Human Interface Device Protocol Description

Relevant Devices

This application note applies to RigExpert **Shackmaster Power 600** supply.

1. Introduction

The Human Interface Device (HID) class specification allows designers to create USB-based devices and applications without the need for custom driver development.

1.1. About this Document

This document describes the information we can obtain from a device via a USB-HID connection and also provides an example of a custom software application on the host side.

1.2. Protocol Description

This document describes the necessary parameters for successful connection to the USB-HID device, and the format of information messages.

1.3. Host-side PC Application

The PC application in this example communicates with the device using the pywinusb library.

The PC application in this example does the following:

- Displays the connected Shackmaster Power device
- Sends a command to turn on/off power supply
- Reads measured values from the device and displays them

2. Shackmaster Power HID protocol

RigExpert device has following **vendor_id** = **0x0483** and **product_id** = **0xa1de**.

Host application will search attached USB devices' vendor IDs to find a particular device needed for an application.

Python code examples:	filter = hid.HidDeviceFilter(vendor_id = 0x0483, product_id = 0xa1de) devices = filter.get_devices()

The USB device stores character strings defining the product, the manufacturer, the serial number, and other descriptive texts.

Shackmaster Power devise has:

- vendor_name = "RigExpert"
- product_name = "ShackPower"
- serial_number = "5003NNNNN" (00001..99999)

All data transmitted to and from the HID device is structured in the form of reports and has its own report_id. In our case we have a constant report_id =7 and its constant length is 64 bytes.

Report requests:

1	**"POWER\n"**	**Get power supply status**

Fill output buffer as:

datatosend = { 0x07, // report_id – 1 byte, always 0x07
0x06, // Length of protocol data unit (PDU)
0x50, 0x4f, 0x57, 0x45, 0x52, 0x0a, // "POWER\n" - Get power supply status
0x00,0x00…. 0x00} //

```
Phase  Data                                                Description       Delt
a
-----  ------------------------------------------------    ---------------   ----
-
OUT    07 06 50 4f  57 45 52 0a  00 00 00 00  00 00 00 00  ..POWER.........  1.9s
c
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
IN     07 04 4f 4e  0d 0a 4f 4e  0d 0a 4f 4e  0d 0a 4f 4e  ..ON..ON..ON..ON  3.9m
s
       0d 0a 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
```

Response: - "ON\r\n", when power supply output is enabled.

```
-----  ------------------------------------------------    ---------------   ----
-
IN     07 05 4f 46  46 0d 0a 4e  0d 0a 4f 4e  0d 0a 4f 4e  ..OFF..N..ON..ON  2.0s
c
       0d 0a 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
```

Response: - "OFF\r\n", when power supply output is disabled.

2	0x0C	Get analog values

Fill output buffer as:

datatosend = { 0x07, // report_id

0x01, // Length of protocol data unit (PDU)

0x0C, // fixed, descriptor 'Get analog values'

0x00,0x00…. 0x00} //

```
Phase  Data                                                    Description        Delt
a
-----  ------------------------------------------------------  ----------------   ----
-
OUT    07 01 0c 00  00 00 00 00  00 00 00 00  00 00 00 00  .@..............  3.9m
s
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
IN     07 2f 0c 8c  00 01 30 00  10 1b 1b 00  eb 01 f4 28  .(....0........(  8.0m
s
       00 08 00 11  02 58 00 23  68 14 e2 16  00 00 00 00  .....X.#h.......
       00 00 00 06  00 01 01 02  32 00 98 32  00 00 33 00  ........2..2..3.
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
```

Response 'Get analog values':

ID offset	Data	Value	Size	Comment
0	0x07			Report ID
1	0x2f	47 bytes		Length
2	0x0c			fixed, descriptor 'Get analog values'
3	0x8c	14.0 V	uint8_t	channel '12V', voltage(x10) , XXX treat as XX.X
4	0x0001	0.1 A	int16_t	channel '12V', amperes(x10)
6	0x30	4.8 V	uint8_t	channel ' USB_1', voltage(x10)
7	0x0010	0.16 A	int16_t	channel ' USB_1', amperes(x100)
9	0x1b	27 *C	int8_t	temperature 'inside', celsium
10	0x1b	27 *C	int8_t	temperature 'outside', celsium
11	0x00eb	235 V	int16_t	mains voltage RMS, volts
13	0x01f4	50.0	int16_t	mains frequency Hz(x10)
15	0x28	40%	uint8_t	fan duty (0...100)%
16	0x0008	8 W	int16_t	total power, Watts
18	0x0011	1.7 A	int16_t	total current, amperes(x10)
20	0x0258	600 W	int16_t	max permitted power, Watts
22	0x0023	3.5 A	int16_t	max permitted amperes(x10)

ID offset	Data	Value	Size	Comment
24	0x6814e216	1746199062	uint32_t	unic3x timestamp in seconds, current date&time (2025-05-02 15:17:42 GMT+0000)
28	0x0000		uint16_t	Error 1 bitmap
30	0x0000		uint16_t	Error 2 bitmap
32	0x00000006	6	uint32_t	device serial number
36	0x00	0		accelerometer dedicated position
37	0x01	1		version: Major
38	0x01	1		version: Minor
39	0x02	2		version: Build
40	0x32	5.0 V	uint8_t	channel ' USB_2', voltage(x10)
41	0x0098	1.52 A	int16_t	channel ' USB_2', amperes(x100)
43	0x32	5.0 V	uint8_t	channel ' USB_3', voltage(x10)
44	0x0000	0.00 A	int16_t	channel ' USB_3', amperes(x100)
46	0x33	5.1 V	uint8_t	channel ' USB_4', voltage(x10)
47	0x0000	0.00 A	int16_t	channel ' USB_4', amperes(x100)
49..63				reserve

3	**"PSW1\r\n"**	**Turn on power supply**

Fill output buffer as:

datatosend = { 0x07, // report_id – 1 byte, always 0x07

0x06, // Length of protocol data unit (PDU)

0x50, 0x53, 0x57, 0x31, 0x0d, 0x0a, // "PSW1\r\n" - power on

0x00,0x00…. 0x00} //

```
Phase  Data                                                 Description       Delta
-----  ---------------------------------------------------  ----------------  -----
OUT    07 06 50 53  57 31 0d 0a  00 00 00 00  00 00 00 00  ..PSW1..........  924ms
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
```

4	**"PSW0\r\n"**	**Turn off power supply**

Fill output buffer as:

datatosend = { 0x07, // report_id – 1 byte, always 0x07

0x06, // Length of protocol data unit (PDU)

0x50, 0x53, 0x57, 0x30, 0x0d, 0x0a, // "PSW0\r\n" - power off

0x00,0x00…. 0x00} //

```
Phase  Data                                                 Description       Delta
-----  ---------------------------------------------------  ----------------  -----
OUT    07 06 50 53  57 30 0d 0a  00 00 00 00  00 00 00 00  ..PSW0..........  924ms
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
       00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
```

3. Application Examples

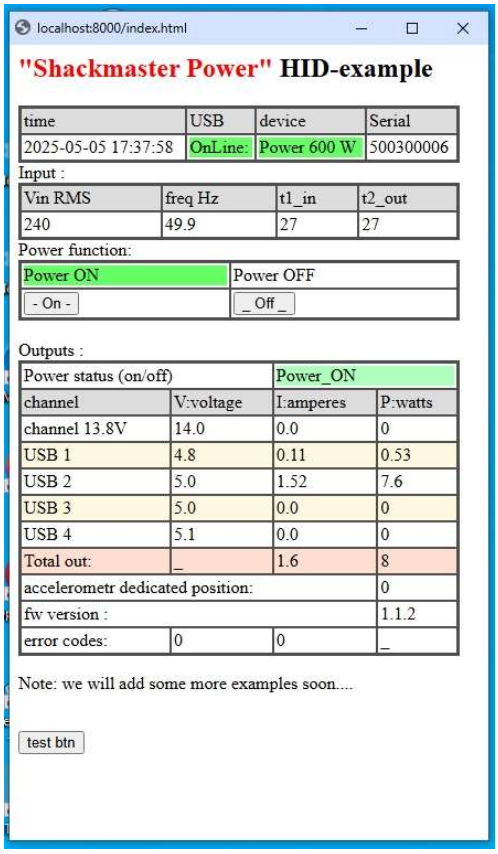
As an example, let's consider a simple desktop application in HTML, JS and Python. We will use EEL as a library for connecting all components.

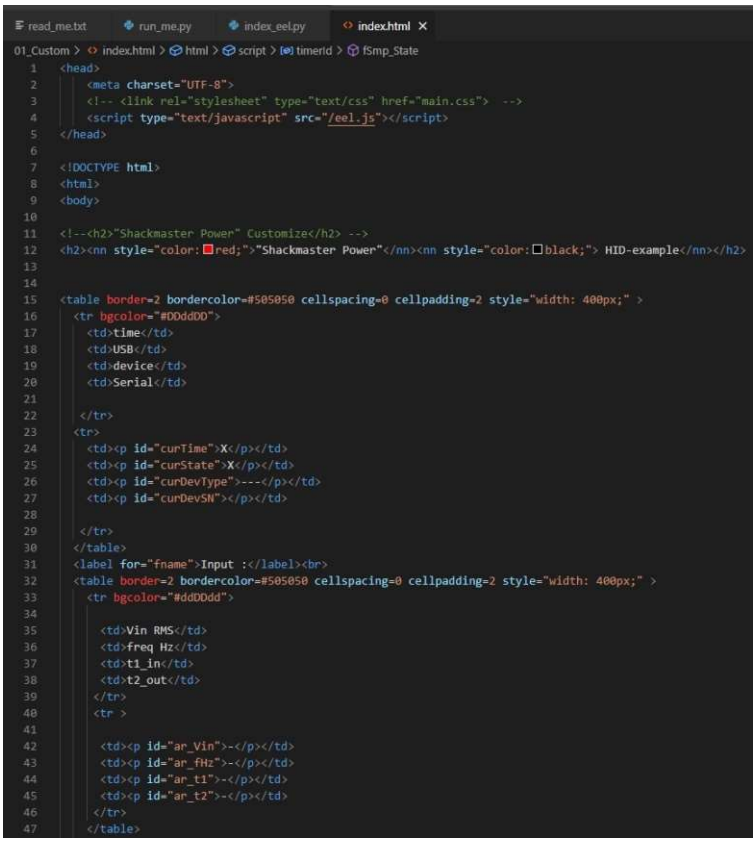
The whole project consists of 3 files:

1) index.html - the file contains everything related to the graphic part of our application
2) index_eel.py - this file serves as a link between JS and Python
3) run_me.py - we use this file to run the code. In it, we import the eel library for communication with the front-end, and pywinusb for communication with the USB-HID devices

First, install the libraries by executing a simple command:
pip install eel
pip install pywinusb

and then run file: run_me.py





For this example, we use a simple page made purely with HTML. You can easily modify it as you like, and also use it as an example of communication with USB-HID devices