UPiS**

4. Onboard enhanced multiple level protection system for the LiPO battery:
    a. Cut-off jumper
    b. PTC fuse
    c. Onboard Thermometer
    d. Over-charge and Over-discharge protection
    e. Over-voltage and Under-voltage protection
5. Onboard Intelligent Automatic LiPO Battery Charger (Charges the battery automatically only if the supply voltage is present and can provide enough current to both feed the RaspberryPi® and charge the battery)
6. RaspberryPi® Hardware ON/OFF Switch
7. Embedded Emulated RTC (Real Time Clock – DS1307) accessible via RaspberryPi® I2C and/or RS232 provided from the System
8. Onboard Analog Thermometer (accessible via RaspberryPi® RS232)
9. Onboard True USB interface (can be used as RS232 – USB Bridge)
10. Programmable Time, RaspberryPi® File Safe Shutdown Button[2]
11. Full monitoring of all UPiS Powering Parameters via RaspberryPi® RS232 port:
    a. Current Consumption
    b. Voltage on each Power source
    c. System Temperature
    d. Battery Level
    e. Powering source
12. RTC based programmed Startup/Shutdown
13. Onboard UPiS Reset Button (resets UPiS and RaspberryPi® but not RTC by cutting the powering of the RaspberryPi® for a very short time)
14. Onboard NO RELAY controlled via RS232 or RaspberryPi® Pin (selectable by jumper GPIO_GEN0)
15. Onboard ESD Protected 1-wire interface, controlled via RS232 or RaspberryPi® Pin (selectable by jumper GPIO_GEN3) with separate 3.3V supply pull-up resistor.
16. Onboard ESD Protected I/O pin, controlled via RS232 or RaspberryPi® Pin (selectable by jumper GPIO_GEN3)
17. Onboard True 12 V RS232 interface to the external world (with level converter)
18. Protected (Resettable fuse 140 mA) 5 VDC output for user applications, with battery backup feature
19. Non-protected 3.3 VDC output for user applications (usually used for 1-wire application), separate and independent from the RaspberryPi® 3.3 supply.
20. Extended Tiny Encryption Algorithm (*XTEA*) cryptographic Customer Software Protection System (with custom defined protection keys)
21. Scripting language
22. LED-based Status Information System
23. Bootloader feature for lifetime firmware update.

[2] Requires that the RaspberryPi® be powered from the second micro USB placed on the UPiS board or from Extended External Powering Input

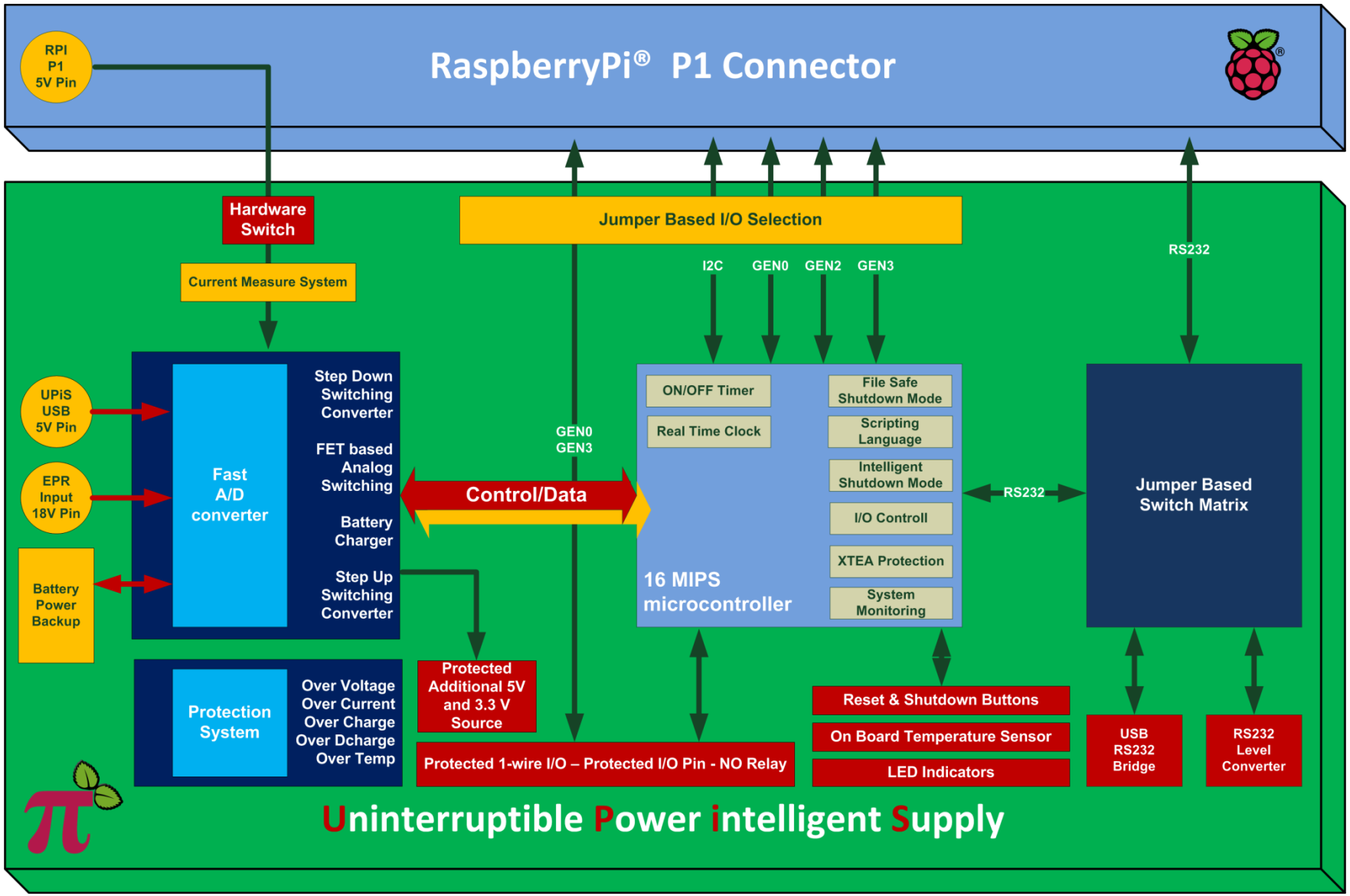

Figure 2 UPiS Block Diagram

System Basics

UPiS Versions

The **UPiS** module is 100% Plug and Play: there is no need to setup anything. There is no need to change the original RaspberryPi® micro USB Power Supply since the **UPiS** module uses exactly the same cabling. You only need to plug-in the **P1** on the RaspberryPi® and switch ON the **UPiS** module in order to supply your system. To take advantage of all the features of the **UPiS**, you will typically connect the original RaspberryPi® micro USB Power Supply to the **UPiS** (thus interposing the UPiS between the power supply and the RaspberryPi®), but if you need to keep the old cabling as it was, then you can elect to keep the power supply directly connected to the RaspberryPi®.

There are two versions of **UPiS** available: **UPiS Basic** and **UPiS Advanced**. Each one could be ordered in two variants, i.e. with top end or stack **P1** connector. The stack version allows plugging on additional user boards, while the top end should be the end of the RaspberryPi® System.

Cable Powering Sources – the Powering Feature

The **UPiS** module is designed to offer automatic power battery back-up to the RaspberryPi® supplied from the following Cable Powering Sources:

- RaspberryPi ® micro USB (5V DC) – available from firmware release V1.20
- Additional micro USB (5V DC)
- Extended External Powering Input (7V DC – 18V DC) [Advanced version only]

It is recommended to use only one Cable Powering Source at a time; however, if more than one of the cable powering sources is available (plug-in), then the **UPiS** runs an automatic process which selects and activates only one. The powering source selection is done according to an implemented priority algorithm. The highest priority has the RaspberryPi® original micro USB supply, so if you fit the cable power supply to the RaspberryPi®, all other sources will be automatic deselected (internally disconnected) even if you have plug in cables to them. The next priority is the extended external powering input and, just as before, if you have connected this powering source, the **UPiS** micro USB will be not active as a power source, it will be only active as a data connection. Finally, the smallest priority has the **UPiS** micro USB powering. The battery back-up is activated and supplying the RaspberryPi® only if all of the cable powering sources are missing. In practice, because we are usually using only one cable power supply (recommended), we are just using the selected cable power source. When power is down, then immediately battery back-up will start up.

Summary of power source priorities:

RaspberryPi® micro USB ⇨ Extended External Powering ⇨ UPiS micro USB ⇨ Battery Back-up

Battery Power Backup – the UPS feature

The **UPiS** is an off-line **UPS** with extremely fast switching time and intermediate power preservation (in fact, it is line-interactive **UPS**). The decision to activate battery back-up is made according to the powering status of the 5V Pin on the **P1** connector as well as current consumption.

The system continuously monitors the RaspberryPi® **P1** 5V Pin, detecting the falling edge of power or lower voltage than the defined threshold.Within 360 µs, it automatically activates the battery back-up powering, thus preventing the Raspberry Pi® from suffering an unexpected power shutdown.

The activation time of battery back-up power is extremely fast and is executed within 120 µs; switching of power backup takes less than 14 µs. In addition, a large tantalum capacitor is supplying the RaspberryPi® system during this off-line time in order to avoid power glitches.

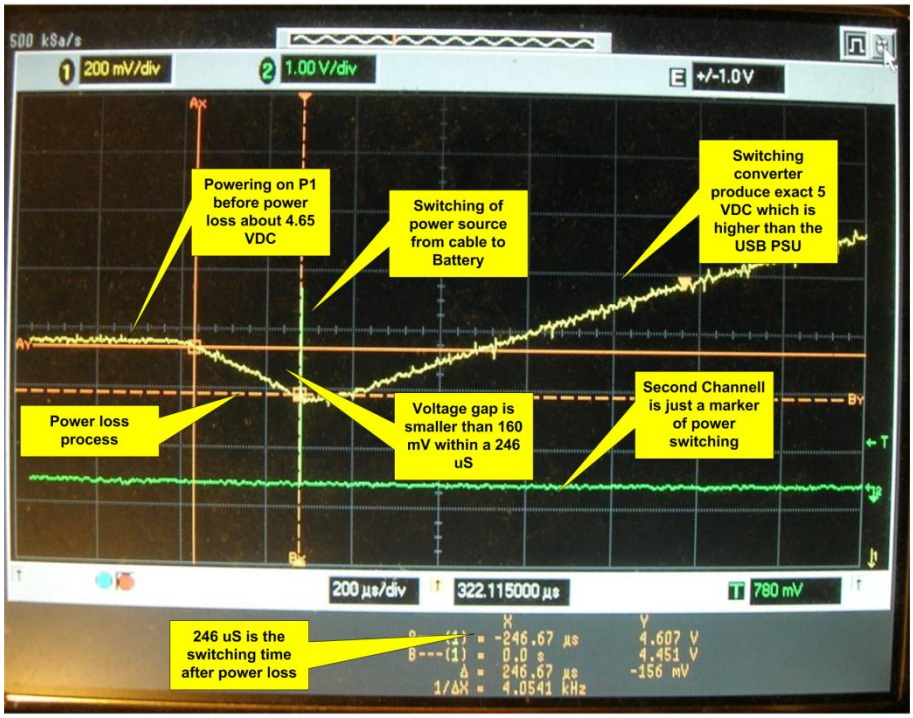

Figure 3 Battery powering switching process after a Cable Power loss

UPiS Add-On – the Intelligent Features

The **UPiS** module, first of all, is an **Advanced Powering System** with battery back-up for the RaspberryPi®. It practically covers most of the possible requirements that users of the RaspberryPi® can have with their system powering. But there's much more than this! Following the main goal of our company, "Intelligent Modules for your RaspberryPi®", we tried to design a device as compact as it possibly could be, yet offering as many features as possible for the standard user of the RaspberryPi®. We tried to design an All-in-One device with Plug & Play capability. In fact, you do not need to set anything in order to use the **UPiS** module, just Plug it on the **P1** connector and Play with it! However, if you need more features, you can easily access them by jumpers, adjusting the **UPiS** exactly to your current project requirements.

The **UPiS** supports the user with a high number of features in a single low cost module. All of these features will be explained in detail in the following sections.

System components

The **UPiS** module consists of a number of subsystems and covers an extensive range of functionalities. A brief description of them is provided below in order to make easier to the user to understand the whole **UPiS** System. However, a detailed description of each functionality, the system setup/installation, basic system operation, superior system operation, and example applications are provided in depth in the next chapters.

UPiS (Basic Version) PCB View

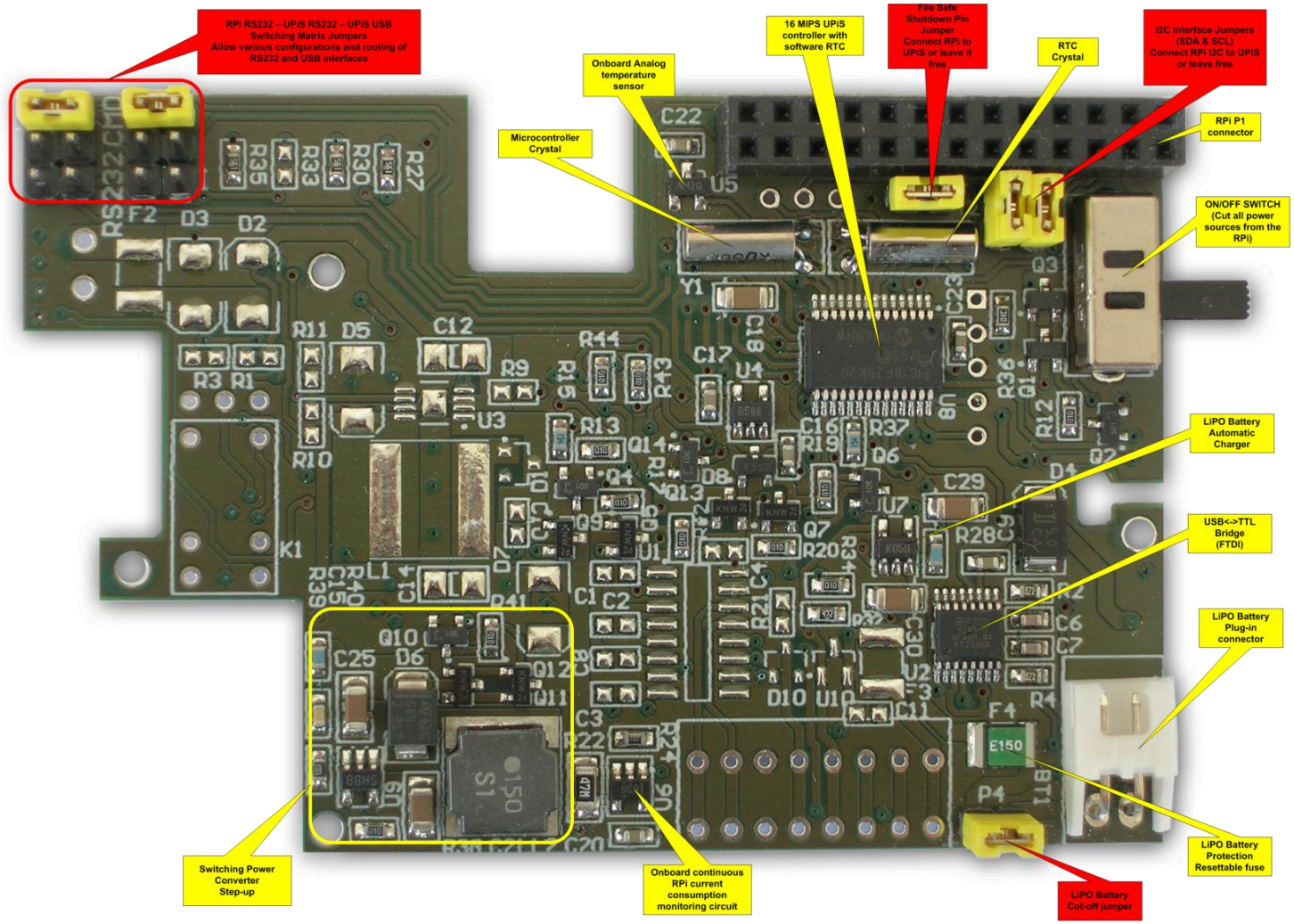

Figure 4 UPiS Basic Bottom PCB View

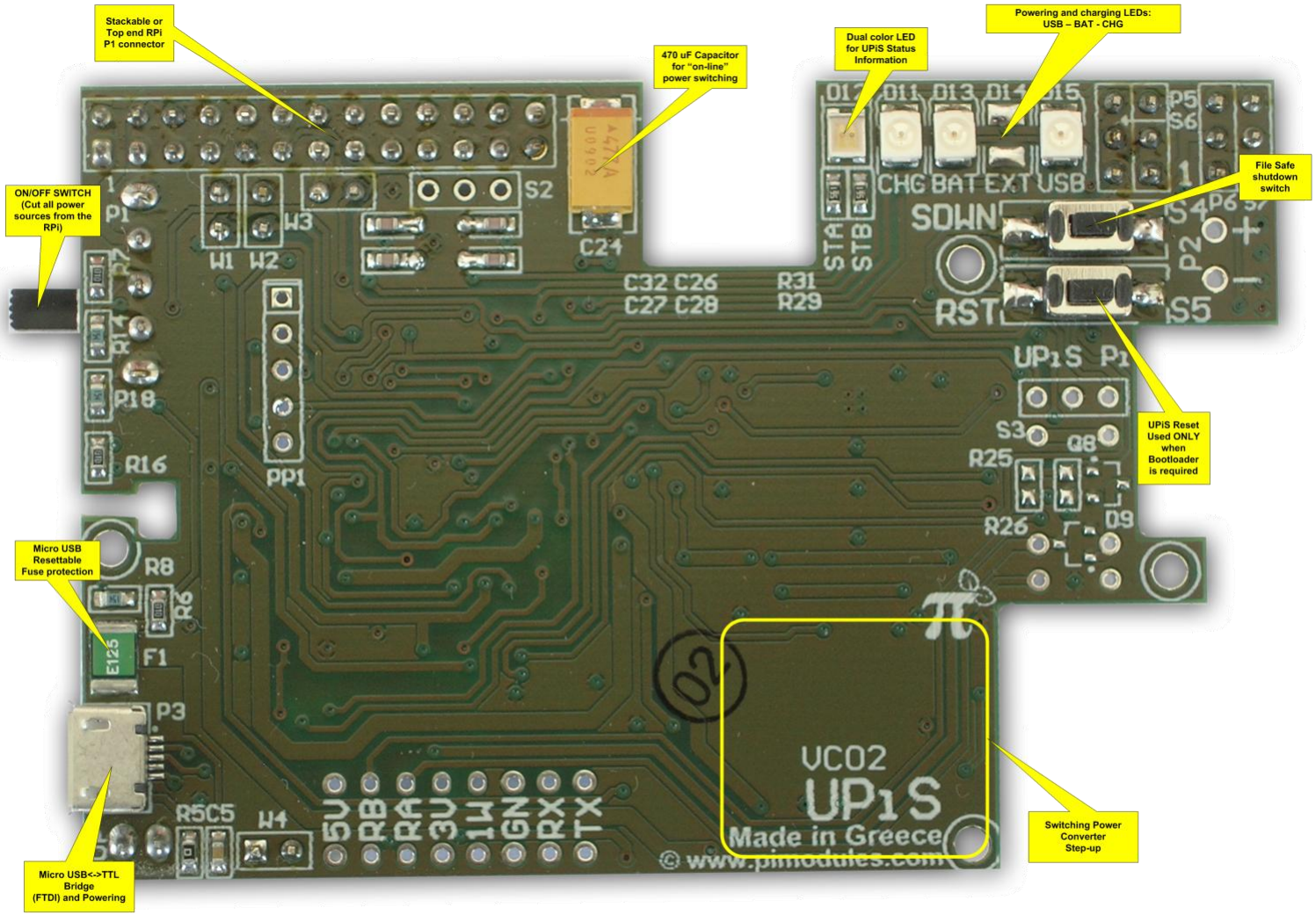

Figure 6 UPiS Basic Top PCB View

UPiS (Advanced Version) PCB View

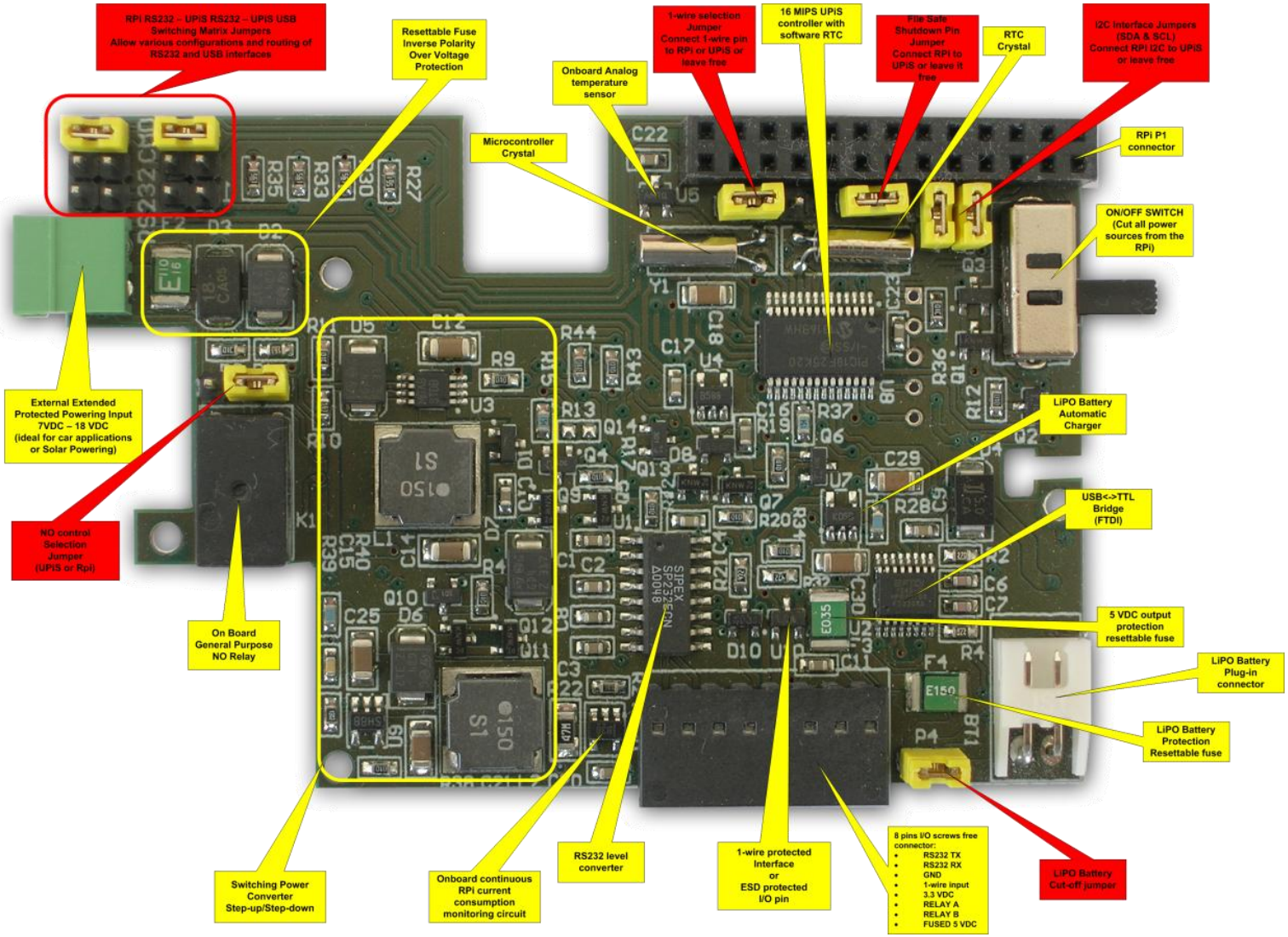

Figure 8 UPiS Advanced Bottom PCB View

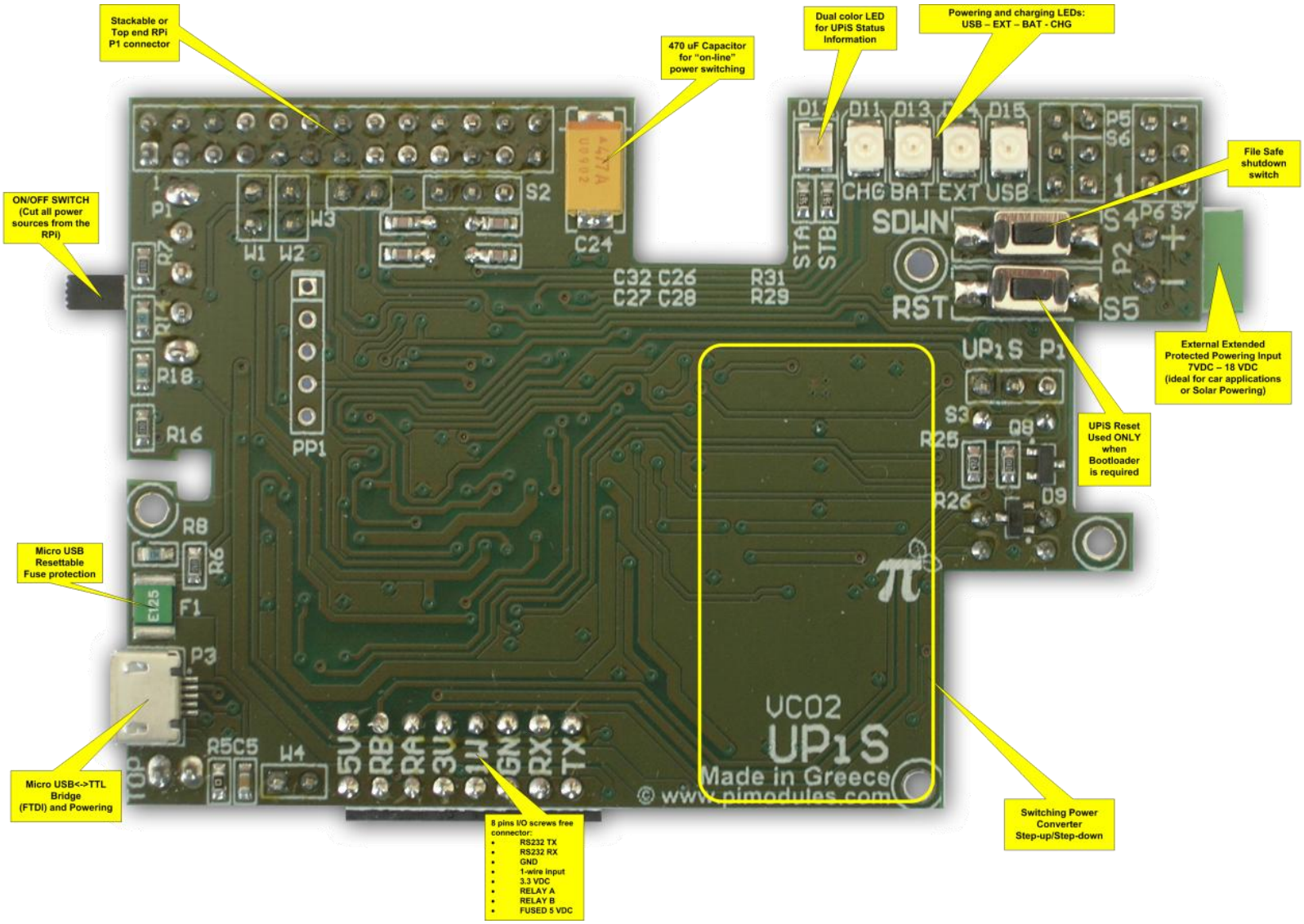

Figure 10 UPiS Advanced Top PCB View

RaspberryPi® micro USB (5 VDC) socket

Valid for the UPiS BASIC and UPiS ADVANCED

The external power supply can be connected directly to this input of the RaspberryPi® board, and the **UPiS** will still provide battery back-up to the RaspberryPi®. If the external power supply is connected directly to the RaspberryPi®, then the **UPiS** will use the **P1** connector both to monitor the powering status and to inject power into the RaspberryPi® when the sensed powering becomes insufficient. However, in this arrangement, some functionality of the **UPiS** is not available. In particular:

- The hardware switch has no effect
- All timed ON/OFF functionalities are not available
- The measurement of current consumption cannot be performed, so the functionalities that depend on it are not available in this powering option
- Low Powering Mode for the RaspberryPi® is not available.

Therefore, to take full advantage of the **UPiS** functionalities, it is recommended to use this powering input mode only if necessary.

Additional UPiS micro USB (5VDC) socket

Valid for the UPiS BASIC and UPiS ADVANCED

This Cable Powering Source is identical to the original RaspberryPi® micro USB and is protected in the same way (with SMB 5V and 1.25 PTC resettable fuse). If the external power supply of the RaspberryPi® is connected to this input of the **UPiS**, then the **UPiS** will feed the RaspberryPi® through the **P1** connector.

All the features of the UPiS can be exploited when using this connection scheme, therefore it is strongly recommended to use this or Extended External Powering connection scheme.

Extended External Powering Input (7 VDC – 18 VDC)

Valid for the UPiS ADVANCED only

This Cable Powering Source is designed to support the outside world of RaspberryPi® application. Through this Cable Powering Source, it can be supplied with power from 7 V DC up to 18 VDC. It is equipped with enhanced protection system which contains: reverse polarity protection, overvoltage protection as also PTC resettable fuse of 1.25 A. A dedicated switching converter converts the incoming power to 5 V DC. The maximum current is 1.4 A; however, for safety reasons, the current monitoring system allows a continuous draw of only 850 mA. This is ideal for car application, solar applications and basically whenever stable power supply is not available.

Onboard Rechargeable LiPO Battery (1150 mAh or 2600 mAh)

Valid for the UPiS BASIC and UPiS ADVANCED

The **UPiS** is equipped with a LiPO battery. It is the power source when cable powering is missing. There are two type of batteries offered depending of version of the **UPiS**. The **UPiS** Basic LiPO Battery capacity is 1150 mAh, and the **UPiS** Advanced LiPO Battery capacity is 2600 mAh. The LiPO batteries have been selected as a backup power sources due to the best relation between size, weight, price, and power capacity. However, special care is needed to avoid unexpected events like overheating, fire or even explosion.

Therefore, you are kindly requested to avoid any mechanical damage of such types of batteries like drilling, cutting or breaking. If you recognize such type of battery damage, please immediately disconnect it from the system and put in a safe place, before recycling it.

LiPO Battery Protection System

Valid for the UPiS BASIC and UPiS ADVANCED

The UPiS is equipped with a series of battery protections in order to avoid any dangerous incidents. There are:

- **Analog thermometer**

  The embedded UPiS microcontroller, based on the integrated analog thermometer, continuously checks the UPiS-PCB-System-Environment-Battery temperature. Whenever it exceeds the 60 Celsius Degrees, it stops the charging process. If it exceeds 60 Celsius Degrees, it initiates the UPiS emergency shutdown procedure and File Safe RaspberryPi® System shutdown. As a result, it cuts the system powering for security reasons.

- **PTC resettable fuse**

  The battery is connected to the system through a 2.6 A PTC resettable fuse. Whenever the battery current exceeds this value (during charging or powering process), the battery will be automatically cut-off from the system and will remain in this stage until powering requirements come back to the normal conditions.

- **Cut-off power jumper**

  Battery powering of the system is going through a cut-off jumper. In order to have the battery connected to the system, it is necessary to short it with a jumper. **It is strongly recommended, for security reasons, to keep this jumper open (battery disconnected) when the system is transported and not human supervised (i.e. via airplanes).** It is important to know that emulated RTC (running on the microcontroller) is powered by the same battery with the rest of the system. Therefore, disconnecting of the system battery will cause immediate loss of its settings.

- **Continuously current monitoring**

  The **UPiS** is equipped with embedded real-time current monitoring. The value of current consumption is continuously monitored by the microcontroller, and whenever it exceeds predefined values, the File Safe RPi System shutdown procedure is initiated. As a result, it cuts the system powering for security reasons.

In addition, each LiPO battery is equipped with protection PCB which controls the over-charge and over-discharge conditions of the battery. This circuit automatically cuts off the battery from the power source when the battery level is lower than 3 V during discharge and higher than 4.2 V during charging. This security system is implemented in order to save battery life, in addition to existing protections already embedded in the battery charger.

Intelligent Automatic LiPO Battery Charger

Valid for the UPiS BASIC and UPiS ADVANCED

UPiS has implemented an automatic charger that control the battery level and charges it as necessary. When the battery is full, it automatically switches to trickle charging in order to keep the battery in good condition. Trickle charging means charging a fully charged battery under no-load at a rate equal to its self-discharge rate, thus enabling the battery to remain at its fully charged level. It is an automatic process, and the user does not need to intervene. The charger is intelligent and fully microcontroller supervised. It is dynamically switching ON and OFF the charging process whenever the powering supply is not able to provide enough current for the RaspberryPi® and battery charging. The integrated battery is charged with a stable current of 212 mA.

Powering Modes

Valid for the UPiS BASIC and UPiS ADVANCED

The RaspberryPi® has various powering requirements depending on the connected power source and System Status. For example, powering requirements and functionalities are different when the System is running on the Battery or on USB cable powering (i.e. when the system is running on the Battery (the UPS feature - there is no battery charging)). Therefore, in order to support it, the **UPiS** follows System Power Requirements and switches automatically to various powering modes. There are 5 different powering states in the internal **UPiS** State Machine:

- USB - **USB** Cable Powering
- EPR - **E**xternal Cable **P**owe**R**ing
- BAT - **Bat**tery Powering
- LPR - Battery **L**ow **P**owe**R**ing
- RPI - **RPi** Powering

Switching between modes is done automatically by internal **UPiS** state machine and depends on various system parameters like Voltage on the **P1** connector (5 V), Voltage on the cable powering inputs (like EPR, USB), current consumption etc. Changes between various powering states can be monitored using scripting language, given user specified information about what powering mode is currently running. However, the switching procedure between each mode is completely automatic, so the user does not need to do anything in order to follow it. The powering mode information is addressed to more advanced users who need it for their dedicated application. A short description of each powering mode for when it is switching on is described below.

USB Mode - USB Cable Powering

The **USB Mode** is the basic running mode of the **UPiS** and is used in most applications. It is when the USB powering cable is connected to the **UPiS** micro USB connector and the RaspberryPi® is running drawing power from the **UPiS** through **P1** connector. The mechanical switch is switched ON. Absence of the **UPiS** micro USB cable power automatically switches the **UPiS** to the **BAT** mode when the RaspberryPi® system is running on power drawing from the internal battery. All functionalities of the System are available and running.

EPR Mode - External Cable Powering

The **EPR Mode** is the second cable powering mode of the **UPiS** and is used in the most embedded applications. This mode happens when Extended External Powering is connected to the power (7V DC – 18V DC) and the RaspberryPi® is drawing power from the **UPiS** through **P1** connector. The **EPR mode** has absolute priority in the **UPiS** System Powering. That means if you have connected 2 cables, the Extended External Powering Cable and the **UPiS** micro USB Cable, the system will automatically select powering from the EPR, and internally disconnect the micro USB cable from powering. Therefore, the micro USB Cable will be still connected, but will act only as a data connection if it is connected to PC. The mechanical switch is switched ON. Absence of the **UPiS** EPR cable power (and not availability of micro USB cable) automatically switches the **UPiS** to the **BAT** mode when the RaspberryPi® system is running on power drawing from the internal battery. All functionalities of the System are available and running.

BAT Mode – BATtery Powering

The **BAT Mode** is a UPS mode which happens when the Cable Power (the EPR, USB and RPI)is not available. When this occurs, the internal switching regulator produces the necessary power from the integrated battery and supplies the RaspberryPi® with power from the **UPiS** through the **P1** connector. Switching time is extremely short (less that 14 uS within 120 uS window), and the RaspberryPi® does not “know” that it is still powered from the battery. If Cable Power returns, then the system switches back to it within 3.6 seconds. This delay in switching back to the cable power is necessary in order make sure that the power source is

stable. Any instability of the cable power within this 3.6 second time could cause the resetting of the internal timer and counts the 3.6 seconds again until cable power is stable for this required time.

LPR Mode – Low Powering Mode

The **LPR Mode** is a variation of the BAT mode and occurs when the RaspberryPi® is not drawing power from the **UPiS**. Therefore, the **UpiS,** in order to save energy, switches off most of the peripherals, as well as the microcontroller and goes in to sleep mode. In this mode, it consumes only 60 uA and the only continuously running peripheral is the RTC. The UpiS periodically checks the Cable Powering Conditions and if it comes back, switches back to the appropriate Cable Powering mode or Battery Powering mode. This mode is also triggered by pressing the **SHDN** button (for a short time), it will start up the RaspberryPi® and switch the BAT mode back.

RPI Mode – RaspberryPi® Cable Powering

This is the final Cable Powering Mode and used very sporadically - the **RPI Mode**. It happens only when the RaspberryPi® is supplied from the original micro USB socket. Since we are still testing this powering mode (especially the battery powering exit to RPI mode), it is not recommended for use until the next version of firmware, which will be included in the full version. A more detailed description will be provided then.

Powering Current Peak Effect Protection

When using the RaspberryPi® current flow can be very high for brief periods of time; therefore, it is possible that the user will see a switch to BAT mode even if the cable is still connected. This is because the power available in the PSU cannot provide the necessary current needed during this short time. This causes a falling voltage range on the **P1** connector, and activates the BAT Mode in order save powering conditions of 5V on **P1** connector. This saves the RaspberryPi® from unexpected resets, and keeps the system powering on BAT Mode for at least 3.6 seconds until powering conditions stabilize again. All functionalities (except of battery charging) of the System are available and running.

Powering Mode	Description	Functionalities		LED Indications	
EPR	External Cable Powering	**EPR Input:**	Connected	**USB**:	Not Light/Light only if cable connected for data
		USB Input:	Disconnected (or only data)	**EXT**:	Light
		RPi USB Input:	Disconnected	**BAT**:	Light
		RPi Powering:	Powered	**CHG**:	Not Light/Light only when battery is charging
		RPi Switch:	ON	**STA**:	Blinking for 100 ms every 500 ms when cable power is connected
		Battery:	Auto/Intelligent Charging if needed		
		5VDC Output:	ON	**STB**:	Not Light (Light for 800 ms every 1 s within 40 s time frame when system is in File Safe Shutdown Procedure)
		Analog Temp:	ON		
		1-wire:	ON if connected		
		12 V RS-232:	ON		
		USB interface:	ON (Jumpers depending)		
		Relay:	ON/OFF		
		RTC:	ON		
		System Monitoring:	ON		
		Script Machine:	ON		
USB	USB Cable Powering	**EPR Input:**	Disconnected	**USB**:	Light
		USB Input:	Connected	**EXT**:	Not Light
		RPi USB Input:	Disconnected	**BAT**:	Light
		RPi Powering:	Powered	**CHG**:	Not Light/Light only when battery is charging
		RPi Switch:	ON	**STA**:	Blinking for 100 ms every 500 ms when cable power is connected
		Battery:	Auto/Intelligent Charging if needed		
		5VDC Output:	ON	**STB**:	Not Light (Light for 800 ms every 1 s within 40 s time frame when system is in File Safe Shutdown Procedure)
		Analog Temp:	ON		
		1-wire:	ON if connected		
		12 V RS-232:	ON		
		USB interface:	ON		
		Relay:	ON/OFF		
		RTC:	ON		
		System Monitoring:	ON		
		Script Machine:	ON		
BAT	Battery Powering	**EPR Input:**	Disconnected	**USB**:	Not Light
		USB Input:	Disconnected	**EXT**:	Not Light

		RPi USB Input:	Disconnected	**BAT**:	Light
		RPi Powering:	Powered	**CHG**:	Not Light
		RPi Switch:	ON	**STA**:	Blinking for 100 ms every 500 ms when cable power is connected
		Battery:	Not charged		
		5VDC Output:	ON	**STB**:	Not Light (Light for 800 ms every 1 s within 40 s time frame when system is in File Safe Shutdown Procedure)
		Analog Temp:	ON		
		1-wire:	ON if connected		Light for 100 ms every 1000 ms when UPiS is running on battery backup power and battery level is higher than 3.4V and lower than 3.6V
		12 V RS-232:	ON		
		USB interface:	ON		
		Relay:	ON/OFF		Light for 100 ms every 500 ms when UPiS Advanced is running on battery backup power and battery level is lower than 3.4V and higher than 3.2V
		RTC:	ON		
		System Monitoring:	ON		
		Script Machine:	ON		
LPR	**Low Powering**	**EPR Input:**	Disconnected	**USB**:	Not Light
		USB Input:	Disconnected	**EXT**:	Light
		RPi USB Input:	Disconnected	**BAT**:	Not Light/Light periodically (every LPRSTA)
		RPi Powering:	Not Powered	**CHG**:	Light only when battery is charging
		RPi Switch:	OFF/ON and SHDW	**STA**:	Not Light
		Battery:	Not charged		
		5VDC Output:	OFF	**STB**:	Not Light
		Analog Temp:	OFF/ON periodically (every LPRSTA)		
		1-wire:	OFF		
		12 V RS-232:	OFF		
		USB interface:	OFF		
		Relay:	OFF		
		RTC:	ON		
		System Monitoring:	OFF/ON periodically (every LPRSTA)		
		Script Machine:	OFF		
RPI	**RPi Powering**	**EPR Input:**	Disconnected	**USB**:	Light
		USB Input:	Disconnected	**EXT**:	Not Light
		RPi USB Input:	Disconnected	**BAT**:	Light
		RPi Powering:	ON	**CHG**:	Not Light/Light only when battery is charging
		RPi Switch:	ON	**STA**:	Blinking for 100 ms every 500 ms when cable power is connected
		Battery:	Auto/Intelligent Charging if needed		
		5VDC Output:	ON	**STB**:	Not Light (Light for 800 ms every 1 s within 40 s time frame

	Analog Temp:	ON	when system is in File Safe Shutdown Procedure)
	1-wire:	ON if connected	
	12 V RS-232:	ON	
	USB interface:	ON	
	Relay:	ON/OFF	**THE RPI MODE IS NOT FULLY IMPLEMENTED YET**
	RTC:	ON	**WILL BE AVAILABLE IN NEXT FIRMWARE UPDATES**
	System Monitoring:	ON	
	Script Machine:	ON	

Figure 12 UPiS Powering Modes: Description - Functionaliries - Inficators

Additional Protected 5 VDC 140 mA source for user applications

Valid for the UPiS ADVANCED only

The RaspberryPi® can be used for an extremely wide range of applications. Many of them are using additional components that require powering. In order to support this range of applications the **UPiS** is offering a 5 VDC power source protected with a PTC resettable fuse. This power source has battery backup for the whole "UPiS + RaspberryPi®" system.

RaspberryPi ® Mechanical ON/OFF Switch

Valid for the UPiS BASIC and UPiS ADVANCED

The RaspberryPi® Computer is an ingenius system. However, due to cost reasons some components are missing. One of them is the Mechanical ON/OFF switch. Users would sometimes like to switch their computer ON or OFF rather than unplugging the cable. In order to overcome this dysfunction, the **UPiS** offers a mechanical switch that completely isolates the power from the RaspberryPi®. This functionality is available only when power is going through the **UPiS** (the micro USB or the Extended External Power). It is not available if you are powering your RaspberryPi® from the original micro USB on the computer board.

Embedded on Board Analog Temperature Sensor accessible via RS232 interface

Valid for the UPiS BASIC and UPiS ADVANCED

The RaspberryPi® ICs, as with the **UPiS,** generates heat. It is important for the user to know at all times what temperature the system is, and if it is too high, take an action (i.e. switch off or start fan with embedded Relay). For that reason a simple analog thermometer has been implemented on the UPiS board. The value of the temperature can be read via RaspberryPi® RS232 port. Its usage will be explained later in this document; however, it is very important to notice that it is possible to setup a trigger on a given temperature and take an action on a high or low temperature automatically, independent of the RaspberryPi®. For security reason the temperature of the UPiS cannot exceed 60 Celsius due to the battery's life cycle. Therefore, if temperature is higher than 60 Celsius, the **UPiS,** as the first step of protection, stops the charging of the battery in order to reduce current consumption. If the temperature is still too high, an emergency shutdown procedure is initiated in order to save the battery and after 40 seconds the power to the RaspberryPi® is stopped.

User Controller onboard NO Relay

Valid for the UPiS ADVANCED only

There are plenty of embedded applications where the RaspberryPi® can be successfully used. Most of them basically require simple switching of "something" after measure or input of "something". Therefore the simple control of a NO Relay is very important in order to cover such simple things in the embedded world. The implemented of the **UPiS** NO Relay can

be controlled directly from the RaspberryPi® via the GPIO_GEN0 or using the command @RON for the Relay ON and @ROFF for the Relay OFF directly from the serial terminal program interface. Which interface will be selected (RaspberryPi® direct GPIO or Serial Terminal) is set by the appropriate jumper setting. It is very important to note that embedded NO Relay should be used for low voltage and low current applications (recommended 24VDC and 500mA). If there is a need to switch higher voltages or current it is recommended to use this NO Relay as a intermediate switch for a more heavy Relay able to safely handle higher power applications.

1-wire interface

Valid for the UPiS ADVANCED only

As it has been described above in the embedded applications, there is usually a need to measure "something". The 1-wire interface is perfect to do this task as there arevarious MEMs equipped with this well known interface. Similar to the NO Relay, it can be handled directly from the RaspberryPi via the GPIO_GEN3, or by using the command @1-wire directly from the serial terminal program interface. The interface which will be selected (RaspberryPi® direct GPIO or Serial Terminal) is set by the appropriate jumper setting. This I/O pin is ESD protected and also contains the required pull-up resistor. Therefore, it is ready to use. Due to wide range of various sensors compatible with the 1-wire interface, the scripting language has been implemented only for the 1-wire temperature sensor interface. The appropriate commands handling this sensor are @1-wirec (for the temperature reading in Celsius degree) and @1-wiref (for temperature reading in Fahrenheit degree). The required 3.3 V for the pull-up resistor is provided from an independent regulator in order to protect the original RaspberryPi® from possible shortcircuits.

iButton interface

Valid for the UPiS ADVANCED only

Another interface which uses the same 1-wire I/O pin is the iButton. These sensors open a wide area of security applications where the RaspberryPi® can use its networking advantage. This sensor protocol will be available for the user in the next firmware releases. A detailed description of the associated Scripting Language Commands will be provided then.

ESD Protected I/O pin interface

Valid for the UPiS ADVANCED only

The same I/O pin is not assigned to the above interfaces, but it can be used as a simple High/Low detector i.e. for a magnetic ON/OFF sensor or any other binary output sensor. Handling it can be done directly from the RaspberryPi® now, or after the next firmware update via the Serial Terminal Program.

USB-RS232 Bridge

Valid for the UPiS BASIC and UPiS ADVANCED

Handling the RaspberryPi® RS232 port is just as important as and the recovery procedure that is available through it, there are also plenty of applications where the RS232 port can be used. An RS232 port with a 12V interfaces is not so common nowadays. However, there are interfaces available on the market that convert the RS232 port to a USB port (with driver called Virtual Com Port). These devices are called USB – RS232 Bridges. In order to save time and money, the **UPiS** has already implemented such interface and offers this USB functionality through the **UPiS** Serial Ports Switching Matrix. A detailed description about this Switching Matrix is provided in the next chapters.

RS232 Interface Level Converter

Valid for the UPiS ADVANCED only

Similar to the above interface, the **UPiS** offers a pure RS232 12V interface for applications that require this original type of interface. This functionality is also available through the **UPiS** Serial Ports Switching Matrix. A detailed description about this Switching Matrix is provided in the next chapters.

UPiS Serial Ports Switching Matrix

Valid for the UPiS BASIC and UPiS ADVANCED

There are available a number of serial ports on the UPiS and RaspberryPi® system. The two devices are capable of being cross connected in a variety of configurations. Connectivity between them is covered from the **UPiS** Serial Ports Switching Matrix. There is a set of jumpers that can be set in various configurations making different serial/USB ports accessibile. A detailed description about the Switching Matrix is provided in the next chapters.

On-Board Simulated Real Time Clock – DS1307

Valid for the UPiS BASIC and UPiS ADVANCED

The RaspberryPi® does not contain a Real Time Clock with battery backup. It contains an RTC but when the system goes down all time information is lost, if there is a loss in connectivity with the network then the RTC also looses all time information. Therefore, there is a need to have a Real Time Clock with battery backup. This functionality is covered by the UPiS via a software simulated RTC. A detailed description of functionality and jumper setting is provided in the proper chapter.

File Safe RaspberryPi® Shutdown Button

Valid for the UPiS BASIC and UPiS ADVANCED

Switching OFF the RaspberryPi® without first initiating the shutdown command, in order to close open files, can be disasterous. This has happened many times, and many times it has caused errors on the SD card and difficulties on the proper running system. In order to support such functionality with just a single button, the File Safe RaspberryPi® Shutdown Button is offered. After installing the proper script, the RaspberryPi® can safely shutdown simply by pressing this button. After pressing the system shutdown button, all files will be closed. This is described in details in next chapters including a sample of the required script written in python.

RaspberryPi® Reset Button

Valid for the UPiS BASIC and UPiS ADVANCED

TBC

Screws free I/O Terminal Block

Valid for the UPiS BASIC and UPiS ADVANCED

TBC

System Parameters Real-Time Monitoring

Valid for the UPiS BASIC and UPiS ADVANCED

TBC

Additional Independent 3.3 VDC 100 mA output for user applications

Valid for the UPiS ADVANCED only

TBC

Cryptographic Software Protection based on Extended Tiny Algorithm (XTEA)

Valid for the UPiS BASIC and UPiS ADVANCED

The RaspberryPi® is a very well designed microcomputer. Due to its very low cost, as well as good quality and huge users' base, it is widely used as an embedded computer for various

applications. Most of them are customer dedicated applications. Our company, in order to protect developers from illegal copy of their embedded applications, offers a software protection key based on XTEA cryptographic algorithm. This feature is currently in testing and will be released for free in one of the next firmware upgrades according to the firmware updates road map listed at the beginning of this document. Developers can protect their applications using their own program keys stored in the **UPiS** microcontroller Flash memory with read protection written on the microcontroller silicon core. Access to it is done via RaspberryPi® RS232 Ports. User has encryption on both paths: reading and writing. A detailed description on the usage of this feature will be provided with related firmware release.

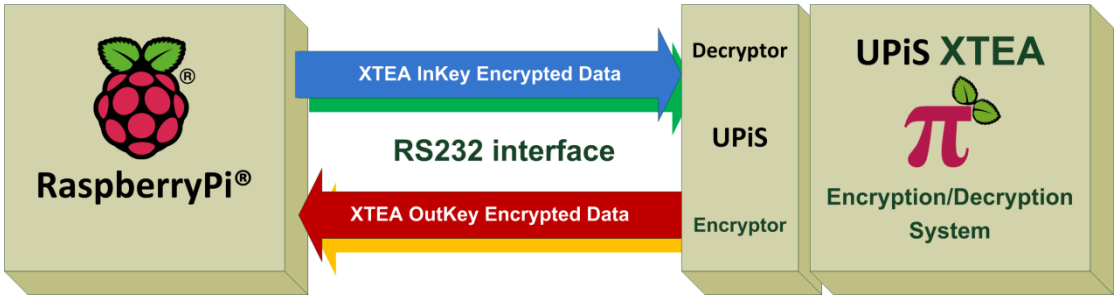

Figure 13 UPiS XTEA Encryption/Decryption System Layout

LEDs based User Simple Interface

Valid for the UPiS BASIC and UPiS ADVANCED

There are 4 single color LEDs and 1 dual color LED that helps the user to see what happen with the system without accessing any command.

- **USB GREEN LED**
  Lights only when USB cable is connected. It does not mean that the system is powered from the USB; it means only that USB cable is connected. Whether or not the **UPiS** System is drawing power from it depends on other connections

- **EXT GREEN LED**
  Lights only when Extended External Power cable is connected. If this cable is connected the USB powering is automatically disconnected.

- **BAT GREEN LED**
  Lights only when battery switching boost converter is working. It does not mean that UPiS/RaspberryPi® is powered from it, this indicates only that switching converter is powered from battery. Whether it is connected to the system or not is a separate decision of the UPiS.

- **CHG GREEN LED**

Lights only when battery is charged (drawing current from the UPiS system).

- **STA GREEN LED**
  Lights different in various states of the UPiS module. A detailed description is provided in the table XX.

- **STB RED LED**
  Lights differently in various states of the UPiS module. A detailed description is provided in the table XX.

Terminal Based User Interface

Valid for the UPiS BASIC and UPiS ADVANCED

The **UPiS** has an implemented Terminal Based User Interface. In order to handle it the output of the **UPiS** Serial Port should be connected to a Terminal Program. There are two types of interfaces that can be connected the serial port of the UPiS microcontroller:

- The RaspberryPi® RS232 Port, or
- The USB <->RS232 interface connected to any personal computer

For the Terminal Program, it could be the RaspberryPi® Linux running a program like minicom, or a program based on a personal computer equipped with any Operating System and connected via USB to **UPiS.**

Scripting Language

Valid for the UPiS BASIC and UPiS ADVANCED

The **UPiS** models are equipped with a variety of sensors. The user can handle them from the RaspberryPi® or directly through the **UPiS**. The selection of which sensor is connected to RaspberryPi® is done using a set of Jumpers. However, some sensors (i.e. analog temperature sensor or current measure system) can be accessed only trough the **UPiS** and read by the RaspberryPi® using the serial port. Based on the sensors, simple actions can be setup using the Advance Scripting Language omitting involvement of the RaspberryPi® i.e. “shutdown RaspberryPi® when temperature is too low”, or “switch ON Relay when temperature is higher than” etc. It can significantly simplifying some processes that could be running independent of the RaspberryPi® even if it is shutdown i.e. cooling fan switching ON/OFF, attaching and powering from a much bigger battery the RaspberryPi® etc.

Bootloader

Valid for the UPiS BASIC and UPiS ADVANCED

Both **UPiS** models are equipped with bootloader functionality. The bootloader is a functionality that allows the **UPiS** system firmware to be uploaded on the user site with the newest version. This ensures that the **UPiS** is flexible and always with the latest firmware. New versions of the firmware are announced on the www.pimodules.com and can be downloaded by the user. The **UPiS** firmware is smart enough and automatically recognizes which hardware model of **UPiS** is running, adjusting the available functionality set to it. The boot loading procedure uses the micro USB interface on the **UPiS**. A detailed description how to do this is provided in the next sections. The Updates Road Map is provided at the beginning of this document.

Basic System Operations

What is in the BOX?

Valid for the UPiS BASIC and UPiS ADVANCED

This package comes with everything you need to start using the **UPiS** right out of the box. It is assembled, tested and contains all required accessories. All Jumpers (except the Battery) are connected to the start-up configuration. A little work is necessary in order to setup the complete RaspberryPi® and **UPiS** in a single full operating system.

Each Box contains the following parts:

- The **UPiS** module assembled and tested
- Dual layer wide temperature adhesive tape (used for battery mounting) stuck on the bottom side of **UPiS battery**
- Set of installed necessary jumpers (yellow)
- Separate packed LiPO battery (1150 mAh for the **UPiS** Basic, and 2600 mAh for the **UPiS** Advanced)
- A male green connector used for the External Extended Powering – only for the **UPiS** Advanced
- A spare 1 Yellow Jumper

Please kindly notice that due to shipping regulations, LiPO batteries are packed in the same box but are physically separated and not connected to the **UPiS** module. It must be connected by the user, and it is a part of the installation procedure. Please also kindly notice that in order to increase the safety of usage of the **UPiS** module all sources of the unprotected power (mainly the enter point of the LiPO battery, and Extended External Power input voltage – 18VDC) are covered by a special isolation in order to avoid any damages of the device due to shortcuts made by improper use.

System Installation

Valid for the UPiS BASIC and UPiS ADVANCED

The installation procedure of the UPiS Modules is divided into the following phases:

- The Battery assembly with the UPiS Modules
- UPiS Module Jumpers check and setup (optional)
- Physical installation of the RaspberryPi®
- RaspberryPi® Software setup (optional)

Battery Assembly

Valid for the UPiS BASIC and UPiS ADVANCED

In order to install battery, you need to unpack it and connect the battery plug to the white connector on the PCB as shown on the picture.

TBC

PICTURE

In addition the battery cable should be conducted through the slot on the UPiS Modules PCB. Then, protection from the wide temperature adhesive tape should be removed and battery should be glued on the top of the UPiS. This operation is not necessary, and battery can be placed in any other place. The following pictures show step by step above procedure.

TBC

PICTURE

UPiS Module Jumpers setup

Valid for the UPiS BASIC and UPiS ADVANCED

The **UPiS** module interfaces with the RaspberryPi® only via 2 pins trough the **P1** connector:

- 5V DC
- GND

All other pins are free to use depending to the user needs. Selection of which other pins are used is made using jumpers that connect **UPiS** selected add-ons peripherals to RaspberryPi®. Each peripheral can be controlled via selected jumper connecting it to the RaspberryPi® or via **UPiS** serial port using proper commands from a terminal program running on the RaspberryPi® or any other computer connected to the **UPiS**. Selection how the required **UPiS** Peripheral will be used is always made by a proper jumper set-up. The peripherals controlled and activated by Jumpers are listed here below:

1. The LiPO battery Cut-off jumper
1. Embedded Emulated RTC (Real Time Clock – DS1307) I2C pins

2. Onboard RS232 – True USB Bridge
3. RaspberryPi® File Safe Shutdown Pin
4. Onboard NO RELAY controlled via RS232 or RaspberryPi® Pin (selectable by jumper GPIO_GEN0)
5. Onboard ESD Protected 1-wire interface, controlled via RS232 or RaspberryPi® Pin (selectable by jumper GPIO_GEN3) with separate 3.3V supply pull-up resistor.
6. Onboard ESD Protected I/O pin, controlled via RS232 or RaspberryPi® Pin (selectable by jumper GPIO_GEN3)
7. Onboard True 12 V RS232 interface to the external world (level converter)

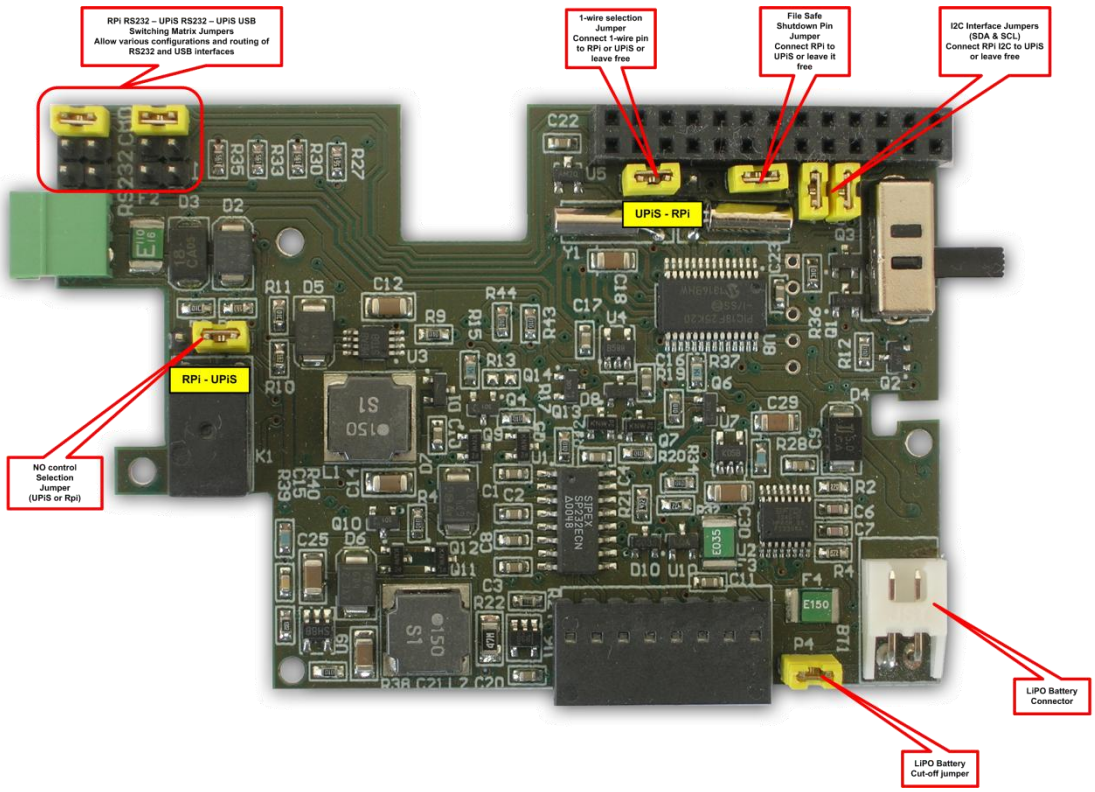

Figure 14 UPiS Advanced Jumpers Places

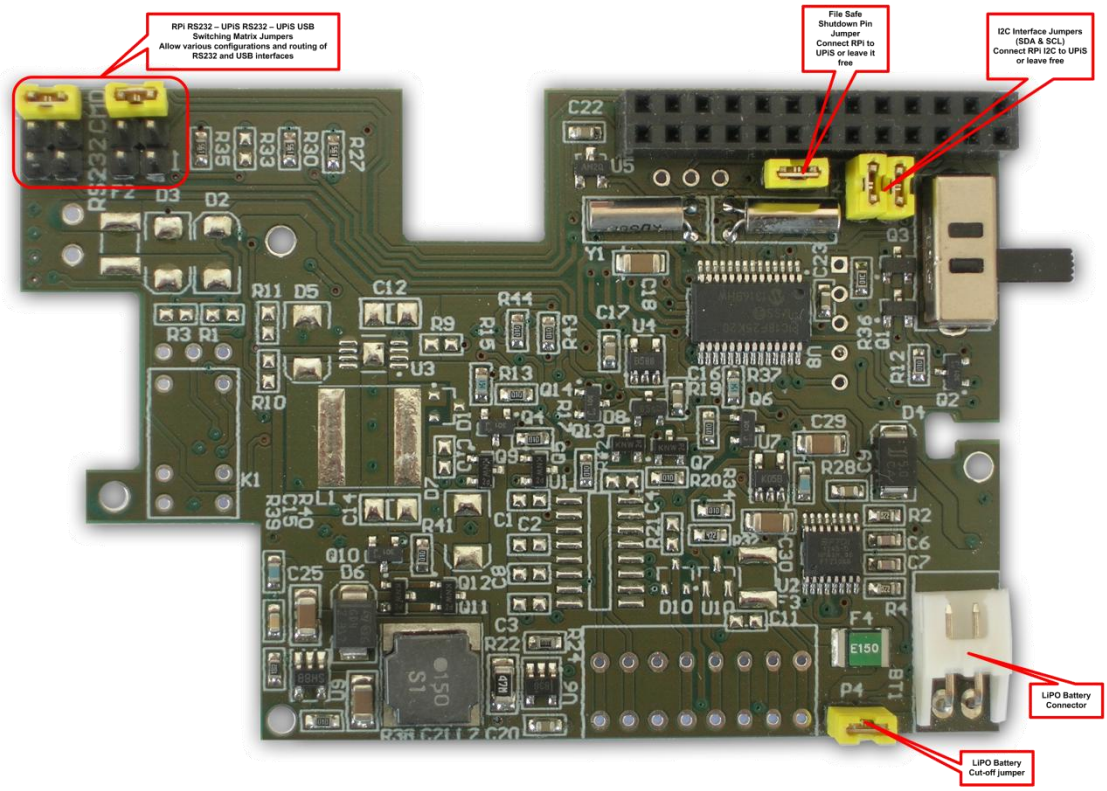

Figure 15 UPiS Basic Jumpers Places

Screws Free Terminal Block

Valid for the UPiS ADVANCED only

The **UPiS** Advanced is equipped with various peripherals. Most of them have external world connectivity. There are cumulated in one Terminal Block in order to allow the user to connect with cables to required external devices. For this reason it has been selected as the most advanced type of Terminal Block since there is no need to use screws. For Cable release all that is needed is a small screwdriver that will fitin to the square hole above the cable. The picture below shows the places where the cable and the tool have to be entered.

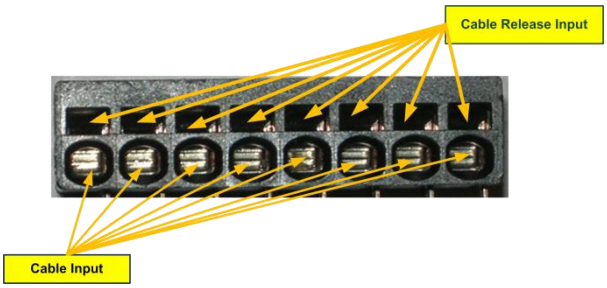

Figure 16 Screw Free Terminal Block I/O

I/Os used on the Terminal Block are the following and marked as listed below:

- **5V** – protected 5V DC output 140 mA
- **RB** – NO Relay Pin B (Normally Open, will be short with PIN A if activated)
- **RA** – NO Relay Pin A (Normally Open, will be short with PIN B if activated)
- **3V** – 3.3 VDC from separated and independent form RaspberryPi® usually used with 1-wire interface for powering. Can be used for any other application maximum 100 mA
- **GN** – UPiS and Raspberry System GND
- **RX** – RS232 Level Converter RXD (Receives Data)
- **TX** – RS232 Level Converter TXD (Transmits Data)

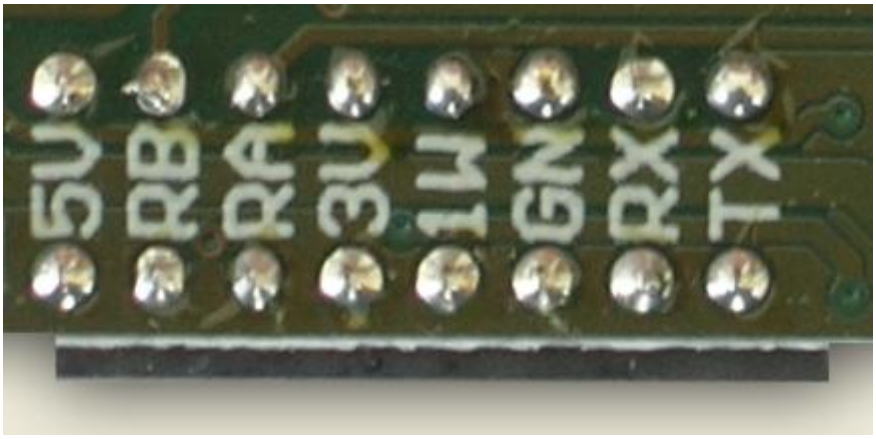

Figure 17 Screws Free Terminal Block

UPiS Serial Ports Switching Matrix

Valid for the UPiS BASIC and UPiS ADVANCED

The system which is folded from the RaspberryPi® and the **UPiS** use for communication with external world and between them serial interfaces. There are:

- RaspberryPi® Serial Port
- UPiS Serial Port
- UPiS micro USB port
- UPiS RS232 Level Converter

Cross connectivity between them is extremely important in order to support the user with the best possible implementaion.

In order to achieve this goal there was a Serial Ports Switching Matrix designed which is based on jumper selection. A detailed description is provided below in its proper section.

UPiS NO Relay Controls

Valid for the UPiS ADVANCED only

The **UPiS** Advanced offers a single low power Normally Open Relay. The outputs of its pins are placed on the screwless 8 Pin I/O Terminal: **RA** and **RB**. It is shown here below.

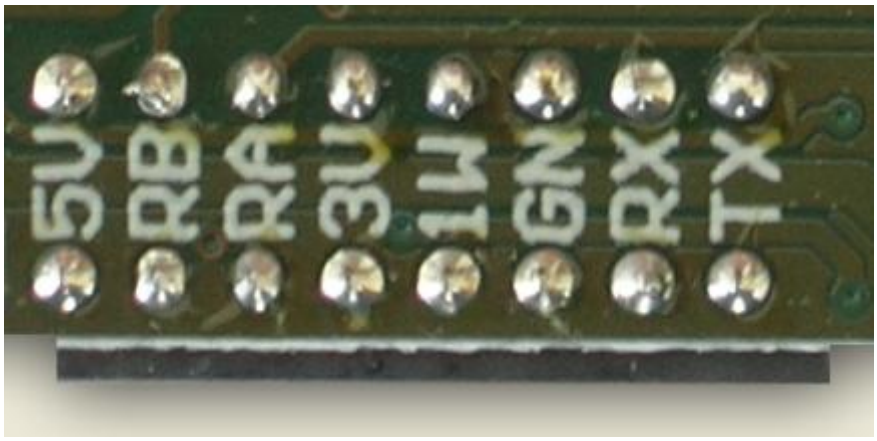

Figure 18 Top View - Screws free I/O 8 Pins Terminal

This NO Relay can be controlled by the **UPiS** commands through the terminal program or directly by the RaspberryPi® from a dedicated Pin GPIO_GEN0 (P1 Connector PIN 11). Selection what controls over the NO Relay is done via selection with a Jumper **S3** as shown on the picture here below.

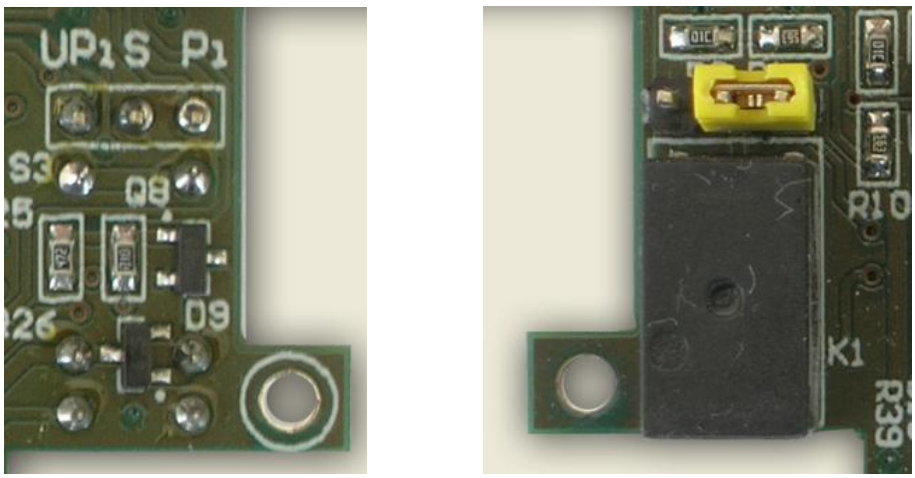

Figure 19 S3 Top and Bottom View – UPiS Control Selected

In the case where the RaspberryPi® has been selected to control the NO Relay it is the user's responsibility to write an appropriate script or program handling this Pin. In the case that the

UPiS has been selected to control the NO Relay, it can be done via terminal from the RaspberryPi® or PC or any other application accessing the Serial Port on the RaspberryPi®. The appropriate commands to do this are:

- @RON – Relay ON (Close)
- @ROFF – Relay OFF (Open)

Battery cut-off jumper

Valid for the UPiS BASIC and UPiS ADVANCED

The integrated battery is equipped with a variety of security protections. One of them is the cut-off jumper – **P4**. The function of this jumper is to galvanic isolate the battery from the **UPiS** system. It is important to note that by removing this jumper, battery is not charged, and the UPiS system is not supplying peheripals including RTC. Therefore, opening it is necessary only when **UPiS** is shipped via Air Plane for security reasons. For normal usage of the **UPiS** this jumper should be always closed.

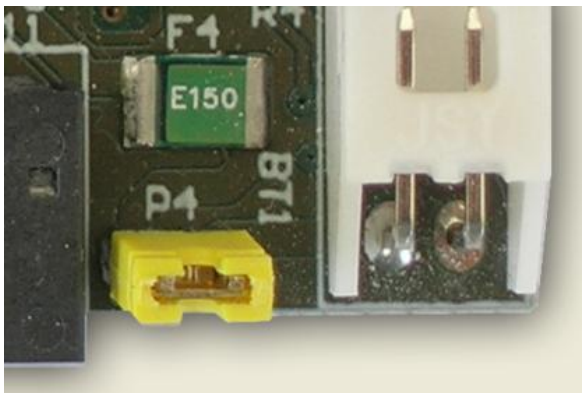

Figure 20 Bottom View Battery Cut-Off Jumper

1-wire selection Jumper

Valid for the UPiS ADVANCED only

The **UPiS** Advanced has an embedded interface for the 1-wire® sensors. This interface is ESD protected. This 1-wire interface can be controlled by the **UPiS** commands through the Terminal Program or directly by the RaspberryPi® from a dedicated Pin GPIO_GEN3 (P1 Connector PIN 15). Selection of what controls over the 1-wire is done via selection with a Jumper **S2** as shown on the picture here below.

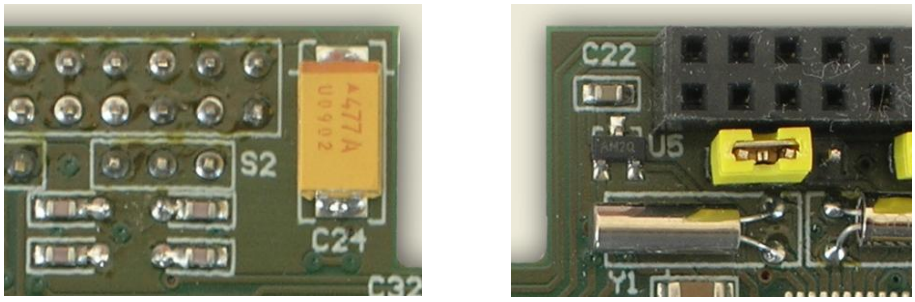

Figure 21 S2 Top and Bottom View – UPiS Control Selected

In a case where the UPiS has been selected to control the 1-wire® interface, it can be done via terminal from the RaspberryPi® or PC or any other application accessing the Serial Port on the RaspberryPi®. The appropriate commands to do it are:

- @1WIREC - 1-wire® Temperature Celsius
- @1WIREF - 1-wire® Temperature Fahrenheit

In the future release of firmware our company is planning to release commands for the i-button® security key.

I²C Connection Jumpers (RTC)

Valid for the UPiS BASIC and UPiS ADVANCED

The **UPiS** is equipped with RTC which is supplied by the same battery as the rest of the system. The RTC is a software emulated DS1307. It communicates with the RaspberryPi® identical with the original DS1307. In order to have connectivity with the RTC the user will need to short 2 jumpers of the I^2C (the SDA and SDL). A detailed description how to use and set-up the **UPiS** RTC is described in another part. Shown below is the connected I^2C to the RaspberryPi®. If the user uses another application where these pins are needed the should be left free. Please kindly notice that it is possible to use multiple I^2C devices on the same bus.

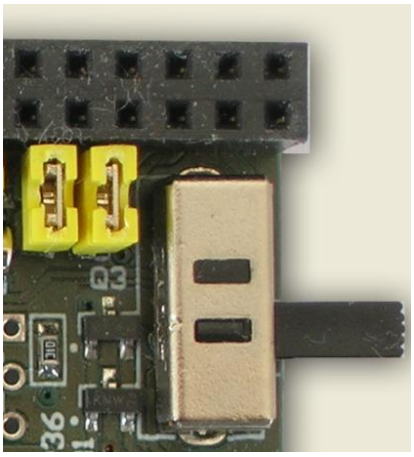

Figure 22 Bottom View - I²C connected to the RaspberryPi

File Safe Shutdown initiation Pin Jumper

Valid for the UPiS BASIC and UPiS ADVANCED

The File Safe Shutdown feature gives the user a proper way to shutdown the RaspberryPi® simply by pushing a single button. However, this procedure needs to be recognized by the RaspberryPi®, and for that reason, a dedicated jumper needs to be closed (set). If it is open then the RaspberryPi® will never know that somebody is requesting the system shutdown. The detailed procedure for setting-up the shutdown is described here below.

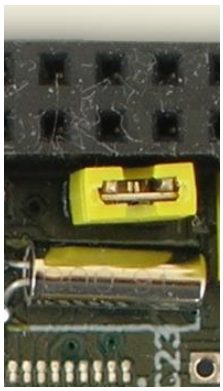

Figure 23 Bottom View the File Safe Shutdown Jumper

RaspberryPi® File Safe Shutdown Procedure and RaspberryPi® Reset

Valid for the UPiS BASIC and UPiS ADVANCED

In order to support the **File Safe Shutdown** procedure a simple script should be written and stored on the RaspberryPi®. There are many simple scripts written concerning this matter

and can be easy found over the internet; however, we provide one example that can be easy implemented. Scripts could be divided into two basic categories:

- Interrupt based and
- Loop based.

User of the **UPiS** module is basically free to use their own script; however, the user should always keep in mind some of the basics of the implemented circuit on the **UPiS** board:

- There are no Pull-Up resistors on the **UPiS** board therefore user needs to setup the RaspberryPi® resistors
- The Pin which has been dedicated to this task is the pin GPIO.27
- Before this functionality will be used user need to put a proper jumper on the **UPiS** Board otherwise it will not be working as the input pin GPIO.27 is continuously scanning to see if low state is.
- If user does not need this functionality, or need this pin, the GPIO.27 pin could be used for other applications and the associated jumper should be open (removed).

There are two basic functionalities associated with the **Shutdown Button.** There are:

- Reset Functionality
- File Safe Shutdown (if a proper script is applied to the RaspberryPi®)

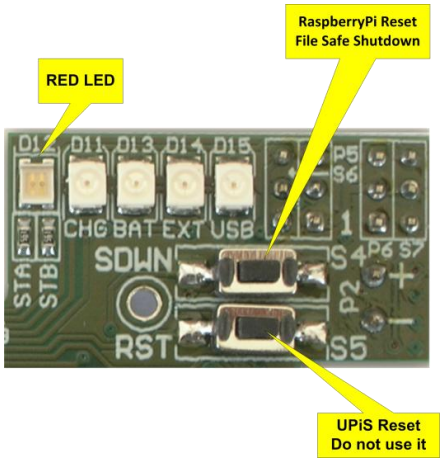

Figure 24 Top View SDWN and RST Buttons

The **Reset Functionality**, is executed when the **Shutdown Button** is pressed for longer than 0.3 second and shorter than 2 second. In practice, it means that the user will need to press the button a little bit longer before releasing it. When the button is pressed, the **UPiS** module will immediately cut power to the RaspberryPi® for 1 second before powering it up again. Cutting power this way is just like the rest and it immediately stops all functionalities of the RaspberryPi®.

Please kindly notice that using of this functionality can cause of file system destroy in the SD card, therefore it is **STRONGLY RECOMMENDED** to use it ONLY if the RaspberryPi® does not response to any input device.

The **File Safe Shutdown Functionality** does not have this limitation, and therefore saves files from any corruption. This is executed when the **Shutdown Button** is pressed for a longer time than 2 seconds. In order to simplify its usage, when UPiS Module recognizes that the button for the **File Safe Shutdown** is pressed it lights the RED led for 1 second, and then flash it every second for 800 ms. In practice, it means that the user needs to press the button until the RED led light and releases it just after that. Following this procedure will see the flashing RED led. This provides optical confirmation that the **File Safe Shutting** down procedure has been initiated and it is in progress. It takes 40 seconds (on the **UPiS** module), and after that you will see that RaspberryPi® has been stopped. If you have initiated the terminal on the RaspberryPi®, or connected other computer via USB, you will also see the following messages on the screen:

UPiS System Stared File Safe Shutdown Procedure

And after finishing the shutting down:

UPiS System Finished File Safe Shutdown Procedure

Here below describes the simple procedure on how to implement the simplest Python script used for the **Safe File Shutdown**.

Open Terminal from the command line or open an LXTerminal session, and use the Nano text editor to add some code to enable the Python script created to run when the RaspberryPi® boots up. Type in:

sudo nano /etc/rc.local

and then add in the following code:

sudo python /home/pi/upis/fshut.py

before the line that says:

exit 0

Press Ctrl+X to exit the Nano editor and when prompted press Y and then Enter in order to save the file you just edited.

Next prepare the directory where the script will be place i.e. “upis” using the following command:

mkdir upis

Using nano editor, write the following script and save on pre-prepared directory “upis” with a name fshut.py

sudo nano /home/pi/upis/fshut.py

The screen below shows the script that need to be entered using nano.

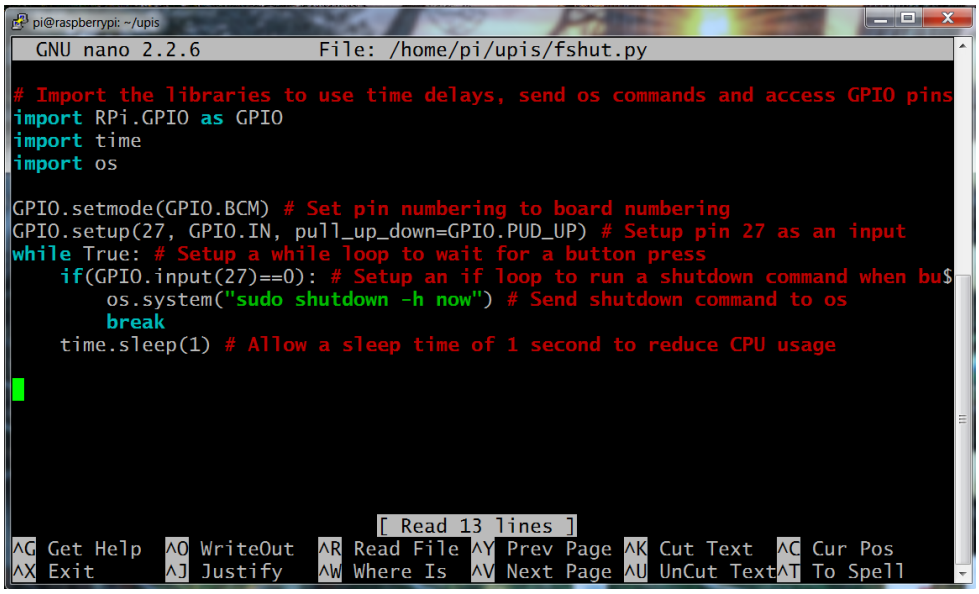

Figure 25 File Safe Shutdown Python Script - fshut.py

```
# Import the libraries to use time delays, send os commands and access GPIO pins
import RPi.GPIO as GPIO
import time
import os

GPIO.setmode(GPIO.BCM) # Set pin numbering to board numbering
GPIO.setup(27, GPIO.IN, pull_up_down=GPIO.PUD_UP) # Setup pin 27 as an input
while True: # Setup a while loop to wait for a button press
  if(GPIO.input(27)==0): # Setup an if loop to run a shutdown command when button press sensed
        os.system("sudo shutdown -h now") # Send shutdown command to os
        break
  time.sleep(1) # Allow a sleep time of 1 second to reduce CPU usage
```

You can easy check if your script is running just writing on the command line

sudo python /home/pi/upis/fshut.py

and then pressing the **Shutdown Button** for more than 2 seconds (until RED led will light on). If you have properly performed the above tasks, the computer should print on the screen the following message and shutdown then.

The system is going down for system halt NOW!

After the **Safe File Shutdown** you can restart your RaspberryPi® using the **Reset Functionality**. This is the proper way of restarting you computer when using the **UPiS** module, because removing the cable activated battery back-up, **without stopping the system can cause damage**. After shutdown of the RaspberryPi® it is recommended to switch off the computer using the hardware switch placed on the **UPiS** PCB. If the system is supplied from battery power back-up then it goes automatically to Low Power Mode. This functionality is also available for the Cable Powering in future Firmware Updates.

Therefore there are the following ways to switch of the RaspberryPi®:

1. Safe File Shutdown (without risk of corrupted files)
2. Switch Off the system with the integrated mechanical switch (with a risk of files corruption)

Setting-up the I2C interface and RTC

Valid for the UPiS BASIC and UPiS ADVANCED

The I^2C Ports on the RaspberryPi® are not enabled by default. Follow these steps to enable the I^2C port and then the RTC communicating trough **I^2C** with RaspberryPi®.

First it is needed to edit the config file that disables the **I^2C** port by default. This setting is stored in /etc/modprobe.d/raspi-blacklist.conf. Use nano to edit this but you can also use any other editor you are comfortable with.

$sudo nano /etc/modprobe.d/raspi-blacklist.conf

Once this file is open find this line *blacklist i2c-bcm2708* and comment it out by adding# to the front of it.

#blacklist spi-bcm2708

#blacklist i2c-bcm2708

Edit /etc/modules

$sudo nano /etc/modules

And add the following:

i2c-bcm2708
i2c-dev
rtc-ds1307

Add the modules to the kernel (they will automatically be added on subsequent boots from /etc/modules):

$sudo modprobe i2c-bcm2708
$sudo modprobe i2c-dev

$sudo modprobe rtc-ds1307

Reboot the system

$sudo reboot

Install **I²C** tools

$sudo apt-get install i2c-tools

Look for ID #68 with i2cdetect

On a 256MB Raspberry Pi Model A:

$sudo i2cdetect –y 0

On a 512MB Raspberry Pi Model B:

$sudo i2cdetect –y 1

The result should look like:

```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Figure 26 I²C UPiS Simulated DS1307 Clock detection

Then as roots do the following for model of RaspberryPi® you have

On a 256MB Raspberry Pi Model A:

$sudo bash
 # echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-0/new_device
 # exit

On a 512MB Raspberry Pi Model B:

$sudo bash
 # echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
 # exit

The result should look like:

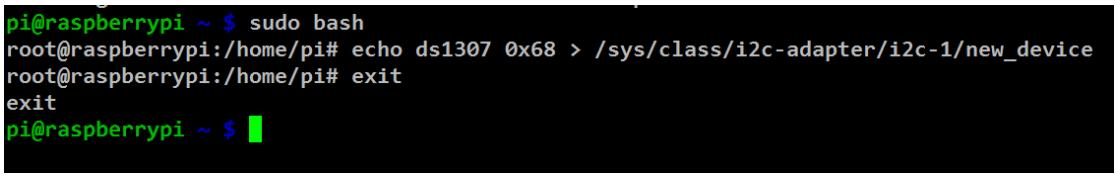

Figure 27 UPiS Simulated DS1307 Clock sudo bash commands execution

Then check for time from the clock (which will show Sat 01 Jan 2000 if it is the first time it is used):

$sudo hwclock -r

Then write the current system time to the clock:

$sudo hwclock -w

Then edit /etc/rc.local:

$sudo nano /etc/rc.local

and add the following before exit 0:

On a 256MB Raspberry Pi Model B:

echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-0/new_device
hwclock -s

On a 512MB Raspberry Pi Model B:

echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
hwclock -s

Superior System Operations

UPiS Firmware Upgrade Procedure

Valid for the UPiS BASIC and UPiS ADVANCED

In order to keep **UPiS** module firmware up-to-date an embedded serial bootloader has been implemented. Invoking it is done when the UPiS module starts from **UPiS** RESET when pressed at the same time as the **SHDN** button. This feature allows user to update the firmware trough the micro USB port placed on the **UPiS** module. In order to upload the new firmware to the **UPiS** dedicated bootloader software needs to be running on the PC. Current version of the boot loader supports only Windows® Operating System. For the UPiS Firmware Update Procedure will be needed:

1. Personal Computer with Windows® OS
2. Micro USB to USB cable
3. The bootloader program. You need to download the bootloader software file with a name UPiS_bootloader.zip from the following location:

   http://www.pimodules.com/downloads/UPiS_bootloader.zip

   Save to a known location on your PC and unzip it to a folder i.e. c:\UPiS\. Unzipping of this file will generate:
   i. UPiS_BL.exe
   ii. UPiS_Bootloader.bat

4. The latest Hex file of the firmware. You need to download the latest UPiS firmware file with a name UPiS_firmware_v1.00.zip from the following location:

   http://www.pimodules.com/downloads/UPiS_firmware_v1.00.zip

   Save to a known location on your PC and unzip it to a folder i.e. c:\UPiS\. Unzipping of this file will generate:
   i. UPiS_V1.00.hex
   ii. UPiS_V1.00_recent_changes.txt

5. Any Terminal program running on the PC (i.e. TearTerm, HyperTerminal or any other). It must be set to 38400 8N2, to make it connection with the **UPiS**
6. The **UPiS** module itself

The UPiS_BL.exe need to be call from the COMMAND PROMPT with the following parameters:

UPiS_BL PORT=COMX BAUD=38400 UPiS_V1.00.hex

Where:

PORT=COMX is the Virtual Com Port which has been recognized by the Windows® OS when UPiS is connected to the PC USB Port (i.e. COM1, COM2 etc).

BAUD=38400 is the data rate in which UPiS communicate with the system. Do not change it.

UPiS_V1.00.hex is the latest firmware update file

However, because calling this software needs to be done from the COMMAND PROMPT it can cause some trouble for non-experienced users. There is another way to use it, which is probably simpler than the first one. This is by running the **UPiS_Bootloader.bat**. It can be done from the graphic environment, just by clicking it. However, before doing it, it must be edited with a notepad, and adopted parameters about the COM port, and probably the .HEX file.

After downloading and adopting the software, the next step is to upload the latest firmware to the **UPiS** module. Here below is described step by step the firmware updating procedure:

1. Shutdown the RaspberryPi® if it is running using File Safe Shutdown Procedure
2. After that, switch OFF the powering of the RaspberryPi® with mechanical switch of the **UPiS**
3. It would also be a good idea to completely unplug the RaspberryPi®; however, it is not necessary, and depends of the Jumper setting you have already set. If you need to set Jumpers, it is best to unplug the RaspberryPi® from the **UPiS**. The most important thing is to completely isolate power to the RaspberryPi® from the **UPiS** module.
4. Set the proper Jumpers in the Serial Ports Connection Matrix
5. Selected connection should be: **UPiS** Serial Port routed to the **UPiS** micro USB Connector

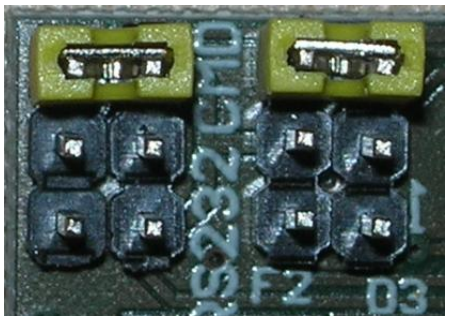

Figure 28 UPiS Bootloader Feature - Jumpers Settings

6. Connect the **UPiS** module through their micro USB socket to USB socket in your PC via cable
7. Be sure that the **UPiS** is connected through its micro USB to computer and powered via the same port

8. The UPiS micro USB should be recognized by the Windows® OS and Virtual Com Port should be assigned. If you doing this for the first time it could take a little bit longer because Windows® OS will have to automatically install the required driver.
9. Using Device Manager, check the Virtual Com Port number assigned to the **UPiS** USB connection i.e. COM29
10. Set the following parameters on the Virtual Com Port:

Bit per Second: 38400
Data Bits: 8
Parity: NONE
Stop Bits: 2
Flow Control: XON/XOFF

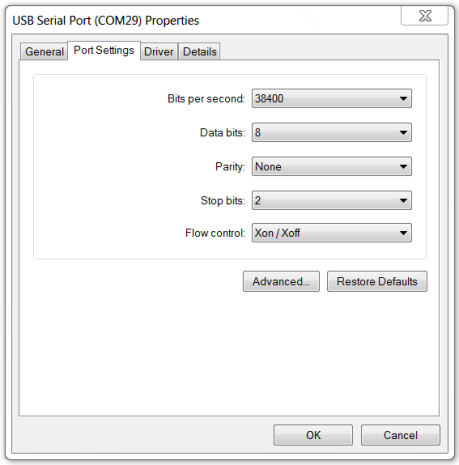

Figure 29 UPiS Bootloader Feature VCP settings

In the Advanced Tab, make the following changes

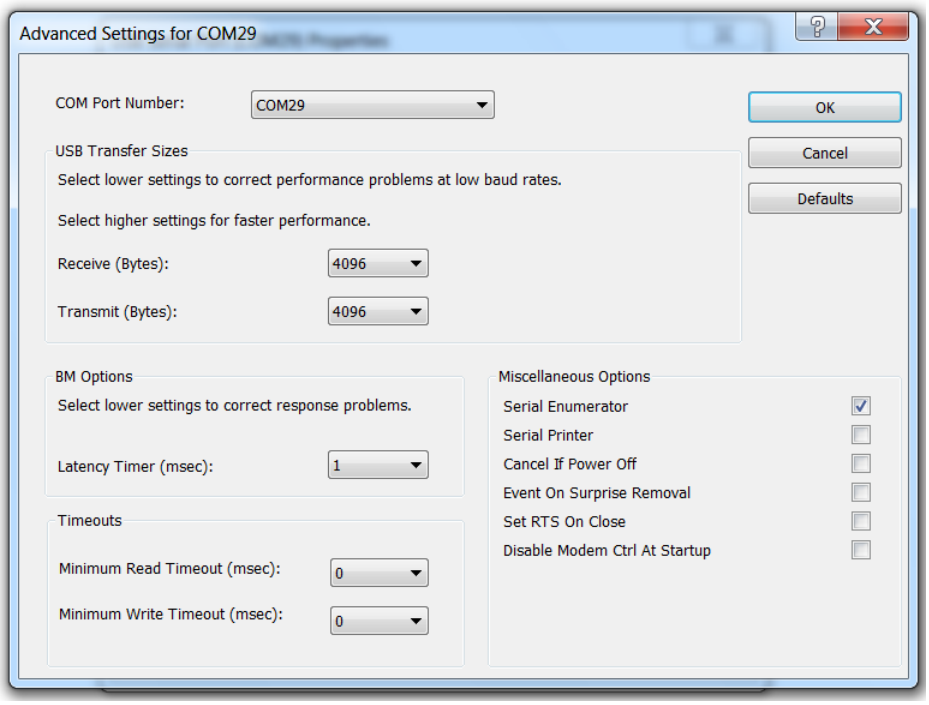

Figure 30 UPiS Bootloader Feature VCP Advanced Settings

Latency Timer: 1

11. You should know the Virtual Com Port in order to use it with the **UPiS_BL.exe** bootloader software.
12. To invoke the bootloading procedure on the **UPiS** press and hold the RST button, while holding the RST button, press and hold the SDWN button. With both buttons being pressed simultaneously, release the RST button, then release the SDWN button.You will then see all of the Green LEDs light, afterwards the (STB) RED Light will illuminate. Your **UPiS** is now in the **bootloading** mode and waiting for the hex file. This procedure can be easy done with one finger due to close placement of these two buttons.

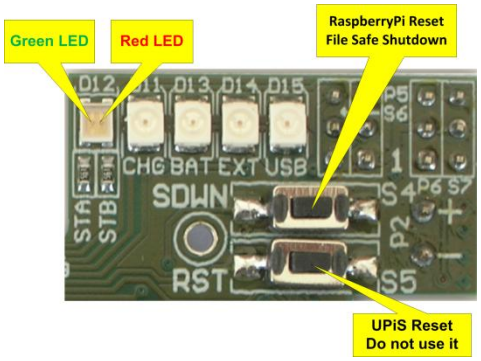

Figure 31 Top View Bootloader LEDs (Green and Red), SDWN and RST Buttons

13. Be sure that the Terminal program is not running because you will have a conflict with the Virtual Com Ports when **UPiS_BL.exe** will be called
14. Run the **UPiS_BL.exe** with required parameters from the COMMAND PROMPT or the modified **UPiS_Bootloader.bat**
15. When you start it, the **(STB) RED** led will be switched to **(STA) Green** and you will see the following picture on the screen.

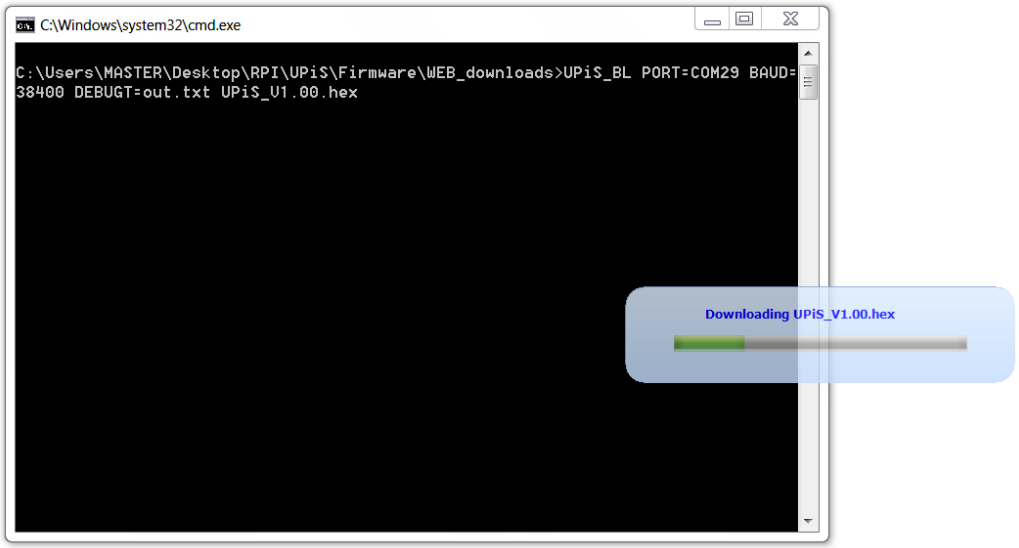

Figure 32 UPiS Bootloader Feature invoking the bootloader

16. The Firmware uploading procedure takes about 30-40 seconds, after that **UPiS** Restarts

17. **It is very important that after downloading the latest firmware update, when the UPiS is still connected to your PC, to run the Terminal Program, select the proper COM port, press the RST button, view themessages sent by UPiS to the Terminal and type the command @FACTORY to set the factory defaults**

18. The following picture will be visible on the terminal screen:

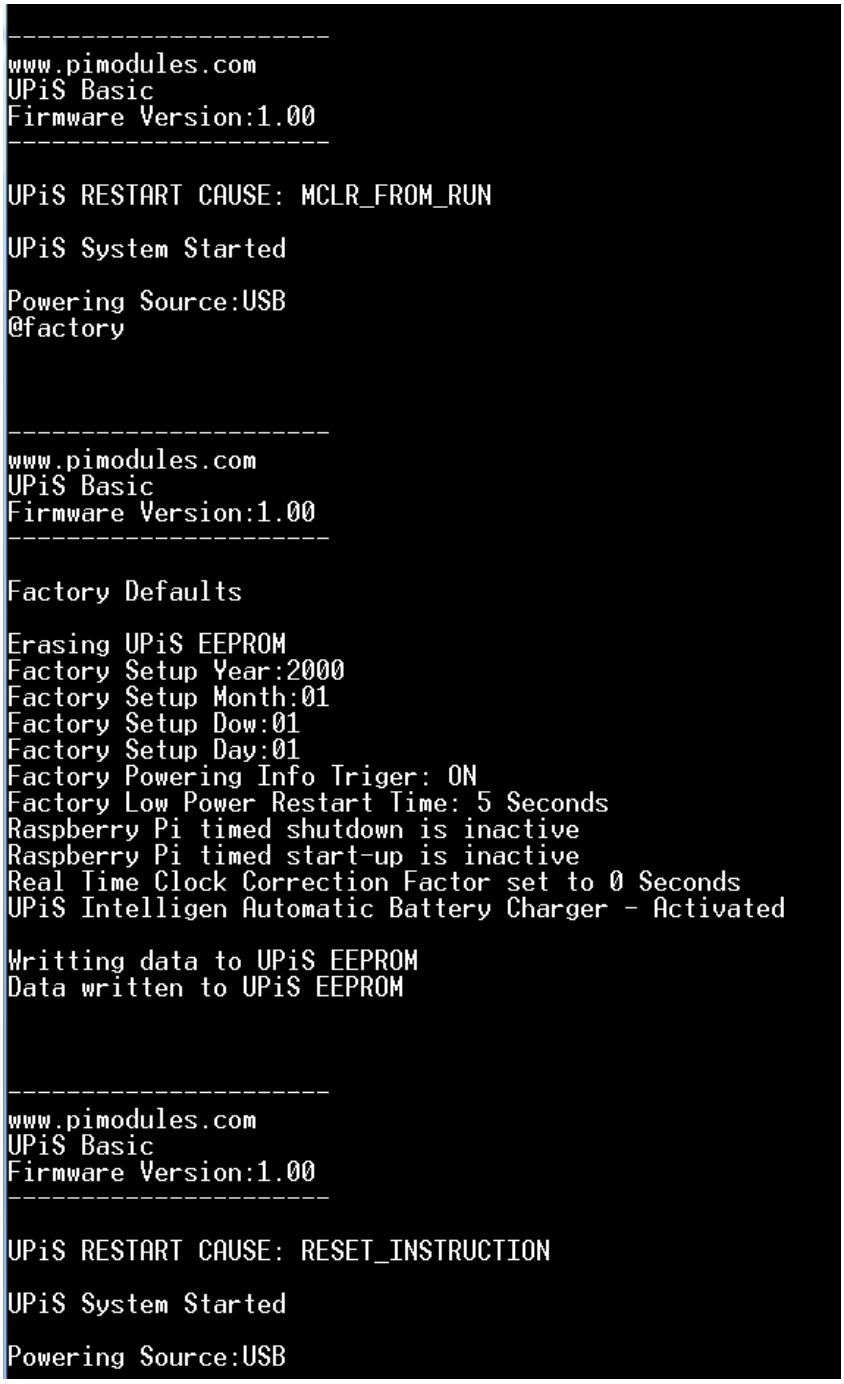

Figure 33 UPiS Bootloader Feature @factory command execution

There is also a good idea to run the command @status in order to see the **UPiS** Status

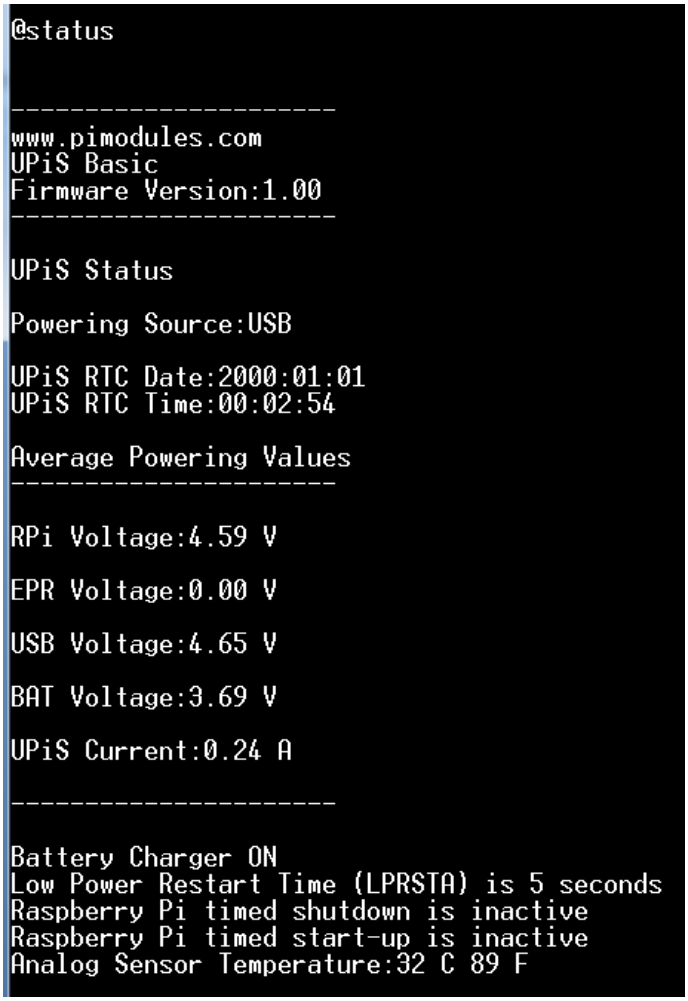

Figure 34 UPiS Bootloader Feature @status command execution

After that, the **UPiS** will be ready to use it with the RaspberryPi®.

There is only one firmware available for both types of **UPiS Basic** and **UPiS Advanced.** The firmware automatic recognizes which type of board is running and automatically selects the available futures.

Routing the Serial and USB Ports

Valid for the UPiS BASIC and UPiS ADVANCED

Both versions of **UPiS modules** are equipped with micro USB interface (for powering and for data transfer). A In addition, the **UPiS Module Advanced** is equipped with an RS232 Level Converter. These interfaces are designed to allow connectivity between various serial links of the **UPiS** module and RaspberryPi® pair. In order to allow thevarious types of connectivity, a Switching Matrix has been implemented. It is a set of jumpers that, with a selectable configuration, forces various connectivity schemes. These are:

1. **Routes RaspberryPi® RS232 to the UPiS Module micro USB Connector**
   Valid for the UPiS BASIC and UPiS ADVANCED

   This setting routes the RS232 RaspberryPi® RS232 to the UPiS Module micro USB connector, which permits viewing all I/O on the PC. The user can change access of this interface in the RaspberryPi® in order to use it on their applications, but it can be used also for Emergency Recovery of the RaspberryPi®. The proper settings of the jumpers are shown here below. The RS232 of the RaspberryPi® is set to the 115200 bps 8N1, so terminal program on the PC should be set to the same rate and other parameters.

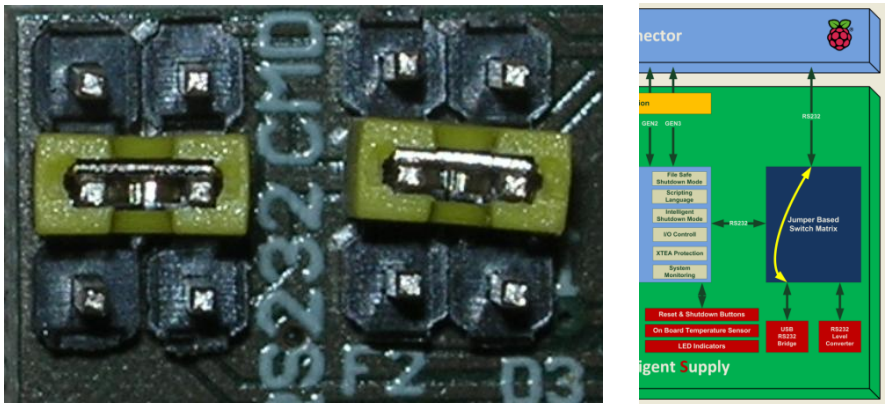

Figure 35 RaspberryPi® RS232 connected to the UPiS Module micro USB

   Disabling the Recovery use of the RaspberryPi® RS232 is explained in details in the next sections and should be done if the user plans to utilize this port for any other applications than the rescue procedure.

2. **Routes RaspberryPi® RS232 to RS232 UPiS Level Converter**
   Valid for the UPiS ADVANCED only

   Similar as in (1) for the RS232 of the RaspberryPi®, this can be routed to the **UPiS** RS232 level converter instead to the **UPiS** micro USB interface. Usage is similar to the feature which is described above. This functionality is available only in the **UPiS** Advanced.

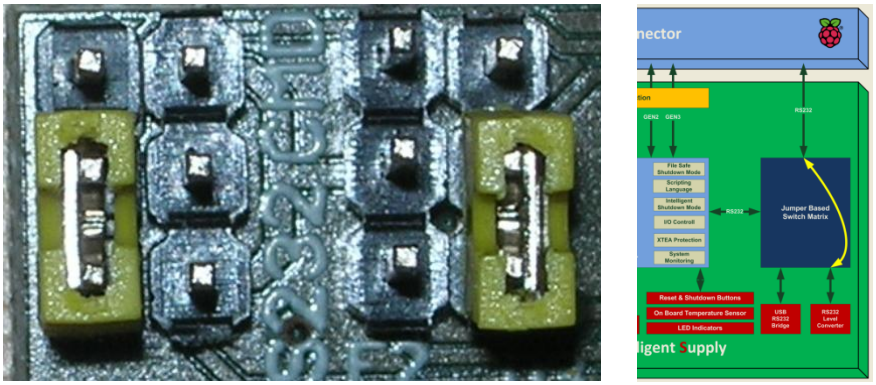

Figure 36 RaspberryPi® RS232 connected to RS232 UPiS Level Converter

3. Route RaspberryPi® RS232 to UPiS Serial Port

Valid for the UPiS BASIC and UPiS ADVANCED

The **UPiS** Advanced and **UPiS** Basic are equipped with their own serial port. This port is used to control the **UPiS** from a terminal program. There is a long list of commands available to control the **UPiS**. If the user likes to control the UPiS from their RaspberryPi® he/she needs to connect the **UPiS** serial port to the RaspberryPi® serial port. A detailed description on how to make this connection is provided in another partr of this manual. However, independent from the setting on the RaspberryPi®, there must also be a physical connection between the UPiS and RaspberryPi® which must be done by setting the proper jumpers. See picture below.

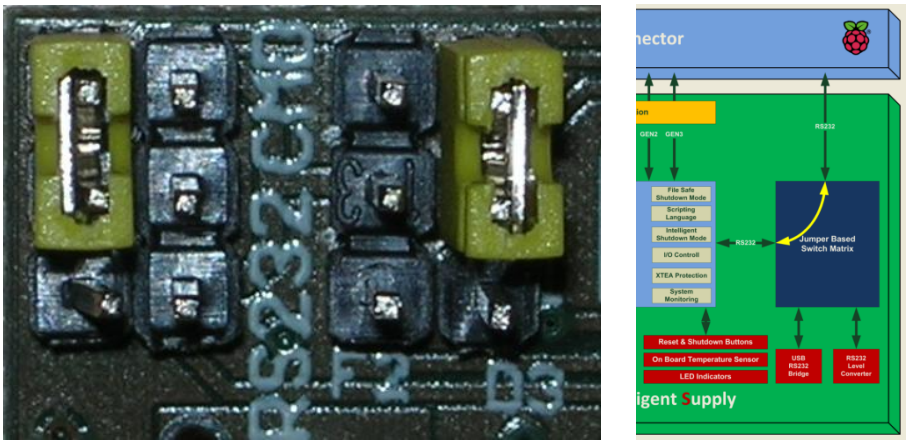

Figure 37 RaspberryPi® RS232 connected to UPiS Serial Port

4. UPiS Serial Port to the UPiS USB Connector

Valid for the UPiS BASIC and UPiS ADVANCED

This setting is similar to the (3); however, it routes the UPiS serial port to the UPiS micro USB. It is used for the firmware bootloading, for debugging purposes or just have a control over the UPiS via terminal running on PC.

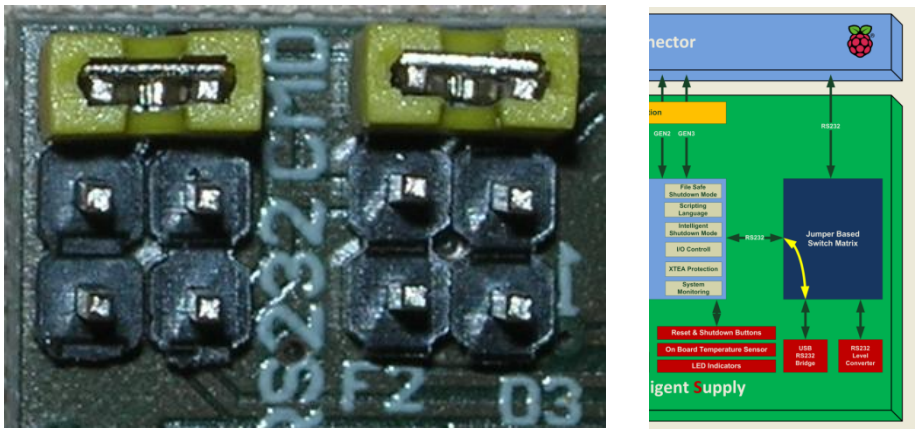

Figure 38 UPiS Serial Port connected to the UPiS USB Connector

5. **RS232 Level Converter Port to the UPiS USB Connector**

   Valid for the UPiS ADVANCED only

   The **UPiS** RS232 level converter and **UPiS** micro USB are completely independent hardware on the board of the UPiS. Therefore, it can be connected and used according to the jumper setting. Sometimes there is a need to have an independent USB to Serial converter; this, configuration of jumpers will give the user an independent USB to RS232 converter that can be utilized in the application built based on RaspberryPi®.

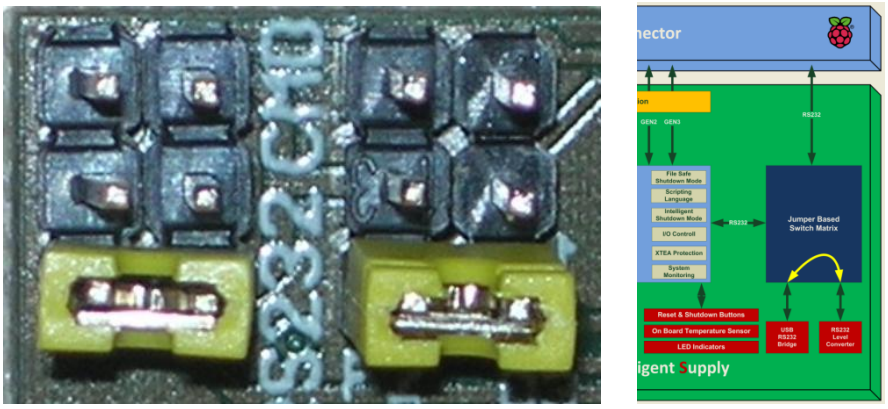

Figure 39 RS232 Level Converter Port connected to the UPiS USB Connector

UPiS Terminal Commands Control

Valid for the UPiS BASIC and UPiS ADVANCED

The **UPiS** is a Plug and Play device in its basic functionality. After attaching the battery and connecting it with the RaspberryPi® there is no need for any additional maintenance in order to use the majority of its features. However, there is a group of users that have more

advanced needs. The **T**erminal **C**ommands **C**ontrol was specifically addressed for them. The **TCC** give such a user plenty of additional features all in a single board. Some of them can be accessed directly from the RaspberryPi® Pins by using the jumper selection, but all of them can be accessed from a single access point using the RaspberryPi® RS232 port connected directly to the **UPiS** Serial Port. This can be done simply using a terminal program, or by using software written in any language (C, Python etc) which has an access to the RaspberryPi® Serial Port. All commands needed to make the necessary changes to the RaspberryPi® are listed below so that the serial port can be customized based on user application. Writing software to handle the serial port in various languages is not a part of this manual; however, all command presented here can be handled from the terminal program.

In addition, if the UPiS Serial Port is routed to the UPiS micro USB interface the functionality can be handled via the PC and associated Terminal Program (i.e. TeraTerm). If a terminal program is utilized, it is not necessary to make any modifications to the RaspberryPi® Serial Port.

Setting Up the RaspberryPi® Serial Port

Valid for the UPiS BASIC and UPiS ADVANCED

By default the RaspberryPi®'s serial port is configured to be used for console input/output. While this is useful if you want to login using the serial port, it means you can't use the Serial Port in your programs. To be able to use the serial port to connect and talk to other devices, the serial port console login needs to be disabled.

Needless to say you will need some other way to login to the RaspberryPi®, it is suggested doing this over the network using an SSH connection.

Disable Serial Port Login

To enable the serial port for your own use you need to disable login on the port. There are two files that need to be edited

The first and main one is:

/etc/inittab

This file has the command to enable the login prompt and this need to be disabled. Edit the file and move to the end of the file. You will see a line similar to:

T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

Disable it by adding a # character to the beginning. Save the file.

#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

Disable Bootup Info

When the RaspberryPi® boots up, all the bootup information is sent to the serial port. Disabling this bootup information is optional and you may want to leave this enabled as it is sometimes useful to see what is happening at bootup. If you have a device connected (i.e. Arduino) at bootup, it will receive this information over the serial port, so it is up to you to decide whether or not this is a problem.

You can disable it by editing the file:

/boot/cmdline.txt

The contents of the file look like this

dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait

Remove all references to ttyAMA0 (which is the name of the serial port). The file will now look like this:

dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait

Reboot

In order you enable the changes you have made, you will need to reboot the Raspberry Pi

sudo shutdown -r now

Test the Serial Port

A great way to test out the serial port is to use the minicom program. If you don't have this installed run

sudo apt-get install minicom

Connect your PC to the RaspberryPi® via Virtual Serial Port using an appropriate jumpers setting (**RaspberryPi® RS232 to the UPiS Module micro USB Connector)**,

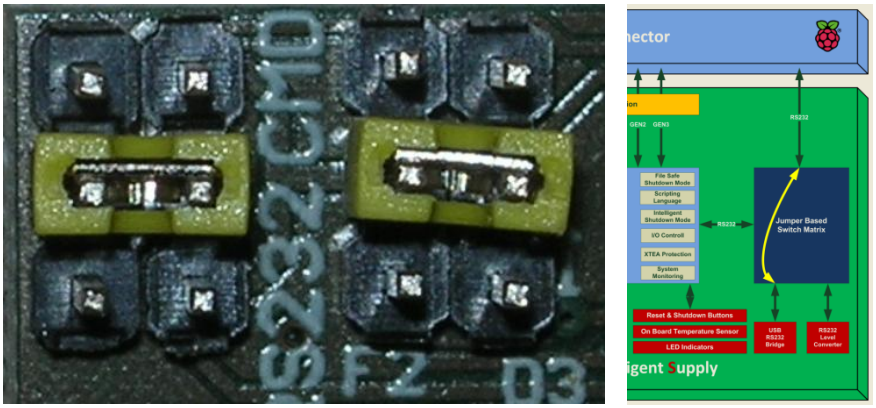

Figure 40 RaspberryPi® RS232 connected to the UPiS Module micro USB

then open TeraTerm or a similar serial terminal program on PC side. Setup a connection using the serial port at 38400 baud.

Now run up minicom on the Raspberry Pi using

minicom -b 38400 -o -D /dev/ttyAMA0

What you type into the minicom terminal screen should appear on the serial PC terminal and vice versa.

UPiS Terminal Commands Set

Valid for the UPiS BASIC and UPiS ADVANCED

In order to simplify the **UPiS TCC** string parsing each command send to **UPiS** ALWAYS start with the **@ sign**. Listed below is a detailed explanation of the commands implemented until now. Please kindly notice that we are working on more commands (functionalities) that will be released soon with the next firmware releases.

COMMAND: @FACTORY

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Factory Defaults.

Usage: **@factory** or **@FACTORY**

Results/Explanation: Return UPiS to Factory Default Values. Should be used always after upload of a new firmware.

COMMAND: @VERSION

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Print the Version

Usage: @version or **@VERSION**

Results/Explanation: Return **UPiS** Hardware and Firmware Version

COMMAND: @STATUS

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Print the UPiS Status

Usage: @status or **@STATUS**

Results/Explanation: Return **UPiS** detailed status that contains all needed parameters of the system like: powering source, Voltages, Current, and Analog Temperature etc.

COMMAND: @RON

Valid for the UPiS ADVANCED only

Meaning: Normally Open Relay switch ON

Usage: @ron or **@RON**

Results/Explanation: Switch the NO Relay ON

COMMAND: @ROFF

Valid for the UPiS ADVANCED only

Meaning: Normally Open Relay switch OFF

Usage: @roff or **@ROFF**

Results/Explanation: Switch the NO Relay OFF

COMMAND: @ANTMPC

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Print the UPiS Analog Thermometer Temperature

Usage: @antmpc or **@ANTMPC**

Results/Explanation: Return **UPiS** Analog Thermometer Temperature in Celsius

COMMAND: @ANTMPF

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Print the UPiS Analog Thermometer Temperature

Usage: **@antmpf** or **@ANTMPF**

Results/Explanation: Return **UPiS** Analog Thermometer Temperature in Fahrenheit

COMMAND: @PM

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Powering Mode

Usage: **@pm** or **@PM**

Results/Explanation: Return **UPiS** actual Powering Mode. There are:

- **USB** - **USB** Cable Powering
- **EPR** - **E**xternal Cable **P**owe**R**ing
- **BAT** - **Bat**tery Powering
- **LPR** - Battery **L**ow **P**ower**i**ng
- **RPI** - **RPi** Powering

COMMAND: @ CHGR ON/OFF

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Switch Automatic Intelligent Charger OFF or ON

Usage: **@chgr on** or **@CHGR ON**

Usage **@chgr off** or **@CHGR ON**

Results/Explanation: Switches the Automatic Intelligent LiPO battery Charger ON or OFF. It is necessary sometimes when you are supplying the RaspberryPi® from a power source with reduced current availability like user PC USB interface. Under normal conditions UPiS check all the time system current consumption and when it is bigger than 750 mA, then it is switching OFF the charger, but sometimes the available current is not enough and is below it, therefore user need to close the battery charging completely. The UPS feature will be still active even if battery is not charged.

COMMAND: @RPI

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Voltage on the RaspberryPi® 5V P1 connector

Usage: **@rpi** or **@RPI**

Results/Explanation: Return the actual exact value of the RaspberryPi® 5V voltage P1 connector

COMMAND: @EPR

Valid for the UPiS ADVANCED only

Meaning: Voltage on the External Extended Powering connector (7 V DC – 18 V DC)

Usage: **@epr** or **@EPR**

Results: Return the actual exact value of the Voltage on the External Extended Powering connector (7 V DC – 18 V DC)

COMMAND: @USB

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Voltage on the UPiS micro USB connector

Usage: **@usb** or **@USB**

Results/Explanation: Return the actual exact value of the voltage on the UPiS micro USB connector

COMMAND: @BAT

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Voltage on the UPiS Battery

Usage: **@bat** or **@BAT**

Results/Explanation: Return the actual exact value of the voltage on the UPiS Battery

COMMAND: @CUR

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: System Current Consumption

Usage: **@cur** or **@CUR**

Results/Explanation: Return the actual exact value of the UPiS and RaspberryPi® system current consumption

Accuracy - 10mA

The measurement contains:

- The RaspberryPi® current consumption

- Battery Charging current consumption
- Relay ON current consumption
- Microcontroller current consumption
- RS232 Level Converter current consumption
- Serial Bridge current consumption
- UPiS LEDs current consumption
- Various Components of the UPiS current consumption

There is a working version that will be released soon with a new firmware, showing separated current consumption of the RaspberryPi® and UPiS

COMMAND: @1WIREC

Valid for the UPiS ADVANCED only

Meaning: Print the 1wire sensor Thermometer Temperature

Usage: @1wirec or **@1WIREC**

Results/Explanation: Return **1wire sensor** Thermometer Temperature in Celsius with accuracy of 0.1 degree in Celsius

COMMAND: @1WIREF

Valid for the UPiS ADVANCED only

Meaning: Print the 1-wire® sensor Thermometer Temperature

Usage: @1wiref or **@1WIREF**

Results/Explanation: Return **1-wire® sensor** Thermometer Temperature in Celsius with accuracy of 0.1 degree in Fahrenheit

COMMAND: @ PWRINFO ON/OFF

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Set ON/OFF the Automatic Trigger when Powering Source is changing

Usage: @pwrinfo on or **@PWRINFO ON**

Usage: @ pwrinfo off or **@ PWRINFO OFF**

Results/Explanation: If set to ON, every time the powering source is changed it will send a message. It is very useful, espcially for remote applications, to know when the powering source has automatically been changed.

COMMAND: @ LPRSTA

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Low Powering Restart Sampling Time

Usage: @lprsta <number 1-99> or **@LPRSTA <number 1-99>**

Usage: @lprsta or **@LPRSTA**

Usage: @lprsta ? or **@LPRSTA ?**

Results/Explanation: Return or set the Low Power Return Time.

When System of UPiS and RaspberryPi® are supplied from battery, the user can shut it down using the **File Safe Shutdown Procedure**, or just switch it off using the mechanical switch. After a couple of seconds, the UPiS will enter the Low Powering Mode (sleeping mode) where current consumption is very small. Usually the UPiS current consumption during the LPR mode is about 60 – 70 uA and the only running peripheral during that time is the RTC. Recovering from this mode is possible in three ways:

- By pressing the SHDN button
- By switching the mechanical switch or
- By entering Cable Power

However, because the UPiS is in very Low Powering Mode, most of the peripherals are not working as they are sleeping and not consuming current. Therefore, the UPiS needs to start up periodically, supply the peripherals and take some measurements in order to see what is going on around it (i.e. if Cable Power has been entered, RaspberryPi® switched ON and consuming current etc.). These checks of peripherals take 9 mS, and consume 300 uA. How often this is done is determined by the value for the LPRSTA time. The default value is 5 seconds, which means that every 5 second the UPiS starts-up for 9 mS, checks their peripherals and if nothing happens, it goes to sleep again. If restart ever happens, based on the required actions listed above, it exits LPR mode and starts running as before. Because during start-up consumption of current is much higher, the user can adjust how often this happens in order to save battery (energy).

Examples:

@LPRSTA 10 (means that even if put the Cable to USB it will be recognized after 10 seconds, and then the system starts-up)

COMMAND: @TIME

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Shows the current UPiS RTC TIME

Usage: @time or **@TIME**

Results/Explanation: Return the TIME of the UPiS RTC

COMMAND: @DATE

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Shows the current UPiS RTC DATE

Usage: **@date** or **@DATE**

Results/Explanation: Return the DATE of the UPiS RTC

COMMAND: @START

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Startup RaspberryPi® after X minutes

Usage: **@start** or **@START**

Disable the START timer procedure (reset the START timer to 0)

Usage: **@start ?** or **@START ?**

Show the set value of the START timer

Usage: **@start !** or **@START !**

Shows the current value of the running START timer. This timer decreases until reaching a value of 0, then startup of the RaspberryPi® occurs

Usage: **@start <number>** or **@START <number>**

Set the current value of the START timer. The <number> must be between 2 and 1000 minutes

Results/Explanation:

A detailed explanation is listed below with the @STOP command

COMMAND: @STOP

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Shutdown RaspberryPi® after X minutes

Usage: **@stop** or **@STOP**

Disable the STOP timer procedure (reset the STOP timer to 0)

Usage: @stop ? or @STOP ?

Shows the set value of the STOP timer

Usage: @stop ! or @STOP !

Shows the current value of the running STOP timer. This timer decreases until it reaches a value of 0, then shutdown, using File Safe Shutdown Procedure, occurs for the RaspberryPi®

Usage: @stop <number> or @STOP <number>

Set the current value of the STOP timer. The <number> must be between 2 and 1000 minutes

Results/Explanation:

There are many applications, especially when running the RaspberryPi® on battery powered supply, when the user needs to start it up for a short time in order to do some task, and then, after some time interval, needs to shut it down again. This approach is ideal when power is not enough, therefore stopping and running the RaspberryPi® save a lot of energy.

The START and STOP command has two predefined timers which count running and sleeping time. Each timer starts when the other is stopped. In order to have this system running both the START and STOP values should be set. The UPiS always executes the STOP command first, and after X amount of minutes, initiates the START command. Both values are stored in the UPiS EEPROM therefore it will run as long as the system is connected to the RaspberryPi®.

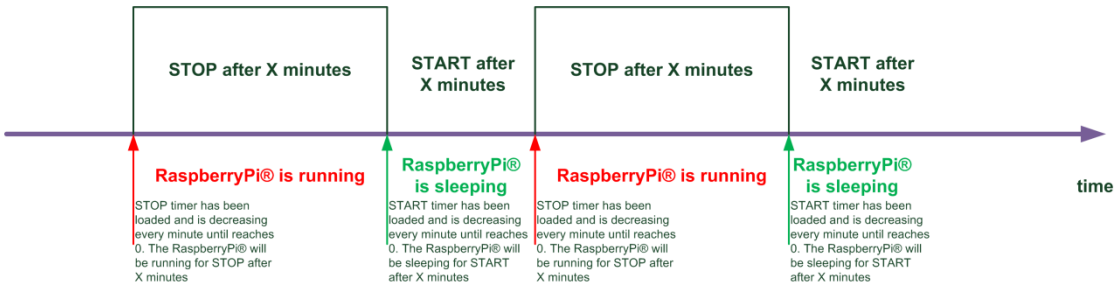

Figure 41 @STOP and @START commands relation

Examples:

@START 5 (means sleep for 5 minutes and then startup)

@STOP 3 (means run for 3 minutes and then shutdown)

This means that the system will shut down after 3 minutes, then will sleep for 5 minutes, then start up and run for 3 minutes, then shutdown and sleep for 5 minutes etc.

COMMAND: @SDWN

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Shutdown the System (UPiS and RaspberryPi®)

Usage: @sdwn or **@SDWN**

Starts the File Safe Shutdown procedure

Results/Explanation:

Shutdown the System (UPiS and RaspberryPi®) by activating the **File Safe Shutdown** procedure, after 40 seconds it cuts the power to the RaspberryPi® and UPiS goes in to sleep mode. There is no need to switch off the RaspberryPi® by mechanical Switch. Recovery from this state is done by pressing the **SDWN** button for a short time or by pressing the **RESET** button (CAUTION: if the r. In order to have proper effects of this command it is necessary to do all describe above step related to **File Safe Shutdown** procedure (proper jumper setting and script running on the RaspberryPi®). RTC is still running during the sleep time.

COMMAND: @RTCCF

Valid for the UPiS BASIC and UPiS ADVANCED

Meaning: Real **T**ime **C**lock **C**orrection **F**actor

Usage: @rtccf or **@RTCCF**

Disable the RTCCF correction procedure (reset the RTCCF counter to 0)

Usage: @rtccf ? or **@RTCCF ?**

Show the set value of the RTCCF counter

Usage: @rtccf ! or **@RTCCF !**

Show the current value of the running RTCCF counter. This counter is decreasing until reaches value of 0 and makes correction.

Usage: @rtccf <number> - or **@RTCCF <number> -**

Set the current value of the RTCCF counter to decrease of 1 second the RTC after <number> seconds. The <number> must be from 0 up to 4294967295 (int32)

Usage: @rtccf <number> + or **@RTCCF <number> +**

Set the current value of the RTCCF counter to increase of 1 second the RTC after <number> seconds. The <number> must be from 0 up to 4294967295 (int32)

Results/Explanation:

The implemented RTC as all other RTCs have a small difference in crystal frequency. This small difference produces a small drift between **UPiS** RTC and other RTCs (i.e. internet time). Cumulating this drift within a long time period can produce a time time drift, typically a couple of seconds, between the **UPiS** RTC and other time sources. This drift is always in the same direction, therefore a small correction can be added, or subtracted, in order to correct it. The **RTCCF** command adds or subtracts this correction every predefined number of seconds. The user will need to measure this drift across a day, or week, time period, calculate the amount of time it took to produce a 1 second drift and add it or subtract it to the RTCCF counter. One Day is 86400 seconds, one Week is 604800 seconds and one Month is 2419200 seconds. Each command used is confirmed by a **UPiS** reply with a detailed description of what has been done.

Examples: **"RTCCF 10000 – "means subtract every 10000 seconds, one second from the RTC**

"RTCCF 10000 + "means add every 10000 seconds, one second to the RTC

Application Examples

TBC

APPENDIX A Default Values

TBC

APPENDIX B Table of LED Indications

Indicators	

LED **CHG** (GREEN)	• LIGHT when charger is working and battery is charged
	• DARK when charger is not working:
	  ○ Battery is fully charged
	  ○ Battery charger is switched OFF (by the integrated microcontroller)
	• FAST BLINKING
	  ○ Battery is not connected
	  ○ Battery is fault

LED **BAT** (GREEN)	• LIGHT when Boost Switching Converter is working (with load or not) • DARK when Boost Switching Converter is working • LIGHT for a short time (9 mS) when system check RaspberryPi® Powering Status in Low Powering Mode
LED **EXT** (GREEN)	• LIGHT when External Extended Powering Cable is connected and system powered from it • DARK when External Extended Powering Cable is not connected
LED **USB** (GREEN)	• LIGHT when UPiS micro USB Powering Cable is connected – it doesn't mean necessarily that system is powering from it • DARK when UPiS micro USB Powering Cable is not connected
Dual Color LED **STA** and **STB**	**STB** (RED LED): • DARK if nothing important is happening • Flashes 10 times every 100 mS when measuring for the first time the mean powering values (Voltages, Current and Temperature). It takes 2 seconds. • LIGHTs for 800 mS every 1 s within 40 s time frame when system is in File Safe Shutdown Procedure • LIGHTs for 100 mS every 1000 mS when UPiS is running on battery backup power and battery level is higher than 3.4V and lower than 3.6V • LIGHTs for 100 mS every 500 mS when UPiS Advanced is running on battery backup power and battery level is lower than 3.4V and higher than 3.2V Note: When Battery level is lower than 3.2 V, the system automatically initiates the File Safe RaspberryPi® Shut Down Procedure in order to save power for the RTC. The cut off power of the LiPO Battery is 3V. **STA** (GREEN LED): • Flashes 10 times every 20 mS when power source has been changed (after 3.6 seconds of cable connection or disconnection)

	• LIGHTs for 100 mS every 500 mS when cable power is connected • LIGHTs for 100 mS every 2000 mS when UPiS is running on battery backup power and battery level is higher than 3.6V (typical running conditions on battery) • DARK if system is in Low Power Mode

APPENDIX C Technical Specifications

Uninterruptible Power intelligent Supply Advanced Technical Specifications Firmware Version 1.00	
RaspberryPi® Interface	
P1 Connector Plug-in	Type: Top End Stack
RaspberryPi® ON/OFF	Hardware ON/OFF switch cutting 5V supply to the RaspberryPi®
P1 Connector PINs used	Standard: • GND: (P1 Connector PINs used 6, 9, 14, 20, 25) • 5V: (P1 Connector PINs used 1, 4)
	Optional Selectable by jumpers: • SDA0: (P1 Connector PIN used: 3) • SCL0: (P1 Connector PIN used: 5) • TXD0: (P1 Connector PIN used: 8) • RXD0: (P1 Connector PIN used: 10) • GPIO_GEN3 as 1-wire/IO pin: (P1 Connector PIN used: 15) • GPIO_GEN2 as SHTDWN pin: (P1 Connector PIN used: 13) • GPIO_GEN0 as RELAY CTRL: (P1 Connector PIN used: 11)
Integrated Battery	
Type	LiPO
Capacity	2600 mAh
Nominal Voltage	3.7 V
Integrated protection	Over Charge/Over Discharge PCB
UPiS on Bard Protection	• Cut-Off Jumper • PTC Resettable Fuse 2.6 A • Analog Thermometer • Microcontroller Supervised Charger with Cut-Off pre-conditions • Continuously current and voltage monitoring
Battery Life	400 Full charge/discharge cycles
Working temperature	-10 up to +60 Celsius degrees
Weight	
Size	
Integrated Automatic Intelligent LiPO Charger	
Charging Modes	• Full Charging Cycle • Trickle Charging
Charging Current	Continuous current 212 mA, Voltage 4.2 V
Charging Protection	• Thermal • Voltage • Current (stop charging if total current of the system exceeds 750 mA)
Cable Powering Inputs	
Extended External Powering	• Input Voltages: ○ 7V DC – 18 VDC

	• Protection:
	  ○ Inverse Polarity
	  ○ PTC Resettable Fuse 1.1 A
	  ○ Transient Voltage Suppression
	• Conversion Type to 5 VDC:
	  ○ Switching Buck type Converter at 1.6 MHz
	• Efficiency:
	  ○ More than 85% at 12V in and 850 mA out

UPiS micro USB	• Input Voltages:
	  ○ 5V DC (within USB Powering Specifications)
	• Protection:
	  ○ Mechanical Inverse Polarity
	  ○ PTC Resettable Fuse 1.25 A
	  ○ Transient Voltage Suppression

P1 Connector (optional) Then system is powered through the RaspberryPi® micro USB	• Input Voltages:
	  ○ 5V DC (within RaspberryPi® powering specifications)
	• Protection:
	  ○ None (however supply is fully protected by the RaspberryPi® supply system
	
	NOTE: Will be available with the new version of Firmware V1.02 update. However, a big part of the UPiS functionality does not work with this source of powering. It is recommended ONLY when it is not possible (due to mechanical cabling) to remove the USB cable powering the Pi.

Cable Powering Source Recognition	Automatic, with priority to External Extended Powering
Cable Powering Outputs	

User Protected 5V	• Protection:
	  ○ PTC Resettable Fuse 140 mA
	• Maximum current:
	  ○ 140 mA

User non-protected 3.3V (Suggested usage is to supply 1-wire interface)	• Protection:
	  ○ IC by itself
	• Maximum current:
	  ○ 100 mA
	NOTE: This is independent and separate from the RaspberryPi® powering system and used in the UPiS to supply RS232 Level Converter Interface. It can also be used for the suggested 1-wire Interface only.

UPS (Power Battery Backup) Functionalities	
UPS Type	Off-line with extremely fast switching time and intermediate power preservation
Power Loss Monitoring	On P1 connector 5V PINs every 120 uS
Power Loss Battery Backup Activation Condition	• Failing detection within 360 us (3 samples) • Voltage level less that 4.3V (programmable) at any sample
Power Battery Backup Activation Time	Less than 14 uS within a 120 uS window (usual 3uS)

Power Return Conditions	3.6 s after power return and stabilization on cable powering inputs

Battery Backup Electrical Specifications	• 5V DC on P1 connector 5 V PINs
	• Total Supplied Current by the Battery System at 5 V:
	  ○ Guaranteed 850 mA
	  ○ Tested 1000 mA

Switching Converter Efficiency	More than 85% at 3.7 V input and 850 mA out
Low Powering Mode Current Consumption (all devices are switched off and system is disconnected from the Raspberry Pi®)	• 70 uA with RTC running • When checking if RaspberryPi® has been switched ON, 300 uA for a period of 9 mS. Repetitive checking, which is programmable by the user, defaults to 5 s
RaspberryPi® Estimated Running Time with typical load of 350 mA	About 5 hours with full battery charge
Interfaces	
RS232 (12V) Level Converter	Connectivity to: • RaspberryPi® Serial Port • UPiS micro USB Port
UPiS micro USB	Connectivity to: • RaspberryPi® Serial Port • UPiS Serial Port • UPiS Serial Port Data Rate 38400 bps 8N2
UPiS Serial Port	Connectivity to: • RaspberryPi® Serial Port • UPiS micro USB • UPiS Serial Port Data Rate 38400 bps 8N2
ESD Protected 1-wire (pull-up to 3.3 V)	Selectable by jumper: • UPiS (read by the RaspberryPi® on its Serial Port) • Directly connection to RaspberryPi® GPIO_GEN3
ESD Protected i-button (pull-up to 3.3 V) (available from firmware version 1.02)	Selectable by jumper: • UPiS (read by the RaspberryPi® on its Serial Port) • Directly connection to RaspberryPi® GPIO_GEN3
ESD Protected I/O Pin (pull-up to 3.3 V)	Selectable by jumper: • UPiS (read by the RaspberryPi® on its Serial Port) • Directly connection to RaspberryPi® GPIO_GEN3
NO RELAY (only for low voltages application)	Selectable by jumper: • UPiS (read by the RaspberryPi® on its Serial Port) • Directly connection to RaspberryPi® GPIO_GEN3 Electrical Guaranteed Load: • 1A 24VDC
Buttons	
UPiS **RST**	Used for: • UPiS System Reset (restarts UPiS, RTC and RaspberryPi®) • Activation of Bootloader functionality
UPiS **SDWN**	Used for: • Activation of Bootloader functionality

	• Restarts RaspberryPi® without resetting UPiS RTC • File Safe RspberryPi® Shutdown Procedure
Indicators	

LED **CHG** (GREEN)	• LIGHT when charger is working and battery is charged
	• DARK when charger is not working:
	  ○ Battery is fully charged
	  ○ Battery charger is switched OFF (by the integrated microcontroller)
	• FAST BLINKING
	  ○ Battery is not connected
	  ○ Battery is fault

LED **BAT** (GREEN)	• LIGHT when Boost Switching Converter is working (with load or not) • DARK when Boost Switching Converter is working • LIGHT for a short time (9 mS) when system check RaspberryPi® Powering Status in Low Powering Mode
LED **EXT** (GREEN)	• LIGHT when External Extended Powering Cable is connected and system powered from it • DARK when External Extended Powering Cable is not connected
LED **USB** (GREEN)	• LIGHT when UPiS micro USB Powering Cable is connected – it doesn't mean necessarily that system is powering from it • DARK when UPiS micro USB Powering Cable is not connected
Dual Color LED **STA** and **STB**	**STB** (RED LED): • DARK if nothing important is happening • Flashes 10 times every 100 mS when measuring for the first time the mean powering values (Voltages, Current and Temperature). It takes 2 seconds. • LIGHTs for 800 mS every 1 s within 40 s time frame when system is in File Safe Shutdown Procedure • LIGHTs for 100 mS every 1000 mS when UPiS is running on battery backup power and battery level is higher than 3.4V and lower than 3.6V • LIGHTs for 100 mS every 500 mS when UPiS Advanced is running on battery backup power and battery level is lower than 3.4V and higher than 3.2V Note: When Battery level is lower than 3.2 V, the system automatically initiates the File Safe RaspberryPi® Shut Down Procedure in order to save power for the RTC. The cut off power of the LiPO Battery is 3V. **STA** (GREEN LED): • Flashes 10 times every 20 mS when power source has been changed (after 3.6 seconds of cable connection or disconnection) • LIGHTs for 100 mS every 500 mS when cable power is

	connected • LIGHTs for 100 mS every 2000 mS when UPiS is running on battery backup power and battery level is higher than 3.6V (typical running conditions on battery) • DARK if system is in Low Power Mode
Sensors	
Analog Temperature Sensor	Accuracy - 1 Degree Celsius Accessible (readable) only via Serial UPiS interface via commands @STATUS or @ANTMPC or @ANTMPF When temperature is higher than 65 degrees Celsius or lower than -10 degrees Celsius then system automatically initiates the File Safe RaspberryPi® Shut Down Procedure.
System Current Sensor	Real Time System Current measure Accuracy - 10mA The measurement contains: • The RaspberryPi® current consumption • Battery Charging current consumption • Relay ON current consumption • Microcontroller current consumption • RS232 Level Converter current consumption • Serial Bridge current consumption • UPiS LEDs current consumption • Various Components of the UPiS current consumption Note: Some measurements will be available with the Firmware update 1.02 as they are undergoing testing.
Battery Voltage Sensor	Real Time System Current measure Accuracy - 10 mV
External Extended Powering Voltage Sensor	Real Time System Current measure Accuracy - 10 mV
micro USB Powering Voltage Sensor	Real Time System Current measure Accuracy - 10 mV
RaspberryPi® P1 5V Powering Voltage Sensor	Real Time System Current measure Accuracy - 10 mV
Additional Features	
Emulated RTC (DS1307) with drift correction system	Emulated RTC based on the crystal of 32768 KHz. Communicates with RaspberryPi® via I2C and emulates the DS1307. It will be provided on the Firmware update 1.01.
Timed Start Up/Shutdown	Programmed on RTC. Timer starts up and shutdown based on 1 minute intervals. • Maximum programmed time is 1000 minutes • Minimum programmed time is 2 minutes
XTEA Encryption System	XTEA Cryptographic system with user keys. Programmable providing Cryptographic protection of professional software written for the RaspberryPi®. It will be provided on the Firmware update 1.02.
Scripting Commands	23 commands are currently implemented. These will be improved and added to according to requests from users and system developments. Each command starts always with '@':

	@FACTORY	Factory Defaults
	@VERSION	Hardware and Firmware Version
	@STATUS	System Status
	@RON	Relay ON
	@ROFF	Relay ON
	@ANTMPC	Analog Temperature Celsius
	@ANTMPF	Analog Temperature Fahrenheit
	@PM	Powering Mode
	@CHGR ON/OFF	Charger ON or OFF
	@RPI	RaspberryPi® P1 5V Voltage level
	@EPR	External Extended Voltage level
	@USB	UPiS USB Voltage level
	@BAT	UPiS Battery Voltage level
	@CUR	System Current Consumption
	@1WIREC	1-wire Temperature Celsius
	@1WIREF	1-wire Temperature Fahrenheit
	@PWRINFO ON/OFF	Powering change source trigger
	@LPRSTA	Low Powering Restart Time
	@TIME	Time
	@DATE	Date
	@START	Timed RaspberryPi® Startup
	@STOP	Timed RTC Correction Factor
	@RTCCF	RaspberryPi® Shutdown

Uninterruptible Power intelligent Supply Basic Technical Specifications Firmware Version 1.00	
RaspberryPi® Interface	
P1 Connector Plug-in	Type: Top End Stack
RaspberryPi® ON/OFF	Hardware ON/OFF switch cutting 5V supply to the RaspberryPi®
P1 Connector PINs used	Standard: • GND: (P1 Connector PINs used 6, 9, 14, 20, 25) • 5V: (P1 Connector PINs used 1, 4)
	Optional Selectable by jumpers: • SDA0: (P1 Connector PIN used: 3) • SCL0: (P1 Connector PIN used: 5) • TXD0: (P1 Connector PIN used: 8) • RXD0: (P1 Connector PIN used: 10) • GPIO_GEN2 as SHTDWN pin: (P1 Connector PIN used: 13)
Integrated Battery	
Type	LiPO
Capacity	1150 mAh
Nominal Voltage	3.7 V
Integrated protection	Over Charge/Over Discharge PCB
UPiS on Bard Protection	• Cut-Off Jumper • PTC Resettable Fuse 2.6 A • Analog Thermometer • Microcontroller Supervised Charger with Cut-Off pre-conditions • Continuously current and voltage monitoring
Battery Life	400 Full charge/discharge cycles
Working temperature	-10 up to +60 Celsius degrees
Weight	
Size	
Integrated Automatic Intelligent LiPO Charger	
Charging Modes	• Full Charging Cycle • Trickle Charging
Charging Current	Continuous current 212 mA, Voltage 4.2 V
Charging Protection	• Thermal • Voltage • Current (stop charging if total current of the system exceed 750 mA)
Cable Powering Inputs	
UPiS micro USB	• Input Voltages: ○ 5V DC (within USB Powering Specifications) • Protection: ○ Mechanical Inverse Polarity ○ PTC Resettable Fuse 1.25 A ○ Transient Voltage Suppression
P1 Connector (optional) Then system is powered	• Input Voltages: ○ 5V DC (within RaspberryPi® Powering

trough the RaspberryPi® micro USB	Specifications) • Protection: ○ NONE (however supply is fully protected by the RaspberryPi® supply system NOTE: Will be available with the new version of Firmware V1.02 update. However, a big part of the UPiS functionality does not work with this source of powering. It is recommended ONLY when it is not possible (due to mechanical cabling) to remove the USB cable powering the Pi.
Cable Powering Source Recognition	Automatic
UPS (Power Battery Backup) Functionalities	
UPS Type	Off-line with extremely fast switching time and intermediate power preservation
Power Loss Monitoring	On P1 connector 5 V PINs every 120 uS
Power Loss Battery Backup Activation Condition	• Falling edge detection within 360 us (3 samples) • Voltage level less that 4.3 V (programmable) at any sample
Power Battery Backup Activation Time	Less than 14 uS within a 120 uS window (usual 3uS)
Power Return Conditions	3.6 s after Power Return and Stabilization on Cable Powering Inputs
Battery Backup Electrical Specifications	• 5V DC on P1 connector 5 V PINs • Total Supplied Current by the Battery System at 5 V: ○ Guaranteed 850 mA ○ Tested 1000 mA
Switching Converter Efficiency	More than 85% at 3.7 V input and 850 mA out
Low Powering Mode Current Consumption (all devices are switched off and system is disconnected from the Raspberry Pi®)	• 70 uA with RTC running • When checking if RaspberryPi® has been switched ON, 300 uA for a period of 9 mS, repetitions of checking is programmed by user, default is 5 s
RaspberryPi® Estimated Running Time with typical load of 350 mA	About 2 hours with full battery charge
Interfaces	
UPiS micro USB	Connectivity to: • RaspberryPi® Serial Port • UPiS Serial Port • UPiS Serial Port Data Rate 38400 bps 8N1
UPiS Serial Port	Connectivity to: • RaspberryPi® Serial Port • UPiS micro USB • UPiS Serial Port Data Rate 38400 bps 8N1
Buttons	
UPiS **RST**	Used for: • UPiS System Reset (restarts UPiS, RTC and RaspberryPi®)

	• Activation of Bootloader functionality
UPiS **SDWN**	Used for: • Activation of Bootloader functionality • Restarts RaspberryPi® without resetting UPiS RTC • File Safe RspberryPi® Shutdown Procedure
Indicators	

LED **CHG** (GREEN)	• LIGHT when charger is working and battery is charged
	• DARK when charger is not working:
	  ○ Battery is full charged
	  ○ Battery charger is switched OFF (by the integrated microcontroller)
	• FAST BLINIKNG
	  ○ Battery is not connected
	  ○ Battery is fault

LED **BAT** (GREEN)	• LIGHT when Boost Switching Converter is working (with load or not) • DARK when Boost Switching Converter is working • LIGHT for a short time (9 mS) when system check RaspberryPi® Powering Status in Low Powering Mode
LED **USB** (GREEN)	• LIGHT when UPiS micro USB Powering Cable is connected – it doesn't mean necessary that system is powering from it • DARK when UPiS micro USB Powering Cable is not connected
Dual Color LED **STA** and **STB**	**STB** (RED LED): • DARK if nothing important happens • Flashes for 10 times every 100 mS when measures first time the mean powering values (Voltages, Current and Temperature). It takes 2 second. • LIGHTs for 800 mS every 1 s within 40 s time frame when system is in File Safe Shutdown Procedure • LIGHTs for 100 mS every 1000 mS when UPiS is running on battery backup powering and battery level is higher than 3.4V and lower than 3.6V • LIGHTs for 100 mS every 500 mS when UPiS Advanced is running on battery backup powering and battery level is lower than 3.4V and higher than 3.2V Note: When Battery level is lower than 3.2 V, then system automatically initiates the File Safe RaspberryPi® Shut Down Procedure in order to save power for the RTC. The cut off power of the LiPO Battery is 3 V. **STA** (GREEN LED): • Flashes for 10 times every 20 mS when powering source has been changed (after 3.6 seconds of cable connection or disconnection) • LIGHTs for 100 mS every 500 mS when cable power is connected

<table>
<tr><td></td><td>• LIGHTs for 100 mS every 2000 mS when UPiS is running on battery backup powering and battery level is higher than 3.6V (typical running conditions on battery)
• DARK if system is in Low Powering Mode</td></tr>
<tr><td colspan="2">Sensors</td></tr>
<tr><td>Analog Temperature Sensor</td><td>Accuracy 1 Celsius Degree
Accessible (readable) only via Serial UPiS interface via commands @STATUS or @ANTMPC or @ANTMPF

When Temperature is Higher than 65 Celsius Degree or lower than -10 Celsius Degree then system automatically initiates the File Safe RaspberryPi® Shut Down Procedure.</td></tr>
<tr><td>System Current Sensor</td><td>Real Time System Current measure
Accuracy 10mA
The measurements contains:
• The RaspberryPi® current consumption
• Battery Charging current consumption
• Microcontroller current consumption
• Serial Bridge current consumption
• UPiS LEDs current consumption
• Various Components of the UPiS current consumption

Note: There are already available separated measurements of the RaspberryPi® current consumption and the rest of the system. It will be provided on the Firmware update 1.02 due to testing process.</td></tr>
<tr><td>Battery Voltage Sensor</td><td>Real Time System Current measure
Accuracy 10 mV</td></tr>
<tr><td>micro USB Powering Voltage Sensor</td><td>Real Time System Current measure
Accuracy 10 mV</td></tr>
<tr><td>RaspberryPi® P1 5V Powering Voltage Sensor</td><td>Real Time System Current measure
Accuracy 10 mV</td></tr>
<tr><td colspan="2">Additional Features</td></tr>
<tr><td>Emulated RTC (DS1307) with drift correction system</td><td>Emulated RTC based on the crystal of 32768 KHz. Communicated with RaspberryPi® via I2C and Emulate the DS1307. It will be provided on the Firmware update 1.01 due to testing process.</td></tr>
<tr><td>Timed Start Up/Shutdown</td><td>Programmed on RTC timer starts up and shutdown based on 1 minute intervals.
• Maximum programmed time is 1000 minutes
• Minimum programmed time is 2 minutes</td></tr>
<tr><td>XTEA Encryption System</td><td>XTEA Cryptographic system with User Keys Programming providing Cryptographic protection of professional software written for the RaspberryPi®. It will be provided on the Firmware update 1.02 due to testing process.</td></tr>
<tr><td>Scripting Commands</td><td>A list of 18 commands is already implemented and will be upgraded with more according to user’s requests and future system developments. Each command starts always with ‘@’ There are:
<table><tr><td>@FACTORY
@VERSION
@STATUS</td><td>Factory Defaults
Hardware and Firmware Version
System Status</td></tr></table></td></tr>
</table>

	@ANTMPC	Analog Temperature Celsius
	@ANTMPF	Analog Temperature Fahrenheit
	@PM	Powering Mode
	@CHGR ON/OFF	Charger ON or OFF
	@RPI	RaspberryPi® P1 5V Voltage level
	@USB	UPiS USB Voltage level
	@BAT	UPiS Battery Voltage level
	@CUR	System Current Consumption
	@PWRINFO ON/OFF	Powering change source trigger
	@LPRSTA	Low Powering Restart Time
	@TIME	Time
	@DATE	Date
	@START	Timed RaspberryPi® Startup
	@STOP	Timed RaspberryPi® Shutdown
	@RTCCF	RTC Correction Factor

WARRANTY

UPiS is warranted to be free from defects in materials and workmanship from the date of purchase for a 12 month period. Warranty excludes, normal wear and tear, accidental or deliberate damage.

We will accept returns ONLY on the following conditions. These conditions do not affect your statutory rights.

In the case of faulty items we will repair the items as per the manufacturer's original warranty.

Products which are returned for warranty repair will be forward shipped at the customer's expense. After assessment, and if the products are covered under warranty, we will pay return freight costs. However, if a fault cannot be found, and the product is deemed to be in good working order or not covered by warranty, the customer pays shipping both ways and pays Pi Modules, a service fee prior to the return of the products according to Pi Modules service price list.

If the goods are covered under warranty, we will cover only the return costs, and the repair or replacement of the product as we see fit.

We will only accept for repair only the items, which are returned in the original packaging (including all parts contained in the packaging) and are undamaged (including original packaging).

Warranty excludes, normal wear and tear, accidental or deliberate damage, any such items will not be replaced.

In the case of refund, no such refund will ever exceed the original purchase cost of the item and will not include any associated postage costs.