



## USB HID报告描述符教程



李大虾

望天一眼，云烟消散如云烟。

69 人赞同了该文章

原文地址：

[Tutorial about USB HID Report Descriptors](#)

USB HID报告描述符是USB主机请求于USB设备的一种描述符。HID设备用报告的形式发送数据到主机，描述符告诉主机如何解释数据。下面将展示如何写一个描述符。

首先，到[USB.org - HID Tools](#)页面找到“Device Class Definition for HID”文档，下面叙述的内容本质上是该文档中的重要部分。

其次，在上述页面获得HID描述符工具，然后读完本教程之后想着如何使用它。手动写HID报告描述符是一件令人头痛不已的事情，本工具可以替代你转换二进制和十六进制，并查找数字代表的意义。

### 什么是USB HID报告描述符？

HID协议使得设备的实现变得简单，设备会定义数据包为HID描述符发送给主机。HID描述符是描述设备数据包的固定代码字节数组，包括设备支持多少个包，包有多大，以及包中每个字节和比特的含义。比如，带有计算程序按键的键盘告诉主机按键是按下还是松开状态，该信息放在数据包4的第6个字节的第2个比特，注意这个位置是设备指定说明的。设备通常将HID描述符存放在ROM里，不必深入理解或分析HID描述符。今天市场上的一些鼠标和键盘硬件实现仅仅使用一个8比特的CPU。--以上来自维基百科

**简单一点开始，做一个标准的鼠标，三个按键，在X轴和Y轴上移动。**因此要发送关于按键和移动的数据。每个按键用1个比特表示，每个字节表示移动一个轴上的有符号整型值，数据结构表示如下：



	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Useless	Useless	Useless	Useless	Useless	Left Button	Middle Button	Right Button
Byte 1	X Axis Relative Movement as Signed Integer							
Byte 2	Y Axis Relative Movement as Signed Integer							

C语言格式如下：

```
1 struct mouse_report_t
2 {
3     uint8_t buttons;
4     int8_t x;
5     int8_t y;
6 }
```

因此在描述符中，第一个项目必须描述按键，包含三个字段：

```
1 USAGE_PAGE (Button)
2 USAGE_MINIMUM (Button 1)
3 USAGE_MAXIMUM (Button 3)
```

每个按键的状态用1个比特表示，0或1：

```
1 LOGICAL_MINIMUM (0)
2 LOGICAL_MAXIMUM (1)
```

有3个比特来表示：

```
1 REPORT_COUNT (3)
2 REPORT_SIZE (1)
```

发送这些变量数据到电脑：

```
1 INPUT (Data,Var,Abs)
```

最终表述按键的表述如下：

```
1 USAGE_PAGE (Button)
2 USAGE_MINIMUM (Button 1)
3 USAGE_MAXIMUM (Button 3)
4 LOGICAL_MINIMUM (0)
5 LOGICAL_MAXIMUM (1)
6 REPORT_COUNT (3)
7 REPORT_SIZE (1)
8 INPUT (Data,Var,Abs)
```

5个没有用的填充比特：

```
1 REPORT_COUNT (1)
2 REPORT_SIZE (5)
3 INPUT (Cnst,Var,Abs)
```

X轴的移动：

```
1 USAGE_PAGE (Generic Desktop)
2 USAGE (X)
```

用1个字节的有符号整型，范围设成-127~127（实际上是-127~128，这样做是为了对称），来表示移动距离：



```
1 LOGICAL_MINIMUM (-127)
2 LOGICAL_MAXIMUM (127)
```

使用整个字节来发送：

```
1 REPORT_SIZE (8)
2 REPORT_COUNT (1)
```

当作坐标变量发送给电脑：

```
1 INPUT (Data,Var,Rel)
```

因此最终X轴移动的表述如下：

```
1 USAGE_PAGE (Generic Desktop)
2 USAGE (X)
3 LOGICAL_MINIMUM (-127)
4 LOGICAL_MAXIMUM (127)
5 REPORT_SIZE (8)
6 REPORT_COUNT (1)
7 INPUT (Data,Var,Rel)
```

那么Y轴移动的呢？可以写成这样：

```
1 USAGE_PAGE (Generic Desktop)
2 USAGE (X)
3 LOGICAL_MINIMUM (-127)
4 LOGICAL_MAXIMUM (127)
5 REPORT_SIZE (8)
6 REPORT_COUNT (1)
7 INPUT (Data,Var,Rel)
8 USAGE_PAGE (Generic Desktop)
9 USAGE (Y)
10 LOGICAL_MINIMUM (-127)
11 LOGICAL_MAXIMUM (127)
12 REPORT_SIZE (8)
13 REPORT_COUNT (1)
14 INPUT (Data,Var,Rel)
```

上述表述虽然没有问题，为了节省内存，可以表述如下：

```
1 USAGE_PAGE (Generic Desktop)
2 USAGE (X)
3 USAGE (Y)
4 LOGICAL_MINIMUM (-127)
5 LOGICAL_MAXIMUM (127)
6 REPORT_SIZE (8)
7 REPORT_COUNT (2)
8 INPUT (Data,Var,Rel)
```

综上，整个的表述是：

```

1  USAGE_PAGE (Button)
2  USAGE_MINIMUM (Button 1)
3  USAGE_MAXIMUM (Button 3)
4  LOGICAL_MINIMUM (0)
5  LOGICAL_MAXIMUM (1)
6  REPORT_COUNT (3)
7  REPORT_SIZE (1)
8  INPUT (Data,Var,Abs)
9  REPORT_COUNT (1)
10 REPORT_SIZE (5)
11 INPUT (Cnst,Var,Abs)
12 USAGE_PAGE (Generic Desktop)
13 USAGE (X)
14 USAGE (Y)
15 LOGICAL_MINIMUM (-127)
16 LOGICAL_MAXIMUM (127)
17 REPORT_SIZE (8)
18 REPORT_COUNT (2)
19 INPUT (Data,Var,Rel)

```

但是，这并未结束，为了让电脑知道这个是鼠标设备，必须这样：

```

1  USAGE_PAGE (Generic Desktop)
2  USAGE (Mouse)
3  COLLECTION (Application)
4      USAGE (Pointer)
5      COLLECTION (Physical)
6
7      ... What we wrote already goes here
8
9      END_COLLECTION
10 END_COLLECTION

```

因此在最后，下面是标准的鼠标USB HID报告描述符：

```

1  0x05, 0x01, // USAGE_PAGE (Generic Desktop)
2  0x09, 0x02, // USAGE (Mouse)
3  0xa1, 0x01, // COLLECTION (Application)
4  0x09, 0x01, // USAGE (Pointer)
5  0xa1, 0x00, // COLLECTION (Physical)
6  0x05, 0x09, // USAGE_PAGE (Button)
7  0x19, 0x01, // USAGE_MINIMUM (Button 1)
8  0x29, 0x03, // USAGE_MAXIMUM (Button 3)
9  0x15, 0x00, // LOGICAL_MINIMUM (0)
10 0x25, 0x01, // LOGICAL_MAXIMUM (1)
11 0x95, 0x03, // REPORT_COUNT (3)
12 0x75, 0x01, // REPORT_SIZE (1)
13 0x81, 0x02, // INPUT (Data,Var,Abs)
14 0x95, 0x01, // REPORT_COUNT (1)
15 0x75, 0x05, // REPORT_SIZE (5)
16 0x81, 0x03, // INPUT (Cnst,Var,Abs)
17 0x05, 0x01, // USAGE_PAGE (Generic Desktop)
18 0x09, 0x30, // USAGE (X)
19 0x09, 0x31, // USAGE (Y)
20 0x15, 0x81, // LOGICAL_MINIMUM (-127)
21 0x25, 0x7f, // LOGICAL_MAXIMUM (127)
22 0x75, 0x08, // REPORT_SIZE (8)
23 0x95, 0x02, // REPORT_COUNT (2)
24 0x81, 0x06, // INPUT (Data,Var,Rel)
25 0xc0, // END_COLLECTION
26 0xc0 // END_COLLECTION

```

这实际上是USB HID文档上的例子，同样也是HID工具提供的例子。

现在有了一些概念，继续研究：

**Usage Pages**（用例页）：文档中好像解释的不够好，概念包含有USAGE、USAGE\_PAGE、USAGE\_MINIMUM、USAGE\_MAXIMUM。在描述符中，首先设置一个USAGE\_PAGE，某些可用的USAGES。在鼠标的例子中，在USAGE\_PAGE (Generic Desktop)允许你使用USAGE (Mouse)，当用例页改为USAGE\_PAGE (Button)，允许在USAGE(X) 和USAGE(Y)之前给按键指定USAGE\_MINIMUM和USAGE\_MAXIMUM，再然后用用例页变回到USAGE\_PAGE (Generic Desktop)。用例页就像是命名空间一般，改变用例页作用于“usages”是否可用。请阅读文档“HID Usage Tables”获得更多细节。



**Collections**（集合）：通过阅读文档关于集合的官方使用方法，用自己的话讲，集合用来组织数据，比如键盘可能內建触摸板，因此键盘数据需要保存在一个应用集合里，触摸板数据则保存在另一个应用集合里。可以给每个集合指定“Report ID”，具体将在后面叙述。

可以将USAGE (Mouse)变为USAGE (Gamepad)，让电脑知道这是一个带1个操作杆和3个按键的游戏手柄。将PS2控制器变为USB游戏手柄怎么样？控制器有16个按键和两个拇指棒，数据结构如下：

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Button	Button	Button	Button	Button	Button	Button	Button
Byte 1	Button	Button	Button	Button	Button	Button	Button	Button
Byte 2	Left X Axis as Signed Integer							
Byte 3	Left Y Axis as Signed Integer							
Byte 4	Right X Axis as Signed Integer							
Byte 5	Right Y Axis as Signed Integer							

C语言的数据结构如下：

```
1 struct gamepad_report_t
2 {
3     uint16_t buttons;
4     int8_t left_x;
5     int8_t left_y;
6     int8_t right_x;
7     int8_t right_y;
8 }
```

让电脑知道这是一个游戏手柄：

```
1 USAGE_PAGE (Generic Desktop)
2 USAGE (Game Pad)
3 COLLECTION (Application)
4     COLLECTION (Physical)
5
6     ...
7
8     END COLLECTION
9 END COLLECTION
```

按键的描述如下：

```
1 USAGE_PAGE (Button)
2 USAGE_MINIMUM (Button 1)
3 USAGE_MAXIMUM (Button 16)
4 LOGICAL_MINIMUM (0)
5 LOGICAL_MAXIMUM (1)
6 REPORT_COUNT (16)
7 REPORT_SIZE (1)
8 INPUT (Data,Var,Abs)
```

拇指棒的四个轴：

```

1  USAGE_PAGE (Generic Desktop)
2  USAGE (X)
3  USAGE (Y)
4  USAGE (Z)
5  USAGE (Rx)
6  LOGICAL_MINIMUM (-127)
7  LOGICAL_MAXIMUM (127)
8  REPORT_SIZE (8)
9  REPORT_COUNT (4)
10 INPUT (Data,Var,Abs)

```

注：Z表示右边拇指棒的X轴，Rx表示右边拇指棒的Y轴。这看起来不合理，但是是目前大多数游戏手柄的工作方式。在《战地：叛逆连队2》上测试是可以工作的。

注：用“absolute”表示类似操作杆的东西，“relative”表示类似鼠标的东西。

结果如下：

```

1  USAGE_PAGE (Generic Desktop)
2  USAGE (Game Pad)
3  COLLECTION (Application)
4      COLLECTION (Physical)
5          USAGE_PAGE (Button)
6          USAGE_MINIMUM (Button 1)
7          USAGE_MAXIMUM (Button 16)
8          LOGICAL_MINIMUM (0)
9          LOGICAL_MAXIMUM (1)
10         REPORT_COUNT (16)
11         REPORT_SIZE (1)
12         INPUT (Data,Var,Abs)
13         USAGE_PAGE (Generic Desktop)
14         USAGE (X)
15         USAGE (Y)
16         USAGE (Z)
17         USAGE (Rx)
18         LOGICAL_MINIMUM (-127)
19         LOGICAL_MAXIMUM (127)
20         REPORT_SIZE (8)
21         REPORT_COUNT (4)
22         INPUT (Data,Var,Abs)
23     END COLLECTION
24 END COLLECTION

```

两个玩家呢？这就显示出集合的方便之处：

```

1  USAGE_PAGE (Generic Desktop)
2  USAGE (Game Pad)
3  COLLECTION (Application)
4      COLLECTION (Physical)
5          REPORT_ID (1)
6          ...
7      END COLLECTION
8  END COLLECTION
9  USAGE_PAGE (Generic Desktop)
10 USAGE (Game Pad)
11 COLLECTION (Application)
12     COLLECTION (Physical)
13         REPORT_ID (2)
14         ...
15     END COLLECTION
16 END COLLECTION

```

填充上数据区域，得到：

```

1  USAGE_PAGE (Generic Desktop)
2  USAGE (Game Pad)
3  COLLECTION (Application)
4      COLLECTION (Physical)
5          REPORT_ID (1)
6          USAGE_PAGE (Button)
7          USAGE_MINIMUM (Button 1)
8          USAGE_MAXIMUM (Button 16)
9          LOGICAL_MINIMUM (0)
10         LOGICAL_MAXIMUM (1)
11         REPORT_COUNT (16)
12         REPORT_SIZE (1)
13         INPUT (Data,Var,Abs)
14         USAGE_PAGE (Generic Desktop)
15         USAGE (X)
16         USAGE (Y)
17         USAGE (Z)
18         USAGE (Rx)
19         LOGICAL_MINIMUM (-127)
20         LOGICAL_MAXIMUM (127)
21         REPORT_SIZE (8)
22         REPORT_COUNT (4)
23         INPUT (Data,Var,Abs)
24     END COLLECTION
25 END COLLECTION
26 USAGE_PAGE (Generic Desktop)
27 USAGE (Game Pad)
28 COLLECTION (Application)
29     COLLECTION (Physical)
30         REPORT_ID (2)
31         USAGE_PAGE (Button)
32         USAGE_MINIMUM (Button 1)
33         USAGE_MAXIMUM (Button 16)
34         LOGICAL_MINIMUM (0)
35         LOGICAL_MAXIMUM (1)
36         REPORT_COUNT (16)
37         REPORT_SIZE (1)
38         INPUT (Data,Var,Abs)
39         USAGE_PAGE (Generic Desktop)
40         USAGE (X)
41         USAGE (Y)
42         USAGE (Z)
43         USAGE (Rx)
44         LOGICAL_MINIMUM (-127)
45         LOGICAL_MAXIMUM (127)
46         REPORT_SIZE (8)
47         REPORT_COUNT (4)
48         INPUT (Data,Var,Abs)
49     END COLLECTION
50 END COLLECTION

```

这样看起来，在数据结构上加上Report ID真的很重要：

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Report ID							
Byte 1	Button	Button	Button	Button	Button	Button	Button	Button
Byte 2	Button	Button	Button	Button	Button	Button	Button	Button
Byte 3	Left X Axis as Signed Integer							
Byte 4	Left Y Axis as Signed Integer							
Byte 5	Right X Axis as Signed Integer							
Byte 6	Right Y Axis as Signed Integer							

C语言数据结构：

```

1  struct multiplayer_gamepad_report_t
2  {
3      uint8_t report_id;
4      uint16_t buttons;
5      int8_t left_x;
6      int8_t left_y;
7      int8_t right_x;
8      int8_t right_y;
9  }

```

在发送数据到电脑上之前必须手动的设置Report ID，目的是为了让电脑知道数据来自哪个游戏手柄。



```
1 multiplayer_gamepad_report_t player1_report;
2 multiplayer_gamepad_report_t player2_report;
3 player1_report.report_id = 1;
4 player2_report.report_id = 2;
```

也可以用Collections和Report ID设成综合性设备。因此包含键盘、鼠标和游戏手柄（2）的描述符如下：

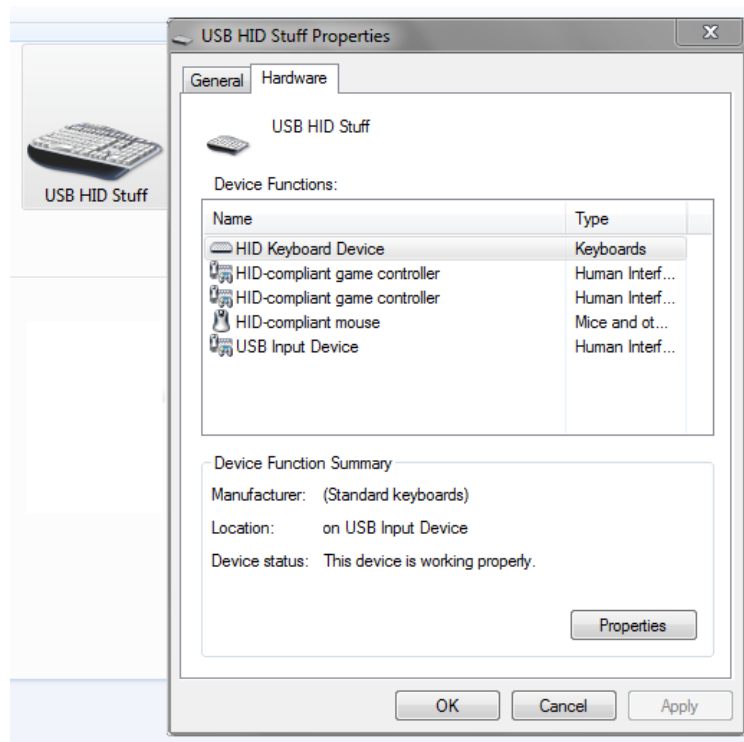
```
1 USAGE_PAGE (Generic Desktop)
2 USAGE (Keyboard)
3 COLLECTION (Application)
4     REPORT_ID (1)
5     ...
6 END_COLLECTION
7 USAGE_PAGE (Generic Desktop)
8 USAGE (Mouse)
9 COLLECTION (Application)
10     USAGE (Pointer)
11     COLLECTION (Physical)
12         REPORT_ID (2)
13         ...
14     END_COLLECTION
15 END_COLLECTION
16 USAGE_PAGE (Generic Desktop)
17 USAGE (Game Pad)
18 COLLECTION (Application)
19     COLLECTION (Physical)
20         REPORT_ID (3)
21         ...
22     END_COLLECTION
23 END_COLLECTION
24 USAGE_PAGE (Generic Desktop)
25 USAGE (Game Pad)
26 COLLECTION (Application)
27     COLLECTION (Physical)
28         REPORT_ID (4)
29         ...
30     END_COLLECTION
31 END_COLLECTION
```

当然数据结构也要增加Report ID：

```
1 struct keyboard_report_t
2 {
3     uint8_t report_id;
4     uint8_t modifier;
5     uint8_t reserved;
6     uint8_t keycode[6];
7 }
8
9 struct mouse_report_t
10 {
11     uint8_t report_id;
12     uint8_t buttons;
13     int8_t x;
14     int8_t y;
15 }
16
17 struct gamepad_report_t
18 {
19     uint8_t report_id;
20     uint16_t buttons;
21     int8_t left_x;
22     int8_t left_y;
23     int8_t right_x;
24     int8_t right_y;
25 }
```

由于改变了数据结构，设备不再支持启动协议，因此不用定义协议。相应的改变usbconfig.h。想要看到这个效果，USnooBie导入工程示例，Windows的“设备和打印”上显示：





编辑于 2017-07-27 17:35