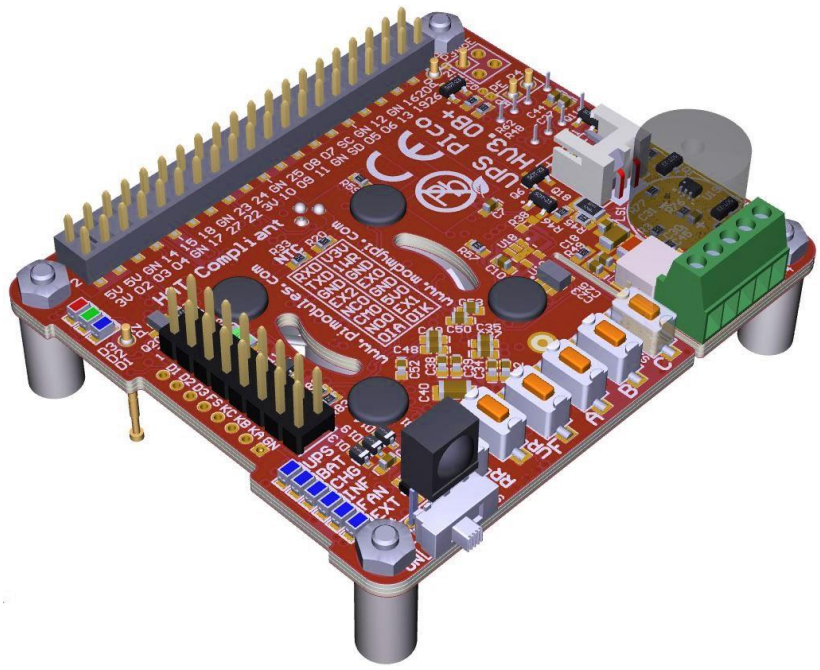
UPS PIco HV3.0A/B/B+ HAT

Versions Stack/Top-End/Plus/Advanced/PoE

Intelligent Mobile Power Bank (Versions B/B+) **and Uninterruptible Power Supply with RTC, Peripherals and I^2C control Interface**

With ultra-High Current Extended Buck Supply of 3.5A (Version B+ Advanced)



User Guide

Especially designed for the Raspberry Pi® 3 Model B+

Compatible with

All former models of Raspberry Pi including ZERO

HAT Compliant

Document Revisions

Version	Date	Modified Sections	Comments
N.A.	06/11/2016	N.A.	First Preliminary Public Document Release
1.0	07/06/2017	all	Public Document Release
2.0	01/02/2018	all	Updated to Version HV3.0B/B+
3.0	15/06/2018	all	Updated to Version HV3.0B/B+ with new functionalities and B+ assembly Instructions
4.0	23/09/2018	all	Updated all paragraphs with new additional functionalities, revised the old one according to the new firmware
5.0	30/09/2019	BS	Updated Basic Scheduler

Table 1 Document Revisions

Table of Contents

Unlocked Firmware Features

Version	Date	Unlocked Features
Initial	Initial	Interrupts driven interaction with Raspberry Pi® based on Daemons
Initial	Initial	Simple status email broadcasting system based on Daemons
Initial	Initial	Intelligent Uninterruptible Power Supply (UPS)
Initial	Initial	3 User defined keys handler
Initial	Initial	3 User defined LEDs handler
Initial	Initial	LiPO and LiFePO4 chemistry battery support
Initial	Initial	Automatic Battery Charger for LiPO and LiFePO4 batteries
Initial	Initial	Battery Charger Power Tracking (Only Version Plus on External Powering Input)
Initial	Initial	Integrated Hardware RTC with Battery Back-up
Initial	Initial	Automatic Files Safe Shutdown and Wake-up (when Cable Power is back)
Initial	Initial	Single button System Shutdown/Startup and ON/OFF for Battery and Cable (Only Version Plus) Powered Applications
Initial - 0x30	Initial/15.03.2017	PWM FAN Control with automatic ON/OFF (added on 15.03.2017)
Initial	Initial	Zero Power Bi Stable Relay Control
Initial	Initial	Programmable Integrated Sounder
Initial	Initial	System Monitoring: Temperatures, Voltages
Initial	Initial	Programmers I^2C PICo Interface
Initial	Initial	Programmers RS232 Interface (Basic Version)
Initial	Initial	Boot loader for onsite firmware update
Initial	Initial	3 x A/D converters (Basic Version, without high voltage interface)
0x30	15.03.2017	3 x A/D converters Raw Data (without conversion to Voltages)
0x30	15.04.2017	Programmable Auxiliary Battery Backed up 5V@750mA and 3.3V@150mA supply
0x34	15.04.2017	User Selectable PIco HV3.0 I^2C addresses (NORMAL, NO_RTC, ALTERNATE)
0x35	20.05.2017	Activated Still Active Timer (watch-dog for the Raspberry Pi)

Table 2 Unlocked Firmware features

Firmware and Manual Fixes/Add on

System Overview

Introduction

The **UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus/Advanced 450** is an advanced **Intelligent Mobile Power Bank** and **Uninterruptible Power Supply** for the Raspberry Pi® A+/B+/2/3 and ZERO/W, that adds a wealth of innovative powering/backup functionality and development features to the innovative microcomputer! The **UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus/Advanced 450 will** automatically shut-down your Raspberry Pi® if there is a power failure, supply mobile applications from battery source, and can be set to automatically monitor and reboot your Raspberry Pi® once power has been restored!

If used as **Mobile Power Bank** it is equipped also with an **Intelligent Externally Accessed** (with **Files Safe Shutdown functionality**) **Power Slide Switch** that allows to safety **System Switch ON/OFF** whenever you like, without worrying about files corruption as it is always properly shutdown the system before battery will be disconnected (keep battery connected until system shutdown)!

The **UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus/Advanced 450** features a **5V 3A total current output** when battery powering, designed for use on the latest Raspberry Pi® 3 Model B+ as also former Raspberry Pi ® modules!

UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus/Advanced 450 offers now **3** User Programmable Keys, **3** separate User programable LEDs with different colors, support for **multiple** and different chemistry of a high capacity batteries, **bi-stable relay** (Zero Power) configured as **DPDT** or **SPDT**, as also **3 x A/D 12-bit** converters with pre-adjustable readings to 5.2V. As also 10V, 20V and 30V voltage conversion (when used with **Terminal Blocks PCB** or separate external resistors). Now, with number of embedded sensors (inbound current, outbound current, temperature, voltages), **true 5V 1-wire** interface, optional high voltage RS232 interface and many, many additional features!!

The **UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus/Advanced 450** is standard equipped with a **450mAh 15C LiPO battery (able to supply up to 6.5A)** specially designed to enable safe shutdown when cable power cuts. Additionally, this can be easily upgraded to the extended 1500mAh, 4000mAh, 8000mAh or even 12000mAh (on Special Order) capacities, which enables prolonged use of a Raspberry Pi for **more than 24 hours** without a power supply connected!

The **UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus/Advanced 450** design support now batteries with different chemistry like: **LiPO, Li-Ion** as also **LiFePO4**. Especially the **LiFePO4** batteries are addressed to applications where temperatures environment is more restricted as can be used for supplying from -**10 degrees up to +60 degrees**. In addition, the **LiFePO4** have a unique extremely long life of charging/discharging that can achieve up to **2000 cycles** or **10 years life time!!**

Now, with new add-on board (**PIco LP/LF Li-Ion 18650 Battery Holder**) you can use all **Li-Ion 18650 batteries** from electronic cigarettes wide available on the local markets approaching total capacity of 7200mAh, as also 18650 LiPO and LiPo4Fe (called also 123).

The **UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus/Advanced 450** with additional **External Supply Powering Input**; that has implemented **Dynamic Power Tracking (**based on **Voltage Proportional Charge Control – especially designed for Solar Cells)**; automatically adjust battery charging current according to power availability from 80mA – 800 mAh, in order to use all available energy from the Solar Panel in case of use. This feature has been especially designed to support **Solar Panel Powering Raspberry Pi® Systems**, as it is adjusting the charging battery current to available Sunning conditions, which is varying due to unstable sunning conditions. The **External Supply Powering Input** is able to accept power from **7 V DC** up to **28 V DC!!** Thus, make it ideal for Cars, Trucks, Buses and any industrial applications where voltage is usually higher than 24V DC. The **External Supply Powering Input** is equipped with Over Current protection, Over Voltage protection, ESD protection as also with Zero Voltage Drop Inverse Polarity Protection protecting Raspberry Pi® System from improper usage, but also offers, due to zero voltage drop, usage of most of available energy from the **Solar Panel** in case of use.

The **UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus/Advanced 450** is powered and the Battery Pack intelligently charged via the GPIO pins on the Raspberry Pi®, therefore no additional cabling or power supply is required (if used Raspberry Pi® PSU 5V supply). Due to that fact **UPS PIco HV3.0B HAT Plus 450** requires no external cable powering and fits within the footprint of the Raspberry Pi®, it is compatible with most of available cases. If powered via External Power Input (7V-28VDC) the there are cases available to hold your designed system. Especially the **Top-End** model can be fit-in to the **Official Raspberry Pi case**.

Also, in case the **UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus/Advanced 450** is powered from the **Extended Power Input,** it allows to **charge the battery even if Raspberry Pi® is not powered.** Thus, functionality in combination with **Events Scheduler** make the system always full of energy when needed to be running.

Professional developers often need to protect their Applications Intellectual Properties. To support them **UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus/Advanced 450** offers the **XTEA** dual path encryption engine that protect the developed software with the unique secure code assigned by Application developer**.**

The new PCB with **2 oz copper** and **4 layers,** is designed especially for high current powering systems offering **Multilayer Copper Thermal Pipes** for increased System Thermal Response and better passive cooling!!

The **UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus/Advanced 450** can also be equipped with an optional **Infra-Red Receiver** interface which is routed directly to GPIO18 if used.

The embedded **Electromagnetic Programmable Sounder** can be used as a **simple buzzer** but also as **music player** due to implemented sound generator and dedicated programmer interface.

The **IoT** developers will find very useful the **3 independent ESD protected 12 bits buffered A/D converters** as also number of internal sensors and sensor interfaces that can be used

for system monitoring such as Battery Voltage, Raspberry Pi Voltage, Inbound/Outbound Current measure, System Temperature and true **5V 1-wire interface**.

The integrated **Hardware RTCC** enables a new extremely usefully feature – the **Events Triggered RTCC Based System Actions Scheduler**. The **Events Triggered RTCC Based System Actions Scheduler** allows to timely start up, or shutdown the **Raspberry Pi®** on various internal or external events that include, data available on RS232, A/D, RTCC, and temperature, or just on requested Time Stamp. It is also possible to trigger actions (i.e. relay switch) without participation of the Raspberry Pi®.

Finally, the **UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus/Advanced 450** features an implemented Automatic Temperature Control **PWM FAN controller**, and can be equipped with a **micro fan kit**, which enables the use of the Raspberry Pi® in extreme conditions including very high temperature environments. The FAN speed can be manually/automatically adjusted according to system temperature conditions linear from 0 % (FAN is OFF) up to 100% by increasing and decreasing rotation speed. Thus, guarantees the possible lowest level of noise and always cool **Raspberry Pi® 3 Model B+.**

Available UPS PIco HV3.0 HAT Models

The following model series of the **UPS Pico HV3.0** has been manufactured:

1. UPS Pico HV3.0A
    a. UPS PIco HV3.0A HAT Stack 450
    b. UPS PIco HV3.0A HAT Top-End 450
    c. UPS PIco HV3.0A HAT Plus 450
2. UPS Pico HV3.0B
    a. UPS PIco HV3.0B HAT Stack 450
    b. UPS PIco HV3.0B HAT Top-End 450
    c. UPS PIco HV3.0B HAT Plus 450
3. UPS Pico HV3.0B+
    a. UPS PIco HV3.0B+ HAT Stack 450
    b. UPS PIco HV3.0B+ HAT Top-End 450
    c. UPS PIco HV3.0B+ HAT Advanced 450 (similar to former called “Plus”)

Each newer model is compatible with all former models in firmware (the firmware automatically recognizes on which model is running), as also cover current (available on production date) available models of the Raspberry Pi and backward.

Specifically, the newest and most innovative model **UPS Pico HV3.0B**+ (all versions: Stack, Top-End and Advanced) has been designed especially for the new Raspberry Pi3 Model B+, however it is compatible with all former models of Raspberry Pi including version ZERO/W.

Due to very progressive design, the newest version the **UPS Pico HV3.0B**+ is also compatible with ongoing official **Raspberry Pi PoE PCB** placed on top of it. In order to have access to Raspberry Pi PoE pins, a simple PoE header is needed to be used, offered by our company in ultra-low cost.

Simplifying, any of models of the Raspberry Pi that contains 40 Pin GPIO connector is compatible with UPS Pico HV3.0.

The differences between each model are listed here below in the table with Technical Specifications however the core features for all models are presented here below.

UPS PIco HV3.0 HAT Core Features

The list of features of the **UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus 450** are as follows:

General

- Raspberry Pi B+ **HAT Compliant** (HAT dimensions and HAT EEPROM)
- **Plug and Play –** Ultra Simple Semi-**Automatic Installation** via GitHub
- Available for download 8GB latest version of NOOBS with preinstalled Daemons
- **Standard Interrupts driven interaction with Raspberry Pi® based on Daemons** using GPIO27 (Pin13) & GPIO22 (Pin15)
- (Optional) **GPIO free** (all GPIOs are available for user application) **interaction with Raspberry Pi®** is based on current consumptions and I2C activity
- Simple **status email broadcasting application based on Daemons** when Powering Status Changed
- Enhanced **System Monitoring** and **Programming API**
- **Labeled J8 Raspberry Pi® GPIO Pins** for Easy Plug & Play of experimental cables
- Standard **THT 40 Pin** connector (not soldered – pass through)
- (optional) **Remote bootloader** for Live Firmware Update (**HV3.0B/B+ only)**
- **Local bootloader** (standard) for Live Firmware Update

Powering Options

- **External Powering Input** (with ON/OFF capability) **3A@7-28VDC** (Version HV3.0A/B Plus only)
- **External Powering Input** (with ON/OFF capability) **3.5A@7-28VDC** (Version HV3.0B+ Advanced only)
- **Intelligent Uninterruptible Power Supply** (UPS)
- **Mobile Battery Power Bank** starts-up without cable power cycling (Version HV3.0B/B+ only)
- **File Safe Shutdown** and **Start-up** Functionality on a Single Button
- **Single slide ON/OFF switch for battery powered (mobile) applications** running without power cycling (with **File Safe Shutdown functionality when switching OFF** (Version HV3.0B/B+ only)
- Possibility to solder **external ON/OFF switch** - Ready Soldering Pads (Version HV3.0B/B+ only)
- **Integrated LiPO Battery** 450 mAh 15C (10-15 Minutes of Power Back-Up)
- **Automatic Shutdown** when low battery indicated
- **Automatic Start-up** when cable power return
- **5V 2.6A Power Backup (Peak Output 5V 3A)**
- **No Additional External Power Input Required.** System is monitoring power status over 5V GPIOs, therefore is compatible with 99.99% of all existing cases
- **Additional programmable 5V power source with battery backup,** available for user applications also when Raspberry Pi is OFF (5V@750mA) **protected with PPTC FUSE** and **reverse current flow diode**, controlled by User and RTC Scheduler.
- **Additional programmable 5V power source with battery backup,** 0.2A@3.3V protected output (sourced from independent and dedicated LDO) , controlled by User and RTC Scheduler.

Supported Batteries Types and Capacities

- **Support for LiPO, LiFePO4 and Li-Ion Chemistry Batteries** on the same PCB (with high current cable connection) with dedicated plastic base
- **Support for Li-Ion 18650 low cost batteries** (from Electronic Cigarettes) with dedicated mounting base PCB screwed on top
- **Support for LiPO 18650 batteries** with dedicated mounting base PCB screwed on top
- **Support for LiFePO4 18650 batteries** with dedicated mounting base PCB screwed on top
- Intelligent **Automatic Battery Charger**
- **Voltage Proportional Charge Control** (only for version Plus/Advanced)
- Available Standard Batteries Capacities are:
    - LiPO 1500 mAh
    - LiPO 4000 mAh
    - LiPO 8000 mAh
    - LiPO 12000 mAh on special order
    - LiFePO4 4000 mAh
    - LiFePO4 8000 mAh
    - LiFePO4 12000 mAh on special order
    - Li-Ion from 1200 mAh up to 7200 mAh
    - Any user selected 18650 battery capacity and chemistry,

Embedded Peripherals and Interfaces

- **3 User Programmable LEDs** for user own application **with additional connectivity** to external User LEDs (HV3.0B/B+ only)
- **3 User Programmable Buttons** for user own application **with additional cable connectivity** to external User Buttons (HV3.0B/B+ only)
- **System File Safe Shutdown/Start-up button** with additional cable connectivity to external button (HV3.0B/B+ only)
- **Single slide ON/OFF switch for battery powered applications** with additional cable connectivity to external User Switch (OFF is always combined with File Save Shutdown capability) (HV3.0B/B+ only)
- **Single slide ON/OFF switch for cable powered application** allowing to switch OFF Raspberry Pi even if micro USB cable is still connected – Raspberry Pi 3 Model B+ PEN option (HV3.0B+ only)
- Standard equipped with **Bi Stable Relay (Latching Relay - Zero Power)** assembled on two different mounting positions:
    - with two galvanic isolated independent contacts DPDT 1A/30V
    - with single high current contacts SPDT 2A/30V (HV3.0B only)
- with two galvanic isolated independent contacts DPDT 2A/30V(HV3.0B+ only)
- Standard equipped with **Opto-Coupler Interface**, useful for High Voltage Interfaces or Interfaces where separated grounding is needed. The Opto-Coupler Interface can be read as digital (Hi/Low) or Analog Value (only version Plus/Advanced)
- Integrated **True 5V,** ESD protected **1-wire interface** (with voltage converter to 3.3V) connected directly to the GPIO4 (HV3.0B/B+ only)

- Integrated **ESD-Protected 3 x 12-bit A/D** converters with voltage conversion embedded calculators and raw data option (implemented in firmware - extensive Lowpass and Olympic Score filtering):
  - 0V-5.2V
  - 0V-10V
  - 0V-20V
  - 0V-30V
- **Infra-Red Receiver** Sensor Interface (IR Not Included) directly connected to the GPIO18
- **Programmable Integrated PWM Sounder** (programmable by user API or Automatic), able to play music
- Integrated **Hardware Real Time Clock (RTC)** with Battery Back-Up
- **PWM fan control** with dedicated Temperature sensor touching the Raspberry Pi ® PCB, based on Raspberry Pi or Embedded Temperature Sensor (Fan need to be ordered separately)
- (optional) On Battery Powered **System Available Running Time** (calculated on battery capacity, Battery Level and System Current Consumption)

- (optional) **second RS232 port** (5V tolerant, or 12V via Terminals Block PCB)

Embedded Sensors

- **Outbound current** measure sensor when Battery powered
- **Inbound current** measure sensor when Cable powered
- **NTC** based **onboard** temperature sensor
- (Optional) TO92 Temperature sensor
- Battery Level Voltage
- Raspberry Pi GPIO 5V level

User/Programmer Interface

- **I²C PICo API Interface** for Control and Monitoring, with over 50 programming/reading registers
- Support for **3 different** users selectable I2C addresses sets:
  - **DEFAULT:** 0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F
  - **NO_RTC:** 0x69, 0x6B
  - **ALTERNATE:** 0x58, 0x59, 0x5A, 0x5B, 0x5C, 0x5D, 0x5E, 0x5F

System Schedulers

- **Basic Time Scheduler**
- **Event Triggered RTC Based System Actions Scheduler** (ETR SAS)
  System can wakeup and sleep on external or internal events like:

  - temperature,
  - 3 x A/D levels,
  - voltage,
  - RS232 data;

- as also can trigger Actions like: Relay, Auxiliary Voltage ON/OFF, RS232 data **with or without involvement** of the Raspberry Pi®. Always based on internal Hardware RTC

Case Compatibility

- **No Additional External Power Input Required (for version Stack and Top-End).** System is monitoring power status over 5V GPIOs, therefore is compatible with 99.99% of all existing cases
- **Fits Inside Most Existing Cases as no extra cabling is needed**
- **Fits inside to the Official Raspberry Pi Case with closed lid** (version Top-End only)

System Monitoring

- **Status Monitoring** – Powering Mode, Inbound current, Outbound current, Powering Voltage, UPS Battery Voltage, Current and Temperature
- **Events Pi** Log feature if Raspberry Pi serial port is available
- **System LEDs** – UPS, BAT, CHG, INF, FAN (optionally selected can be mapped to User LEDs)
- **System Healthy**, that informs user remotely if Raspberry Pi and UPS Pico HV3.0 are running properly and system is power protected (based on various internal system triggers)

User Applications Security

- (optional) **2**-way **XTEA** Based **Encryption Engine** for **User Intellectual Properties** protection

System Protection

- Direct **Raspberry Pi® Hardware Reset Button** via Spring Test Pin (pogo pin)
- **Programmable Watch-Dog Hardware** feature (**St**ill **A**live Timer)
- **PPTC 2.6A fuse**
- **ZVD circuit** on 5V GPIO connections
- **Microcontroller watch-dog**
- Over Temperature protection
- Over Current protection
- Zero Voltage Drop Inverse Polarity Protection (only version Plus/Advanced)

System Design

- Designed and Analyzed with one of the most advanced CAD/CAM Tools - Mentor Graphics PADS (www.pads.com)
- Design Based on Microchip 16-bit 16MIPS micro controller
- Industrial Design Originated

PCB Construction

- **2 oz copper** PCB manufactured for proper high current supply
- 8mils track/8mils gap technology **4 layers PCB**

- PCB Surface Finishing - Immersion Gold
- **Multilayer Copper Thermal Pipes** for increased System Thermal Response and better passive cooling

UPS PIco HV3.0 HAT Add-On equipment

The **UPS PIco HV3.0** can be combined with additional already available parts. There are:

- UPS PIco HV3.0 Fan Kit
- UPS PIco HV3.0 Relay Kit, Bi-Stable (Latching), Zero Power Relay, configurable for a double DPDT 1A/30V or single SPDT 2A/30V.
- Infra-Red Receiver which is routed directly to GPIO18 via the PCB for remote IR operations when installed
- PIco LP/LF Li-Ion 18650 Battery Holder (single or double) that allows using all Li-Ion 18650 batteries from electronic cigarettes wide available on the local markets, as also 18650 LiPO and 18650 LiPo4Fe (known as 123 type).
- LiPO Battery 4000 mAh
- LiPO Battery 8000 mAh
- LiFePO4 Battery 4000 mAh
- LiFePO4 Battery 8000 mAh
- LiFePO4 or LiPO 12000 mAh (only on special order)
- Terminal Blocks PCB offering 12V RS232 interface, and all I/O interfaces Terminal Blocks capabilities
- Dedicated UPS PIco HV3.0 Plexiglas case for battery 4000 mAh LiFePO4, LiPO or 18650 Battery Holder

UPS PIco HV3.0 HAT 450 Technical Specifications

Features	UPS PIco HV3.0B HAT Models		
	UPS PIco HV3.0B HAT Stack 450	UPS PIco HV3.0B HAT Stack 450 Plus	UPS PIco HV3.0B HAT Top-End 450
Raspberry Pi®			
Raspberry Pi® System Compatibility			
Compatible Raspberry Pi Models	Designed for Raspberry Pi® 3 Compatible with Pi2, Pi3, Pi Zero/W, A+, B+	Designed for Raspberry Pi® 3 Compatible with Pi2, Pi3, Pi Zero/W, A+, B+	Designed for Raspberry Pi® 3 Compatible with Pi2, Pi3, Pi Zero/W, A+, B+
Cases Compatibility			
Cases	Most of the cases ModMyPi cases PiModules PIco case	Most of the cases ModMyPi cases PiModules PIco case	Most of the cases **Recommended Raspberry Pi Original Case**
Raspberry Pi® GPIO Usage (occupation)			
Permanent use of I^{2}C **User selectable addresses**	GND, 5V, SDA0, SCL0 I^{2}C Addresses 1: **68 69 6a 6b 6c 6d 6e 6f** I^{2}C Addresses 2: **58 59 5a 5b 5c 5d 5e 5f** I^{2}C Addresses 3: **69 6b**	GND, 5V, SDA0, SCL0 I^{2}C Addresses 1: **68 69 6a 6b 6c 6d 6e 6f** I^{2}C Addresses 2: **58 59 5a 5b 5c 5d 5e 5f** I^{2}C Addresses 3: **69 6b**	GND, 5V, SDA0, SCL0 I^{2}C Addresses 1: **68 69 6a 6b 6c 6d 6e 6f** I^{2}C Addresses 2: **58 59 5a 5b 5c 5d 5e 5f** I^{2}C Addresses 3: **69 6b**
Selectable use of Raspberry Pi® RS232	GND, TXD0, RXD0 OFF(HiZ)	GND, TXD0, RXD0 OFF(HiZ)	GND, TXD0, RXD0 OFF (HiZ)
Selectable use of Raspberry Pi® GPIO **Optional**	GPIO_GEN22 (pulse train generator) GPIO_GEN27 (System Shutdown initiator) GPIO_GEN18 (if IR receiver is used) GPIO_GEN4 (if 1-wire is used) None of GPIO used	GPIO_GEN22 (pulse train generator) GPIO_GEN27 (System Shutdown initiator) GPIO_GEN18 (if IR receiver is used) GPIO_GEN4 (if 1-wire is used) None of GPIO used	GPIO_GEN22 (pulse train generator) GPIO_GEN27 (System Shutdown initiator) GPIO_GEN18 (if IR receiver is used) GPIO_GEN4 (if 1-wire is used) None of GPIO used
Interactions with Raspberry Pi®			
Standard **Optional**	GPIO_GEN22 (pulse train generator) GPIO_GEN27 (pulse replying and System Shutdown initiator) I^{2}C and current measure	GPIO_GEN22 (pulse train generator) GPIO_GEN27 (pulse replying and System Shutdown initiator) I^{2}C and current measure	GPIO_GEN22 (pulse train generator) GPIO_GEN27 (pulse replying and System Shutdown initiator) I^{2}C and current measure
Batteries and Charger			
Supported Batteries Types			
LiPO 3.7V with silicone high current cables			
	Standard - LiPO 450 mAh	Standard - LiPO 450 mAh	Standard - LiPO 450 mAh (dedicated to be used with Raspberry Pi Original Case)
	Optional - LiPO 4000 mAh	Optional - LiPO 4000 mAh	
		Optional - LiPO 8000 mAh	
LiFePO4 3.2V with silicone high current cables			
	Optional – LiFePO4 4000	Optional - LiFePO4 4000 mAh	
		Optional - LiFePO4 8000 mAh	
		Optional - LiFePO4 12000 mAh (due to big size of batter only on special order)	
Li-Ion 3.7V with silicone high current cables	Optional – Li-Ion 3200 mAh	Optional – Li-Ion 3200 mAh	Optional – Li-Ion 3200 mAh
Additional Batteries Options			
Pico Single LP/LF/Li-Ion 18650	Held 18650 single batteries (all	Held 18650 batteries (all supported	Held 18650 batteries (all supported

Battery Holder **Pico Double Li-Ion 18650 Battery Holder**	supported types) up to 3200 mAh, with extra reverse polarity protection Held 18650 double batteries (ONLY LI-Ion Type) up to 3200 mAh, with extra reverse polarity protection	types) up to 3200 mAh, with extra reverse polarity protection Held 18650 double batteries (ONLY LI-Ion Type) up to 3200 mAh, with extra reverse polarity protection	types) up to 3200 mAh, with extra reverse polarity protection Held 18650 double batteries (ONLY LI-Ion Type) up to 3200 mAh, with extra reverse polarity protection
Battery Life Charge/Discharge Cycles			
LiPO	450 cycles	450 cycles	450 cycles
LiFePO4	2000 cycles	2000 cycles	2000 cycles
Li-Ion	300 cycles	300 cycles	300 cycles
Battery Charger			
	Standard - Continues fixed current 303 mAh	Automatic Dynamic Power Tracing (Voltage Proportional Charge Control – especially designed for Solar Cells support) Charger with charging current 100 mA – 800 mA, triggered by voltage changes on the 5V GPIO or External Power Source	Standard - Continues fixed current 303 mAh
Charging Modes			
LiPO	Automatic Selected: Full Charging Cycle Trickle Charging	Automatic Selected: Full Charging Cycle Trickle Charging	Automatic Selected: Full Charging Cycle Trickle Charging
LiFePO4	Automatic Selected: Full Charging Cycle Trickle Charging	Automatic Selected: Full Charging Cycle Trickle Charging	Automatic Selected: Full Charging Cycle Trickle Charging
Li-Ion	Automatic Selected: Full Charging Cycle Trickle Charging	Automatic Selected: Full Charging Cycle Trickle Charging	Automatic Selected: Full Charging Cycle Trickle Charging
Battery Protection			
450 mAh	On board cut-off protection system when thermal, overcharge or over discharge	On board cut-off protection system when thermal, overcharge or over discharge	On board cut-off protection system when thermal, overcharge or over discharge
High Capacity Li-Ion, LiPO and LiFePO4	On board cut-off protection system when thermal, overcharge or over discharge On battery, PCM additional protection	On board cut-off protection system when thermal, overcharge or over discharge On battery PCM additional protection	On board cut-off protection system when thermal, overcharge or over discharge On battery PCM additional protection
Battery Electrical Isolation System	Battery is Electrically Isolated (however cable connected) until system start up for the first time and receive 5V from GPIO	Battery is Electrically Isolated (however cable connected) until system start up for the first time and receive 5V from GPIO or 7-28V from EXT	Battery is Electrically Isolated (however cable connected) until system start up for the first time and receive 5V from GPIO
Optional	Slide ON/OFF switch (external or internal), OFF always with File Save shutdown functionality	Slide ON/OFF switch (external or internal), OFF always with File Save shutdown functionality	Slide ON/OFF switch (external or internal), OFF always with File Save shutdown functionality
Battery Back-Up			
System Battery Backup	Standard – 5V 2.6A current continuous supply to Raspberry Pi via GPIO Pins	Standard – 5V 2.6A current continuous supply to Raspberry Pi via GPIO Pins	Standard – 5V 2.6A current continuous supply to Raspberry Pi via GPIO Pins
Auxiliary 5V and 3V3 Battery Backed Supply on PIco I/O Pins	Standard – 5V 750 mA current and 3V3 continuous supplies on PIco I/O Pin battery backed, with possibility to continuous supply auxiliary devices with Raspberry Pi disconnected. Total system current should not exceed 3A.	Standard – 5V 750 mA current and 3V3 continuous supplies on PIco I/O Pin battery backed, with possibility to continuous supply auxiliary devices with Raspberry Pi disconnected. Total system current should not exceed 3A.	Standard – 5V 750 mA current and 3V3 continuous supplies on PIco I/O Pin battery backed, with possibility to continuous supply auxiliary devices with Raspberry Pi disconnected. Total system current should not exceed 3A.
Battery Back-up Type			
UPS	UPS Standby Type, with switch over time of 250 uS, during switching time the protected system (Raspberry Pi® with added hardware) is powered by	UPS Standby Type, with switch over time of 250 uS, during switching time the protected system (Raspberry Pi® with added hardware) is powered by	UPS Standby Type, with switch over time of 250 uS, during switching time the protected system (Raspberry Pi® with added hardware) is powered by

	auxiliary online power source for maximum 10mS, therefore no power gap on GPIO during switching time	auxiliary online power source for maximum 10mS, therefore no power gap on GPIO during switching time	auxiliary online power source for maximum 10mS, therefore no power gap on GPIO during switching time
Powering Monitoring Point	Raspberry Pi® GPIO 5V	Raspberry Pi® GPIO 5V	Raspberry Pi® GPIO 5V
UPS Activation Powering Triggers	GPIO 5V pins <=4.65V Proprietary Algorithm of Falling Power Peak Analysis	GPIO 5V pins <=4.65V Proprietary Algorithm of Falling Power Peak Analysis	GPIO 5V pins <=4.65V Proprietary Algorithm of Falling Power Peak Analysis
Cable Powering Reactivation	After 3s of continuously cable powering (without spikes)	After 3s of continuously cable powering (without spikes) on any cable power source (GPIO or External)	After 3s of continuously cable powering (without spikes)
Intelligent Mobile Power Bank			
Direct Battery Powering with Internal/External ON/OFF Slide Switch	ON/OFF Slide Switch with File Safe Shutdown functionality when switching to OFF (keep battery powering ON until system shutdown)	ON/OFF Slide Switch with File Safe Shutdown functionality when switching to OFF (keep battery powering ON until system shutdown)	ON/OFF Slide Switch with File Safe Shutdown functionality when switching to OFF (keep battery powering ON until system shutdown)
Cable Powering Sources			
Cable Powering Sources			
Raspberry Pi ® GPIO 5V Pins	2.6 A	2.6 A	2.6 A
External Power Source 7 - 28 VDC		3A max (adjusted according dynamic power tracking algorithm - Voltage Proportional Charge Control – especially designed for Solar Cells)	
Additional Features - Peripherals			
HAT Compliant			
HAT EEPROM	Exists	Exists	Exists
HAT Dimensions	Compliant	Compliant	Compliant
PIco I/O Interface			
Independent from Raspberry Pi ® 3.3 V supply @200 mA With battery Back-up (Raspberry Pi ® can be OFF when this power Auxiliary 3.3 V source is available)	Yes	Yes	Yes
ESD Protected True 5V 1-wire interface	Yes	Yes	Yes
Independent from Raspberry Pi ® 5.0 V supply @750 mA With battery Back-up (Raspberry Pi ® can be OFF when this power Auxiliary 5 V source is available)	Yes	Yes	Yes
12 Bit A/D converters ESD protected, pre-scaled to 5V, 10V, 20V and 30V (on TB PCB) with Sampling rate 100K SPS, buffered	Yes	Yes	Yes
3V3/5V0 RS232 Port that can be programmed as: primary Raspberry Pi® Port Secondary (independent from the existing on Raspberry Pi®)	Yes	Yes	Yes
Optical Isolated Interface (readable as digital or analog)	none	Yes	none
Primary 3 Pin Bi-stable (Zero Power) Relay Interface Rating (resistive) Maximum Switching Current/Voltage on Terminal Block Current/Voltage on 16 Pin Header	Yes (Optional) with two galvanic isolated independent contacts DPDT 1A/30V with single high current contacts SPDT 2A/30V	Yes (Standard) with two galvanic isolated independent contacts DPDT 1A/30V with single high current contacts SPDT 2A/30V	Yes (Optional) with two galvanic isolated independent contacts DPDT 1A/30V with single high current contacts SPDT 2A/30V

PIco Terminals Block Extension PCB (Supplied separately)			
12 V RS232 converter attached to primary or secondary Serial Port	Yes (Optional with TB PCB)	Yes (Optional with TB PCB)	Yes (Optional with TB PCB)
Terminal Block on Each PIco I/O Interface listed above	Valid only for existing Interfaces	Valid only for existing Interfaces	Valid only for existing Interfaces
PIco Plus Terminal Block Standard Interface			
DC in 7 – 28 V with Power Tracking	none	Yes	none
Secondary 3 Pin Bi-stable (Zero Power) Relay Interface	Optional if Relay Installed	Yes	Optional if Relay Installed
Hardware User Interface			
System LEDs Indicators	UPS, BAT, CHG, INF, FAN	UPS, BAT, CHG, INF, FAN, EXT	UPS, BAT, CHG, INF, FAN
User LEDs Indicators	Blue, Green, Red With capability to connected external LEDs	Blue, Green, Red With capability to connected external LEDs	Blue, Green, Red With capability to connected external LEDs
System Keys	RPiR, UPSR, FSSD	RPiR, UPSR, FSSD	RPiR, UPSR, FSSD
User programmable Keys	AKEY, BKEY, CKEY	AKEY, BKEY, CKEY	AKEY, BKEY, CKEY
External Connectivity to PIco Keys	FSSD, AKEY, BKEY, CKEY With capability to connected external KEYs) ON/OFF slide Switch	FSSD, AKEY, BKEY, CKEY With capability to connected external KEYs) ON/OFF slide Switch	FSSD, AKEY, BKEY, CKEY With capability to connected external KEYs) ON/OFF slide Switch
Audio Interface	Electromagnetic Transducer, with programmable sound duration and frequency, able to play music	Electromagnetic Transducer, with programmable sound duration and frequency, able to play music	Electromagnetic Transducer, with programmable sound duration and frequency, able to play music
Other Features			
Battery Backed Hardware Real Time Clock and Calendar	Yes Only when UPS (power cycling is used)	Yes Only when UPS (power cycling is used)	Yes Only when UPS (power cycling is used)
Bi-Stable (Zero Power) Relay	Yes (optional)	Yes	Yes (optional)
Passive Cooling System	Based on multiple copper layers thermal pipes for heating dissipation	Based on multiple copper layers thermal pipes for heating dissipation	Based on multiple copper layers thermal pipes for heating dissipation
Automatic Active Cooling System (FAN)	Yes (optional if FAN installed) based on temperature of the Raspberry Pi® PCB read by separate external Sensor	Yes (optional if FAN installed) based on temperature of the Raspberry Pi® PCB read by separate external Sensor	Yes (optional if FAN installed) based on temperature of the Raspberry Pi® PCB read by separate external Sensor
Automatic File Safe Shutdown Functionality	Yes	Yes	Yes
Raspberry Pi® Reset via POGO Pin	Yes	Yes	Yes
Automatic Restart on Power Return	Yes	Yes	Yes
Events Triggered RTCC Based System Actions Scheduler	Yes	Yes Extended on more Events	Yes
Real Time Raspberry Pi® current measure	Yes (both ways) Incoming to UPS PIco Outgoing from UPS PIco	Yes (both ways) Incoming to UPS PIco Outgoing from UPS PIco	Yes (both ways) Incoming to UPS PIco Outgoing from UPS PIco
Real Time Battery Capacity Measure	Yes (based on System current consumption)	Yes (based on System current consumption)	Yes (based on System current consumption)
Secondary Serial Port (based on software driver)	Yes (future firmware option)	Yes (future firmware option)	Yes (future firmware option)
IR interface	Yes	Yes	Yes
Optimized design for a very low noise A/D operation	Yes Split grounds, extended Improved filtering on PSU High Speed Separate Tracing	Yes Split grounds, extended Improved filtering on PSU High Speed Separate Tracing	Yes Split grounds, extended Improved filtering on PSU High Speed Separate Tracing
Optimized Ultra Low Power design for a long time Battery System Operation	Yes	Yes	Yes
XTEA Encryption	Yes	Yes	Yes
Extended Raspberry Pi® Watch-Dog (Still Alive)	Yes	Yes	Yes
System Monitoring	Battery Voltage, Raspberry Pi® Voltage,	Battery Voltage, Raspberry Pi® Voltage,	Battery Voltage, Raspberry Pi® Voltage,

	Current Consumption by Raspberry Pi® and PIco, Temperature	External Voltage, Current Consumption by Raspberry Pi®, Temperature	Current Consumption by Raspberry Pi® and PIco, Temperature
I2C PIco Programmer Interface	Yes	Yes	Yes
RS232 @command Interface on Primary and Secondary Serial Port	Yes	Yes	Yes
Bootloader for Live Firmware Update	Yes	Yes	Yes
PCB Construction			
PCB Manufacturing	4 Layers, 2 OZ Copper, 8mils/8mils Immersion Gold Plated PB Free alloy assembly	4 Layers, 2 OZ Copper, 8mils/8mils Immersion Gold Plated PB Free alloy assembly	4 Layers, 2 OZ Copper, 8mils/8mils Immersion Gold Plated PB Free alloy assembly

Table 3 UPS PIco HV3.0 HAT Stack 450 Technical Specifications

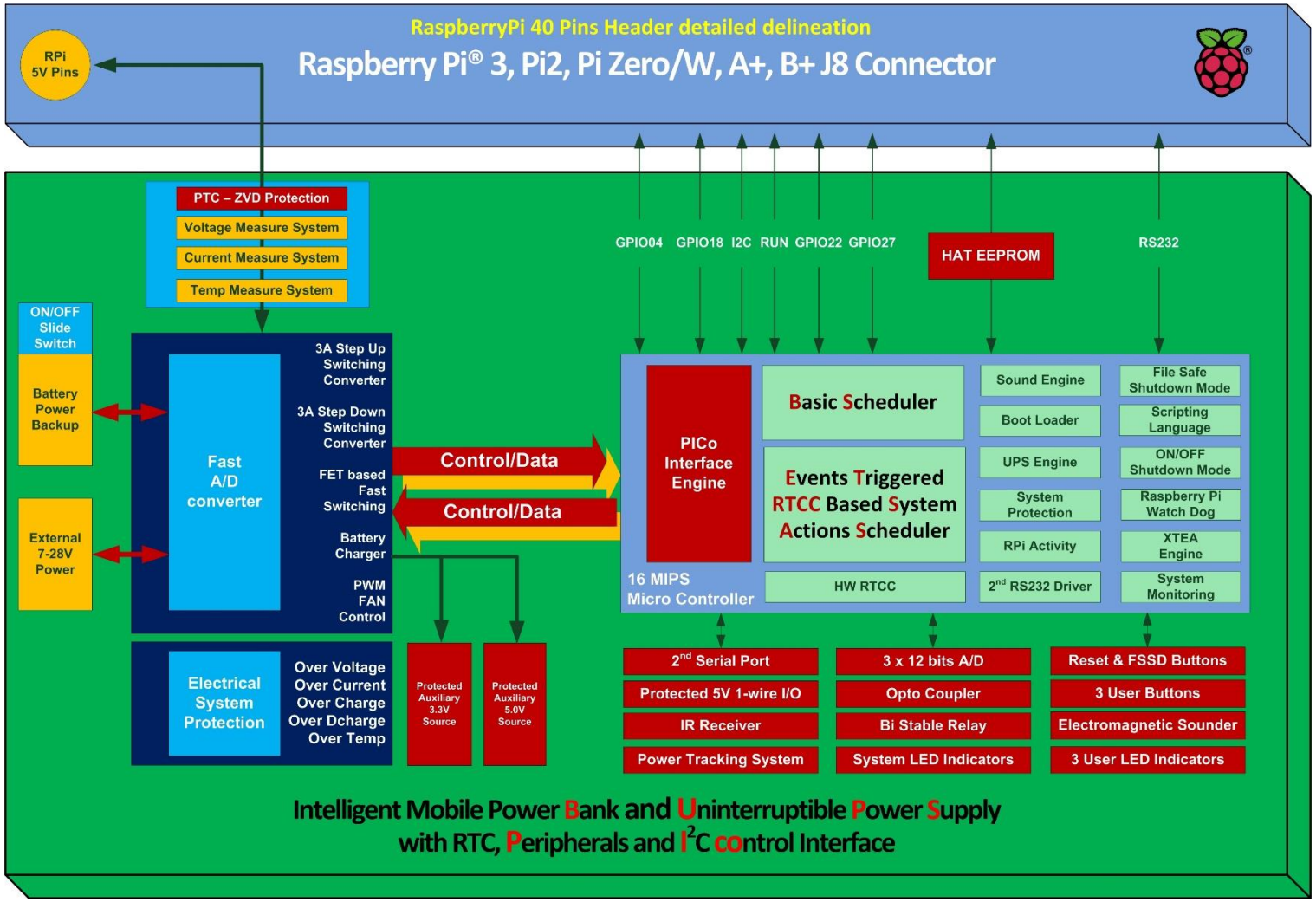


Figure 1 UPS PIco HV3.0 Simplified Block Diagram

Setting up Procedure

What is in the BOX?

This package comes with everything you need to start using the **UPS PIco HV3.0A/B/B+ HAT** right out of the box. It is assembled, tested and contains all required accessories. A little work is necessary to setup the complete Raspberry Pi® and **UPS PIco HV3.0A/B/B+ HAT** in a single full operating system, and this is instructed below. Each Box contains the following parts:

Model / Parts	UPS PIco HV3.0A/B/B+ HAT Stack 450	UPS PIco HV3.0A/B/B+ HAT Plus/Advanced 450	UPS PIco HV3.0A/B/B+ HAT Top-End 450
Core Pico HV3.0 HAT PCB	1	1	1
40 Pin THT header	1 x Stack (long pins)	1 x Stack (long pins)	1 x Top-End (short pins)
Dual layer wide temperature adhesive tape (used for battery mounting)	1	1	1
Set of spacers (plastic spacers, rubber stick, or screws and plastic spacer tubes, depending of production lot)	1 set of dedicated to high profile case mounting	1 set of dedicated to high profile case mounting	1 set of dedicated to low profile case mounting
Ultra-high current LiPO battery 450 mAh, with 6A current draw	1 pc	1 pc	1pc
Gold Reset Pin (POGO pin)	1 pc 2 pcs for model HV3.0B+	1 pc 2 pcs for model HV3.0B+	1 pc 2 pcs for model HV3.0B+
Electromagnetic Sounder	1 x High Profile	1 x High Profile	1 x Low Profile
2 x 8 pins Headers Strips	1 set	1 set	1 set
ON/OFF Slide Switch (only UPS PIco HV3.0B/B+)	1 pc	1 pc	1 pc
1.27mm HAT WP EEPROM Jumper, available in versions manufactured after 15th of February 2018 (only UPS PIco HV3.0B/B+)	1 pc	1 pc	1 pc
2mm Jumper used when Pico HV3.0B+ is configured for former (non B+) Raspberry Pi 3, (only UPS PIco HV3.0B+)	1 pc	1 pc	1 pc
Terminal Blocks (5 way) 2.54 pith	none	1	none
Bi-Stable Relay	none	1	none

Please kindly notice that, due to shipping regulations, LiPO batteries are packed in the same box but are physically or electrically separated (disconnected) and not connected to the **UPS PIco HV3.0A/B/B+ HAT** module. It must be connected by the user, and it is a part of the installation procedure.

Some few parts need to be assembled (soldered), it is extremely easy to be done by the end user himself. However, our e-shop (www.pimodules.com) is offering in addition the assembly service with a very low just symbolic price, for customer that are not equipped with soldering tools or do not have a proper soldering skill.

Differences between UPS PIco HV3.0A and HV3.0B, HV3.0B+ HATs

Following market request and rapid Raspberry Pi new models releasing, our company was obligated to manufacture a new versions of the **UPS Pico HV3.0A HAT** called **UPS Pico HV3.0B HAT** and then the **UPS Pico HV3.0B+ HAT**. Both new HATs are similar and are handled by the same firmware. The firmware loaded to the PCB is internally recognized (via bootloader) on which version of hardware is running, so any extra firmware features are automatically activated if valid (existing) to each board. Like the former HV3.0A the new version HV3.0B/B+ Stack, Top-End and Plus (now called Advanced) are based on the same PCB. Both versions the HV3.0A and the new HV3.0B/B+ are identical in mechanical dimensions as also in LEDs/Buttons placements, so the version HV3.0B/B+ can easy replace the old HV3.0A if a product has been designed based on it without any changes in the design. The UPS Pico HV3.0B/B+ are just better than the former one. Here below is a list of changes/differences of these three versions listed here below as added/implemented in the new HV3.0B/B+:

- Much better designed PCB, with more focused to lower noise, higher current, much cooler
- Much better designed PCB, with much more extended multilayer copper thermal pipes (mainly based on 2oz external copper PCB)
- Much better quality of push buttons used on the board, with legs
- Very low quiescence current 3.3V LDO (less than 1uA), used to supply PIco micro controller when sleeping
- Changed from 3.3V 1-wire to the true 5V ESD protected 1-wire interface
- The second NTC existing on the HV3.0A Plus version controlling the battery charger temperatures thresholds has been removed and his functionality has been passed to the firmware (valid for version HV3.0B/B+ Plus/Advanced only)
- Added jumper for HAT EEPROM WP control 1.27mm (in former boards HV3.0A it has been controlled by firmware)
- Added 2mm pitch jumper that allows assembly of the HV3.0B+ for any former model of Raspberry Pi
- Added additional God Plated Pin (POGO) to the HV3.0B+, that use the PEN functionality of the Raspberry Pi 3 Models B+
- Added pass through holes no the HV3.0B+, that use the PoE functionality of the Raspberry Pi 3 Models B+ if original PoE PCB is used
- Added a dedicated 2mm pitch header for selected I/O:
    a. USER LED 1
    b. USER LED 2

c. USER LED 3

That allows to be connected in parallel external LEDs to the system, just with simple cables

- Added a dedicated 2mm pitch header for selected I/O:
    a. KEY FSSD
    b. USER KEY A
    c. USER KEY B
    d. USER KEY C

That allows to be connected in parallel external KEYs to the system on a single flat cable together with external LEDs

- On UPS Pico HV3.0B added an additional bi-stable relay position. The result now is to have two available soldering positions:
    a. bi-stable relay with two galvanic isolated independent contacts DPDT 1A/30V
    b. bi-stable relay with single high current contacts SPDT 2A/30V. The PCB tracks are now wider and able to handle 2A current without any problem

- On UPS Pico HV3.0B+ changed the bi-stable relay position and type, to stronger one with two galvanic isolated independent contacts DPDT 2A/30V each

- Redesigned the High Voltage Buck Converter (7-28VDC), with special consideration for the lower noise and lower working temperature (valid for version HV3.0B/B+ Plus/Advanced only). Added a very advanced multilayer passive thermal pipes system, in order to decrease the system total temperature originated from it.

- Redesigned the **High Voltage Buck Converter** (7-28VDC) able to supply with **3.5A** supply, used different IC (valid for version HV3.0B+ Plus only)

- Added extra circuit allowing charging the battery in case when Raspberry Pi® is not powered (valid for version HV3.0HB/B+ Plus/Advanced only)

- Added **Magic Slide Switch**. A revolutionary functionality that allows to use the **UPS Pico HV3.0B/B+ HAT on mobile applications**, without cable powering and power cycling, just with battery – as an intelligent power bank

- Additionally, the **Magic Slide Switch** on the version **HV3.0B+** using the Raspberry Pi 3 Models B+ PEN is able to cut the power from the system even if micro USB cable is still connected (always making Files Safe Shut Down)

- Changed the HOT LED to INF LED (with slightly different/improved functionality)

- Added the **Remote Bootloader**, feature that allows to upload a new firmware remotely even if the uploading failed. System is (based on Raspberry Pi® and UPS

Pico) is power backed up and ready for firmware upload (if activated) until a proper upload is made. It is very usefully feature for remote located systems, where human access is not possible or very difficult.

It is important to notice that all (Stack, Top-End, and Plus) versions of HV3.0A will be no longer manufactured as it is replaced by the HV3.0B/B+ which is 100% backward compatible with the former one (mechanically and in firmware).

Hardware Setup for the UPS PIco HV3.0A/B/B+ HAT Stack/Top-End/Plus/Advanced

All **UPS PIco HV3.0 HAT** modules (each version separately A, B and B+) are based on the same PCB and differ only on assembly options. Therefore, users should know that on each board some components are missing or replaced by another one depending to the version (Stack, Top-End or Plus/Advanced)

The differences between Version Stack and Top-End are in two points:

- the THT connector does not contain Up-Standing Pins on the version Top-End
- and the Buzzer is ultra-low profile to be able to fit-in to the Official Raspberry Pi case

The differences between Version Stack/Top-End and Plus/Advanced are mainly in the following points:

- The latching (Bi-Stable) relay is offered as a standard in the version Plus/Advanced (need to be ordered optionally for the versions Stack/Top-End)
- The High Voltage Supply 7-28 VDC is present only in the version Plus/Advanced (the HV3.0B offer up to 3A, and the HV3.0B+ offers up to 3.5A).
- The battery charger IC is much more advanced, programmable and with charging variable and higher current support in all versions Plus/Advanced
- The opto-coupler is offered as a standard in the version Plus, and not offered in the others

On each of **UPS PIco HV3.0A/B/B+** are plenty of I/Os, and other User Interfaces (Keys, Sounder, etc). The below pictures show each version I/Os.

Hardware Interfacing/Interaction with Raspberry Pi®

The **UPS PIco HV3.0A/B/B+ HAT** module is plug on the top of the Raspberry Pi® micro-computer. It is using the GPIOs for interaction with it, as also dedicated software installed on the Raspberry Pi® - called Daemons. Only few GPIOs are mandatory to have system cooperative with the Raspberry Pi®, all others are optional and can be used only if needed. Detailed specifications of them are listed below.

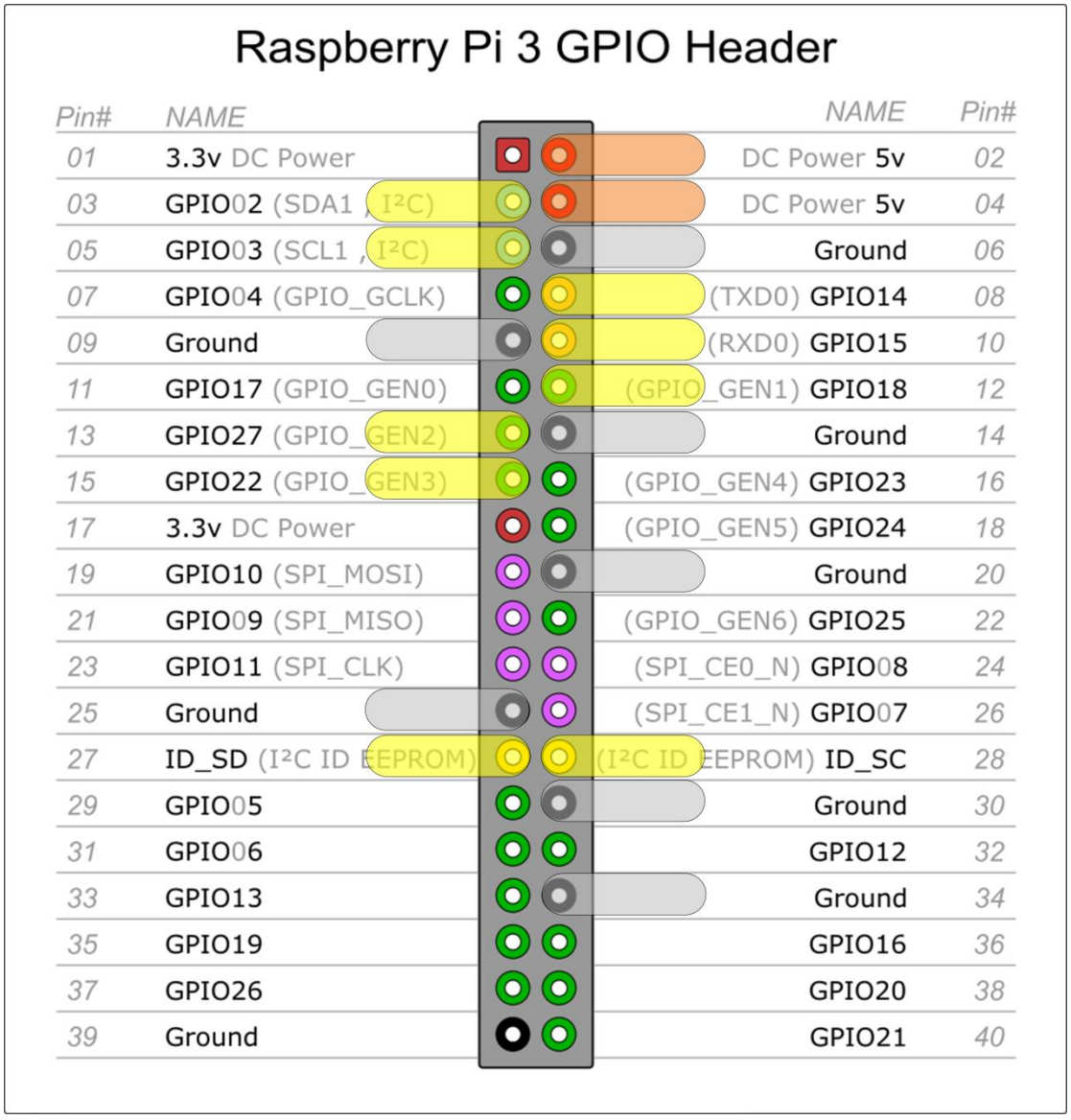


Figure 2 UPS PIco HV3.0A/B/B+ GPIO Used Pins

GPIO (Pin #)	Activity	Functionality
5V – **#02**, **#03** – marked Orange	Powering 5V	Used for monitoring of 5V and when absent the Raspberry Pi® system is powered via it. Protected by ZVD circuit and PPTC fuse of 2.6A
Ground - **#06**, **#14**, **#20**, **#30**, **#34**, **#09**, **#25** - marked Grey	System Ground	System Ground connected to the Raspberry Pi® Ground
TXD0 and RXD0 – GPIO14 and GPIO15 – **#08**, **#10** – marked Yellow	Serial Connection to Raspberry Pi ®	Used for System Monitoring, or firmware uploading – **default is HiZ** (disconnected from the GPIOs), only of user activate them are connected to the GPIOs. During boot loading process are automatically connected, and after that disconnected
IR Input – GPIO18 – **#12** – marked Yellow	Used only if IR soldered on their place on the **UPS PIco HV3.0**	Only if IR is soldered this GPIO18 is valid, all **other cases are HiZ.**
ID_SC – **#28** – marked Yellow	Used for the HAT EEPROM	Used for the HAT EEPROM
I^2C – SDA – GPIO02 Used as I2C SDA – **#03** – marked as Yellow	Used as I2C SDA	Used as I2C SDA for communication with Raspberry Pi®
I^2C – SCL – GPIO03 Used as I2C SDA – **#05** – marked as Yellow	Used as I2C SCL	Used as I2C SCL for communication with Raspberry Pi®
GPIO27 – **#13** – marked as Yellow	Used as Pulse Train send by **UPS PIco HV3.0** to the Raspberry Pi®	Used as Pulse train to fire Daemons Interrupt in the Raspberry Pi®. This functionality allows PIco to recognize if Raspberry Pi is shutting down, or running properly
GPIO22– **#15** – marked as Yellow	Used as Pulse Train Response from the Raspberry Pi® to the **UPS PIco HV3.0**	Used as Pulse train to fire Interrupts in the UPS **PIco HV3.**0 confirming response of the Raspberry Pi®, as also to shut down the Raspberry Pi®.
ID_SD – **#27** – marked Yellow	Used for the HAT EEPROM	Used for the HAT EEPROM

Table 4 UPS PIco HV3.0A/B/B+ GPIO Usage

UPS Pico HV3.0 Stack, Top-end and Plus/Advanced PCB Interfaces

UPS Pico HV3.0A

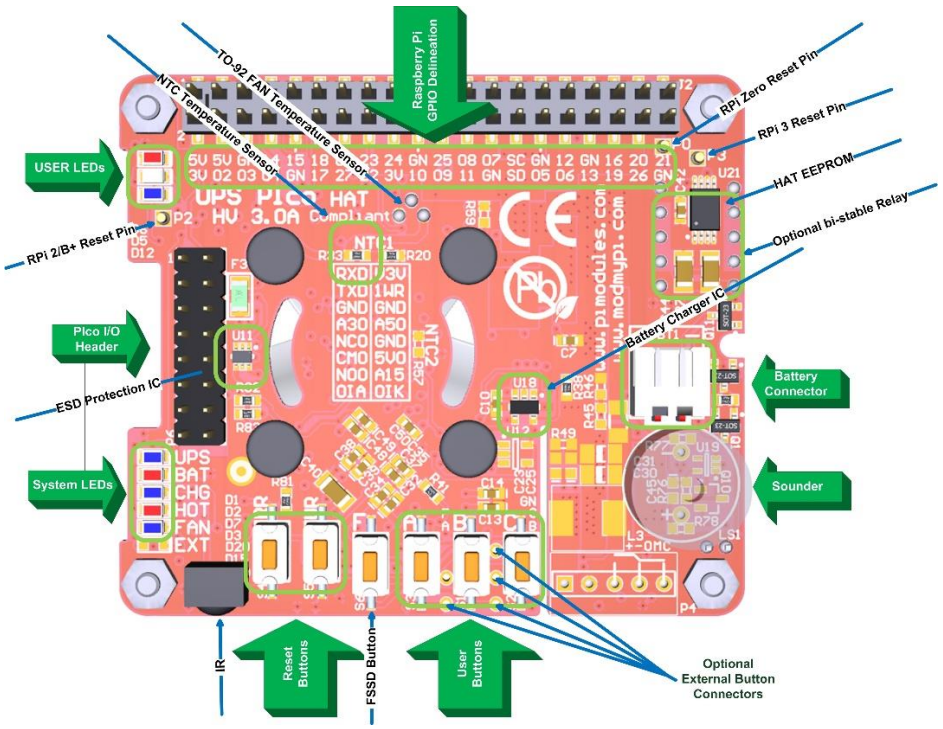


Figure 3 UPS PIco HV3.0A Stack/Top-End

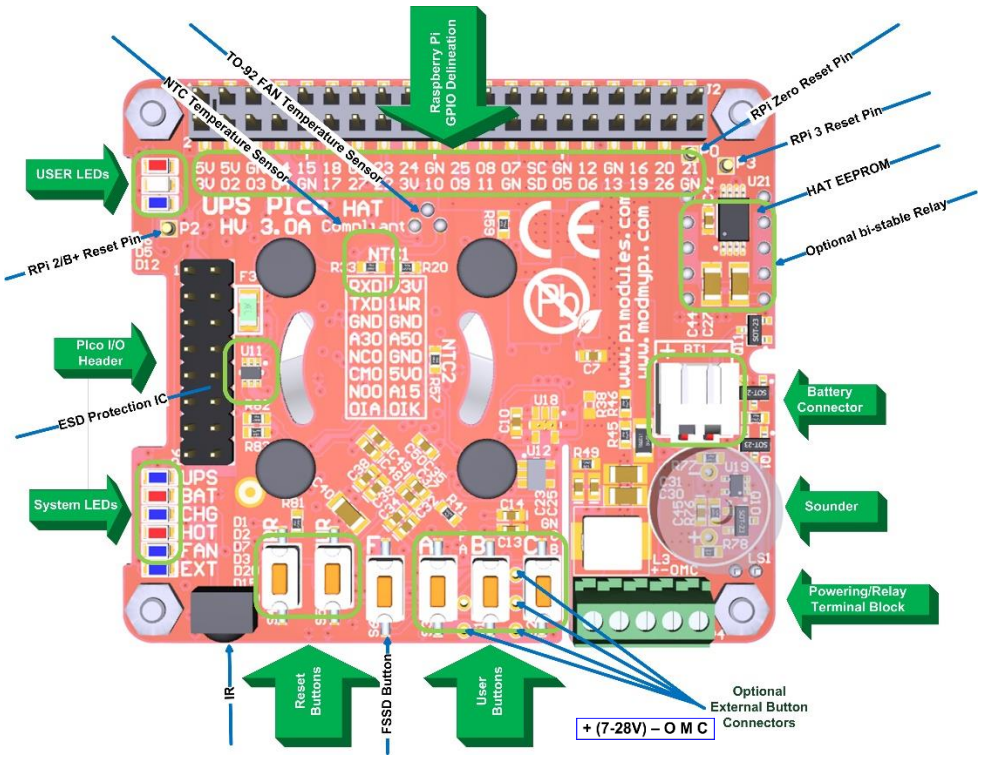


Figure 4 UPS PIco HV3.0A Plus

Interface	Name on PCB	Functionality
40 Pin SMD Connector	**J2** (Black one placed on	Used for Pass Through the Stack or Top End Connector.

with delineation	the bottom side)	Delineation helps users to find a proper GPIO if needed
User LEDs	None - (just 3 LEDs) placed on the left-up corner of PCB	3 color LEDs (Blue, Red, Green) accessed via I2C used for user applications
Gold Pin (POGO pin)	**P0**, **P2**, **P3**	Used for hardware reset of the Raspberry Pi®, each place is specified by the number, therefore: P0 – means Raspberry Pi® ZERO P2 – means Raspberry Pi® 2 P3 – means Raspberry Pi® 3
On Board Temperature Sensor 1	**NTC 1**	Used for PCB temperature measure, as also as an indicator of the environment temperature
On Board Temperature Sensor 2	**NTC 2**	Used by Battery Charger to control the charging process automatically (only in version Plus)
PIco I/O 16 pin (2x8) header	none	Used for various I/O handled by **UPS PIco HV3.0**, detailed described in next chapters
Bi-stable Relay	None - (right up corner – on bottom side)	Bi-stable Relay soldered on bottom, used only for various user applications
Battery Connector	**BAT1**	Battery connector, here should be plug in the battery (any type or size)
System LEDs	**UPS**, **BAT**, **CHG**, **HOT**, **FAN**, **EXT**	System LEDs used by **UPS PIco HV3.0** for messaging to the user on various conditions. Detailed described on next chapters
Sounder	None (inside of circle, just marked '**+**' and '**-**' for soldering)	Used for Sound Generation on various **UPS PIco HV3.0** conditions or user applications
Infra-Red Receiver	**IR** U4	If soldered, then interface the Raspberry Pi® with IR receiver, used for the any IR application
Hardware Reset Buttons	Buttons **RR** and **UR**	Hardware Reset Buttons: **RR** – **R**aspberry Pi® Hardware **R**eset **UR** – **U**PS PIco HV3.0 hardware **R**eset
FSSD Button	Button **F**	**F**ile **S**afe **S**hut **D**own **B**utton – detailed description is in next chapters
User Application Buttons	Buttons **A**, **B**, **C**	Buttons used for User Applications
Cable extensions for the listed Buttons	Holes (THT pads): **F**, **A**, **B**, **C**, **G**	Used for cable extensions of the following buttons if user like to have external buttons i.e. screwed externally on the case: F – FSSD A – Button B – Button C – Button G – GND for buttons
Extended Power Supply (7-28)	**+**, **GND**	Extended Power Supply (7-28) for version UPS PIco HV3.0 Plus
Bi-Stable Relay contacts 1st set	**O**, **M**, **C**	Contacts for the Bi-Stable Relay (1st set) O – Opened when Relay is Reset M – Common C – Closed when Relay is Reset
Connector for the FAN	**LS1**	Used to connect FAN when mounted the FAN kit (placed on bottom)

Table 5 UPS PIco HV3.0A HAT Interfaces

UPS Pico HV3.0B

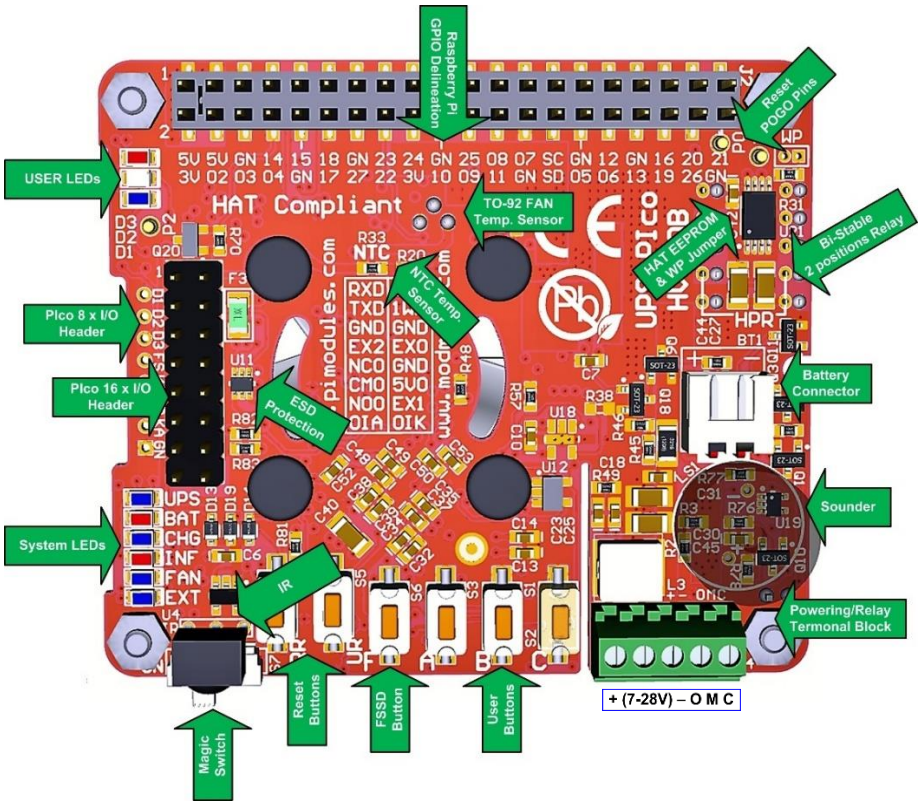


Figure 5 UPS PIco HV3.0B Stack/Top-End/Plus - Top Side

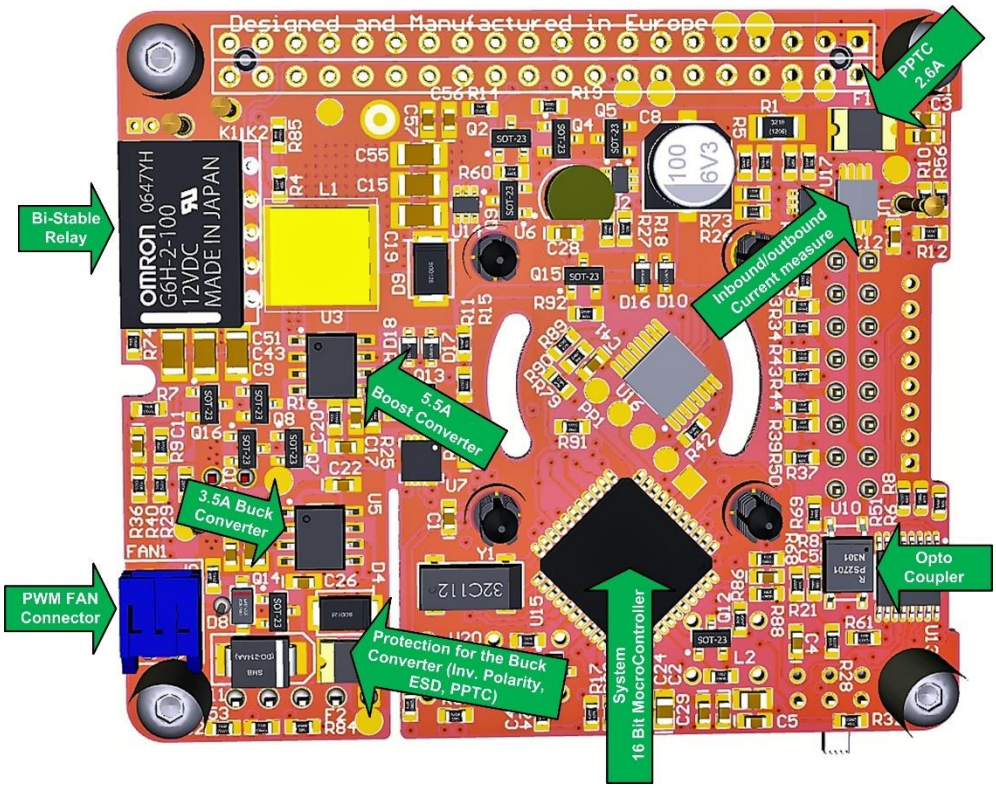


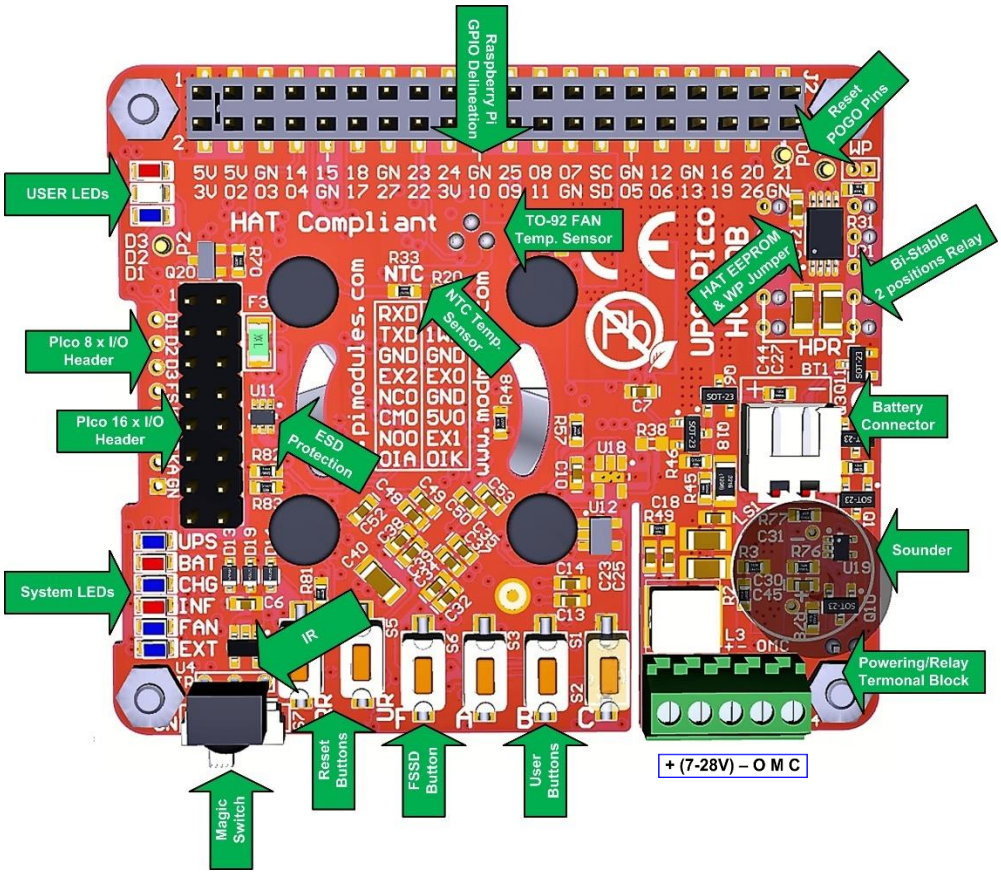
Figure 6 UPS PIco HV3.0B Stack/Top-End/Plus - Bottom Side

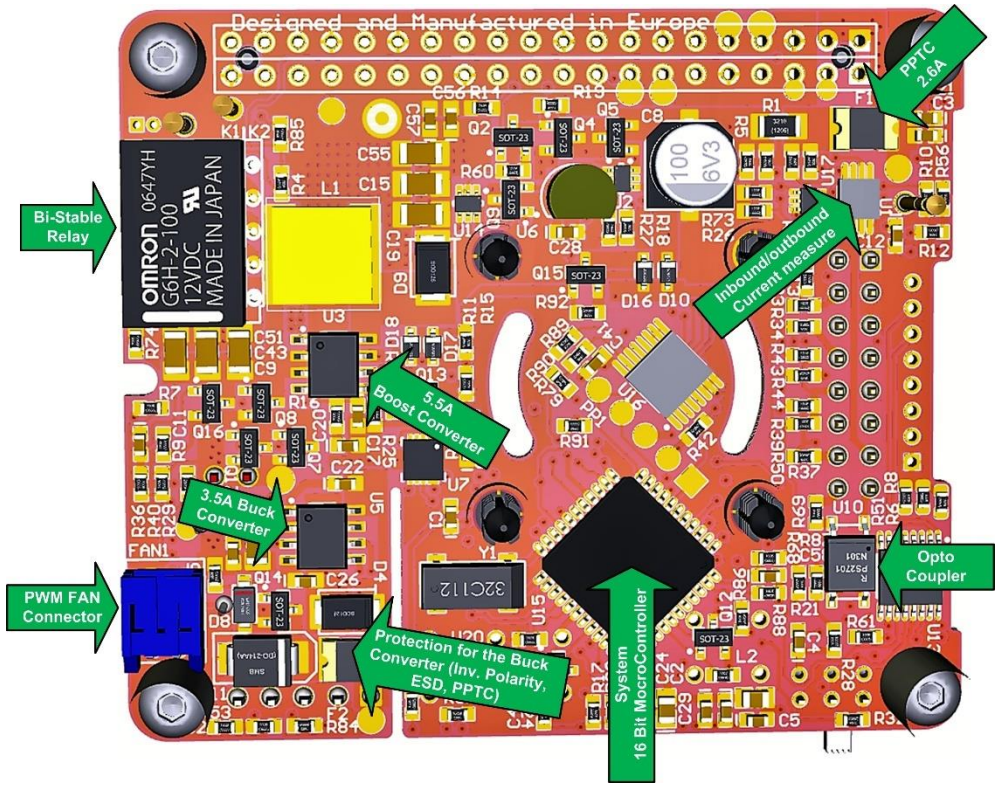
Interface	Name on PCB	Functionality
40 Pin SMD Connector with delineation	**J2** (Black one placed on the bottom side)	Used for Pass Through the Stack or Top End Connector. Delineation helps users to find a proper GPIO if needed
User LEDs	None - (just 3 LEDs) placed on the left-up corner of PCB	3 color LEDs (Blue, Red, Green) accessed via I2C used for user applications
Gold Pin (POGO pin)	**P0**, **P2**, **P3**	Used for hardware reset of the Raspberry Pi®, each place is specified by the number, therefore: P0 – means Raspberry Pi® ZERO/W P2 – means Raspberry Pi® 2 P3 – means Raspberry Pi® 3
On Board Temperature Sensor 1	**NTC**	Used for PCB temperature measure, as also as an indicator of the environment temperature
Pico I/O 8 pin header	none	Used for cable extensions of the User LEDs if user like to have them outside lighting i.e. screwed externally on the case: D1 – BLUE D2 – GREEN D3 – RED Used for cable extensions of the following buttons if user like to have external buttons i.e. screwed externally on the case: FS – FSSD KA – Button KB – Button KC – Button GN – GND for buttons
PIco I/O 16 pin (2x8) header	none	Used for various I/O handled by **UPS PIco HV3.0**, detailed described in next chapters
Bi-stable Relay and HPR	None or HPR	Bi-stable Relay soldered on bottom, used only for various user applications. Bi-stable Relay can be soldered on two positions: DPDT 1A/30V and SPDT 2A/30V (marked as HPR)
Battery Connector	**BAT1**	Battery connector, here should be plug in the battery (any type or size)
System LEDs	**UPS**, **BAT**, **CHG**, **INF**, **FAN**, **EXT**	System LEDs used by **UPS PIco HV3.0** for messaging to the user on various conditions. Detailed described on next chapters
Sounder	None (inside of circle, just marked '**+**' and '**-**' **for** soldering)	Used for Sound Generation on various **UPS PIco HV3.0** conditions or user applications
Infra-Red Receiver	**IR** U4	If soldered, then interface the Raspberry Pi® with IR receiver, used for the any IR application, connected directly to GPIO18
Hardware Reset Buttons	Buttons **RR** and **UR**	Hardware Reset Buttons: **RR** – **R**aspberry Pi® Hardware **R**eset **UR** – **U**PS PIco HV3.0 hardware **R**eset
FSSD Button	Button **F**	**F**ile **S**afe **S**hut **D**own **B**utton – detailed description is in next chapters
User Application Buttons	Buttons **A**, **B**, **C**	Buttons used for User Applications

Extended Power Supply (7-28)	**+, GND**	Extended Power Supply (7-28) for version UPS PIco HV3.0 Plus
Bi-Stable Relay contacts 1st Position	**O, M, C on Terminal Block and on the 16-pin header** Position **K1**	Contacts for the Bi-Stable Relay (1st set) O – Opened when Relay is Reset M – Common C – Closed when Relay is Reset bi-stable relay with two galvanic isolated independent contacts DPDT 1A/30V
Bi-Stable Relay contacts 2nd Position (High PoweR)	**O, M, C on Terminal Block Only** Position **K2**	Contacts for the Bi-Stable Relay (2nd set) O – Opened when Relay is Reset M – Common C – Closed when Relay is Reset bi-stable relay with single high current contacts SPDT 2A/30V.
Magic Slide ON/OFF Switch	**S7**	Used for "on the Go" ON/OFF applications when system is powered as Intelligent Power Bank
Connector for the FAN	**LS1**	Used to connect FAN when mounted the FAN kit (placed on bottom)

Table 6 UPS PIco HV3.0B HAT Interfaces

UPS Pico HV3.0B+





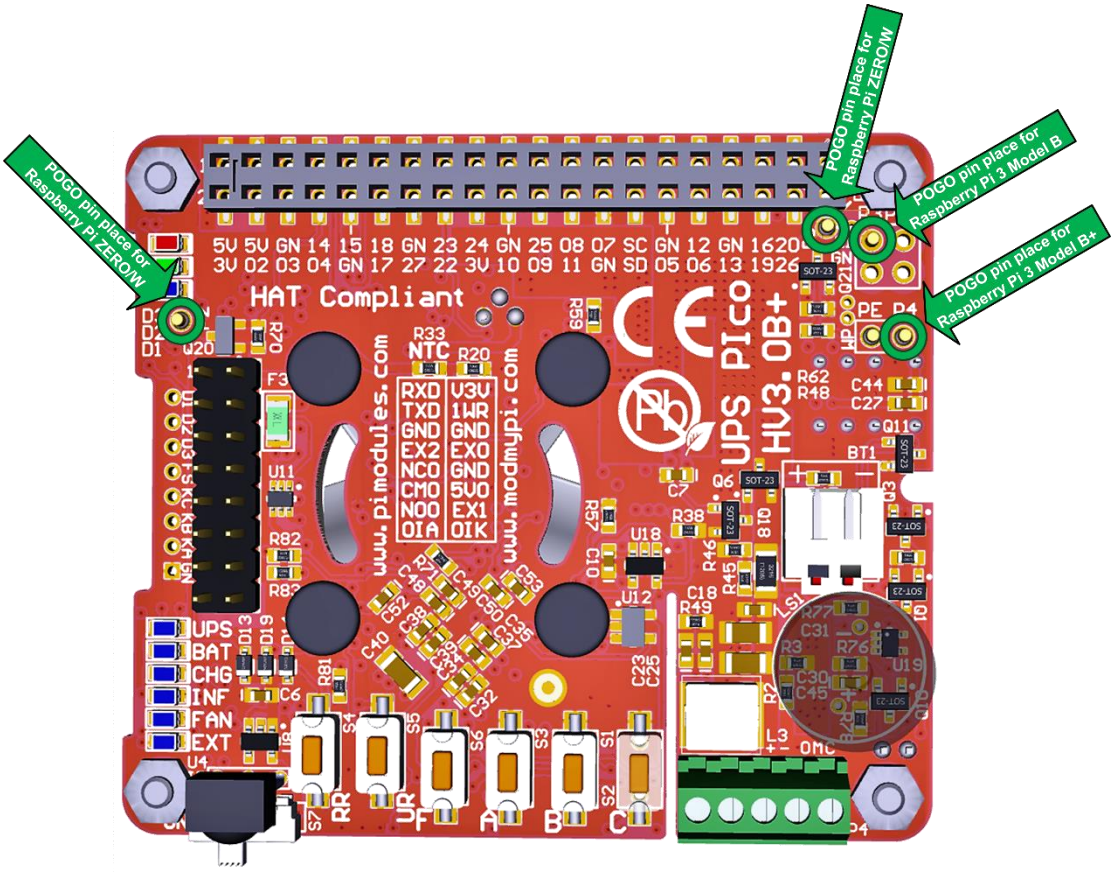
Interface	Name on PCB	Functionality
40 Pin SMD Connector with delineation	**J2** (Black one placed on the bottom side)	Used for Pass Through the Stack or Top End Connector. Delineation helps users to find a proper GPIO if needed
User LEDs	None - (just 3 LEDs) placed on the left-up corner of PCB	3 color LEDs (Blue, Red, Green) accessed via I2C used for user applications
Gold Pin (POGO pin)	**P0**, **P2**, **P3**	Used for hardware reset of the Raspberry Pi®, each place is specified by the number, therefore: P0 – means Raspberry Pi® ZERO/W P2 – means Raspberry Pi® 2 P3 – means Raspberry Pi® 3
On Board Temperature Sensor 1	**NTC**	Used for PCB temperature measure, as also as an indicator of the environment temperature
Pico I/O 8 pin header	none	Used for cable extensions of the User LEDs if user like to have them outside lighting i.e. screwed externally on the case: D1 – BLUE D2 – GREEN D3 – RED Used for cable extensions of the following buttons if user like to have external buttons i.e. screwed externally on the case: FS – FSSD KA – Button KB – Button KC – Button GN – GND for buttons
PIco I/O 16 pin (2x8) header	none	Used for various I/O handled by **UPS PIco HV3.0**, detailed described in next chapters
Bi-stable Relay and HPR	None or HPR	Bi-stable Relay soldered on bottom, used only for various user applications. Bi-stable Relay can be soldered on two positions: DPDT 1A/30V and SPDT 2A/30V (marked as HPR)
Battery Connector	**BAT1**	Battery connector, here should be plug in the battery (any type or size)
System LEDs	**UPS**, **BAT**, **CHG**, **INF**, **FAN**, **EXT**	System LEDs used by **UPS PIco HV3.0** for messaging to the user on various conditions. Detailed described on next chapters
Sounder	None (inside of circle, just marked '**+**' and '**-**' **for** soldering)	Used for Sound Generation on various **UPS PIco HV3.0** conditions or user applications
Infra-Red Receiver	**IR** U4	If soldered, then interface the Raspberry Pi® with IR receiver, used for the any IR application, connected directly to GPIO18
Hardware Reset Buttons	Buttons **RR** and **UR**	Hardware Reset Buttons: **RR** – **R**aspberry Pi® Hardware **R**eset **UR** – **U**PS PIco HV3.0 hardware **R**eset
FSSD Button	Button **F**	**F**ile **S**afe **S**hut **D**own **B**utton – detailed description is in next chapters
User Application Buttons	Buttons **A**, **B**, **C**	Buttons used for User Applications

Extended Power Supply (7-28)	**+, GND**	Extended Power Supply (7-28) for version UPS PIco HV3.0 Plus
Bi-Stable Relay contacts 1st Position	**O, M, C on Terminal Block and on the 16-pin header** Position **K1**	Contacts for the Bi-Stable Relay (1st set) O – Opened when Relay is Reset M – Common C – Closed when Relay is Reset bi-stable relay with two galvanic isolated independent contacts DPDT 1A/30V
Bi-Stable Relay contacts 2nd Position (High PoweR)	**O, M, C on Terminal Block Only** Position **K2**	Contacts for the Bi-Stable Relay (2nd set) O – Opened when Relay is Reset M – Common C – Closed when Relay is Reset bi-stable relay with single high current contacts SPDT 2A/30V.
Magic Slide ON/OFF Switch	**S7**	Used for "on the Go" ON/OFF applications when system is powered as Intelligent Power Bank
Connector for the FAN	**LS1**	Used to connect FAN when mounted the FAN kit (placed on bottom)

Table 7 UPS PIco HV3.0B+ HAT Interfaces

Configuring UPS Pico HV3.0B+ HAT to be assembled for the Raspberry Pi 3 Models B Gold Plated Reset Pin – POGO Pin (RUN)

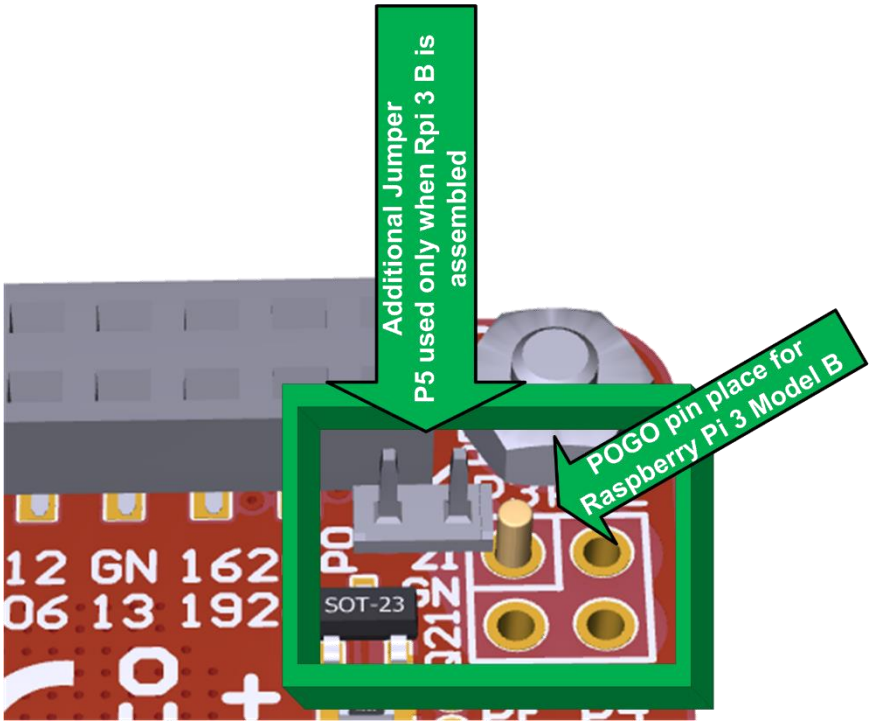
The Newest Model of the **UPS Pico HV3.0B+ HAT** has been designed especially for the **Raspberry Pi 3 Model B+.** However, it is compatible with all former models of the Raspberry Pi (those that have build with 40 pins connector) including the Raspberry Pi ZERO/W. The main difference between all models of the Raspberry Pi is the place of the **RUN** (Hardware Reset) pin. It is placed on different places on each model or Raspberry Pi. These different places of it, is mapped on various position on the UPS Pico HV3.0. This **RUN** pin is touching by the Gold-Plated Reset (POGO Pin) pin of the UPS Pico HV3.0 and activate some functionalities of the **UPS Pico HV3.0**. It is strongly recemented to use this pin in order have full functional UPS Pico HV3.0. Therefore, user need to place and solder this pin on specified by the Raspberry Pi Model place like show below picture. However, user need to take a special care if UPS Pico HV3.0B+ is assembled for the Raspberry Pi 3 Model B. This is because the Gold-Plated Reset (POGO Pin) is placed one on of the PoE pins.



Special Jumper (P5) placement on UPS Pico HV3.0B+ when Assembled for the Raspberry Pi 3 Model B

The **UPS Pico HV3.0B+ HAT** has been designed to be compatible with all former models of the Raspberry Pi. However unfortunately the position of the **RUN** pin on the **Raspberry Pi 3 Model B** is contradictory with the **PoE** pins on the **Raspberry Pi 3 Model B+**. Therefore, our

company in order to keep compatibility of the new **UPS Pico HV3.0B+ HAT** with all former models of the Raspberry Pi (specially with the Raspberry Pi 3 Model B), made a special design consideration – using a dedicated jumper, used if **UPS Pico HV3.0B+ HAT** is assembled for the **Raspberry Pi 3 Model B**. It is valid only when **UPS Pico HV3.0B+ HAT** is assembled for the **Raspberry Pi 3 Model B.** Detailed instructions how to assembly/Solder the Gold-Plated Reset Pins are in different part of this manual on page



Gold Plated Power Enable Pin – POGO Pin (PEN) for the Raspberry Pi 3 Model B+

The Raspberry Pi 3 Model B+ has implemented a unique feature Power Enable Pin – the **PEN**. This pin is placed near to the **RUN** pin. This **PEN** feature allows to switch ON/OFF the raspberry Pi 3 Model B+ even if the micro USB cable is connected. This feature is used in various functions implemented in the **UPS Pico HV3.0B+ HAT.** To use them, user need to solder the second Gold-Plated Pin on the **PE** position like show below picture. Therefore, the **UPS Pico HV3.0B+ HAT** need to have soldered 2 x Gold-Plated Pins.

UPS Pico HV3.0 Stack, Top-end and Plus/Advanced handmade components assembly

Assembly of the Magic (slide) ON/OFF Switch (UPS PIco HV3.0B/B+ HAT only)

The newer Version of **UPS PIco HV3.0B/B+ HAT** (Versions B/B+) is equipped with a **Magic (Slide) ON/OFF Switch**. This Switch is called **Magic**, as it is multifunctional, programmable, and adding a huge difference in powering schemes of the Raspberry Pi®.

First, the **Magic Switch** is not needed to be soldered. As it is designed, if this switch is not soldered, or placed on position OFF, the **UPS PIco HV3.0B/B+ HAT** acts identically as the former **UPS PIco HV3.0A HAT**. Also, in addition user can solder (using cables) an External ON/OFF Switch. As for the ON/OFF are used MOSFET based circuit, the ON/OFF feature can be activated by external Relay, PIR or any device that offers a short circuit (Arduino).

If the **Magic Switch** is used, some PICo registers need to be programmed to activate his features. Therefore, only soldering of it is not enough for a proper usage and using it without proper preparation can cause unexpected effects. A detailed programming of associated registers to it, are presented in next chapters, in section related to usage of the **UPS PIco HV3.0B/B+ HAT**.

Ensure that you have prepared the **UPS PIco HV3.0B/B+ HAT**. Please follow below instructions:

- Locate the S7 on the PCB, and be sure that it is not the U4 – IR (InfraRed Receiver)

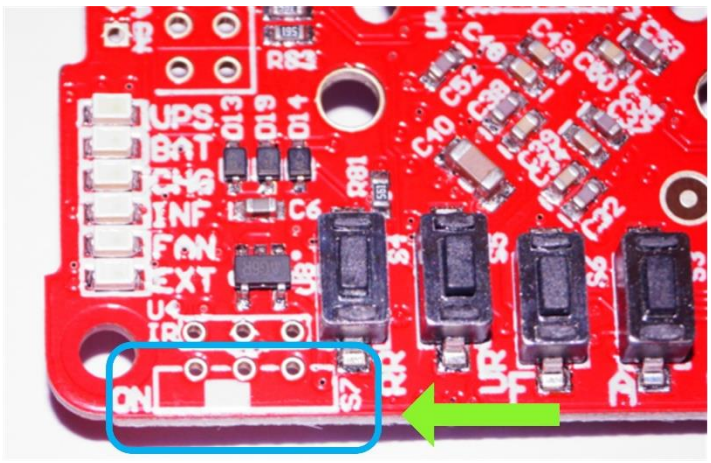


Figure 7 UPS PIco HV3.0B/B+ Magic Switch place

- If you like to use external Slide Switch solder to these 3 pads, cables

- Then solder cables to the selected External Slide Switch. There are no special requirements for the External Slide Switch regarding the switching current or voltage.

- If you like to use provided in the package micro slide switch prepare and put it to the proper place in the PCB

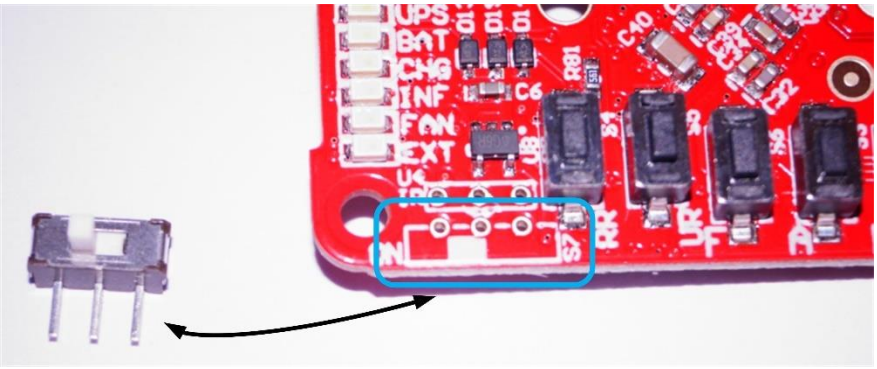


Figure 8 UPS PIco HV3.0B/B+ Magic Switch placement

- Then revert the PCB upside down and solder, and after cut the Magic Switch pins

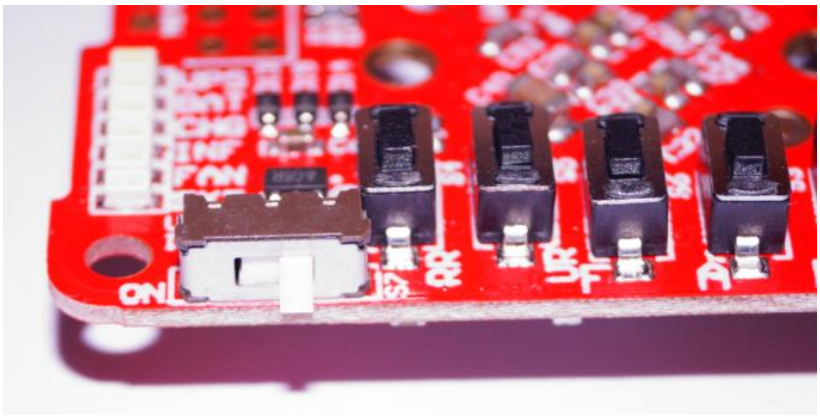


Figure 9 UPS PIco HV3.0B/B+ Magic Switch on their place

- The Magic Switch must be by default placed on the OFF position

- Any soldering of Magic Switch must be done without battery connected!!!

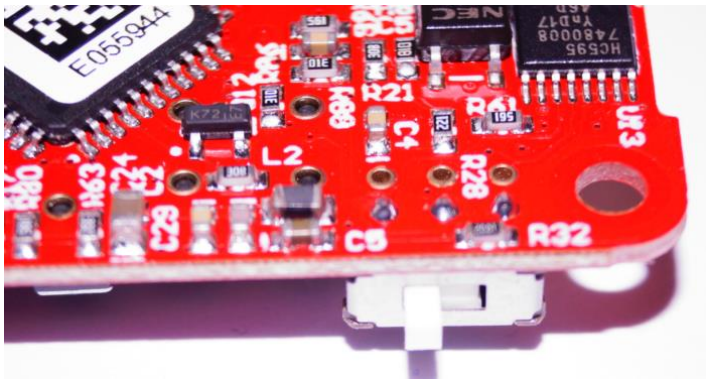


Figure 10 UPS PIco HV3.0B/B+ Magic Switch soldered

Assembly of the THT 40 Pins (2x20) connector

Ensure that you have prepared the Stack or Top-End THT Connector. To properly pass through the THT connector please follow below instructions:

- Prepare your **UPS PIco HV3.0 HAT,** and make sure that the black SMD 40 pins connector is available

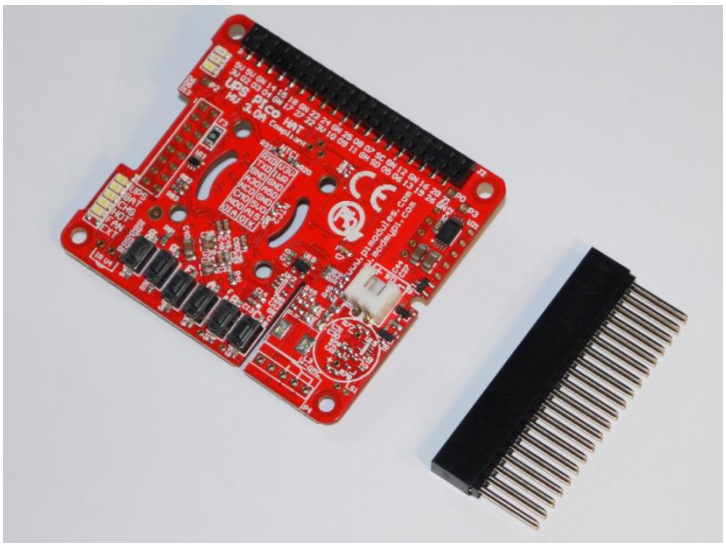


Figure 11 UPS PIco HV3.0 and 40 THT Stack Header

- Put your **UPS PIco HV3.0 HAT** upside down, and make sure that the black SMD 40 pins connector is touching the table

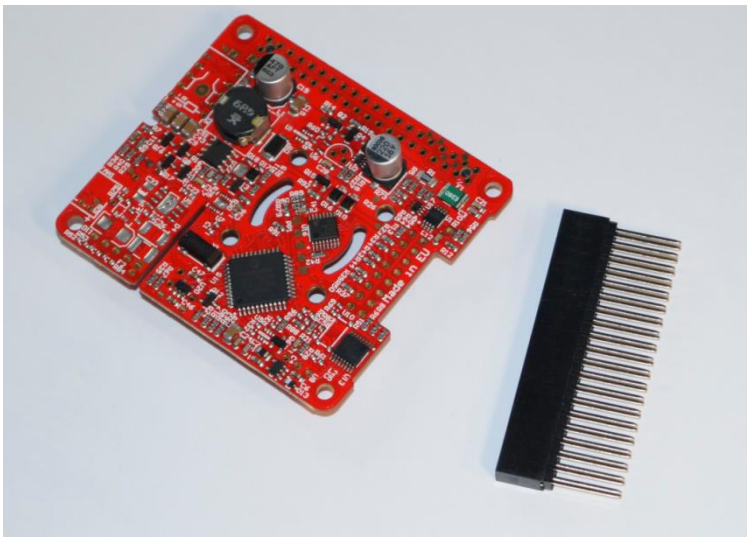


Figure 12 UPS PIco HV3.0 and 40 THT Stack Header bottom side

- Apply the THT 40 Pins connector carefully though the holes on the **UPS PIco HV3.0 HAT SMD**. Apply pressure to the connector making sure you have placed the PCB on something stable, like a table, so the connector can easily fit when it's applied with pressure.

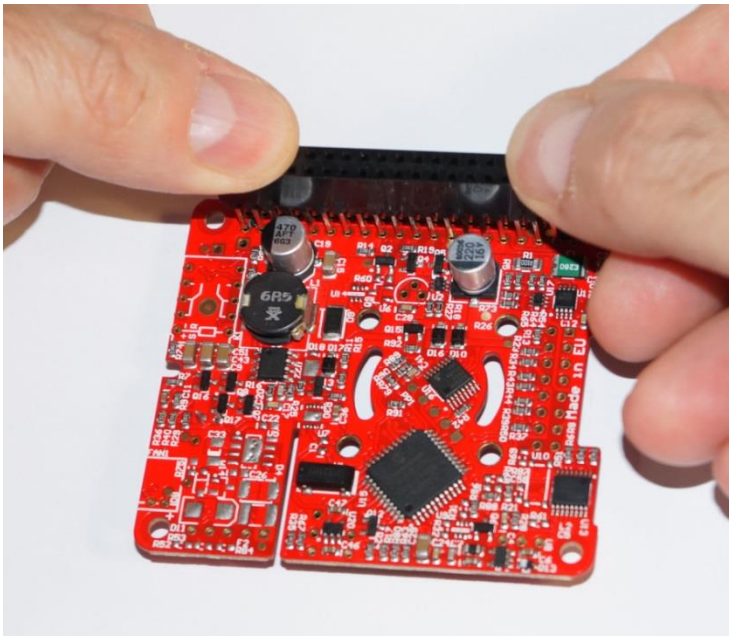


Figure 13 UPS PIco HV3.0 passing the 40 Pins THT connector

- Press the THT 40 Pins connector on the plastic side to complete pass its pins trough, until end of them reaches the bottom of the PCB

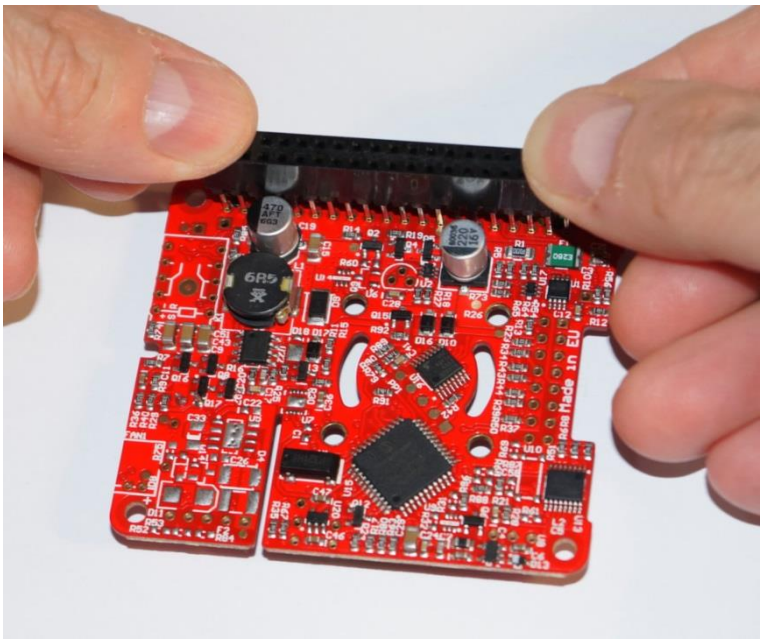


Figure 14 UPS PIco HV3.0 partially passed the 40 Pins THT connector

- Put the PCB and the semi passed connector on the opposite side (this time on the proper one) and then press the PCB on the connector side slowly and carefully until the complete pins pass through, always pressing only the SMD connector and not the PCB itself.

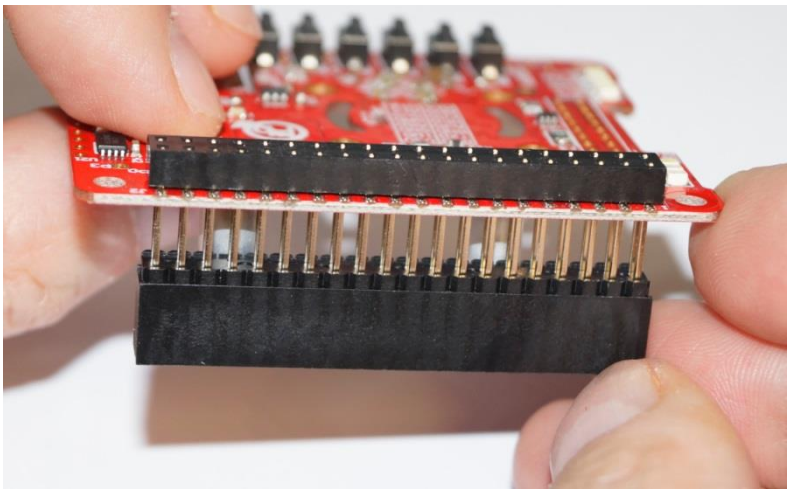


Figure 15 UPS PIco HV3.0 partially passed the 40 Pins THT connector side view

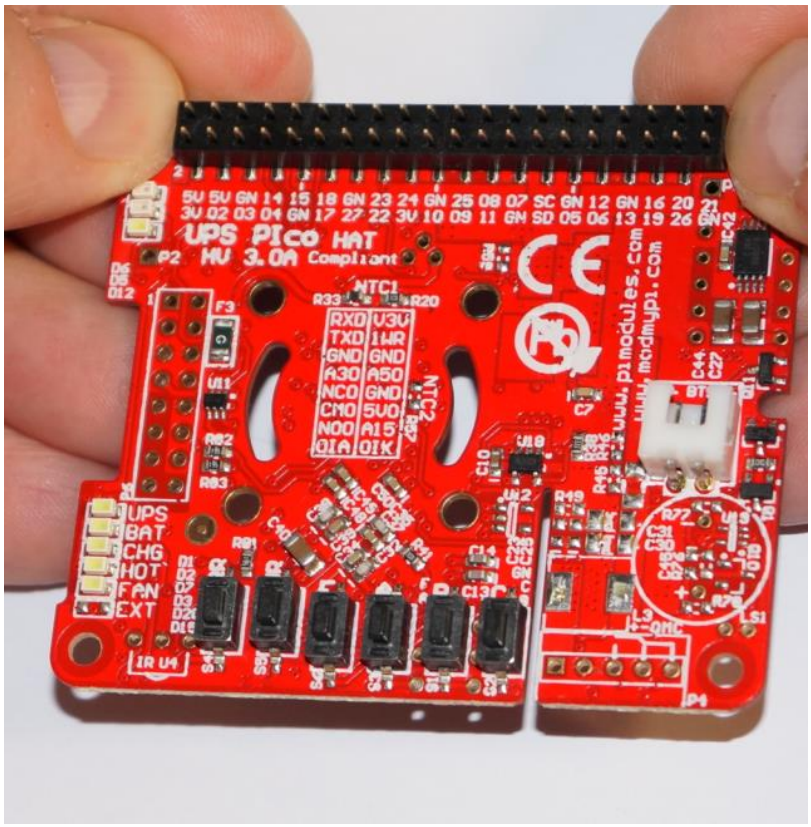


Figure 16 UPS PIco HV3.0 partially passed the 40 Pins THT connector top view

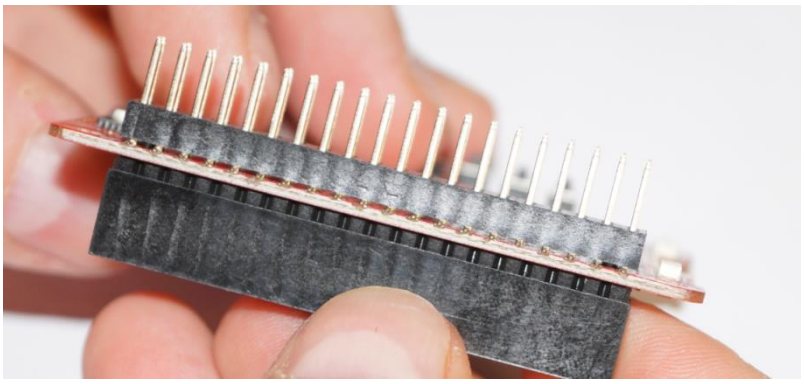


Figure 17 Figure 8 UPS PIco HV3.0 full passed the 40 Pins THT connector side view

Assembling the THT 40 pin Connector for the Top-End Version is the same, the only difference is that the top pins does not exist.

A1ssembly of the Gold-Plated Hardware Reset Pin (POGO Pin)

The **Gold Plated Hardware Reset Pin** (POGO Pin) is used to provide various additional functionalities to the **UPS PIco HV3.0 HAT**. It is not necessary, however strongly recomened as additional functionalities covered by it make the **UPS PIco HV3.0 HAT** system more co-operative. It is used with the following functionalities already implemented in the **UPS PIco HV3.0 HAT**, they are:

- Button for Hardware Reset of Raspberry Pi®
- Watch Dog ("Still Alive?") functionality - Automatically Resetting (Restarting) of the Raspberry Pi® when hung-up
- Resetting (Restarting) of the Raspberry Pi® when cable power returns during shutting down process.

There is a very simple hand work needed to solder this pin to the Raspberry Pi®

Place on your desk the **Raspberry Pi®** the **UPS PIco HV3.0 HAT** and the **Gold-Plated Reset Pin**.

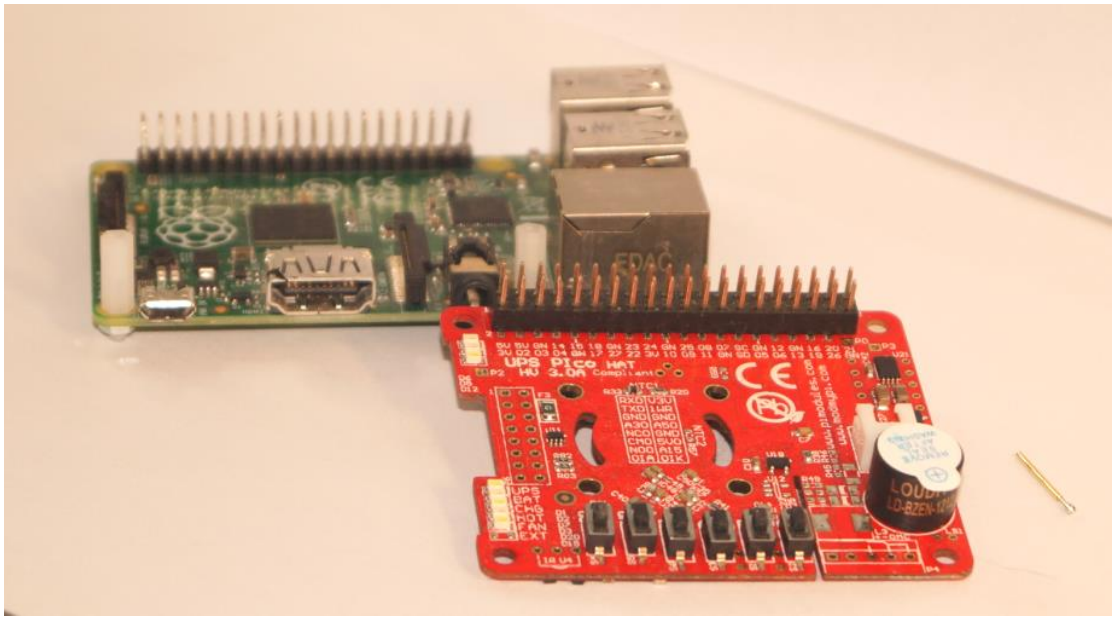


Figure 18 Raspberry Pi, UPS PIco HV3.0 HAT, and Gold-Plated Reset Pin

Drag on the **Gold Plated Reset Pin** through the hole on the **UPS PIco HV3.0 HAT** as shown in the pictures below. Take care to drag on the right direction. Select the proper hole for your Raspberry Pi ® model (P0, P2, P3) marked on the **UPS PIco HV3.0 HAT PCB** (Related to Raspberry Pi® models Zero, 2 and 3)

Drag on the **Gold Plated Reset Pin** on the direction as shown on the below picture.

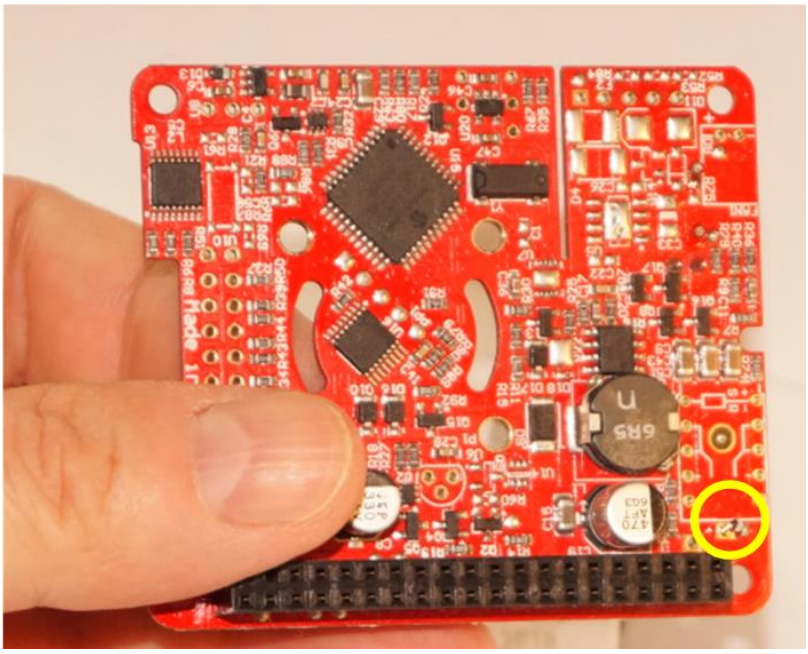


Figure 19 Gold-Plated Reset Pin on the place RPi3

Make sure to screw the poper spacers on the side where HDMI connector of the Raspberry Pi® (on the oposite side of the 40 pin connector and screw them). This will ensure that the distance between **UPS PIco HV3.0 HAT** and Raspberry Pi® is proper and provide a resistance when **UPS PIco HV3.0 HAT** buttons are pressed.

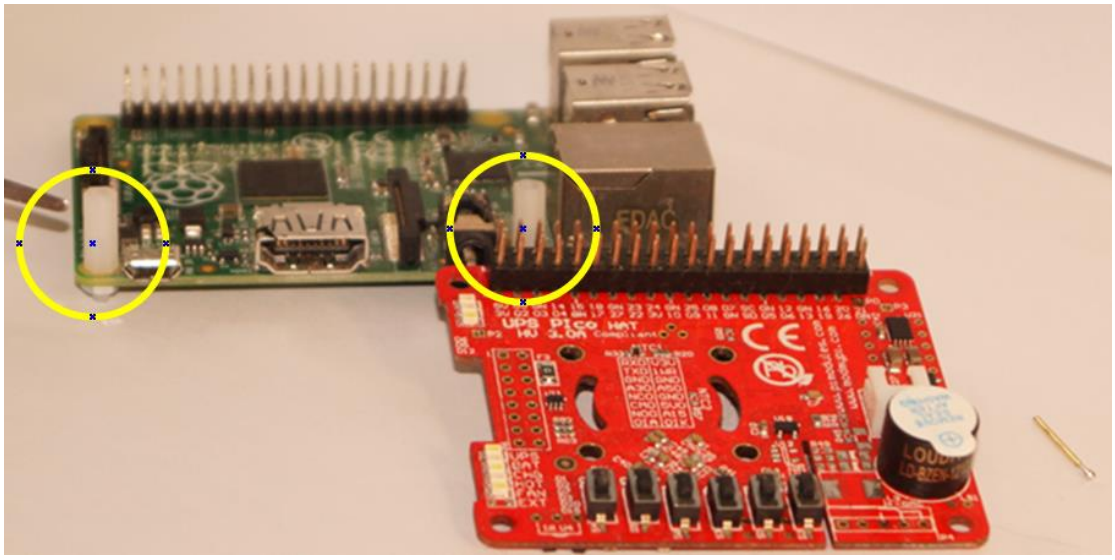


Figure 20 Spacers screwed on their places

Put the **UPS PIco HV3.0 HAT** on the Raspberry Pi®, and take care to center the head of the **Gold Plated Reset Pin** to the center of the RUN square pad hole.

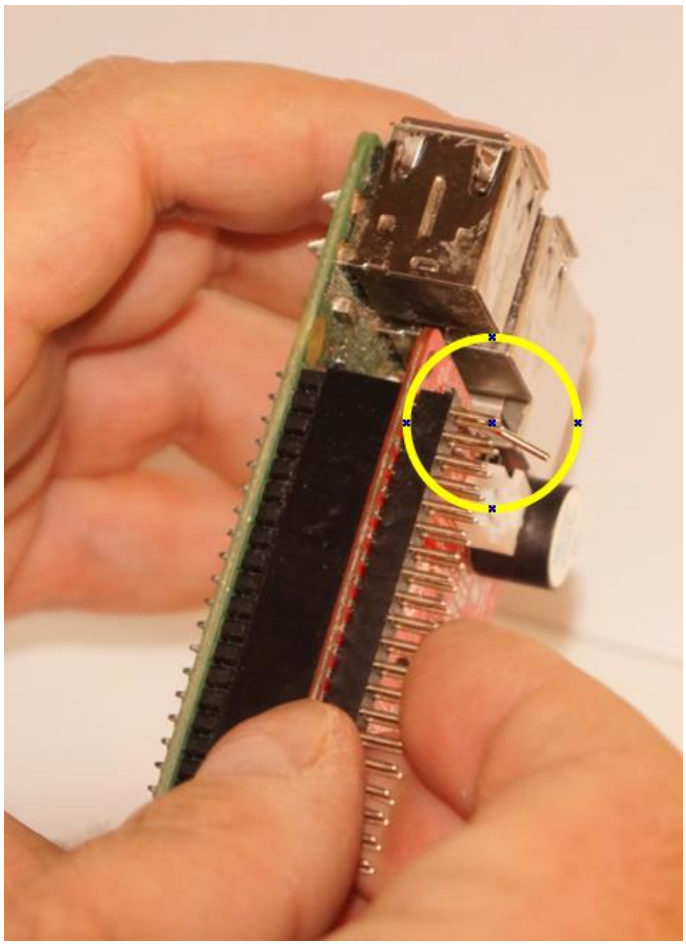


Figure 21 UPS PIco HV3.0 on the Raspberry Pi with Gold-Plated Reset Pin

Check it, by pressing the pin on the Top. Then, using a soldering tool, solder the **Gold-Plated Reset Pin** on the top of the PCB only. Take care to heat up properly the pin before you will add the tin. After soldering it will look like in the picture below. Make sure that the **Gold-Plated Reset Pin** after soldering touches properly the RUN on the Raspberry Pi®.

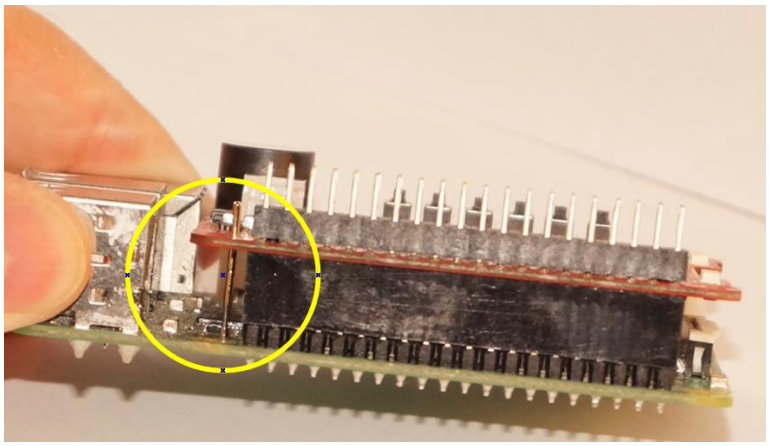


Figure 22 The Gold-Plated Reset Pin must point exactly on the hole of the RUN pad

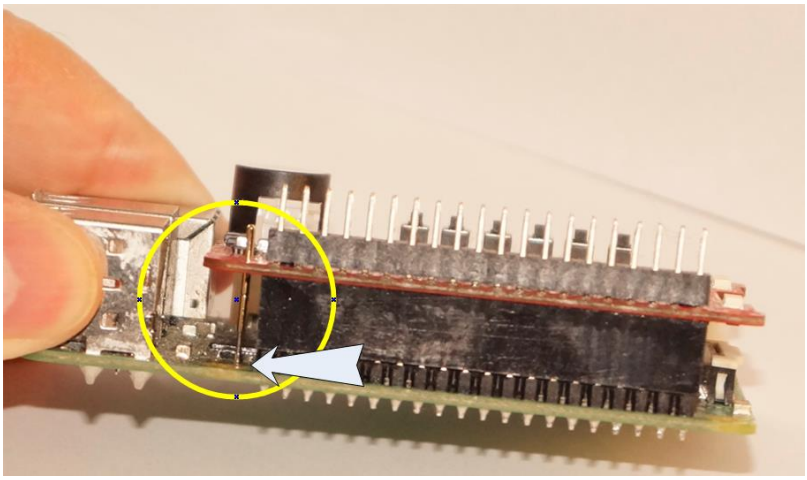


Figure 23 Pointing exactly on the hole of the RUN pad

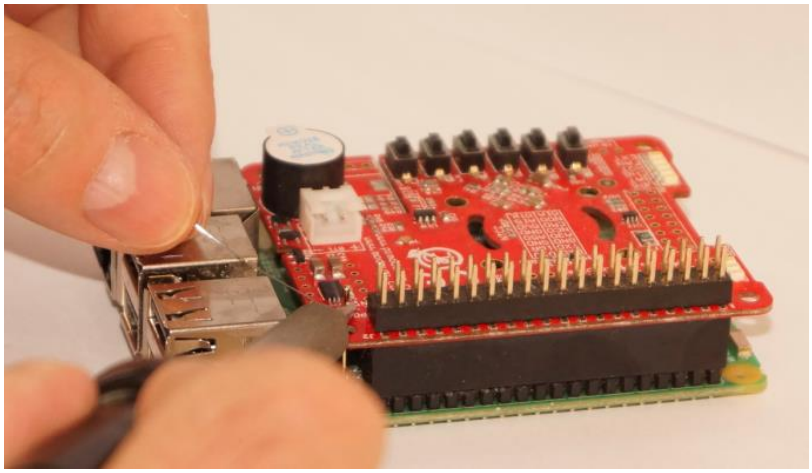


Figure 24 Soldering the Gold-Plated Reset Pin on the PIco

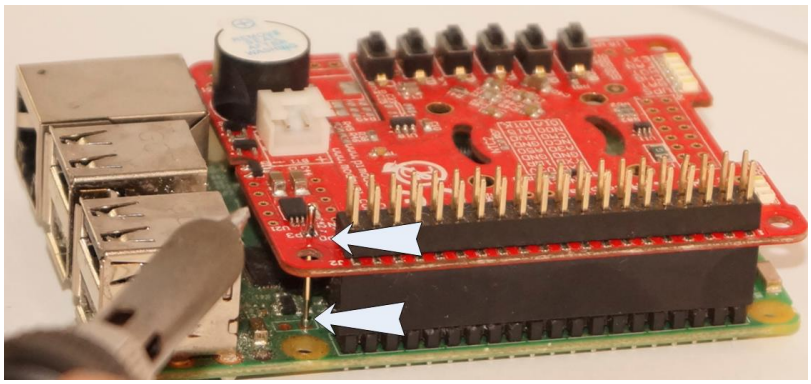


Figure 25 Soldered Gold-Plated Reset Pin is touching exactly the RUN pad

Then to make **Gold Plated Reset Pin** internal spring working, you need to re-solder it by pressing down for about 1.5 – 2 mm. Press it down with screw driver and heat up with a soldering tool. Then remove the soldering tool, keeping pressing the pin down. After about 5 seconds, you can put out the screw driver.

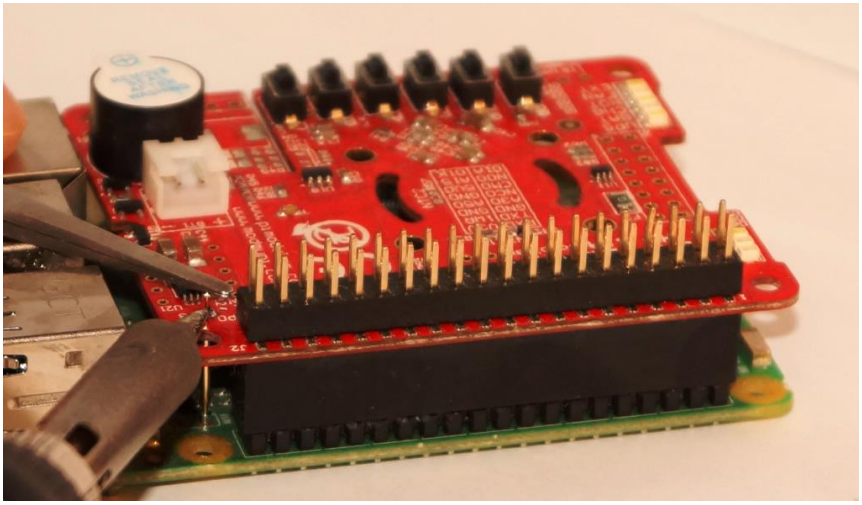


Figure 26 Heating and pressing of the Gold-Plated Reset Pin

You will make the **Gold-Plated Reset Pin** ready.

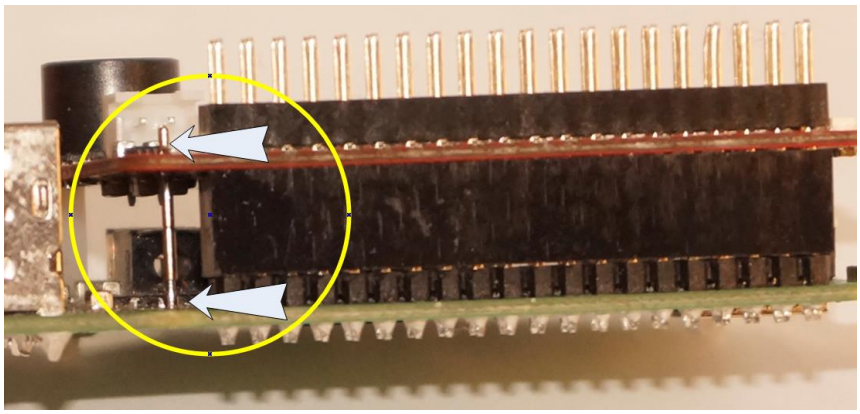


Figure 27 Gold-Plated Reset Pin properly soldered

To test it, make Raspberry Pi® working and reset it by pressing the **R** button on the **UPS PIco HV3.0 HAT.**

More advanced usage of the **Gold Plated Reset Pin** is described in another chapter.

Assembly of the Buzzer (Sounder)

If you would like to use the buzzer, you can solder it on now. Ensure that this is done with the correct polarity. Positive "**+**" on the board should match the positive "**+**" on the buzzer. The soldering procedure is same for the High and Low (used in TopEnd Version) profile buzzer.

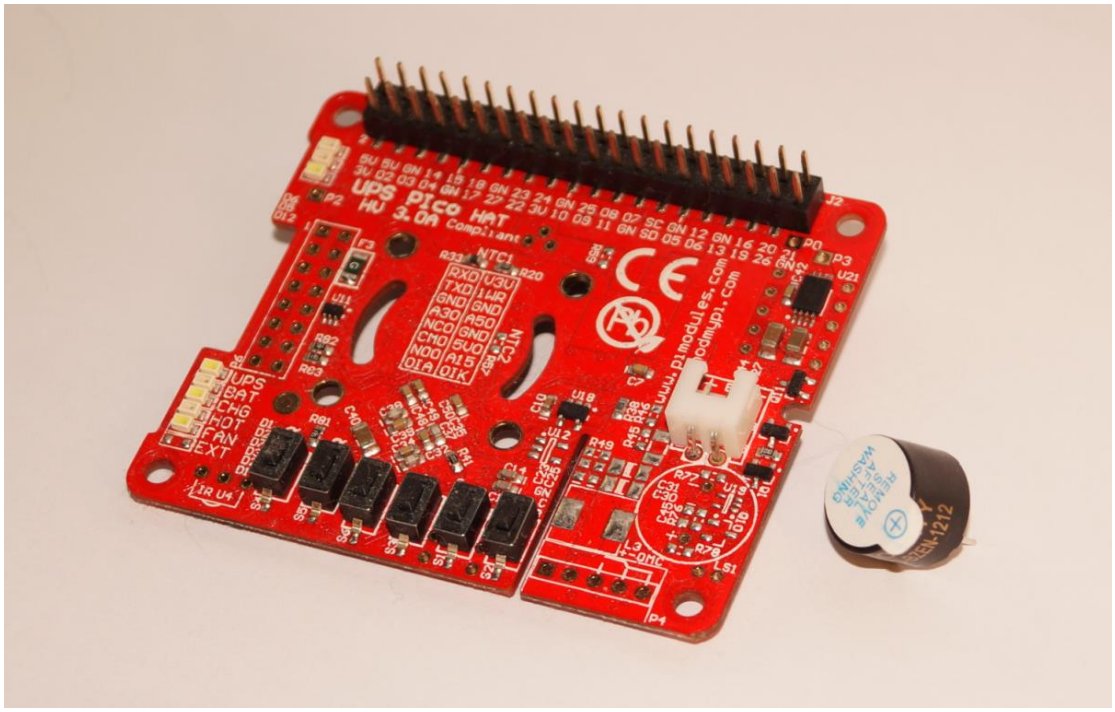


Figure 28 Prepare the UPS PIco HV3.0 HAT and buzzer

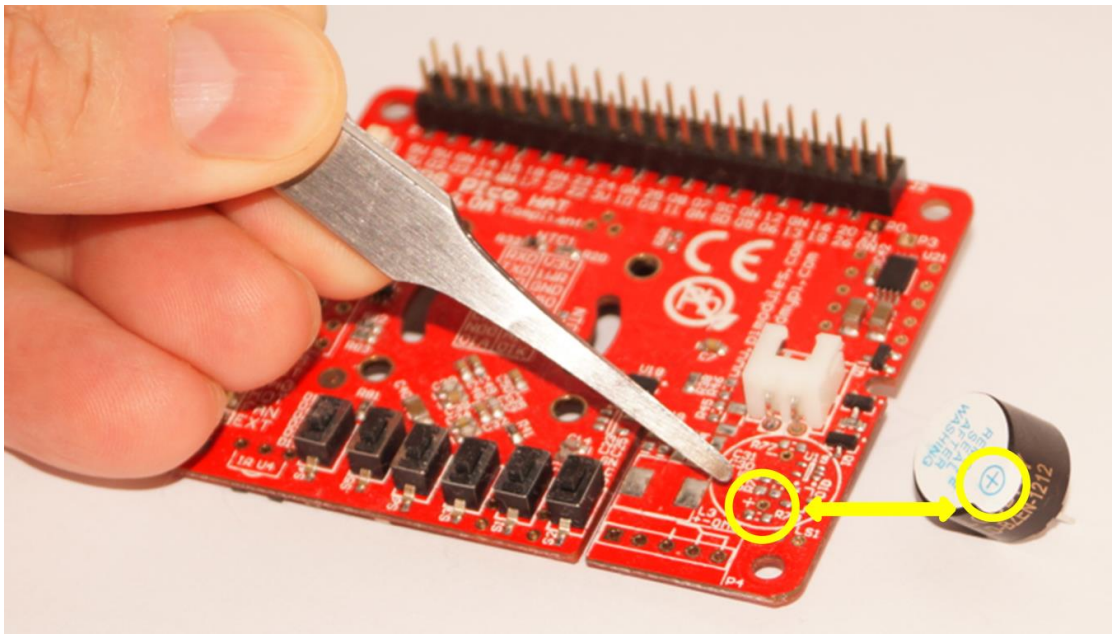


Figure 29 Make sure to solder "+" of the buzzer to the proper place

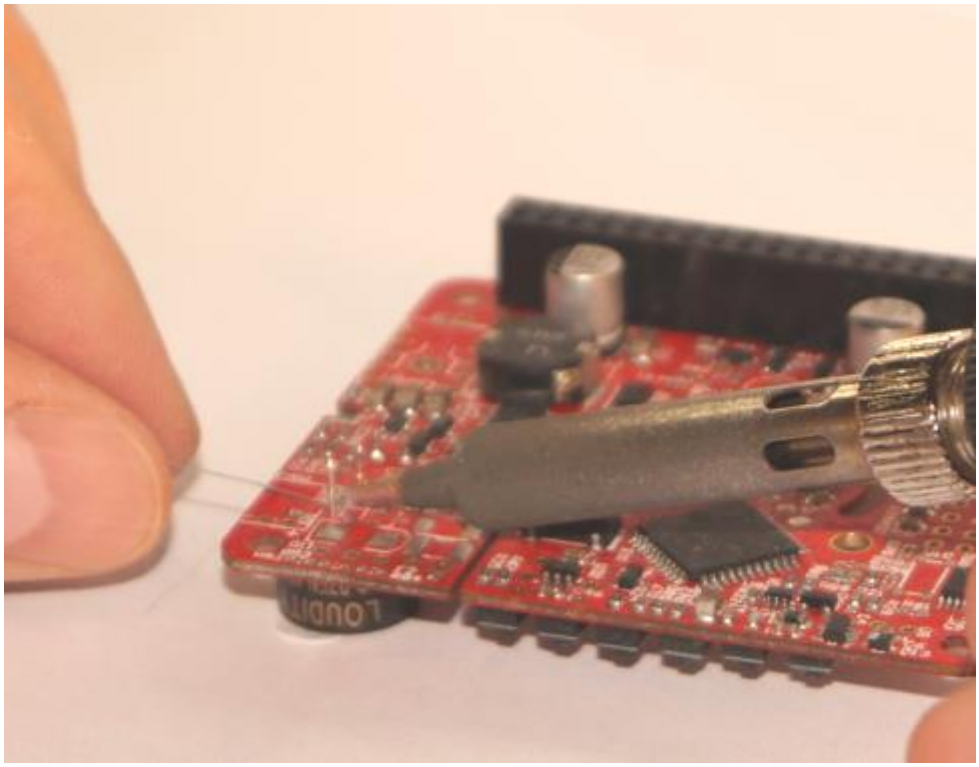


Figure 30 Flip upside down the PCB and colder the pins, then cut the outstanding legs

If you would like to install the Gold-Plated Reset Pin, please do so now following the instructions below. It is not mandatory to install the pin now, but it will

enable full function of the **UPS PIco HV3.0 HAT** module. However, it can be installed in next stages.

If you would like to install the **PIco FAN Kit**, please do so now following the instructions below. It is not mandatory to install the fan kit now, but it will enable full functionality of the **UPS PIco HV3.0 HAT** module.

Assembly of the FAN Kit

One of the add on available for the **UPS PIco HV3.0 HAT** is the FAN Kit. This Kit contains everything what is needed to make it installed on the UPS **PIco HV3.0 HAT** module. There are:

- 1 x Ultra Low Noise DC FAN
- 1 x TO-90 Temperature sensor
- 4 x white 2mm spacers
- 4 x plastic tree clips
- 1 x 2mm connector for the Fan

These instructions will guide you through the installation of the TO92 and FAN.

Figure 31 FAN Kit Contents

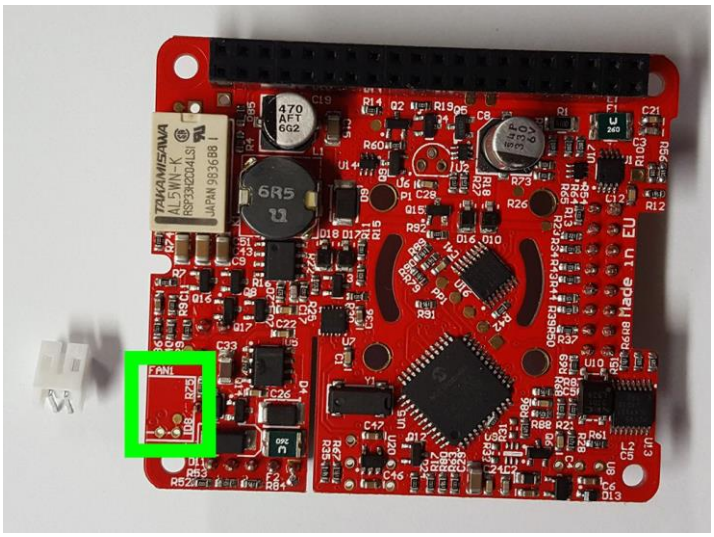


Figure 32 2mm FAN connector placement

Start by soldering the FAN 2mm connector to the **UPS PIco HV3.0 HAT** PCB

Please make sure that before soldering of the 2mm connector, the sounder has been soldered. If not, please solder first the sounder and after that the FAN connector.

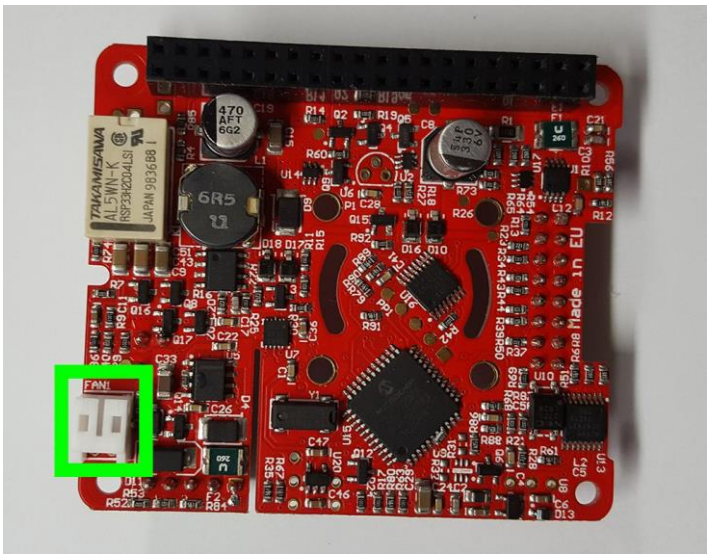


Figure 33 Soldered 2mm FAN connector

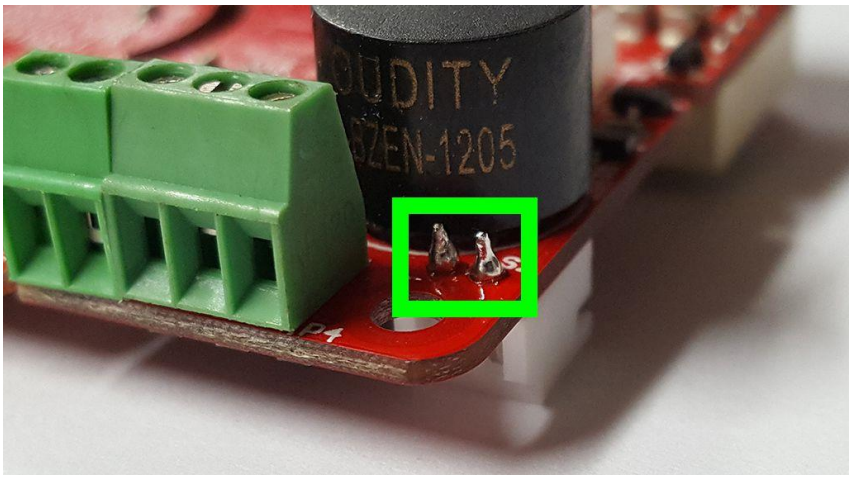


Figure 34 2mm connector on the top side of PCB after soldering

After soldering of the 2mm connector, please cut the outstanding legs.

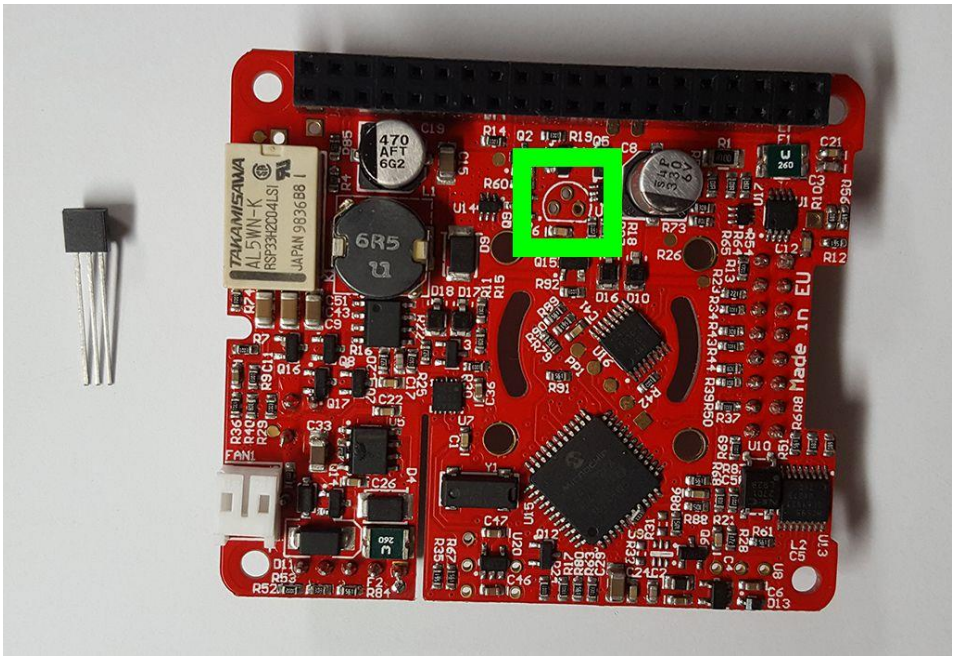


Figure 35 Temperature Sensor fitment place

Next, we'll solder on the TO92 Temperature Sensor. Start by inserting the TO92 into the 3 through holes on the PCB.

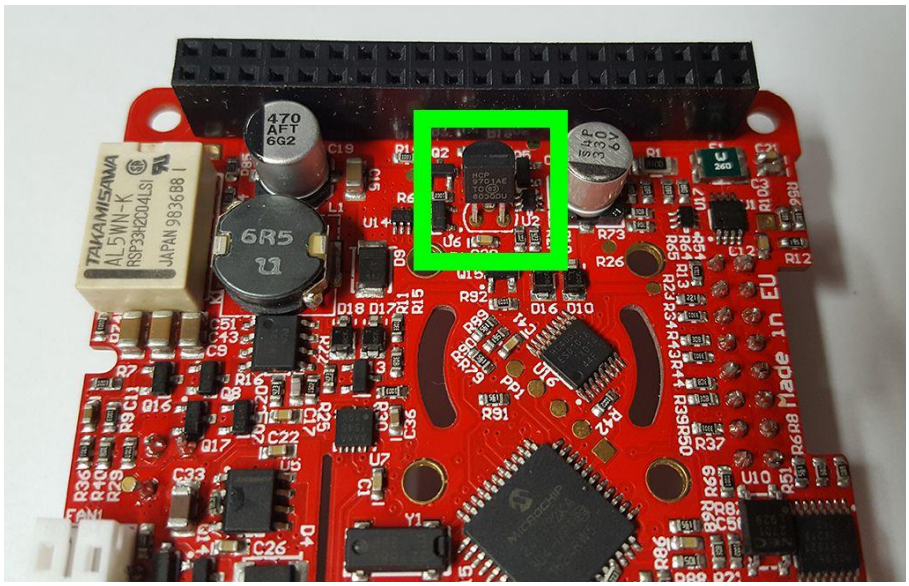


Figure 36 Temperature Sensor passed on the PCB

Flip the **UPS PIco HV3.0 HAT** PCB over, Put the PCB on the Raspberry Pi. Make sure that the spacers have been screwed on the Raspberry Pi and keep the right distance between **UPS PIco HV3.0 HAT** PCB and Raspberry Pi PCB. Press little bit the sensor legs down, to touch the Raspberry Pi PCB and bend the legs out slightly to hold the TO92 in place. It is important to have physical contact of the sensor with Raspberry Pi PCB or to be very close to it (0.5mm – 1mm), to have a proper measure of temperature.

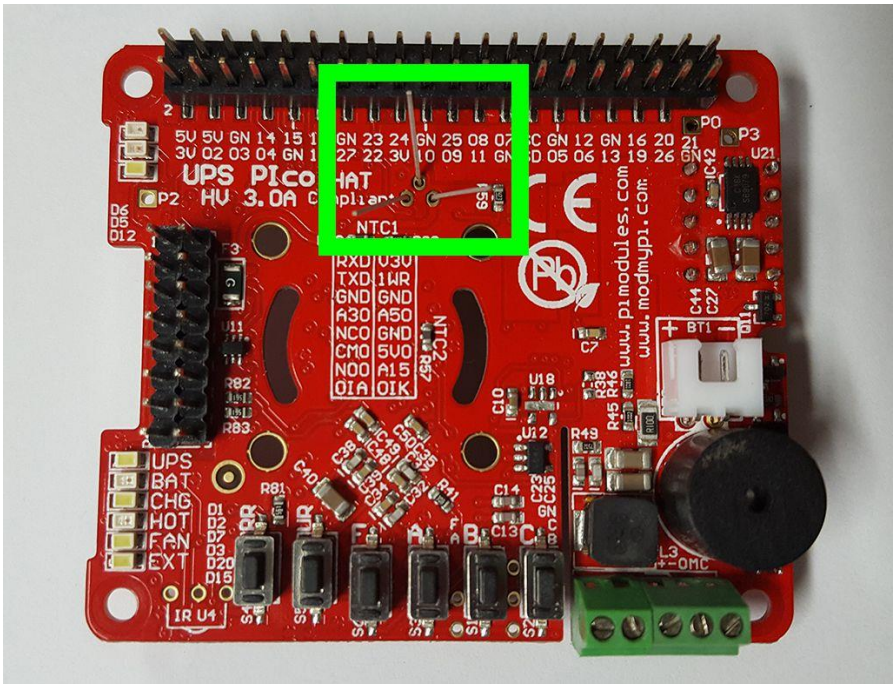


Figure 37 Temperature Sensor legs before soldering

Solder and trim the legs and cut the remain parts.

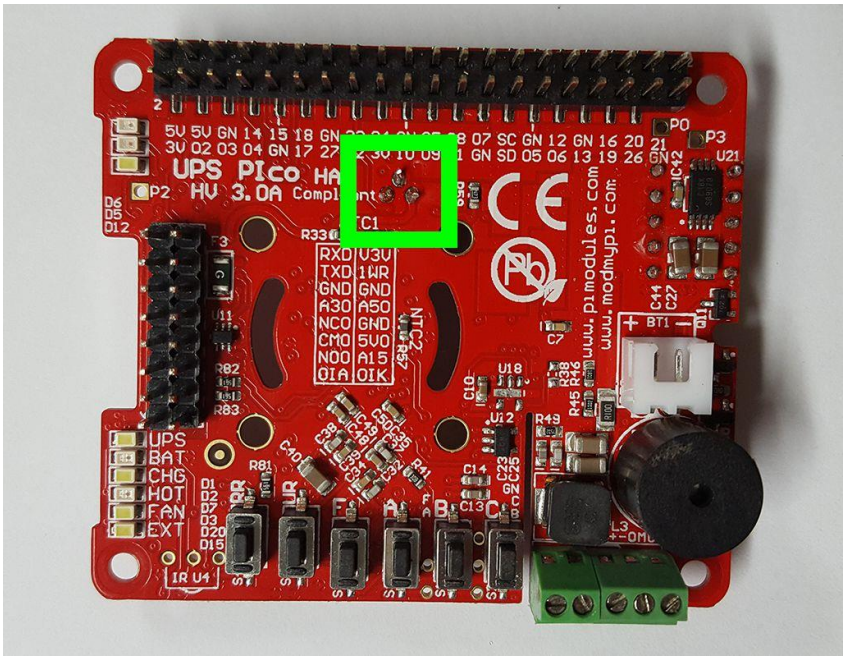


Figure 38 Soldered Temperature Sensor

Now it's time to add the fan. Start by pressing the four studs through the fan mounting holes, from the top of the **UPS PIco HV3.0 HAT**. Do it very carefully, and preferred before installed the Gold-Plated Reset Pin (as when it is installed, it is easy to break it, when pressing them)

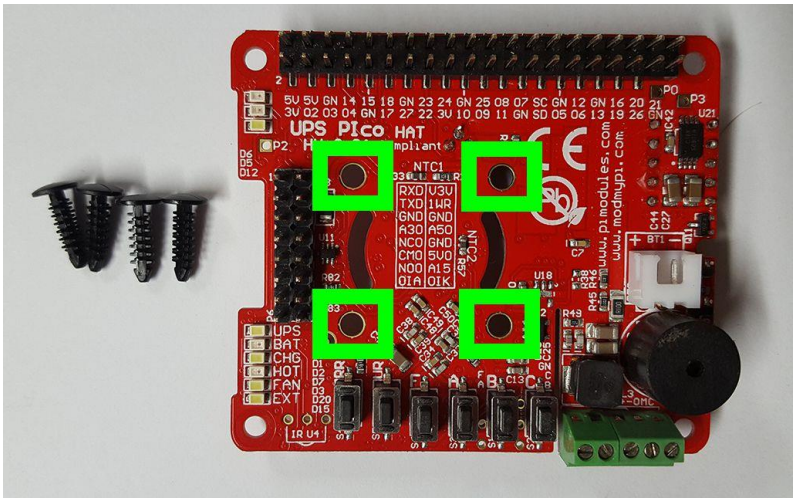


Figure 39 Preparation of the Plastic Tree Clips for FAN

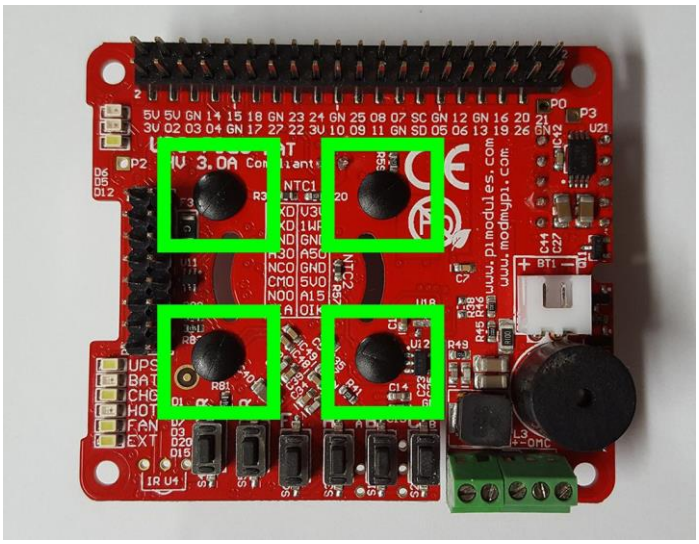


Figure 40 Plastic Tree Clips mounted on the PCB ready for FAN assembly

Flip the **UPS PIco HV3.0 HAT** PCB over.

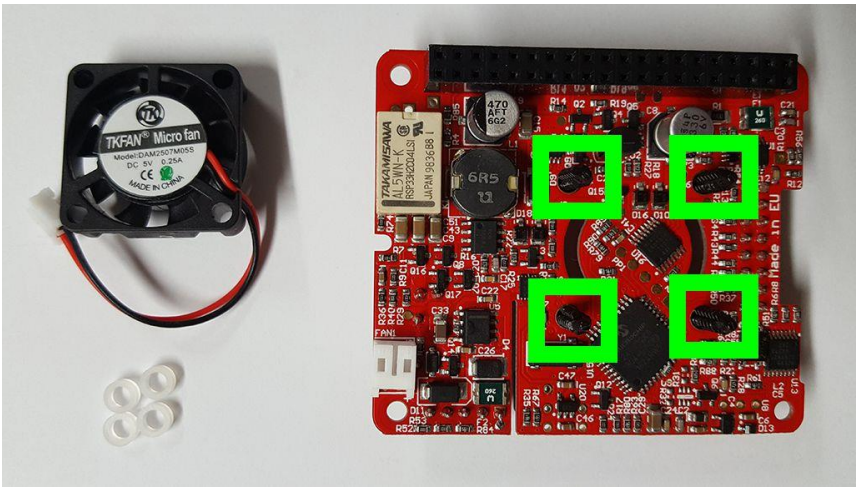


Figure 41 Plastic Tree Clips on the PCB Bottom side

Add a spacer to each of the studs. Finally, add the fan and connect the FAN wire up. The fan blows air towards the label on the FAN. There are 2 ways to mount the FAN. If decided to have this facing down the blow cold air directly onto the SoC of the Pi. It cools better the SoC however collect more dust from outside. If you put on the opposite way (the label on the side of the PCB) then it cools the whole PCB and collect less dust.

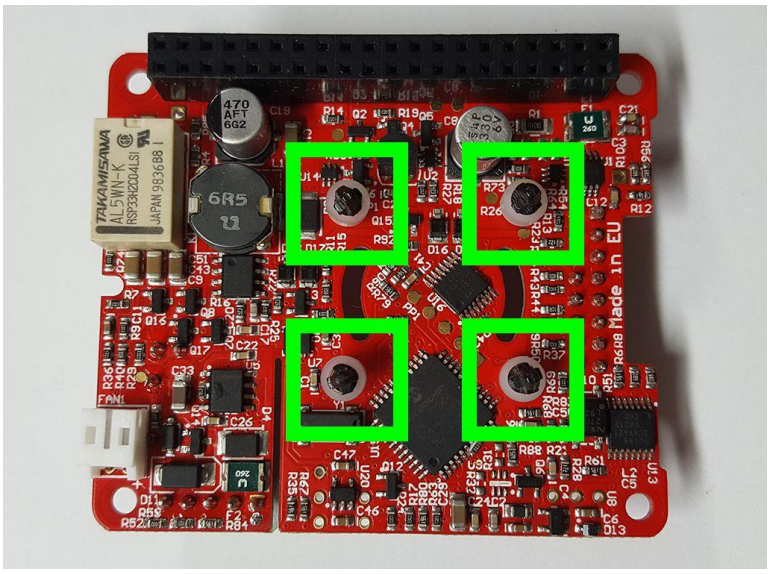


Figure 42 Plastic Tree Clips on the PCB Bottom side with 2mm spacers passed

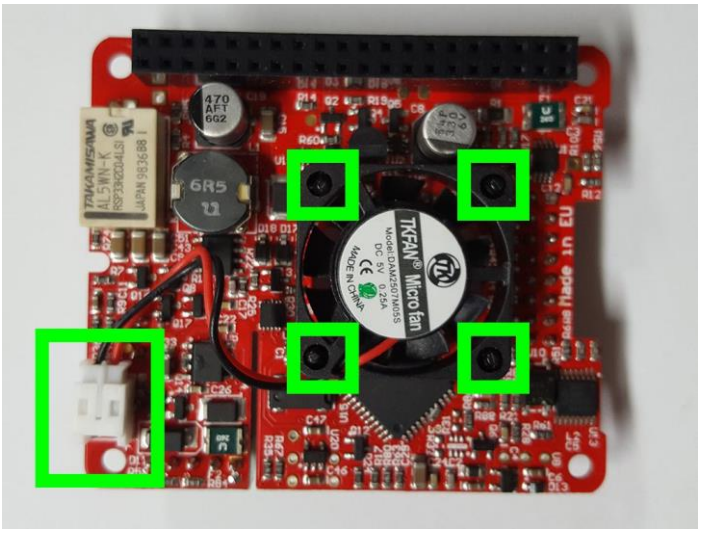


Figure 43 FAN placed on the UPS PIco HV3.0 PCB

When placing the FAN on the Plastic tree clips be very carefully to avoid damaging the FAN propel when pressing it.

Assembly of the Bi-Stable Relay

If you would like to install the Bi-Stable Relay kit, please do so now following the instructions below. Please kindly notice the there are differences in bi-stable relay placement between the versions HV3.0A and HV3.0B/B+. The HV3.0B/B+ has 2 places where the relay can be placed:

- with two galvanic isolated independent contacts DPDT 1A/30V
- with single high current contacts SPDT 2A/30V (HV3.0B only)

Instead of the HV3.0A, that has only one place where the bi-stable relay can be placed

- with two galvanic isolated independent contacts DPDT 1A/30V

It is not mandatory to install the Bi-Stable Relay now, but it will enable full function of the **UPS PIco HV3.0 HAT** module. User need to follow the below steps when assembling it to the **UPS PIco HV3.0 HAT** PCB.

Prepare the UPS PIco HV3.0 PCB, Bi-Stable Relay and 3 ways Terminal Block

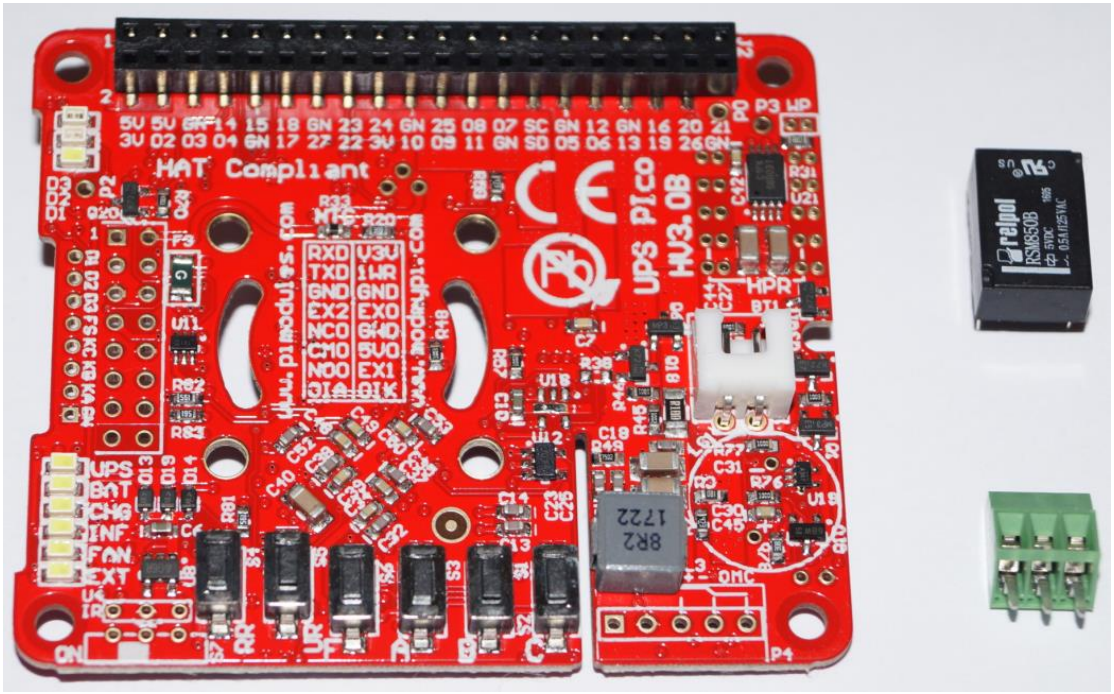


Figure 44 Ready for assembly UPS PIco HV3.0 HATB/B+, Bi-Stable Relay and Terminal Block

Decide on what position bi-stable relay need to be placed (depending to your application if you need higher or lower current). The high current position is marked as HPR on the top side of the PCB. It is valid only for the HV3.0B/B+, see below pictures

Make sure that marker of the Bi Stable Relay (white bold line) and on PCB are on the same direction. It is very important because if Bi Stable Relay will be soldered in a wrong way, it is not possible (ultra-difficult) to de-solder it again and correct

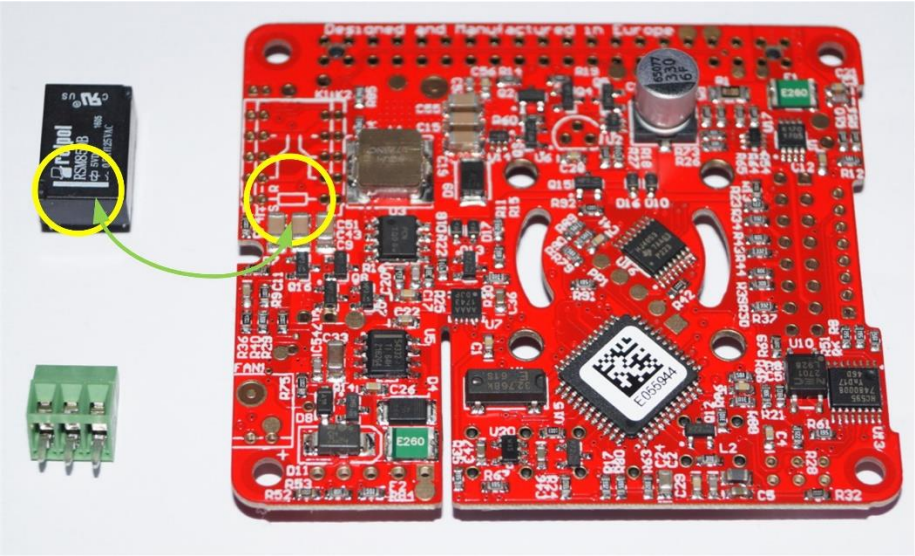


Figure 45 UPS PIco HV3.0 on Bottom Side, and Bi-Stable Relay

Decide what configuration of Bi Stable Relay you need for your application (**H**igh **P**owe**R – K2**, or **L**ow **P**owe**R – K1**) and pass the relay pins through the holes on selected place (K2 or K1)

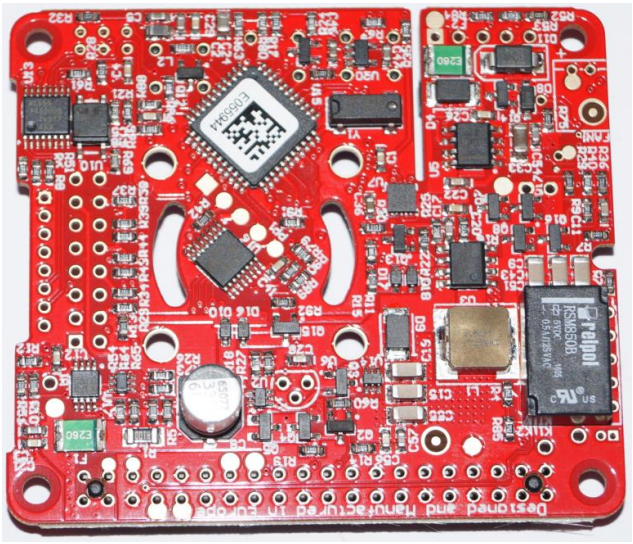


Figure 46 Bi-Stable Relay on the PCB place K2

If relay is used in UPS Pico HV3.0A, then only one position is available.

Put the PCB with Relay to the opposite side and solder **only** one pin. It is very important to solder only one pin, as if you made a mistake it will be easy to correct it.

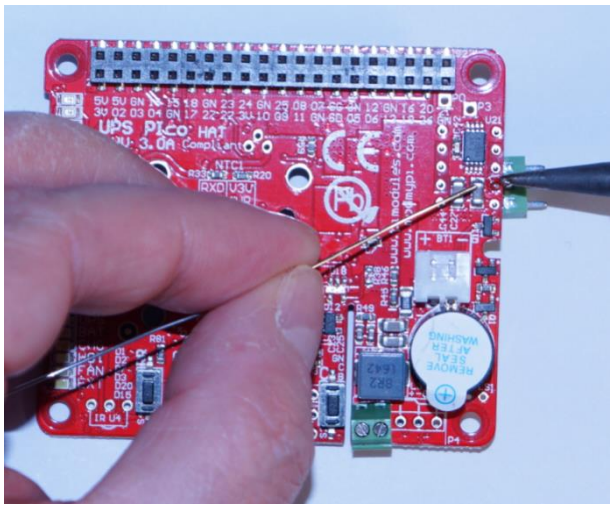


Figure 47 Soldering of one pin of the Bi-Stable Relay

Then make sure that everything (after double checking) are OK and proceed with soldering of all other Pins.

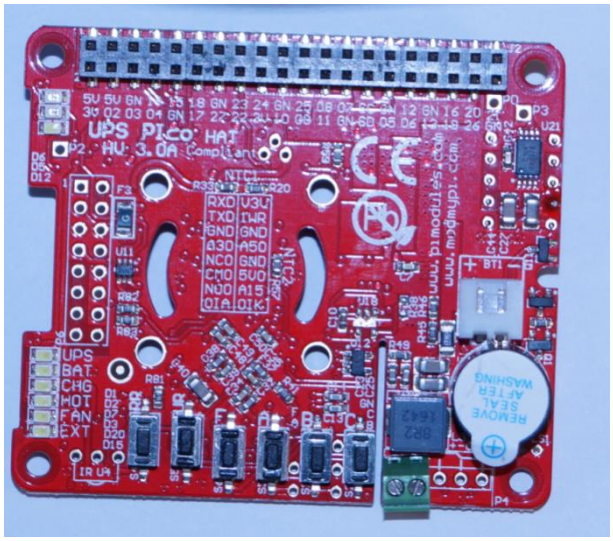


Figure 48 Soldered one pin of the Bi-Stable Relay

Solder all the rest of Bi-Stable Relay very carefully to avoid any short-cut with other near placed components.

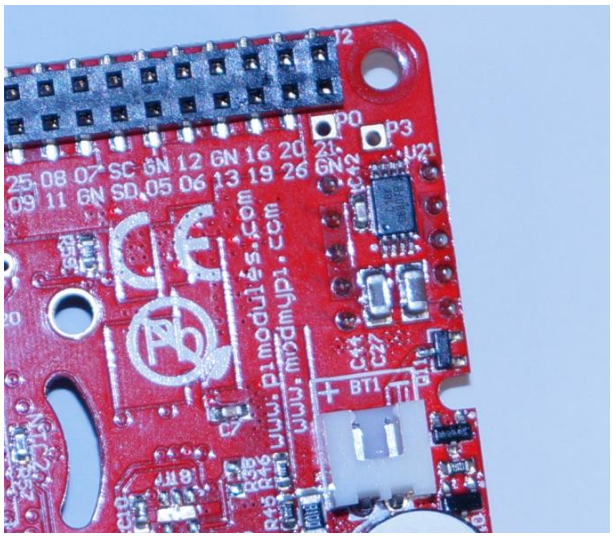


Figure 49 Completely Soldered Bi-Stable Relay

After completing of the soldering, cut the outstanding over the PCB pins (very carefully – PCB is very density!!!)

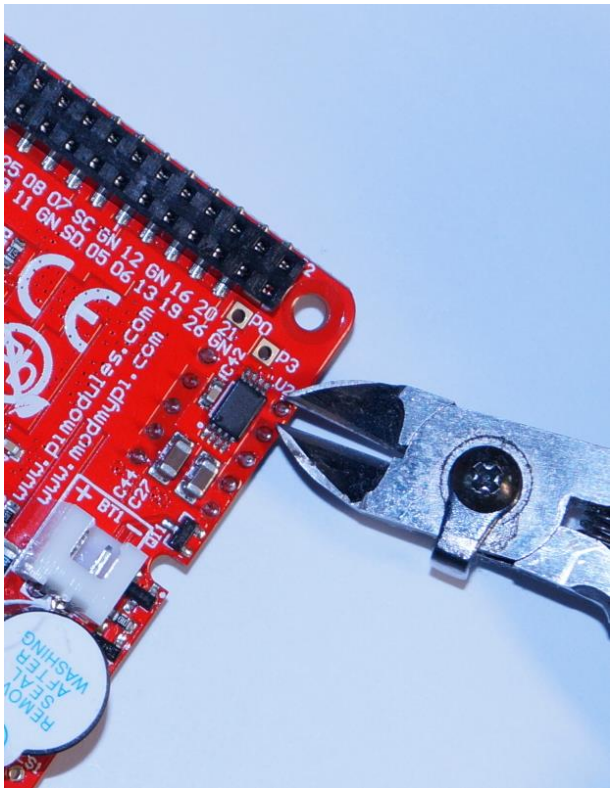


Figure 50 Cutting of the outstanding pion of the soldering Bi-Stable Relay

Prepare the 3 ways terminal block. Make sure that cables holes are in the proper side (looking outside)

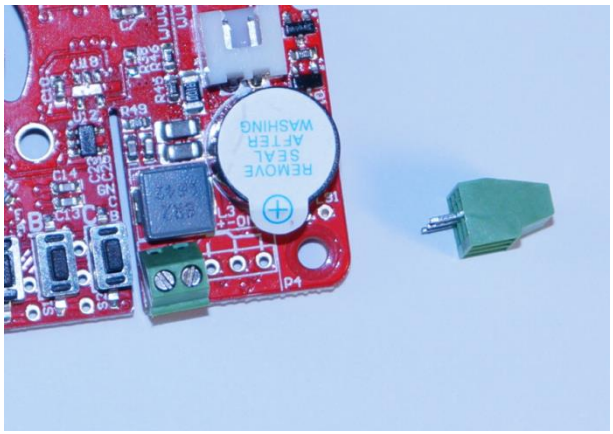


Figure 51 Terminal Blocks for the Bi-Stable

Pass the Terminal Block through the holes. Make sure that Terminal Blocks cables holes looks to the outside side.

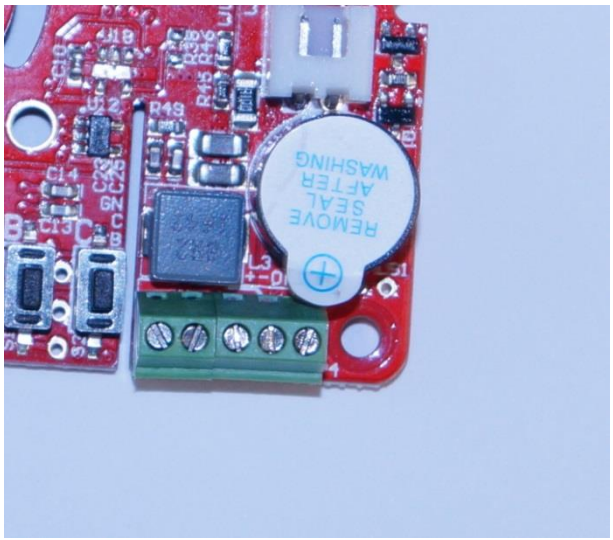


Figure 52 Terminal Block on the proper side

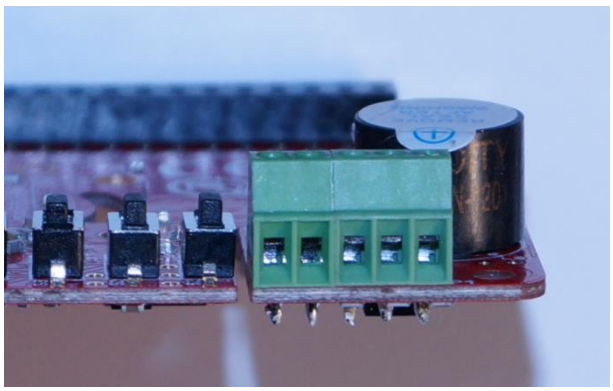


Figure 53 Soldered Terminal Blocks

Solder Terminal Block Pins and cut the outstanding legs over the PCB, be very careful with other components placed near to them (SMD Resistors).

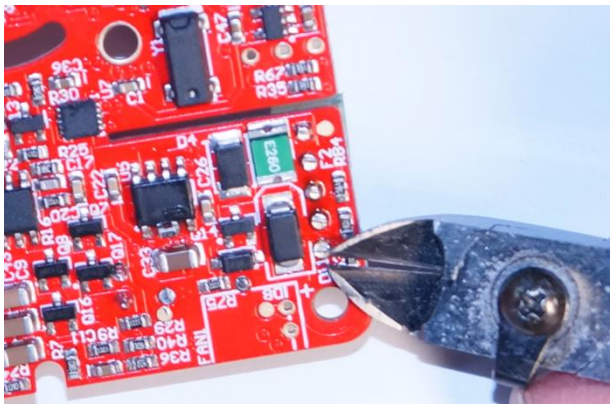


Figure 54 Cutting of the Terminal Block outstanding legs

Power Supply Unit Recommendations

Please ensure that you are using a good quality Power Supply Unit available for powering of the Raspberry Pi and **UPS PIco HV3.0 HAT**. A PSU 5V@2.5A is recommended, however for more advanced applications 5V@3.0A PSU is preferred. This will ensure that there is enough current to recharge the PIco's battery. Low quality PSUs, or PSUs with bad quality of supply cables cause a voltage drops on the Raspberry Pi® 5V GPIOs that are recognized by the PIco and force a wrong functionality. It is also mandatory to have good quality micro USB powering cable. Please avoid PSUs that use dual USB connectors as there are double voltage drops on both USB connections (micro USB, and USB socket).

Once you have all parts correctly installed, we're ready to proceed with software installation

Software Setup for UPS PIco HV30 Stack/Top-End/Plus

There are a few very simple steps than need to be followed to setup the software. In any case if user cannot follow these guides, a ready SD image (8GB) is available on our forum, always with the latest NOOBS, with all software procedures installed. It can be used also as a running example on a separate SD card for debugging purposes.

There is no need to have placed the **UPS PIco HV3.0 HAT** on top of the Raspberry Pi® when installing the software, however can be also placed on top. The presence of the UPS PIco HV3.0 HAT does not affect software installation.

The software installation procedure consists the following steps that need to be executed

- Operating System Installation (Raspbian)
- Activation of I/Os (i.e. I^2C) as also some libraries installation
- Daemons Installation
- RTC Installation
- If needed in the future, firmware updates installation (this operation need to have installed **UPS PIco HV3.0 HAT** hardware)

Installation of the Operating System (Raspbian)

Please download and proceed with installation of the latest NOOBS or install a separate RASPBIAN on your SD card. If you like you can use also a ready to use installed image that can be downloaded directly from our forum. It consists always the latest NOOBS installed with all stuff needed included. There is also an image restore program included in the ZIP file. It can be downloaded from the following link.

http://pimodules.com/products/ups-pico-hv3-0a-b-b/common-updates

The installed software for interaction with the Raspberry Pi, is using the GPIOs GPIO_GEN27 and GPIO_GEN22. These GPIOs are used to send and receive pulse train to/from the Raspberry Pi. It is also used to initiate the shutdown procedure when/if it is needed. The Daemons are monitoring these GPIOs and fire-up and interrupt on the Raspberry Pi side. This approach is very flexible and does guarantee that interaction even if huge files are copied and Raspberry Pi is ultra-busy with other tasks.

Installation Procedure of Daemons and email broadcasting System

1. Install Raspberry Pi Operation System (i.e. NOOBs)
2. Enable the I^2C
3. Ensure that Python is installed and updated, by using the following command

sudo apt-get install python-rpi.gpio

4. Ensure to run below line

sudo apt-get -y install git python-dev python-serial python-smbus

python-jinja2 python-xmltodict python-psutil python-pip

(Take note of the line-wrapping above, it should all be on one line)

5. Note that some of the above can also be install with pip as below:

sudo pip install jinja2

sudo pip install xmltodict

(Obviously after python-pip has been installed)

6. Clone Raspberry Pi daemons and email broadcasting system from the GitHub using the following command

sudo git clone https://github.com/modmypi/PiModules

7. Move to the required directories where software has been copied.

8. First to the email broadcasting system (package)

cd PiModules/code/python/package

9. Then proceed with the installation of the email package software

sudo python setup.py install

more information about the package usage and details are available at

https://github.com/modmypi/PiModules

10. Second to the System Monitoring and File Safe Shutdown Daemons (picofssd)

cd ../upspico/picofssd

11. Then proceed with the installation of the picofssd daemons software

sudo python setup.py install

12. Once the script has been installed, it can be installed to the `SystemD` with the following command

sudo systemctl enable picofssd.service

13. Now when the Pi is rebooted the daemon should start automatically.

The Daemons can be started and stopped in the usual way for SystemD:

sudo systemctl start picofssd.service

sudo systemctl stop picofssd.service

Important Notices:

1. Both PIco packages must be installed even if not used.
2. It is very important to start/stop the Daemons Service when doing Hardware Reset of the PIco HV3.0 to avoid undefined situations with pulse train recognition procedure by the system. Resetting the PIco with Not Stopped the Daemon Service can cause an unexpected system shutdown (however without card corruption - system will just safety shutdown).

Installation Procedure of the UPS PIco HV3.0 Hardware RTC

1. Ensure to run below line

   sudo apt-get -y install i2c-tools

2. Edit by running the following line

   sudo nano /etc/modules

   and check, make sure to have the following items in the file and add what is missing:

   i2c-bcm2708

   i2c-dev

   rtc-ds1307

3. Edit by running the following line

   sudo nano /boot/config.txt

4. and add the following to this file:

   enable_uart=1

   dtoverlay=i2c-rtc,ds1307

5. Edit by running the following line

   sudo nano /etc/rc.local

6. and add the following line before "**exit 0**"

   sleep 4; hwclock -s &

7. Reboot system by

   sudo reboot

8. Remove the **fake-hwclock** which interferes with the RTC **hwclock**

   sudo apt-get -y remove fake-hwclock

   sudo update-rc.d -f fake-hwclock remove

9. Run

   sudo nano /lib/udev/hwclock-set

10. and comment out these three lines:

```
#if [ -e /run/systemd/system] ; then
# exit 0
#fi
```

11. Run **date** to verify the time is correct.

12. Plug in Ethernet or WiFi (if not plugged before) to let the Pi sync the right time from the Internet. Once that's done, run:

```
sudo hwclock -w
```

13. to write the time, and another

```
sudo hwclock -r
```

13. to read the time

That's it! Next time you boot the time will automatically be synced from the RTC module.

Automatic Installation Scripts

TBC

Bootloader Feature – keep the system up to date

UPS PIco HV3.0A/B/B+ HAT is a very flexible hardware platform that offers a wide range of features. Most of them are software programable. Therefore, during the time, new versions with additional features are released. It is mandatory for the user, to have the ability to upload the newest firmware version whenever it is released, to keep the **UPS PIco HV3.0A/B/B+** HAT up to date. The firmware upload to the **UPS PIco HV3.0A/B/B+** is done by running a small piece of software located in dedicated memory part in the micro controller called boot sector. This memory part is protected from any erase, so even if uploading of the new firmware procedure fails, this bootloader will never fail.

The execution (the invoking) of the bootloader can be done from a software level by running of some dedicated commands, or manually by pressing of dedicated key sequence. The bootloader is equipped with additional protection mechanism called watch-dog, and if within 32 seconds from invoking of it system not start uploading the new firmware, **UPS PIco HV3.0A/B/B+** HAT will be reset, and start execution of the old already existing firmware. The bootloader functionality ensures that the **UPS PIco HV3.0A/B/B+** HAT is up-to-date, and allows users to report various changes that can be implemented on the user's side. It is extremely useful functionality and ensures that the product has longevity.

As the bootloader uses the Raspberry Pi® Serial Port (RS232), it is mandatory to have it free on the Raspberry Pi® (without any hardware occupying it). It is also important that you ensure that there is no software using it. As well if minicom has been used, please restart the Raspberry Pi, as minicom keeps the RS232 interface occupied.

When doing the firmware upload, it is also mandatory to have the system cable powered (there is no UPS protection provided during that time) as also have stopped the Daemons to avoid any undefined conditions with used GPIOs.

Firmware updates Procedure of the UPS PIco HV3.0 (on RPi3)

Serial Port disable Procedure

To disable serial communication over the UART long enough to do a Firmware update. You need to do two things.

1. Run

   sudo nano /boot/config.txt and add this line:

   dtoverlay=pi3-disable-bt

2. Run

   sudo systemctl disable hciuart

3. Run

   sudo systemctl stop

4. Run

   sudo systemctl disable

5. You could also disable the console by removing this line from **/boot/cmdline.txt** if present, then reboot:

   console=ttyAMA0,115200

As it has been written before the **UPS PIco HV3.0 A** HAT features an embedded serial bootloader which allows users to update the unit's firmware. The firmware can be uploaded using a dedicated python script, called **9600_picofuHV3.0.py**

It is mandatory to have previously installed python and I2C-tools on the Raspberry Pi. You will install these during initial PIco setup outlined previously in this entity. Please install **smbus** support for python to enable additional functionality. Simply run the following command (with an internet connection):

sudo apt-get install python-smbus

The first task which is done by the **UPS PIco HV3.0 A** HAT after reset is to check if bootloader has been requested. If not, then the rest of the firmware runs. Otherwise, the **UPS PIco HV3.0 A** HAT lights the Orange User LED and waits for the firmware upload from the Raspberry Pi ®, and when firmware starts uploading, change from Orange User LED to Blue User LED.

There are two ways to invoke the bootloader mode and to upload the new firmware:

Automatic Bootloader Initiation

Remember to have system cable powered as during the boot loading procedure system is not protected from power losses

The bootloader is invoked by running the following command line:

sudo i2cset -y 1 0x6b 0x00 0xff

sudo python 9600_picofuHV3.0.py -v -f

The should be replaced with the name of the last firmware update, or the firmware you wish to use.

When firmware starts the upload procedure, the Orange User LED will have lit, and then when firmware starts uploading the Blue User LED will lit and UPS LED will be blinking.

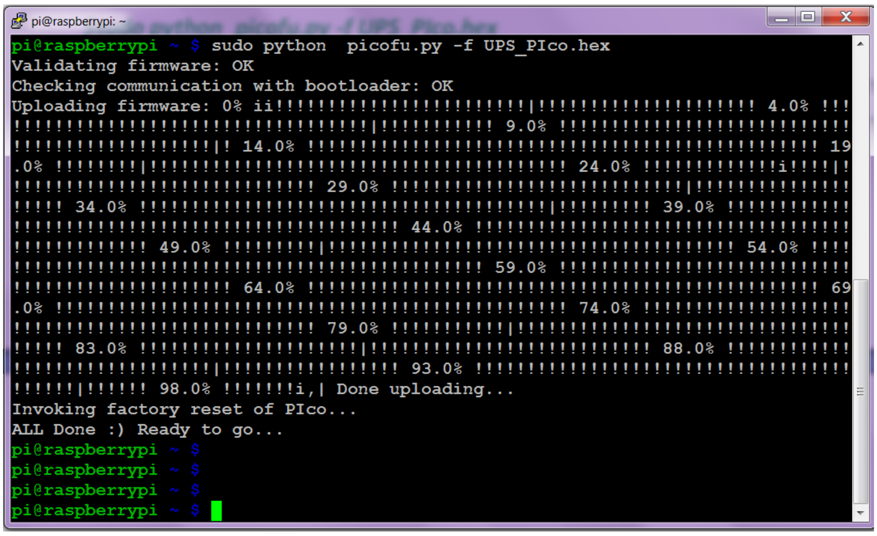


Figure 55 SSH screen when firmware uploading

Once complete the system with output **ALL Done :) Ready to go. . .**

We would recommend that you now shutdown your Pi and **UPS PIco HV3.0 A** HAT completely to ensure that all changes are integrated. Once you've rebooted your system, you can check the UPS PiCo firmware version using the following command:

sudo i2cget -y 1 0x69 0x26

In this case the system should output **0xXX**, signifying that the firmware has updated correctly.

Manual Bootloader Initiation

For emergency reasons (i.e. faulty upload, upload of a wrong corrupted files etc.) The **UPS PIco HV3.0 A** HAT has the ability to invoke the bootloader manually, via the on-board buttons. You can do this instead of using the automatic initiation outlined above. However, user need to have physically access to the device, as needs to push buttons.

It is very important to **stop** (before upload) and then **start** (after upload) the Daemons Services when doing Hardware Reset of the PIco HV3.0 to avoid undefined situations with pulse train recognition procedure by the system. Resetting the PIco with Not Stopped the Daemon Service can cause an unexpected system shutdown (however without card corruption). Please use the below command to stop the FSSD service

sudo systemctl stop picofssd.service

The following procedure needs to be followed:

- Press and hold the **UR** button
- Continue to hold the **UR** button, and press and hold the **F** button.
- Release the **UR** button, but keep holding the **F** button
- Release the **F** button

The Orange User LED will have lit, and system will be able to receive the firmware update

Then write the following command on the Raspberry Pi command line

sudo python 9600_picofuHV3.0.py -v -f

The should be replaced with the name of the last firmware update, or the firmware you wish to use.

If within 32 second after boot loader initiation the firmware is not start uploaded, the UPS PIco HV3.0 will be reset to normal working conditions by internal Watch Dog mechanism. This has been implemented for security reasons for remote firmware upload.

Post-Firmware Update procedure

After firmware upload some steps are needed to return the system to previous state.

Please follow below steps to do that.

Run

sudo nano /boot/config.txt

and REMOVE this line:

dtoverlay=pi3-disable-bt

Run

sudo systemctl enable hciuart

Run

sudo systemctl enable

Re-add this line to **/boot/cmdline.txt** just before **console=tty1** line:

console=ttyAMA0,115200

Then run the following:

sudo systemctl start picofssd.service

Reboot system by **sudo reboot**

Take note you should see the UPS LED blinking steady at 400ms when the OS is ready.

Using the UPS PIco HV3.0A/B/B+ HAT

The **UPS PIco HV3.0A/B/B+ HAT** is a complete and flexible cable/battery powering system, that provides also a protection from cable powering losses and save the SD card from corruption (the UPS functionality). In addition, it is offering a plenty of additional features that make it unique on the market. Compared with other similar Raspberry Pi® UPS or Powering Systems is the most advanced than any other. The usage and their capabilities will be described here below. There have been divided in following entities:

- Running the System for the first time
- System Functionality and Features
- The PICo (I^2C) Interface
- User Applications Hardware Interfaces
- Measuring and Monitoring System
- Basic System Scheduler
- Events Triggered RTC Based System Actions Scheduler

Running the System for the first time

Once proceeded with Hardware and Software installation, user can start using of the complete system. Ensure that **UPS PIco HV3.0A/B/B+ HAT** is properly placed on the Raspberry Pi® top, and spacers are screwed. Plug-in the battery to the BT1 socket (battery can be plugged/unplugged also when system is running - cable powered, however we recommend to plug-it from the beginning) and apply power to the Cable Power Inputs. They can be the Raspberry Pi® micro USB, or the EXT (7-28V DC) power, or both at the same time. **UPS PIco HV3.0A/B/B+ HAT** is protected with ZVD circuits and both powering sources can be supplied at the same time without any problem. If system is used in “on the go mode”, as exclusively battery powered system (as an Intelligent Power Bank), battery should be plugged-in before system will be switched ON (valid for the **UPS PIco HV3.0B/B+ HAT**) with **Magic Switch**.

The **Magic Switch** can be used ONLY if before of use a proper register have been setup. This procedure is detailed described in next chapter. Using of **Magic Switch** before a proper setup it, can cause unexpected effects like (absence of possibility to switch OFF with Magic Switch, or absence of File Save Shutdown). Therefore, it is required to setup system for the first time using cable powering, setting the **Magic Switch Register** (if planned to be used) and then use system as an Intelligent Power Bank powered.

After cable power applying Raspberry Pi® will start booting and during that time the UPS Blue LED will lit continuously. After about 30-40 seconds when Raspberry Pi® boots-up and properly installed Daemons starts running the UPS LED should be blinking about 2 times per second as far Cable Power is still connected. If the UPS LED is not blinking, that means the

Daemons are wrong installed, and user need to check the installation process again. If the UPS LED is blinking properly remove any cable power applied and the UPS LED should be blinking much slower – once every 2 seconds. These two steps ensure you that the Daemons are installed correctly and **UPS PIco HV3.0A/B/B+ HAT** running properly. Your system is ready and protected. If you will not apply the cable power again, after 60 seconds of running on battery, your system will be forced by **UPS PIco HV3.0A/B/B+ HAT** to safe shutdown. If Cable power will be applied, your system will boots-up and start running again. This is the basic usage, and if you have recognized all stages, you are ready. Enjoy your new UPS PIco installed and protecting your system. For furthermore advanced usage you need to follow the next chapters.

If used the **Magic Switch** is used (if previously is setup the appropriate register) - switch it ON without cable power applying. The system starts booting up, Raspberry Pi® will start booting and during that time the UPS Blue LED will lit continuously. After about 30-40 seconds when Raspberry Pi® boots-up and properly installed Daemons starts running the UPS LED should be blinking about 1 time every 2 seconds as far **Magic Switch** is ON. If the UPS LED is not blinking, that means the Daemons are wrong installed, and user need to check the installation process again. If you move the **Magic Switch** to position OFF again, then Safe Shutting Down will be started, UPS LED will light continuously, and after 30-40 seconds system shutdown and disconnect battery source.

System Functionality and Features

The **UPS PIco HV3.0A/B/B+ HAT** core functionality is to provide powering battery back-up and protect the Raspberry Pi® system from micro SD card corruption if power loss occurs during writing to micro SD card as also supplying the Raspberry Pi® based systems as Intelligent Power Bank with Safe Shutdown when OFF.

However, due to implementation of enhanced battery powering system it can be used for any kind of Battery or Cable Powered Application.

The **UPS PIco HV3.0A/B/B+ HAT** is plugged on top of the Raspberry Pi® and it is continually monitoring the GPIO 5V Pins. The proprietary implemented algorithm analyzes the powering status on these GPIO's and recognizes when cable powering is going to be lost. If so, then within 10 uS applies the Battery Back-Up power and when cable power returns release it. The **UPS PIco HV3.0A/B/B+ HAT** powering analyzer check the stability of the cable powering and only if it is stable for more than 3 seconds release the battery power Back-up returning to Cable powering.

In case of usage as **Intelligent Power Bank** (without Cable Power Source) the checks the system power and when is switching OFF ensure that system will be properly shuttled down, without SD card corruption

All functionality of the **UPS PIco HV3.0A/B/B+ HAT** can be monitored or changed/forced via enhanced set of System Variables (System Registers) accessed through the I^2C interface. This Interface is described in detail in another chapter. It is called **P**eripherals **I**^{2}C **Co**ntrol **I**nterface - the **PICo Interface** - and practically allows user to change most of system parameters via command line (if SSH or Terminal is used) or via any language interface (Python, C, etc.). Some of the System Parameters can be also monitored via Raspberry Pi® using minicom® by using of the Raspberry Pi® Serial Port (if it is released for other applications) or again higher-level language interfaces.

The **PICo Interface** is occupying pre-defined (with possibility to change their location) addresses on the Raspberry Pi® address I^2C space. By default, they are **0x68**, **0x69**, **0x6A**, **0x6B**, **0x6C**, **0x6D**, **0x6E**, **0x6F**. In next chapters will be analyzed how to use of these System Registers. There are specified in the Table 8 UPS PIco HV3.0 HAT I2C addresses.

The installed software for interaction with the Raspberry Pi® (Daemons), is using the GPIOs GPIO_GEN27 (sending pulse train to the Raspberry Pi® and initiating the Safe Shutdown) and GPIO_GEN22 (replying by the Raspberry PI® to **UPS PIco HV3.0A/B/B+ HAT**). These GPIOs are used to send and receive pulse train to/from the Raspberry Pi®. They are also used to initiate the shutdown procedure when/if it is needed. The Daemons are monitoring these GPIOs and fire-up and interrupt on the Raspberry Pi® side. This approach is very flexible and does guarantee that interaction even if huge files are copied and/or Raspberry Pi® is ultra-busy with other tasks. It is not allowed to use these GPIOs for any other Raspberry Pi® functionality.

There is already ready (and under extensively testing process) alternative interaction system with the Raspberry Pi ® which is not using any GPIO. Therefor all GPIOs are free for user

applications. However, this new interaction system will be released in new firmware versions.

The PICo (I²C) Interface - Peripherals I²C Control Interface

The **P**eripherals **I**²C **Co**ntrol – The **PICo Interface** – is an implementation of **I²C** interface adapted to easy control of the peripherals connected to the Raspberry Pi® via simple command line or trough programming language. By using human understandable simple commands, control of the **UPS PIco HV3.0 HAT** peripherals are made extremely simple. Control at programming language level is also possible and easy. The core concept of the **UPS PIco HV3.0 HAT** interface is that all peripheral device control and data exchange between it and Raspberry Pi® variables are common for the **I²C interface** as also for the peripheral itself. Therefore, any change of them by either party, Raspberry Pi® and the peripheral, causes immediate update and action.

There are two types of registers available:

Common, where data are stored in the same place and any change on it will cause action on the **UPS PIco HV3.0 HAT** Module

Mirror, where are copy of data stored on internal variables of the **UPS PIco HV3.0** Module, they are protected, so changes on it will not implies the **UPS PIco HV3.0** Module functionality and will be overwritten immediately when **UPS PIco HV3.0** Module recognized changes on them

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

- 0x69 ->UPS PIco HV3.0 Module Status Registers Specification
- 0x6A -> UPS PIco Hardware RTC Registers Direct Access Specification
- 0x6B -> UPS PIco Module Commands
- Events Triggered RTC Based System Actions Scheduler Commands
    - 0x6c -> Start Time Stamp
    - 0x6d -> Actions Running Time Stamp
    - 0x6e -> Events Stamp
    - 0x6f -> Actions Stamp

The location address of them can be changed. This procedure is described in next chapters.

System Cold Start, Warm Start, Default Start, "on the Go" Start and UPS LEDs behaviors

Cold Start

Cold Start is called when Cable Power is applied for the first time to the System after battery connection, the Raspberry Pi ® is starting up, and **UPS PIco HV3.0 HAT** is using all parameters stored in the internal EEPROM (default or user changed).

This start-up is called **Cold Start**, and means that System is starting up for the first time without power cycling as battery is connected for the first time.

Note: If you are doing a Cold Start, and battery is connected, as the Raspberry Pi® is protected also during the booting process, it is enough to connect cable power just for 2 seconds. The System will continue starting-up with battery power back-up (without cable power applied).

Warm Start

This is the most used, and normal type of System Start-up. It happens when System is Cable Power or FSSD button is pressed, after Safe Shutdown of the system, and **UPS PIco HV3.0 HAT**. This start-up is called **Warm Start**, and means that System is starting up from Low Power Mode (Power Cycling), RTC is running as system is battery powered and is in Low Powering Mode.

Note: If system is Warm Started, the Cable Power need to be applied for minimum 8 seconds to be recognized. This is the most used System Startup.

Default Start

When System is Cable Power user has a possibility to restore the factory defaults. To do that the following steps need to be followed:

- Press and hold the **UR** button
- Continue to hold the **UR** button, and press and hold the **C** button.
- Release the **UR** button, but keep holding the **F** button
- Release the **F** button

Ten flashes of the User LEDs will be visible (for about 5 seconds) during that time the Internal **UPS PIco HV3.0 HAT** EEPROM will erased and written with factory default values, including factory default battery. After that the system will start running normally with new (default) settings.

"On the Go" Start

TBC

Battery Powering Protection

Due to shipping regulations in some countries, it is required to ship the **UPS PIco HV3.0 HAT** with battery connected, however without system to be powered. Therefore, to cover this requirement, a dedicated battery connectivity protection system has been implemented. It works in the following way:

1. When system is not cable powered (via Raspberry Pi ® or via External Powering) connecting of battery does not cause system powering, as connected to the **UPS PIco HV3.0 HAT** battery is in fact electrically disconnected. It has been implemented by using a high current/ultra-low resistance MOSFET switch (12 mOhm/7A) in default (hardware forced to OFF condition).

2. There is no possibility to start the system (even if battery remain connected to their socket) until External Cable (to Raspberry Pi® micro USB socket, or External Power to **UPS PIco HV3.0 HAT,** or GPIO 5V) Power applied.

3. Only when External Cable Power applied (to Raspberry Pi® micro USB socket, or External Power to **UPS PIco HV3.0 HAT,** or GPIO 5V) the system will be restarted with "cold start" (using the last stored setup in the EEPROM). It will remain powered (the **UPS PIco HV3.0 HAT)**, even if Raspberry Pi® is not powered, and continuously monitoring the power conditions.

4. During External Cable Power powering (to Raspberry Pi® micro USB socket, or External Power to **UPS PIco HV3.0 HAT,** or GPIO 5V), battery can be connected or disconnected by user at any time. Only if battery is connected system is offering SD card protection, and UPS functionality.

5. If user wish to disconnect electrically the battery from the system, should press the **R** button for more than 2 seconds, after system FSSD (File Safe System Shutdown) with disconnected External Cable Power powering. This will cause an electrical disconnection of the battery from the system. Note that RTC will be not working after that. Restarting system in such condition need to apply External Cable Power powering (to Raspberry Pi® micro USB socket, or External Power to **UPS PIco HV3.0 HAT**, or GPIO 5V) again.

Power Monitoring Algorithm over GPIO (5V) pins (NEW in 0x50)

All **UPS PIco HV3.0 HAT** has implemented a powering monitoring algorithm in order to detect power loses or instability. This algorithm is monitoring every 16 us the 5V over GPIO pins (undependably if **UPS PIco HV3.0 HAT** is power via Raspberry's micro USB or Extended power input of the Pico). The feature that allows to monitor powering via GPIO offering a huge flexibility in enclosures selection as practically any enclosure can be used, as no extra holes are required.

This Power Monitoring Algorithm is parametrized, allows user to adopt to selected very specific powering needs. However, in 99.9% of cases user not needs any adoption, is such adaptation is required, there is embedded a mechanism helping to do that.

In addition in the firmware version 0x50 has been implemented algorithm that automatically adjust the proper values to weak Power Supply Units (supply cable is not proper, or current/voltage are too low). This automatic adjustment can not cover all PSU however is trying to set the best possible parameters for that. In a few cases when PSU is really bad, and can not be automatically adjusted, **UPS PIco HV3.0 HAT** is informing user by blinking LEDS, and beeps.

How Power Monitoring Works?

For power monitoring (final firmware version 0x50 and up) is used an automatic A/D conversion with thresholds, that is checking every 16 us (within an interrupt) the 5V GPIO pins. Power loses, or instabilities are continuously monitored and if are dangerous for system running are immediately recovered by battery backup. This back-up is working until powering of the system stabilize and automatically return to cable powering after shorter or longer time needed. When **UPS PIco HV3.0 HAT** detects power fluctuation automatically switches to ultra-fast 8us checking and monitor it within pre-defined time window.

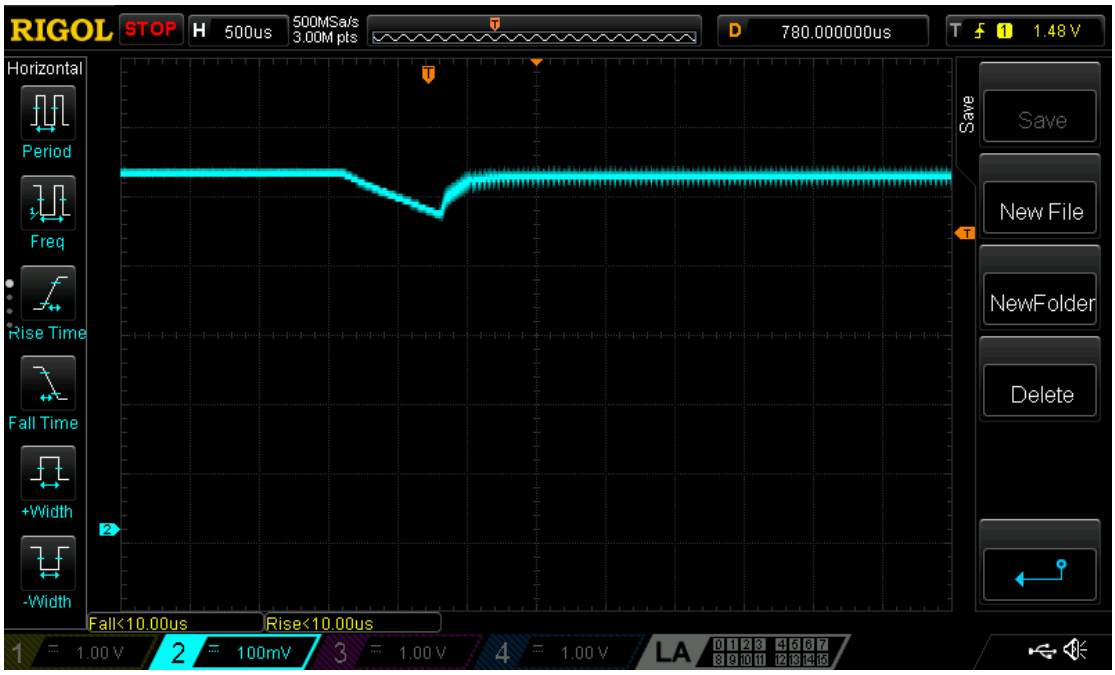


Figure 56 Typical Cable Power Loose Effect recovered by PIco HV3.0

Here above is shown typical power loose event, that has been recovered within 120 us. The dropping down powering voltage on GPIO pins was detected and filtered (from possible unwanted powering spikes), and in specific time recovered by switching ON to the 5V boost supplied form integrated battery. On below picture is visible (the yellow line) how **UPS PIco HV3.0** HAT was checking the powering status and when time frame passed switches immediately to battery powering backup.

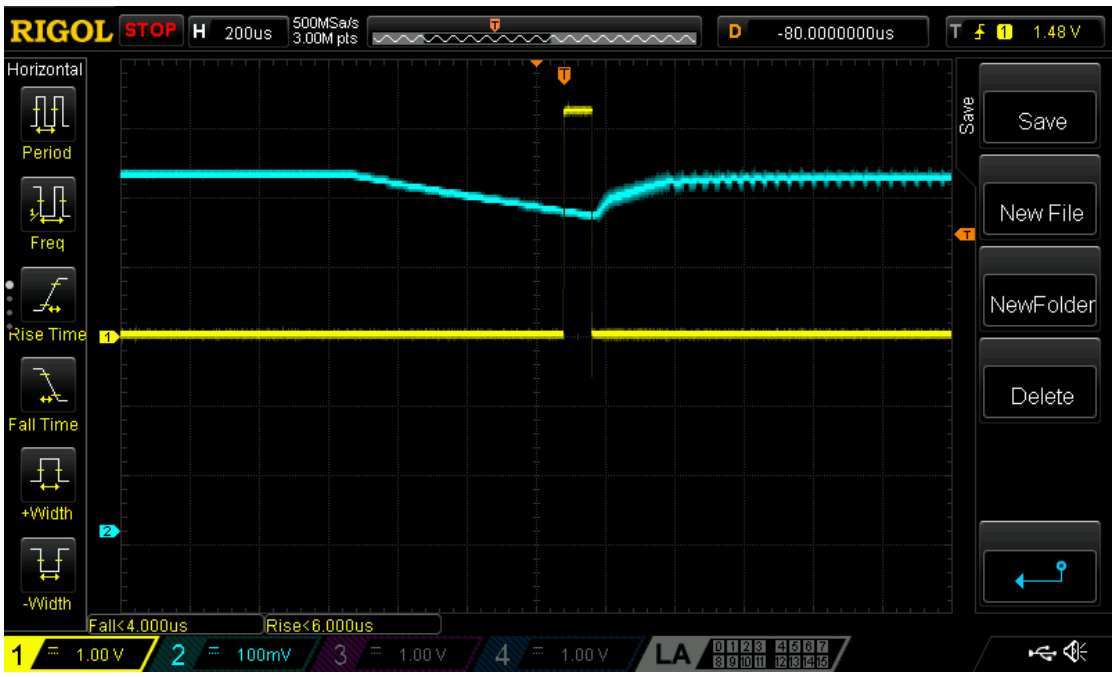


Figure 57 Pico HV3.0 Power Recovery timing

There are a set of parameters are embedded to support, easy recording, changing and analyzing of the power loose conditions.

These parameters should be only changed very experienced user, that are understand properly the Power Monitoring Algorithm, and are able to define properly the new conditions. For all other users, it is recommended to keep these parameters as they set by manufacturer. If for any reason, unwanted parameters have been set-up and user cannot make the system working properly again, user factory defaults feature (by I2C or buttons) in order to make system stable again.

In addition to cover non experienced user in the firmware 0x50 and up our company implemented an automatic adjustment algorithm that can be activated with dedicated command. During this automatic adjustment process, power back up is not offered, and used need to keep the system properly cable power via micro USB cable or via Extended Powering Cable (7-28VDC).

There are 2 types of Registers:

- **Monitoring Registers**, that are monitoring in real time the levels A/D when system is entering the A/D interrupt. They are storing the current measured in less than 1 us A/D value. However, measures are not computed (filtered), so it is possible that sometimes could be slightly different (due to noises). Multiply power cutting ensure you that results are measured properly. These Registers can be only read. Any writing to them makes no effect, as will be override with new values. The stored values remain until next event and can be read by user.
- **Executing Registers**, that are changing in fact the Algorithm Parameters, and adopting it to dedicated powering needs. These Registers should contain new values. It is recommended to change one by one values and make some experiments, checking the monitoring Registers in order be sure that new parameters complains with the new Powering needs or run automatic adjustment algorithm.

0x69 -> Monitoring Registers				
Address	**Name**	**Size**	**Type**	**Explanation**
0x02	rpi_cmp_level	word	Read	Comparator (activation) Level. This Variable is used to monitor on what exact voltage (measured without filtering) comparator has been activated. This value is stored when system detected cable power lose and stay in this register until next cable powering lose. It is in 10^{th} of volts. Can be read at any time to know exactly on what voltage level comparator has been activated.

0x04	rpi_swc_level	word	Read	Battery back-up switch Level. This Variable is used to monitor on what exact voltage (measured without filtering) Pico decided to switch to battery powering has been activated. This value is stored when system detected cable power lose and stay in this register until next cable powering lose. It is in 10th of volts. Can be read at any time to know exactly on what voltage level battery back-up have been activated.
0x6B -> Executing Registers				
Address	**Name**	**Size**	**Type**	**Explanation**
0x1A	pwr_spk_duration	byte	R/W	Powering spikes (gaps) filtering time duration. This variable is used to define the duration of powering gaps (spikes) that should be not taken in to consideration. Each number means 10 us. Default value is set to 250 us (number 0x19, or 25). User can increase/decrease this value. Depending of quality of PSU used. Power gaps shorter than defined value (i.e.0x0f = 150us) will be rejected
0x1B	pwr_rfc_level	byte	R/W	Monitoring Comparator Reference Voltage. This the voltage when monitoring comparator is firing. The following values can be stored: **0x00** – means disable the monitoring process, Pico power backup is disabled, and if cable power lost system will stop immediately. The embedded RTC will be not running then. All other functions are active and can be used. With this value Pico is not charging or using battery – battery is electrically disconnected by the system **0x01** – means that comparator will be activated if voltage lower than **4.918V** will be detected. It not means that Pico will switch to battery back-up just this voltage will fire the Powering monitoring Algorithm. However due to tolerance of voltage divider resistors used the real voltage can be slightly lower or higher. User by reading of the **rpi_cmp_level** will know the exact powering voltage level when the comparator has been fired. This value is not recommended due to possibility of many firings of comparator. The firing of comparator not necessary means that system will switch to battery back-up, switching to battery back-up requires more filtering parameters to be complied. **0x02** - means that comparator will be activated if voltage lower than **4.645V** will be detected. It not means that Pico will switch to battery back-up just this voltage will fire the Powering monitoring Algorithm. However due to tolerance of voltage divider resistors used the real voltage can be slightly lower or higher. User by reading of the **rpi_cmp_level** will know the exact powering

				voltage level when the comparator has been fired. This value is not recommended due to possibility of many firings of comparator. The firing of comparator not necessary means that system will switch to battery back-up, switching to battery back-up requires more filtering parameters to be complied. **This value is the default vale** **0x03** - means that comparator will be activated if voltage lower than **4.372V** will be detected. It not means that Pico will switch to battery back-up just this voltage will fire the Powering monitoring Algorithm. However due to tolerance of voltage divider resistors used the real voltage can be slightly lower or higher. User by reading of the **rpi_cmp_level** will know the exact powering voltage level when the comparator has been fired. This value is not recommended due to possibility of many firings of comparator. The firing of comparator not necessary means that system will switch to battery back-up, switching to battery back-up requires more filtering parameters to be complied. **0x04** - means that comparator will be activated if voltage lower than **4.096V** will be detected. It not means that Pico will switch to battery back-up just this voltage will fire the Powering monitoring Algorithm. However due to tolerance of voltage divider resistors used the real voltage can be slightly lower or higher. User by reading of the **rpi_cmp_level** will know the exact powering voltage level when the comparator has been fired. This value is not recommended due to possibility of many firings of comparator. The firing of comparator not necessary means that system will switch to battery back-up, switching to battery back-up requires more filtering parameters to be complied. Usually these values are slightly lower than calculate above. The exact values are known when user read the **rpi_cmp_level.**
0x1C	pwr_cut_level	byte	R/W	Maximum drop voltage allowed after firing the comparator, before battery backup is activated. Used together with **pwr_spk_duration** means that whatever will happens earlier will activate battery back-up. It is measured in mV and can be set from 0.1V up to 0.9V. A detailed graphical explanation is provided here below

				Default value is 0x03, means 0.3V voltage drop after firing of the Comparator.

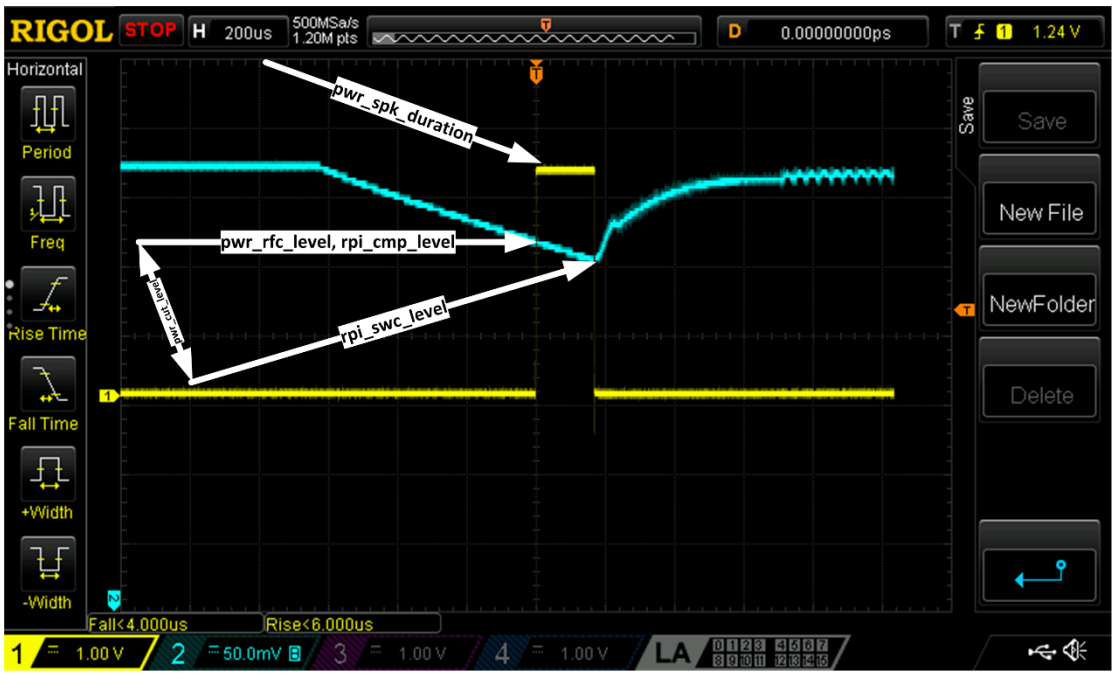


Figure 58 Power Monitoring Algorithm Explanation of Registers used

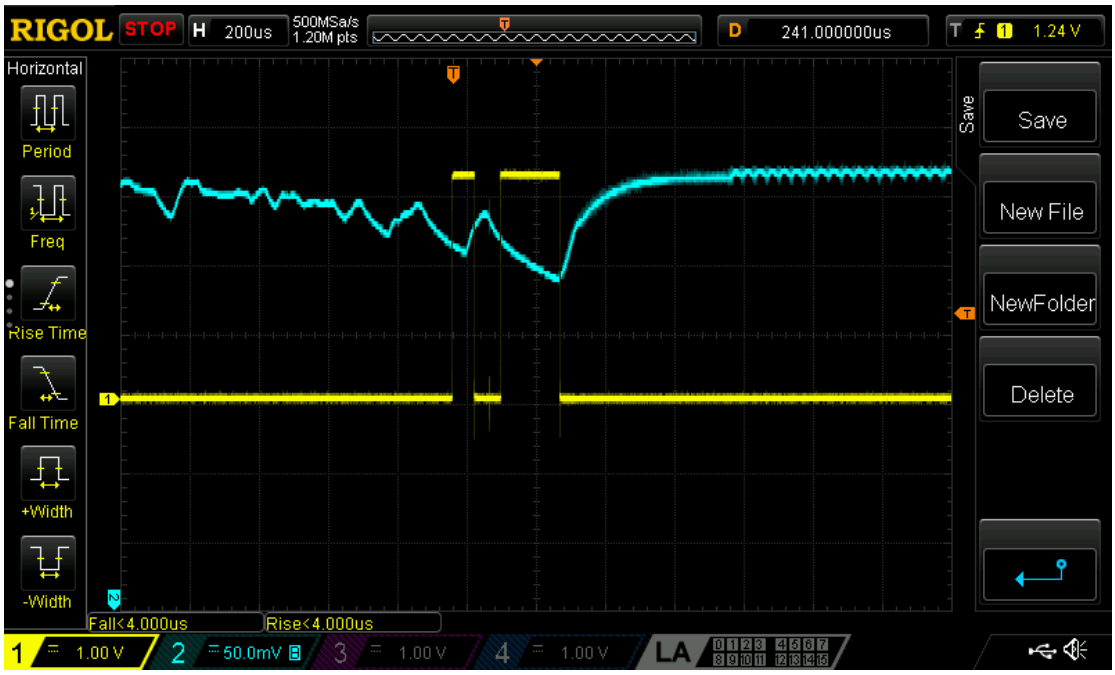


Figure 59 Power Monitoring Algorithm Filter short power gap before the valid power lose

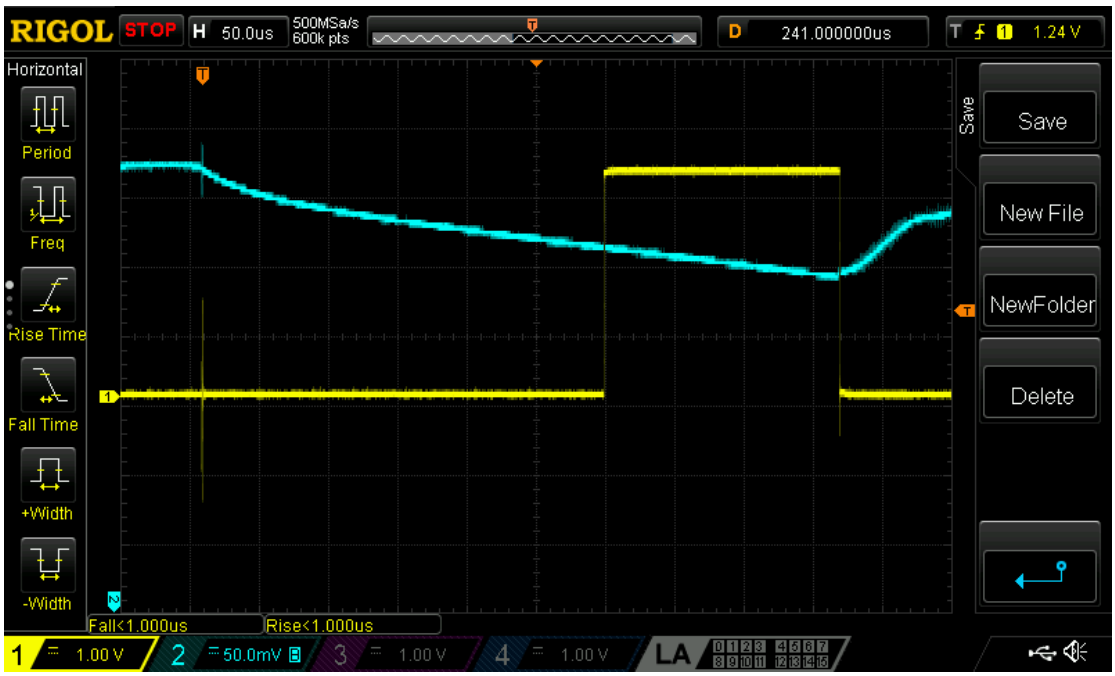


Figure 60 Power Monitoring Algorithm Flittered ultra-short power gap

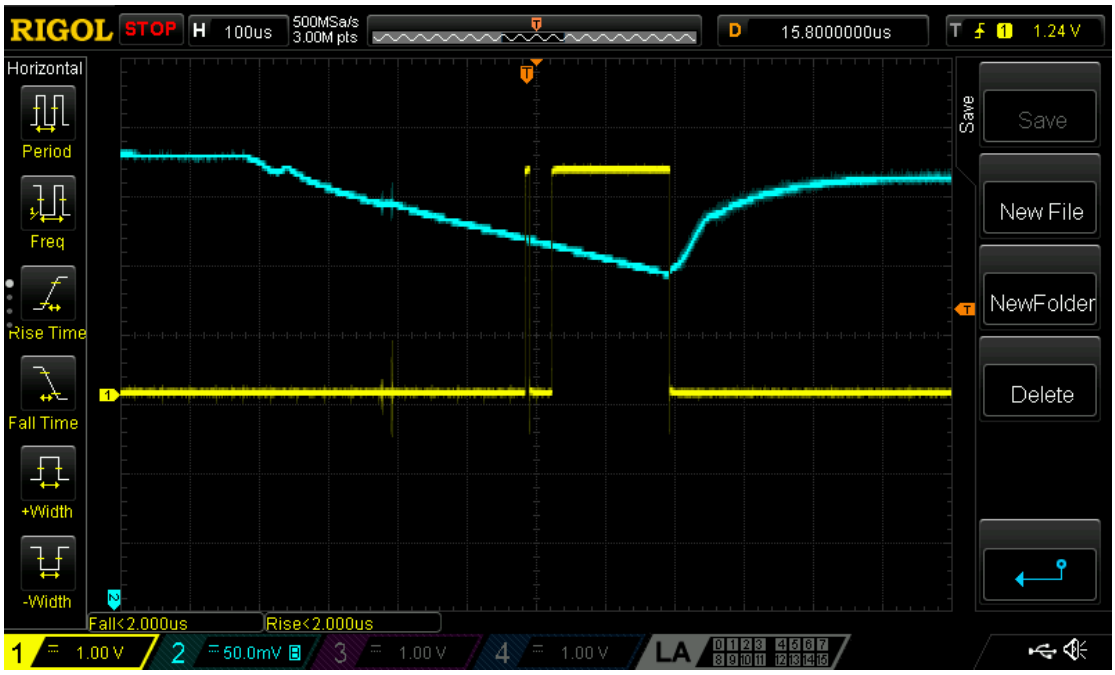


Figure 61 Power Monitoring Algorithm Flittered short power gap before the valid power lose

Example of use Monitoring Registers

sudo i2cget -y 1 0x69 0x02 w

returns ***450*** means that Comparator has been fired at 4.5V (rpi_cmp_level), after event

sudo i2cget -y 1 0x69 0x04 w

returns ***430*** means that battery back-up has been activated at 4.3V (rpi_swc_level), after event

Example of Settings of new values

Assume that we need system to be comparator fired at **4.645V**, filter power gaps shorter than 256 us, and switch to battery back-up, is powering voltage after comparator firing drops for another 0.4V. Therefore, PIco will filter any gaps that are shorter that 256 us, and if not higher that (4.645V – 0.4V) 4.245V. Whatever fluctuation on powering lower than 4.645V, that takes longer than 250us, or continuously dropping down to lower than 4.245V (even if are shorter than 256us) will cause activation of battery power back-up.

sudo i2cset -y 1 0x6b 0x1a 0x25 pwr_spk_duration set to **250us**

sudo i2cset -y 1 0x6b 0x1b 0x02 pwr_rfc_level set to **4.645V**

sudo i2cset -y 1 0x6b 0x1c 0x04 pwr_cut_level set to **0.4V**

With below alternative setup, system will more frequently fired the comparator (due to spikes), however will return to normal work immediately. Just when during the testing power gap duration power drop down to less than 4.518 battery back-up will be activated

sudo i2cset -y 1 0x6b 0x1a 0xff pwr_spk_duration set to **2500us**

sudo i2cset -y 1 0x6b 0x1b 0x01 pwr_rfc_level set to **4.918V**

sudo i2cset -y 1 0x6b 0x1c 0x04 pwr_cut_level set to **0.4V**

With above alternative setup, system will more frequently fired the comparator (due to spikes), however will return to normal work immediately. Just when during the testing power gap duration power drop down to less than **4.518V** battery back-up will be activated

Disabling the UPS Pico HV3.0 HAT Battery Back-up functionality

The **UPS PIco HV3.0 HAT** in some cases (mainly when located/installed remotely) need to be disabled (permanently or temporary) the Battery Back-up functionality. The current firmware offers this feature. When battery backup disables, Raspberry Pi® is then not protected on cable power lose. If such cable power loss happens, then Raspberry Pi® will stop working immediately. It is possible that micro SD card will be destroyed. Therefore, users using this feature need to do it very carefully. This battery back-up feature does not imply other **UPS PIco HV3.0 HAT** functionality like bi Stable Relay, A/D converter, buzzer, User LEDs etc. The embedded RTC will be not working in case of disabling of battery back-up feature.

If this feature will be activated, the integrated battery will be electrically disconnected from the system.

Example of use

sudo i2cset -y 1 0x6b 0x1b 0x00 disable the battery back-up feature

sudo i2cset -y 1 0x6b 0x1b 0x02 (from 0x01…0x04) enable the battery back-up feature

The Magic ON/OFF (Slide) Switch functionality

The newer version of **UPS PIco HV3.0B/B+ HAT** (Version B and B+) is equipped with an **ON/OFF Magic Slide Switch**. This Switch is called Magic, as it is multifunctional, programmable, and adding a huge difference in powering schemes of the Raspberry Pi®. It is allowing to use Raspberry Pi ® as an independent device powered exclusively from battery, without any cable powering source. This powering feature of **Intelligent Power Bank**, is called "**on the Go**"

First, the **Magic Switch** is not soldered by default, and it is not necessary to be soldered. It is required only if user is planning to use the **UPS PIco HV3.0B/B+ HAT** as a mobile battery powered application. In all other cases, absence of this switch does not imply the standard functionality of the **UPS PIco HV3.0B/B+ HAT.** If for any reason, the **ON/OFF Magic Switch** has been soldered, but user not need his additional functionality, should be placed on position **OFF**. Also, if user need to have external access to Intelligent ON/OFF functionality, then an external ON/OFF Switch will longer cables can be soldered instead of the micro switch.

The following functionalities are assigned to the **Magic Switch**:

- Intelligent ON/OFF with Files Save Shutdown when OFF and battery power cut (Integrated RTC is not running, schedulers are not running)
- Intelligent ON with Files Save Shutdown when OFF without battery power cut (Integrated RTC is running, schedulers are running)

Here below are shown the **Magic Switch** functionalities based on Switch position and programming registers values. The following PICo registers are associated to the **Magic Switch** functionality:

Magic Switch State	Assigned Functionality	Programming Register values	UPS Pico HV3.0 HAT Version	Initial State	System Powering Behaviors
Absent or not soldered	Standard Power Cycling. System must be initialized with inserting of the powering cable to Raspberry Pi® micro USB or to EPR input (7-28V) for the UPS Pico HV3.0A/B/B+ HAT	Not Affected	UPS Pico HV3.0A HAT	Default State	Standard FSSD functionality and Low Powering as described in other chapters
		Do not program any value (keep with default value of 0x00)	UPS Pico HV3.0B/B+ HAT	Default State	Standard FSSD functionality and Low Powering as described in other chapters
Soldered original slide micro switch or external slide switch in position OFF	Standard Power Cycling. System must be initialized with inserting of the powering cable to Raspberry Pi® micro USB or to EPR input (7-28V) for the UPS Pico HV3.0 HAT	Not Affected	UPS Pico HV3.0A HAT	Default State	Standard FSSD functionality and Low Powering as described in other chapters
		Do not program any value (keep with default	UPS Pico HV3.0B/B+ HAT	Default State	Standard FSSD functionality and Low Powering as described in other chapters

		value of 0x00)				
Soldered original slide micro switch or external slide switch in position OFF	File Safe Shutdown procedure initiated	Not Affected on_the_go=0xAA	UPS Pico HV3.0A HAT UPS Pico HV3.0B/B+ HAT	Default State	Standard FSSD functionality and Low Powering as described in other chapters	

Setting-up the Magic Switch

With Cable powering connected do the following

1. Make sure that Magic Switch is on Position ON (on the LEDs side)

2. Check the Magic Switch Register

    i2cget -y 1 0x6b 0x16

    should be 0x00 or 0xff

3. Then program it

    i2cset -y 1 0x6b 0x16 0xaa

4. Remove Powering Cable

5. It is still running, but on battery mode

6. Switch the Magic Switch to OFF position

7. System start shutdown, and after short time cut the power

8. Whenever you like can make it ON/OFF (of is always with system shutdown first)

IMPORTANT NOTICE: When the **Magic Switch Register** is programmed, the battery running register is automatically set to 0xff, so system is running on battery until it is low, then system automatically shut down. If you like to have it running shorter, you need to reprogram the register for a shorter time with just after Magic Switch programming

Using of cable powering when programmed is the Magic Switch is not allowed with current version of firmware, except of the first time, as described above.

UPS PIco HV3.0 Battery Type/Profile Selection

The **UPS PIco HV3.0** is supporting the following chemistry battery types:

- the **LiPO** with nominal voltages 3.7V and charging 4.2V
- the **LiFePO4** with nominal voltages 3.2V and charging 3.65V
- the **Li-Ion** with nominal voltages 3.65/3.7V and charging 4.2V
- the **NiMH/NiCd** with nominal voltages 3.6V (3 cells) and charging 4.2V
- the **SAL** (Lead Acid) with nominal voltages 3.6V (3 internal cells) and charging 4.2V

The **UPS PIco HV3.0** is sold by default with the small LiPO 450 mAh battery. However, for dedicated applications user can choose one of the other batteries' chemistry supporting by **UPS PIco HV3.0**. The main differences between various batteries chemistries are:

- Charing temperature range
- Discharging Temperature Range
- Number of charging/discharging cycles
- Battery Cost
- Power density (how big is battery)
- C factor (how much current battery can provide when discharging)

Battery Chemistry	Charing temperature range	Discharging Temperature Range	Number of charging/discharging cycles	Power density
LiPO	0°C to 40°C	0°C to 40°C	400-450	Very high
LiFePO4	0°C to 40°C	0°C to 45°C	2000	High
Li-Ion	0°C to 45°C	–20°C to 60°C	300-400	Highest
NiMH/NiCd	0°C to 45°C	–20°C to 65°C	200	low
SAL (Lead Acid)	–20°C to 50°C	–20°C to 50°C	200	Very low

From Electrical point of view the most important factors for battery are the working temperature and C factor. These two factors are defining if the battery is good for dedicated application or not. The working temperature (Charging/Discharging) is defined based on application working conditions. User need to consider all scenarios to be sure that battery will be all the time within the specified temperature conditions. In example, if system is

placed in an isolated plastic case, and temperature (due to Raspberry Pi) is all the time increasing, then the internal temperature (so also the battery) will be high. Especially for Lithium based batteries temperature is a very important factor. The second and very important is the C factor.

Maximum Current supplying by battery (C Factor)

The C factor for any battery defines how much current can be drawn from battery when discharged. It is different for each battery chemistry and type. In example the standard 450 mAh battery that comes with **UPS PIco HV3.0** are **15C**. That means the max current provided by this battery when discharged is 15 x 0.450A=6.75A!!! Indeed, this small battery can provide near to 7A of current.

User Application Current consumption calculation example

When designing battery powered system application (even with battery backup), it is very important to know/estimate the power/current requirements. The below calculations need to be used as a guideline for any Battery Powered Application. For the calculations we will make some assumptions that can be adjustment to specific User Application if/when needed.

- Raspberry Pi 3 (any model) draws current of minimum 400 mA@5V, if not running any dedicated software. If running more advanced software current consumptions will increase.
- USB devices (HDD, LTE, GSM, etc) connected to the Raspberry Pi draws their current @5V. User need to take into considerations the maximum (peak) current required by this hardware.
- Our calculations will be done for the standard LiPO 450 mAh battery of 15C
- For our calculation we assume that Raspberry Pi and all other connected devices draw totally 1A@5V

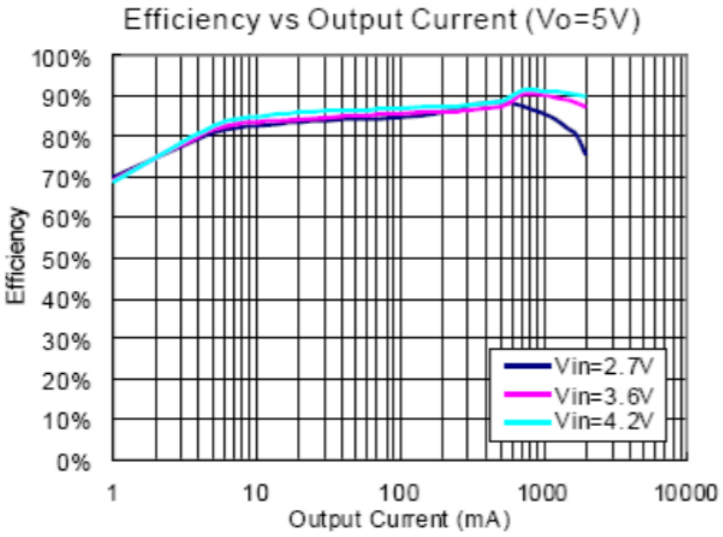


Figure 62 UPS PIco HV3.0 Boost Converter Efficiency

Calculation Algorithm

- Calculate the total power required for 5V.

For our example it is 5V x 1A = **5W @ 5V**

- Calculate the total power required if system is powered by 3.7V (in our example LiPO battery)

5WA/3.7 V = **1.35 A @ 3.7V**

- Calculate the power losses due boost converter used. The **UPS PIco HV3.0** for generating the 5V from battery is using technology called boost converter that converts the battery voltage to 5V. We use the best technology provided in the market, however this conversion always causes a power loss that are loosed on thermal (a part of supplied energy is converter to thermal). The Boost converter loses are changing depending to battery voltage. In example when battery is full changed, and their voltage is 4.2V, loses are small (about 10%) if voltage drop down when battery is discharging, loses increases. We propose to use mean value of 20% for 3.7V and 25% for 3V. Therefore, for our calculations case we will use 20% of thermal loses (called also boost converter efficiency – indicated with Greek letter of "**η**"). We need to multiply the current consumption by loses in order to know the max current requirements from battery.

1.35 A * 1.2 = **1.62A @ 3.7V** when battery powered, or **5.99 W @3.7V**

- This result means that if we are powering Raspberry Pi that totally uses at 1A @5V (5W) we need to supply by battery 1.62A @ 3.7V.

As our battery is 0.45 A 15C can supply the system with 1.62A @ 3.7V. (max current of this battery is 0.45A x 15 = 6,75A)

- This result allows us to easy calculate how long last 0.45 A 15C 3.7V battery when system is powered by this battery.

It is 0.45A/1.62A = 27 minutes -> **15-20 minutes** of real working time (please always use lower numbers, as battery is not going to be completely discharged, we have also some loses on battery cables, PCB tracks, MOSFET switches etc.)

Selecting of a proper battery for your application is one of the core requirements when designing a new system.

User can select one for supported batteries, however always need to consider the worst case for current consumption and working temperature. In addition, need to use **22AGW** (existing already in all of our batteries or battery holders) cables and relatively short to avoid voltage drops on them.

Our company is offering a plenty of different batteries that are supporting the **UPS PIco HV3.0** applications.

We always offer the best quality batteries with the best prices. There are 4 new add-ons for that can hold different batteries chemistry supply. They are:

- PIco Double Li-Ion 18650 Battery Holder
- PIco Single LP/LF, Li-Ion 18650 Battery Holder
- PIco triple NiHM/NiCd AA Battery Holder
- PIco triple NiHM/NiCd AAA Battery Holder

In addition, our company supports with various batteries capacities and various chemistries, that can be directly connected to the **UPS PIco HV3.0**

There are:

- The standard LiPO battery 450 mAh which comes with the **UPS PIco HV3.0** – 15C
- The enhanced LiPO battery with capacity 4000 mAh - 2C
- The enhanced LiPO battery with capacity 8000 mAh – 2C
- The enhanced LiFePO4 battery with capacity 3000 mAh 2C
- The enhanced LiFePO4 battery with capacity 4000 mAh 2C
- The enhanced LiFePO4 battery with capacity 8000 mAh – 2C

- The Li-Ion Battery 10400 mAh – 2C
- The LiPO battery with capacity 1500 mAh – 2C
- More batteries types and chemistries are on the way

Batteries with different chemistry offers different unique features and needs to be specified on the system setup when changed. This ensure that **UPS PIco HV3.0** will adjust the proper charging/working profile. It is not allowed to use different batteries with not related profile to them. There is no protection from this action, therefore using of a wrong profile can cause an unexpected result. Exceptions on this rule is LiPO and Li-Ion batteries where differences are very small, however it is also to them recommended to use a proper battery profile

0x6B -> UPS PIco Module Commands

0x07	battype	Byte	Common	R/W	Defines used battery chemistry type: 0x53 – LiPO (ASCII : **S**) used in version Stack/Top-End 0x46 – LiFePO4 (ASCII : **F**) used in version Stack/Top-End 0x49 – Li-Ion (ASCII : **I**) used in version Stack/Top-End 0x48 – NiMH (ASCII : **H**) used in version Stack/Top-End 0x4C – SAL (ASCII : **L**) used in version Stack/Top-End 0x50 – LiPO (ASCII : **P**) used in version Plus/Advanced 0x51 – LiFePO4 (ASCII : **Q**) used in version Plus/Advanced 0x4F – Li-Ion (ASCII : **O**) used in version Plus/Advanced 0x4D – NiMH (ASCII : **M**) used in version Plus/Advanced 0x41 – SAL (ASCII : **A**) used in version Plus/Advanced **Other codes are not allowed**

By selecting battery type that **UPS PIco HV3.0** will adjust the proper charging/working profile. Below table shows the Low Battery and Full Battery Threshold for each type, implemented in the current firmware version.

Battery Type	Low Battery Threshold	Full Battery Threshold
LiPO	3.5V	4.2V
LiFePO4	2.95V	3.6V
Li-Ion	3.4V	4.2V
NiMH/NiCd	3.2V	4.2V

SAL (Lead Acid)	3.1V	4.2V

Example of use

sudo i2cset -y 1 0x6b 0x07 0x46 LiFePO4 (ASCII : F) used in version Stack/Top-End

sudo i2cset -y 1 0x6b 0x07 0x51 LiFePO4 (ASCII: Q) used in version Plus/Advanced

sudo i2cset -y 1 0x6b 0x07 0x53 LiPO (ASCII: S) used in version Stack/Top-End

sudo i2cset -y 1 0x6b 0x07 0x50 LiPO (ASCII: P) used in version Plus/Advanced

Caution: The **UPS PIco HV3.0 HAT** PCB has declared always the default battery chemistry type, and when firmware update is executed the battery chemistry is always changed to PCB defaults, therefore it is needed to re-declare the battery type after firmware update to the used one by the system. Some industrial customers have default declared the LiFePO4 in their systems, but usual the default battery chemistry is LiPO.

Battery Charger Monitoring and Control

The Battery Charger is controlled by **UPS PIco HV3.0 HAT** automatically, and when needed is switch ON or OFF. The state when **UPS PIco HV3.0 HAT** is switched charger ON/OFF can be monitored on the Charger Status Register that is located at the address **0x69 0x20**. A detailed description of it is provided here below.

0x20	charger	Byte	Mirror	Read	Information about charger IC status. **Read: 0x00** – Charger IC is OFF and battery is not charged **Write: 0x01** – Charger IC is ON and battery is charged For Version UPS PIco HV3.0 Stack/TopEnd the charging current is fixed to 300 mA. For Version UPS PIco HV3.0 Plus the charging current is dynamically changed based on powering conditions on micro USB powering input or External Powering Input. The charging current on that version is from 100 mA – 800 mA. With maximum of the system to be 1200 mA (will be activated in the future). If the **charger** register is set ON, meant that charger circuits has been activated, however if battery is really charged depends to the internal conditions of the charger IC. When current is flowing to the battery (charging) then CHG LED is lighting continuously. It is possible that charger register can be read as 0x01, but CHG LED will be off, because system set to charge battery however buttery is full, so no current is flowing. The CHG LED always shows if current is flowing to battery or not. This is valid for all batteries chemistry types.

Some very specific applications, requiring having the ability to disable the charger, and enable only when some specific conditions met. In order to cover such applications a special Charger Control Register has been implemented. With this Register user can Enable/Disable charging feature. This Register us placed at the address **0x6b @ 0x1d** and can be set only when system is cable powered.

Example of use

sudo i2cset -y 1 0x69 0x20 Read the current battery charger status (when on automatic mode)

sudo i2cset -y 1 0x6b 0x1d 0x00 Disable the battery charger

sudo i2cset -y 1 0x6b 0x1d 0x01 Enable Automatic Mode of the battery charger

Low Battery threshold changing

By selecting battery type (chemistry) **UPS PIco HV3.0 HAT** automatically sets the threshold for charging as also the threshold for the system shutting down on low battery. However, on some cases, user need to change this battery low threshold. It can be done by writing to a specific PICo register. The following values are possible to be set.

0x6B -> UPS Pico HV3.0 Module Commands

low_bat address **0x1E** (byte)	**Referred battery low voltage threshold**
0	2.9V
1	2.95V
2	3.0V
3	3.1V
4	3.2V
5	3.3V
6	3.4V
7	3.5V
8	3.6V

Example of use

sudo i2cget -y 1 0x6b 0x1e Read the current low battery threshold

sudo i2cset -y 1 0x6b 0x1e **0x05** set it to 3.3V

Low Battery LED and Beeper

By selecting battery type (chemistry) **UPS PIco HV3.0 HAT** automatically sets the threshold for charging as also the threshold for the system shutting down on low battery. There is an information on user by switching ON the BAT LED and Start regular Beeping. This level is slightly higher of any low battery shutdown threshold selected by system or user. It is 0.06V higher than level of low battery level selected. When battery is low, user will see the BAT LED lit, and short beeping, after a short time, system will be automatic shutdown

Powering Modes

UPS PIco HV3.0 HAT powering selection and monitoring functionality is based on internal firmware-based **state machine**. This state machine is deciding on Powering State (called also Powering Mode) based on various parameters like powering source, battery level, current level, RTC etc. The actual Powering Mode each time is stored in internal register and can be accessed by PICo interface over address 0x69, location 0x00.

The following Powering Modes are available:

- RPi (which consist both sub modes EPR (7-28VDC) and RPi (5V))
- BAT (which consist both sub modes BAT and LPR)

User can at any time check the powering mode the system is from command line, or software interface, remotely or on site. The meaning is:

- 0x01 Powering from Cable (Raspberry Pi® or External)
- 0x02 Powering from Battery

Example of use

sudo i2cget -y 1 0x69 0x00

User should receive response 0x00 or 0x01.

UPS PIco HV3.0 HAT Low Powering functionality – Power Cycling

One of the most important features of the **UPS PIco HV3.0 HAT** is the **Power Cycling**. Power Cycling as specified before is the core firmware State Machine that is handling the whole system powering behaviors. The Power Cycling feature is handling the System Shutdown, System Start-up as also battery charging.

The following scenarios has been implemented in the current firmware version that are covering 100% of possible cases.

Raspberry Pi® Shutdown Scenarios

The **UPS PIco HV3.0 HAT** is entering Low Powering Mode in the following scenarios

UPS PIco HV3.0 Stack/Top-End

The Raspberry Pi based System running scenario is when System is powered by cable (via micro USB interface). If this cable powering fails system is automatic switching to the battery back-up and System (based on Raspberry Pi, PIco and possible additional hardware)

continues working. This change of powering source is indicated to user by "beep", as also by changing of the UPS LED blinking speed (which is going to blink much slower than with a cable powering). User can indicate the powering mode by reading the ***sudo i2cget -y 1 0x69 0x00*** and see that is has been changed to battery powered. If this condition continues (lack of cable supply power) system after 70 seconds (user can program the battery running time) will initiate the **F**ile **S**afe **S**hut **D**own Procedure (FSSD) and as a result System will enter to the Low Powering Mode – **LPR**. Another way to entering the **LPR** mode is by pressing the **F** button. This will active immediately the **FSSD** procedure, and System after a required time to shutdown will enter the **LPR** mode.

During the **LPR** mode, the embedded RTC will continue working. The wake-up from the LPR mode can be done in the following ways:

- By pressing the **F** button
- By Applying the Cable Power to micro USB socket (or changing the powering conditions)
- By Event from the Basic Scheduler or SAS ETR scheduler

Due to ultra-low power implementation recognizing of the Cable Power re-entering is done every 10 seconds. Therefore, the longest time when re-entering cable will be recognized is 10 seconds, after this time system will be restarted and continue working.

It is not needed to re-enter the power cable during the **LPR** mode to restart the System. User can just press the **F** key and system will wake-up and run on the battery power. On this case (due to specific interrupt-based **F** key implementation) the wake-up will be immediate.

UPS PIco HV3.0 Plus/Advanced

The **UPS PIco HV3.0 HAT Plus/Advanced** has an additional Cable power input the EPR (7-28V DC). Therefore, there are 2 ways of cable powering of the System:

- The Raspberry Pi micro USB socket, and
- The EPR input (7-28 VDC)

As far the System is powered via micro USB Raspberry Pi socket, the **UPS PIco HV3.0 HAT Plus/Advanced** behavior in the same way with the version **Stack/Top-End**. However, if powered via EPR cable input (7-28V DC), there are some changes in cable power handling. The powering scenario is that system is powered i.e. from 12 V source. If the EPR cable power source will be absent, System will automatic change the powering to Battery, and inform about its user in a similar way when powered by micro USB via Raspberry Pi. If absence of EPR cable powering will continue, System after 70 seconds (user can program the battery running time) will proceed with **FSSD**, and then enter to the **LPR** mode. However, there is also an additional possibility to shut down the system, without cutting the EPR cable power. User can use the **F** key, and just press it. Then the system with proceeded with **FSSD**, and after that enter the **LPR** mode. When entering the **LPR** mode, **UPS PIco HV3.0 HAT**

Plus/Advanced will cut the system powering even if cable power is connected. This functionality allows to have ON/OFF cable powered system with running RTC.

User can also change powering conditions, by removing the EPR Cable, plug on the micro USB cable or re-entering the EPR Cable. On all that cases, due to ultra-low power implementation recognizing of the Cable Power re-entering is done every 10 seconds. Therefore, the longest time when re-entering cable will be recognized is 10 seconds, after this time system will be restarted and continue working.

During the **LPR** mode, the embedded RTC will continue working. The wake-up from the LPR mode can be done in the following ways:

- By pressing the **F** button
- By Applying the Cable Power to micro USB socket (or changing the powering conditions)
- By Applying the Cable Power to EPR power input (or changing the powering conditions)
- By Event from the Basic Scheduler or SAS ETR scheduler

System Information – SysInfo

System information information's about running/booting/shutting down of the **UPS PIco HV3.0 HAT** is important for Application Developer. They are stored on the Pico Register Called **SysInfo** located at address **0x69** and position 0**x28.** It is 16 bits wide, and there are bitwise allocated to different meaning.

0x28	**SysInfo**	word	Mirror	Read	Read the System Information **Read:** 0x---X bits 3:0 Means System FSSD Reason: 0x1 - FSSD button 0x2 - low battery 0x3 - Timed FSSD 0x4 - Timed Simple Scheduler 0x5 - Timed ETR Scheduler 0x6 - Event **Read:** 0x--X- bits 7:4 Means System Wakeup Reason: 0x1 - FSSD button 0x2 – RPi Voltage Applied 0x3 - Running RPI (reset/reboot) 0x4 - EPR Voltage Applied 0x5 - Timed Simple Scheduler 0x6 - Timed ETR Scheduler 0x7 - Event **Read:** 0x-X-- bits 11:8 Means Pico Restart Reason: 0x0 - RESTART_POWER_UP 0x1 - RESTART_BROWNOUT 0x4 - RESTART_WATCHDOG 0x6 - RESTART_SOFTWARE 0x7 - RESTART_MCLR 0xE - RESTART_ILLEGAL_OP 0xF - RESTART_TRAP_CONFLICT OR OTHER **Write:** 0x0000 – Clearing the variable

Table 9 UPS PIco HV3.0 HAT SysInfo PIco Register

Example of use

sudo i2cget -y 1 0x69 0x28 w to get stored data

sudo i2cset -y 1 0x69 0x28 0x0000 w to clear stored data after read

"PIco is Running" Feature

Many users are using the Raspberry Pi® in remote places where is difficult to access the UPS LED and see it blinking. Therefore, is needed to check and confirm that **UPS PIco HV3.0A HAT** is running and protecting the Raspberry Pi®. For that reason, a dedicated PIco register has been implemented that allows remote user to proof that PIco is working properly. This register is placed on the I^2C address 0x69 at location 0x22. If the **UPS PIco HV3.0A/B/B+ HAT** is working (properly) this register value is updated every 1 millisecond. To proof that **UPS PIco HV3.0A/B/B+ HAT** is working need to read 2 times with time difference bigger than 1 millisecond. The read values need to be different.

Example of use

i2cget -y 1 0x69 0x22 w && i2cget -y 1 0x69 0x22 w

User should receive response like this (two different 16-bit numbers)

0x823f

0x8247

UPS PIco HV3.0 HAT Still Alive (STA) Functionality

The **UPS PIco HV3.0 HAT**, offers to the user a protection mechanism for the possibility of the Raspberry Hang-up (freeze of it). In a case that Raspberry Pi® freeze, the **UPS PIco HV3.0 HAT**, will automatically hardware reset it, using Gold Plated Reset Pin (POGO Pin) that must be soldered to have such functionality. The default state is that Still Alive functionality is disabled.

The Still Alive functionality is based on 8-bit timer located at address **0x6b** and position **0x05** that his value is decreasing every second when his value is different from 0xff. If it reaches 0x00 **UPS PIco HV3.0 HAT** resets hardware the Raspberry Pi®. The default value after restart/start of the **UPS PIco HV3.0 HAT** is 0xff (disabled).

To activate it, user need to write to this register value different than 0xff, and rewrite new value every defined time (by its written value).

The following options are available, and can be used at any time:

- Writing of 0xff cause disable of this STA timer
- Writing of 0x01 – 0xfe cause start of down counting (every second) of this STA timer until it reaches the 0x00 when the Raspberry Pi® will be hardware Reset
- Writing of 0x00 cause immediate and unconditional Raspberry Pi® hardware Reset

Example of use

sudo i2cset -y 1 0x6b 0x05 0x00 unconditional resets the Raspberry Pi®

sudo i2cset -y 1 0x6b 0x05 0x0f Sets the STA timer to 15 seconds (if within 15 seconds software running on Raspberry Pi® not write new values, Raspberry Pi® will be hardware reset by PIco.

sudo i2cset -y 1 0x6b 0x05 0xff disable the STA timer

User Selectable PIco HV3.0 I2C addresses

The **UPS PIco HV3.0** interacts with Raspberry Pi® via I²C interface. There is pre-selected (default) addresses that are used by **UPS PIco HV3.0 HAT**. However, user may need to change them to adopt the system to different address area in their application. In addition, the integrated Hardware RTC may be not used and the address of the 0x68 that is assigned to it, will be used by another external RTC provided by user. The **UPS PIco HV3.0** offers a mechanism that allows to change this addresses to different ones.

The following addresses are available:

- **DEFAULT** where used I2C addresses are: 0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F
- **NO_RTC** where used I2C addresses are: 0x69, 0x6B
- **ALTERNATE** where used I2C addresses are: 0x58, 0x59, 0x5A, 0x5B, 0x5C, 0x5D, 0x5E, 0x5F

Addresses DEFAULT	Addresses NO_RTC	Addresses ALTERNATE	Address Usage
0x68	not used	0x58	Used for UPS PIco HV3.0 Hardware RTC. If RTC is activated in the Raspberry Pi will be visible as UU
0x69	0x69	0x59	Used for system monitoring
0x6A	not used	0x5A	Contains RTC registered accessible independently by user
0x6B	0x6B	0x5B	Used
0x6C	not used	0x5C	Used for the RTC Scheduler
0x6D	not used	0x5D	Used for the RTC Scheduler
0x6E	not used	0x5E	Used for the RTC Scheduler
0x6F	not used	0x5F	Used for the RTC Scheduler

Table 10 UPS PIco HV3.0 HAT I2C addresses

Due to limited memory footprint user need to load different firmware for selected I²C address area. Therefore there are 3 different firmware released each time:

- **DEFAULT**
- **NO_RTC**

- **ALTERNATE**

The former option to change “only the fly” the available I2C addresses has been removed. However user still keep the possibility to use (as before) the same I2C addresses, just by loading different firmware

UPS PIco HV3.0 HAT User Applications Hardware Interfaces

The **UPS Pico HV3.0 HAT** is equipped with a set of **User Applications Hardware Interfaces** that allows to rapid setting-up of various applications without necessity of other additional HAT HATs (PCBs). It contains:

- System and User LEDs
- System and User Buttons
- Sound Generation System
- Bi-Stable Relay
- Auxiliary 5V@750mA and 3.3V@150mA interface
- IR Receiver Interface
- Programmable Auxiliary 5V@750 mA and 3.3V@150 mA Powering Sources
- RS232 Interface dedicated to wake-up on command
- User Selectable PIco HV3.0 I^2C addresses (NORMAL, NO_RTC, ALTERNATE)
- ESD protected 1-wire Interface
- Opto-Coupler interface

UPS PIco HV3.0 HAT LEDs

The **UPS Pico HV3.0 HAT** is equipped with 9 LEDs (that provides information about the **UPS Pico HV3.0 HAT** system status and user information). There are 6 System LEDs and 3 User Application LEDs. The System LEDs are described here on below table:

System LEDs Indications		
UPS LED		
	OFF	System is not running or is in Low Power Mode (only HW RTC is running)
	Lighting continuously	System (PIco + RPi) is booting or shutting down
	Blinking every 400 ms for 400 ms	System (PIco + RPi) is running on cable powering (after booting time)
	Blinking every 1200 ms for 400 ms	System (PIco + RPi) is running on battery powering
BAT LED		

	OFF	Battery level is **above** warning thresholds: - For LiPO Battery **3.5V** - For LiFePO4 **2.95V**
	Lighting continuously	Battery level is **below** warning thresholds: - For LiPO Battery **3.5V** - For LiFePO4 **2.95V**
CHG LED		
	OFF	Battery is not Charged (full)
	Lighting continuously	Battery is Charged (and current is flowing to the battery) If battery is Full, even if Charger is ON, current is not flowing to the battery, then CHG LED is OFF
INF LED (former HOT LED)	Blinking for a short time	This LED is used for various Information purposes, mainly when Time Schedulers are used
FAN LED		
	OFF	FAN is not running
	Lighting continuously	FAN is running
EXT LED		
	OFF	External Cable powering is disconnected (7-28VDC)
	Lighting continuously	External Cable powering is connected (7-28VDC)

Table 11 System LEDs description

The User LEDs are dedicated for user applications and can be handled by the **PICo** (I^2C) interface. One of them is **Orange**, the second one is **Green**, and the third is **Blue.**

In the **UPS Pico HV3.0B/B+ HAT** there is an additional option to use external **LEDs** connected in parallel with the **User LEDs**. Thus, allows user to make their own applications where Pico User LEDs will be used.

Connectivity of external User LEDs can be done via cables using the 2mm header as shown her below. It is recommended to use in series a resistor (of 250 Ohm), however in any case the Pico LED resistor is used in this connectivity.

Accessing of the User LEDs can be done by the following **PICo** Commands.

Example of use

sudo i2cset -y 1 0x6b 0x09 0x01 for ON the Orange LED

sudo i2cset -y 1 0x6b 0x09 0x00 for OFF the Orange LED

sudo i2cset -y 1 0x6b 0x0A 0x01 for ON the Green LED

sudo i2cset -y 1 0x6b 0x0A 0x00 for OFF the Green LED

sudo i2cset -y 1 0x6b 0x0b 0x01 for ON the Blue LED

sudo i2cset -y 1 0x6b 0x0b 0x00 for OFF the Blue LED

0x09	User LED Orange	Byte	Common	R/W	**User LED Orange ON - Write:** 0x01 **User LED Orange OFF - Write:** 0x00
0x0A	User LED Green	Byte	Common	R/W	**User LED Green ON - Write:** 0x01 **User LED Green OFF - Write:** 0x00
0x0B	User LED Blue	Byte	Common	R/W	**User LED Blue ON - Write:** 0x01 **User LED Blue OFF - Write:** 0x00

Table 12 User LEDs Commands Specifications

UPS PIco HV3.0B/B+ System-Users LEDs Mapping

The **UPS Pico HV3.0 HAT** is equipped with User LEDs as described above that can be used for any user information. However, in the models **UPS Pico HV3.0B/B+ HAT** there are ready soldering PADs that allow to solder external (via cables) LEDs and use them as an external indicator. Sometimes it is usefully to map to this User LEDs system functionalities LEDs and have some or most of them available for viewing if system based on **UPS Pico HV3.0 HAT and Raspberry Pi® is** placed in cases. The LEDs Mapping allows to "map" (copy) selected system LEDs indications to any User LED. The mapping does not change the "normal" system LEDs indication as it is defined.

0x17	Mapping_User_LED_Orange	Byte	Common	R/W	There is single byte register **0b76543210** where the corresponding bits when written with "1" are: - Bit 0th is mapping UPS LED - Bit 1st is mapping BAT LED - Bit 2nd is mapping CHG LED - Bit 3rd is mapping INF/HOT LED - Bit 4th is mapping FAN LED - Bit 5th is mapping REL LED - Bit 6th is mapping RPIV LED - Bit 7th is mapping EXT LED
0x18	Mapping_User_LED_Green	Byte	Common	R/W	There is single byte register **0b76543210** where the corresponding bits when written with "1" are: - Bit 0th is mapping UPS LED - Bit 1st is mapping BAT LED - Bit 2nd is mapping CHG LED - Bit 3rd is mapping INF/HOT LED - Bit 4th is mapping FAN LED - Bit 5th is mapping REL LED - Bit 6th is mapping RPIV LED - Bit 7th is mapping EXT LED

0x19	Mapping_User_LED_Blue	Byte	Common	R/W	There is single byte register **0b76543210** where the corresponding bits when written with "1" are: - Bit 0th is mapping UPS LED - Bit 1st is mapping BAT LED - Bit 2nd is mapping CHG LED - Bit 3rd is mapping INF/HOT LED - Bit 4th is mapping FAN LED - Bit 5th is mapping REL LED - Bit 6th is mapping RPIV LED - Bit 7th is mapping EXT LED

Mapped User LEDs can be used with their command for ON/OFF however when updated stratus if the "System" the value of it will be overwritten. It is possible also to map 2 or more system LEDs to a single User LED, however it makes reading more complicated.

Example of use

sudo i2cset -y 1 0x6b 0x17 0x01 (0b00000001) Mapping of System EXT LED on the User Orange LED

sudo i2cset -y 1 0x6b 0x18 0x00 (0b00100000) Mapping of System UPS LED on the User Green LED

sudo i2cset -y 1 0x6b 0x18 0x00 (0b00010000) Mapping of System BAT LED on the User Blue LED

UPS PIco HV3.0 HAT Buttons

The **UPS Pico HV3.0 HAT** is equipped with 6 buttons that can be used in various ways. Three of them are dedicated for user applications and can be handled by user through the **PICo** (I^2C) interface system, all other are specific for various **UPS Pico HV3.0 HAT** functionalities. All of them can be used for some start-up functionalities when **UPS Pico HV3.0 HAT** is reset. A detailed description of all buttons and their usage is provided on below table.

It is very important to start/stop the Daemons Service when doing Hardware Reset (UR) of the PIco HV3.0 HAT in order to avoid undefined situations with pulse train recognition procedure by the system. Resetting the PIco with Not Stopped the Daemon Service can cause an unexpected system safe shutdown (however without card corruption)

Button	Description	Usage	Additional Functionalities
RR	**R**aspberry Pi® Hardware **R**eset	Make Raspberry Pi Hardware Reset when pressed. To be used need installed (soldered) the	

		Gold Plated Reset Pin. **NOTE1**: Resetting of the Raspberry Pi®, can corrupt files on the SD card if used **NOTE2**: Resetting of the Raspberry Pi®, does **not** affect the **UPS PIco** (including PIco RTC)	NONE
UR	**U**PS Pico HV3.0 HAT Hardware **R**eset	Make the **UPS Pico HV3.0 HAT** Hardware Reset when pressed. **NOTE1**: Resetting of the **UPS PIco** does not reset the Raspberry Pi® **only** if cable powered. **NOTE2**: Resetting of the **UPS PIco** does **NOT reset** the Integrated Hardware RTC.	When pressed with combination with other buttons activate various start-up functionalities. The procedure is to press first the **UR** button, and then another one, then release the **UPSR** button and then release the other button (**A, B, C)**.
F	**F**ile Safe Shut Down (FSSD)	When pressed initiate the File Safe Shutdown Procedure. If Raspberry Pi® + **UPS Pico HV3.0 HAT** system is battery powered, after FSSD finished **UPS Pico HV3.0 HAT** will cut the power. Pressed again (need to have installed the Gold Plated Reset Pin for the restart option), start the Raspberry Pi® + **UPS Pico HV3.0 HAT** system again. In the battery powered System can be used as ON/OFF (files safe) button	When used with **UR** button, invokes the bootloader (light the Red User LED). The bootloader can be invoked also from the PICo interface.
A	**User Key A**	Can be used for User Application – Read the status via PICo (I^2C) or RS232 interface	NONE
B	**User Key B**	Can be used for User Application – Read the status via PICo (I^2C) or RS232 interface	When used with **UR** button, Set the **Hardware RTC** to default values
C	**User Key C**	Can be used for User Application – Read the status via PICo (I^2C) or RS232 interface	When used with **UR** button, sets system default values.

Table 13 UPS PIco HV3.0 HAT Buttons

User Buttons

User buttons **A**, **B** and **C** in the **UPS Pico HV3.0 HAT** are analog buttons, that means when pressed change the level voltage that is read by integrated dedicated A/D converter, read value of it is interpreted by the firmware and the result is loaded to a proper system variable and can be read by user. The **F** button is a digital interrupt driven button and his value can not be read by the user.

Each User Key and **FSSD** Key (button) has an additional THT pads that allows to be used by user for their own mounted buttons outside of the **UPS PIcoHV3.0 HAT** or **case**. Each of such user added buttons (keys) need just short to the system ground when pressed**.**

Reading of User Buttons values can be done by accessing the system variable by any software or by command line.

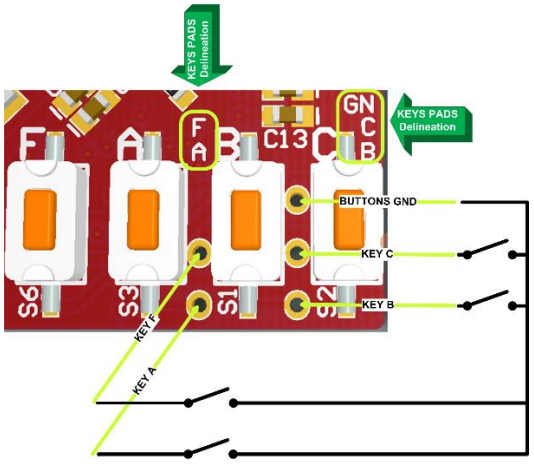


Figure 63 UPS PIco HV3.0A External Connectivity

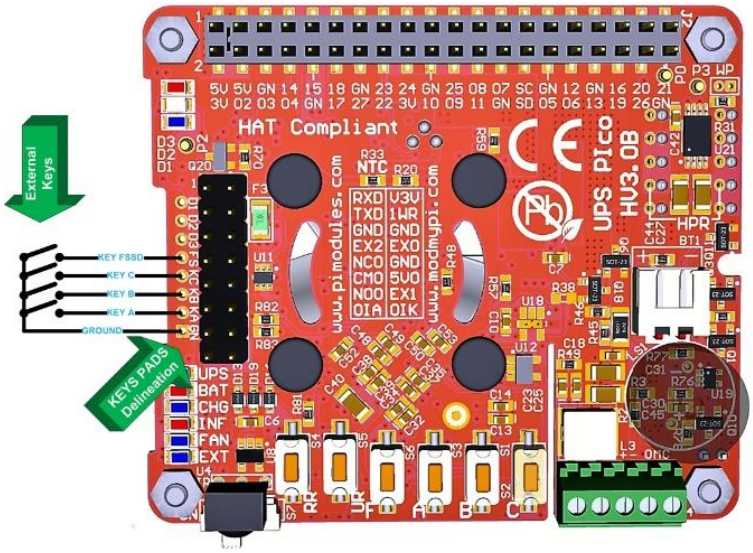


Figure 64 UPS PIco HV3.0B/B+ External Connectivity

To make working external keys (buttons), user need to solder cables to the THT key pads. It is not recommended to use a very long cable (due to analog implementation of the keyboard), their length should not be longer than 100 – 150 mm. However, we never tested longer cable and if user need to use longer cables should test them on their site. To make external keys workable, user need to short each one with **GN THT** pad when pressed. In example if user need to have external access to the FSSD **(F)** button, need to install button that short the **F** pad with the **GN** pad when pressed. Similar approach should be followed with other keys. Above picture show how external key (buttons) need to be connected. It is not needed to connect all of them.

The **key** register holds the latest value of pressed key, so user need to write i.e. **0x00** after reading, to recognize that new key has been pressed (when pressed again, even the same key). Current implementation requires timed (pooling) reading of this register to recognize that a key has been pressed.

Example of use

sudo i2cget -y 1 0x69 0x1A should return 1, 2 or 3

The pressed key value will remain in the register until new value will be written when key (new one or the same) will be pressed. Therefore, user should write zero to this register after reading to recognize and read again the new (or the same) key if (when) pressed.

Sudo i2cset -y 1 0x69 0x1A 0x00

UPS Pico HV3.0 HAT Sound Generation System

The **UPS Pico HV3.0 HAT** is equipped with Enhanced Sound Generation System. It is providing a user audio interface on various states of UPS **Pico HV3.0 HAT** conditions, but it is also available for dedicated user applications offering the whole range of acoustic frequencies full programmable by user.

There are 2 registers that are responsible for the generating sound **bfreq** and **bdur**. To generate sound user, need to program first the required frequency and then the required duration is 10th of ms.

Current implementation need to program one be one sound when generated. The maximum duration is 255 x 10 ms = 2.55 seconds

Additionally, it is possible to deactivate it permanently, by setting the bmode register to 0x00.

The default value is active

0x0D	bmode	Byte	Common	R/W	**Integrated Sounder Mode** **Read:** Anytime, Return actual **bmode** value **Write:** 0x00 – Unconditional Disable the Sounder **Write:** 0x01 – Unconditional Enable the Sounder **Default Value:** 0x01
0x0E	bfreq	Word	Common	R/W	**Frequency of sound in Hz**
0x10	bdur	Byte	Common	R/W	**Duration of sound in 10th of ms (10 = 100 ms)**

Example of use

sudo i2cset -y 1 0x6b 0x0d 0x00 Deactivate permanently the buzzer (no sounds will be played)

sudo i2cset -y 1 0x6b 0x0d 0x01 Activate permanently the buzzer (default value)

In order to play sound buzzer need to be activated firstly.

sudo i2cset -y 1 0x6b 0x0e 1047 w Set the frequency to C (1047 Hz) note

Sound will be not generated until **bdur** will be set, as the **bfreq** just set the frequency. Therefore the activation and duration of sound is done via register **bdur.**

sudo i2cset -y 1 0x6b 0x10 100 Set the duration to 1 second

After Sound execution, the **bdur** register is 0 again.

UPS PIco HV3.0 Bi Stable Relay

The **UPS PIco HV3.0 HAT** can be equipped with Bi Stable Relay with single coil. This Relay is standard offered with version UPS PIco HV3.0 HAT Plus/Advnced, it can be also ordered separately and added to the **UPS PIco HV3.0 HAT** Stack/TopEnd. In both cases this Relays is not mounted on the PCB and user need to do it by himself. The assembly (soldering) of the Bi Stable Relay on the **UPS PIco HV3.0 HAT** PCB it is very easy task and can be done by anybody using simple soldering iron. However it is also possible that this assembling can be ordered to be done by our company, if customer order directly on the eshop or any other eshop that offer such service.

The main benefit of the Bi Stable Relay (latching) is the power consumption. The Bi Stable Relays consume power only when switching from one state to another. All other times are not consuming power at all. So, we called it Zero Power Relay. Driving of such relays are little bit more complicated than the usual ones, however user not need to care about that as electrical drivers are assembled on and offered with each UPS PIco HV3.0 HAT PCB. Each Bi-Stable Relay does not have states **NO** (Normal Open) or **NC** (Normal Close), it has **Reset** and **Set** state instead. By switching of the Bi Stable Relay, user changes the state from **Reset** to **Set** and vice versa. User should know that If Bi Stable Relay change their status **Reset/Set** will stay on it (also when power will be completely removed) until new switch commend will be send. If Bi Stable Relay receives one command **Reset** or **Set**, sending multiple times of the same will not change anything. To change status must be send opposite one i.e. if s **Reset**, the opposite one is **Set**.

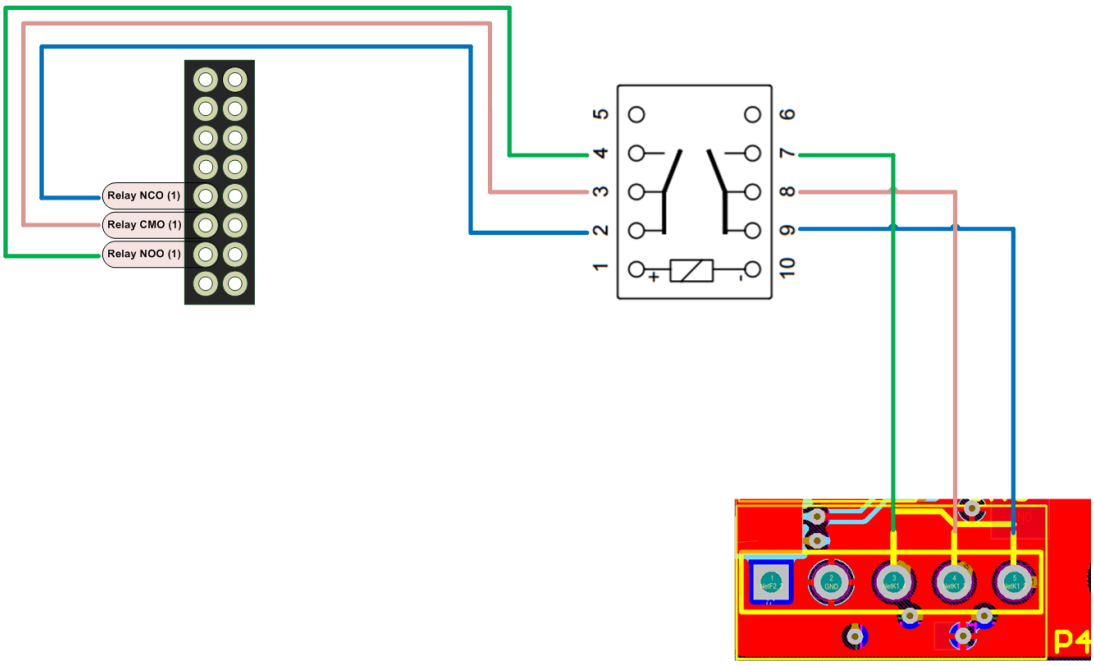
Bi Stable Relay Contact description	Meaning
Normally Close Output (on Set State)	On Set State this contact is closed and have connection with Common. Similar to the “normal” Relay NC (Normal Close)
Common Output (on Reset/Set State)	On Reset/Set States this contact is switching between NCO/NOO
Normally Open Output(on Set State)	On Set State this contact is Open and does not have connection with Common. Similar to the “normal” Relay NO (Normal Open)

Bi Stable Relay Basic Technical Specifications

Arrangement	2 form C (DPDT)
Contacts Material	Gold overlay silver alloy
Contacts Resistance (initial)	Maximum 50 mΩ (at 1 A 6 VDC)
Contacts Rating (resistive)	0.5 A 125 VAC or 1 A 30 VDC
Contacts Maximum Carrying Current	2 A
Contacts Maximum Switching Power	62.5 AV/30 W
Contacts Maximum Switching Voltage	250 VAC, 220 VDC
Contacts Operate (at nominal voltage)	Maximum 6 ms
Contacts Release (at nominal voltage)	Maximum 4 ms

Due to construction of **UPS PIco HV3.0 PCB** we do not allow to use Integrated Bi Stable Relay for switching of higher voltages/currents other than **32 VDC/1A** per switching contacts. The Integrated Bi Stable Relay contain a pair of independent switching contacts that can be used separated for 2 different and independent switching devices or connected in parallel if higher current (double) is needed.

Due to PCB construction It is absolutely not allowed to use the Integrated Bi Stable Relay for switching 220 V AC at any current, even very low.



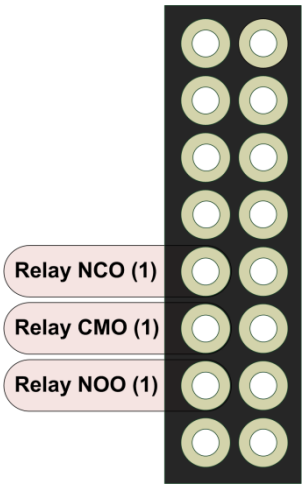
Example of use

sudo i2cset -y 1 0x6B 0x0c 0x00 should Reset the Bi Stable Relay

sudo i2cset -y 1 0x6B 0x0c 0x01 should Set the Bi Stable Relay

Each time when Bi Stable Relay is changing his state a characteristic "**tick**" is audible. Multiple execution of the same command is not changing anything.

0x0C	brelay	Byte	Common	R/W	**Zero Power Bi Stable Relay** **Write:** 0x01 Set **Write:** 0x00 Reset



UPS Pico HV3.0 HAT Programmable Auxiliary 5V@750 mA and 3.3V@150 mA Powering Sources

The **UPS PIco HV3.0 HAT** is equipped with Auxiliary **5V@750mA** and **3.3V@150mA** Power Sources with LDO that are independent from the 5V of the Raspberry Pi®. There are programmable and battery backed up (if programmed/activated by user), provide continuously supply even if Raspberry Pi® is switched OFF. The Auxiliary **5V@750mA** is over current protected with PPTC fuse on the **750mA** as also reverse current draw with Schottky diode. Therefore, due to small voltage drop the final voltage is about 4.85V, instead of the 5.0V. The **3.3V@150mA** is only protected with LDO itself embedded over current protection. These Auxiliary Power Sources are addressed to supply devices that need to be running even if Raspberry Pi® is switched OFF i.e. USB HUB, PIR Sensor, additional external high current relay, addon PCBs with extra hardware etc.

0x06	enable5V	Byte	Common	R/W	Defines usage of the Auxiliary 5V@750mA: 0x00 – Auxiliary 5V and 3.3V are not battery backed 0x01 – Auxiliary 5V and 3.3V are battery backed **Default Values is OFF** **Other codes are not allowed**

Example of use

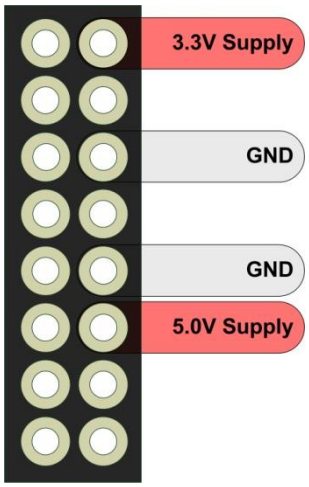
sudo i2cset -y 1 0x6B 0x06 0x00

the Auxiliary 5V and 3.3V will be not battery backed-up and stop working when power will be cut-off on the GPIO 5V, therefore will be present until Raspberry Pi® is powered by cable power or battery. When system, shutdown and **UPS PIco HV3.0 HAT** enter to Lower Power Mode these Auxiliary Powering Sources will cut out.

sudo i2cset -y 1 0x6B 0x06 0x01

the Auxiliary 5V and 3.3V will be battery backed-up and will continue supply also when 5V will be not available on the GPIO 5V, therefore will be present also after Raspberry Pi® is shutdown and not powered, and **UPS PIco HV3.0 HAT** enter to Lower Power Mode.

On that case the power usage on Low Powering Mode are increased by LDO and boost converter quiescent current (around of 3 mA in total) and current drawn by connected devices.



UPS Pico HV3.0 HAT IR Receiver Interface

The **UPS PIco HV3.0 HAT** is equipped with IR receiver interface. It is directly routed to the GPIO18. It can be used for any application like Media Player. If IR Receiver is not soldered, then the GPIO18 is free for any other application. An excellent tutorial how to use IR is provided by www.thepihut.com at below link:

https://thepihut.com/blogs/raspberry-pi-tutorials/raspberry-pis-remotes-ir-receivers

UPS PIco HV3.0 Serial Port(s)

The **UPS PIco HV3.0 HAT** is equipped with two independent serial ports. The first one is connected directly to the Raspberry Pi ® Serial Port (via GPIO 14 and 15), and the second one is available for dedicated user applications.

UPS PIco HV3.0 Serial Port A

This Serial Port is connected directly to Raspberry Pi® and if disable it is still electrically connected, just is set to HiZ. This port is used for System Monitoring via serial port or for new firmware upload. This Serial Port can be used only if it is set properly in the Raspberry Pi® - released from debugging using **sudo raspi-config**.

If this port is used by Raspberry Pi® or other HATs, it is needed to disable it on the **UPS PIco HV3.0**.

Setting Value	Meaning
0x00	**UPS PIco HV3.0** Serial Port is **OFF** **Default value**
0xff	**UPS PIco HV3.0** Serial Port is **ON** and data rate is set to **115200** pbs

If activated a set of messages will be send by this serial port to terminal program running in the Raspberry Pi®. This set of messages is continuously updating. If used bootloader firmware update, system is switching to bootloader mode and then it is no need to activate it.

Example of use

sudo i2cset -y 1 0x6b 0x02 0x00 Disable PIco RS232 and set tri-state the TXD and RXD pins

sudo i2cset -y 1 0x6b 0x02 0xff Enable the PIco RS232 and set the data rate to 115200 bps

UPS PIco HV3.0 Serial Port B

This Serial Port is connected directly to 16 Pin header of the **UPS PIco HV3.0 HAT** and its functionality is activate/deactivate the System based on serial message. This Serial Port is

This allows connecting the Raspberry Pi® serial port to the external RS232 12V interface (via Terminals Blocks PCB) or to 5V tolerant without any additional Jumpers or Cables. In addition the second Serial port of the **UPS PIco HV3.0** can be used as a second serial port routed directly to the I2C interface (this option is not unlocked yet, and will be available within one of the next firmware update). The **UPS PIco HV3.0** by default is set OFF and Raspberry Pi® Serial port can be used for any other applications. If it is needed it can be set

ON, as also the set the data rate. Setting the data rate sets it for both **UPS PIco HV3.0** serial ports.

After any firmware update the **UPS PIco HV3.0** Serial Ports(s) must be set again. It is done via PIco variable **rs232_rate**. The following settings are available:

UPS PIco HV3.0 FAN Control (Active Cooling System)

The **UPS PIco HV3.0 HAT** can be equipped with Active Cooling System based on micro FAN and dedicated temperature sensor. The PIco FAN is full PWM controlled rotation speed from 0% up to 100%. It can be manually set ON or OFF on per-selected speed, as also automatically based on preset temperature threshold. It can be done via the following registers placed at the I^2C address 0x6b (0x11, 0x12, 0x13).

0x11	fmode	Byte	Common	R/W	**Integrated Fan Running Mode** **Read:** Anytime, Return actual **fmode** value **Write:** 0x00 – Unconditional Disable the FAN with selected speed from the **fspeed** **Write:** 0x01 – Unconditional Enable the FAN FAN with selected speed from the **fspeed** **Write:** 0x02 – Automatic ON/OFF with defined speed in the fspeed, ON when temperature read in sensor T0-92 is higher than **fttemp** threshold, OFF when lower. Default value is set to 0x02 – Automatic ON/OFF **When written 0x02 to this register data are stored in the internal EEPROM. So, even if UPS PIco HV3.0 will be reset, automatic setup will be recovered. All other data (0x00, or 0x01) are not stored in the internal EEPROM.** **When UPS PIco is going down to the LPR mode, the FAN is automatically disabled, and enabled again when the UPS PIco returns to normal work**
0x12	fspeed	Byte	Common	R/W	**Integrated Fan Speed** **Read:** Anytime, Return actual **fspeed** value **Write:** 00 – Selected speed when OFF is 0% (not running) **Write:** 100 – Selected speed when ON is 100% (full speed running) **Any other (0-100) number is allowed and means % of speed and current consumption** Default speed is set to 50% **Any data written to this register are stored in the internal EEPROM. So, even if UPS PIco HV3.0 will be reset, will be recovered.**
0x13	fstat	Byte	Mirror	Read	**Read:** Anytime, Return actual if **FAN** is actually running or not (for remote users) When FAN is set to be running (even if not connected physically) the FAN LED is lighting. The intensity of the FAN LED is depending of the FAN Speed (PWM)
0x14	fttemp	Byte	Mirror	R/W	**Integrated Fan Temperature Threshold in Automatic Mode**

					BCD Fan Running threshold temperature in Celsius, 2 digits i.e. 35, means 35 Celsius. In order to be used (automatic FAN ON/OFF) need to set **fmode** to 0x02. Maximum temperature is 60 Celsius. Higher values will be ignored. FAN will start at 36 Celsius and stop at 35 Celsius. **Read:** Anytime, Return actual **fspeed** value **Write:** 00 – 60 Sets the TO-92 temperature Threshold for the Automatic FAN Start/STop **Default value is set to 35 Celsius**

Example of use – Manual FAN ON/OFF

sudo i2cset -y 1 0x6b 0x13 100 ***Set the FAN speed to 100***

sudo i2cset -y 1 0x6b 0x12 0x01 ***Set the FAN ON***

sudo i2cset -y 1 0x6b 0x12 0x00 ***Set the FAN OFF***

Example of use – Automatic FAN ON/OFF

sudo i2cset -y 1 0x6b 0x13 100 ***Set the FAN speed to 100***

sudo i2cset -y 1 0x6b 0x12 0x02 ***Set the FAN ON as an Automatic***

The default setup is Automatic Mode with 35 Celsius and 50% of FAN speed, so user do not need to change anything if like just to use the FAN. If higher cooling performance is needed (however with more noise) then the **fspeed** should be set to 100 (100%), similar with temperature threshold **fttemp**. However please kindly notice that FAN speed and temperature threshold have been set in order to have best performance with lowest noise.

UPS PIco HV3.0 HAT Measuring and Monitoring System

The **UPS PIco HV3.0** offer to the user an extended Measuring and Monitoring System that measure and report many system parameters trough installed sensors. Each sensor is reporting the **UPS PIco HV3.0 HAT** status via dedicated variables (PICo Registers). In addition there is access to the integrated 12 bits 3 x A/D converters. All monitoring system data are collected in a single entity called **PIco Status** and exists at the I^2C address **0x69**. Detailed specifications for each variable (register) as also examples are provided in next pages. The sensors are:

- Powering Mode
- System Error
- Battery Powered Available Running Time (calculated on battery capacity and system current consumption)
- Battery Level
- Raspberry Pi® GPIO 5V Level
- External Powering Level
- Incoming (from Raspberry Pi® GPIO 5V) current
- Outgoing (to Raspberry Pi® GPIO 5V) current
- Incoming (from External Powering) current
- A/D converter 0 Level
- A/D converter 1 Level
- A/D converter 2 Level
- Key pressed (described in the User Applications Hardware Interfaces)
- Embedded NTC temperature (measured on PIco PCB)
- TO-92 Sensor Temperature (measured near to the Raspberry Pi® PCB)
- Opto Coupler
- Integrated Charger Status
- Running PIco validation
- PCB versions

- Bootloader Versions
- Firmware Version

Green marked features has been not activated yet.

Powering Mode

Raspberry Pi® GPIO 5V Level

External Powering Level

UPS PIco HV3.0 12-bit A/D converters

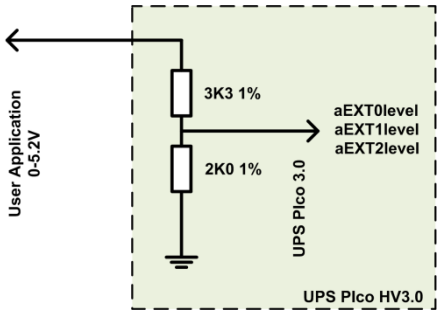
The **UPS PIco HV3.0** is equipped with 3 x 12 bits multichannel A/D converter. Access to their conversion data is possible via dedicated PICo Registers placed at the I^2C address 0x69 (0x14, 0x16, 0x18) separately for each channel. Those A/D converters read continuously data every 250 uS with conversion time of 3.5 uS per sample. However due to implemented low noise software enhanced filtering in the firmware the effective rate data rate is around of 0.002 sec per reading (each A/D register values is refreshed every 2000us or 500 Hz).

Each of the A/D converters is pre-scaled to measure voltage 0-5.2V with implemented on the UPS PIco HV3.0 HAT resistor divider. They are named aEXT0, aEXT1, aEXT2. However, there is also a possibility for the user (if use Terminal Block PCB or additional external resistor) to use two of them as pre scaled of 0-10V, 0-20V, or 0-30V. These two A/D converters are named aEXT1 and aEXT2.

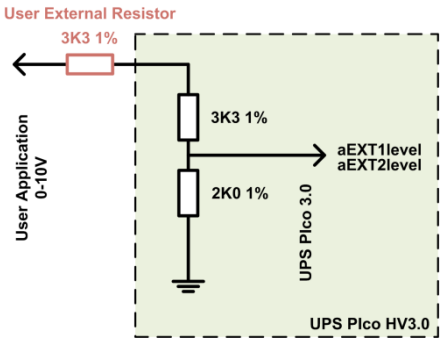
Due to electrical requirements of the integrated A/D converters the impedance is set to low values, therefore some high impedance sensors cannot be read properly as could require higher impedance of A/D converter interface. On such cases it is recommended to use Voltage Follower that converts the sensor high impedance to UPS PIco HV3.0 HAT A/D converters lower one.

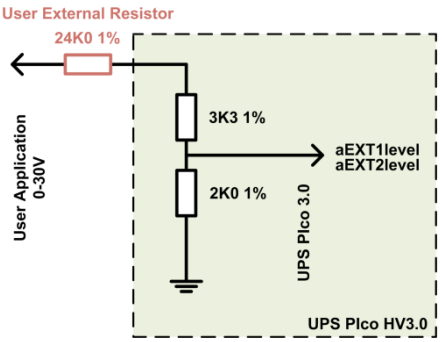
This functionality (of the Voltage Follower) has been implemented on the Terminal Blocks PCB, where on one of the A/D's converters (the aEXT0) a Voltage Follower has been assigned and allows to convert high impedance of any possible used sensor to low impedance on the A/D side. A detailed description of the Terminal Blocks PCB and their functionalities are described in separate section of this manual.

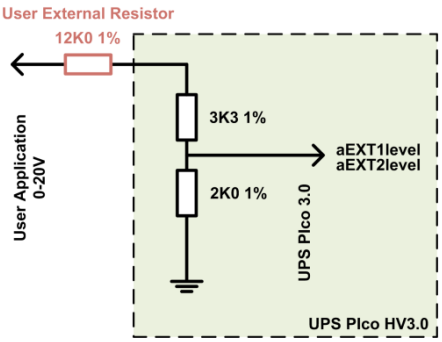
The basic circuit of all A/D converters (the resistor dividers) are shown here below, it is same to all implemented A/D converters in the UPS PIco HV3.0 HAT.



If user decide to use Higher Voltage Interface as pre-scaled of 0-10V, 0-20V, or 0-30V the **Terminal Blocks PCB** should be used, or an **additional resistor** need to be added externally, like in the below pictures. In addition, the register **setA_D (0x08)** at address **0x6b** should be set to a proper value according the below table to keep the proper voltage conversion. If user not like to use embedded voltage converter, then it is needed to disable it via a proper command and read the raw data directly from the related register. All A/D readings have internal reference of 2.048V and are filtered by the firmware with “Olympic Score” and “Low Pass” Filtering.







Caution: The **UPS PIco HV3.0** has implemented an ESD protection on each A/D converter input. This protection protects the system from ESD discharges and **does not** from continuously high voltage applied.

Therefore, it is very important if High Voltage used (10V, 20V or 30V), to be sure that a proper resistor(s) has been used. If smaller resistor(s) that required will be used, then the A/D input will be destroyed permanently, and very possible also the whole UPS PIco HV3.0 HAT PCB

Therefore, user need to take care to be sure that a proper values of resistor(s) has been used.

setA_D Values	AEXT0level Scale	AEXT1level Scale	AEXT2level Scale	AEXT1 Resistor	AEXT2 Resistor
0x00	5.2V	5.2V	5.2V	0K	0K
0x01	5.2V	5.2V	10V	0K	3K3
0x02	5.2V	5.2V	20V	0K	12K0
0x03	5.2V	5.2V	30V	0K	24K0

setA_D Values	AEXT0level Scale	AEXT1level Scale	AEXT2level Scale	AEXT1 Resistor	AEXT2 Resistor
0x00	5.2V	5.2V	5.2V	0K	0K
0x10	5.2V	10V	5.2V	3K3	0K
0x20	5.2V	20V	5.2V	12K0	0K
0x30	5.2V	30V	5.2V	24K0	0K

** Red marked table settings are not unlocked yet in current firmware version*

Any combination of data provided on above table is allowed. The register **setA_D** is 8 bit. The 4th MSB bits are responsible for the **AEXT1level** pre-scale, and the 4th LSB bits are responsible for the **AEXT2level** pre-scale.

On the **UPS PIco HV3.0** PCB the AEXT**0level** is marked as **A50,** the **AEXT1level** is marked as **A15** and the **AEXT2level** is marked as **A30.**

If user need read raw data, then there is a need to write **0xFF** to the **setA_D** register. With raw data option the basic standard resistor divider is used, and the input voltage cannot exceed the 5.2V. The maximum reading is 4095 (12 bit A/D). All A/D readings have internal reference of 2.048V and are filtered by the firmware with "Olympic Score" and "Low Pass" Filtering.

Example of use

sudo i2cget -y 1 0x69 0x14 w ***should return value of the aEXT0level***

sudo i2cget -y 1 0x69 0x16 w ***should return value of the aEXT1level***

sudo i2cget -y 1 0x69 0x18 w ***should return value of the aEXT2level***

sudo i2cset -y 1 0x6b 0x08 0x00 ***sets all A/D readings to pre sacled 0-5.2V (default)***

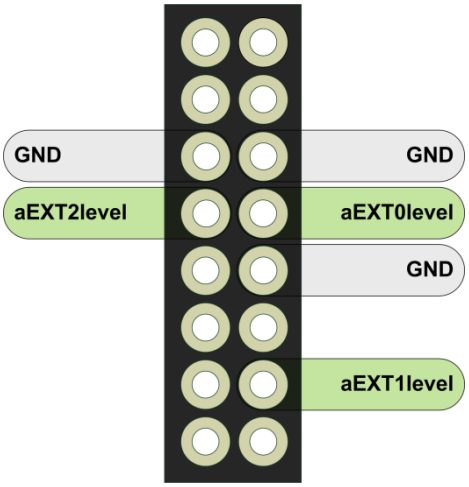
sudo i2cset -y 1 0x6b 0x08 0xff ***sets all A/D readings to raw data (0x0000-0x0fff)***

Registers Located at 0x69 I2C address related to A/D readings

0x14	aEXT0level	Word	Mirror	Read	Means value of the first A/D converter pre scaled to 5.2V. Higher voltage could not be supplied. Readings are in 10th of mV in BCD format
0x16	aEXT1level	Word	Mirror	Read	Means value of the second A/D converter pre scaled to 5.2V. Higher voltage could be supplied with an external resistor divider. Readings are in 10th of mV in BCD format. **If added an extra resistor can be used as pre scaled to 10, 20 or 30V.**
0x18	aEXT2level	Word	Mirror	Read	Means value of the second A/D converter pre scaled to 5.2V. Higher voltage could be supplied with an external resistor divider. Readings are in 10th of mV in BCD format. **If added an extra resistor can be used as pre scaled to 10, 20 or 30V.**

Registers Located at 0x6B I2C address related to A/D settings

0x08	setA_D	Byte	Common	R/W	Defines the pre scaler of the **AEXT1level** and the **AEXT2level** registers. The 4th MSB bits are responsible for the **AEXT1level** pre-scale, and the 4th LSB bits are responsible for the **AEXT2level** pre-scale. **Read:** Anytime, Return actual **setA_D** value **Write:** 0x00 – 5.2V prescale for the **AEXT2level** **Write:** 0x01 – 10V prescale for the **AEXT2level** **Write:** 0x02 – 20V prescale for the **AEXT2level** **Write:** 0x03 – 30V prescale for the **AEXT2level** **Write:** 0x00 – 5.2V prescale for the **AEXT1level** **Write:** 0x10 – 10V prescale for the **AEXT1level** **Write:** 0x20 – 20V prescale for the **AEXT1level** **Write:** 0x30 – 30V prescale for the **AEXT1level** **Write:** 0xFF – all A/D registers will contain raw data **RED Marked** – not implemented yet



Improving Accuracy of 12-bit A/D converters -

The **UPS PIco HV3.0 HAT** A/D converters system (including measure Raspberry Pi® 5V, EPR measured voltage, Battery Level Voltage, thermal measures, etc.) is based on micro controller internal voltage reference as also on resistors dividers. These resistors used for internal voltage dividers have accuracy of 1%, and the internal voltage reference around of 2%. Therefore the total system accuracy for measures is about 3%. Due to that fact, measured A/D levels, as also voltages on battery can vary of 3%. Therefore measured Battery Level can show 4.3V instead of real 4.2V (4.4V-4.2=0.2V, 0.2/4.2=4.7%) or 4.0V instead of 4.2V. This reported values are very unique as usually the deviation of resistors accuracy is absorbed by the deviation of the voltage reference accuracy. This has been measured on our company during the testing process on many thousands of Pico and statistically the total measures deviation is about +/-0.2V for battery level measure. This

accuracy is enough for all standard applications, however for some is not enough. In order to improve the measuring accuracy to better than 1% has been implemented a dedicated Accuracy Improving System.

The LiPO battery charger is based on independent charger IC that has his own very accurate voltage reference for charging with accuracy better than 0.75%. This feature has been used by the Pico Accuracy Improving System. This system is based on very accurate voltage level when LiPO battery is charged. When LiPO battery is fully charged then their level is exact 4.2V. Basing on this information Pico Accuracy Improving System generating a correction factor that is stored in the **UPS PIco HV3.0 EEROM**, and used for further measures. Using this feature user can significant improve system measuring accuracy. In order to do that accuracy adjustment, user need to:

- Start up their system based on **UPS PIco HV3.0** and **Raspberry Pi ®**
- Make sure that system is cable powered (micro USB or EPR either) and LiPO battery is connected
- Wait until their LiPO battery has been fully charged (the CHG LED will be off)
- Usually to have charged battery (and CHG LED OFF) user need to wait some hours (for small one of 450 mAh)
- User need to be sure that LiPO battery is connected and fully charged. It is mandatory requirement, as everything is based on the fully charged level of LiPO battery measure.
- Make a simple read of battery level by using command

sudo i2cget -y 1 0x69 0x08 w

- User will read with above command the battery level value near to the 4.2V
- Then user need to execute command doing calculating the accuracy deviation factor

sudo i2cset -y 1 0x6b 0x00 0x11

- After execution of above command user can check again the battery voltage level

sudo i2cget -y 1 0x69 0x08 w

- This factor will be saved in the **UPS PIco HV3.0 EEPROM**
- It is recommended to execute this command once, however multiple execution of it can provide little bit better (really very little bit) correction
- It is very important to be sure that the LiPO battery is fully charged and the CHG LED is OFF, after charging.

- Using of this command improperly will decrease accuracy of the measuring system
- With factory defaults recall this factor will be returned to zero, therefore any mistake can be easy corrected
- There is no access to value of the correction factor via I^2C to be read

Embedded NTC temperature

UPS PIco HV3.0 HAT is equipped with NTC sensor placed on top of PCB. This sensor can be used as information of the system environment temperature. User can read their value by using the following command:

sudo i2cget -y 1 0x69 0x1b

The result will be temperature in Celsius Degrees in BCD format. It is important to notice that this an temperature indicator, placed on the **UPS PIco HV3.0 HAT** pcb, and not an accurate thermometer. Therefore it provides just an indication of temperature and not exact mesure.

TO-92 Sensor Temperature

UPS PIco HV3.0 HAT can be equipped with T0-92 temperature sensors (MCP9701A-E/TO) with accuracy of ±2°C. User can read their value by using the following command:

***sudo i2cget -y 1 0x69* 0x1c**

The result will be temperature in Celsius Degrees in BCD format. It is important to notice that this an temperature indicator, placed on the **UPS PIco HV3.0 HAT** pcb, and not an accurate thermometer. Therefore it provides just an indication of temperature and not exact mesure.

Integrated Charger Status

- PCB versions
- Bootloader Versions
- Firmware Version

UPS PIco HV3.0 System Time Schedulers

The UPS PIco HV3.0 has implemented 2 independent, Time Schedulers. There are:

- The **B**asic Time **S**cheduler (**BS**)
- and, The **E**vent **T**riggered **R**TC Based **S**ystem **A**ctions **S**cheduler (**ETR SAS**)

Both schedulers cannot be used at the same time, and if the first one is selected, the second is deselected and vice versa. The default value is selecting the **B**asic **S**cheduler. Running of Time Schedulers increasing the current consumption during the sleep (LPR) mode. The **BS** increases it about 50 uA, and the **ETR SAS** depending of the monitored Evens selection.

On models of Raspberry Pi equal or newer of B+ (including model 4), schedulers are also switching ON/OFF (also on micro USB powering by using the PE pin) power supply of the Raspberry Pi, therefore Schedulers can be used with powering cables connected to the Raspberry Pi on above models. Power consumption from the cable Power Supply on that case is about 10mA on USB powering and 0mA on EPR powering. The 5V is provided to the Raspberry Pi is also to all other HATs via 5V GPIO, however the Raspberry Pi is OFF. This is achieved with the PEN (Power Enable) pin located on the Raspberry Pi PCB buck power converter. In order to use that feature it is needed to have soldered the PE Gold Plated Pin in addition to the Reset Pin.

The Register responsible for the Scheduler selection is located in the **0x6b** Registers Set

0x6B -> UPS PIco **0x20** Time Scheduler Selector

This register is used to select the System Time Scheduler. Three values are possible 0x00 (default value) – which means deselected all schedulers, 0x01 – which means **Basic Scheduler** selected, or 0x02 – which means **Event Triggered RTC Based System Actions Scheduler**. Setting of it is necessary to select the proper System Time Scheduler.

sudo i2cset -y 1 0x6b 0x20 0x01 to select **BS** (default value)

or

sudo i2cset -y 1 0x6b 0x20 0x02 to select **ETR SAS**

Basic Scheduler

This scheduler is basically used when **UPS PIco** simple scheduler is needed, just to make **ON/OFF** the Raspberry Pi (so no needed to use the complex settings of the ETR SAS). There are only few registered involved in this scheduler and setting up of them is rapid. User need just to set how long Raspberry Pi® should be running, after that, how long Raspberry Pi® should be not running, and how many times it should happen. The time resolution of the **BS** is based on **1 minute**, however everything is adjusted with 1 second accuracy, as each start/stop action is executed at the beginning (first second) of internal RTC counted minute (even if the internal RTC is not set, it is always running – however some features need to have it set). Below picture explain the logic behind of this **B**asic **S**cheduler. In order rapid to use BS it is not needed to have setup the RTC, however some features need to have it set-up.

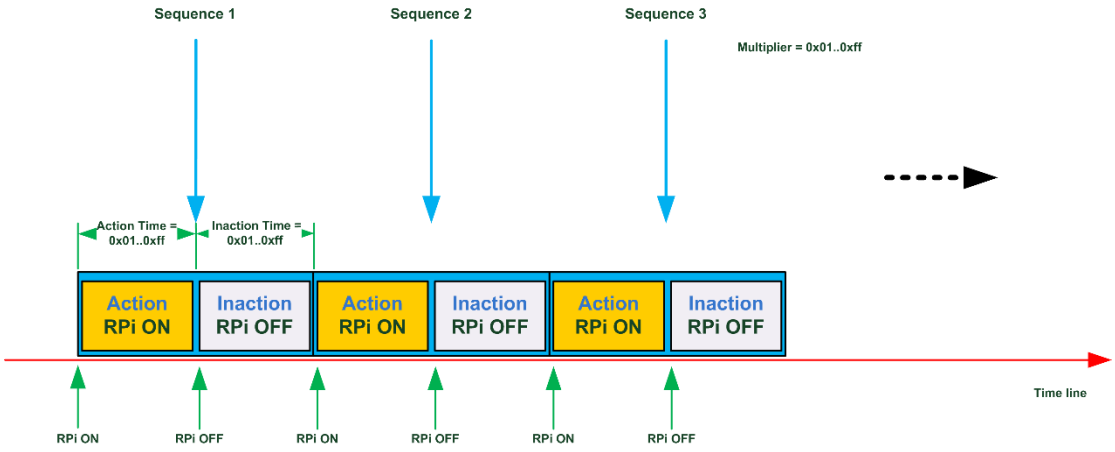


Figure 65 Single Basic Scheduler

BS Definitions

There are some definitions that need to be specified to have better understanding of the **Basic Scheduler** functionality. There are basically similar to the **ETR SAS** definitions (described in the next chapter), however are simplified due to adaptation to this simple **Basic Scheduler**. There are:

BS Action – It is **ON** the Raspberry Pi® only

BS Inaction – It is **OFF** the Raspberry Pi® only

BS Action Time – Specify time between **BS Action** (ON of the Raspberry Pi®) and their opposite state (OFF of the Raspberry Pi®). In example if Raspberry Pi® **Power ON**, the opposite state is Raspberry Pi® **Power OFF**. Therefore, **BS Action Time** is time between ON and OFF of the Raspberry Pi®. So, simply saying "how long the Raspberry Pi will be ON"

BS Inaction Time – Specify time between **BS Inaction** (OFF of the Raspberry Pi®) and their opposite state (ON of the Raspberry Pi®). In example if Raspberry Pi® **Power OFF**, the

opposite state is Raspberry Pi® **Power ON**. Therefore, **BS Inaction Time** is time between OFF and ON of the Raspberry Pi®. So, simply saying "how long the Raspberry Pi will be OFF"

BS Sequence – The **BS Sequence** defines both states (**ON** and **OFF**) together as single entity. This definition is usefully for further explanation of **BS Multiplier**. User can have such sequence multiplied or run infinitely.

BS Start Time – The **BS Start Time** defines starting time in 2 BCD bytes hours and minutes. However by default this value is set to 0xFFFF, and means start in next minute (immediately) after activation with **BS_RUN** register.

BS Multiplier – Defines how many times **BS Sequences** will be repeated from the **BS Sequence**, up to 254 times or infinitely which is 255 (0xFF).

Basic Scheduler Involved PICo Registers

The following PICo Registers are involved in the **Basic Scheduler** programming. There are:

- **Time_scheduler_Selector** - placed address **0x6B** and location **0x20** (default 0x01)
- **BS_action_time** - placed address **0x6B** and location **0x21** (default 0x01 minute)
- **BS_inaction_time** - placed address **0x6B** and location **0x22** (default 0x01 minute)
- **BS Start Time** - placed address **0x6B** and location **0x24** (default 0xFFFF)
- **BS_multipier** - placed address **0x6B** and location **0x26** (default 0xFF infinitive)
- **BS_RUN** - placed address **0x6B** and location **0x27** (default 0x00 OFF)

Detailed description of all Registries related to the **Basic Scheduler** are located at the **0x6B -> UPS PIco Module Commands** and there are:

0x6B -> UPS PIco **0x21** BS_action_time (in minutes)

This register defines how the Raspberry Pi® long will be ON (called Action) and running after starting up of Basic Scheduler. The default value is 0x01. Each value is in minutes. The maximum time is 0xff (255 minutes). This register can be read when Raspberry Pi® is running, user will see the decreased value as time is passed. The value cannot be lower than 0x01.

0x6B -> UPS PIco **0x22** BS_inaction_time (in minutes)

This register defines how long the Raspberry Pi will be OFF (called inaction) and not running after starting up of Basic Scheduler. The default value is 0x01. Each value is in minutes. The maximum time is 0xff (255 minutes). This register can be read when Raspberry Pi® is running, user will see the decreased value as time is passed. The value cannot be lower than 0x01.

0x6B -> UPS PIco **0x23** BS_Start Time (4 digits in BCD)

0x6B -> UPS PIco **0x24** BS_multiplier

This register defines how many times the **Basic Scheduler Sequence** (set of one Action and one Inaction - Raspberry Pi® ON/OFF) will be running. It can make it running counted times (from 1 up to 254), or if programmed 0xff (255) then the Action will be executed unlimited times (repeated continuously). This register can be read when Raspberry Pi® is running, user will see the decreased value as counter is passed.

0x6B -> UPS PIco **0x25** BS_RUN

This register is used to make **Basic Scheduler** running (start) or not (stop). However, there are some smart variations that allows the Basic Scheduler to be used more flexible and easier. Therefore, with this register setting **BS** can start immediately, on date changing (00:00) – here required is to have set-up the RTC properly the time, as also start with Active state and then go to Inactive or the opposite, Start with Inactive state and then go to Active one.

0x20	Time_Scheduler_Selector	Byte	Common	R/W	Selects which **Scheduler** is used: - 0x01 Basic Scheduler (default) - 0x02 ETR SAS Only one can be selected, and each programming is referred to it. **Default value : 0x01**
0x21	BS_action_time	Byte	Common	R/W	**Basic Scheduler Action Time** in minutes. Allowed values are 0x01 – 0xff. **Default value : 0x01**
0x22	BS_inaction_time	Byte	Common	R/W	**Basic Scheduler Inaction Time** in minutes. Allowed values are 0x01 – 0xff. **Default value : 0x01**
0x22	BS_start_time	word	Common	R/W	**Basic Scheduler Start Time** in BCD. **Default value : 0xffff**
0x25	BS_multipier	Byte	Common	R/W	**Basic Scheduler Sequence Multiplier**. Allowed values are 0x01 – 0xff. **Default value : 0xff** Value of 0xff means running (repeating) unlimited times after starting.
0x27	BS_ RUN	Byte	Common	R/W	Specify when **Basic Scheduler** is starting and running or not. Allowed values are: - 0xff BS Starts in first defined start time with resolution to minute, with executing first the action state - 0x00 BS Stops immediately in next minute, after finishing last state **Default is 0x00 (not running). When changing**

					the BS to 0xff (running), system automatically update the battery running time to unlimited (0xff) and if

Table 14 Basic Scheduler involved Registers

Basic Scheduler Optical Indications

When **B**asic **S**cheduler is activated (BS_ RUN=0xFF, 0xFA, 0xFB), the **INF LED** is blinking for 50 ms every 1 second. This happens as far the Raspberry Pi is cable powered and running. In the Low Powering Mode, the **INF LED** as all others are OFF. This feature allow user to know when **BS** is active or not.

All programmed **BS** values are stored in the internal **EEPROM**, therefore if your system reset/restart the **BS** will continue running until **BS_ RUN=0x00.**

Activating of Basic Scheduler UPS PIco automatically changes the battery_run_time to unlimited (0xFF) to avoid collisions with programmed schedulers.

Deactivating of Basic Scheduler UPS PIco automatically changes the battery_run_time to 70 second (0x01)

Programming (setting-up) of Basic Scheduler is not allowed when system is Battery Powered.

The Basic Scheduler can be used when system is supplied only with battery

If during the Basic Scheduler execution powering condition changed (i.e. enter the powering Cable) system will behavior as without Basic Scheduler therefore will start-up.

All other functionalities related to Raspberry Pi running functionalities (i.e. STA or Low Battery) are still active when the Basic Scheduler is running

BS Example 1st - Simple Raspberry Pi® ON/OFF executed infinitive times for 1 minutes (ON/OFF every minute), starting immediately

We need to start up - set ON - the Raspberry Pi®, keep it running for 1 minutes, shutdown it, and after 1-minute start it again. This will be repeated infinitely.

Time_Scheduler_Selector = 0x01;	Select the Basic Scheduler
BS_action_time = 0x01;	Sets duration of Action (ON) time to 1 minute (Raspberry Pi® will run for 1 minute and then shutdown)
BS_inaction_time =0x01;	Sets duration of Inaction (OFF) time to 1 minute (Raspberry Pi® will be sleeping for 1 minute)
BS_multipier = 0xff;	This will be repeated forever
BS_ RUN =0xff;	When user decide, just activate the Basic Scheduler, and it is start immediately

The data entering should looks like below (it important to follow the below order to avoid any mistake in programming):

1. Make sure to select the **BS** as a current scheduler

   sudo i2cset -y 1 0x6b 0x20 0x01 for making BS as selected scheduler

2. Enter **BS Action Time**, on our case it is 1 minute

   sudo i2cset -y 1 0x6b 0x21 0x01 for duration time 1 minute

3. Enter **BS Inaction Time**, on our case it is 1 minutes

   sudo i2cset -y 1 0x6b 0x22 0x01 for repetition time 2 minutes

4. Enter **BS Multiplier**, on our case it is infinitive

   sudo i2cset -y 1 0x6b 0x23 0xff for infinitive running

5. Check if programmed values are OK, by running the below python script

   sudo python status_bs.py

If everything is as expected, run the Basic Scheduler to start immediately

sudo i2cset -y 1 0x6b 0x24 0xff

BS Example 2nd- Simple Raspberry Pi® ON/OFF executed 100 times for 1 minutes (ON/OFF every minute), started beginning next day (00:00)

We need to start up - set ON - the Raspberry Pi®, keep it running for 1 minutes, shutdown it, and after 1-minute start it again. This will be repeated 100 times.

Time_Scheduler_Selector = 0x01;	Select the Basic Scheduler
BS_action_time = 0x01;	Sets duration of Action (ON) time to 1 minute (Raspberry Pi® will run for 1 minute and then shutdown)
BS_inaction_time =0x01;	Sets duration of Inaction (OFF) time to 1 minute (Raspberry Pi® will sleeping for 1 minute)
BS_multipier = 0x64;	This will be repeated 100 times
BS_ RUN =0xfb;	When user decide, just activate the Basic Scheduler, and it is start beginning next day

The data entering should looks like below (it important to follow the below order to avoid any mistake in programming):

The data entering should looks like below (it important to follow the below order to avoid any mistake in programming):

1. Make sure to select the **BS** as a current scheduler

   sudo i2cset -y 1 0x6b 0x20 0x01 for making BS as selected scheduler

2. Enter **BS Action Time**, on our case it is 1 minute

   sudo i2cset -y 1 0x6b 0x21 0x01 for duration time 1 minute

3. Enter **BS Inaction Time**, on our case it is 1 minutes

   sudo i2cset -y 1 0x6b 0x22 0x01 for repetition time 2 minutes

4. Enter **BS Multiplier**, on our case it is infinitive

   sudo i2cset -y 1 0x6b 0x23 0xff for infinitive running

5. Check if programmed values are OK, by running the below python script

   sudo python status_bs.py

If everything is as expected, run the Basic Scheduler to start beginning next day

sudo i2cset -y 1 0x6b 0x24 0xfb

Events Triggered RTC Based System Actions Scheduler

Not activated yet in current firmware version

The **E**vents **T**riggered **R**TC Based **S**ystem **A**ctions **S**cheduler (**ETR SAS**) is a very advanced functionality that allows user to implement a simple timed Actions (usually ON/OFF) of the Raspberry Pi®, but also a very complicated Actions Schedules depended to External Events and Time without or with involvement of Raspberry Pi®. This functionality can be perfectly combined with **IoT** or any other time dependent applications. The time resolution of the **ETR SAS** is based on **1 minute**, however everything is adjusted with 1 second accuracy, as each action start/stop is executed at the beginning (first second) of internal RTC counted minute. There are implemented 4 parallel working **ETR SAS** running with different Set-up's. That means that i.e. user can set the 1st **ETR SAS** running at night (00:00 – 06:00) every 10 minutes (repeated 20 times), the 2nd **ETR SAS** to run at morning time (06:00 – 10:00) every 30 minutes, and the rest of the day every 1 minute based on 3rd **ETR SAS**. The **ETR SAS** can be based on the **RTC**, but can be also time independent and execute Action triggered by external Event (i.e. A/D). The **Action** can be simple ON/OFF the Raspberry Pi® but also independent of the Raspberry PI® (without switching it ON) just activate the Auxiliary 5V@750mA or Bi-Stable Relay switching. Combination of all **ETR SAS** produce in the result a very complicated state machine able to implement practically any schedule is needed by user.

ETR SAS Definitions

There are some definitions that need to be specified to have better understanding of the **ETR SAS** functionality. There are:

Scheduler – A State Machine that maintains the schedules (states)

Schedules – Set of **Sequences** that contains number of **Actions**, which execution of them is based on **Events** or **RTC**, and are executing by **Scheduler**. In example **Action** is ON/OFF Raspberry Pi® and **Event** is A/D data that fires the **Action (**ON/OFF Raspberry Pi®). **Action** is ON/OFF Raspberry Pi® on requested date and time. **Sequences** contains these **Actions** that can be repeated within one Schedule (i.e. every day) or can happens only once. The definition of Sequence is required to specify the Repetition Time of Sequences.

Sequences – contain one or more **Actions**.

RTC – Real Time Clock (the hardware clock/calendar used/embedded by the **UPS PIco HV3.0A/B/B+ HAT**), synchronized with Raspberry Pi® RTC. It is mandatory to have the PIco RTC synchronized with Raspberry Pi® RTC. It can be easy check with

sudo i2cdetect -y 1

the address **0x68** must be **UU**.

Event – Occurrence that is used for triggering the **Scheduler**. It can be A/D level, IR, 1-wire, RS232, etc.

Action – Result of the **Scheduler** activity. It can be ON/OFF the Raspberry PI®, ON/OFF the Auxiliary 5V@750mA, Set/Reset of the Bi-Stable Relay, 1-wire, RS232 data, etc. The Action does not need to have running the Raspberry Pi®, as it is "above of it" however can contain the Raspberry Pi ® ON/OFF procedure.

Action Duration Time – Specify time between **Action** and their opposite state. In example if **Action** is Raspberry Pi® **Power ON**, the opposite state is Raspberry Pi® **Power OFF**. Therefore, **Duration** is time between ON and OFF the Raspberry Pi®. In case of use i.e. bi-Stable Relay the **Duration** is time between **Set** and **Reset**. In case of use i.e. programmable **5V@750mA** the **Duration** is time between 5V@750mA **ON** and **OFF**. There is no need to have Raspberry Pi® running at the same time, however it could be. These two **Actions** are independent.

Action Repetition Time – Each **Action** can be repeated within the same **Schedule**. The **Action Repetition** defines the **time** between beginnings of first and the repeated Action. In example Action will be defined to be executed at 10:00. The **Action Repetition Time** will define Repetition of the same action after XX min (Repetition time) i.e. 30 minutes. Therefore, if **Action** is switch Raspberry Pi® at 10:00 for **Action Duration** of 10 minutes, the **Action Repetition Time** will be, to switch it again after 30 minutes. Thus, could be repeated **Action Multiplier** times if needed. The Action Repetition Time could be 0.

Action Multiplier – Defines how many times **Action** will be repeated within the same Sequence.

Sequence Repetition Time – One **Schedule** contains multiple **Sequences**. That could be or not repeated. Each **Sequence** contains one or more **Actions**. The Sequence Repetition Time defines the repletion of the Sequence. In example every day, every week, etc.

The below picture shows 2 Schedules that are executed by Scheduler. One of the Schedules contain Sequence with multiple Actions (3) repeated every day. The second one contains Schedule with single action (Relay ON/OFF) that is repeated every day (in different time). Both are running independently. This will help you for better understanding of each definition.

ETR SAS Definitions Dependencies

The **UPS PIco HV3.0** execute (or not if not activated) the **ETR SAS Scheduler**. The **Scheduler** contains up to 4 **Schedules** that are executed separately when the **Scheduler** is running. User can activate from 1 to 4 **Schedules**. Each Schedule have their own **Sequence**. The **Sequence** can be repeated or not within their **Schedule**. Each **Sequence** hold a number (at least 1) of **Actions**. Each **Action** can be repeated within the same **Sequence** counted times. Actions can be triggered by **RTC** or **Events.**

Raspberry Pi® ETR SAS Self Programming

It is possible that invoked by ETR SAS (switched ON) Raspberry Pi® after achieving of their external events goals will re-program parameters (Schedules) of their ETR SAS with different parameters or stop execution of it. Thus, make practically unlimited options of running of ETR SAS.

Template for ETR SAS preparation

To simplify preparation of user born ETR SAS, there is provided a template where user can easy enter (draw) their own schedule, and the simple program PICo SAS registers based on that. To use it, it is needed to print the page with it (it is also provided in a separate PDF).

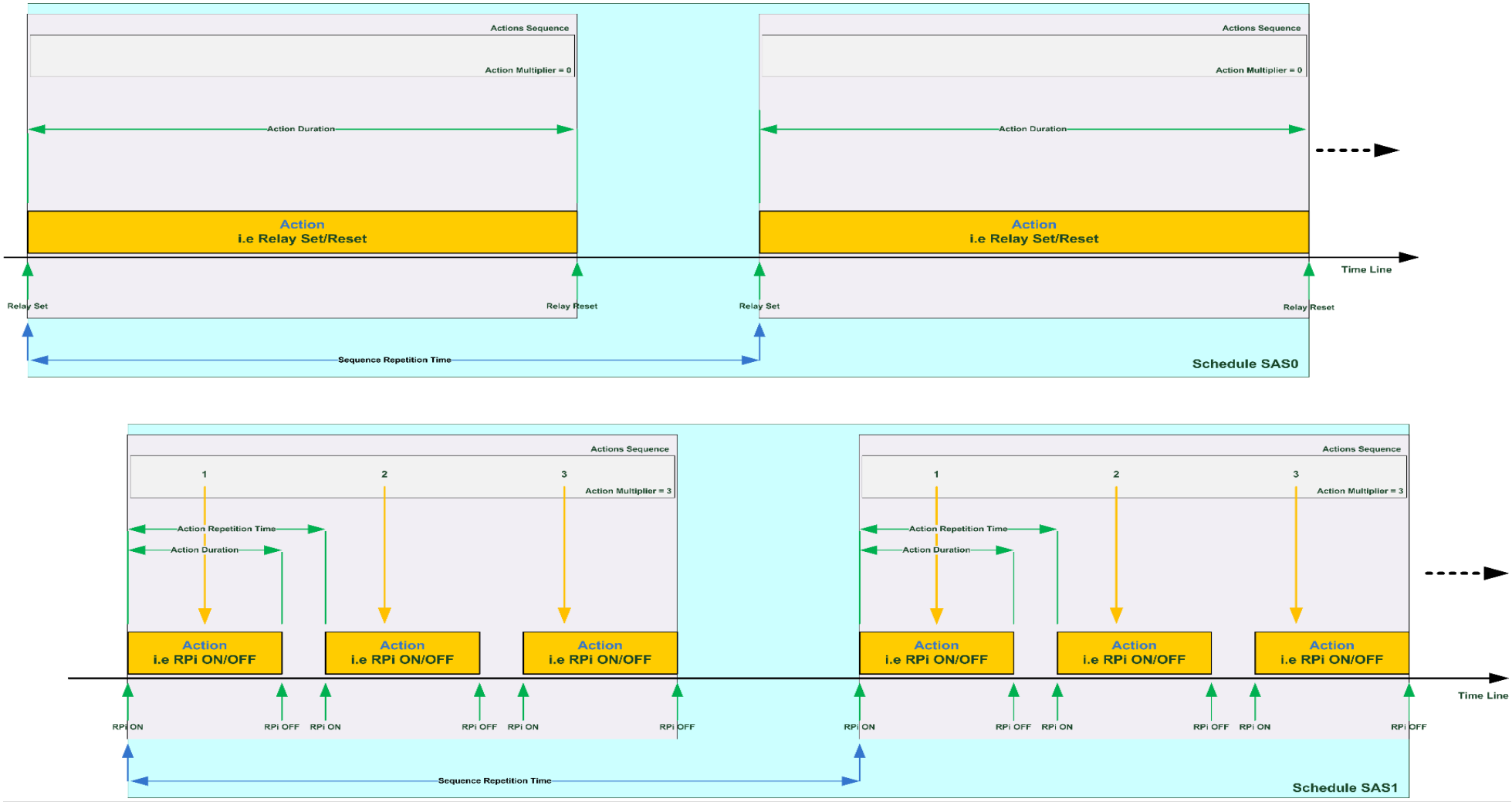


Figure 66 Graphical Presentation of ETR SAS definitions

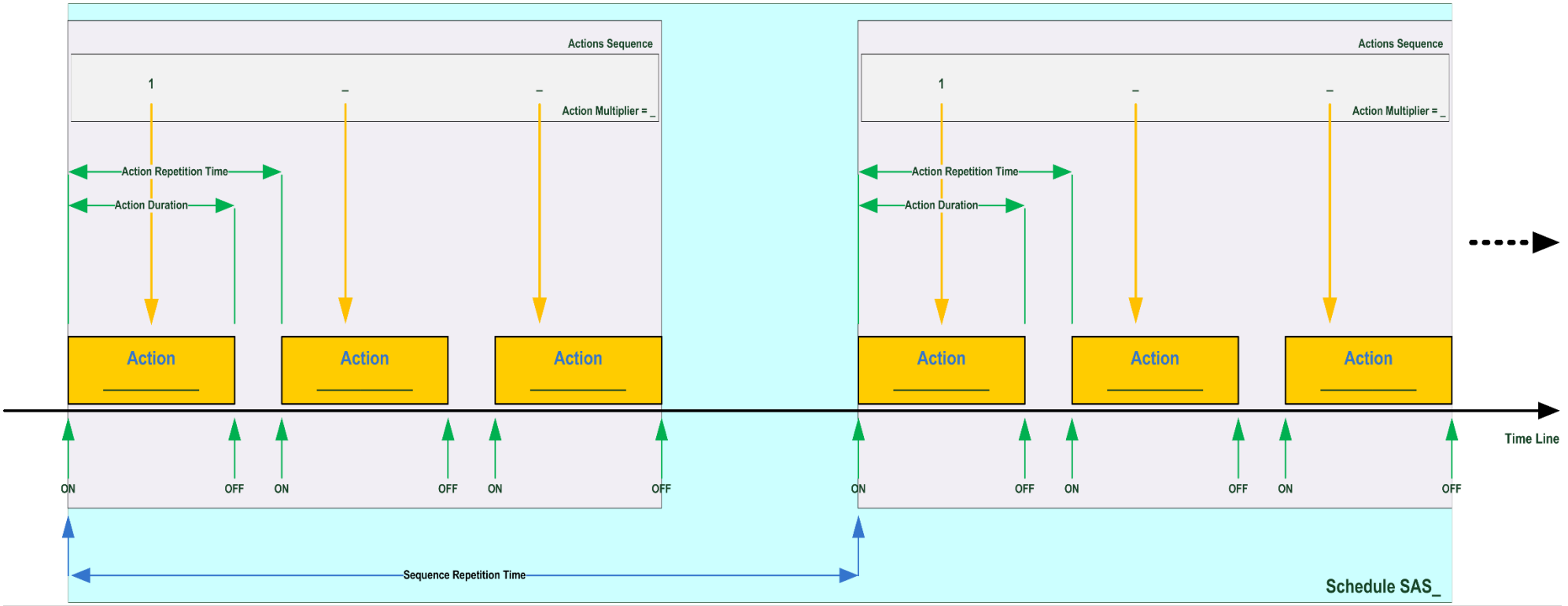


Figure 67 Template for ETR SAS user preparation

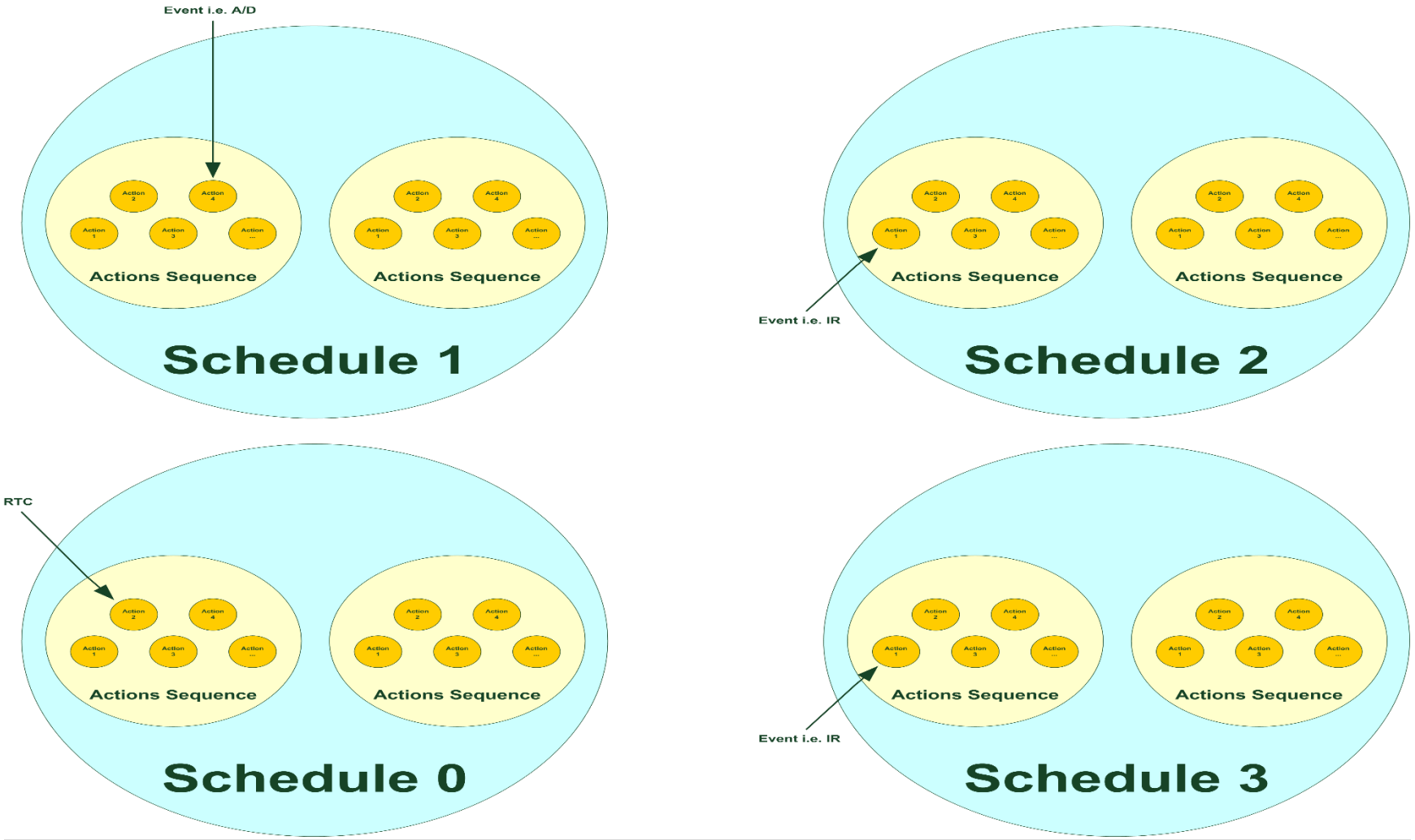


Figure 68 ETR SAS Definitions Dependencies

ETR SAS Involved PICo Registers/Sets

The following PICo Registers are involved in the **ETR SAS** Schedules programming.

0x6A -> UPS PIco Hardware RTC Registers Direct Access

This set of registers has mirror image of the Hardware UPS PIco HV3.0 RTC (each value separately). They can be read at any time. However, cannot be written as it will be overwritten by UPS PIco HV3.0 firmware. They can be used for reference of the HW RTC for various application.

0x6B -> UPS PIco 0x16 SAS Selection Register

This register is used to select the current SAS (from 0 – 3) for programming purposes. As far the **ETR SAS** is programming (set-up) user can write values to it selecting the active **ETR SAS** for programming or reading related values. After ETR SAS activation, values will be overwritten with current ETR SAS execution.

0x6B -> UPS PIco 0x17 SAS RUN Register

This register is used to make **ETR SAS** running (all activated Schedules). By setting the SAS RUN all activated SAS Scheduled Actions will be executed in their time frame. It is not possible to change any related to ETR SAS register value when SAS RUN is active. To do so, you need to deactivate the SAS RUN first.

0x6B -> UPS PIco 0x18 Next_Action_Rtime Register

This register is used to give information to the user about remining time for the next ongoing event (for any Schedule and any Sequence – just next one). This information is given when remining time to the next Action is smaller than 24 hours (1439 minutes)

0x6B -> UPS PIco 0x19 Time Scheduler Selector

This register is used to select the System Time Scheduler. Two values are possible 0x00 (default value) – which means **Basic Scheduler**, or 0x01 – which means **Event Triggered RTC Based System Actions Scheduler**. Setting of it is necessary to select the proper System Time Scheduler.

0x6c -> ETR SAS Start Time Stamp Registers

This set of registers holds all required values to set up the Start Time of the Action(s)

0x6d -> ETR SAS Actions Running Time Stamp

This set of registers holds all required values to set up the Action Running Time Stamp, Repetitions, Multiplier, etc.

0x6e -> ETR SAS Actions Stamp

This set of registers holds all defined Actions that can be used (i.e. Raspberry Pi® ON/OFF, Bi-Stable Relay ON/OFF etc.)

0x6f -> ETR SAS Events Stamp

This set of registers contains all defined Events that can be used for Actions Triggering

ETR SAS Working Examples

Please check below settings to understand exactly and properly the meanings and the **ETR SAS** functionality. Below examples with their settings up should be used as leads for user dedicated Schedules settings.

Definition of the 1st Example - Simple Raspberry Pi® ON/OFF Schedule executed 1 time for 1 minutes and repeated every day

We need to start up - set ON - the Raspberry Pi® at 10:00 (date for the first-time start-up is 20th August 2017). The Raspberry Pi® will run for 1 minute (executing their tasks), then system will shut down. Next day the above Sequence (has only one Action) will be repeated. Please check below settings to understand exactly and properly the meaning of the **ETR SAS** vocabulary. Involved Registers and programmed values for this example are the following:

0x6A -> UPS PIco Hardware RTC Registers Direct Access

Not needed. Can be used just for monitoring or any other application.

0x6B -> UPS PIco 0x16 SAS Selection Register

SAS_number = 0x00; means it will use for that Schedule the SAS0

0x6B -> UPS PIco 0x17 SAS RUN Register

SAS_RUN = 0x00; during programming phase and then must be set to 0x01 to make ETR SAS running

0x6B -> UPS PIco 0x19 Time Scheduler Selector

Time_Scheduler_Selector = 0x01; This register is used to select the System Time Scheduler. Two values are possible 0x00 (default value) – which means **Basic Scheduler**, or 0x01 – which means **Event Triggered RTC Based System Actions Scheduler**. Setting of it is necessary to select the proper System Time Scheduler.

0x6c -> ETR SAS Start Time Stamp Registers

Active = 0x01; means that this Sequence is active when Scheduler will be activated (so will be executed). Sequence contains one or more Actions.

Year = 0x17; means it will start at Year 2017

Month = 0x08; means it will start at August

Mday = 0x20; means it will start at 20th of declared above month (August)

Hour = 0x10; means it will start at 10 morning time

Minute = 0x00; means it will start exactly at 10:00

0x6d -> ETR SAS Actions Running Time Stamp

Action Duration = 0001; means it will run for 1 minute

Action Repetition Time = 00; means it will be not repeated within the same sequence (any value is allowed if Action Multiplier = 0x01)

Action Multiplier = 01; means it will happen 1 time within the same Sequence

Sequence Repetition Time = 24; means this (above described) sequence will be repeated every day (every 24 hours).

0x6e -> ETR SAS Actions Stamp

Action RPi_PON = 0x01; means that Action of this Scheduler will be **Raspberry Pi® ON** and (after Action Duration Time - in our example 1 minute) **OFF**

0x6f -> ETR SAS Events Stamp

Not involved. Keep all values 0x00. Does not matter whatever is entered.

Definition of the 2nd Example - Simple Bi-Stable Relay ON/OFF Schedule executed 1 time for 3 minutes and repeated every day.

We need to start up - set ON - the Bi-stable Relay at 09:59 (date for the first-time start-up is 20th August 2017). The Bi-stable Relay will run for 3 minutes starting up one minute before the Raspberry Pi® on Example 1st (executing their tasks), then 1 minute after will be OFF. Next day the above Sequence (that has only one Action) will be repeated. If both Schedules will be activated then i.e. powering can be supplied to external device, Raspberry Pi® then will be activated and when task is finished, Raspberry Pi and then (after one minute) the external device will be deactivated.

Please check below settings to understand exactly and properly the meaning of the **ETR SAS** vocabulary. Involved Registers and programmed values for this example are the following:

0x6A -> UPS PIco Hardware RTC Registers Direct Access

Not needed. Can be used just for monitoring or any other application.

0x6B -> UPS PIco 0x16 SAS Selection Register

SAS_number = 0x01; means it will use for that Schedule the SAS1

0x6B -> UPS PIco 0x17 SAS RUN Register

SAS_RUN = 0x00; during programming phase and then must be set to 0x01 to make **ETR SAS** running

0x6B -> UPS PIco 0x19 Time Scheduler Selector

Time_Scheduler_Selector = 0x01; This register is used to select the System Time Scheduler. Two values are possible 0x00 (default value) – which means **Basic Scheduler**, or 0x01 – which means **Event Triggered RTC Based System Actions Scheduler**. Setting of it is necessary to select the proper System Time Scheduler.

0x6c -> ETR SAS Start Time Stamp Registers

Active = 0x01; means that this Sequence is active when Scheduler will be activated (so will be executed). Sequence contains one or more Actions.

Year = 0x17; means it will start at Year 2017

Month = 0x08; means it will start at August

Mday = 0x20; means it will start at 20th of declared above month (August)

Hour = 0x09; means it will start at 09-hour morning time

Minute = 0x59; means it will start exactly at 09:59

0x6d -> ETR SAS Actions Running Time Stamp

Action Duration = 0003; means it will run for 3 minutes (2 minutes longer than Raspberry Pi® in the 1st Example)

Action Repetition Time = 00; means it will be not repeated within the same sequence (any value is allowed if Action Multiplier = 0x01)

Action Multiplier = 01; means it will happen 1 time within the same Sequence

Sequence Repetition Time = 24; means this (above described) sequence (that hold one action) will be repeated every day (every 24 hours).

0x6e -> ETR SAS Actions Stamp

Action BR_Set = 0x01; means that Action of this Scheduler will be **Bi-Stable Relay Set** and (after Action Duration Time) **Reset**

0x6f -> ETR SAS Events Stamp

Not involved. Keep all values 0x00.

Definition of the 3rd Example - Simple Raspberry Pi® ON/OFF Schedule executed 60 times for 1 minute every 2 minutes and repeated every day

We need to start up - set ON - the Raspberry Pi® at 10:00 (date for the first start-up is 20th August 2017). The Raspberry Pi® will run for 1 minutes (executing their tasks – i.e. making photos), then system will shut down. This will be repeated every 2 minutes for 60 times (totally for 2 hours – 120 minutes). Next day the above Sequence (having 60 Actions) will be repeated. Please check below settings to understand exactly and properly the meaning of the **ETR SAS** vocabulary. Involved Registers and programmed values for this example are the following:

0x6A -> UPS PIco Hardware RTC Registers Direct Access

Not needed. Can be used just for monitoring or any other application.

0x6B -> UPS PIco 0x16 SAS Selection Register

SAS_number = 0x00; means it will use for that Schedule the SAS0 (not use this with above examples)

0x6B -> UPS PIco 0x17 SAS RUN Register

SAS_RUN = 0x00; during programming phase and then must be set to 0x01 to make **ETR SAS** running

0x6B -> UPS PIco 0x19 Time Scheduler Selector

Time_Scheduler_Selector = 0x01; This register is used to select the System Time Scheduler. Two values are possible 0x00 (default value) – which means **Basic Scheduler**, or 0x01 – which means **Event Triggered RTC Based System Actions Scheduler**. Setting of it is necessary to select the proper System Time Scheduler

0x6c -> ETR SAS Start Time Stamp Registers

Active = 0x01; means that this Sequence is active when Scheduler will be activated (so will be executed). Sequence contains one or more Actions.

Year = 0x17; means it will start at Year 2017

Month = 0x08; means it will start at August

Mday = 0x20; means it will start at 20th of declared above month (August)

Hour = 0x10; means it will start at 10 morning time

Minute = 0x00; means it will start exactly at 10:00

0x6d -> ETR SAS Actions Running Time Stamp

Action Duration = 0001; means it will run for 1 minute

Action Repetition Time = 02; means it will be repeated after 2 minutes from starting time of the earlier one

Action Multiplier = 60; means it will happen 60 times within the same Sequence

Sequence Repetition Time = 24; means this (above described) sequence will be repeated every day (every 24 hours).

0x6e -> ETR SAS Actions Stamp

Action RPi_PON = 0x01; means that Action of this Scheduler will be **Raspberry Pi® ON** and (after Action Duration Time) **OFF**

0x6f -> ETR SAS Events Stamp

Not involved. Keep all values 0x00.

Setting-up the ETR SAS

To set-up the **ETR SAS** user need to continue with some simple steps. Current firmware implementation requires I^2C interface, however to simplify the settings-up, a monitoring of settings is implemented also via serial port. It is not necessary to use this monitoring tool, however it is very usefully, as each setting up step is confirmed by messages send to the Serial Terminal. It is also possible to use a simple python script that is printing out on the SSH the status read from the I^2C (and then the Serial Port Terminal – i.e. minicom – is not needed).

If user like to use this Serial Port Monitoring option, then need to activate the Serial Port on the **UPS PIco HV3.0A HAT**, and make available the serial port on the Raspberry Pi®, as also start second SSH session where the minicom will be running. To do that please follow below steps:

Make sure that the serial port in Raspberry Pi® is free for user application. This task has been described in very details in the chapter related to the firmware update.

Make sure that i.e. minicom is installed

sudo apt-get install minicom

Activate the Serial Port in the **UPS PIco HV3.0A HAT** i.e. to **38400** bps

sudo i2cset -y 1 0x6b 0x02 0x04

Start second SSH session or terminal and on the second SSH run the minicom

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

With or without monitoring, user can start the setting-up of the **ETR SAS**. To set-up it is needed to program **ETR SAS** registers for depending application need to be time scheduled.

There are 4, independent timed, **ETR SASs** called **Schedules** running at the same time. Only these ones that has been activated will be executed (so if user activate the SAS0 - will be executed the SAS0 only, if user activate the SAS0 and SAS1 – will be executed SAS0 and SAS1, etc.). They need to be setup to have a working system (only those are used). After programming user need to set the **Scheduler** running (by setting ON the 0x6B -> UPS PIco 0x17 SAS RUN Register), and have **Schedules** to be executed. Therefore, if one or more ETR SAS is not used, then need to be deactivated and not need to be set-up. The setup procedure is executed for the selected **ETR SAS**, so to setup it; need to be selected one, and this one that will be programmed (set-up). There is no need to program all **ETR SAS**, just these ones that will be running (used). If one or more ETR SAS selected to be used, user need to set its number on the selection register placed on the **0x6b** address and location **0x16**, each one when programming it.

0x16	SAS_number	Byte	Common	R/W	Define the current ETR SAS that is programmed or read. Default is 0. Allowed number are 0,1,2,3.
0x17	SAS_RUN	Byte	Common	R/W	Specify when **Scheduler** is running or not. Default is 0x00 (not running). Allowed values are 0x00 and 0x01
0x18	Next_Action_Rtime	Word	Mirror	R/W	Specify the minutes of the next Action in minutes if it is less than 24 hours to their execution (1439 minutes or less)
0x19	Time_Scheduler_Selector	Byte	Common	R/W	Selects which **Scheduler** is used: - 0x00 (default) Basic Scheduler - 0x01 ETR SAS Only one can be selected, and each programming is referred to it.

Table 15 ETR SAS Registers in the 0x6B set

First, user need to make selected the ETR SAS by writing to the **Time_Scheduler_Selector** register. Two values are possible 0x00 (default value) – which means **Basic Scheduler**, or 0x01 – which means **Event Triggered RTC Based System Actions Scheduler**. Setting of it is necessary to select the proper System Time Scheduler. For the case of the ETR SAS (current examples), the following should be done

sudo i2cset -y 1 0x6b 0x19 0x01 to select **ETR SAS**

Before user start programming or reading anything need to select the **SAS_number** by writing to it required number (i.e. 0x00) that all steps are referring to it.

sudo i2cset -y 1 0x6b 0x16 0x00

The 0x00 is the default value set by the **UPS PIco HV3.0A HAT** firmware. Each below programming steps will be addressed to this one selected with above command.

This follows all programing steps as described below. All of them are specified to selected **SAS_number**.

Here below are presented step by step programming each of examples defined above.

Setting-up them user should know that each entry is basically protected from wrong data entering. If user write a wrong data system will inform about it, with multiple blinking of User LEDs (all three) and long beep, alternatively if data are accepted it will just blink once and short beep. In addition, messages are send over the PIco Serial Port to the Raspberry Pi®. However, system is not protected from any kind of user mistakes, therefore it is usefully to use the given empty paper printed template (in PDF format) to have a better view of your current under preparation Schedule.

Setting Up of the 1st Example - Simple Raspberry Pi® ON/OFF Schedule executed 1 time for 1 minute and repeated every day

0x6A -> UPS PIco Hardware RTC Registers Direct Access

Setting up of these Registers is not possible. There are used only for monitoring.

0x6B -> UPS PIco 0x16 SAS Selection Register

Before start setting-up of ETR SAS user need to select the current Scheduler. In case of the 1st Example it is number 0;

sudo i2cset -y 1 0x6b 0x16 0x00

0x6c -> ETR SAS Start Time Stamp Registers

Setting up of the ETR SAS **Start Time Stamp** require to program the following registers, as described in detail here below:

Address	Name	Size	Type	R/W	Explanation
0x00	active	Byte	Common	R/W	Activation Stamp 0x00 not active (Stop), 0xff active (Start) of current SAS (Scheduler)
0x01	minute	Byte	Common	R/W	Starting Minute of hour in BCD - 2 digits (0-59) i.e. 22
0x02	hour	Byte	Common	R/W	Starting Hour of the Day in BCD - 2 digits (0-23) i.e. 22
0x03	mday	Byte	Common	R/W	Starting Day of the Month in BCD - 2 digits (1-31) i.e. 22
0x04	month	Byte	Common	R/W	Starting Month in BCD - 2 digits (1-12) i.e. 12
0x05	year	Byte	Common	R/W	Starting Year in BCD - 2 digits (0-99) i.e. 16

Table 16 ETR SAS Start Time Stamp Registers

According to specification of the 1st Example need to set below values:

Active = 0x01; means that this Sequence is active when Scheduler will be activated (so will be executed when scheduler is running). Sequence have one or more Actions.

Year = 0x17; means it will start at Year 2017

Month = 0x08; means it will start at August

Mday = 0x20; means it will start at 20th of declared above month (August)

Hour = 0x10; means it will start at 10 hours morning time

Minute = 0x00; means it will start exactly at 10:00

The data entering should looks like below (it important to follow the below order to avoid any mistake in programming):

1. Activate the current (already selected ETR SAS) by entering 0xff or (0x01) to the proper register

sudo i2cset -y 1 0x6c 0x00 0x01 for making the Schedule (ETR SAS) Active

2. Enter **Year** for start, year must be the same with actual or higher, and **not** the past. Two digits only in BCD format i.e. 2017 should be 0x17. Possible to program up to 2099.

   sudo i2cset -y 1 0x6c 0x05 0x17 for starting year (in BCD)

3. Enter **Month** for start, month must be the same with actual or higher, and **not** the past. Two digits in BCD format i.e. August should be 0x08. Allowed from 0x01 up to 0x12.

   sudo i2cset -y 1 0x6c 0x04 0x08 for starting month (in BCD)

4. Enter **Month Date** for start, month day must be the same with actual or higher, and **not** the past. Two digits in BCD format i.e. 10th should be 0x10 Allowed from 0x01 up to 0x31. Current version of firmware recognizes the leap year (so February as 28/29), as also end of months as 30/31 so user not need to take extra care to avoid such mistakes. Just when entering the improper Month Date, wrong data will be not accepted.

   sudo i2cset -y 1 0x6c 0x03 0x20 for starting month date (in BCD)

5. Enter **Hour** for start, hour must be the same with actual or higher, and **not** the past. Two digits in BCD format i.e. 20:00 should be 0x20. Allowed from 0x00 up to 0x23, and always in 24h format.

   sudo i2cset -y 1 0x6c 0x02 0x10 for starting hour (in BCD)

6. Enter **Minute** for start, minute must be the same with actual or higher, and **not** the past. Two digits in BCD format i.e. 45th should be 0x45. Allowed from 0x00 up to 0x59.

   sudo i2cset -y 1 0x6c 0x01 0x00 for starting minute (in BCD)

7. A simple python script can be used. This script can be executed at any time during programming of any registered of ETR SAS.

sudo python status_etrsas.py

Below steps enters Duration of Running and Repetition time.

0x6d -> ETR SAS Actions Running Time Stamp

The second **ETR SAS** registers set is the **Running Time Stamp**, located at **0x6d**. It contains set of 4 registers, that specify detailed time **Action Duration** should run, as also the **ETR SAS Action Repetition Time, Action Multiplayer** and **Sequence Repetition Time** Register.

Setting-up them user should know that each entry is protected from wrong data. If user write a wrong data system will inform about it, with multiple blinking of User LEDs (all three) and long beep, alternatively if data are accepted it will just blink once and short beep. In addition, messages are send over the PIco Serial Port to the Raspberry Pi® like in previous settings. By default, all **Repetitions** are set to default values as specified in below table.

Address	Name	Size	Type	R/W	Explanation
0x00	Action_Duration	Word	Common	R/W	In minutes. From 1-1439 minutes (23hours and 59 minutes), default value is 0x0001 (Action Duration of 1 minute) Higher numbers than 1439 (decimal) will be ignored 0xffff in hex means a special action (not implemented yet)
0x02	Action_Repetition_Time	Byte	Common	R/W	Define Repeated Time of the same Action in minutes. Measures time from beginning of Previous Action to the beginning of the next one. Time from 1 to 255 every XX minutes: 00 – not repeated (only once for Duration time, on programed time) i.e. 0x0010 means every 10 minutes (from one Action beginning to next Action beginning) Default value is 00 (no repetition)
0x03	Action_Multiplier	byte	Common	R/W	Defines how many times the Action will be repeated within the same Sequence (0 – 255) Default value is 0, Action not repeated Causion: If One of the registeres **0x02** and **0x03** is 0, action is not repeated.
0x04	Sequence_Repetition_Time	byte	Common	R/W	Define Repeated Time of the Saquence in hours. Measures time from beginning of Previous Sequence to the beginning of the next one. within the same Schedule. 1 – every hour 2 – every 2 hours .... 24 – every 24 hours The total time of repeated Actions must be lower that repetition time. System check it automaticly and wrong values are rejected. A message and LED blinking oocurs. Default value is 24, Sequence will be repeated next day The total time of (**Action_Duration** + **Action_Repetition_Time**) * **Action_Multiplier** must be smaller of the **Sequence_Repetition_Time**

Table 17 ETR SAS Actions Running Time Stamp

According to specification of the 1st Example need to set below values:

Action Duration = 0x0001; means it will run for 1 minute

Action Repetition Time = 0x02; means it will be repeated after 2 minutes from starting time

Action Multiplier = 0x060; means it will happen 60 times within the same Sequence

Sequence Repetition Time = 0x24; means this (above described) sequence will be repeated every day (every 24 hours).

The data entering should looks like below (it important to follow the below order to avoid any mistake in programming):

1. Enter **Action_Duration** to specify how long system must running from 1-1439 minutes.

   ***sudo i2cset -y 1 0x6d 0x00 0001* w** running for 1 minutes

2. Enter **Action_Repetition_Time** to define the Repeated Time of the same Action in minutes. Measuring time from beginning of Previous Action to the beginning of the next one

   sudo i2cset -y 1 0x6d 0x02 02 repeated every 2 minutes

3. Enter **Action_Multiplier** to define how many times the **Action** will be repeated within the same **Sequence**

   sudo i2cset -y 1 0x6d 0x03 60 repeated 60 times

4. Enter **Sequence_Repetition_Time** to define Repeated Time of the Saquence

   sudo i2cset -y 1 0x6d 0x04 24 repeated every 24 hours

0x6e -> ETR SAS Actions Stamp

The third **ETR SAS** registers set is the **Actions Stamp**, located at **0x6e**. Current firmware implementation contains set of 3 registers, that specify **Action Stamps** should be activated if used. The same **Actions** can be assigned to various Schedules, as also the same Schedule can have (activate) more than one **Action**. User need to carefully check the activated actions before use them. It is very usefully and strongly recommended to draw on a paper the scenario planned to be used, helpfully for doing that is the **Template for ETR SAS user preparation.**

Address	Name	Size	Type	R/W	Explanation
0x00	RPi_PON	Byte	Common	R/W	Raspberry Pi Power ON Activate: 0x01 Deactivate: 0x00 Default: 0x00 Write 0x01 to have this Action Active and switch Raspberry Pi® Powering ON
0x01	5V_PON	Byte	Common	R/W	Auxiliary 5V@750mA and 3V3 Power ON Activate: 0x01 Deactivate: 0x00 Default: 0x00

					Write 0x01 to have this Action Active and switch Auxiliary 5V@750mA and 3V3 Powering ON
0x02	BR_Set	Byte	Common	R/W	Bi-Stable Relay Set Activate: 0x01 Deactivate: 0x00 Default: 0x00 Write 0x01 to have this Action Active and Set the Bi-Stable Relay

Table 18 ETR SAS Actions Stamp

Acording to specification of the 1st Example need to set below values:

Action RPi_PON = 0x01; means that Action of this Scheduler will be **Raspberry Pi® ON** and (after Action Duration Time) **OFF**

The data entering should looks like below (it important to follow the below order to avoid any mistake in programming):

1. Enter **Action RPi_PON** to specify what action need to be executed

   sudo i2cset -y 1 0x6e 0x00 01 for RPi_PON = 0x01

0x6f -> ETR SAS Events Stamp

Not involved, as also not implemented in this version of firmware.

Setting Up of the 2nd Example - Simple Bi-Stable Relay ON/OFF Schedule executed 1 time for 15 minutes and repeated every day.

0x6A -> UPS PIco Hardware RTC Registers Direct Access

Setting up of these Registers is not possible. There are used only for monitoring.

0x6B -> UPS PIco 0x16 SAS Selection Register

Before start setting-up of ETR SAS user need to select the current Scheduler. In case of the 2nd Example it is number 1;

sudo i2cset -y 1 0x6b 0x16 0x01

0x6c -> ETR SAS Start Time Stamp Registers

Setting up of the **ETR SAS Start Time Stamp** need to program their registers, as described in detail here below. Acording to specification of the 1st Example need to set below values:

Active = 0x01;	means that this Sequence is active when Scheduler will be activated (so will be executed). Sequence conatin one or more Actions.
Year = 0x17;	means it will start at Year 2017
Month = 0x08;	means it will start at August
Mday = 0x20;	means it will start at 20th of delclared above month (August)
Hour = 0x09;	means it will start at 09 morning time
Minute = 0x59;	means it will start exactly at 09:59

The data entering should looks like below (it important to follow the below order to avoid any mistake in programming):

1. Activate the current (already selected ETR SAS) by entering 0xff or (0x01) to the proper register

   sudo i2cset -y 1 0x6c 0x00 0x01 for making the Schedule (ETR SAS) Active

2. Enter **Year** for start, year must be the same with actual or higher, and not the past. Two digits only in BCD format i.e. 2017 should be 0x17. Possible to program up to 2099.

   sudo i2cset -y 1 0x6c 0x05 0x17 for starting year (in BCD)

3. Enter **Month** for start, month must be the same with actual or higher, and not the past. Two digits in BCD format i.e. August should be 0x08. Allowed from 0x01 up to 0x12.

sudo i2cset -y 1 0x6c 0x04 0x08 for starting month (in BCD)

4. Enter **Month Date** for start, month day must be the same with actual or higher, and not the past. Two digits in BCD format i.e. 10th should be 0x10 Allowed from 0x01 up to 0x31. Current version of firmware recognizes the leap year (so February as 28/29), as also end of months as 30/31 so user not need to take extra care to avoid such mistakes. Just when entering the unproper Month Date, wrong data will be not acceted.

   sudo i2cset -y 1 0x6c 0x03 0x20 for starting month date (in BCD)

5. Enter **Hour** for start, hour must be the same with actual or higher, and not the past. Two digits in BCD format i.e. 20:00 should be 0x20. Allowed from 0x00 up to 0x23, and always in 24h format.

   sudo i2cset -y 1 0x6c 0x02 0x09 for starting hour (in BCD)

6. Enter **Minute** for start, minute must be the same with actual or higher, and not the past. Two digits in BCD format i.e. 59th should be 0x59. Allowed from 0x00 up to 0x59.

   sudo i2cset -y 1 0x6c 0x01 0x59 for starting minute (in BCD)

7. Use this simple command line (without any python script) to check the entered and stored data at any time. User can write his own simple python script if needed. This line can be executed at any time during entry process.

   sudo i2cget -y 1 0x6c 0x05 && i2cget -y 1 0x6c 0x04 && i2cget -y 1 0x6c 0x03 && i2cget -y 1 0x6c 0x02 && i2cget -y 1 0x6c 0x01 && && i2cget -y 1 0x6c 0x00

Below steps enters Duration of Running and Repetition time.

0x6d -> ETR SAS Actions Running Time Stamp

The second **ETR SAS** registers set is the **Running Time Stamp**, located at **0x6d**. Acording to specification of the 1st Example need to set below values:

Action Duration = 0x0001; means it will run for 1 minute

Action Repetition Time = 0x02; means it will be repeated afte 2 minutes from starting time

Action Multiplier = 0x060; means it will happen 60 times within the same Sequence

Sequence Repetition Time = 0x24; means this (above described) sequence will be repeated every day (every 24 hours).

The data entering should looks like below (it important to follow the below order to avoid any mistake in programming):

1. Enter **Action_Duration** to specify how long system must running from 1-1439 minutes.

   sudo i2cset -y 1 0x6d 0x00 0001 **w** running for 1 minute

2. Enter **Action_Repetition_Time** to define the Repeated Time of the same Action in minutes. Measuring time from beginning of Previous Action to the beginning of the next one

   sudo i2cset -y 1 0x6d 0x02 02 repeated every 2 minutes

3. Enter **Action_Multiplier** to define how many times the **Action** will be repeated within the same **Sequence**

   sudo i2cset -y 1 0x6d 0x03 60 repeated 60 times

4. Enter **Sequence_Repetition_Time** to define Repeated Time of the Saquence

   sudo i2cset -y 1 0x6d 0x04 24 repeated every 24 hours

0x6e -> ETR SAS Actions Stamp

The third **ETR SAS** registers set is the **Actions Stamp**, located at **0x6e**. Acording to specification of the 1st Example need to set below values:

Action RPi_PON = 0x01; means that Action of this Scheduler will be **Raspberry Pi® ON** and (after Action Duration Time) **OFF**

The data entering should looks like below (it important to follow the below order to avoid any mistake in programming):

2. Enter **Action RPi_PON** to specify what action need to be executed

   sudo i2cset -y 1 0x6e 0x00 01 for RPi_PON = 0x01

0x6f -> ETR SAS Events Stamp

Not involved, as also not implemented in this version of firmware.

Solar Panel Connectivity

UPS PIco HV3.0 HAT

Factory Defaults Setting

UPS PIco HV3.0 HAT is offering feature that allows to revert all setting to Factory Default at any time. To do this user has 2 ways: based on command line (automatic recall), and manually. Both are described here below.

Command Line Factory Defaults Recall

Write the following command on the Raspberry Pi® command line. It is recommended to recall factory defaults only when system is Cable Powered.

sudo i2cset -y 1 0x6b 0x00 0xdd

Manually Factory Defaults recall

You can do this instead of using the command line initiation outlined above. However, user need to have physically access to the device, as needs to push buttons.

The following procedure needs to be followed:

- Press and hold the **UR** button
- Continue to hold the **UR** button, and press and hold the **C** button.
- Release the **UR** button, but keep holding the **C** button
- Release after 2 second the **C** button

The User LEDs (all of them) will blinking for a short time, then **UPS PIco HV3.0 HAT** will be restarted. It is mandatory, to have system cable powered during factory recall process (totally about 5 seconds)

UPS PIco HV3.0 HAT settings on Factory Defaults recall

The following setting are loaded when Factory Defaults recall.

1. **UPS PIco HV3.0 HAT** EEPROM ERASE
2. **Still Alive Timer** sets to 0xff (OFF)
3. **Hardware RTC** sets to default values:

   2000-01-01 00:00:00
4. **FAN Speed** sets to 50%

5. **FAN** sets to Automatic mode, and running for a short time, so FAN LED lit
6. **FAN TO92 sensor** temperature sets to 35 Celsius
7. Auxiliary **5V@750mA** and **3.3V@150mA** sets to OFF
8. **Running on Battery** sets to 70 seconds
9. **A/D converters (setA_D**) sets to mode 0x00 (0-5.2V)
10. **LEDs** sets to be ON when needed (activated by PIco)
11. **User LEDs Mapping** set to 0x00 for each color
12. **UPS PIco HV3.0 HAT Serial Port A** sets to OFF (available to use Raspberry Pi® with other applications)
13. **UPS PIco HV3.0 HAT Serial Port B** sets to OFF
14. **I²C** sets to **DEFAULT** where used I2C addresses are: 0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F
15. **Battery Type** sets to default version of the one stored in the bootloader (LiPO)
16. **Power Monitoring Variables** are set to:
    - pwr_rfc_level=02
    - pwr_cut_level=03
    - pwr_spk_duration=25
17. **Battery Charger** set to ON
18. A/D Accuracy Improving System correction factor set to 0
19. **Bi-Stable Relay** (Latching) is reset
20. **Sysinfo** set to 0x0000
21. Selected **Basic Scheduler**, and deactivated, the following parameters are set:
    - Scheduler_Sel=0x01;
    - BS_action_time=0x01;
    - BS_inaction_time=0x01;
    - BS_Start_time=0xffff;
    - BS_multipier=0xff;
    - BS_RUN=0x00;

22. The **ETR SAS** sets to be de-activated, and values sets to:

- active=0x00;
- min=0x00;
- hour=0x00;
- mday=0x00;
- var.month=0x00;
- year=0x00

A Complete description of the UPS PIco HV3.0 HAT Programmers Registers

0x69 ->UPS PIco HV3.0 Module Status Registers Specification

Address	Name	Size	Type	R/W	Explanation
0x00	mode	Byte	Mirror	Read	Powering Mode – Read ONLY, Writing has no effect on the system and will be overwritten by UPS PIco HV3.0 with the new value **Read:** **0x01** - RPI_MODE (means cable powering mode USB or EPR) **0x02** - BAT_MODE
0x08	batlevel	Word	Mirror	Read	Means value of Battery Voltage in 10th of mV in BCD format
0x0a	rpilevel	Word	Mirror	Read	Means value of Voltage supplying RPi on J8 5V Pin in 10th of mV in BCD format
0x0c	eprlevel	Word	Mirror	Read	Means value of Extended Voltage supplying RPi on Extended Voltage input (7-28VDC) in 10th of mV in BCD format
0x0e	curilevel	Word	Mirror	Read	
0x10	curolevel	Word	Mirror	Read	
0x14	aEXT0level	Word	Mirror	Read	Means value of the first A/D converter pre-scaled to 5.2V. Higher voltage could not be supplied. Readings are in 10th of mV in BCD format
0x16	aEXT1level	Word	Mirror	Read	Means value of the second A/D converter pre-scaled to 5.2V. Higher voltage could be supplied with an external resistor divider. Readings are in 10th of mV in BCD format. **If added an extra resistor can be used as pre-scaled to 10, 20 or 30V.**
0x18	aEXT2level	Word	Mirror	Read	Means value of the second A/D converter pre scaled to 5.2V. Higher voltage could be supplied with an external resistor divider. Readings are in 10th of mV in BCD format. **If added an extra resistor can be used as pre scaled to 10, 20 or 30V.**
0x1a	key	Byte	Common	R/W	User Kay Pressed information **Read: 0x01** – Pressed key **A** **Read: 0x02** – Pressed key **B** **Read: 0x03** – Pressed key **C** **Write: 0x00** – Reset (clear) after the current reading and prepare for the next one.
0x1b	NTC	Byte	Mirror	Read	Temperature in Celsius degree of the embedded NTC1 sensor placed on the top of PCB. Values in BCD format.
0x1c	TO92	Byte	Mirror	Read	Temperature in Celsius degree of the TO-92 sensor placed on the bottom of PCB. It is valid only if this sensor is soldered. It is available in the PIco Fan Kit. Values in BCD format.
0x20	charger	Byte	Mirror	Read	Information about charger IC status. **Read: 0x00** – Charger IC is OFF and battery is not charged **Write: 0x01** – Charger IC is ON and battery is charged For Version UPS PIco HV3.0 Stack/TopEnd the charging

					current is fixed to 300 mA. For Version UPS PIco HV3.0 Plus the charging current is dynamically changed based on powering conditions on micro USB powering input or External Powering Input. The charging current on that version is from 100 mA – 800 mA. With maximum of the system to be 1200 mA (will be activated in the future). If the **charger** register is set ON, meant that charger circuits has been activated, however if battery is really charged depends to the internal conditions of the charger IC. When current is flowing to the battery (charging) then CHG LED is lighting continuously. It is possible that **charger** register can be read as 0x01, but CHG LED will be off, because system set to charge battery however buttery is full, so no current is flowing. The CHG LED always shows if current is flowing to battery or not. This is valid for all batteries chemistry types.
0x22	pico_is_running	Word	Mirror	Read	It is a 16-bit unsigned variable that value of it, is changing every 1 ms within the main loop of the firmware. Reading two times of this variable must return a different value (with interval longer than 1 ms), if not, means that system hangs-up, and need to be reset, if not restarted by other PIco protection internal mechanism (watch-dog, and supervising watch dog). As these protection mechanisms are always restarting the system when something goes wrong, reason of existence of this variable is just to confirm to the remote user that everything is working well and give feedback to the remote user that system is running properly. As it is a mirror variable, writing to it nothing change, will be again re-written with the newer internal value.
0x24	pv				PCB Version - current available versions: A, B, C
0x25	bv				Bootloader Version - current available versions: S - BL_PIco HV 3.0 Stack/TopEnd default LP Battery F - BL_PIco HV 3.0 Stack/TopEnd default LF Battery P - BL_PIco HV 3.0 Plus default LP Battery Q - BL_PIco HV 3.0 Plus default LF Battery
0x26	fv				Firmware Version: current 0x35 dated 01/05/2017

0x6A -> UPS PIco Hardware RTC Registers Direct Access Specification

Address	Name	Size	Type	R/W	Explanation
0x00	seconds	Byte	Mirror	Read	seconds in BCD
0x01	minutes	Byte	Mirror	Read	minutes in BCD
0x02	hours	Byte	Mirror	Read	hours in BCD
0x03	wday	Byte	Mirror	Read	week day in BCD
0x04	mday	Byte	Mirror	Read	month day in BCD
0x05	month	Byte	Mirror	Read	month in BCD
0x06	year	Byte	Mirror	Read	year in BCD

0x6B -> UPS PIco Module Commands

Address	Name	Size	Type	R/W	Explanation
0x00	pico_state	Byte	Common	R/W	**Write: 0xcc** – Unconditional File Safe Shutdown and (and Power OFF when battery powered) **Write: 0xdd** - then restore factory defaults Will stay in the values of 0xdd until factory defaults restored, and then will be set to 0x00 **Write: 0xee** - Reset the UPS PIco CPU, it cause start-up values i.e. RTC will be set to 01/01/2000 **Write: 0x11** – when battery is fully charged, and BAT led is not lit. This will be doing fine adjustment on all A/D converters (including one user A/D converters as also on system A/D converters). More info provided in the appropriate chapter. **Write:** 0xFF - Call the UPS PIco Bootloader, **Orange** Led will be light. Recover from this state can be done **only** by pressing the RST button, new firmware upload or automatically after 16 seconds if nothing happens. All interrupts are disabled during this procedure. It should be used with RPi Uploading firmware script. Use it very carefully and only when is needed – when firmware uploading. Do not play with it; this is not toy functionality. **Powering of the pair UPS PIco+RPi must be done via RPi micro USB socket during boot loading process due to following UPS PIco Resets after firmware uploading or when returning from this mode.** **Due to required protection for the RPi from the unconditional reset (files corruption), it is not possible to enter to this mode when system is powered in a different way than in RPI Powering Mode.**
0x01	bat_run_time	Byte	Common	R/W	**On Battery Powering Running Time when cable power loses or not exist. After that time a File Safe Shut Down Procedure will be executed, and System will be shut downed without restart. Battery power will be disconnected. System is in sleep mode (LPR) and RTC is running.** **If Raspberry Pi cable power returns again system will be start automatically.** **If during the sleep mode (LPR) the F button will be pressed for longer time than 2 seconds (with battery or cable powering) Raspberry Pi will re-start again.** **Value of 0xff (255) disable this timer, and**

					system will be running on battery powering until battery discharge to 3.4V for LP battery and 2.8V fro LF Battery type. **<u>Factory default value is 70 seconds</u>** **Each number stands for 1 minute of Battery Running. Default Value is 0, and the highest Value is 0xFE. If user will enter i.e. 2, the Battery Running time will be 60 seconds + 2 x 60 seconds = 180 seconds. After that time system will be shutdown. If user after that will press again F button system will restart and run for 180 seconds again and then shutdown.** **Read:** Anytime, Return actual **fssd_timeout** value **Write:** 0x00 – 0xFF **Any change on this register will cause immediate writing of the new value to the PIco EEPROM**
0x02	rs232_rate	Byte	Common	R/W	Writing to this register sets the UPS PIco HV3.0 Serial Port to the following settings: **0x00 - UPS PIco HV3.0** Serial Port is OFF **Default value** **0x01 - UPS PIco HV3.0** Serial Port is ON and data 115200 pbs
0x05	STA_timer	Byte	Common	R/W	**Still Alive Timeout Counter in seconds** **Read:** Anytime, Return actual **sta_timer** value **Write:** 0xff – Disable the counter (default value) **Write:** 0x01 – 0xfe Enable and Start down counting of the Still Alive Timer in resolution of 1 second, until reaches value of 0x00 which initiate Unconditional Hardware Reset Procedure **Write**:0x00 – Initiate immediately File Safe Shutdown Procedure and system restart with similar conditions as described below In order to use it as Still Alive (type of watchdog) timer, user needs to upload value from **0x01** to **0xfe** earlier than defined time of seconds. Not uploading of this value will cause System Unconditional Hardware Reset (so System to be Restarted) In order to have this feature working the Gold Plated Reset Pin must be instaled This feature is working on Battery or Cable powering

					After execution of the STA Restart the sta_timer is set again to 0xff (disabled).
0x06	enable5V	Byte	Common	R/W	Defines usage of the Auxiliary 5V@750mA: 0x00 – Auxiliary 5V and 3.3V are not battery backed-up 0x01 – Auxiliary 5V and 3.3V are battery backed-up **Default Values is OFF** **Other codes are not allowed**
0x07	battype	Byte	Common	R/W	Defines used battery chemistry type: 0x46 – LiFePO4 (ASCII: F) used in version Stack/TopEnd 0x51 – LiFePO4 (ASCII: Q) used in version Plus 0x53 – LiPO (ASCII: S) used in version Stack/TopEnd 0x50 – LiPO (ASCII: P) used in version Plus **Other codes are not allowed**
0x08	setA_D	Byte	Common	R/W	Defines the pre scaler of the **AEXT1level** and the **AEXT2level** registers. The 4th MSB bits are responsible for the **AEXT1level** pre-scale, and the 4th LSB bits are responsible for the **AEXT2level** pre-scale. **Read:** Anytime, Return actual **setA_D** value **Write:** 0x00 – 5.2V prescale for the AEXT**2level** **Write:** 0x01 – 10V prescale for the AEXT**2level** **Write:** 0x02 – 20V prescale for the AEXT**2level** **Write:** 0x03 – 30V prescale for the AEXT**2level** **Write:** 0x00 – 5.2V prescale for the AEXT**1level** **Write:** 0x10 – 10V prescale for the AEXT**1level** **Write:** 0x20 – 20V prescale for the AEXT**1level** **Write:** 0x30 – 30V prescale for the AEXT**1level** **Write:** 0xFF – all A/D registers will contain raw data **RED Marked** – not implemented yet
0x09	User LED Orange	Byte	Common	R/W	**User LED Orange ON - Write:** 0x01 **User LED Orange OFF - Write:** 0x00
0x0A	User LED Green	Byte	Common	R/W	**User LED Green ON - Write:** 0x01 **User LED Green OFF - Write:** 0x00
0x0B	User LED Blue	Byte	Common	R/W	**User LED Blue ON - Write:** 0x01 **User LED Blue OFF - Write:** 0x00
0x0C	brelay	Byte	Common	R/W	**Zero Power Bi Stable Relay** **Write:** 0x01 Set **Write:** 0x00 Reset
0x0D	bmode	Byte	Common	R/W	**Integrated Sounder Mode** **Read:** Anytime, Return actual **bmode** value **Write:** 0x00 – Unconditional Disable the Sounder **Write:** 0x01 – Unconditional Enable the Sounder **Default Value:** 0x01
0x0E	bfreq	Word	Common	R/W	**Frequency of sound in Hz**
0x10	bdur	Byte	Common	R/W	**Duration of sound in 10th of ms (10 = 100 ms)**
0x11	fmode	Byte	Common	R/W	**Integrated Fan Running Mode** **Read:** Anytime, Return actual **fmode** value

					Write: 0x00 – Unconditional Disable the FAN with selected speed from the **fspeed** **Write:** 0x01 – Unconditional Enable the FAN FAN with selected speed from the **fspeed** **Write:** 0x02 – Automatic ON/OFF with defined speed in the fspeed, ON when temperature read in sensor T0-92 is higher than **fttemp** threshold, OFF when lower. When UPS PIco is going down to the LPR mode, the FAN is automatically disabled, and enabled again when the UPS PIco returns to normal work Default value is set to 0x02 – Automatic ON/OFF
0x12	fspeed	Byte	Common	R/W	**Integrated Fan Speed** **Read:** Anytime, Return actual **fspeed** value **Write:** 00 – Selected speed when OFF is 0% (not running) **Write:** 100 – Selected speed when ON is 100% (full speed running) **Any other (0-100) number is allowed and means % of speed and current consumption** Default speed is set to 50% **Any data written to this register are stored in the internal EEPROM. So, even if UPS PIco HV3.0 will be reset, will be recovered.**
0x13	fstat	Byte	Mirror	Read	**Read:** Anytime, Return actual **if FAN** is actually running or not (for remote users) When FAN is set to be running (even if not connected physically) the FAN LED is lighting. The intensity of the FAN LED is depending of the FAN Speed (PWM)
0x14	fttemp	Byte	Mirror	R/W	**Integrated Fan Temperature Threshold in Automatic Mode** BCD Fan Running threshold temperature in Celsius, 2 digits i.e. 35, means 35 Celsius. In order to be used (automatic FAN ON/OFF) need to set **fmode** to 0x02. Maximum temperature is 60 Celsius. Higher values will be ignored. FAN will start at 36 Celsius and stop at 35 Celsius. **Read:** Anytime, Return actual **fspeed** value **Write:** 00 – 60 Sets the TO-92 temperature Threshold for the Automatic FAN Start/Stop **Default value is set to 35 Celsius**
0x15	LED_OFF	Byte	Common	R/W	Added LED OFF, that switch OFF all software controlled LEDs. CHG, FAN, EXT can not be switched off as are connected to the hardware and controlled by it. By writing the

					0x00 to LEDOFF disable the LEDs. Default is 0x01 (means LED ON)
0x16	SAS_number	Byte	Common	R/W	Define the current ETR SAS that is programmed or read. Default is 0. Allowed number are 0,1,2,3.
0x17	SAS_RUN	Byte	Common	R/W	Specify when **Scheduler** is running or not. Default is 0x00 (not running). Allowed values are 0x00 and 0x01
0x18	Next_Action_Rtime	Word	Mirror	R/W	Specify the minutes of the next Action in minutes if it is less than 24 hours to their execution (1439 minutes or less)
0x19	Time_Scheduler_Selector	Byte	Common	R/W	Selects which **Scheduler** is used: - 0x00 (default) Basic Scheduler - 0x01 ETR SAS Only one can be selected, and each programming is referred to it.
0x1A	BS_duration_time	Byte	Common	R/W	**Basic Scheduler Action Duration Time** in minutes. Allowed values are 0x01 – 0xfe. Default is 0x01
0x1B	BS_repetition_time	Byte	Common	R/W	**Basic Scheduler Action Repetition Time** in minutes. Allowed values are 0x01 – 0xff. Default is 0x02
0x1C	BS_multipier	Byte	Common	R/W	**Basic Scheduler Action Multiplier**. Allowed values are 0x00 – 0xff. Default is 0x01
0x1D	BS_ RUN	Byte	Common	R/W	Specify when **Basic Scheduler** is running or not. Default is 0x00 (not running). Allowed values are 0x00 and 0x01

Events Triggered RTC Based System Actions Scheduler Commands

0x6c -> Start Time Stamp

Address	Name	Size	Type	R/W	Explanation
0x00	active	Byte	Common	R/W	Activation Stamp 0x00 not active (Stop), 0xff active (Start) of current SAS
0x01	minute	Byte	Common	R/W	Starting Minute of hour in BCD - 2 digits (0-59) i.e. 22
0x02	hour	Byte	Common	R/W	Starting Hour of the Day in BCD - 2 digits (0-23) i.e. 22
0x03	mday	Byte	Common	R/W	Starting Day of the Month in BCD - 2 digits (1-31) i.e. 22
0x04	month	Byte	Common	R/W	Starting Month in BCD - 2 digits (1-12) i.e. 12
0x05	year	Byte	Common	R/W	Starting Year in BCD - 2 digits (0-99) i.e. 16
0x06	error	Byte	Mirror	Read	ETR SAS errors

0x6d -> Actions Running Time Stamp

Address	Name	Size	Type	R/W	Explanation
0x00	error	Byte	Mirror	Read	ETR SAS errors
0x01	Duration	Word	Common	R/W	In BCD 4 digits minutes 1-9999
0x03	Repetition	Word	Common	R/W	In BCD 2 digits (1-9999) every XXXX minutes: 0x0000 – not repeated (only once for Duration time, on programed time) 0x0001 – 0x9999 – every 1-9999 minutes i.e. 0x0010 means every 10 minutes

0x6e -> Events Stamp

Address	Name	Size	Type	R/W	Explanation

0x6f -> Actions Stamp

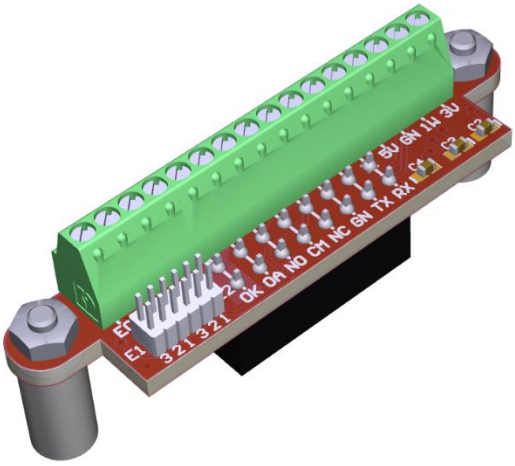
Currently Implemented Only Power Up System – permanently selected.

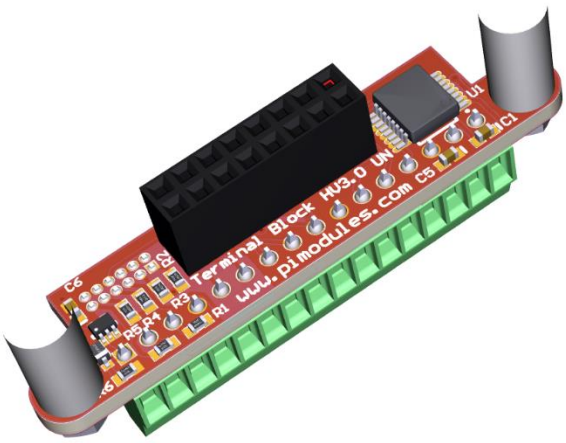
Address	Name	Size	Type	R/W	Explanation
0x00	RPi_PON	Byte	Common	R/W	Raspberry Pi Power ON Activate: 0x01 Deactivate: 0x00 Default: 0x00 Write 0x01 to have this Action Active and switch Raspberry Pi® Powering ON
0x01	5V_PON	Byte	Common	R/W	Auxiliary 5V@750mA and 3V3 Power ON Activate: 0x01

					Deactivate: 0x00 Default: 0x00 Write 0x01 to have this Action Active and switch Auxiliary 5V@750mA and 3V3 Powering ON
0x02	BR_Set	Byte	Common	R/W	Bi-Stable Relay Set Activate: 0x01 Deactivate: 0x00 Default: 0x00 Write 0x01 to have this Action Active and Set the Bi-Stable Relay

Future implementation: FAN, Charger, Relay, Auxiliary 5V, RPi, Sound, LED,

UPS PIco Terminals Block HV3.0 UNiversal PCB with Instrumental A/D, selectable voltages rage for each A/D and 12V RS232





User Guide

Designed for the UPS Pico HV3.0 A/B and B+

Compatible with

Raspberry Pi® 2, Pi Zero, A+, B+,

"Raspberry Pi" is a trademark of the Raspberry Pi® Foundation

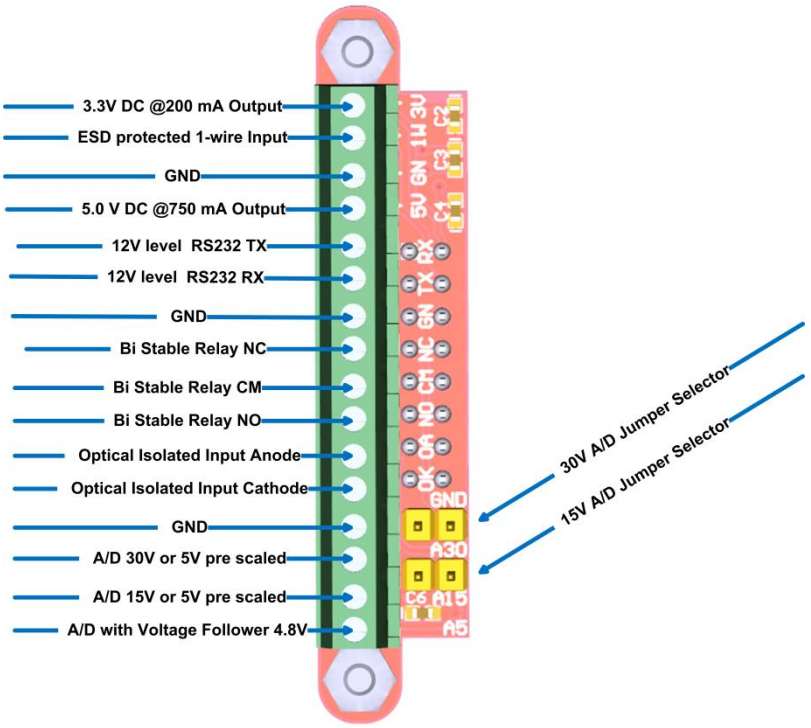
Introduction

The **UPS PIco Terminals Block HV3.0 Universal PCB** is an advanced Terminals Blocks PCB that adds a wealth of extra functionality and development features to the innovative **UPS PIco HV3.0 A/B/B+**!

The following listed features are offered by the **UPS PIco HV3.0 Terminal Blocks PCB**:

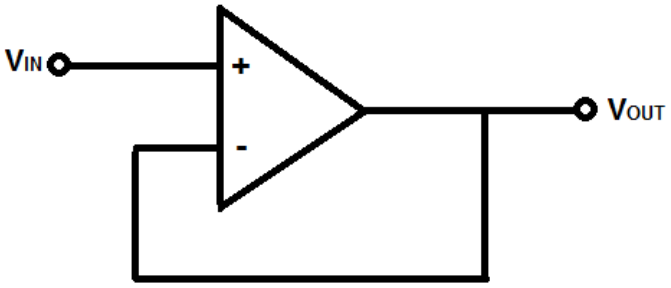
- Terminal Block Connectivity on Independent from Raspberry Pi, and battery backed-up 3.3 V 200 mA supply (available also when Raspberry Pi is not powered)
- Terminal Block Connectivity on ESD protected true 5V 1-wire interface
- Terminal Block Connectivity on Independent from Raspberry Pi and battery backed-up 5V source 750 mA (available also when Raspberry Pi is not powered)
- 12V RS232 Interface Level Converter connectable to the Raspberry Pi Primary Serial Port or Independent Secondary Serial Port offered by UPS PIco HV3.0A/B/B+ with Terminal Block Connectivity
- Terminal Block Connectivity on Auxiliary interface to the bi-stable (zero power) Relay offered by the UPS PIco HV3.0A/B/B+
- Terminal Block Connectivity on Optical Isolated Interface – readable as digital or analog input offered by the UPS PIco HV3.0A/B/B+
- Terminal Block Connectivity on ESD Protected 12-bit A/D converters pre-scaled to: 5V, 20V and 30V (user selectable) accessed by I^2C on Raspberry Pi®
- ESD input additionally protected 12-bit A/D converter with high impedance Voltage Follower and scaled of 0 - 4.8V A/D with Terminal Block Connectivity

Introduction



What is a Voltage Follower?

A **voltage follower** (also called a unity-gain amplifier, a buffer amplifier, and an isolation amplifier) is an op-amp circuit which has a voltage gain of 1.



This means that the op amp does not provide any amplification to the signal. The reason it is called a voltage follower is because the output voltage directly follows the input voltage, meaning the output voltage is the same as the input voltage. Thus, for example, if 10V goes into the op amp as input, 10V comes out as output. A voltage follower acts as a buffer, providing no amplification or attenuation to the signal.