天池渔隐

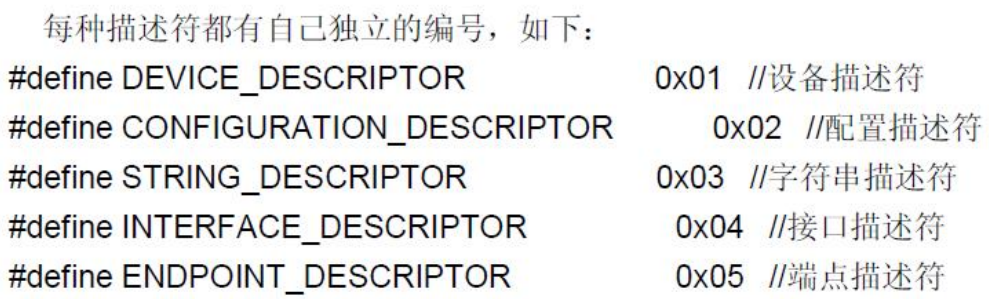
昵称：天池渔隐
园龄：7年1个月
粉丝：0
关注：5
+加关注

USB描述符详细讲解

在USB中USB HOST 是通过各种描述符来识别设备的，有 **设备描述符，接口描述符，端点描述符，字符描述符，报告描述符**等

USB HID 设备是通过报告来传送数据的，报告有：**输入报告 和 输出报告**
输入报告：是设备发送给主机的，例如 usb鼠标将鼠标移动和鼠标点击的信息返回给电脑，键盘将按键数据返回给电脑。
输出报告：是主机发送给USB设备的，例如键盘上的数字键盘锁定灯和大写字母锁定灯等。报告是一个数据包，里面包含的是所要传送的数据。
输入报告是通过中断输入端点输入的。
报告描述符：是描述一个报告以及报告里面的数据是用来干什么的。通过它，USB HOST 可以 分析出报告里面的数据所要表达的意思。

USB电气特性
标准USB使用4根线：5V电源线（VBus），差分数据线负（D-），差分数据线正（D+），地（GND）
USB线缆及插头插座
USB设备的插入检测机制
USB主机如何检测到设备插入的那？首先在USB集线器的每个下游端口的D+和D-上，分别接上一个15K的下拉电阻，而在USB设备端，在D+或者D-上接上1.5K的上拉电阻，高速设备接在D+上，低俗设备上拉接在D-上。这样当有设备插入到集线器时，就将差分数据线上的一条拉高了，集线器检测到这个状态后，它就报告给USB主控制器，这样就检测到设备插入了。
USB设备的枚举过程
usb主机检测到USB设备插入后，就要对设备进行枚举了。枚举的作用就是从设备是那个读取一些信息，知道设备是什么样的设备，如果通信，这样主机就可以根据这些信息假造合适的驱动程序。调试USB设备，很重要的一点就是USB枚举过程，只要枚举成功了，那就成功一大半了。
USB的一种传输模式---控制传输
这种传输在USB中是非常重要的，它保证数据的正确性，在设备的枚举过程中都是控制传输。
控制传输分为三个过程：1.建立过程；2可选的数据过程；3状态过程。
建立过程都是由USB主机发起，它开始于一个Setup令牌包，后面紧跟着一个DATA0包，如果是控制输入传输，那么数据过程就是输入数据；如果控制输出传输，那么数据过程就是输出数据。数据过程之后是状态过程。状态过程刚好与数据过程的数据传输放喜爱那个相反。
首先：主机检测到USB设备插入后，就会先对设备进行复位，复位后，USB主机就会对地址为0的设备发送获取设备描述符的标准请求。所有的USB设备在总线复位后其地址都为0，这样主机就可以跟那些刚刚插入的设备通过地址0通信，
获取玩**设备描述符**后，主机就会获取**配置描述符9个字节，** 主机获取到配置描述符后，根据里面的配置集合总长度，在获取配置结合。配置集合包括配置描述符，接口描述符，端点描述符等。
USB的描述以及各种描述符之间的依赖关系



1.设备描述符

```
//定义标准的设备描述符结构

typedef struct _DEVICE_DCESCRIPTOR_STRUCT
{
BYTE blength;                          //设备描述符的字节数大小
BYTE bDescriptorType;                  //设备描述符类型编号
WORD bcdUSB;                           //USB 版本号
BYTE bDeviceClass;                     //USB 分配的设备类代码
BYTE bDeviceSubClass;                  //USB 分配的子类代码
BYTE bDeviceProtocol;                  //USB 分配的设备协议代码
BYTE bMaxPacketSize0;                  //端点 0 的最大包大小
WORD idVendor;                         //厂商编号
WORD idProduct;                        //产品编号
WORD bcdDevice;                        //设备出厂编号
BYTE iManufacturer;                    //设备厂商字符串的索引
BYTE iProduct;                         //描述产品字符串的索引
BYTE iSerialNumber;                    //描述设备序列号字符串的索引
BYTE bNumConfigurations;               //可能的配置数量
}
DEVICE_DESCRIPTOR_STRUCT, * pDEVICE_DESCRIPTOR_STRUCT;
```

2.配置描述符

```
//定义标准的配置描述符结构

typedef struct _CONFIGURATION_DESCRIPTOR_STRUCT
{
BYTE bLength;                          //配置描述符的字节数大小
BYTE bDescriptorType;                  //配置描述符类型编号
WORD wTotalLength;                     //此配置返回的所有数据大小
BYTE bNumInterfaces;                   //此配置所支持的接口数量
BYTE bConfigurationValue;              //Set_Configuration 命令所需要的参数值
BYTE iConfiguration;                   //描述该配置的字符串的索引值
BYTE bmAttributes;                     //供电模式的选择
BYTE MaxPower;                         //设备从总线提取的最大电流
}
CONFIGURATION_DESCRIPTOR_STRUCT, * pCONFIGURATION_DESCRIPTOR_STRUCT;
```

2.接口描述符

//定义标准的接口描述符结构

```
typedef struct _INTERFACE_DESCRIPTOR_STRUCT
{
BYTE bLength;                    //接口描述符的字节数大小
BYTE bDescriptorType;            //接口描述符的类型编号
BYTE bInterfaceNumber;           //该接口的编号
BYTE bAlternateSetting;          //备用的接口描述符编号
BYTE bNumEndpoints;              //该接口使用的端点数，不包括端点 0
BYTE bInterfaceClass;            //接口类型
BYTE bInterfaceSubClass;         //接口子类型
BYTE bInterfaceProtocol;         //接口遵循的协议
BYTE iInterface;                 //描述该接口的字符串索引值
}
INTERFACE_DESCRIPTOR_STRUCT, * pINTERFACE_DESCRIPTOR_STRUCT;
```

一个USB设备只有一个设备描述符--N1配置描述符 -----N个接口描述符--------N个端点描述符：定义了端点的大小和类型
N2配置描述符
N3配置描述符
。。。
主机获取描述符时，1获取设备描述符->2获取配置描述符
设备描述符记录的信息有：设备所使用的USB协议版本号，设备类型，端点0的最大包大小，厂商ID（VID）和产品ID（PID），设备版本号，厂商字符串索引，产品字符串索引，设备序列号索引，可能的配置数。
配置的描述符主要记录的信息有：配置所包含的接口数，配置的编号，供电方式马事发支持远程唤醒，电流需求量
接口描述符：只要记录接口编号，接口的端点数，接口所使用的类子类协议。
端点描述符：端点号，方向，端点的传输类型，最大包长度，查询时间间隔。
字符串描述符：提供一些方便人们查阅的信息不是必须的。
USB总线上传输数据以包为基本单位。一个包被分成不同的域。根据不同类型的包，所包含的域是不一样的，但是不通的包有共同的特点，就是同步域开始紧跟着一个包标识符PID,最终以包结束符EOP来结束这个包。
包标识符PID是用来标识一个包的类型。它总共有8位，其中USB协议使用的只有4位，PID0～PID3 另外4位PID4～PID7是PID0～PID3的取反用来检验PID。USB协议规定了4类包，
令牌包PID1～0 为01，
数据包PID1~0为11
握手包PID1~0为10
特殊包PID1~0为00不同类的包又分成几种具体的包。
令牌包：用来启动一次USB传输。输出，输入，建立和帧起始。

设备描述符：一个设备只有一个设备描述符

```
typedef struct _USB_DEVICE_DESCRIPTOR_
{
   BYTE       bLength,
   BYTE       bDescriptorType,
   WORD       bcdUSB,
   BYTE       bDeviceClass,
   BTYE       bDeviceSubClass,
   BYTE       bDeviceProtol,
   BYTE       bMaxPacketSize0,
   WORD       idVenderI,
   WORD       idProduct,
   WORD       bcdDevice,
   BYTE       iManufacturer,
   BYTE       iProduct,
   BYTE       iSerialNumber,
   BYTE       iNumConfiguations
}USB_DEVICE_DESCRIPTOR;
```

bLength：描述符大小．固定为0x12．
bDescriptorType：设备描述符类型．固定为0x01．
bcdUSB：USB 规范发布号．表示了本设备能适用于那种协议，如2.0=0200，1.1=0110等．

bDeviceClass：类型代码（由USB指定）。当它的值是0时，表示所有接口在配置描述符里，并且所有接口是独立的。当它的值是1到FEH时，表示不同的接口关联的。当它的值是FFH时，它是厂商自己定义的．
bDeviceSubClass：子类型代码（由USB分配）．如果bDeviceClass值是0，一定要设置为0．其它情况就跟据USB-IF组织定义的编码．

bDeviceProtocol：协议代码（由USB分配）．如果使用USB-IF组织定义的协议，就需要设置这里的值，否则直接设置为0。如果厂商自己定义的可以设置为FFH．
bMaxPacketSize0：端点0最大分组大小（只有8,16,32,64有效）．

idVendor：供应商ID（由USB分配）．

idProduct：产品ID（由厂商分配）．由供应商ID和产品ID，就可以让操作系统加载不同的驱动程序．

bcdDevice：设备出产编码．由厂家自行设置．

iManufacturer：厂商描述符字符串索引．索引到对应的字符串描述符． 为0则表示没有．
iProduct：:产品描述符字符串索引．同上．
iSerialNumber：设备序列号字符串索引．同上．
bNumConfigurations：可能的配置数．指配置字符串的个数

配置描述符：配置描述符定义了设备的配置信息，一个设备可以有多个配置描述符

```
typedef struct _USB_CONFIGURATION_DESCRIPTOR_
{
   BYTE     bLength,
   BYTE     bDescriptorType,
   WORD     wTotalLength,
   BYTE     bNumInterfaces,
   BYTE     bConfigurationValue,
   BYTE     iConfiguration,
   BYTE     bmAttributes,
   BYTE     MaxPower
}USB_CONFIGURATION_DESCRIPTOR;
```

bLength：描述符大小．固定为0x09．
bDescriptorType：配置描述符类型．固定为0x02．
wTotalLength：返回整个数据的长度．指此配置返回的配置描述符，接口描述符以及端点描述符的全部大小．
bNumInterfaces：配置所支持的接口数．指该配置配备的接口数量，也表示该配置下接口描述符数量．
bConfigurationValue：作为Set Configuration的一个参数选择配置值．
iConfiguration：用于描述该配置字符串描述符的索引．
bmAttributes：供电模式选择．Bit4-0保留，D7:总线供电，D6:自供电，D5:远程唤醒．
MaxPower：总线供电的USB设备的最大消耗电流．以2mA为单位．

接口描述符：接口描述符说明了接口所提供的配置，一个配置所拥有的接口数量通过配置描述符的bNumInterfaces决定

```
typedef struct _USB_INTERFACE_DESCRIPTOR_
{
   BYTE     bLength,
   BYTE     bDescriptorType,
   BYTE     bInterfaceNumber,
   BYTE     bAlternateSetting,
   BYTE     bNumEndpoint,
   BYTE     bInterfaceClass,
   BYTE     bInterfaceSubClass,
   BYTE     bInterfaceProtocol,
   BYTE     iInterface
}USB_INTERFACE_DESCRIPTOR;
```

bLength：描述符大小．固定为0x09．
bDescriptorType：接口描述符类型．固定为0x04．
bInterfaceNumber: 该接口的编号．
bAlternateSetting：用于为上一个字段选择可供替换的位置．即备用的接口描述符标号．
bNumEndpoint：使用的端点数目．端点0除外．
bInterfaceClass：类型代码（由USB分配）．
bInterfaceSubClass：子类型代码（由USB分配）．
bInterfaceProtocol：协议代码（由USB分配）．
iInterface：字符串描述符的索引

端点描述符：USB设备中的每个端点都有自己的端点描述符，由接口描述符中的bNumEndpoint决定其数量

```
typedef struct _USB_ENDPOINT_DESCRIPTOR_
{
   BYTE       bLength,
   BYTE       bDescriptorType,
   BYTE       bEndpointAddress,
   BYTE       bmAttributes,
   WORD       wMaxPacketSize,
   BYTE       bInterval
}USB_ENDPOINT_DESCRIPTOR;
```

bLength：描述符大小．固定为0x07．
bDescriptorType：接口描述符类型．固定为0x05．
bEndpointType：USB设备的端点地址．Bit7，方向，对于控制端点可以忽略，1/0:IN/OUT．Bit6-4，保留．BIt3-0：端点号．
bmAttributes：端点属性．Bit7-2，保留．BIt1-0：00控制，01同步，02批量，03中断．
wMaxPacketSize：本端点接收或发送的最大信息包大小．

bInterval：轮训数据传送端点的时间间隔．对于批量传送和控制传送的端点忽略．对于同步传送的端点，必须为１，对于中断传送的端点，范围为１－２５５．

字符串描述符：其中字符串描述符是可选的．如果不支持字符串描述符，其设备，配置，接口描述符内的所有字符串描述符索引都必须为０

```
typedef struct _USB_STRING_DESCRIPTION_
{
    BYTE      bLength,
    BYTE      bDescriptionType,
    BYTE      bString[1];
}USB_STRING_DESCRIPTION;
```

bLength：描述符大小．由整个字符串的长度加上bLength和bDescriptorType的长度决定．
bDescriptorType：接口描述符类型．固定为0x03．
bString[1]：Unicode编码字符串．

天池渔隐
关注 - 5
粉丝 - 0
+加关注

« 上一篇：常用的算法思想
» 下一篇：树莓派 原理图 摄像头接口定义

发表于 2016-01-18 17:29 天池渔隐 阅读(7902) 评论(0) 编辑 收藏