第8章 USB 接口 HID 设备

HID （Human Interface Device ，人机接口设备）是 USB 设备中常用的设备类型，是直接与人交互的 USB 设备，例如键盘、鼠标与游戏杆等。在 USB 设备中， HID 设备的成本较低。另外， HID 设备并不一定要有人机交互功能，只要符合 HID 类别规范的设备都是 HID 设备。

Wndows 操作系统最先支持的 HID 设备。在 windows 98 以及后来的版本中内置有 HID 设备的驱动程序，应用程序可以直接使用这些驱动程序来与设备通信。

在设计一个 USB 接口的计算机外部设备时，如果 HID 类型的设备可以满足需要，可以将其设计为 HID 类型设备，这样可以省去比较复杂的 USB 驱动程序的编写，直接利用 Windows 操作系统对标准的 HID 类型 USB 设备的支持。

8.1 HID 设备简介

8.1.1 HID 设备的特点

交换的数据储存在称为报表 （Report ）的结构内， 设备的固件必须支持 HlD 报表的格式。 主机通过控制和中断传输中的传送和请求报表来传送和接收数据。 报表的格式非常灵活。

每一笔事务可以携带小量或中量的数据。低速设备每一笔事务最大是 8B，全速设备每一笔事务最大是 64B，高速设备每一笔事务最大是 1024B 。一个报表可以使用多笔事务。

设备可以在未预期的时间传送信息给主机， 例如键盘的按键或是鼠标的移动。 所以主机会定时轮询设备，以取得最新的数据。

HID 设备的最大传输速度有限制。主机可以保证低速的中断端点每 10ms 内最多 1 笔事务，每一秒最多是 800B 。保证全速端点每 lms 一笔事务，每一秒最多是 64000B 。保证高速端点每 125 us 三笔事务，每一秒最多是 24.576MB 。

HID 设备没有保证的传输速率。如果设备是设置在 10ms 的时距，事务之间的时间可能等于或小于 10ms 。除非设备是设置在全速时在每个帧传输数据，或是在高速时在每个微帧传输数据。 这是最快的轮询速率， 所以端点可以保证有正确的带宽可供使用。

HID 设备除了传送数据给主机外，它也会从主机接收数据。只要能够符合 HlD 类别规范的设备都可以是 HID 设备。

设备除了 HlD 接口之外，它可能同时还包含有其他的 USB 接口。例如影像显示设备可能使用 HID 接口来做亮度、对比度的软件控制，而使用传统的影像接口来传送要显示的数据。USB 扩音器可以使用实时传输来播放语音， 同时使用 HID 接口来控制音量、 低音等。

HID 类别设备的规范文件主要是以下两份：

Device Class Definition for Human interface Devices

HID Usage Tables

其中前者是 HID 的基本规范文件，后者可以是前者的附件，为开发人员提供实际的控制类型的描述。文件是用来定义让主机了解以及使用 HID 数据的数值。这两份文件是由 USB Device Working Group 制定的，可以在网址 http://www.usb.org/developers/hidpage/#Class _Definition 下载。

8.1.2 HID 设备的硬件要求

HID 接口必须符合 Device Class Definition for Human interface Devices 规范内所定义的 HID 类别的需求。在此文件内描述了所需的描述符、传输的频率以及传输的类型等。为了符合规范， HID 接口的端点与描述符都必须符合数个要求。

所有的 HID 传输都是使用默认控制管道或是一个中断管道， HID 设备必须有一个中断输入端点来传送数据到主机，中断输出端点则不是必需的。

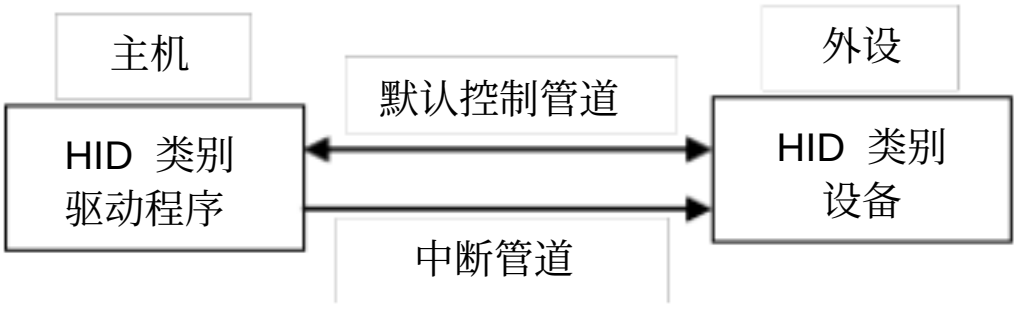

图 8-1 HID 传输的传输类型

表 8-1 HID 设备的传输类型

传输类型	数据来源	数据类型	是否需要管道
控制	设备（输入）	没有严格时间限制的数据	是
	主机（输出）	没有严格时间限制的数据 或是没有中断输出管道时的任何数据	
中断	设备（输入）	定时或低延迟的数据	是
	主机（输出）	定时或低延迟的数据	是

主机与设备之间所交换的数据，可以分成两种类型：

低延迟的数据，必须尽快地到达目的；

配置或其他的数据，没有严格时间限制的需求。

中断管道是控制管道之外的另一种数据交换的方式，特别适合使用在接收端需要定时或是尽可能及时收到数据的时候。中断输入管道携带数据到主机，中断输出管道则是携带数据到设备。在总线忙的时候，控制管道可能会被延迟，而中断管道保证会有可得到的带宽。 HID 不需要一定有中断输出管道。如果没有中断输出管道，主机会在控制管道上使用 HID 设备特有的 Set_Report 请求来传送所有的报表。

8.1.3 HID 固件的要求

主机的驱动程序要与 HID 设备通信，设备的固件必须符合下列需求：

设备的描述符必须识别该设备包含有 HID 接口。

除了默认控制管道外，固件必须另外支持一个中断输入管道。

固件必须包含一个报表描述符来定义要传送与接收的设备数据。

如果要传送数据，固件必须支持 Get_Report 控制传输与中断输入传输。如果要接收数据，固件必须支持 Set_Report 控制传输与选择性的中断输出传输。

所有的 HID 数据都必须使用定义过的报表格式来定义报表中数据的大小与内容。 设备可以支持一个或多个报表。在固件中的一个报表描述符用来描述此报表，以及如何使用报表数据的信息。

在每一个报表中的一个数值，定义此报表是一个输入（ Input）、输出（ Output ）或是特征（ Feature ）报表。主机在输入报表中接收数据，在输出报表中传送数据，特征报表可以在任何方向传递。

Windows 98 以及后来版本的 HID 驱动程序使用中断传输来传递输入报表。输出报表的传输类型要根据设备支持的端点与 Windows 的版本而定。Windows 98 Gold 只符合 HID 1.0 规范，它的 HID 驱动程序使用控制传输来传递输出报表。 Windows 98 SE 、Wndows 2000 符合 HID 1.1 规范， HID 驱动程序在有中断输出端点时使用中断传输， 否则使用控制传输来传递输出报表。特征报表都是使用控制传输。

8.2 HID 设备描述符

HID 设备连接到 USB 主机后，主机通过发送 Get_Descriptor 请求读取 HID 设备的描述符，了解描述符对了解 USB 设备是至关重要的。

8.2.1 HID 设备的描述符

HID 设备除了支持 USB 设备的 5 种标准描述符之外，还支持 HID 设备特有的 3 种描述符。这些描述符是：

USB 标准描述符：设备、配置、接口、端点和字符串描述符。

HID 特有的描述符： HID 、报表（ Report ）和实体（ Physical ）描述符。

从描述符的关联关系看， HID 描述符是关联于接口 。所以如果一个 HID 设备有 2 个端点，设备不需要每个端点有一个 HID 描述符。

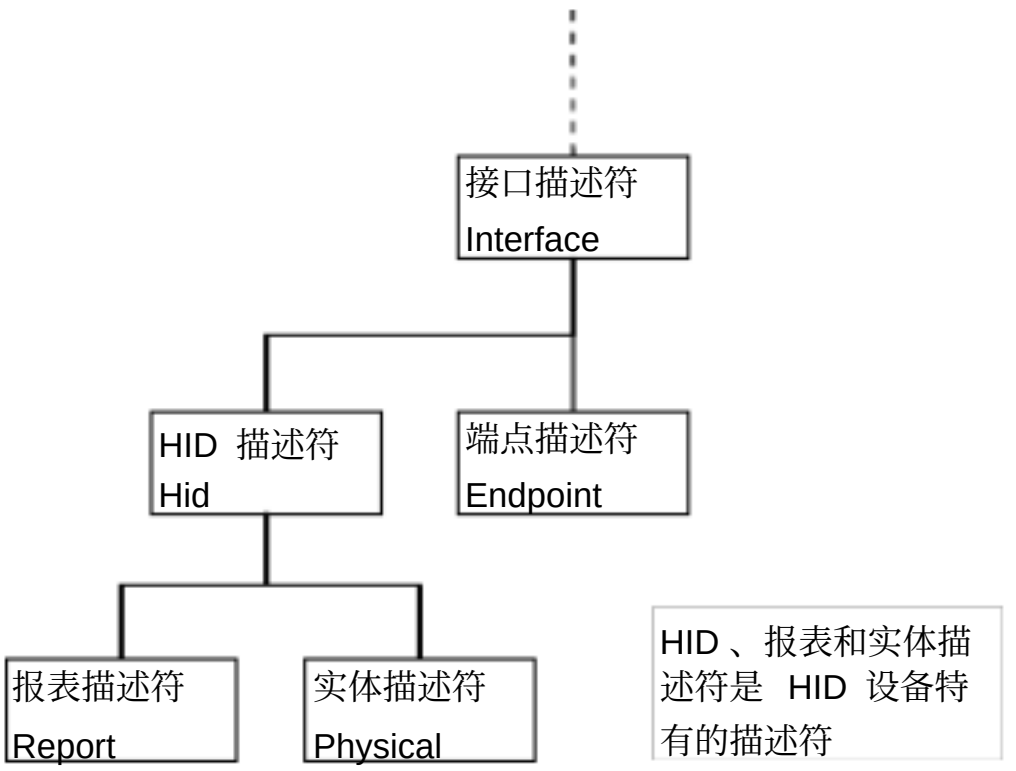

图 8-2 HID 描述符的关联关系

从前面的 USB 描述符可以看出一个规律，描述符的第一、二字节分别是描述符的长度和类型，描述符的类型字段（ bDescriptorType ）表明描述符的种类，下表列出了不同描述符的类型字段数值。

表 8-2 HID 的描述符

类型	描述符	应用	数值
标准	设备 Device	所有设备必须有，只能一个	01
	配置 Configuration	所有设备必须有，至少一个	02
	字符串 String	可选择	03
	接口 Interface	每一个接口一个	04
	端点 Endpoint	除端点 0 之外的每个端点一个	05
	设备限定 Device_Qualiffier	同时支持全速与高速的设备必须有一个	06
	Other_Speed_Configuration		07
	Interface_power		08
类别	HID	HID 设备必须有	21
	Hub		29
HID 特定	报表 Report	HID 设备必须有	22
	实体 Physical	可选择的	23

对于一个 HID 设备，设备描述符与配置描述符没有 HID 特定的信息。其设备描述符的 bDeviceClass 和 bDeviceSubClass 字段的值为 0，接口描述符的 bInterfaceClass 字段值为 03，表示设备的该接口是 HID 类别。在接口描述符中其他包含 HID 特定信息的字段还有子类别码（ bInterfaceSubClass ）与协议码（ bInterfaceProtocol 字段）。

在接口描述符中子类别码字段等于 1 表示此设备支持启动接口（ Boot Interface ）。如果设备有启动接口，即便主机的 HID 没有加载驱动程序，设备也可以使用。这种情形可能发生在计算机是由 DOS 直接启动，在启动时观看系统设置画面或使用 Wndows 的安全模式时。

含有启动接口的键盘或鼠标可以使用 BIOS 或许多主机支持的默认简单协议。HID 规范定义了键盘与鼠标的启动接口协议。

如果设备没有启动接口，并且接口描述符中协议码字段是 1，表示设备支持键盘接口，协议码字段是 2，表示支持鼠标接口。接口描述符中协议码字段是 0，表示设备不支持启动协议。

在 HID Usage Tables 规范中定义了键盘与鼠标的启动描述符（ Boot Descriptor ）。BIOS 不需要从设备中读取描述符，因为它知道启动协议，并且假设设备支持启动协议。所以要启动的设备不需要在固件内包含启动接口描述符，它只要在主机尚未要求在报表描述符中的定义协议时支持启动协议即可。在操作系统加载 HID 驱动程序后会使用 Set_Protocol 请求，将设备由启动协议转换成报表协议。

8.2.2 HID 描述符

HID 描述符的主要作用是用来识别 HID 通信所使用的额外描述符。下表是 HID 描述符结构。

表 8-3 HID 描述符结构

偏移量	字段	字节数	数值类型	说明
0	bLength	1	Numeric	描述符字节数
1	bDescriptorType	1	Constant	0x21 = HID 描述符
2	bcdHID	2	Numeric	HID 规范版本号（ BCD ）
4	bCountryCode	1	Numeric	硬件设备所在国家的国家代码
5	bNumDescriptors	1	Numeric	类别描述符数目（至少有一个报表描述符）
6	bDescriptorType	1	Constant	类别描述符的类型
7	wDescriptorLength	2	Numeric	报表描述符的总长度

9	[bDescriptorType]...	1	Constant	附加的描述符的类型，可选的
10	[wDescriptorLength]...	2	Numeric	附加的描述符的总长度，可选的

bcdHID ：设备与其描述符所遵循的 HID 规范的版本号码，此数值是 4 个 16 进制的 BCD 格式字符。例如 版本 1.1 的 bcdHID 是 0110h 。（2 bytes ）

bCountryCode ：硬件目的国家的识别码。如果不说明，该字段为 0。

bDescriptorType ：HID 描述符附属的描述符的类型（报表或实体） 。每一个 HID 都必须至少支持一个报表描述符。一个接口可以支持多个报表描述符，以及一个或多个实体描述符。

HID 描述符的偏移量为 9 和 10 的 bDescriptorType 和 wDescriptorLength 可以重复存在多个。

1. 报表描述符

报表描述符定义了执行设备功能的数据格式和使用方法。

报表描述符和 USB 的其他描述符是不一样的，它不是一个简单的表格， 报表描述符是 USB 所有描述符中最复杂的 。报表描述符非常复杂而有弹性，因为它需要处理各种用途的设备。报表的数据必须以简洁的格式来储存，这样才不会浪费设备内的储存空间以及数据传输时的总线时间。

实际上可以这样理解， 报表内容的简洁，是通过报表描述符全面的、复杂的数据描述实现的 。

报表描述符必须先描述数据的大小与内容。报表描述符的内容与大小因设备的不同而不同，在进行报表传输之前，主机必须先请求设备的报表描述符，只有得到了报表描述符才可正确解析报表的数据。

报表描述符是报表描述项目（ Item ）的集合，每一个描述项目都有相对统一的数据结构，项目很多，通过编码实现。

（1）项目

报表描述符由描述 HID 设备的数据 项目（ Item ） 组成。

项目的 第一个字节 （项目前缀）由三部分构成：

项目标志（ item Tag ）：说明项目的功能，

项目类型（ item Type ）：说明项目的数据类型，

项目长度（ item Size ）：说明项目的数据部分的长度。

HID 的项目有短项目和长项目两种，其中短项目的格式如下图。

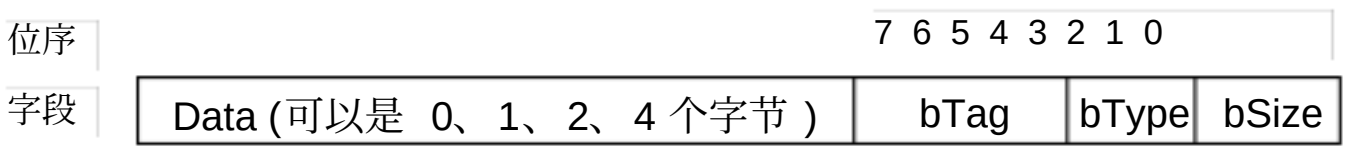

图 8-3 HID 报表短项目格式

短项目的数据字节数由 bSize 的值定义， bSize 为 0、1、2、3 时 Data 部分的字节数分别为 0、1、2、4 个字节。（nn 为数据长度）

短项目的项目类型由 bType 定义， bType 为 0、1、2 时分别为 Main 、Global 和 Local 类型。 (见后面的表 8-4 HID 项目列表)

长项目可以携带较多的数据，其格式如下图。

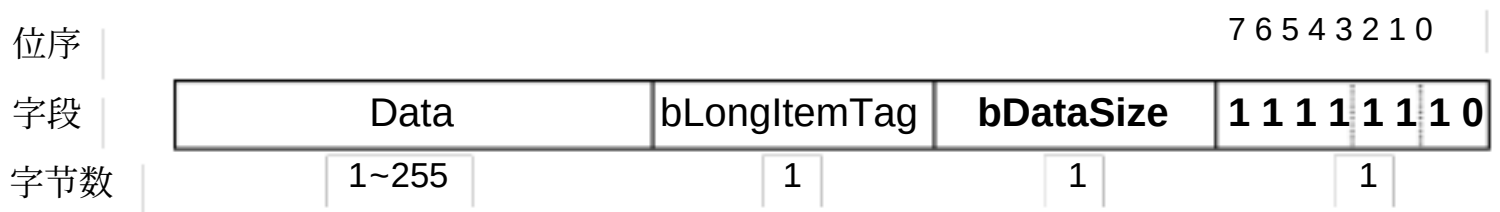

图 8-4 HID 报表长项目格式

项目中的第一个字节为上图中的 特定值 时表明该项目是一个 长项目 。长项目中的 bDataSize 说明 Data 部分的字节数, bLongItemT ag 在 HID 规范中没有定义。

下面是通过汇编实现的一个简单的报表描述符，描述符的每一行是一个项目，该描述符描述了一个 从设备接收 2 个字节的输入报表 和发送 2 个字节到设备的输出报表 。

```
HID_Report_desc_table:
        db  06h, A0h, FFh           ; Usage Page(Vendor defined)          定义设备功能
        db  09h, A5h                ; Usage(Vendor Defined)               定义用法
        db  A1h, 01h                ; Collection(Application)             开一个集合
        db  09H, A6h                ;       Usage(Vendor defined)         定义用法

                                    ;       输入报表
        db  09h, A7h                ;       Usgae(Vendor defined)         定义用法
        db  15h, 80h                ;       Logical Minimum               定义输入最小值 =-128
        db  25h, 7Fh                ;       Logical Maximum               定义输入最大值 =+127
        db  75h, 08h                ;       Report Size                   定义报表数据项大小 =8
        db  95h, 02h                ;       Report Count                  定义报表数据项个数 =2
        db  81h, 02h                ;       Input(Data,Variable,Absolute)       输入项目

                                    ;       输出报表
        db  09h, A9h                ;       Usgae(Vendor defined)         定义用法
        db  15h, 80h                ;       Logical Minimum               定义输入最小值 =-128
        db  25h, 7Fh                ;       Logical Maximum               定义输入最大值 =+27
        db  75h, 08h                ;       Report Size                   定义报表数据项大小 =8
        db  95h, 02h                ;       Report Count                  定义报表数据向个数 =2
        db  91h, 02h                ;       Output(Data,Variable,Absolute)      输出项目

        db  C0h                     ; End Collection                      关闭集合
```

（2）项目的分类

报表的项目有 Main 、Global 和 Local 三大类， 每一类都有多个不同的项目， 实现不同的描述。

Main 类项目用于定义报表描述符中的数据项。 也可以组合其中的若干数据项成为一个集合。 Main 项目可以分为带数据的 Main 项目和不带数据的 Main 项目。带数据项的 Main 用于生成报表中的数据项，包括 Input 、Output 和 Feature 项目。不带数据的 Main 项目不生成报表中的数据项，包括 Collection 和 End Collection 项目。

Global 类项目实现对数据的描述，用来识别报表并且描述报表内的数据，包括数据的功能、最大与最小允许值以及数据项的大小与数目等。改变由 Main 类项目生成的项目状态表。 Global 类项目描述对后续的所有项目有效，除非遇到有新的 Global 类项目。

Local 类项目定义控制的特征， 这一类项目的作用域不超过下一个 Main 项目， 所以在每一 Main 项目之前可能有多个 Local 项目。 Local 项目用于描述后面的 Input 、 Output 和 Feature 项目。

下表列出的是全部的项目的前缀字和简要功能说明。

表 8-4 HID 项目列表

项目类型	项目标志（ Tag ）	项目前缀， nn 为数据长度	功能说明
Main 类项目（00）	Input	1000 00 nn	定义输入报表，主机利用该信息解析设备提供的数据。主机向控制端口发送 Get_Report 实现输入
	Output	1001 00 nn	创建输出报表， 通过向设备发送 Set_Report 实现输出
	Feature	1011 00 nn	定义送往设备的设置信息
	Collection	1010 00 nn	定义 2 个以上数据（ Input 、 Output 和 Feature ）的关系为集合， Collection 开始一个集合， 之后的 End Collection 结束集合。 Collection 项目的数据部分说明 Collection 的类型
	End Collection	1100 00 nn	
Global 类项目（01）	Usage Page	0000 01 nn	指定设备的功能 （06h,A0h,FFh） 另外由于 Usage项目有 32位数据值， Usage Page 项目用于为 Usage项目在报表描述符中占居存储空间。用于存放后续的 Usage项目的高 16位。
	Logical Minimum	0001 01 nn	定义变量或数组项目的逻辑最小值和最大值 (-128,+127) (15h, 80h) (25h, 7Fh)
	Logical Maximum	0010 01 nn	
	Physical Minimum	0011 01 nn	定义变量或数组项目的物理最小值和最大值，分别和 Logical Minimum 、 Logical Maximum 对应
	Physical Maximum	0100 01 nn	
	Unit Exponent	0101 01 nn	定义数值是基于 10 的指数
	Unit	0110 01 nn	单位
	Report Size	0111 01 nn	指定报表数据区域所包含的位数 （=8）
	Report ID	1000 01 nn	报表 ID，该项目在报表中插入一个字节的报表 ID
	Report Count	1001 01 nn	报表中数据域的数目 (=2)
	Push	1010 01 nn	将 Global 项目状态表送入堆栈
	Pop	1011 01 nn	从堆栈恢复 Global 项目状态表
		1100 01 nn – 1111 01 nn	保留
Local 类项目（10）	Usage	0000 10 nn	用法索引值，表示对项目或集合建议的用法，用于当一个项目描述多个控制，对每一个变量和数组元素都有建议的用法（ 09h,A7h ）
	Usage Minimum	0001 10 nn	定义阵列或位图中控制操作的第一个和最后一个用法
	Usage Maximum	0010 10 nn	
	Designator Index	0011 10 nn	确定用于控制的实体，指向物理描述符中的目标
	Designator Minimum	0100 10 nn	定义阵列或位图目标的起始和终止索引值
	Designator Maximum	0101 10 nn	
	String Index	0111 10 nn	确定字符串描述符中的索引值
	String Minimum	1000 10 nn	定义用于阵列或位图控制中字符串序列索引值的最小值和最大值
	String Maximum	1001 10 nn	
	Delimiter	1010 10 nn	定义一组 Local 项目的开始和结束， 1=开始， 0=结束
		1010 10 nn – 1111 10 nn	保留

在这些项目中， Usage Page 用来指定设备的功能，而 Usage 项目用来指定个别报表的功能。 Usage Page 项目相当于是 HID 的子集合， Usage 相当于是 Usage Page 的子集合。

2. 报表描述符的项目

（1）Input、Outpot 和 Feature 项目

这 3 个项目用来定义报表中的数据字段。

Input 项目可以应用到任何控制、 计数器读数或其他设备传给主机的信息。 一个输入报表包含一个或多个 Input 项目，主机使用中断输入传输来请求输入报表。

Ouput 项目用来定义主机传送给设备的信息。一个输出报表包含一个或多个 Outpot 项目。输出报表包含控制状态的数据。 如果有中断输出管道， HID1.1 兼容主机使用中断输出传输来传送输出报表，否则使用 Set_Report 控制请求。

Feature 项目应用到主机传送给设备的信息，或是主机从设备读取 Feature 项目。一个特征报表包含一个或多个 Feature 项目， Feature 项目通常是包含影响设备与其组件整体行为的配置。特征报表通常是控制可以使用实际的控制面板调整的设置，例如主机可以使用虚拟控制面板来让用户选择控制特征。 主机使用 Set_Report 与 Get_Report 请求来传送与接收特征报表。

在每一个 Input、Output 和 Feature 项目的前缀字之后是 32 位描述数据，目前最多定义了 9 个位，余的位则是保留。位 0~8 的定义中只有位 7 不能应用于 Input 项目，除此之外其他的位定义都适应于 Input、Output 和 Feature 项目。

表 8-5 Input、Output 和 Feature 项目的数据项说明

数据字段			含义说明
位	值	名称	
0	0	Data	数据：表示项目的内容是可更改的（读/写）。
	1	Constant	常数：表示项目的内容是不可更改的（只读）。
1	0	Array	数组：报告全部控制的状态。如在键盘报表中每一个键在报表中占一位，报表传输全部键的状态，可以同时按下任意多个键。
	1	Variable	变量：报告作用中的控制。如在键盘报表中只报告按下的键的编号，可以同时按下的键的数目等于报表的计数（Global 类项目 Report Count ）
2	0	Absolute	绝对：表示数值以一个固定值为基准。游戏杆通常是报告绝对数据（游戏杆目前的位置）。
	1	Relative	相对：表示数据的改变以上一个读数为基准。鼠标通常是报告相对数据（鼠标的移动位置）。
3①	0	No Wrap	如果设置为 1 表示回转，当数值超过最小值到最大值的范围时将回转，如果最小值是 0 而最大值是 10，超过最大值的下一个数值是 0。
	1	Wrap	
4①	0	Linear	线形：表示测量的数据与报表的数据有线性的关系。
	1	Non-Linear	非线性：表示测量的数据与报表的数据没有线性的关系。
5①	0	Preferred	优选状态：表示控制在没有用户交互时会回到一个特定的状态。如按钮就有优选状态，在无操作时保持未按下的状态。
	1	Non-Preferred	非优选状态： 它维持在上一个用户选择的状态。如交替的开关就没有优选状态。
6①	0	No Null Position	无空状态位置：表示控制永远在传送有效的数据。
	1	Null State	空状态： 表示控制支持一个没有传送有效数据的状态。如操纵杆可能具有一个多方向的按钮开关，在没有按下时在空状态，这时控制将传送一个在 Logical Minimum 与 Logical Maximum 范围之外的数值来表示它在空状态。
7②	0	Non-Volatile	不可变的： 表示设备只有在主机请求时才改变数值。当主机传送一个报表并且不要改变不可变项目时，如果该项目是定义成相对（Relative ）的，数值 0 表示不改变数据，如果不可变项目是定义成绝对 （Absolute ）的，超出范围外的数值则表示不改变数据。

	1	Volatile	可变的：表示设备可以自己改变数值，并不是必须主机传送报表要求给设备来改变数值。例如设备控制面板可以由主机软件传送一个报表给设备，也可以由用户自己按设备上的实际按钮。
8 ①	0	Bit Field	位字段：表示每一个位或是一个字节内的一组位可以代表一份数据。
	1	Buffered Bytes	缓冲字节：表示信息包含一个或多个字节，缓冲字节的报表大小必须是 8。
9~31 位			保留

注：①：该位不能应用到数组。

②：只应用于 Output 和 Feature 项目，对于 Input 项目该位保留。

（2）Collection 和 End Collection 项目

所有的报表类型都可以使用 Collection 与 End Collection 项目来将相关的 Main 类型项目组成群组。这两个项目分别用于打开和关闭集合。所有在 Collection 与 End Collection 项目之间的 Main 类型项目都是 Collection 的一部分。

Collection 有 3 种类型：Application 、Physical 与 Logical，其项目的数据项的值分别为 1、0 和 2。厂商也可以自己定义 Collection 类型，数据项的值为 80h~FFh 保留给厂商定义。End Collection 项目无数据项。

Application Collection 包含有共同用途的项目或执行单一功能的项目。例如键盘的开机描述符将键盘的按键与 LED 指示灯数据集合成一个 Application Collection 。所有的报表必须在一个 Application Collection 内。

Physical Collection 包含在一个单一几何点上的数据项目，可以将每个位置的数据集合成一个 Physical Collection 。在设备报告多个传感器的位置的时候，使用 Physical Collection 指明不同的数据来自不同的传感器。

Logical Collection 形成一个数据结构，包含由 Collection 所连结的不同类型的项目。例如数据缓冲区的内容以及缓冲区内字节数目的计数。

（3）Usage Page 和 Usage 项目

Usage page 项目的数据部分为 1~2 个字节，目前的定义全部都是一个字节。Usage Page 定义了常用的设备功能，关于 Usage Page （以及其他项目）的具体定义内容，可以查阅 HID Usage tables （http://www.usb.org/developers/hidpage/#Class_Definition ），下表是来自 HID Usage tables 的 Usage Page 定义。

表8-6 Usage Page 定义

Page ID	Page Name
00	Undefined
01	Generic Desktop Controls
02	Simulation Controls
03	VR Controls
04	Sport Controls
05	Game Controls
06	Generic Device Controls
07	Keyboard/Keypad
08	LEDs
09	Button
0A	Ordinal
0B	Telephony
0C	Consumer
0D	Digitizer
0E	Reserved

0F	PID Page
10	Unicode
11-13	Reserved
14	Alphanumeric Display
15-3f	Reserved
40	Medical Instruments
41-7F	Reserved
80-83	Monitor pages
84-87	Power pages
88-8B	Reserved
8C	Bar Code Scanner page
8D	Scale page
8E	Magnetic Stripe Reading (MSR) Devices
8F	Reserved Point of Sale pages
90	Camera Control Page
91	Arcade Page
92-FEFF	Reserved
FF00-FFFF	Vendor-defined

关于 Usage Page 的每一个有效定义项，都有一个相应的下一级定义，如 Usage Page 的数据项数值为 1，则设备定义为 Generic Desktop Controls ，关于该类设备的具体功能可以在 HID Usage Tables 中查到具体的定义。下表是 HID Usage Tables 中对 Generic Desktop Controls 设备的功能定义。

表 8-7 Generic Desktop Controls 用法定义

Usage ID	Usage Name	Usage Type	参阅 HID Usage Tables 中的相关章节
00	Undefined		
01	Pointer	CP	4.1
02	Mouse	CA	
03	Reserved		
04	Joystick	CA	4.1
05	Game Pad	CA	
06	Keyboard	CA	
07	Keypad	CA	
08	Multi-axis Controller	CA	
09	Tablet PC System Controls	CA	
0A-2F	Reserved		
30	X	DV	4.2
31	Y	DV	
32	Z	DV	
33	Rx	DV	
34	Ry	DV	
35	Rz	DV	
36	Slider	DV	4.3
37	Dial	DV	
38	Wheel	DV	
39	Hat switch	DV	
3A	Counted Buffer	CL	4.6

3B	Byte Count	DV	
3C	Motion Wakeup	OSC	4.3
3D	Start	OOC	
3E	Select	OOC	
3F	Reserved		
40	Vx	DV	4.3.1
41	Vy	DV	
42	Vz	DV	
43	Vbrx	DV	
44	Vbry	DV	
45	Vbrz	DV	
46	Vno	DV	
47	Feature Notification	DV,DF	4.8
48	Resolution Multiplier	DV	
49-7F	Reserved		
80	System Control	CA	4.5
81	System Power Down	OSC	
82	System Sleep	OSC	4.5.1
83	System Wake Up	OSC	
84	System Context Menu	OSC	4.5
85	System Main Menu	OSC	
86	System App Menu	OSC	
87	System Menu Help	OSC	
88	System Menu Exit	OSC	
89	System Menu Select	OSC	
8A	System Menu Right	RTC	
8B	System Menu Left	RTC	
8C	System Menu Up	RTC	
8D	System Menu Down	RTC	
8E	System Cold Restart	OSC	4.5.1
8F	System Warm Restart OSC		
90	D-pad Up	OOC	4.7
91	D-pad Down	OOC	
92	D-pad Right	OOC	
93	D-pad Left	OOC	
94-9F	Reserved		
A0	System Dock	OSC	4.5.1
A1	System Undock	OSC	
A2	System Setup	OSC	
A3	System Break	OSC	4.9
A4	System Debugger Break	OSC	
A5	Application Break	OSC	
A6	Application Debugger Break	OSC	
A7	System Speaker Mute	OSC	4.5.1
A8	System Hibernate	OSC	
A9-AF	Reserved		
B0	System Display Invert	OSC	4.10
B1	System Display Internal	OSC	
B2	System Display External	OSC	
B3	System Display Both	OSC	
B4	System Display Dual	OSC	
B5	System Display Toggle Int/Ext	OSC	
B6	System Display Swap Primary/Secondary	OSC	
B7	System Display LCD Autoscale	OSC	
B8-FFFF	Reserved		

用法（ Usage ）定义了各种各样设备特性，对于 Usage Page 的每一项都定义了常用的各种用法。

用法说明了 3 种信息，即控制、集合和数据。控制说明设备的状态，如 on/off 、Enable/Disable 等。集合说明控制和数据的组合关系。

上表中的用法类型（ Usage Type ）描述了应用程序如何处理由 Main 类型项目生成的数据，具体的定义和详细说明请参阅 HID Usage Tables 。

（4）Report ID 项目

Report ID 放在信息包中报表数据之前，设备可以支持多个相同类型的报表，每一个报表包含不同的数据与其特有的 ID。

在报表描述符中， Report ID 项目作用于其后续所有的项目， 直到遇到下一个 Report ID 为止。如果报表描述符中没有 Report ID 项目，默认的 ID 值是 0，描述符不能定义一个为 0 的 Report ID ，输入报表、输出报表与特征报表可以分享同一个 Report ID 。

在 Set_Report 和 Get_Report 请求传输中， 主机在设置事务的 wValue 字段的低字节中指定一个 Report ID 。在中断传输中如果接口支持一个以上的 Report ID ， Report ID 必须是传送报表中的第一个字节。如果接口只支持数值为 0 的默认 Report ID ，此 Report ID 不应该在中断传输中随着报表一起传送。

（5）Logical Minimum 和 Logical Maximum 项目

Logical Minimum 与 Logical Maximum 项目定义报表的变量 （Variable ）或阵列（Array ）数据的限制范围，此限制范围以逻辑单位来表示。例如设备报表的一个电流值读数是 500mA ，而一个单位是 2mA，则 Logical Maximum 值等于 250 。

负数值以 2 的补码来表示。 如果 Logical Minimum 与 Logical Maximum 都是正数， 就不需要有正负号位。不管 Logical Minimum 与 Logical Maximum 是以有正负号或是无正负号的数值来表示，设备都可以正确地传输数据。数据的接收者必须知道数据是否可以是负值。

（5）Physical Minimum 和 Physical Maximum 项目

Physical Minimum 和 Physical Maximum 项目定义数值的限制范围，该限制范围使用 Unit 项目定义的单位来表示。 上例中设备报表的一个电流值读数是 500mA ， 单位是 2mA ，Logical Maximum 值等于 250，而 Physical Maximum 值是 500。

Logical Minimum 与 Logical Maximum 值说明了设备返回数值的边界，可以根据 Physical Minimum 和 Physical Maximum 值对数据进行偏移和比例变换。

（6）Unit Exponent 项目

Unit Exponent 项目定义了在使用逻辑范围和实际范围将设备的返回数值转换成实际数值时，使用 10 的多少次方对数值进行定标。 Unit Exponent 的值的编码为 4 位补码，代表 10 的指数范围是 -8~+7 。

表8-8 Unit Exponent 数值表

代码	00h	01h	02h	03h	04h	05h	06h	07h	08h	09h	0Ah	0Bh	0Ch	0Dh	0Eh	0FH
数值	0	1	2	3	4	5	6	7	-8	-7	-6	-5	-4	-3	-2	-1

根据以上 5 个项目的值可以换算出报表传送数据 （逻辑数据） 与物理数据的转换关系。

物理数据值 = 逻辑数据值÷分辨率

分辨率 = （LogicalMaximum - LogicalMinimum ）÷
（（PhysicalMaximum - PhysicalMinimum ）× $10^{UnitExponent}$ ）

（7）Unit 项目

Unit 项目指定报表数据在使用 Physical 与 Unit Exponent 项目转换后使用什么度量单位，以及单位的幂指数值。 Unit 的数值部分可以长达 4 字节，按照 4 位为一段分段，可以分为 8 个半字节段，由高到低分别为半字节 7、半字节 6、…、半字节 0。每一个半字节对应不同的基本单位， 其数值表示单位的指数值， 采用 4 位 2 的补码表示， 取值范围是 -8~+7 之间。

从半字节 0~6 由下表给出了具体的定义， 其中半字节 0 表示测量系统， 半字节 7 保留。例如在半字节 0 数值为 1（表示采用线性公制测量系统） 的条件下， 半字节 1 表示长度 （单位为厘米），如果其数值为 1 表示厘米，数值为 2 表示（厘米） ^{2}，成为面积单位。半字节 3 表示时间（单位为秒） ，如果其数值为 -2，表示（秒） $^{-2}$。

表 8-9 Unit 单位的定义

半字节序号	测量项目	数值含义				
		0	1	2	3	4
0	测量系统	无	线性、公制	角度、公制	线性、英制	角度、英制
1	长度		厘米	半径	英寸	度
2	质量		克		石拉（ slug ）	
3	时间		秒			
4	温度		开式度（ Kelvin ）		华式度	
5	电流		安培			
6	亮度		烛光			
7	保留					

虽然表中只是定义了有限的基本单位，但可以通过这些基本单位的组合派生出大多数其它的常用单位。

例如报表使用一个字节传送一个从 -20 到 110 华氏度温度值，可以定义以下报表描述项目：

```
Logical Minimum = -128
Logical Maximum = 127
Physical Minimum = -20
Physical Maximum = 110
Unit Exponent = 0
Unit = 30003h
```

Unit 的半字节 0=3 选择英制线性测量系统，半字节 4=3 选择华氏温度单位。

130 （110+20 ）华氏度的数值范围线性分布到了 256 和有效数值区域，每一位相当于 0.51 华氏度，这样就提高了分辨率。

（8）Report Size 和 Report Count 项目

Report Size 项目指定 Input 、 Output 与 Feature 项目字段的大小，以位为单位。

Report Count 项目指定 Input 、 Output 与 Feature 项目包含的字段数目。

例如两个 8 位的字段， Report Size 等于 8，而 Report Count 等于 2。8 个 1 位的字段，Report Size 等于 1，而 Report Count 等于 8。

Input 、Output 与 Feature 项目报表可以有多个项目， 每一个项目可以有自己的 Report Size 和 Report Count 项目。

（9）Push 和 Pop 项目

Push 项目将一个 Global 项目状态表格的副本压入 CPU 的堆栈内。 Global 项目状态表格包含所有之前定义的 Gobal 项目的目前设置。

Pop 项目恢复之前压入堆栈的 Global 项目状态的储存状态。

（10）Usage、Usage Minimum 和 Usage Maximum 项目

这 3 个项目输入 Local 类型项目。

Usage 项目和 Global 类型的 Usage Page 项目协同描述项目或集合的功能。

一个报表可以指定一个 Usage 给许多个控制，或是指定不同的 Usage 给每一个控制。如果一个报表项目之前有一个 Usage，此 Usage 应用到该项目的所有控制。如果一个报表项目之前有一个以上的 Usage，每一个 Usage 应用到一个控制，Usage 与控制是按顺序结合的。

例如下面报表描述符的一个局部，报表含有 2 个输入字节，第一个字节的用法是 x，第 2 个字节是 y。

```
Report Size (8)
Report Count (2)
Usage (x)
Usage (y)
Input (Data, Variable, Absolute)
```

如果一个报表项目之前有一个以上的 Usage，而且控制的数目多于 Usage 的数目，每一个 Usage 与一个控制对应，最后一个 Usage 则应用到所有剩余的控制。

例如在下面报表包含 16 个字节输入数据，第一个字节对应用法 x，第 2 个字节对应用法 y，剩余的 14 个字节对应厂商定义的用法。

```
Usage (x)
Usage (y)
Usage (Vendor defined)
Report Size (8)
Report Count (16)
Input (Data, Variable, Absolute)
```

Usage Minimum 和 Usage Maximum 可以指定一个 Usage 给多个控制或是数组项目。将从 Usage Minimum 到 Usgae Maximun 定义的用法顺序对应到多个控制中。

例如在一个键盘描述符中定义的标准键盘的左、右修饰键的输入项目中，使用一个字节的 8 位分别输入键盘的左、右 Ctrl 键、Shift 键、Alt 键和 GUI 键，从 HID Usage tables 文档中的第 10 节可以查到关于键盘用法的定义，其中上述 8 个修饰键的用法定义值为 224 到 231。以下是报表描述符的修饰键部分描述。

```
Usage Page (1)                  ; 1 = Generic Desktop Controls
Usage (6)                       ; 6 = Keyboard
Collection (1)                  ; 1 = Application
Usage Page (7)                  ; 7 = Keyboard/Keypad
Usage Minimum (224)
Usage Maximum (231)
Logical Minimum (0)
Logical Maximum (1)
Report Size (1)
Report Count (8)
Input (Data, Variable, Absolute)
……
```

8.3 USB 接口的键盘描述符范例

下面作为一个例子，介绍一个 USB 接口的 101 键盘的全部描述符。该键盘固件部分由一个微处理器实现全部控制功能，下面列出的代码为微处理器汇编实现描述符定义。

8.3.1 设备的描述符

设备描述符的代码如下。

```
;=========================================
; Device descriptor              设备描述符
;=========================================
DEVICE_DESC_DATA:
DB 0x12                 ; bLength = 18        ，该描述符长度为 18 字节
DB 0x01                 ; bDescriptorType = 01            ，表明是设备描述符
DB 0x10, 0x01           ; bcdUSB   ，USB设备版本号 =1.1
DB 0x00                 ; DeviceClass       ，设备类码，  HID 设备为 0，类别在接口描述符中定义
DB 0X00                 ; DeviceSubClass       ，设备子类码，  DeviceClass    为0时该字段必须为 0
DB 0X00                 ; bDevicePortocol        ，协议码，  DeviceClass    为0时该字段必须为 0
DB 0x08                 ; bMaxPacketSize0        ，端点 0的最大包尺寸
DB 0xFF, 0xFF           ; bVendor    ，厂商 ID ，由 USB实现者论坛确定的
DB 0x01, 0x00           ; bProduct      ，产品 ID
DB 0x00, 0x01           ; bcdDevice      ，设备版本号为 1.00
DB 0x04                 ; iManufacturer         ，厂商字符串的索引值，见字符串描述符
DB 0x0E                 ; iProduct      ，产品字符串的索引值，见字符串描述符
DB 0x30                 ; iSerialNumber         ，产品序列号字符串的索引值，见字符串描述符
DB 0X01                 ; bNumConfigurations          ，配置数目只有 1个
```

8.3.2 配置描述符

配置描述符的代码如下。

```
;=========================================
; Configuration descriptor                    配置描述符
;=========================================
CONFIG_DESC_DATA:
DB 0x09                 ; bLength=9      ，该描述符长度为 9字节
DB 0x02                 ; bDescriptorType = 02            ，表明是配置描述符
DB 0x3B, 0x00           ; wTotalLength = 59          ，配置、接口、端点和 HID 描述符的总和字节数
DB 0x02                 ; bNumInterfaces = 2          ，本配置支持的接口数目为 2个
DB 0x01                 ;  bConfigurationValue          = 1  ，本配置描述符的标识符
DB 0x00                 ; iCongfiguration = 0              ，配置描述符说明字符串的索引值，      0表示无
DB 0XA0                 ; bmAttributes        ，电源及唤醒设置，  USB1.1 版中 D7=1 ， D6=0 表示总线供电
                        ;            D5=1            表示支持远程唤醒
DB 0X32                 ; MaxPower = 50       ，本设备最大耗电为 50X2mA=100mA
```

需要说明的是 wTotalLength 的值，该数值为配置描述符长度（ 9）加上后续的键盘的接口描述符长度（ 9）、端点描述符长度（ 7）、HID 描述符长度（ 9），以及该配置下的鼠标的接口描述符长度（ 9）、端点描述符长度（ 7）、HID 描述符长度（ 9），共 59 个字节。关于鼠标的相关描述符在下面的叙述中省略了。

8.3.3 接口描述符

接口描述符的代码如下。

```
;=======================================
; Interface descriptor                  接口描述符
;=======================================
InterfaceDescriptor0:
DB 0x09                 ; bLength = 9        ，该描述符长度
DB 0x04                 ; bDescriptorType = 4          ，表明是接口描述符
DB 0x00                 ; bInterfaceNumber = 0          ，此接口的识别标识符
DB 0x00                 ; bAlternateSetting = 0            ，表示此接口无替代设置值
DB 0x01                 ; bNumEndpoints = 1        ，本接口的端点数目，    HID 设备使用端点  1
DB 0x03                 ; bInterfaceClass = 3          ，表示该设备是  HID 类别
DB 0x01                 ; bInterfaceSubClass = 1           ，表示支持启动接口
DB 0x01                 ; bInterfaceProtocol = 1           ，表示支持键盘协议
DB 0x00                 ; iInterface = 0          ，接口描述符说明字符串的索引值，      0表示无字符串
```

8.3.4 HID 描述符

HID 描述符的代码如下。

```
;=======================================
; HID descriptor    HID              描述符
;=======================================
HIDDescriptor0:
DB 0x09                 ; bLength = 9        ，该描述符长度
DB 0x21                 ; bDescriptorType = 21h            ，表明是 HID 描述符
DB 0x00, 0x01           ; bcdHID = 0100        ，HID 规范版本为  1.00
DB 0x00                 ; bCountryCode = 0          ，硬件设备所在国家的国家代码，      0表示未指明
DB 0x01                 ; nNumDescriptors = 1          ，表示支持的描述符有   1个，即一个报表描述符
DB 0x22                 ; bDescriptorType = 22h            ，描述符类别，表示支持的描述符是报表描述符
DB 0x3F, 0x00           ; wDescriptorLength = 63           ，表示支持的报表描述符的长度
```

8.3.5 端点描述符

端点描述符的代码如下。

```
;=======================================
; EndPoint descriptor                   端点描述符
;=======================================
EndpointDescriptor0:
DB 0x07                 ; bLength = 7        ，该描述符长度
DB 0x05                 ; bDescriptorType = 5          ，表明是端点描述符
DB 0x81                 ; bEndpointAddress = 1000 0001b                ，表示 1号输入端点
DB 0X03                 ; bmAttributes = 00000011b                ，表示中断类型端点
DB 0x08, 0x00           ;   wMaxPacketSize = 8   ，端点发送和接收的最大包尺寸为    8
DB 10                   ; bInterval = 10          ，表示中断端点轮询时间间隔为    10ms
```

8.3.6 字符串描述符

字符串描述符的代码如下。

```
;=======================================
; String descriptor                    字符串描述符
;=======================================
StringDescriptor0:
DB 0x04                 ; bLength = 4        ，字符串描述符  0的长度为 4
DB 0x03                 ; bDescriptorType = 3          ，表明是字符串描述符
DB 0x09, 0x00           ; wLANGID = 0009h      ，表明是英语
DB 0x0A                 ; bLength = 10        ，字符串描述符的长度为   10
DB 0x03                 ; bDescriptorType = 3          ，表明是字符串描述符
DB 0x41, 0x00, 0x43, 0x00, 0x4D, 0x00, 0x45, 0x00
                        ; bString =          “ACME“，Unicode   编码的字符串
DB 0x22                 ; bLength = 34        ，字符串描述符的长度为   34
DB 0x03                 ; bDescriptorType = 3          ，表明是字符串描述符
DB 0x4C, 0x00, 0x6F, 0x00, 0x63, 0x00, 0x61, 0x00
DB 0x74, 0x00, 0x6F, 0x00, 0x72, 0x00, 0x20, 0x00
DB 0x4B, 0x00, 0x65, 0x00, 0x79, 0x00, 0x62, 0x00
DB 0x6F, 0x00, 0x61, 0x00, 0x72, 0x00, 0x64, 0x00
                        ; bString =          “Locator Keyboard        “，Unicode   编码的字符串
DB 0x0E                 ; bLength = 14        ，字符串描述符的长度为   14
DB 0x03                 ; bDescriptorType = 3          ，表明是字符串描述符
DB 0x41, 0x00, 0x42, 0x00, 0x43, 0x00, 0x21, 0x00
DB 0x22, 0x00, 0x23, 0x00
                        ; bString =          “ABC123 “，Unicode   编码的字符串
```

8.3.7 报表描述符

报表描述符的代码如下。

```
;=======================================
;HID Reports Descriptor               报表描述符
;=======================================
DB 0x05, 1              ; Usage Page (1: Generic Desktop)
DB 0x09, 6              ; Usage (6: Keyboard)            表示报表定义的是  HID 键盘
DB 0xA1, 1              ; Collection (1: Application) ====================          集合开始
                        ;
                        ;       以下定义了键盘的修饰键输入报表，共有      8个键，组成一个字节
                        ;       用法见 HID Usage Table       中的第 10 节中的键盘用法定义
DB 0x05, 7              ;       Usage page (7: Key Codes)
DB 0x19, 224            ;       Usage Minimum (224)
DB 0x29, 231            ;       Usage Maximum (231)
DB 0x15, 0              ;       Logical Minimum (0)
DB 0x25, 1              ;       Logical Maximum (1)
DB 0x75, 1              ;       Report Size (1)
DB 0x95, 8              ;       Report Count (8)
DB 0x81, 2              ;       Input (Data,Variable,Absolute)
                        ;
                        ;       以下定义了一个保留字节的输入报表
DB 0x95, 1              ;       Report Count (1)
DB 0x75, 8              ;       Report Size (8),
DB 0x81, 1              ;       Input (Constant) = Reserved Byte
                        ;
                        ;       以下定义了键盘的  LED指示灯输出报表项目，共有    5个指示灯
                        ;       用法见 HID Usage Table       中的第 11 节中的 LED用法定义
DB 0x95, 5              ;       Report Count (5)
```

```
DB 0x75, 1          ;       Report Size (1)
DB 0x05, 8          ;       Usage Page (Page# for LEDs)
DB 0x19, 1          ;       Usage Minimum (1)
DB 0x29, 5          ;       Usage Maximum (5)
DB 0x91, 2          ;       Output (Data, Variable, Absolute)
                    ;
                    ;       以下定义了 3个填充位，与前面的 5 个LED指示灯数据组成一个完整的字节
DB 0x95, 1          ;       Report Count (1)
DB 0x75, 3          ;       Report Size (3)
DB 0x91, 1          ;       Output (Constant)
                    ;
                    ;       以下定义了键盘的按键值输入报表项目，  共6 个字节，存放键编号 （0~101 ）
                    ;       用法见 HID Usage Table  中的第 10 节中的键盘用法定义
                     ;      这样的设计可以允许一次输入 6个按键的键值
DB 0x95, 6          ;       Report Count (6)
DB 0x75, 8          ;       Report Size (8)
DB 0x15, 0          ;       Logical Minimum (0)
DB 0x25, 101        ;       Logical Maximum (101)
DB 0x05, 7          ;       Usage Page (7: Key Codes)
DB 0x19, 0          ;       Usage Minimum (0)
DB 0x29, 101        ;       Usage Maximum (101)
DB 0x81, 0          ;       Input (Data, Array)

DB 0xC0             ; End_Collection ================================        集合结束
```

通过上面的报表描述符的定义，确定了键盘的输入报表和输出报表的数据格式。其中输入报表共 8 个字节，输出报表只有 1 个字节。

图 8-4 键盘的输入报表格式

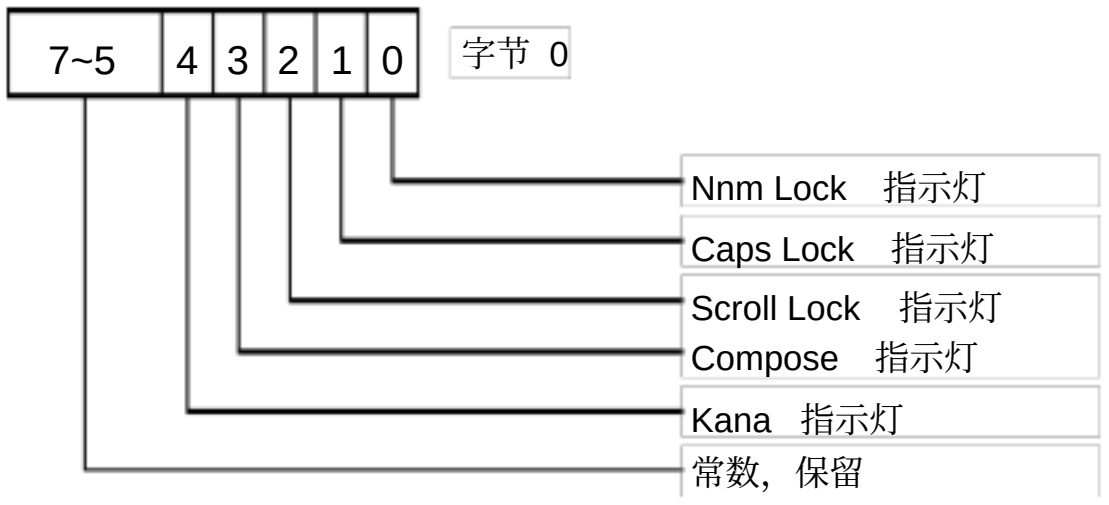

图 8-5 键盘的输出报表格式

8.4 HID 的特定请求

除了 USB 设备的 11 个标准请求外， HID 规范另外还定义了 6 个 HID 特定控制请求。所有的 HID 设备都必须支持 Get_Report 请求，同时支持启动的设备必须支持 Get_Protocol 请求和 Set_Protocol 请求，其他的请求是可选择的。如果设备没有中断输出端点，此设备需要支持 Get_Report 请求来从主机读取数据。

在控制传输的设置阶段的数据包中的 8 个字节中的第一字节 bmRequestType 的编码含义参阅第 6 章中的 USB 标准请求。第 2 个字节 bRequest 定义请求的内容。 wValue 因请求的不同而不同。 wIndex 指明 HID 所在的接口。

表8-10 HID特定的请求

bmRequestType	bRequest （值）	wValue	wIndex	wLength	数据阶段
1 01 00001b	Get_Report (1)	报表类型，报表 ID	接口	报表长度	报表
0 01 00001b	Set_Report (9)	报表类型，报表 ID	接口	报表长度	报表
1 01 00001b	Get_Idle (2)	0，报表 ID	接口	1	闲置时间
0 01 00001b	Set_Idle (10)	闲置时间，报表 ID	接口	0	无
1 01 00001b	Get_Protocol (3)	0	接口	1	0: 启动协议 1: 报表协议
0 01 00001b	Set_Protocol (11)	0: 启动协议 1: 报表协议	接口	0	无

8.4.1 Get_Report 请求

Get_Report 的作用是启用主机使用控制传输，来从设备读取数据。

在使用时 wValue 字段的高字节是报表类型， 1 表示 Input 报表， 2 表示 Output 报表，3 表示 Feature 报表。 wValue 的低字节是报表的 Report ID ，如果没有定义 Report ID ，该字节为设 0。

在携带请求的控制传输的数据阶段， HID 设备回传指定的报表内容。

HID 规范不建议使用该请求获得未经定时的数据，这样的数据建议使用中断输入管道获得。

该请求用来取得在主机初始化设备时的特征项目状态和其他信息。使用开机协议的主机可以使用此请求来获得按键或鼠标数据。

8.4.2 Set_Report 请求

Set_Report 请求的参数含义和 Get_Report 一样，但 Set_Report 请求的数据方向与 Get_Report 相反，在后面的数据阶段，主机传送报表到 HID 设备，这样的输出报表可以用于复位设备的控制，复位产生的效果取决于对应的控制的类型是相对（ Reletive ）的还是绝对（ Absolute ）的。

8.4.3 Set_Idle 请求

Set_Idle 请求的作用是静默一个在中断输入管道的特定的报表，直到一个发生一个相关的事件或过去了规定的时间，当数据从上一个报表后没有改变时，可以通过限制中断输入端点的报表频率来节省传输带宽。 HID 设备不是必需支持此请求。

wValue 字段的高字节是设置的闲置时间，是报表之间的最大间隔时间。该字节为 0

表示闲置时间为无限长，在这种情况下，设备只有在报表数据有改变时才传送报表，否则设备传回一个 NAK 。

wValue 字段的低字节指示此请求应用的报表的 Report ID。如果低字节是 0，此请求应用到设备的所有输入报表。

闲置时间以 4ms 为单位， 范围在 4ms ～1020ms 之间。如果报表的数据自从上一次报表后有改变，或是接收到一个请求，设备会传送一个报表。

如果报表的数据没有改变，而且从上一次报表后过去的时间自尚未达到规定的闲置时间，设备会传回一个 NAK 。如果报表的数据没有改变， 而且持续时间已经达到的闲置时间，设备会传送一个报表。

闲置时间设置为 0 表示无限长的闲置时间，设备只有在报表的数据有改变时才会传送一个报表，对于其他的中断输入请求则是传回 NAK 。

在检测 HID 设备时， Wndows 的 HID 驱动程序会试图将闲置时间设置成 0。如果 HID 设备不支持此请求，主机会收到传回的 Stall。

8.4.4 Get_Idle 请求

Get_Idle 请求的作用是过的设备的当前闲置时间，在数据阶段， HID 设备回传一个字节的闲置时间值。

8.4.5 Get_Protocol 请求

Get_Protocol 请求的作用是主机获取设备目前作用的是启动协议还是报表协议。

在数据阶段中设备回传的 1 个字节信息包中的数据值为 0 表示启动协议， 为 1 表示报表协议。

启动设备必需支持该请求。

8.4.6 Set_Protocol 请求

Set_Protocol 的作用是主机指定设备使用启动协议或报表协议。

在数据阶段中主机传送的 1 个字节信息包中的数据值为 0 表示指定启动协议， 为 1 表示指定报表协议。

启动设备必需支持该请求。

8.5 HID 程序设计

HID 设备是 Windows 系统提供了完善支持的一类，应用程序可以通过标准的 API 函数调用，实现与 HID 设备的通信。 Windows 系统提供了几千个 API 函数，作为应用程序与操作系统的接口，与 HID 相关的 API 函数被封装在 hid.dll 、 setupapi.dll 和 kernal32.dll 文件中。

8.5.1 HID 访问使用的 API 函数

文件 hid.dll 中提供了很多个 API ，因为与 HID 设备通信有一定的复杂性。首先，在应用程序与 HID 传输数据之前， 应用程序必须先识别该设备并且读取它的报表信息，这些动作需要调用多个 API 函数才可以实现。 应用程序需要寻找连接到系统上的是哪些 HID 设备，然后读取每个设备的信息直到查找到所需的属性。如果是客户化的设备，应用程序可以寻找特定的厂商与产品 ID，或者应用程序可以寻找特定类型的设备，例如键盘或鼠标。

表8-11 用于 HID设备的 API 函数

用于了解 HID 设备情况的 API 函数（ hid.dll ）	
HidD_GetAttributes	请求获得 HID 设备的厂商 ID、产品 ID 和版本号
HidD_FreePreparsedData	释放函数 HidD_GetPreparsedData 所使用的资源
HidD_GetHidGuid	请求获得 HID 设备的 GUID
HidD_GetIndexString	请求获得由索引识别的字符串
HidD_GetManufactureString	请求获得设备制造商字符串
HidD_GetPhysicalDescriptor	请求获得设备实体字符串
HidD_GetPreparsedData	请求获得与设备能力信息相关的缓冲区的代号
HidD_GetProductString	请求获得产品字符串
HidD_GetSerialNumberString	请求获得产品序列号字符串
HidP_GetButtonCaps	请求获得 HID 报表中所有按钮的能力
HidP_GetCaps	请求获得用于描述设备能力的结构的指针
HidP_GetLinkCollectionNotes	请求获得描述在顶层集合中的连接集合（ Link Collection ）关系的结构的数组
HidP_GetSpecificButtonCaps	请求获得报表中按钮的能力，该请求可以设定一个 Usage Page 、Usage 或是 Link Collection
HidP_GetSpecificValueCaps	请求获得报表中数值的能力，该请求可以设定一个 Usage Page 、Usage 或是 Link Collection
HidP_GetValueCaps	请求获得 HID 报表中所有数值的能力
HidP_MaxUsageListLength	请求获得 HID 报表中可以回传的按钮的最大数目， 该请求可以设定一个 Usage Page
HidP_UsageListDifference	比较两个按钮列表， 并且求出在一个列表中设定而在另一个列表中没有设定的按钮
用于从设备读取、向设备传送报表的 API 函数（ hid.dll ）	
HidD_GetFeature	从设备读取一个特征报表
HidD_SetFeature	向设备传送一个特征报表
HidP_GetButtons	从设备读取包含每个按下的按钮的用法（ Usage ）的缓冲区的指针，该请求可以设定一个 Usage Page
HidP_GetButtonEx	从设备读取包含每个按下的按钮的 Usage 和 Usage Page 的缓冲区的指针
HidP_GetScaledUsageValue	从设备读取一个已经经过比例因子调整的有符号数值
HidP_GetUsageValue	从设备读取一个指向数值的指针
HidP_GetUsageValueArray	从设备读取包含多个数据项的 Usage 的数据
HidP_SetButtons	向设备传送设置按钮的数据
HidP_SetScaledUsageValue	将一个实际数值转换成设备使用的逻辑数值， 并将其插入到报表中
HidP_SetUsageValue	向设备传送数据
HidP_SetUsageValueArray	向设备传送包含多个数据项的 Usage 的数据
HidD_FlushQueue	清空输入缓冲区
HidD_GetNumInputBuffer	获得驱动程序用于存储输入报表的环形缓冲区的大小， 默认值是 8
HidD_SetNumInputBuffer	设置驱动程序用于存储输入报表的环形缓冲区的大小
用于查找和识别设备的 API 函数（ setupapi.dll ）	
SetupDiGetClassDevs	获得 HID 的信息， 针对已安装的设备， 回传一个指向其信息集的代码
SetupDiEnumDeviceInterfaces	请求获得设备信息群内的一个设备的信息
SetupDiGetDeviceInterfaceDetail	请求获得设备的路径
SetupDiDestroyDeviceInfoList	释放 SetupDiGetClassDevs 使用的资源
用于打开、关闭设备和实现数据传送的 API 函数（ kernal32.dll ）	

CreatFile	取得设备的路径后，调用该函数获得设备代号
WriteFile	向设备传送输出报表
ReadFile	从设备读取输入报表
CloseHandle	关闭设备，释放 CreateFile 所使用的资源

8.5.2 查找 HID 的过程

在实现 HID 的访问之前，首先要查找指定（根据设备的厂商 ID、产品 ID 和产品序列号）的 HID。查找指定设备的过程如下：

? 调用函数 HidD_GetHidGuid 获得 USB 设备的 GUID ；

? 调用函数 SetupDiGetClassDevs ，获得一个包含全部 HID 信息的结构数组的指针，下面根据此数组逐项查找指定的 HID ；

? 调用函数 SetupDiEnumDeviceInterfaces ，填写 SP_DEVICE_INTERFACE_DATA 结构的数据项，该结构用于识别一个 HID 设备接口；

? 调用函数 SetupDiGetDeviceInterfaceDetail ，获得一个指向该设备的路径名；

? 调用函数 CreateFile ，获得设备句柄；

? 调用函数 HidD_GetAttributes ，填写 HIDD_ATTRIBUTES 结构的数据项，该结构包含设备的厂商 ID、产品 ID 和产品序列号，比照这些数值确定该设备是否是查找的设备。

查找 HID 的流程如下图。

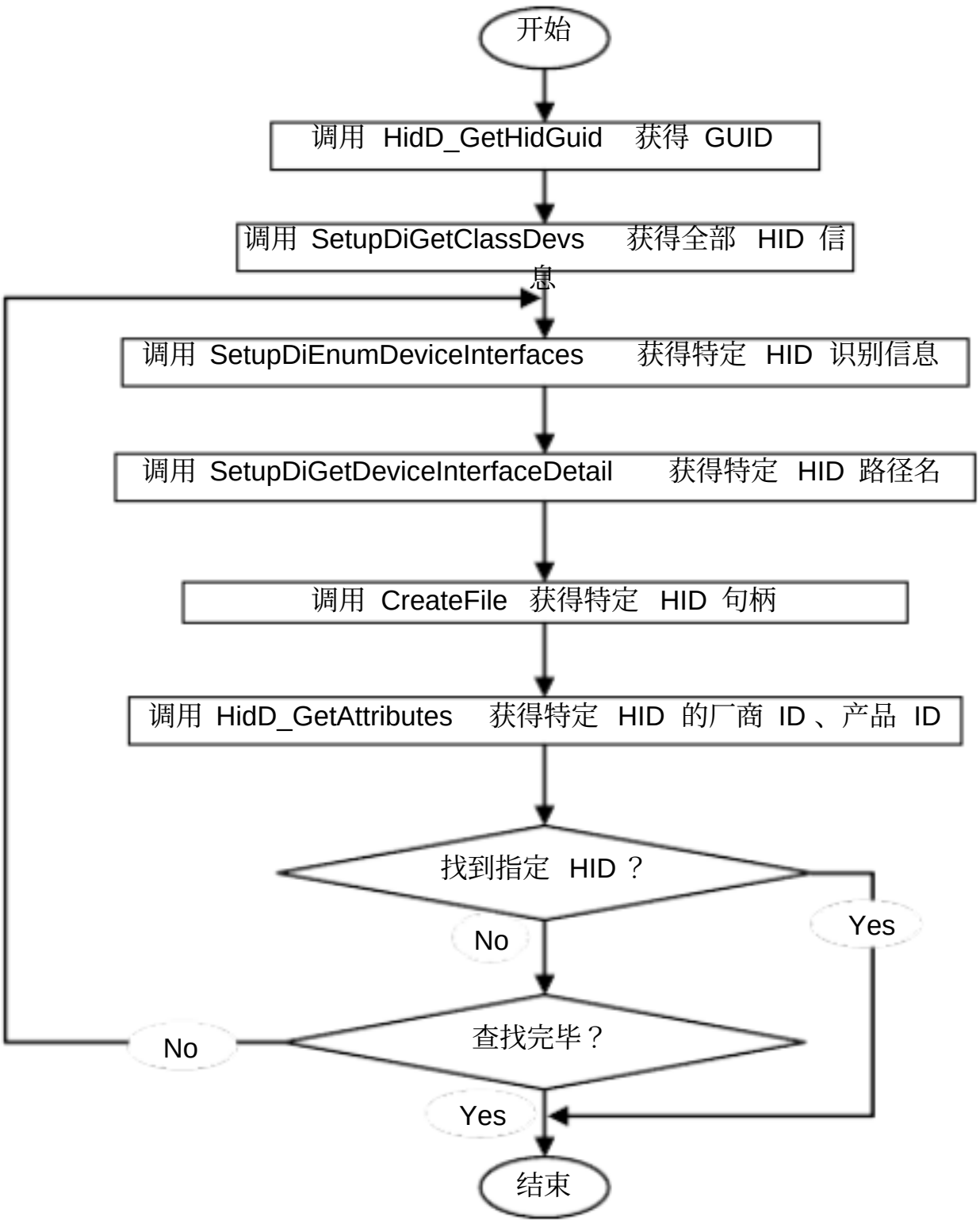

图 8-6 查找设备的流程

下面介绍 VB 实现的查找过程。

(1) 获得 GUID

应用程序要与 HID 设备通信之前，必须先获得 HID 类别的 GUID （Globally Unique Indentifer ）。GUID 是一个 128 位的数值，每个对象都有惟一的 GUID。HID 类别的 GUID 包含在 hidclass.h 文档内，可以接引用，或是使用 HidD_GetHidGuid 函数来取得 HID 类别的 GUID 。

```
, 函数声明
Public Declare Function HidD_GetHidGuid Lib "hid.dll" (ByRef HidGuid As GUID
) As Long

, GUID 结构说明
Public Type GUID
    Data1 As Long
    Data2 As Integer
    Data3 As Integer
```

```
    Data4(7) As Byte
End Type

, 变量说明
Dim HidGuid as GUID

, 调用
Call HidD_GetHidGuid(HidGuid)
```

(2) 获得 HID 的结构数组

得到 GUID 后调用 SetupDiGetClassDevs 函数传回所有已经连接并且检测过的 HID 包含其信息的结构数组的地址。

```
, 函数声明
Public Declare Function SetupDiGetClassDevs Lib "setupapi.dll" Alias
"SetupDiGetClassDevsA"( _
        ByRef ClassGuid As GUID, _
        ByVal Enumerator As String, _
        ByVal hwndParent As Long, _
        ByVal Flags As Long) _
As Long
, 常量说明
Public Const DIGCF_PRESENT = &H2
Public Const DIGCF_DEVICEINTERFACE = &H10

, 调用
hDevInfoSet = SetupDiGetClassDevs(
        HidGuid, _                    , 通过 HidD_GetHidGuid 函数获得 GUID
        vbNullString, _
        0, _
        (DIFCG_PRESENT or DIFCG_DEVICEINTERFACE) _
)
```

该函数的 ClassGuid 参数值为通过 HidD_GetHidGuid 函数获得 GUID。Enumerator 与 hwndParent 参数没有用到, Flags 参数是两个定义在 setupapi.h 文档内的系统常数。代码中的 Flags 常数告诉 SetupDiGetClassDevs 函数只寻找目前存在（连接并且检测过）的设备接口，并且是 HID 类别的成员。

返回值 hDevInfoSet 是包含所有连接并且检测到的全部 HID 的信息的结构数组的地址。在程序中并不需要访问 hDevInfoSet 的元素，只需要将 hDevInfSet 值传给 SetupDiEnumDeviceInterfaces 函数即可。

当程序不再需要 hDevInfoSet 指向的数组时，应该调用 SetupDiDestroyDeviceInfo List 函数来释放资源。

下面在 hDevInfoSet 指向的结构数组中查找。

(3) 识别 HID 接口

接下来调用 SetupDiEnumDeviceInterfaces ，填写 SP_DEVICE_INTERFACE_DATA 结构的数据项，该结构用于识别一个 HID 设备接口。

```
, 函数声明
Public Declare Function SetupDiEnumDeviceInterfaces Lib "setupapi.dll" (
   ByVal DeviceInfoSet As Long, _
```

```
    ByVal DeviceInfoData As Long, _
    ByRef InterfaceClassGuid As GUID, _
    ByVal MemberIndex As Long, _
    ByRef DeviceInterfaceData As SP_DEVICE_INTERFACE_DATA _
)As Long

, 结构定义
Public Type SP_DEVICE_INTERFACE_DATA
  cbSize As Long
  InterfaceClassGuid As GUID
  Flags As Long
  Reserved As Long
End Type

, 变量说明
Dim Result as Long
Dim MemberIndex as Long
Dim MyDeviceInterfaceData as SP_DEVICE_INTERFACE_DATA

, 调用
MyDeviceInterfaceData.cbSize = LenB(MyDeviceInterfaceData)
MemberIndex = 0
Result = SetupDiEnumDeviceInterfaces(
        DeviceInfoSet, _                    ,SetupDiGetClassDevs        的返回值
        0, _
        HidGuid, _                          , 通过 HidD_GetHidGuid      函数获得的 GUID
        MemberIndex, _                      , 第一次调用设置为 0
        MyDeviceInterfaceData _
)
```

参数 cbSize 是 SP_DEVICE_INTERFACE_DATA 结构的大小，以字节为单位。在调用 SetupDiEnumDeviceInterfaces 函数之前， cbSize 必须储存在结构内来当做传递的参数。

参数 HidGuid 和 DeviceInfoSet 是函数之前的传回值。

DeviceInfoData 是 SP_DEVICE_INTERFACE_DATA 结构的指针， 用来限制检测特定设备的接口。 MemberIndex 是 DeviceInfoSet 数组的索引值，在遍历整个数组的循环中 MemberIndex 递增。 MyDeviceInterfaceData 是回传的结构，用来识别 HID 的一个接口。

(4) 获得设备路径

下面通过调用 SetupDiGetDeviceInterfaceDetail 函数用来获得另外一个结构： SP_DEVICE_INTERFACE_DETAIL_DATA 。此结构与前一个函数 SetupDiEnumDeviceInterfaces 所识别的设备接口有关。 结构的 DevicePath 成员是一个设备路径，应用程序可以用此路径来实现与该设备的通信。

```
, 函数声明：
Public Declare Function SetupDiGetDeviceInterfaceDetail Lib "setupapi.dll" _
  Alias "SetupDiGetDeviceInterfaceDetailA" (_
  ByVal DeviceInfoSet As Long, _
  ByRef DeviceInterfaceData As SP_DEVICE_INTERFACE_DATA, _
  ByVal DeviceInterfaceDetailData As Long, _
  ByVal DeviceInterfaceDetailDataSize As Long, _
  ByRef RequiredSize As Long, _
  ByVal DeviceInfoData As Long _
) As Long
```

```
, 结构声明
Public Type SP_DEVICE_INTERFACE_DETAIL_DATA
   cbSize As Long
   DevicePath As Byte
End Type

, 变量定义
Dim Needed as Long, DetailData as long
Dim MyDeviceInterfaceDetailData As SP_DEVICE_INTERFACE_DETAIL_DATA
Dim DetailDataBuffer() as Byte
Dim DevicePathName As String

, 调用
MyDeviceInterfaceData.cbSize = LenB(MyDeviceInterfaceData)

Result = SetupDiGetDeviceInterfaceDetailA(
        DeviceInfoSet, _
        DeviceInterfaceData, _
        0, _
        0, _
        Needed, _
        0 _
)

DetailData = needed
MyDeviceInterfaceDetailData.cbSize = Len(MyDeviceInterfaceDetailData)
ReDim DetailDataBuffer(Needed)

Call RtlMoveMemory(DetailDataBuffer(0), MyDeviceInterfaceDetailData, 4)
Result = SetupDiGetDeviceInterfaceDetailA(
        DeviceInfoSet, _
        DeviceInterfaceData, _
        VarPtr(DetailDataBuffer(0)), _
        DetailData, _
        Needed, _
        0 _
)

, 转换成字符串，再转换成 Unicode 并从中删除 4 个字符得到设备路径
DevicePathName = Cstr(DetailDataBuffer())
DevicePathName = StrCovn(DevicePathName, vbUnicode)
DevicePathName = Right$(DevicePathName, Len(DevicePathName)-4)
```

DeviceInterfaceDetailDataSize 包含 DeviceInterfaceData 结构的大小，以字节为单位。但是，在调用 SetupDiGetDeviceInterfaceDetail 函数之前无法知道此数值的大小，如果没有 DeviceInterfaceDetailDataSize 函数，SetupDiGetDeviceInterfaceDetail 函数不会传回所需的结构。

这个问题的解决方式是两次调用 SetupDiGetDeviceInterfaceDetail 函数。第一次调用时 GetLastError 函数会传回错误信息，即： The data erea passed to a system call is too small，但是 RequireSize 参数会包含正确的 DeviceInterfaceDetailDataSize 数值。再次调用时传递此传回值，函数就会执行成功。

(5) 获得设备句柄

取得设备的路径以后，就可以准备开始与设备通信。使用 CreateFile 来打开一个 HID 设备，并且取得此设备的句柄，使用设备的句柄来与设备交换数据。当不再需要访问此设备时，应该调用 CloseHandle 函数来关闭设备并释放系统资源。

```
, 函数声明：
Public Declare Function CreateFile Lib "kernel32" Alias "CreateFileA" ( _
    ByVal lpFileName As String, _
    ByVal dwDesiredAccess As Long, _
    ByVal dwShareMode As Long, _
    ByRef lpSecurityAttributes As Long, _
    ByVal dwCreationDisposition As Long, _
    ByVal dwFlagsAndAttributes As Long, _
    ByVal hTemplateFile As Long _
) As Long

, 常量定义
Public Const GENERIC_READ = &H80000000
Public Const GENERIC_WRITE = &H40000000
Public Const FILE_SHARE_READ = &H1
Public Const FILE_SHARE_WRITE = &H2
Public Const OPEN_EXISTING = 3

, 调用
HidDevice = CreateFile( _
        DevicePathName, _
    GENERIC_READ Or GENERIC_WRITE, _
    (FILE_SHARE_READ Or FILE_SHARE_WRITE), _
    0, _
    OPEN_EXISTING, _
        0, _
        0
)
```

(6) 获得设备的厂商 ID、产品 ID 和产品序列号

要识别一个设备是否是所要的设备，只要比较此设备的厂商与产品 ID 即可。HidD_GetAttributes 函数用来取得一个包含厂商与产品 ID 以及产品的版本号码的结构。

```
, 函数声明
Public Declare Function HidD_GetAttributes Lib "hid.dll" ( _
ByVal HidDeviceObject As Long, _
        ByRef Attributes As HIDD_ATTRIBUTES _
) As Long

, 结构说明
Public Type HIDD_ATTRIBUTES
    Size As Long
    VendorID As Integer                    , 厂商 ID
    ProductID As Integer                   , 产品 ID
    VersionNumber As Integer               , 产品版本号
End Type

, 变量定义
Dim DeviceAttributes As HIDD_ATTRIBUTES

, 调用
DeviceAttributes.Size = LenB(DeviceAttributes)
Result = HidD_GetAttributes( _
        HidDevice, _                       , 由 CreateFile 函数返回
        DeviceAttributes
)
```

HidDevice 是由 CreateFile 函数所传回的设备句柄。如果 CreateFile 函数传回的是非零值，DeviceAttributes 结构就会填写正确值。 应用程序可以由 DeviceAttributes 结构取得厂商 ID、产品 ID 以及产品的版本号码。

如果厂商与产品 ID 不是想查找的，应用程序应该使用 CloseHandle 函数来关闭该设备，然后移到下一个 SetupDiEnumDeviceInterfaces 所检测到的下一个 HID 继续查找。 当应用程序检测到指定的设备或是检测完全部 HID，它应该调用 SetupDiDestroyDeviceInfoList 函数来释放 SetupDiGetClassDevs 函数所保留的资源。

8.5.3 获得 HID 的能力

获得设备的能力是可以进一步了解 HID 的手段，但不是必须的。如果在查找设备时，如果通过厂商 ID、产品 ID 和产品序列号可以确定特定的设备，一般是已知设备的细节信息了。如果不通过厂商 ID、产品 ID 和产品序列号确定设备，另一个方法是通过获得设备能力来确定设备。

获得 HID 的能力的过程主要经过以下几个步骤：

先使用 HidD_GetPreparsedData 函数获得一个包含设备能力信息的缓冲区的指针，调用 HidP_GetCaps 获得描述 HID 的能力的数据结构；

调用 HidP_GetValueCaps 取得描述能力的数值。

(1) 获得描述 HID 能力的数据结构

```
, 函数声明
Public Declare Function HidD_GetPreparsedData Lib "hid.dll" (_
        ByVal HidDeviceObject As Long, _
        ByRef PreparsedData As Long
) As Long

, 变量定义
Dim PreparsedData As Long

, 调用
Result = HidD_GetPreparsedData (HidDevice, _                    , 由 CreateFile 获得的设备句柄
                PreparsedData _
)
```

PreparsedData 是一个包含数据的缓冲区的指针。 程序并不需要访问缓冲区内的数据，只需要将缓冲区的开始地址传递给其他的 API 函数。当应用程序不再需要 PreparsedData 时，它应该调用 HidD_FreePreparsedData 函数来释放系统资源。

接下来调用 HidP_GetCaps ，该函数传回一个结构，里面包含设备能力的信息，包括设备的 Usage Page 、Usage 、报表长度以及按钮能力和数值能力等的数目。 如果不使用厂商与产品 ID 来识另设备， HidP_GetCaps 函数传回的设备能力信息可以帮助决定是否继续与此设备通信。

```
, 函数声明
Public Declare Function HidP_GetCaps Lib "hid.dll" ( _
        ByVal PreparsedData As Long, _
        ByRef Capabilities As HIDP_CAPS
) As Long

, 结构定义
Public Type HIDP_CAPS
```

```
    Usage As Integer                          ,  用法
    UsagePage As Integer                      ,  用法页
    InputReportByteLength As Integer          ,  输入报表长度
    OutputReportByteLength As Integer         ,  输出报表长度
    FeatureReportByteLength As Integer        ,  特征报表长度
    Reserved(16) As Integer
      NumberLinkCollectionNodes As Integer    ,  由函数 HidP_GetLinkCollectionNodes
                                              ,  返回的顶层集合连接数目
    NumberInputButtonCaps As Integer          ,  由函数 HidP_GetButtonCaps 返回的
                                              ,  包含输入按钮能力的结构 HIDP_BUTTON_
                                              , CAPS 的数目
      NumberInputValueCaps As Integer         ,  由函数 HidP_GetValueCaps 返回的
                                              ,  包含输入数值能力的结构 HIDP_VALUE_
                                              ‘ CAPS 的数目
    NumberInputDataIndices As Integer         ,  在输入报表中有关按键和数值的数据
                                              ,  指示器（ Data Indices ）的数目
    NumberOutputButtonCaps As Integer         ,  由函数 HidP_GetButtonCaps 返回的
                                              ,  包含输出按钮能力的结构 HIDP_BUTTON_
                                              , CAPS 的数目
    NumberOutputValueCaps As Integer          ,  由函数 HidP_GetValueCaps 返回的
                                              ,  包含输出数值能力的结构 HIDP_VALUE_
                                              ‘ CAPS 的数目
    NumberOutputDataIndices As Integer        ,  在输出报表中有关按键和数值的数据
                                              ,  指示器（ Data Indices ）的数目
    NumberFeatureButtonCaps As Integer        ,  由函数 HidP_GetButtonCaps 返回的
                                              ,  包含特征按钮能力的结构 HIDP_BUTTON_
                                              , CAPS 的数目
    NumberFeatureValueCaps As Integer         ,  由函数 HidP_GetValueCaps 返回的
                                              ,  包含特征数值能力的结构 HIDP_VALUE_
                                              ‘ CAPS 的数目
    NumberFeatureDataIndices As Integer       ,  在特征报表中有关按键和数值的数据
                                              ,  指示器（ Data Indices ）的数目
End Type

,  变量定义
Dim Capabilities As HIDP_CAPS

,  调用
Result = HidP_GetCaps (_
      PreparsedData, _                    ,  由函数 HidD_GetPreparsedData 定义
      Capabilities
)
```

HidP_GetCaps 函数填写 Capabilities 结构中的数据项, Capabilities 结构成员说明了 HID 的基本信息。这些信息包括：

用法页和用法： UsagePage 、Usage ；

输入、输出和特征报表长度： InputReportByteLength 、OutputReportByteLength 和 FeatureReportByteLength ；

由函数 HidP_GetLinkCollectionNodes 返回的顶层集合连接数目 NumberLinkCollectionNodes

在输入、输出和特征报表的按钮、数值和数据指示器的的数目。

(2) 获得描述 HID 数值能力的数据结构

通过 HidP_GetCaps 获得的 HID 的基本能力信息不是能从设备得到全部信息, 还可以调用另一个函数 HidP_GetValueCap ，该函数可以获得报表描述符中的数值和按钮的能力。HidP_GetValueCap 返回的是包含了报表描述符中全部数值信息的结构的指针。

```
, 函数声明
Public Declare Function HidP_GetValueCaps Lib "hid.dll" (_
        ByVal ReportType As Integer, _
        ByRef ValueCaps As Byte, _
        ByRef ValueCapsLength As Integer, _
        ByVal PreparsedData As Long _
) As Long

, ReportType 常量定义
Public Const HidP_Input = 0
Public Const HidP_Output = 1
Public Const HidP_Feature = 2

Public Type HidP_Value_Caps
        UsagePage As Integer
        ReportID As Byte
        IsAlias As Long
        BitField As Integer
        LinkCollection As Integer
        LinkUsage As Integer
        LinkUsagePage As Integer
        IsRange As Long
        IsStringRange As Long
        IsDesignatorRange As Long
        IsAbsolute As Long
        HasNull As Long
        Reserved As Byte
        BitSize As Integer
        ReportCount As Integer
        Reserved2 As Integer
        Reserved3 As Integer
        Reserved4 As Integer
        Reserved5 As Integer
        Reserved6 As Integer
        LogicalMin As Long
        LogicalMax As Long
        PhysicalMin As Long
        PhysicalMax As Long
        UsageMin As Integer
        UsageMax As Integer
        StringMin As Integer
        StringMax As Integer
        DesignatorMin As Integer
        DesignatorMax As Integer
        DataIndexMin As Integer
        DataIndexMax As Integer
End Type


, 变量定义
Dim ValueCaps(1023) As Byte
, 调用
Result = HidP_GetValueCaps (_
        HidP_Input, _
```

```
        ValueCaps(0), _
        Capabilities.NumberInputValueCaps, _
        PreparsedData
)
```

(3) 输出报表到设备

当应用程序取得 HID 设备的句柄，并且知道输出报表的字节数目后，它就可以传送输出报表给此设备。 应用程序先将要传送的数据复制到一个缓冲区内， 然后调用 WriteFile 函数。缓冲区的大小等于 HidP_GetCaps 函数返回的 HIDP_CAPS 结构的 OutputReportByteLength 属性值。这个大小值等于报表的字节大小， 再加上一个字节的 Report ID 。Report ID 是缓冲区的第一个字节。

HID 驱动程序用来确定输出报表的传输类型，根据 Windows 的版本以及 HID 接口有无中断输出端点而定。应用程序不需要干预，低阶的驱动程序会自动处理。

```
, 函数声明
Public Declare Function WriteFile Lib "kernel32" (_
        ByVal hFile As Long, _
        ByRef lpBuffer As Byte, _
        ByVal nNumberOfBytesToWrite As Long, _
        ByRef lpNumberOfBytesWritten As Long, _
        ByVal lpOverlapped As Long _
) As Long

, 变量定义
Dim SendBuffer() As Byte
Dim OutputReportData(7) As Byte
Dim Count as Long

, 调用
ReDim SendBuffer(Capabilities.OutputReportByteLength - 1)
SendBuffer(0) = 0                          , Report ID
, 将准备的数据从 OutputReportData 复制到 SendBuffer
For Count = 1 To Capabilities.OutputReportByteLength - 1
   SendBuffer(Count) = OutputReportData(Count - 1)
Next Count
NumberOfBytesWritten = 0
Result = WriteFile( _
        HidDevice, _                       , 由 CreateFile 函数返回的设备句柄
        SendBuffer(0), _                   , 输出报表缓存
        CLng(Capabilities.OutputReportByteLength), _                , 要输出字节数
        NumberOfBytesWritten, _                                     , 实际输出的字节数
        0
)
```

如果函数返回的 Result 数值不等于零，表示函数成功执行。

如果接口只支持数值为 0 的 Report ID ，这个 Report ID 并不传送， 但需要出现在应用程序传给 WriteFile 函数的缓冲区内。

WriteFile 函数在 HID 通信中最常发生的错误是 CRC error 。此错误表示主机控制器试图要传送报表， 但是没有从设备收到预期的响应。 通常该错误不是发生在 CRC 计算时所检测到的错误，而是因为主机没有收到固件预期的响应。

(4) 从设备输入出报表

当应用程序取得 HID 设备的句柄，并且知道输入报表的字节数目后，就可以从此设备读取输入报表。应用程序先声明一个缓冲区来储存数据，然后调用 ReadFile 函数。用来储

存数据的缓冲区大小等于 HidP_GetCaps 函数所返回的 HIDP_CAPS 结构的 InputReport ByteLength 属性值。

```
, 函数声明
Public Declare Function ReadFile Lib "kernel32" ( _
        ByVal hFile As Long, _
        ByRef lpBuffer As Byte, _
        ByVal nNumberOfBytesToRead As Long, _
        ByRef lpNumberOfBytesRead As Long, _
        ByVal lpOverlapped As Long _
) As Long

, 变量定义
Dim Count
Dim NumberOfBytesRead As Long
Dim ReadBuffer() As Byte                    , 输入缓冲区, 字节 0 为 Report ID
ReDim ReadBuffer(Capabilities.InputReportByteLength - 1)

, 调用
Result = ReadFile( _
   HidDevice, _                             , 由 CreateFile 函数返回的设备句柄
   ReadBuffer(0), _                         , 输入缓冲区首地址
   CLng(Capabilities.InputReportByteLength), _                  , 要读取的字节数
   NumberOfBytesRead, _                     , 读到的字节数
      0
)
```

ReadBuffer 字节数组包含报表的数据。如果函数返回的 Result 数值不等于零，表示函数成功执行。

通过 ReadFile 读取的缓冲区的第一个字节是 Report ID ，后续是从设备读取的报表数据。如果接口只支持一个 Report ID ，此 Report ID 不在总线上传输， 但会出现在 ReadBuffer 缓冲区内。

调用 ReadFile 函数不会立刻开始总线上的传输， 只是主机在定时的中断输入传输中读取一个报表。如果没有未读取的报表，就等待下一个传输完成。主机在检测设备后开始请求报表，当 HID 驱动程序加载后，驱动程序将报表储存在环状缓冲区内。当缓冲区已经填满并有新的报表到达时， 旧的报表会被覆盖。 调用 ReadFile 函数会读取缓冲区内最旧的报表。

在 Windows 98 SE 以及后来的版本中，默认的环状缓冲区尺寸是 8 个报表。应用程序可以使用 HidD_SetNumInputBuffers 函数来设置缓冲区的大小。如果应用程序没有频繁的请求报表，有些报表就会丢失。如果不想要丢失报表，就应该改用特征报表。

如果报表的数据从上一次传输后就没有改变，闲置速率决定设备是否要传送报表。在检测设备时 HID 驱动程序会试图将设备的闲置速率设置为 0，这表示除非报表的数据有改变否则 HID 不会传送报表。没有可以改变闲置速率的 API 函数。

如果设备拒绝将闲置速率设置为 0，可以传回 Stall 来响应 Set_Idle 请求来通知主机设备不支持该请求。

如果设备不支持 Set_Idle 请求，而且应用程序只要读取一次报表，固件可以设置成只传送一次报表。 在送报表后， 固件可以设置端点传回 NAK 来响应输入令牌信息包。 如果设备有新的数据要传送，固件可以设置端点来传回该数据。否则设备会在主机轮询端点时继续传送相同的报表，应用程序也会重复地读取相同的报表。

上面程序中 ReadFile 的最后一个参数为 0，表示 ReadFile 调用是阻塞的。当应用程序在环形缓冲区为空时调用 ReadFile ，应用程序将会被挂起，直到有输入报表为止，否则只能按下 Ctrl＋Alt＋Del 来关闭应用程序，或是从总线上移除设备。

采用多线程方式编程可以较好的解决这个问题， 在另一个线程中调用 ReadFile 可以避免主线程被挂起，在使用 Visual Basic 编写多线程应用程序会遇到困难，这是因为 Visual Basic 本身不支持多线程。 而在 Visual C++ 编写 API 方式通信程序时可以采用多线程方式，ReadFile 函数调用发生在一个独立的线程，这样可以实现重叠 I/O 操作。

解决的办法之一是保证设备永远有数据传送，可以将固件设计为输入端点永远启用并且准备响应要求。如果没有新的数据传送，设备可以传送上一次的数据，或是传回一个特定代码来指示没有新的数据。

也可以采用这样的方法，在调用 ReadFile 之前，应用程序先调用 WriteFile 来传送一个报表，报表内可以包含一个特定代码来告诉固件准备传送数据，这样当应用程序调用 ReadFile 时，设备的端点就会启用并且有数据准备传送。

比较好的方法是使用 ReadFile 的重叠选项。在重叠的读取时 ReadFile 函数会立即返回（即使没有可读数据） ，然后应用程序调用 WaitForSingleObject 函数来读取数据。WaitForSingleObject 函数可以设置暂停，如果数据在指定时间内尚未抵达，此函数会传回一个码来指示此情况，然后应用程序可以使用 CancelIo 函数来取消读取动作。

要使用重叠 I/O，CreateFile 函数必须在 dwFlagsAndAttributes 参数中传递一个重叠的结构。 应用程序调用 CreateEvent 函数建立一个事件对象， 在 ReadFile 完成后此事件对象会被设置成信号状态。 当应用程序调用 ReadFile 时，它传递一个重叠结构的指针， 重叠结构的 hEvent 参数是一个事件对象的代号。 应用程序调用 WaitForSingleObject 函数来传递此事件代号，以及一个以 ms 为单位的指定时间间隔。在读取到数据或到达该时间间隔时，此函数才返回。

```
， 函数声明
Public Declare Function ReadFile Lib "kernel32" ( _
        ByVal hFile As Long, _
        ByRef lpBuffer As Byte, _
        ByVal nNumberOfBytesToRead As Long, _
        ByRef lpNumberOfBytesRead As Long, _
        ByVal lpOverlapped As Long _
) As Long

Public Declare Function CreateEvent Lib "kernal32.dll" Alias CreatEventA( _
        ByVal SecurityAttributes As Long, _
        ByVal bManualReset As Long, _
        ByVal bInitialState As Long, _
        ByVal lpName As String
) As Long

Public Declare Function WaitForSingleObject Lib “kernal32.dll ” (_
        ByVal hHandle as Long, _
        ByVal dwMilliseconds as Long
) as Long

， 重叠结构声明
Public Type OVERLAPPED
        Internal as Long
        InternalHigh as Long
        Offset as Long
        OffsetHigh as Long
        hEvent as Long
```

```
End Type

, 常量定义
Public Const FILE_FLAG_OVERLAPPED = &H40000000

, 变量定义
Dim EventObject as Long
Dim HIDOverlapped as OVERLAPPED

, 部分代码
EventObject = CreatEvent (0&, True, True,                    “ ”)
HIDOverlapped.hEvent = EventObject
HIDOverlapped.Offset = 0
HIDOverlapped.OffsetHigh = 0

HidDevice = CreateFile( _
        DevicePathName, _
    GENERIC_READ Or GENERIC_WRITE, _
    (FILE_SHARE_READ Or FILE_SHARE_WRITE), _
    0, _
    OPEN_EXISTING, _
        FILE_FLAG_OVERLAPPED, _
        0
)       , 获得 HID 设备句柄

Result = ReadFile( _
   HidDevice, _                                  , 由 CreateFile      函数返回的设备句柄
   ReadBuffer(0), _                              , 输入缓冲区首地址
   CLng(Capabilities.InputReportByteLength), _                          , 要读取的字节数
   NumberOfBytesRead, _                          , 读到的字节数
      HIDOverlapped
)       , 读取报表，同时传送一个指针到重叠结构

Result = WaitForSingleObject (EventObject, 5000)
        , 等待 ReadFile     完成，超时间隔设为    5 秒
```

(5) 特征报表的传送

应用程序调用 HidD_SetFeature 函数传送一个特征报表到设备。

```
, 函数声明
Public Declare Function HidD_SetFeature Lib "hid.dll" ( _
        ByVal HidDeviceObject As Long, _
        ByRef ReportBuffer As Byte, _
        ByVal ReportBufferLength As Long
) As Long

, 调用
Result = HidD_SetFeature( _
   HidDevice, _                                  , 由 CreateFile      函数返回的设备句柄
   SendBuffer(0), _                              , 输出缓冲区首地址
   CLng(Capabilities.FeatureReportByteLength)                           , 特征报表长度字节数
)
```

API 函数 HidD_GetFeature 用于从设备读取特征报表，通过对该 API 函数的调用，主机控制器以控制传输送出 Get_Feature 请求，并在数据阶段，设备回传特征报表。

```
, 函数声明
Public Declare Function HidD_GetFeature Lib "hid.dll" ( _
        ByVal HidDeviceObject As Long, _
        ByRef ReportBuffer As Byte, _
        ByVal ReportBufferLength As Long
) As Long

, 调用
Result = HidD_GetFeature( _
    HidDevice, _                                    , 由 CreateFile 函数返回的设备句柄
    ReadBuffer(0), _                                , 输出缓冲区首地址
    CLng(Capabilities.FeatureReportByteLength)                  , 特征报表长度字节数
)
```

(6) 关闭设备

当结束与 HID 的通信，需要调用 CloseHandle 函数关闭通信

```
, 函数声明
Public Declare Function CloseHandle Lib "kernel32" ( _
        ByVal hObject As Long _
 ) As Long

, 调用
Result = CloseHandle(HidDevice)
```

8.6 HID 实验

本节介绍通过高级接口实验台进行的一个 HID 编程实验。

在 USB 设备软件的开发过程中，借助于一些工具软件的测试会对 USB 设备的信息获取和通信过程又更深入的理解。USB 测试软件有很多，如 USBView 、BusHound 等，下面针对一个具体的 HID 设备的 API 通信软件的开发，对这两个工具软件作一简单介绍。

高级接口实验台中，有一个 HID 实验，在该实验中，实验台通过固件软件设计了一个简单的 HID 仿真设备，在 PC 的 API 通信软件的开发过程中，可以借助 USB 工具软件对实验台的 HID 设备进行测试。

8.6.1 获得描述符

将高级接口实验台通过 USB 电缆与 PC 连接后，在实验台上通过菜单选择 HID 实验，实验台显示器显示器上会显示 HID 实验界面。

实验台的 HID 实验是一个为了学习 HID 编程而专门设计的一个简单的 HID 仿真设备，设备中设计了 8 个寄存器（ R1~R8 ），可以通过 USB 接口与主机交换数据。其中 R1 和 R2 两个寄存器只是数据存储单元，主机可以对这两个寄存器进行附值，也可以读取寄存器的值。寄存器 R3~R8 构成了一个日期和时钟，6 个寄存器的值分别表示年、月、日、时、分、秒。时钟在当前值的基础上运行，可以通过主机对时钟进行设置，也可以读取当前的时钟值。

在实验台上设计了自动回传功能，如果自动回传打开，时钟每一秒向主机传送一次报

表。实验台可以显示发送和接收的有效数据字节数。

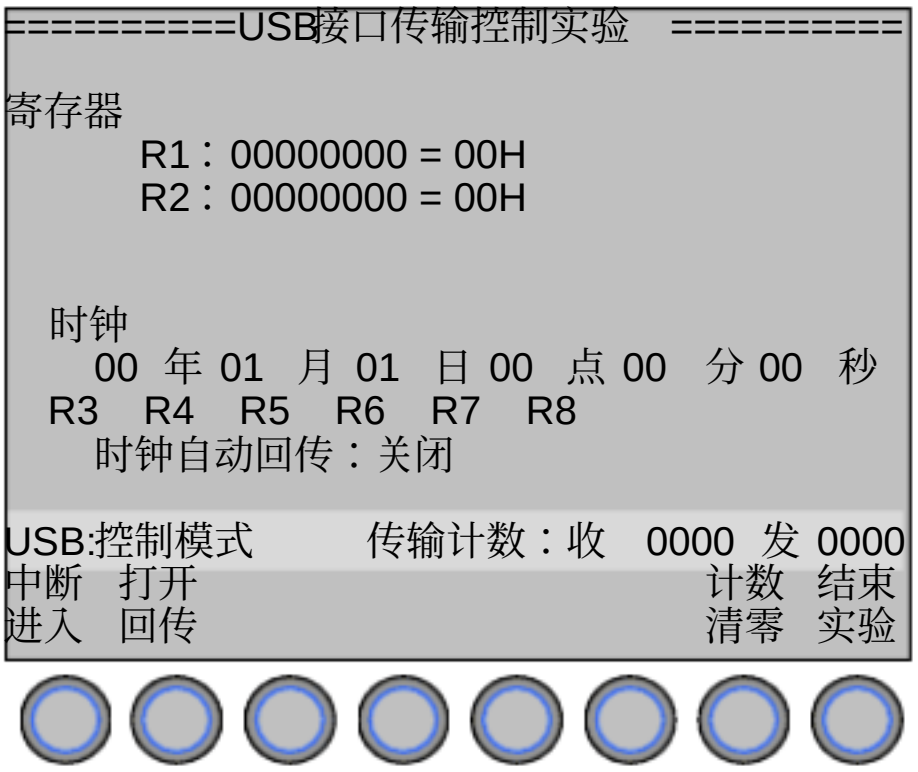

图 8-7 高级接口实验台 HID 实验界面

USBView 是 Microsoft 提供的一个简单的 USB 测试软件，该工具软件是一个完全的绿色软件，只有一个 EXE 程序文 将件，不需要安装，在 Windows 环境下直接运行。

USBView 其主要功能是获得 USB 设备的各个描述符。

可以运行 USBView 获得实验台仿真的 HID 设备的描述符，运行 USBView 后显示以下程序界面。

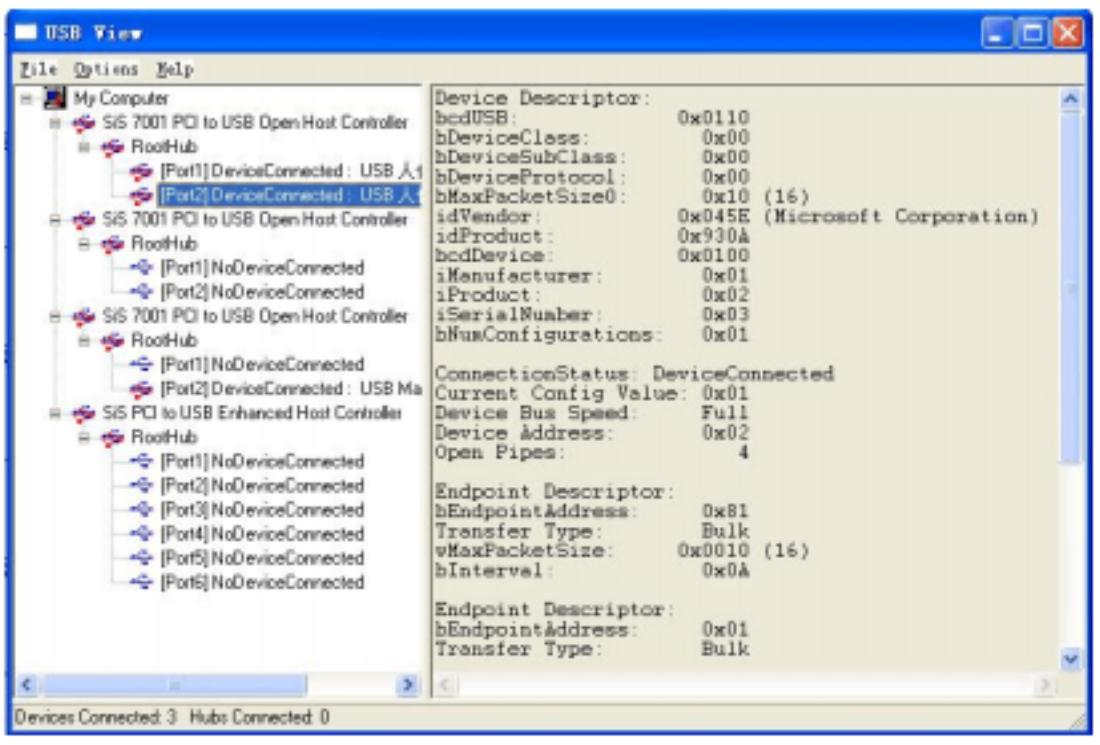

图 8-8 USBView 运行界面

窗口的左侧显示全部的 USB 设备连接树，在其中找到一个显示“ USB 人体工程学设备”的分支，选中后右侧窗口显示全部的（报表描述符除外）描述符。 USBView 显示的高级接口实验台的 HID 仿真设备的描述符如下。

Device Descriptor:	
bcdUSB:	0x0110
bDeviceClass:	0x00

bDeviceSubClass:	0x00
bDeviceProtocol:	0x00
bMaxPacketSize0:	0x10 (16)
idVendor:	0x045E (Microsoft Corporation)
idProduct:	0x930A
bcdDevice:	0x0100
iManufacturer:	0x01
iProduct:	0x02
iSerialNumber:	0x03
bNumConfigurations:	0x01
ConnectionStatus: DeviceConnected	
Current Config Value:	0x01
Device Bus Speed:	Full
Device Address:	0x02
Open Pipes:	4
Endpoint Descriptor:	
bEndpointAddress:	0x81
Transfer Type:	Bulk
wMaxPacketSize:	0x0010 (16)
bInterval:	0x0A
Endpoint Descriptor:	
bEndpointAddress:	0x01
Transfer Type:	Bulk
wMaxPacketSize:	0x0010 (16)
bInterval:	0x0A
Endpoint Descriptor:	
bEndpointAddress:	0x82
Transfer Type:	Interrupt
wMaxPacketSize:	0x0040 (64)
bInterval:	0x01
Endpoint Descriptor:	
bEndpointAddress:	0x02
Transfer Type:	Interrupt
wMaxPacketSize:	0x0040 (64)
bInterval:	0x01

8.6.2 设备的初始化

在将高级接口实验台连接到计算机并操作实验台进入 HID 接口实验后， Windows 系统会发现设备并读取设备的各种描述符， 可以通过一个 USB 工具软件截取 Windows 和设备之间的请求应答过程。工具软件 BusHound 可以实现这个功能。

BunHound 是一个功能全面的总线分析仪软件，可以实现对计算机通过各种接口连接的设备的通信过程进行截取和分析。图 8-9 是 BusHound 的设备连接树（ Devices ）界面。

当高级接口实验台与计算机通过 USB 接口连接并操作实验台进入 HID 实验，可以在 BusHound 的设备连接树中找到一个“ USB 人体工程学设备”项目。选中该项目可以实现对设备的通信过程的截取。

选中高级接口实验台的 HID 设备项，进入到采集（ Capture ）界面，先控制实验台退出 HID 实验，按 BusHound 的采集界面的 Stop 按钮，再按 Run 按钮，然后控制实验台进入 HID 实验界面。 Windows 系统发现设备并请求设备的各种描述符，然后完成对设备的必要的设置， BusHound 可以采集到以上通信过程。

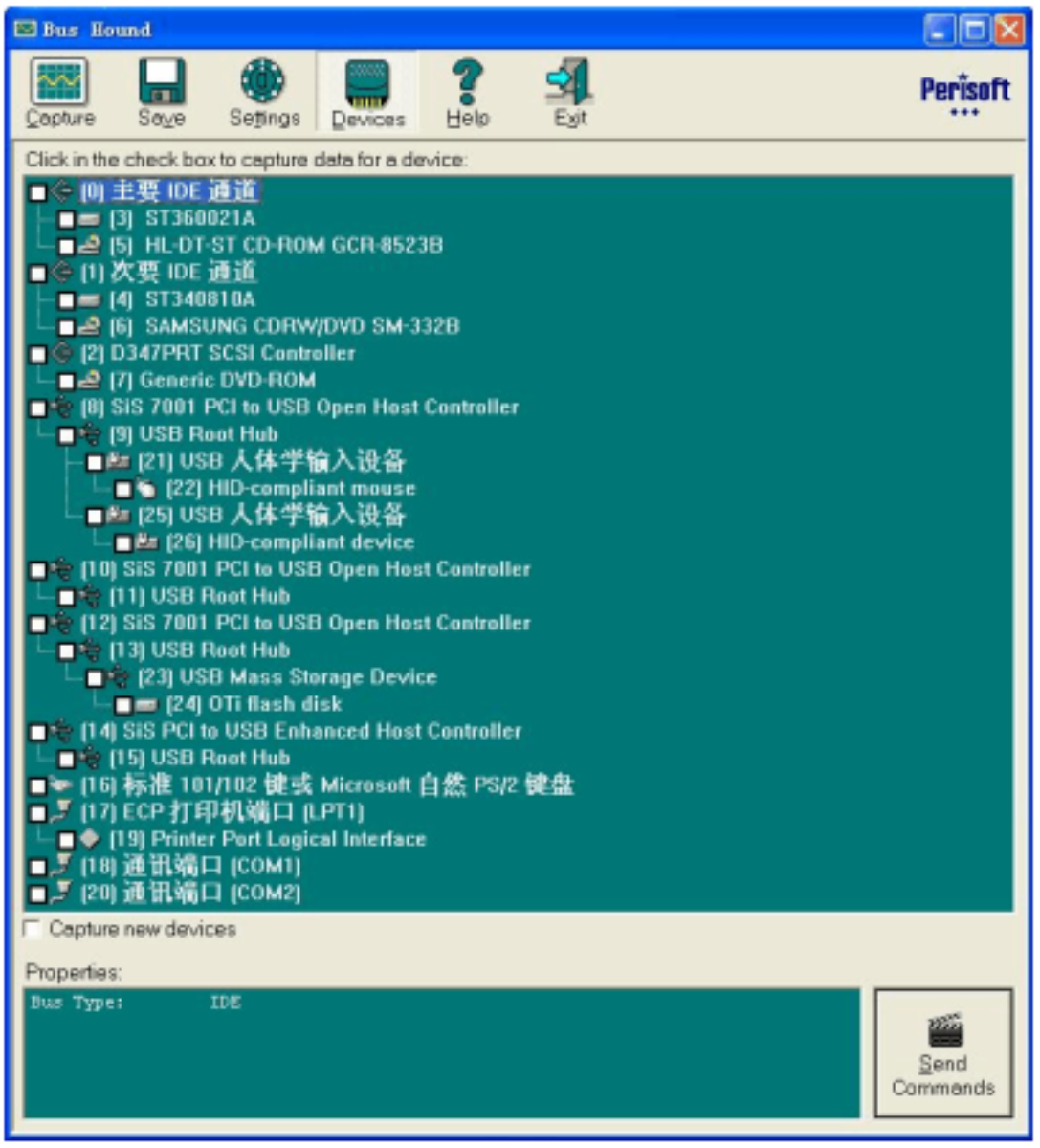

图 8-9 BusHound 的设备树显示界面

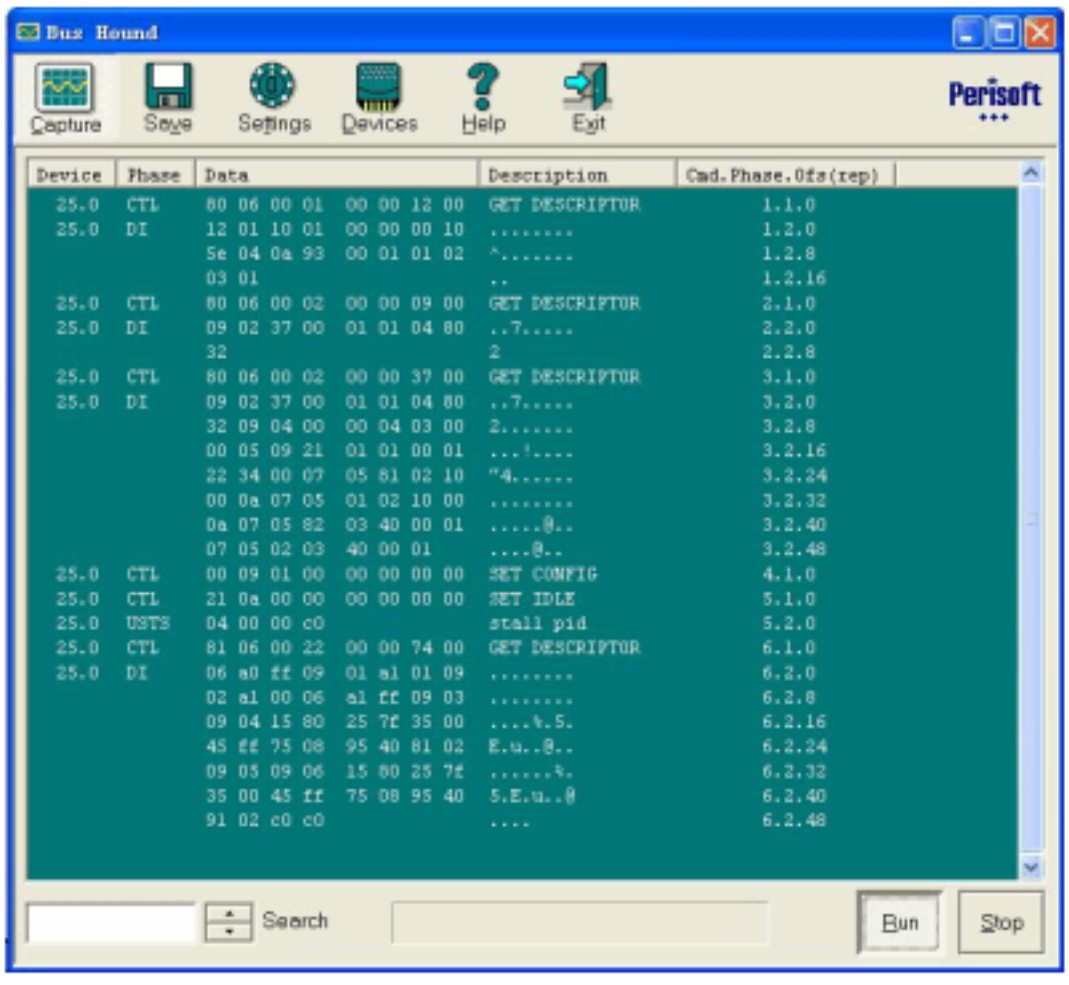

图 8-10 BusHound 的采集显示界面

下面分析 BusHound 截取的数据内容。

? 主机发送： 80 06 00 01 00 00 12 00
Get_Descriptor 请求，请求设备回传设备描述符。

? 设备发送： 12 01 10 01 00 00 00 10 5e 04 0a 93 00 01 01 02 03 01
设备描述符内容： USB 版本 =1.1 、类别 /协议码 =0、 EP0 的最大包尺寸 =10 、VID=045E 、PID=930A 、版本 =1.0 、厂商、 产品和序列号字符串索引、 配置数 =1 。

? 主机发送： 80 06 00 02 00 00 09 00
Get_Descriptor 请求，请求设备回传配置描述符。

? 设备发送： 09 02 37 00 01 01 04 80 32
配置描述符内容：该描述符及后续描述符总长度 =55、支持接口数 =1、配置标示符 =1、总线供电、最大 100mA。

? 主机发送： 80 06 00 02 00 00 37 00
Get_Descriptor 请求，请求设备回传配置、接口、 HID 和端点描述符。

? 设备发送： 09 02 37 00 01 01 04 80 32 09 04 00 00 04 03 00 00 05 09 21 01 01 00 01 22 34 00 07 05 81 02 10 00 0a 07 05 01 02 10 00 0a 07 05 82 03 40 00 01 07 05 02 03 40 00 01 按顺序为：
9 字节配置描述符：含义同上。
9 字节接口描述符：标示符 =0、支持的端点 =4、类别 =HID。
9 字节 HID 描述符：版本 =1.01、有 1 个报表描述符（长度为 52）
7 字节端点描述符： 1 号批量输入，包尺寸 =16。
7 字节端点描述符： 1 号批量输出，包尺寸 =16。
7 字节端点描述符： 2 号中断输入，包尺寸 =64、1ms 轮询。
7 字节端点描述符： 2 号中断输出，包尺寸 =64、1ms 轮询。

? 主机发送： 00 09 01 00 00 00 00 00
Set_Configuation 请求，配置号 =1。

? 主机发送： 21 0a 00 00 00 00 00 00
Set_Idle 请求，设定间隔时间为 0。

? 设备发送： 04 00 00 c0
STALL ，不支持请求。

? 主机发送： 81 06 00 22 00 00 74 00
Get_Descriptor 请求，请求设备回传报表描述符。

? 设备发送： 06 a0 ff 09 01 a1 01 09 02 a1 00 06 a1 ff 09 03 09 04 15 80 25 7f 35 00 45 ff 75 08 95 40 81 02 09 05 09 06 15 80 25 7f 35 00 45 ff 75 08 95 40 91 02 c0 c0
报表描述符。

8.6.3 HID 测试程序的实现

使用 Visual Basic 编写一个针对高级接口实验台的应用程序，实现与设备的通信。该测试程序实现以下功能：

? 查找设备：根据指定的 VID 和 PID 实现设备查找；
? 获得设备能力：调用相应的 API ，获得设备的能力；
? 发送报表：将界面输入的 R1~R8 的数值按照报表的指定格式传送到设备；
? 接收报表：将设备的界面显示的 R1~R8 的数值读取到应用程序；
? 显示 API 函数调用信息：显示每一个 API 函数的调用结果；
? 关闭设备。

下图是程序运行的界面显示。

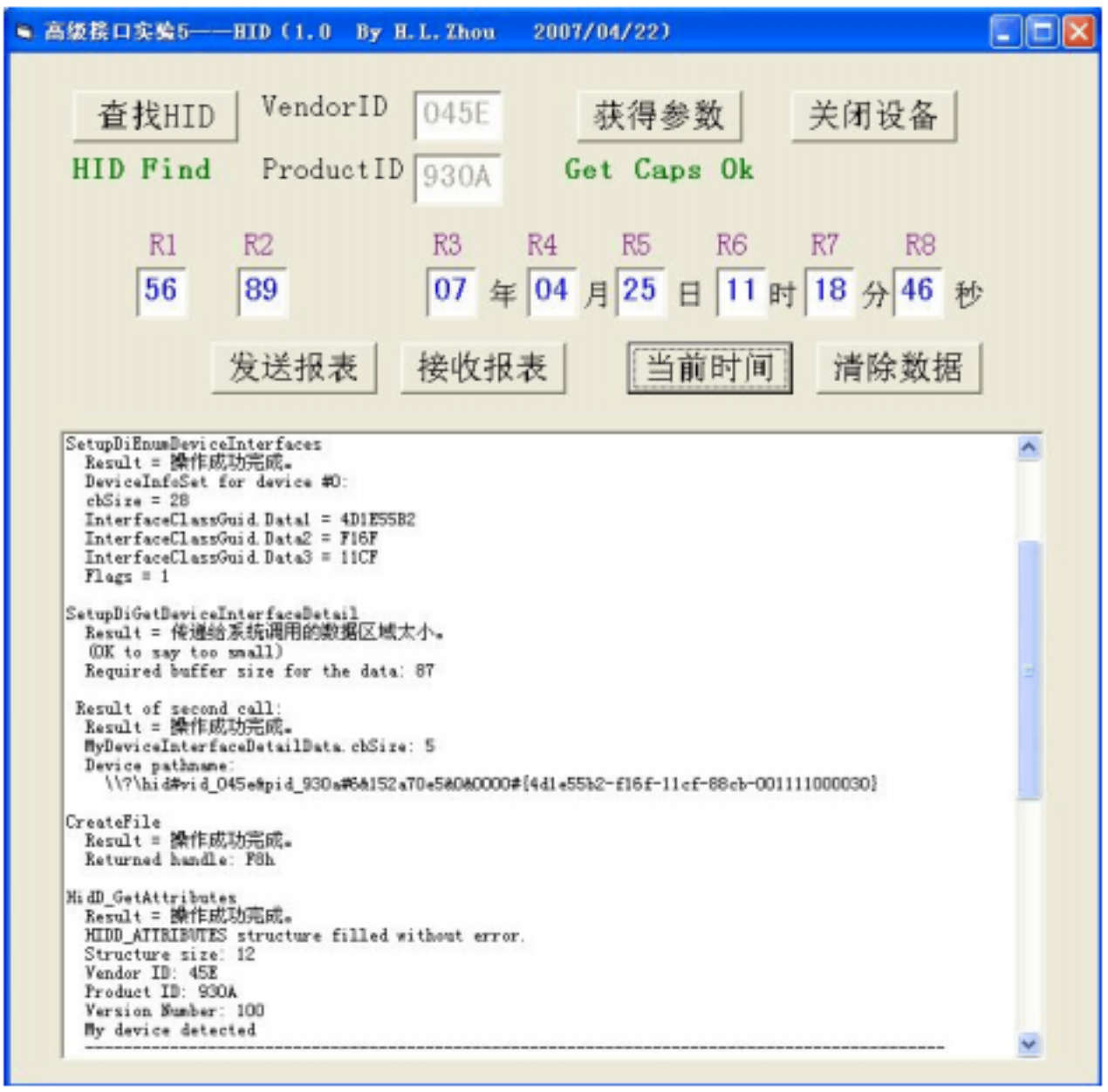

图 8-11 针对实验台的 HID 测试程序显示界面

在获得能力的 API 函数的调用结果中，输入报表和输出报表的长度为 65 字节，在发送和接收报表时的长度设定都需要设为 65。但在该实验中真正用到的有效数据只有 8 个寄存器的数据， 占用其中的 8 个字节， 报表的首字节 （第 0 个）为 0，表示 Report ID 。第 1、2 个字节分别 55h、AAh 为用户定义的标示字。 第 3、4 字节为用户定义命令编码， 输出报表中为 01、08，输入报表中为 02、08。从第 5 到第 12 字节顺序为 R1、R2、…R8 的值。之后的数据为保留字节。

```
'HID查找、能力检测、输入报表、输出报表演示程序
'修改自许永和的《介面设计与实习　　使用 Bisual Basic》


Option Explicit

'变量定义 ********************************************************************************
Dim Capabilities As HIDP_CAPS
Dim DataString As String
Dim DetailData As Long
Dim DetailDataBuffer() As Byte
Dim DeviceAttributes As HIDD_ATTRIBUTES
Dim DevicePathName As String
Dim DeviceInfoSet As Long
Dim ErrorString As String
Dim HidDevice As Long
```

```
Dim LastDevice As Boolean
Dim MyDeviceDetected As Boolean
Dim MyDeviceInfoData As SP_DEVINFO_DATA
Dim MyDeviceInterfaceDetailData As SP_DEVICE_INTERFACE_DETAIL_DATA
Dim MyDeviceInterfaceData As SP_DEVICE_INTERFACE_DATA
Dim Needed As Long
Dim OutputReportData(7) As Byte
Dim PreparsedData As Long
Dim Result As Long
Dim Timeout As Boolean

Dim MyVendorID As Long              'VID、 PID
Dim MyProductID As Long

'****************************************************************************************
'查找全部的  HID设备，直到找到  VID、 PID符合的一个  HID
'如果找到，  MyDeviceDetected为 True
'****************************************************************************************

Function FindTheHid() As Boolean

Dim Count As Integer
Dim GUIDString As String
Dim HidGuid As GUID
Dim MemberIndex As Long

LastDevice = False
MyDeviceDetected = False

'调用  HidD_GetHidGuid函数获得  GUID

Result = HidD_GetHidGuid(HidGuid)
Call DisplayResultOfAPICall("GetHidGuid")

GUIDString = Hex$(HidGuid.Data1) & "-" & Hex$(HidGuid.Data2) & "-" & Hex$(HidGuid.Data3) & "-"

For Count = 0 To 7
    If HidGuid.Data4(Count) >= &H10 Then
        GUIDString = GUIDString & Hex$(HidGuid.Data4(Count)) & " "
    Else
        GUIDString = GUIDString & "0" & Hex$(HidGuid.Data4(Count)) & " "
    End If
Next Count

lstResults.AddItem " GUID for system HIDs: "
lstResults.AddItem " " & GUIDString

'调用  SetupDiGetClassDevs函数获得指向  HID信息集的指针

DeviceInfoSet = SetupDiGetClassDevs(HidGuid, vbNullString, 0, (DIGCF_PRESENT Or DIGCF_DEVICEINTERFACE))

Call DisplayResultOfAPICall("SetupDiClassDevs")
DataString = GetDataString(DeviceInfoSet, 32)
lstResults.AddItem " DeviceInfoSet:" & DeviceInfoSet


'下面循环，从  MemberIndex=0开始，查找指定  HID
MemberIndex = 0
```

```
Do
    '调用  SetupDiEnumDeviceInterfaces函数获得   SP_DEVICE_INTERFACE_DATA结构指针

    MyDeviceInterfaceData.cbSize = LenB(MyDeviceInterfaceData)
    Result = SetupDiEnumDeviceInterfaces(DeviceInfoSet, 0, HidGuid, MemberIndex, MyDeviceInterfaceData)

    Call DisplayResultOfAPICall("SetupDiEnumDeviceInterfaces")
    If Result = 0 Then LastDevice = True

    '如果调用成功
    If Result <> 0 Then
        '显示获得的信息
        lstResults.AddItem " DeviceInfoSet for device #" & CStr(MemberIndex) & ": "
        lstResults.AddItem " cbSize = " & CStr(MyDeviceInterfaceData.cbSize)
        lstResults.AddItem " InterfaceClassGuid.Data1 _
                = " & Hex$(MyDeviceInterfaceData.InterfaceClassGuid.Data1)
        lstResults.AddItem " InterfaceClassGuid.Data2 _
                = " & Hex$(MyDeviceInterfaceData.InterfaceClassGuid.Data2)
lstResults.AddItem " InterfaceClassGuid.Data3 _
                = " & Hex$(MyDeviceInterfaceData.InterfaceClassGuid.Data3)
lstResults.AddItem " Flags = " & Hex$(MyDeviceInterfaceData.Flags)


'调用  SetupDiGetDeviceInterfaceDetail函数，获得  SP_DEVICE_INTERFACE_DETAIL_DATA结构
'注意：该函数需要调用两次，最后获得设备路径

MyDeviceInfoData.cbSize = Len(MyDeviceInfoData)
Result = SetupDiGetDeviceInterfaceDetail(DeviceInfoSet, MyDeviceInterfaceData, 0, 0, Needed, 0)

DetailData = Needed

Call DisplayResultOfAPICall("SetupDiGetDeviceInterfaceDetail")
lstResults.AddItem " (OK to say too small)"
lstResults.AddItem " Required buffer size for the data: " & Needed

'存储结构的长度
MyDeviceInterfaceDetailData.cbSize = Len(MyDeviceInterfaceDetailData)

ReDim DetailDataBuffer(Needed)

'存储结构的前  4 和字节，  cbSize
Call RtlMoveMemory(DetailDataBuffer(0), MyDeviceInterfaceDetailData, 4)

'再一次调用
Result = SetupDiGetDeviceInterfaceDetail(DeviceInfoSet, _
        MyDeviceInterfaceData, VarPtr(DetailDataBuffer(0)), DetailData, Needed, 0)

Call DisplayResultOfAPICall(" Result of second call: ")
lstResults.AddItem " MyDeviceInterfaceDetailData.cbSize: " & CStr(MyDeviceInterfaceDetailData.cbSize)

DevicePathName = CStr(DetailDataBuffer())
DevicePathName = StrConv(DevicePathName, vbUnicode)                '转换成  Unicode
DevicePathName = Right$(DevicePathName, Len(DevicePathName) - 4)   '删除  4 个字节
lstResults.AddItem " Device pathname: "
lstResults.AddItem "     " & DevicePathName

'调用  CreateFile函数，获得设备句柄：    HidDevice
```

```
HidDevice = CreateFile(DevicePathName, GENERIC_READ Or GENERIC_WRITE, _
            (FILE_SHARE_READ Or FILE_SHARE_WRITE), 0, OPEN_EXISTING, 0, 0)

Call DisplayResultOfAPICall("CreateFile")
lstResults.AddItem " Returned handle: " & Hex$(HidDevice) & "h"

'调用  HidD_GetAttributes获得 HID的 VID、PID

DeviceAttributes.Size = LenB(DeviceAttributes)
Result = HidD_GetAttributes(HidDevice, DeviceAttributes)

Call DisplayResultOfAPICall("HidD_GetAttributes")
If Result <> 0 Then
    lstResults.AddItem " HIDD_ATTRIBUTES structure filled without error."
Else
    lstResults.AddItem " Error in filling HIDD_ATTRIBUTES structure."
End If

lstResults.AddItem " Structure size: " & DeviceAttributes.Size
lstResults.AddItem " Vendor ID: " & Hex$(DeviceAttributes.VendorID)
lstResults.AddItem " Product ID: " & Hex$(DeviceAttributes.ProductID)
lstResults.AddItem " Version Number: " & Hex$(DeviceAttributes.VersionNumber)

'看看是不是指定的  VID、PID
If (DeviceAttributes.VendorID = MyVendorID) And (DeviceAttributes.ProductID = MyProductID) Then
    lstResults.AddItem " My device detected "
    lstResults.AddItem " ----------------------------------------------------------------------------------------"
    lblHID.Caption = "HID Find"
    MyDeviceDetected = True
    cmdGetCaps.Enabled = True
    cmdClose.Enabled = True
    txtVendorID.Enabled = False
    txtProductID.Enabled = False
Else
    MyDeviceDetected = False
    Result = CloseHandle(HidDevice)
    DisplayResultOfAPICall ("CloseHandle")
End If
End If

MemberIndex = MemberIndex + 1  '准备查找下一个


Loop Until (LastDevice = True) Or (MyDeviceDetected = True)

End Function

'*****************************************************************************************
'获得上一个  API函数的执行信息
'*****************************************************************************************

Private Function GetErrorString(ByVal LastError As Long) As String

Dim Bytes As Long
Dim ErrorString As String
ErrorString = String$(129, 0)
Bytes = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM, 0&, LastError, 0, ErrorString$, 128, 0)

If Bytes > 2 Then          '去掉其中的回车
    GetErrorString = Left$(ErrorString, Bytes - 2)
```

```
End If

End Function

'*****************************************************************************************
'清除数据域显示
'*****************************************************************************************

Private Sub cmdClear_Click()

txtR1.Text = ""
txtR2.Text = ""
txtYear.Text = ""
txtMonth.Text = ""
txtDay.Text = ""
txtHour.Text = ""
txtMinute.Text = ""
txtSecond.Text = ""
End Sub

'*****************************************************************************************
'查找  HID
'*****************************************************************************************

Private Sub cmdFindHID_Click()
Call FindTheHid
End Sub

'*****************************************************************************************
'显示  API 函数的执行结果
'*****************************************************************************************

Private Sub DisplayResultOfAPICall(FunctionName As String)

Dim ErrorString As String
lstResults.AddItem ""
ErrorString = GetErrorString(Err.LastDllError)
lstResults.AddItem FunctionName
lstResults.AddItem " Result = " & ErrorString
End Sub

'*****************************************************************************************
'程序初始化
'*****************************************************************************************

Private Sub Form_Load()
frmMain.Show
tmrDelay.Enabled = False
lstResults.Clear
MyVendorID = &H45E
MyProductID = &H930A
End Sub

'*****************************************************************************************
'获得  HID的能力信息
'*****************************************************************************************

Private Sub cmdGetCaps_click()
```

```
Dim ppData(29) As Byte
Dim ppDataString As Variant

'调用   HidD_GetPreparsedData获得一个缓冲区指针

Result = HidD_GetPreparsedData(HidDevice, PreparsedData)
Call DisplayResultOfAPICall("HidD_GetPreparsedData")

Result = RtlMoveMemory(ppData(0), PreparsedData, 30)
Call DisplayResultOfAPICall("RtlMoveMemory")

ppDataString = ppData()
ppDataString = StrConv(ppDataString, vbUnicode)

'调用   HidP_GetCaps获得  HID_CAPS结构数据

Result = HidP_GetCaps(PreparsedData, Capabilities)

Call DisplayResultOfAPICall("HidP_GetCaps")
lstResults.AddItem " Last error: " & ErrorString
lstResults.AddItem " Usage: " & Hex$(Capabilities.Usage)
lstResults.AddItem " Usage Page: " & Hex$(Capabilities.UsagePage)
lstResults.AddItem " Input Report Byte Length: " & Capabilities.InputReportByteLength
lstResults.AddItem " Output Report Byte Length: " & Capabilities.OutputReportByteLength
lstResults.AddItem " Feature Report Byte Length: " & Capabilities.FeatureReportByteLength
lstResults.AddItem " Number of Link Collection Nodes: " & Capabilities.NumberLinkCollectionNodes
lstResults.AddItem " Number of Input Button Caps: " & Capabilities.NumberInputButtonCaps
lstResults.AddItem " Number of Input Value Caps: " & Capabilities.NumberInputValueCaps
lstResults.AddItem " Number of Input Data Indices: " & Capabilities.NumberInputDataIndices
lstResults.AddItem " Number of Output Button Caps: " & Capabilities.NumberOutputButtonCaps
lstResults.AddItem " Number of Output Value Caps: " & Capabilities.NumberOutputValueCaps
lstResults.AddItem " Number of Output Data Indices: " & Capabilities.NumberOutputDataIndices
lstResults.AddItem " Number of Feature Button Caps: " & Capabilities.NumberFeatureButtonCaps
lstResults.AddItem " Number of Feature Value Caps: " & Capabilities.NumberFeatureValueCaps
lstResults.AddItem " Number of Feature Data Indices: " & Capabilities.NumberFeatureDataIndices

'调用   HidP_GetValueCaps获得  HID 能力的数值

Dim ValueCaps(1023) As Byte

Result = HidP_GetValueCaps(HidP_Input, ValueCaps(0), Capabilities.NumberInputValueCaps, PreparsedData)

Call DisplayResultOfAPICall("HidP_GetValueCaps")
lstResults.AddItem " ------------------------------------------------------------------------------------------------"
lblCaps.Caption = "Get Caps Ok"

cmdTrans.Enabled = True
cmdReceive.Enabled = True
End Sub

'*****************************************************************************************
'输出报表到   HID
'*****************************************************************************************

Private Sub cmdTrans_Click()


Dim Count As Integer
Dim NumberOfBytesToSend As Long
```

```
Dim NumberOfBytesWritten As Long
Dim SendBuffer() As Byte
ReDim SendBuffer(Capabilities.OutputReportByteLength - 1)

'填写报表数据到数组    SendBuffer
Count = 0
SendBuffer(Count) = 0          '第一個位元組是    Report ID
Count = Count + 1
SendBuffer(Count) = &H55
Count = Count + 1
SendBuffer(Count) = &HAA
Count = Count + 1
SendBuffer(Count) = &H1
Count = Count + 1
SendBuffer(Count) = &H8
Count = Count + 1
SendBuffer(Count) = Val("&H" + txtR1.Text)
Count = Count + 1
SendBuffer(Count) = Val("&H" + txtR2.Text)
Count = Count + 1
SendBuffer(Count) = Val(txtYear.Text)
Count = Count + 1
SendBuffer(Count) = Val(txtMonth.Text)
Count = Count + 1
SendBuffer(Count) = Val(txtDay.Text)
Count = Count + 1
SendBuffer(Count) = Val(txtHour.Text)
Count = Count + 1
SendBuffer(Count) = Val(txtMinute.Text)
Count = Count + 1
SendBuffer(Count) = Val(txtSecond.Text)
Count = Count + 1

'调用   WriteFile 函数,发送报表

NumberOfBytesWritten = 0
Result = WriteFile(HidDevice, SendBuffer(0), CLng(Capabilities.OutputReportByteLength), NumberOfBytesWritten, 0)

Call DisplayResultOfAPICall("WriteFile")
lstResults.AddItem " Output Report" + Str(NumberOfBytesWritten) + " bytes"
End Sub


'*****************************************************************************************
'从 HID读取报表
'注意：以下代码为非重叠调用，必须保证        HID 输出报表
'*****************************************************************************************

Private Sub cmdReceive_click()

Dim Count
Dim NumberOfBytesRead As Long
Dim ReadBuffer() As Byte
ReDim ReadBuffer(Capabilities.InputReportByteLength - 1)

'调用   ReadFile 函数，读取报表
Result = ReadFile(HidDevice, ReadBuffer(0), CLng(Capabilities.InputReportByteLength), NumberOfBytesRead, 0)

Call DisplayResultOfAPICall("ReadFile")
lstResults.AddItem " Input Report" + Str(NumberOfBytesRead) + " bytes"
```

```
'将输入报表的数据填写到显示介面的相应数据域
txtR1.Text = Hex$(ReadBuffer(5))
txtR2.Text = Hex$(ReadBuffer(6))
txtYear.Text = IIf(ReadBuffer(7) < 10, "0" + Trim(Str$(ReadBuffer(7))), Trim(Str$(ReadBuffer(7))))
txtMonth.Text = IIf(ReadBuffer(8) < 10, "0" + Trim(Str$(ReadBuffer(8))), Trim(Str$(ReadBuffer(8))))
txtDay.Text = IIf(ReadBuffer(9) < 10, "0" + Trim(Str$(ReadBuffer(9))), Trim(Str$(ReadBuffer(9))))
txtHour.Text = IIf(ReadBuffer(10) < 10, "0" + Trim(Str$(ReadBuffer(10))), Trim(Str$(ReadBuffer(10))))
txtMinute.Text = IIf(ReadBuffer(11) < 10, "0" + Trim(Str$(ReadBuffer(11))), Trim(Str$(ReadBuffer(11))))
txtSecond.Text = IIf(ReadBuffer(12) < 10, "0" + Trim(Str$(ReadBuffer(12))), Trim(Str$(ReadBuffer(12))))
End Sub

'*****************************************************************************************
'关闭设备，释放资源
'*****************************************************************************************

Private Sub cmdClose_Click()


'调用  CloseHandle关闭 HID

Result = CloseHandle(HidDevice)
Call DisplayResultOfAPICall("CloseHandle (HidDevice)")

'调用  SetupDiDestroyDeviceInfoList和 HidD_FreePreparsedData释放占用的资源

Result = SetupDiDestroyDeviceInfoList(DeviceInfoSet)
Call DisplayResultOfAPICall("DestroyDeviceInfoList")
Result = HidD_FreePreparsedData(PreparsedData)
Call DisplayResultOfAPICall("HidD_FreePreparsedData")

lstResults.Clear
cmdClose.Enabled = False
cmdGetCaps.Enabled = False
cmdTrans.Enabled = False
cmdReceive.Enabled = False
lblHID.Caption = ""
lblCaps.Caption = ""
txtVendorID.Enabled = True
txtProductID.Enabled = True
End Sub

'*****************************************************************************************
'将当前日期和时间填写到界面的数据域
'*****************************************************************************************

Private Sub cmdNow_Click()

txtHour.Text = IIf(Hour(Now()) < 10, "0" + Hour(Now()), Hour(Now()))
txtMinute.Text = IIf(Minute(Now()) < 10, "0" + Trim(Str(Minute(Now()))), Minute(Now()))
txtSecond.Text = IIf(Second(Now()) < 10, "0" + Trim(Str(Second(Now()))), Second(Now()))
txtYear.Text = IIf((Year(Now()) - 2000) < 10, "0" + Trim(Str(Year(Now()) - 2000)), Str(Year(Now()) - 2000))
txtMonth.Text = IIf(Month(Now()) < 10, "0" + Trim(Str(Month(Now()))), Str(Month(Now())))
txtDay.Text = IIf(Day(Now()) < 10, "0" + Trim(Str(Day(Now()))), Day(Now()))
End Sub

'*****************************************************************************************
'获得信息字符串
'*****************************************************************************************
```

```
Private Function GetDataString(Address As Long, Bytes As Long) As String

Dim Offset As Integer
Dim Result$
Dim ThisByte As Byte

For Offset = 0 To Bytes - 1
        Call RtlMoveMemory(ByVal VarPtr(ThisByte), ByVal Address + Offset, 1)
        If (ThisByte And &HF0) = 0 Then
                Result$ = Result$ & "0"
        End If
        Result$ = Result$ & Hex$(ThisByte) & " "
Next Offset

GetDataString = Result$
End Function
```