USB HID Devices
Revision 1.0

GHI Electronics, LLC

Updated – June 22, 2006

This USB class includes vast range of HID devices. USBwiz HID driver support those that has only output interrupt Endpoint for HID Report sending.

HID Report is the data that is retuned from the HID, HID Generate this Report and send it to USB host – USBwiz - whenever it has new change like for example when stroking button on USB Keyboard or moving USB mouse. And USBwiz user then can get this Report by RH command.

HID Report Data is arranged in a standard way but it defers from device to other. For simplicity, we added some example of accessing common HID which are Keyboards, Mice and Joystick and how to parse HID Report Data.

To access HID:

First, this HID must be enumerated like any other USB device. We will initialize HID which is Attached to USB port 1, to USB device handle 0 as an example

UI 1>0

Second, HID Driver must be initialized to take care of this HID using the registering command and USB pipe must be chosen to access the Output Endpoint.

UH 0>3

Note: the previous initialization process is required to perform only once after connecting HID

Then USBwiz will output Report Data size that is send by the HID which is 4 Bytes for Mice and 8 Bytes for Keyboards. Now the USBwiz is ready get Data from HID which can be performed by Read HID Pipe. Data will be not by translated into ASCII HEX so the data will appear as strange characters if using Hyper terminal – which used to output incoming data on serial port as characters - .

RH 3

If the HID has no report to send then USBwiz will return error code 0xB5 which is practically not an error.

USB Keyboard Report Structure:

Parsing Standard USB Keyboard Report data:
Report size: 8 Bytes

Byte1: Modifier Byte or Reserved Constant.
Byte2 –Byte7: Key arrays bytes Table 1-2

Modifier Keys Byte:
Every Button is represented in one bit 0=Button up 1=Button down

Modifier Key	**Bit Order**
Left CTRL	0
Left SHIFT	1
Left ALT	2
Left GUI	3
Right CTRL	4
Right SHIFT	5
Right ALT	6
Right GUI	7

The following example shows the reports generated by a user typing ALT+CTRL+DEL, using a bitmap for the modifiers and a single array for all other keys taken from HID Specification:

Buttons Press Sequence	**Modifier Byte**	**Array Byte**
Left ALT down	00000100b	00h
Right CTRL down	00010100b	00h
DEL down	00010100b	4Ch
DEL up	00010100b	00h
Right CTRL up	00000100b	00h
Left ALT up	00000000b	00h

Key Array Bytes can be more or less than 6 bytes. And each byte represents a pressed key. So a 6-byte Array accepts up to 6 pressed buttons at the same time. But if the pressed keys exceeded 6, the key board will report a phantom state index code “Error Rollover Usage ID =0x01” instead of pressed buttons Usage ID codes.

The following example taken from HID specification that shows important cases for 4-Byte array keyboard:

Key Event	**Modifier Byte**	**Array**	**Array**	**Array**	**Comment**
None	00000000B	00H	00H	00H	
RALT down	01000000	00	00	00	
None	01000000	00	00	00	Report current key state even when no

					new key events.
A down	01000000	04	00	00	
X down	01000000	04	1B	00	
B down	01000000	04	05	1B	Report order is arbitrary and does not reflect order of events.
Q down	01000000	01	01	01	Phantom state. Four Array keys pressed. Modifiers still reported.
A up	01000000	05	14	1B	
B and Q up	01000000	1B	00	00	Multiple events in one report. Event order is indeterminate.
None	01000000	1B	00	00	
RALT up	00000000	1B	00	00	
X up	00000000	00	00	00	

The following table shows Usage ID Codes of Standard Keyboards:

Usage ID (Hex)	Usage Name	Remarks
00	Reserved (no event indicated)	Status indicator, Not a physical Button
01	Keyboard ErrorRollOver	Status indicator, Not a physical Button
02	Keyboard POSTFail	Status indicator, Not a physical Button
03	Keyboard ErrorUndefined	Status indicator, Not a physical Button
04	Keyboard a and A	remapped for other languages
05	Keyboard b and B	
06	Keyboard c and C	remapped for other languages
07	Keyboard d and D	
08	Keyboard e and E	
09	Keyboard f and F	
0A	Keyboard g and G	
0B	Keyboard h and H	
0C	Keyboard i and I	
0D	Keyboard j and J	
0E	Keyboard k and K	
0F	Keyboard l and L	
10	Keyboard m and M	remapped for other languages
11	Keyboard n and N	
12	Keyboard o and O	remapped for other languages
13	Keyboard p and P	remapped for other languages
14	Keyboard q and Q	remapped for other languages

15	Keyboard r and R	
16	Keyboard s and S	remapped for other languages
17	Keyboard t and T	
18	Keyboard u and U	
19	Keyboard v and V	
1A	Keyboard w and W	remapped for other languages
1B	Keyboard x and X	remapped for other languages
1C	Keyboard y and Y	remapped for other languages
1D	Keyboard z and Z	remapped for other languages
1E	Keyboard 1 and !	remapped for other languages
1F	Keyboard 2 and @	remapped for other languages
20	Keyboard 3 and #	remapped for other languages
21	Keyboard 4 and $	remapped for other languages
22	Keyboard 5 and %	remapped for other languages
23	Keyboard 6 and ^	remapped for other languages
24	Keyboard 7 and &	remapped for other languages
25	Keyboard 8 and *	remapped for other languages
26	Keyboard 9 and (	remapped for other languages
27	Keyboard 0 and)	remapped for other languages
28	Keyboard Return (ENTER)	Keyboard Enter and Keypad Enter generate different Usage codes
29	Keyboard ESCAPE	
2A	Keyboard DELETE (Backspace)	
2B	Keyboard Tab	
2C	Keyboard Spacebar	
2D	Keyboard - and (underscore)	remapped for other languages
2E	Keyboard = and +	remapped for other languages
2F	Keyboard [and {	remapped for other languages
30	Keyboard] and }	remapped for other languages
31	Keyboard \ and |	
32	Keyboard Non-US # and ~	
33	Keyboard ; and :	remapped for other languages
34	Keyboard ‘ and “	remapped for other languages
35	Keyboard Grave Accent and Tilde	remapped for other languages
36	Keyboard, and <	remapped for other languages
37	Keyboard . and >	remapped for other languages
38	Keyboard / and ?	remapped for other languages
39	Keyboard Caps Lock	
3A	Keyboard F1	
3B	Keyboard F2	
3C	Keyboard F3	
3D	Keyboard F4	
3E	Keyboard F5	
3F	Keyboard F6	
40	Keyboard F7	
41	Keyboard F8	
42	Keyboard F9	
43	Keyboard F10	
44	Keyboard F11	
45	Keyboard F12	
46	Keyboard PrintScreen	

47	Keyboard Scroll Lock	
48	Keyboard Pause	
49	Keyboard Insert	
4A	Keyboard Home	
4B	Keyboard PageUp	
4C	Keyboard Delete Forward	
4D	Keyboard End	
4E	Keyboard PageDown	
4F	Keyboard RightArrow	
50	Keyboard LeftArrow	
51	Keyboard DownArrow	
52	Keyboard UpArrow	
53	Keypad Num Lock and Clear	
54	Keypad /	
55	Keypad *	
56	Keypad -	
57	Keypad +	
58	Keypad ENTER	Keyboard Enter and Keypad Enter generate different Usage codes
59	Keypad 1 and End	
5A	Keypad 2 and Down Arrow	
5B	Keypad 3 and PageDn	
5C	Keypad 4 and Left Arrow	
5D	Keypad 5	
5E	Keypad 6 and Right Arrow	
5F	Keypad 7 and Home	
60	Keypad 8 and Up Arrow	
61	Keypad 9 and PageUp	
62	Keypad 0 and Insert	
63	Keypad . and Delete	
64	Keyboard Non-US \ and |	
65	Keyboard Application	
66	Keyboard Power	
67	Keypad =	
68	Keyboard F13	
69	Keyboard F14	
6A	Keyboard F15	
6B	Keyboard F16	
6C	Keyboard F17	
6D	Keyboard F18	
6E	Keyboard F19	
6F	Keyboard F20	
70	Keyboard F21	
71	Keyboard F22	
72	Keyboard F23	
73	Keyboard F24	
74	Keyboard Execute	
75	Keyboard Help	
76	Keyboard Menu	
77	Keyboard Select	
78	Keyboard Stop	

79	Keyboard Again
7A	Keyboard Undo
7B	Keyboard Cut
7C	Keyboard Copy
7D	Keyboard Paste
7E	Keyboard Find
7F	Keyboard Mute
80	Keyboard Volume Up
81	Keyboard Volume Down
82	Keyboard Locking Caps Lock
83	Keyboard Locking Num Lock
84	Keyboard Locking Scroll Lock
85	Keypad Comma
86	Keypad Equal Sign
8A	Keyboard International4
8B	Keyboard International5
8C	Keyboard International6
8D	Keyboard International7
8E	Keyboard International8
8F	Keyboard International9
90	Keyboard LANG1
91	Keyboard LANG2
92	Keyboard LANG3
93	Keyboard LANG4
94	Keyboard LANG5
95	Keyboard LANG6
96	Keyboard LANG7
97	Keyboard LANG8
98	Keyboard LANG9
99	Keyboard Alternate Erase
9A	Keyboard SysReq/Attention
9B	Keyboard Cancel
9C	Keyboard Clear
9D	Keyboard Prior
9E	Keyboard Return
9F	Keyboard Separator
A0	Keyboard Out
A1	Keyboard Oper
A2	Keyboard Clear/Again
A3	Keyboard CrSel/Props
A4	Keyboard ExSel
A5-CF	Reserved
B0	Keypad 00
B1	Keypad 000
B2	Thousands Separator
B3	Decimal Separator
B4	Currency Unit
B5	Currency Sub-unit
B6	Keypad (
B7	Keypad)

B8	Keypad {	
B9	Keypad }	
BA	Keypad Tab	
BB	Keypad Backspace	
BC	Keypad A	
BD	Keypad B	
BE	Keypad C	
BF	Keypad D	
C0	Keypad E	
C1	Keypad F	
C2	Keypad XOR	
C3	Keypad ^	
C4	Keypad %	
C5	Keypad <	
C6	Keypad >	
C7	Keypad &	
C8	Keypad &&	
C9	Keypad |	
CA	Keypad ||	
CB	Keypad :	
CC	Keypad #	
CD	Keypad Space	
CE	Keypad @	
CF	Keypad !	
D0	Keypad Memory Store	
D1	Keypad Memory Recall	
D2	Keypad Memory Clear	
D3	Keypad Memory Add	
D4	Keypad Memory Subtract	
D5	Keypad Memory Multiply	
D6	Keypad Memory Divide	
D7	Keypad +/-	
D8	Keypad Clear	
D9	Keypad Clear Entry	
DA	Keypad Binary	
DB	Keypad Octal	
DC	Keypad Decimal	
DD	Keypad Hexadecimal	
DE-DF	Reserved	
E0	Keyboard LeftControl	Used if modifier byte is not supported
E1	Keyboard LeftShift	Used if modifier byte is not supported
E2	Keyboard LeftAlt	Used if modifier byte is not supported
E3	Keyboard Left GUI	Used if modifier byte is not supported
E4	Keyboard RightControl	Used if modifier byte is not supported
E5	Keyboard RightShift	Used if modifier byte is not supported
E6	Keyboard RightAlt	Used if modifier byte is not supported
E7	Keyboard Right GUI	Used if modifier byte is not supported
E8-FFFF	Reserved	

USB Standard Mouse Report Structure:

Parsing Standard USB Mouse Report data:
Report size: 4 Bytes

Byte0		Byte1	Byte2	Byte3
5bits	3bits			
Reserved	Buttons b0 left b1 right b2 middle	X position	Y position	Scroll Position
Constant	Variable	Variable	Variable	Variable
NULL	Absolute	Relative to the last position	Relative to the last position	Relative to the last position

0	Up=0
	Down=1
		-127 +127	-127 +127	-127 +127

Example 1:

Accessing USB Keyboard:

After starting USBwiz and running the firmware from boot loader by R command. GHI Electronics Header will appear followed by Firmware version, then commands can be used to access USB keyboard as following: commands are in blue and they are always followed by Carriage return to be executed. USBwiz output is in Red.

```
GHI Electronics, LLC

----------------------

   USBwiz (TM) 2.08

!00

UI 0>0                 Enumerate USB Device on Port 0 to USB device handle 0

!00

UH 0>1               Register device of handle 0 as an HID and use pipe number 1 to get HID report data

!00

$08            USBwiz states that HID report size is 8 bytes which is the standard size for USB keyboards

!00

RH 1

!00
```

8 bytes will be sent if available – i.e. someone stroke a key or more – user can store this data in some array and parse it according the Keyboard Report Structure

```
!00
```

For example if the 8 bytes were:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x03	0x04	0x06	0x00	0x00	0x00	0x00	0x00

According to USB Keyboard Report Structure stated previously in this tutorial, Left SHIF and Left CTRL are pressed and button A and button C are down.
Byte 0 is 0x03 = 0b00000011 so the first two bits are 1s, the first one means Left CTRL is pressed and the second one means that Left SHIF is pressed according to Modifiers Keys Bytes table.
Byte 1 is 0x04 means button A is down
Byte 2 is 0x06 means button C is down
It more that 7 buttons apart from modifiers buttons, Report data will be all 0x01 from Byte 1 to Byte 7 stating an error.

Example 2:

Accessing USB Mouse:

After starting USBwiz and running the firmware from boot loader by R command. GHI Electronics Header will appear followed by Firmware version, then commands can be used to access USB mouse as following: commands are in blue and they are always followed by Carriage return to be executed. USBwiz output is in Red.

```
GHI Electronics, LLC

----------------------

    USBwiz (TM) 2.08

!00

UI 0>0                  Enumerate USB Device on Port 0 to USB device handle 0

!00

UH 0>1              Register device of handle 0 as an HID and use pipe number 1 to get HID report data

!00

$04             USBwiz states that HID report size is 4 bytes which is the standard size for USB keyboards

!00

RH 1

!00
```

4 bytes will be sent if available – i.e. someone stroke a key or more – user can store this data in some array and parse it according the Mouse Report Structure

```
!00
```

For example if the 4 bytes were:

Byte 0	Byte 1	Byte 2	Byte 3
0x01	0x04	0xFD	0x00

According to USB Mouse Report Structure stated previously in this tutorial, Left mouse button is pressed, and the mouse is moved 4 dots to the left and 3 dots down relatively to the old position and scroll wheels are not changed.
Byte 0 is 0x01 = 0b00000001 means the first one means Left button is pressed.
Byte 1 is 0x04 means movement 4 dots to right
Byte 2 is 0xFD means movement 3 dots down

References:

- USB Device Class Definition for Human Interface Devices www.usb.org
- USB HID Usage Table www.usb.org

There is no guarantee on the data in this document. Always consult www.usb.org