Ultra Short Installation Procedure of the UPS PIco HV3.0 Daemons

and UPS PIco HV3.0 email broadcasting System

1. Install Raspberry Pi Operation System (i.e. NOOBs)
   - **Disable the serial port (only if you need to upload a new firmware) via raspi-config**
   - **Enable the I2C**

2. Ensure that Python is installed and updated, by using the following command

   sudo apt-get install python-rpi.gpio

3. Ensure to run below line

   sudo apt-get 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)

4. 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)

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

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

6. Move to the required directories where software has been copied.
7. First to the email broadcasting system (package)

   sudo cd PiModules/code/python/package

8. 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

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

For interaction of UPS PIco HV3.0 is used fixed Raspberry Pi GPIOs:
GPIO_GEN27 and GPIO_GEN22

These GPIOs are used for sending Train Pulse as also to initiate the Files Safe System Shutdown (FSSD) and should be not used in any other applcations

cd ../upspico/picofssd

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

    sudo python setup.py install

11. Once the script has been installed, it can be installed to the `SysVInit` system with the following command

    sudo update-rc.d picofssd defaults

12. Enable to run at boot time with the following command

    sudo update-rc.d picofssd enable

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

UPS LED (Blue) Indications	
UPS LED is OFF	System is not running or is in Low Power Mode (only HW RTC is running)
UPS LED is lighting continuously	System (PIco + RPi) is booting or shutting down
UPS LED is blinking every 600 ms	System (PIco + RPi) is running on cable powering (after booting time)
UPS LED is blinking every 1800 ms	System (PIco + RPi) is running on battery powering

Ultra Short Installation Procedure of the UPS PIco HV3.0 Hardware RTC

1. Proceed with the installation of the i2c-tools using the following command

   sudo apt-get install i2c-tools

2. Now edit the /etc/modules file

   sudo nano /etc/modules

3. Make sure to have the following items in the file and add what is missing

   i2c-bcm2708
   i2c-dev
   rtc-ds1307

4. Now edit the file /etc/rc.local

   sudo nano /etc/rc.local

5. and Add the following lines, before "**exit 0**"

   echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
   (sleep 4; hwclock -s) &

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

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

1. Disable Bluetooth. Add this to the end of /boot/config.txt:

dtoverlay=pi3-disable-bt

(which disables Bluetooth)

Do this to stop Bluetooth trying to use UART:

sudo systemctl disable hciuart

2. Disable the serial console. Do this to stop the serial console on the UART:

sudo systemctl stop serial-getty@ttyAMA0.service

To stop it from starting again when rebooted:

sudo systemctl disable serial-getty@ttyAMA0.service

To re-enable it when you have finished updating firmware:

sudo systemctl enable serial-getty@ttyAMA0.service

To start it without rebooting:

sudo systemctl start serial-getty@ttyAMA0.service

You could also disable the console by editing /boot/cmdline.txt and rebooting

The UPS PiCo features an embedded serial bootloader which allows users to manually update the unit's firmware. The firmware can be uploaded using a dedicated python script, called **9600_picofuHV3.0.py**

Bootloader is small piece of firmware stored permanently in the micro controller flash memory of the UPS Pico, located in the special protected area. It can not be erased by user without dedicated hardware tools. In order to upload new firmware, an invocation of the bootloader routine is needed. It can be done manually or automatically if I2C-tools is running and installed.

The bootloader functionality ensures that the UPS PIco 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.

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 document. 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

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.

The first task which is done by the UPS PIco after reset is to check if bootloader has been requested. If not, then the rest of the firmware runs. Otherwise, the UPS PIco lights the big RED LED and waits for the firmware upload from the Raspbeerry Pi.

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

Automatic Initiation

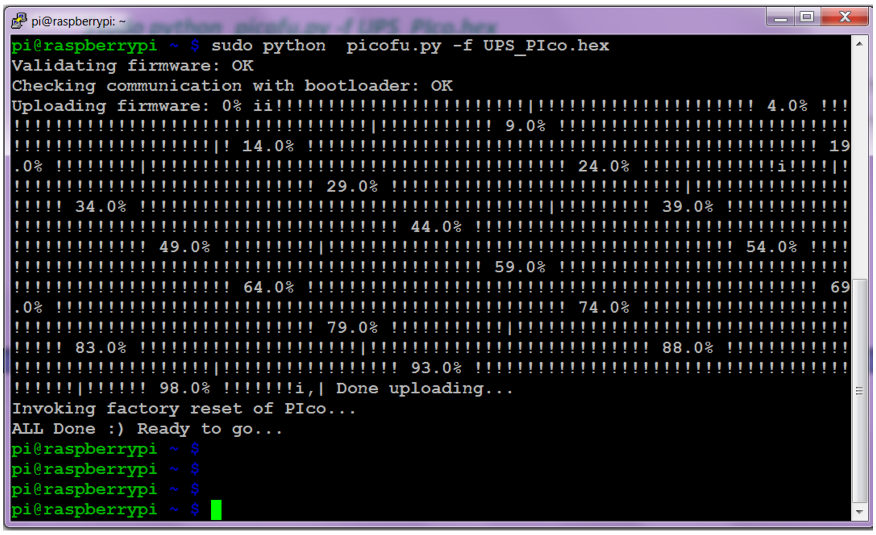
The bootloader is invoked by running the following command line

sudo i2cset -y 1 0x6b 0x00 0xff

sudo python 9600_picofuHV3.0.py -v -f UPS_PIco_HV3.0_main.hex

The **UPS_PIco_HV3.0_main.hex** 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 lit, and then when firmware starts uploading the User Blue Led will Lit and UPS LED will be blinking.



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

We would recommend that you now shutdown your Pi and UPS PiCo completely in order 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 Initiation

The UPS PiCo 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.

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 User Orange LED will light, 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 UPS_PIco_HV3.0_main.hex

If within 16 second after bootloadd initiation the firmware will be not initiated the UPS PIco HV3.0 will be reset to normal working conditions by internal Watch Dog mechanism. This has been implemented for security reasons, if firmware is uploaded remotely.

Ultra Short description of the UPS PIco HV3.0 Programmers Registers

The PICo (I^2C) 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** peripherals is made extremely simple. Control at programming language level is also possible and easy. The core concept of the **PICo interface** is that all peripheral device control and data exchange between it and Raspberry Pi® variables are common for the **I^2C interface** as also for the peripheral itself. Therefore any change of them by either party, Raspberry Pi® and the peripheral, causes immediate update and action.

Two types of variables are available:

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

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 0x01 - RPI_MODE (means cable powering mode) 0x02 - BAT_MODE
0x08	batlevel	Word	Mirror	Read	Means value of Battery Voltage in 10^{th} of mV in BCD format
0x0a	rpilevel	Word	Mirror	Read	Means value of Voltage supplying RPi on J8 5V Pin in 10^{th} of mV in BCD format
0x0c	eprlevel	Word	Mirror	Read	Means value of Extended Voltage supplying RPi on Extended Voltage input (7-28VDC) in 10^{th} of mV in BCD format
0x14	a5v0level	Word	Mirror	Read	Means value of the first A/D converter pre scaled to 5.2V. Higher voltage could not be supplied without a resistor divider. Readings are in 10^{th} of mV in BCD format

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.
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
0x25	bv				Bootloader Version - current available versions: S - BL_PIco HV 3.0A Stack/TopEnd default LP Battery F - BL_PIco HV 3.0A Stack/TopEnd default LF Battery P - BL_PIco HV 3.0A Plus default LP Battery Q - BL_PIco HV 3.0A Plus default LF Battery
0x26	fv				Firmware Version: current 0x0E dated 01/11/2016

In order to access the **0x69** variables the following commands need to be executed from the OS command line or programming language interface

sudo i2cget -y 1 0x69 0x00 b

The result will be the powering mode: 1 or 2

sudo i2cget -y 1 0x69 0x08 w

The result will be the Battery Level value in 10^{th} of mili volts

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:** 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 happen. 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.** **Factory default value is 60 seconds** **Value higher than 15 seconds are only accepted** **Each number represent 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**
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:** 0x01 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
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** 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 ON 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**
0x13	fstat	Byte	Mirror	Read	**Read:** Anytime, Return actual **if FAN** is actually

					running or not (for remote users)

Figure 1 0x6B -> UPS PIco Module Commands

Events Triggered RTC Based System Actions Scheduler

Not Unlocked in the firmware yet

0x6c -> Start Time Stamp

Address	Name	Size	Type	R/W	Explanation
0 or **0x00**	active	Byte	Common	R/W	Stamp Activation 0x00 not active 0xff active (stop)
1 or **0x01**	hour	Byte	Common	R/W	Starting Day Hour in BCD - 2 digits (0-23) i.e. 22
2 or **0x02**	minute	Byte	Common	R/W	Starting Hour Minute hour in BCD - 2 digits (0-59) i.e. 22
3 or **0x03**	mday	Byte	Common	R/W	Starting Month Day in BCD - 2 digits (1-31) i.e. 22
4 or **0x04**	month	Byte	Common	R/W	Starting Month in BCD - 2 digits (1-12) i.e. 12
5 or **0x05**	year	Byte	Common	R/W	Starting Year in BCD - 2 digits (0-99) i.e. 16

0x6d -> Running Time Stamp

Address	Name	Size	Type	R/W	Explanation
0 or **0x00**	D_repetition	Byte	Common	R/W	Days repetition in BCD 2 digits (0-99) every XX days: 00 – not repeated (only once if all repetitions are 0) 1-99 – every 1-99 days i.e. 7 means every week (i.e. every Monday) i.e. 10 means every 10 days i.e. 1 means every day
1 or **0x01**	H_repetition	Byte	Common	R/W	In BCD 2 digits (0-23) every XX days: 00 – not repeated (only once if all repetitions are 0) 1-23 – every 1-23 hours i.e. 7 means every 7 hours
2 or **0x02**	M_repetition	Byte	Common	R/W	In BCD 2 digits (1-59) every XX minutes: 00 – not repeated (only once if all repetitions are 0) 1-59 – every 1-59 minutes i.e. 7 means every 7 minutes
3 or **0x03**	H_Duration	Byte	Common	R/W	In BCD 2 digits hours 1-24 hours
4 or **0x04**	M_Duration	Byte	Common	R/W	In BCD 2 digits minutes 0-59

0x6e -> Events Stamp (NOT implemented yet)

Address	Name	Size	Type	R/W	Explanation

0x6f -> Actions Stamp (NOT implemented yet)

Currently Implemented only Power Up System – permanently selected.

Future implementation: FAN, Charger, Relay

Address	Name	Size	Type	R/W	Explanation