UPS_PIco_0x5C_25_02_2016.hex recent changes

Added Features, Bug fixes
----

1. Added mirror variable to 0x69 -> UPS PIco Module Status Registers, located at (14 or 0x0E) with name **Pico_run**

Explanation:
It is a 16 bit unsigned variable that its value is changing every 10 ms within the main firmware loop. Reading two times of this variable must return a different value (with interval longer than 10 ms). If not, means that system hangs-up, and need to be reset. However the implemented additional protection system restarted the PIco if something goes wrong. Two mechanisms have been implemented and improved since the last version:

- watch-dog (running every 2 seconds),
- and a PIco system health supervising (running every 1 second)

Reason of existence of the **Pico_run** 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.

Example (Usage):

```
sudo i2cget -y 1 0x69 0x0e w
```

or

```
sudo i2cget -y 1 0x69 0x0e w && sudo i2cget -y 1 0x69 0x0e w
```

Two readings (with interval of minimum 10 ms) must return every time when read different. Read and compared, if both reading are equal, means system hang-up, if both reading are not equal, system is running properly. However during normal operation it should never happen. This variable is only for remote users to be sure that system is working properly

Example (Usage):

```
sudo i2cget -y 1 0x69 0x0e w && sudo i2cget -y 1 0x69 0x0e w

0x15b2
0x15c1
```

Simple python script can be done by user to do this testing automatically.

2. Reviewed the internal watch-dog, so now it is more sensitive and not excuse any delays in execution. This internal watch-dog, is running every 2 seconds. The watch-dog reset, it is emergency reset protection, and when executed cut the power of the Raspberry Pi, and restart the system. However it cannot happen under any normal conditions of the UPS PIco system. The restart of the PIco after UPSR or watch-dog is called **cold start**, and cause resetting of the RTC.

3. Second level of the system health supervising monitor has been implemented. This system is monitoring now the system health every second, and if conditions not met, restarts the PIco, without resting the RTC or cutting the power. This type of restart of the PIco is called **hot start**, and can happen at any time, when for any reason systems not running properly. It is running every second. The **hot start** does not cut the power, and does not reset the RTC.

4. Improved the factory setup. In some cases a hand initiated factory setup has been required. Now, after firmware update, system always has a proper factory setup. When new firmware is uploaded, the system restarts, making LED and FAN tests, and then restarts again, setting the factory defaults, and repeating the LED and FAN tests.

5. Added information over RS232 that is printing on terminal (i.e. minicom) when charger is ON or OFF. Requested to have activated

RS232 on the PIco if RS232 is used, default RS232 is ON (activated).

6. Added variable that can be read by user (remotely) to see if charger is activated or not

Example (Usage):

```
sudo i2cget -y 1 0x69 0x10

0x01 - means Charger is ON
       If the CHG LED is not lit, means that charger is ON, but
       battery is fully charged

0x00 - means Charger is OFF
```

7. Rewritten and significantly improved the battery protection system. PIco is now not allowing deep discharge of the battery on any case. The threshold of the cut-off battery is now 3.05V - 3.15V. It is activated always if battery level is lower than it.

8. Improved the picofu.py to **picofu3.py** with properly running factory request, so always after system firmware update PIco has a default values for their internal variables

9. Improved the handling of the KEYA and KEYB, now when pressed, must be reset to be sensitive again. So, beep is audible only once if not reset by writing to an appropriate variable 0;

Example (Usage):

```
sudo i2cset -y 1 0x69 0x09 0  reset the KEYA if preset, and make it
again available for use.
```

This new implementation gives time to the software running on the Raspberry Pi to read the KEY status

10. Solved bug with none charging of some deep discharged batteries, that stays on discharged even if charging was active. Now, any deep discharged battery (deep, or non deep discharged) are recovered and charged.

11. Solved a small bug, that make buzzer non audible on some cases. Now user can use the buzzer for his customized application

Example (Usage):

sudo i2cset -y 1 0x6b 0x0e 1 is means ON

sudo i2cset -y 1 0x6b 0x0e 0 is means OFF

sudo i2cset -y 1 0x6b 0x0e 2 is means Automatic

12. Running time on Battery has been set to default value of 60 seconds