Windows 主机端与自定义 USB HID 设备通信详解

说明：

- 以下结论都是基于 Windows XP 系统所得出的，不保证在其他系统的适用性。
- 在此讨论的是 HID 自定义设备，对于标准设备，譬如 USB 鼠标和键盘，由于操作系统对其独占，许多操作未必能正确执行。

1． 所使用的典型 Windows API

CreateFile

ReadFile

WriteFile

以下函数是 DDK 的内容：

HidD_SetFeature

HidD_GetFeature

HidD_SetOutputReport

HidD_GetInputReport

其中， CreateFile 用于打开设备； ReadFile 、 HidD_GetFeature 、 HidD_GetInputReport 用于设备到主机方向的数据通信； WriteFile 、HidD_SetFeature 、 HidD_SetOutputReport 用于主机到设备方向的数据通信。鉴于实际应用，后文主要讨论 CreateFile ， WriteFile ， ReadFile ， HidD_SetFeature 四个函数，明白了这四个函数，其它的可以类推之。

2． 几个常见错误

当使用以上 API 时，如果操作失败，调用 GetLastError() 会得到以下常见错误：

6 ： 句柄无效

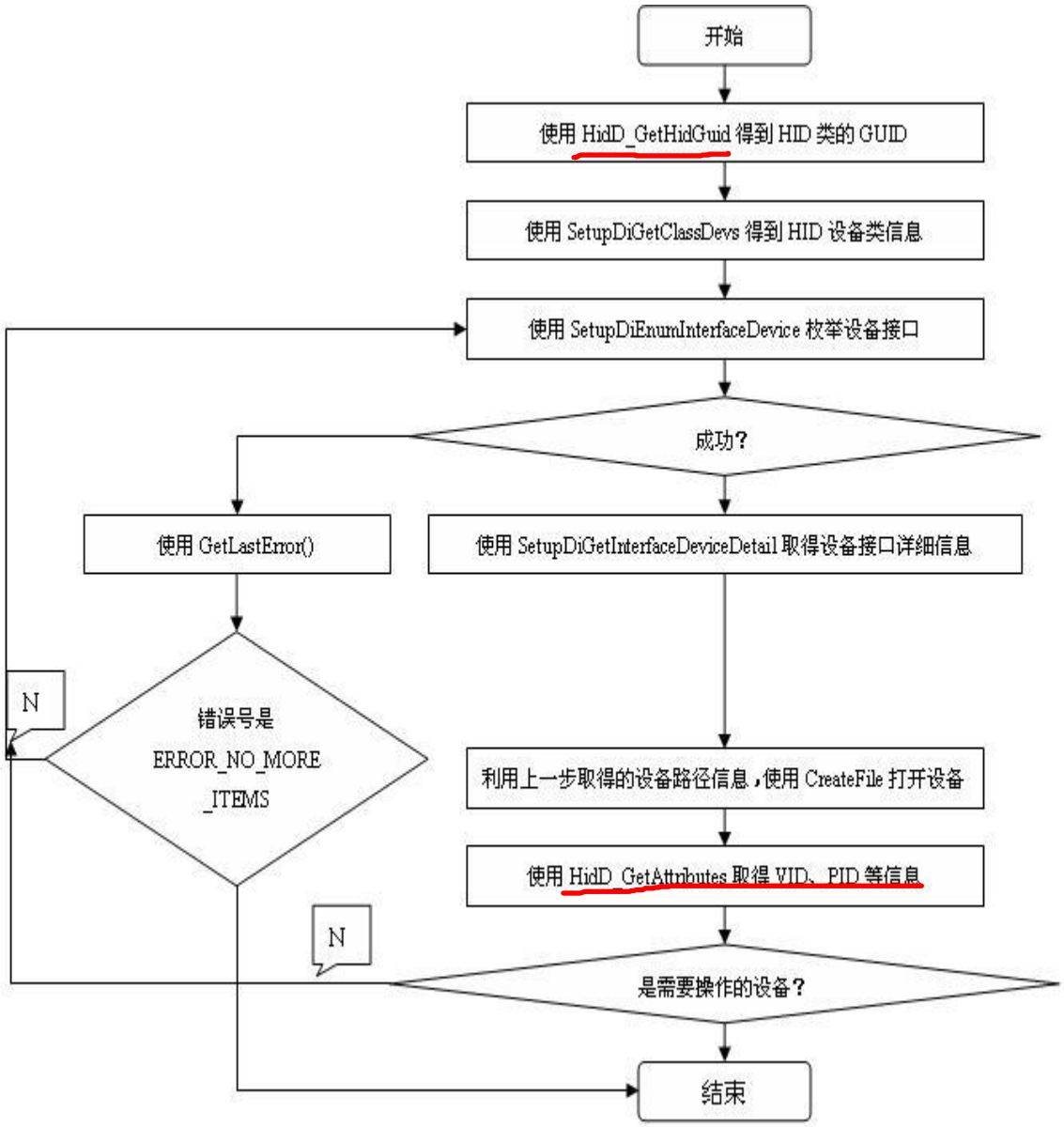
23 ： 数据错误（循环冗余码检查）

87 ： 参数错误

1784 ： 用户提供的 buffer 无效

后文将会详细说明这些错误情况。

3. 主机端设备枚举程序流程



4. 函数使用说明

```
CreateFile(devDetail->DevicePath,                    // 设备路径
           GENERIC_READ | GENERIC_WRITE,             // 访问方式
           FILE_SHARE_READ | FILE_SHARE_WRITE,       // 共享模式
           NULL,
           OPEN_EXISTING,                            // 文件不存在时，返回失败
           FILE_FLAG_OVERLAPPED,                     // 以重叠（异步）模式打开
           NULL);
```

在这里， CreateFile 用于打开 HID 设备，其中设备路径通过函数 SetupDiGetInterfaceDeviceDetail 取得。 CreateFile 有以下几点需要注意：

- 访问方式： 如果是系统独占设备，例如鼠标、键盘等等，应将此参数设置为 0 ，否则后续函数操作将失败（譬如 HidD_GetAttributes ）；也就是说，不能对独占设备进行除了查询以外的任何操作，所以能够使用的函数也是很有限的，下文的一些函数并不一定适合这些设备。在此顺便列出 MSDN 上关于此参数的说明：

If this parameter is zero, the application can query file and device attributes without accessing the device. This is useful if an application wants to determine the size of a floppy disk drive and the formats it supports without requiring a floppy in the drive. It can also be used to test for the file's or directory's existence without opening it for read or write access。

- 重叠（异步）模式：此参数并不会在此处表现出明显的意义，它主要是对后续的 WriteFile ， ReadFile 有影响。如果这里设置为重叠（异步）模式，那么在使用 WriteFile ， ReadFile 时也应该使用重叠（异步）模式，反之亦然。这首先要求 WriteFile ， ReadFile 的最后一个参数不能为空（ NULL ）。否则，便会返回 87 （参数错误）错误号。当然， 87 号错误并不代表就是此参数不正确，更多的信息将在具体讲述这两个函数时指出。此参数为 0 时，代表同步模式，即 WriteFile ， ReadFile 操作会在数据处理完成之后才返回，否则阻塞在函数内部。

```
ReadFile(hDev,                    // 设备句柄，即 CreateFile 的返回值
         recvBuffer,              // 用于接收数据的 buffer
         IN_REPORT_LEN,           // 要读取数据的长度
         &recvBytes,              // 实际收到的数据的字节数
         &ol);                    // 异步模式
```

在这里，ReadFile 用于读取 HID 设备通过中断 IN 传输发来的**输入报告** 。有以下几点要注意：

1 、ReadFile 的调用不会引起设备的任何反应，即 HID 设备与主机之间的中断 IN 传输不与 ReadFile 打交道。实际上主机会在最大间隔时间（由设备的端点描述符来指定）内轮询设备，发出中断 IN 传输的请求。“读取”即意味着从某个 buffer 里面取回数据，实际上这个 buffer 就是 HID 设备驱动中的 buffer 。这个 buffer 的大小可以通过 HidD_SetNumInputBuffers 来改变。在 XP 上缺省值是 32 （个报告）。

2 、读取的数据对象是输入报告，也即通过中断输入管道传入的数据。所以，如果设备不支持中断 IN 传输，那么是无法使用此函数来得到预期结果的。实际上这种情况不可能在 HID 中出现，因为协议指明了至少要有一个中断 IN 端点。

3 、IN_REPORT_LEN 代表要读取的数据的长度（实际的数据正文 + 一个 byte 的报告 ID ），这里是一个常数，主要是因为设备固件的信息我是完全知道的，当然知道要读取多少数据（也就是报告的长度）；不过也可以通过另外的函数（ HidD_GetPreparsedData ）来事先取得报告的长度，这里不做详细讨论。因为很难想象在不了解固件信息的情况下来做自定义设备的 HID 通信，在实际应用中一般来说就是固件与 PC 程序匹配着来开发。此参数如果设置过大，不会有实质性的错误，在 recvBytes 参数中会输出实际读到的长度；如果设置过小，即小于报告的长度，会返回 1784 号错误（用户提供的 buffer 无效）。

4 、关于异步模式。前面已经提过，此参数的设置必须与 CreateFile 时的设置相对应，否则会返回 87 号错误（参数错误）。如果不需要异步模式，此参数需置为 NULL 。在这种情况下， ReadFile 会一直等待直到数据读取成功，所以会阻塞住程序的当前过程。

```
WriteFile(hDev,                    // 设备句柄，即 CreateFile 的返回值
          reportBuf,               // 存有待发送数据的 buffer
          OUT_REPORT_LEN,          // 待发送数据的长度
          &sendBytes,              // 实际收到的数据的字节数
          &ol);                    // 异步模式
```

在这里， WriteFile 用于传输一个**输出报告** 给 HID 设备。有以下几点要注意：

1、 与 ReadFile 不同， WriteFile 函数被调用后，虽然也是经过驱动程序，但是最终会反映到设备中。也就是说，调用 WriteFile 后，设备会接收到输出报告的请求。如果设备使用了中断 OUT 传输，则 WriteFile 会通过中断 OUT 管道来进行传输；否则会使用 SetReport 请求通过控制管道来传输。

2、OUT_REPORT_LEN 代表要写入的数据长度（实际的数据正文 + 一个 byte 的报告 ID ）。如果大于实际报告的长度，则使用实际报告长度；如果小于实际报告长度，会返回 1784 号错误（用户提供的 buffer 无效）。

3、 reportBuf [0] 必须存有待发送报告的 ID ，并且此报告 ID 指示的必须是输出报告，否则会返回 87 号错误（参数错误）。这种情况可能容易被程序员忽略，结果不知错误号所反映的是什么，网上也经常有类似疑问的帖子。顺便指出，输入报告、输入报告、特征报告这些报告类型，是反映在 HID 设备的报告描述符中。后文将做举例讨论。

4、 关于异步模式。前面已经提过，此参数的设置必须与 CreateFile 时的设置相对应，否则会返回 87 号错误（参数错误）。如果不需要异步模式，此参数需置为 NULL 。在这种情况下， WriteFile 会一直等待直到数据读取成功，所以会阻塞住程序的当前过程。

```
HidD_SetFeature(hDev,                          // 设备句柄，即 CreateFile 的返回值
                reportBuf,                     // 存有待发送数据的 buffer
                FEATURE_REPORT_LEN);           //buffer 的长度
HidD_SetOutputReport(hDev,                     // 设备句柄，即 CreateFile 的返回值
                reportBuf,                     // 存有待发送数据的 buffer
                OUT_REPORT_LEN);               //buffer 的长度
```

HidD_SetFeature 发送一个**特征报告** 给设备， HidD_ SetOutputReport 发送一个**输出报告** 给设备。注意以下几点：

1、 跟 WriteFile 类似，必须在 reportBuf [0] 中指明要发送的报告的 ID ，并且和各自适合的类型相对应。也就是说， HidD_SetFeature 只能发送特征报告，因此报告 ID 必须是特征报告的 ID ； HidD_SetOutputReport 只能发送输出报告，因此报告 ID 只能是输出报告的 ID 。

2、 这两个函数最常返回的错误代码是 23 （数据错误）。包括但不仅限于以下情况：

- 报告 ID 与固件描述的不符。
- 传入的 buffer 长度少于固件描述的报告的长度。

据有关资料反映（非官方文档），只要是驱动程序对请求无反应，都会产生此错误。

5. 常见错误汇总

- HID ReadFile
  - Error Code 6 (handle is invalid)
    传入的句柄无效
  - Error Code 87 (参数错误)
    很可能是 createfile 时声明了异步方式，但是读取时按同步读取。
  - Error Code 1784 (用户提供的 buffer 无效):
    传参时传入的“读取 buffer 长度”与实际的报告长度不符。

- HID WriteFile
  - Error Code 6 (handle is invalid)
    传入的句柄无效
  - Error Code 87 （参数错误）
    - CreateFile 时声明的同步 / 异步方式与实际调用 WriteFile 时传入的不同。
    - 报告 ID 与固件中定义的不一致（ buffer 的首字节是报告 ID ）
  - Error Code 1784 (用户提供的 buffer 无效)
    传参时传入的“写入 buffer 长度”与实际的报告长度不符。

- HidD_SetFeature

- HidD_SetOutputReport
  - Error Code 1 (incorrect function)
    不支持此函数，很可能是设备的报告描述符中未定义这样的报告类型（输入、输出、特征）
  - Error Code 6 (handle is invalid)
    传入的句柄无效
  - Error Code 23 （数据错误（循环冗余码检查））
    - 报告 ID 与固件中定义的不相符（ buffer 的首字节是报告 ID ）
    - 传入的 buffer 长度少于固件定义的报告长度（报告正文 +1byte, 1byte 为报告 ID ）
    - 据相关资料反映（非官方文档），只要是驱动程序不接受此请求（对请求无反应），都会产生此错误

6. 报告描述符及数据通信程序示例

报告描述符（注意表中的每个数据都占 1 个字节）：

```
const uint8_t CustomHID_ReportDescriptor[CUSTOMHID_SIZ_REPORT_DESC] =


{

    0x05, 0x8c, /* USAGE_PAGE (ST Page) */

    0x09, 0x01, /* USAGE (Demo Kit) */

    0xa1, 0x01, /* COLLECTION (Application) */



    // The Input report

    0x09,0x03, // USAGE ID - Vendor defined

    0x15,0x00, // LOGICAL_MINIMUM (0)

    0x26,0x00, 0xFF, // LOGICAL_MAXIMUM (255)

    0x75,0x08, // REPORT_SIZE (8bit)

    0x95,0x40, // REPORT_COUNT (64Byte)

    0x81,0x02, // INPUT (Data,Var,Abs)
```

```
    // The Output report
    0x09,0x04, // USAGE ID - Vendor defined
    0x15,0x00, // LOGICAL_MINIMUM (0)
    0x26,0x00,0xFF, // LOGICAL_MAXIMUM (255)
    0x75,0x08, // REPORT_SIZE (8bit)
    0x95,0x40, // REPORT_COUNT (64Byte)
    0x91,0x02, // OUTPUT (Data,Var,Abs)

    0xc0 /* END_COLLECTION */
}; /* CustomHID_ReportDescriptor */
```

下面用一个简单的示例来描述 PC 端与 USB HID 设备进行通信的一般方法。

```
void HIDSampleFunc()
{
    HANDLE      hDev;
    BYTE        recvDataBuf[1024];
    BYTE        reportBuf[1024];
    DWORD       bytes;
    hDev = OpenMyHIDDevice(0); // 打开设备，不使用重叠（异步）方式 ;
    if (hDev == INVALID_HANDLE_VALUE){
        printf("INVALID_HANDLE_VALUE\n");
        return;
    }
    reportBuf[0] = 0; // 输出报告的报告 ID 是 0
```

```
    for(int i=0;i<REPORT_COUNT;i++){
        reportBuf[i+1]=i+1;
    }
    if (!WriteFile(hDev, reportBuf, REPORT_COUNT+1, &bytes, NULL)){// 写入数据到设备
        printf("write data error! %d\n",GetLastError());
        return;
    }
    if(!ReadFile(hDev, recvDataBuf, REPORT_COUNT+1, &bytes, NULL)){ // 读取设备发给主机的数据
        printf("read data error! %d\n",GetLastError());
        return;
    }
    for(int i=0;i<REPORT_COUNT;i++){
        printf("0x%02X ",recvDataBuf[i+1]);
    }
    printf("\n\r");
}
HANDLE OpenMyHIDDevice(int overlapped)
{
    HANDLE hidHandle;
    GUID hidGuid;
    HidD_GetHidGuid(&hidGuid);
    HDEVINFO hDevInfo = SetupDiGetClassDevs(&hidGuid,NULL,NULL,(DIGCF_PRESENT |
DIGCF_DEVICEINTERFACE));
    if (hDevInfo == INVALID_HANDLE_VALUE)
    {
        return INVALID_HANDLE_VALUE;
```

```
    }

    SP_DEVICE_INTERFACE_DATA devInfoData;

    devInfoData.cbSize = sizeof (SP_DEVICE_INTERFACE_DATA);

    int deviceNo = 0;

    SetLastError(NO_ERROR);

    while (GetLastError() != ERROR_NO_MORE_ITEMS)

    {

        if (SetupDiEnumInterfaceDevice (hDevInfo,0,&hidGuid,deviceNo,&devInfoData))

        {

            ULONG  requiredLength = 0;

            SetupDiGetInterfaceDeviceDetail(hDevInfo,

                &devInfoData,

                NULL,

                0,

                &requiredLength,

                NULL);

            PSP_INTERFACE_DEVICE_DETAIL_DATA devDetail =
(SP_INTERFACE_DEVICE_DETAIL_DATA*)malloc(requiredLength);

            devDetail->cbSize = sizeof(SP_INTERFACE_DEVICE_DETAIL_DATA);

            if(!SetupDiGetInterfaceDeviceDetail(hDevInfo,

                &devInfoData,

                devDetail,

                requiredLength,

                NULL,

                NULL))

            {
```

```
                    free(devDetail);
                    SetupDiDestroyDeviceInfoList(hDevInfo);
                    return INVALID_HANDLE_VALUE;
                }
                if (overlapped)
                {
                    hidHandle = CreateFile(devDetail->DevicePath,
                        GENERIC_READ | GENERIC_WRITE,
                        FILE_SHARE_READ | FILE_SHARE_WRITE,
                        NULL,
                        OPEN_EXISTING,
                        FILE_FLAG_OVERLAPPED,
                        NULL);
                }
                else
                {
                    hidHandle = CreateFile(devDetail->DevicePath,
                        GENERIC_READ | GENERIC_WRITE,
                        FILE_SHARE_READ | FILE_SHARE_WRITE,
                        NULL,
                        OPEN_EXISTING,
                        0,
                        NULL);
                }
                free(devDetail);
```

```
            if (hidHandle==INVALID_HANDLE_VALUE)
            {
                SetupDiDestroyDeviceInfoList(hDevInfo);
                free(devDetail);
                return INVALID_HANDLE_VALUE;
            }
            _HIDD_ATTRIBUTES hidAttributes;
            if(!HidD_GetAttributes(hidHandle, &hidAttributes))
            {
                CloseHandle(hidHandle);
                SetupDiDestroyDeviceInfoList(hDevInfo);
                return INVALID_HANDLE_VALUE;
            }
            if (USB_VID == hidAttributes.VendorID
                && USB_PID  == hidAttributes.ProductID)
            {
                printf("找到了我想要的设备，哈哈哈....\n");
                break;
            }
            else
            {
                CloseHandle(hidHandle);
                ++deviceNo;
            }
        }
```

```
    }

    SetupDiDestroyDeviceInfoList(hDevInfo);

    return hidHandle;

}
```