[**闲暇就玩USB之： USB鼠标和键盘**](http://blog.csdn.net/dfysy/article/details/5914863)

标签： [descriptor](http://www.csdn.net/tag/descriptor)[report](http://www.csdn.net/tag/report)[keyboard](http://www.csdn.net/tag/keyboard)[application](http://www.csdn.net/tag/application)[input](http://www.csdn.net/tag/input)[output](http://www.csdn.net/tag/output)

2010-09-29 18:22 3368人阅读 [评论](http://blog.csdn.net/dfysy/article/details/5914863#comments)(4) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/dfysy/article/details/5914863#report)

本文章已收录于：

http://static.blog.csdn.net/images/category_icon.jpg分类：

其它嵌入式*（4）* http://static.blog.csdn.net/images/arrow_triangle%20_down.jpghttp://static.blog.csdn.net/images/arrow_triangle_up.jpg

[作者同类文章](http://blog.csdn.net/dfysy/article/category/713829)*X*

版权声明：本文为博主原创文章，未经博主允许不得转载。

其实这个问题很多人都玩过了，而且HID Spec上有标准例子，但是USB鼠标和键盘的确很有意思，而且俺还尝试了一点和别人不一样的东西，在此以记之。

HID SPEC上讲的键盘和鼠标都是支持boot的，就是可以被Bios支持的，比如在开机的时候设置Bios的时候就可以用。因此那个Report Descriptor真的是相当的复杂啊，都63个字节了，就差一个字节就超过俺的EP0的Max Pack Size。其实介绍Report Descriptor的最好网络文章是《USB/HID设备报告描述符详解》，看用词像个台湾同胞写的，可以在下列地址阅读：

<http://blog.chinaunix.net/u2/63560/showart.php?id=1900045>

其实这个似乎都还是比较复杂，我做了一个不支持boot的键盘的Report Descriptor，只支持一个字节的输入，其实一个字节也是可以输入101个键的，HID Spec里面的Descriptor其实是支持6个键同时输入的，所以用了6个字节。下面俺的简陋型HID Descriptor就是这个样子的：

char HidBoardReportDescriptor[23] = {  
    0x05, 0x01,                    // USAGE\_PAGE (Generic Desktop)  
    0x09, 0x06,                    // USAGE (Keyboard)  
    0xa1, 0x01,                    // COLLECTION (Application)  
    0x05, 0x07,                    //   USAGE\_PAGE (Keyboard)  
    0x19, 0x1E,                    //   USAGE\_MINIMUM (Keyboard ! and 1)  
    0x29, 0x25,                    //   USAGE\_MAXIMUM (Keyboard \* and 8)  
    0x15, 0x00,                    //   LOGICAL\_MINIMUM (0)  
    0x25, 0xff,                    //   LOGICAL\_MAXIMUM (255)  
    0x75, 0x08,                    //   REPORT\_SIZE (8)  
    0x95, 0x01,                    //   REPORT\_COUNT (1)  
    0x81, 0x00,                    //   INPUT (Data,Ary,Abs)  
    0xc0                           // END\_COLLECTION  
};

Descriptor中的几个术语大概是这个意思，俺的通俗理解：

Usage Page：相当于用法类别，或者功能类别，用我的地址做比较，相当于“北京市海淀区”的“北京市”

Usage：相当于具体的用法，比如“海淀区”，一个完整的用法需要Usage Page和Usage配合才能完整表达，用地址类比一下。。。好像不能用“北京市海淀区”，海淀区中国只听说过这一个！比如说“石门坎”吧，云贵川有几百个地方叫“石门坎”，因此必须说明“四川省宁南县华弹镇石门坎”才有意义，兄弟，扯得有点远了！

由于Usage Page是全局的，因此只声明一次就行了，除非下面要什么新的Usage Page。而Usage是要一个一个的声明的。但是101个键要写101次太麻烦，因此使用USAGE\_MINIMUM 和USAGE\_MAXIMUM来定义一个范围，比如我上面的蓝色的两行就把1-8八个键都描述了。

LOGICAL\_MINIMUM 和 LOGICAL\_MAXIMUM 对应的是输入数据的范围，超出这个范围不予处理。这个Lgical值的0表示没输入，1和上面的USAGE\_MINIMUM是对应，也就是输入1对应计算机的1，如果把USAGE\_MINUM和USAGE\_MAXIM改成如下：

    0x19, 0x04,                    //   USAGE\_MINIMUM (Keyboard a and A)  
    0x29, 0x0B,                    //   USAGE\_MAXIMUM (Keyboard h and H)  
那样输入1对应的就是计算机端的'A'了。

即原来的对应是1-->0x1E("1"键或"!"键),修改后为1-->0x04("a" or "A"),关于每个USAGE对应的按键，在HID标准中有描述，这样就把逻辑值和最后的键对应起来了。

REPORT\_SIZE:表面的输入的位宽度，俺的是8位， REPORT\_COUNT是这样的数有几个，俺的只有一个。后面的INPUT表示有一个输入。

COLLECTION在俺看来就是个大括号。

这样一个键盘的简单Descriptor就OK了。鼠标的就采用Boot就可以了。

HID的Report Descriptor是可以支持多个设备的，比如同时支持一个鼠标和键盘，这时就需要Report ID来参与了，这样在上传数据的时候就需要多一个字节表示Report ID来标识是哪个设备的数据，Report ID位于发送数据的第一个字节。切记，Report ID不能为0，因为系统默认已经用过了。（有人说这种复合设备在Configuration里Subclass不能为Boot，但是好像没关系的）

比如假设鼠标的Report ID是2，则发送的数据应该如下：

0x02,00,05,00, 00

后面的红色部分是原来不采用Report ID时发送的数据。下面是一个采用HID SPEC的键盘和鼠标Descriptor做的支持两个设备的Descriptor。一旦枚举成功，恭喜你，你会在设备管理器看到多出了一个鼠标和一个键盘.

const char HidBoardReportDescriptor[] = {  
// Descriptors for Keyboard  
  0x05, 0x01,                    /\* USAGE\_PAGE (Generic Desktop) \*/  
  0x09, 0x06,                    /\* USAGE (Keyboard) \*/  
  0xa1, 0x01,                    /\* COLLECTION (Application) \*/  
  0x85, 0x03,                    /\*   Report ID (3) \*/  
  0x05, 0x07,                    /\*   USAGE\_PAGE (Keyboard) \*/  
 0x19, 0xe0,                    //   USAGE\_MINIMUM (Keyboard LeftControl)  
 0x29, 0xe7,                    //   USAGE\_MAXIMUM (Keyboard Right GUI)  
 0x15, 0x00,                    //   LOGICAL\_MINIMUM (0)  
 0x25, 0x01,                    //   LOGICAL\_MAXIMUM (1)  
 0x75, 0x01,                    //   REPORT\_SIZE (1)  
 0x95, 0x08,                    //   REPORT\_COUNT (8)  
 0x81, 0x02,                    //   INPUT (Data,Var,Abs)  
 0x95, 0x01,                    //   REPORT\_COUNT (1)  
 0x75, 0x08,                    //   REPORT\_SIZE (8)  
 0x81, 0x03,                    //   INPUT (Cnst,Var,Abs)  
 0x95, 0x05,                    //   REPORT\_COUNT (5)  
 0x75, 0x01,                    //   REPORT\_SIZE (1)  
 0x05, 0x08,                    //   USAGE\_PAGE (LEDs)  
 0x19, 0x01,                    //   USAGE\_MINIMUM (Num Lock)  
 0x29, 0x05,                    //   USAGE\_MAXIMUM (Kana)  
 0x91, 0x02,                    //   OUTPUT (Data,Var,Abs)  
 0x95, 0x01,                    //   REPORT\_COUNT (1)  
 0x75, 0x03,                    //   REPORT\_SIZE (3)  
 0x91, 0x03,                    //   OUTPUT (Cnst,Var,Abs)  
 0x95, 0x06,                    //   REPORT\_COUNT (6)  
 0x75, 0x08,                    //   REPORT\_SIZE (8)  
 0x15, 0x00,                    //   LOGICAL\_MINIMUM (0)  
 0x25, 0xFF,                    //   LOGICAL\_MAXIMUM (255)  
 0x05, 0x07,                    //   USAGE\_PAGE (Keyboard)  
 0x19, 0x00,                    //   USAGE\_MINIMUM (Reserved (no event indicated))  
 0x29, 0x65,                    //   USAGE\_MAXIMUM (Keyboard Application)  
 0x81, 0x00,                    //   INPUT (Data,Ary,Abs)  
 0xc0,                           // END\_COLLECTION

// Descriptors for Mouse  
  0x05, 0x01,                    /\* Usage Page (Generic Desktop)  \*/  
  0x09, 0x02,                    /\* Usage (Mouse)                 \*/  
  0xA1, 0x01,                    /\* Collection (Application)      \*/  
  0x09, 0x01,                    /\*   Usage (Pointer)               \*/  
  0xA1, 0x00,                    /\*   Collection (Physical)         \*/  
  0x85, 0x02,                    /\*     Report ID (2) \*/  
  0x05, 0x09,                    /\*     Usage Page (Buttons)          \*/  
  0x19, 0x01,                    /\*     Usage Minimum (01)            \*/  
  0x29, 0x03,                    /\*     Usage Maximum (03)            \*/  
  0x15, 0x00,                    /\*     Logical Minimum (0)           \*/  
  0x25, 0x01,                    /\*     Logical Maximum (1)           \*/  
  0x75, 0x01,                    /\*     Report Size (1)               \*/  
  0x95, 0x03,                    /\*     Report Count (3)              \*/  
  0x81, 0x02,                    /\*     Input (Data, Variable, Absolute)\*/  
  0x75, 0x05,                    /\*     Report Size (5)                 \*/  
  0x95, 0x01,                    /\*     Report Count (1)                \*/  
  0x81, 0x01,                    /\*     Input (Constant)    ;5 bit padding \*/  
  0x05, 0x01,                    /\*     Usage Page (Generic Desktop)       \*/  
  0x09, 0x30,                    /\*     Usage (X)                          \*/  
  0x09, 0x31,                    /\*     Usage (Y)                       \*/  
  0x09, 0x38,                    /\*     Usage (Wheel)                   \*/  
  0x15, 0x81,                    /\*     Logical Minimum (-127)          \*/  
  0x25, 0x7F,                    /\*     Logical Maximum (127)           \*/  
  0x75, 0x08,                    /\*     Report Size (8)                 \*/  
  0x95, 0x03,                    /\*     Report Count (3)                \*/  
  0x81, 0x06,                    /\*     Input (Data, Variable, Relative)\*/  
  0xC0,                          /\*   End Collection                  \*/  
  0xC0                           /\* End Collection                  \*/

};