VC++ USB HID + C8051F340 入门笔记

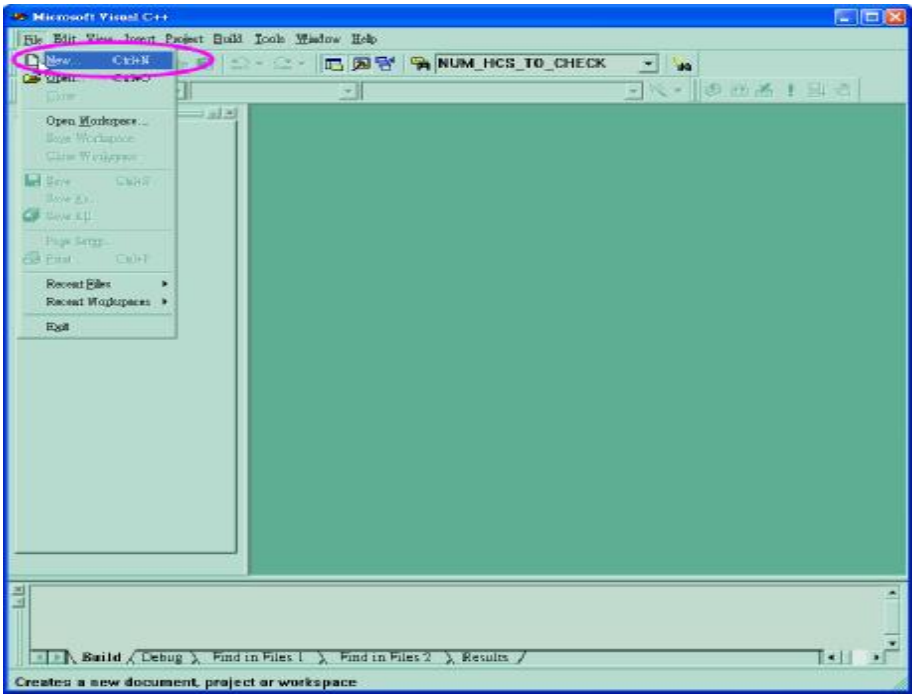
岳生生（2009-12-16）

Note：哈哈，完成 USB HID 的单片机编程后，就想用 VC++编写一个上位机，通信通信。终于功夫不负有心人，成功了。从接触 VC++到编写出这个上位机，确实碰到了很多挫折。因此我写了这个笔记，希望对想入门 VC++ USB HID 编程的朋友一些帮助。

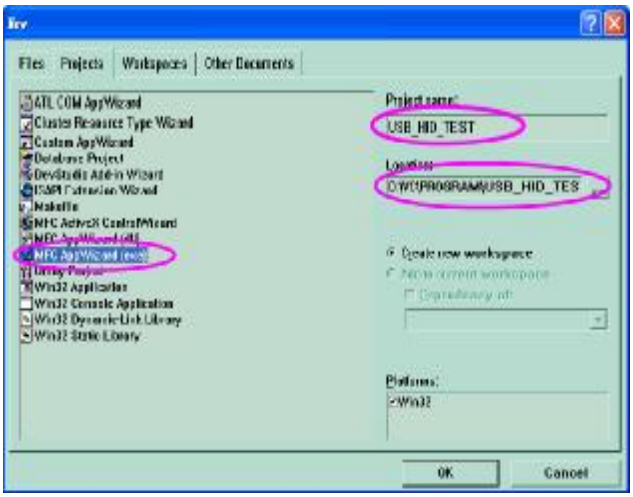
e-mail:yss133_dpj@163.com

目录

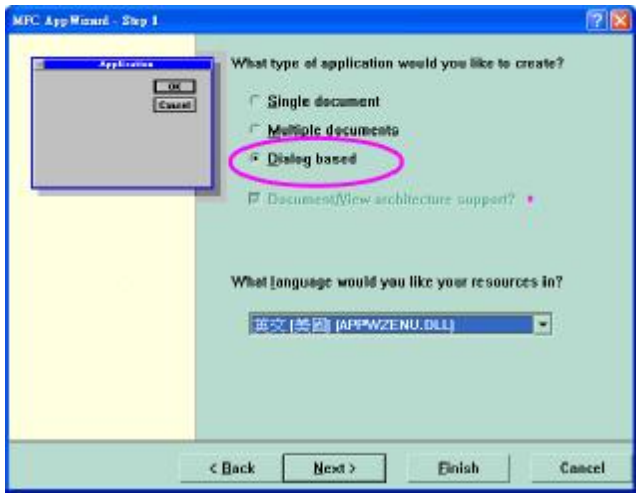
1 建立工程



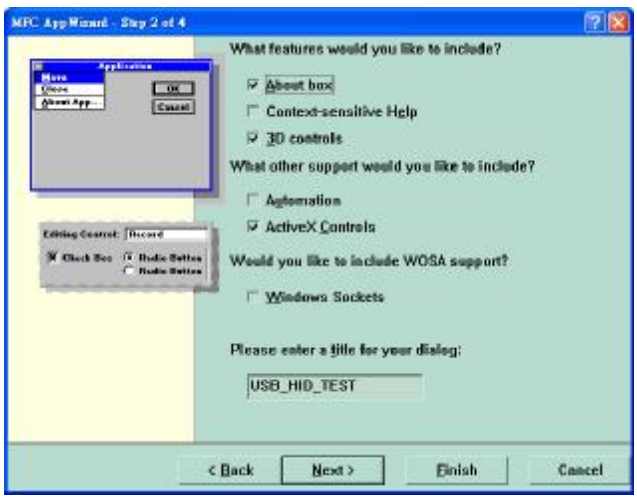
1-1 file à New…



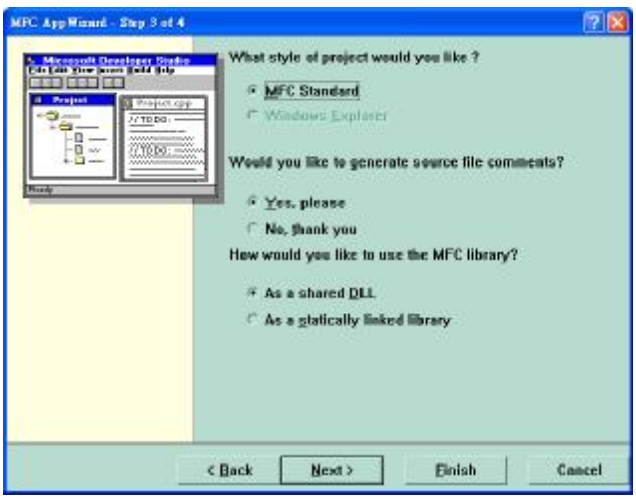
1-2 选择 MFC AppWizard[exe]，输入工程名及位置



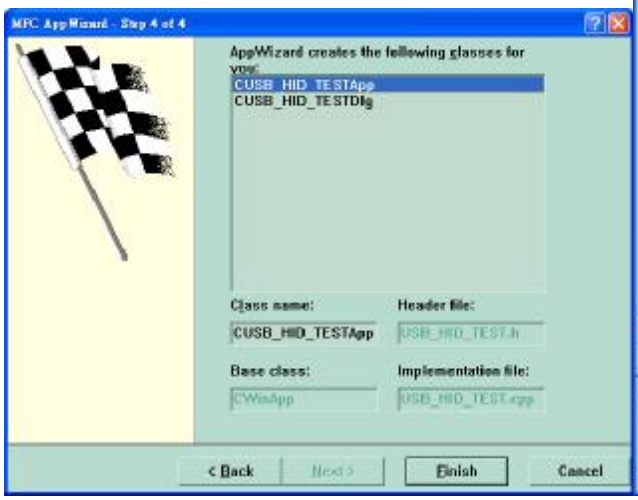
1-3 选择 Dialog based



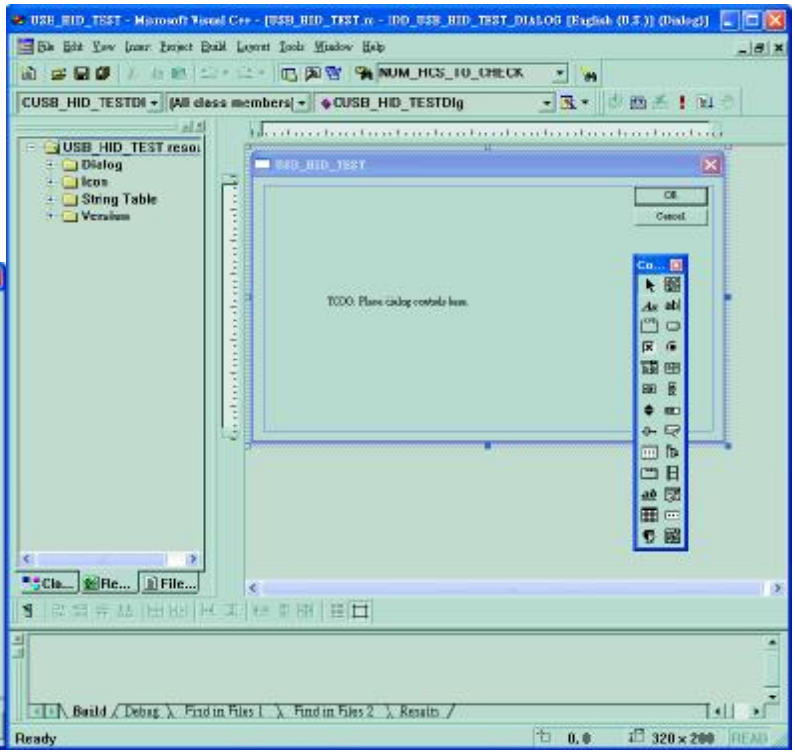
1-4 默认，点击 next



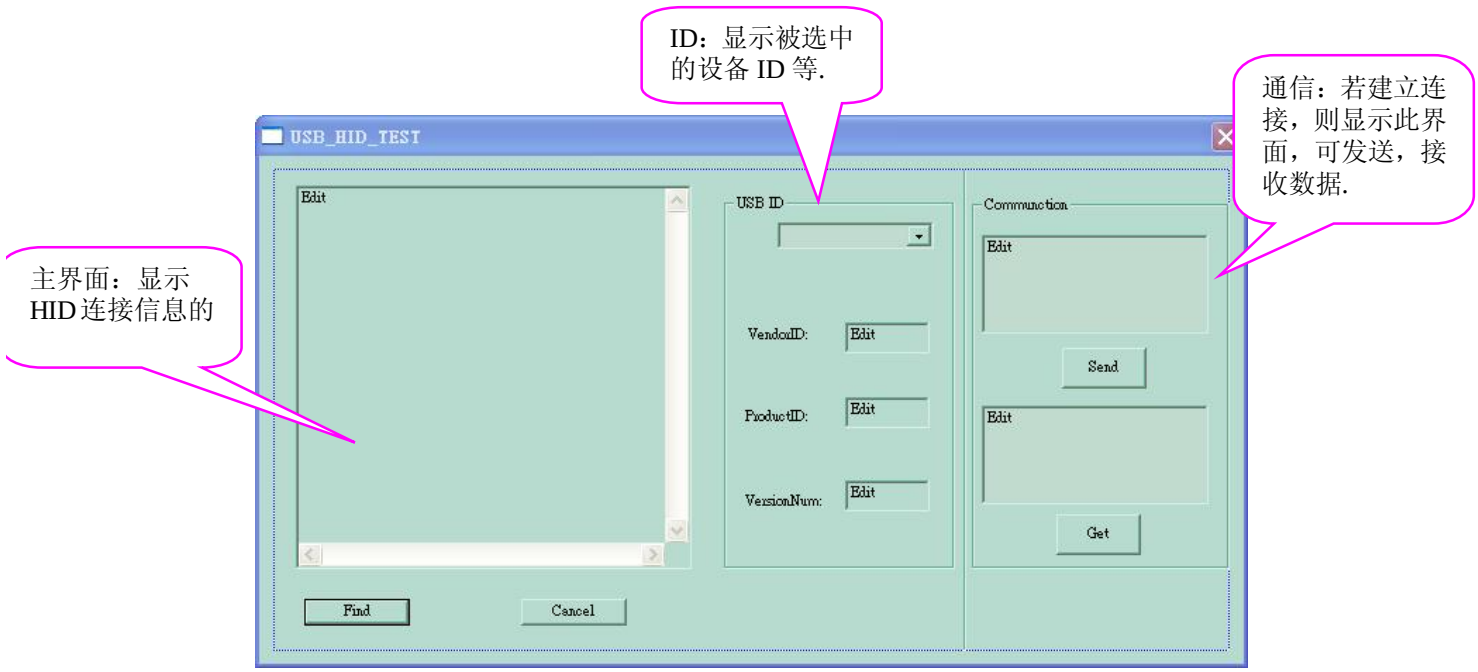
1-5 默认，点击 next



1-6 点击 finish



1-7 生成的工程



1-8 规划界面

界面规划说明：

一开始运行程序，显示除 Communction 外的界面。

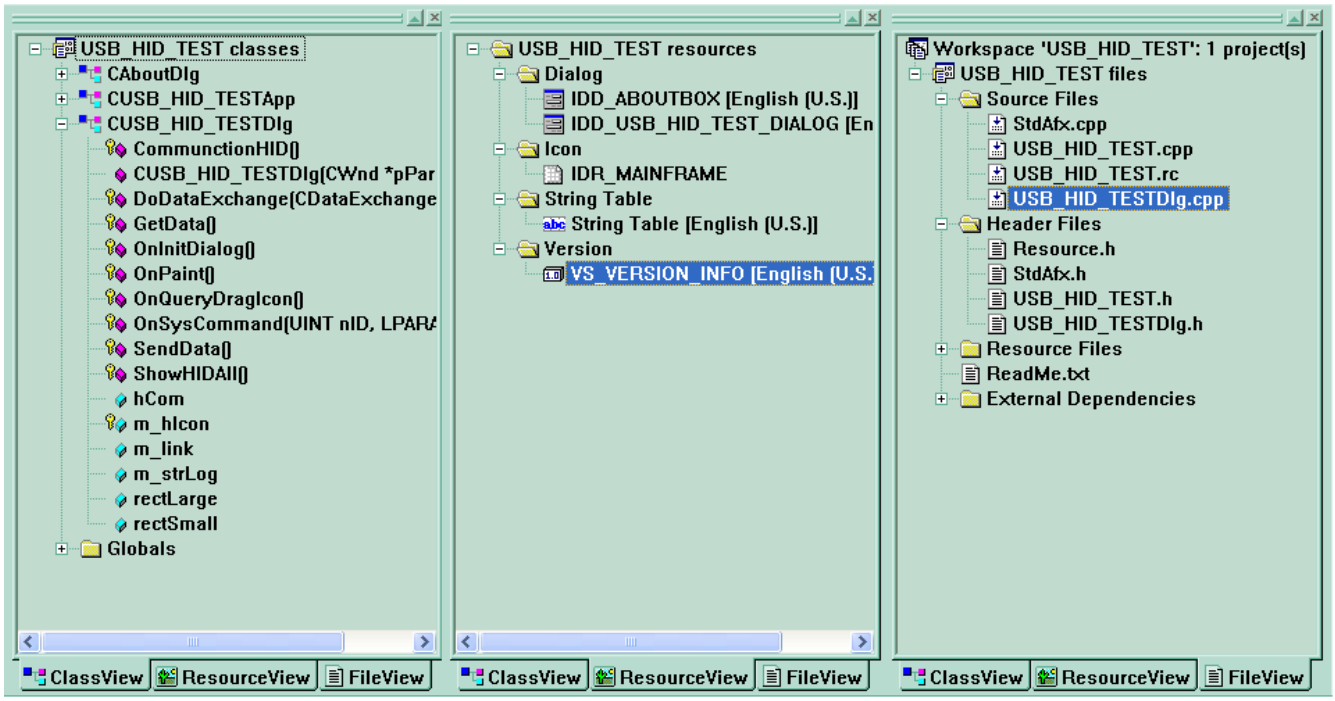
首先点击 find，会查找可以所有的 HID 设备，并获取 ID。

查找到设备后，在 USB ID 栏可以选择，要进行通信的连接。同时会把对应的 ID 显示在对应的文本框中。如果建立连接，则显示 Communction 界面，在此界面中可以进行发送和接收数据。

2 S/W 程序设计

2.1 准备阶段

2.1.1 VC++知识



2.1.1-1 三个视图框

- ClassView 视图列出了此工程用到的类。这里主用到 CUSB_HID_TESTDlg，内部定义对应的函数及变量。其中一些变量的说明：hCom 为 HID 设备通信的句柄。M_strLog 为 CString 型，用于存储显示的信息。rectLarge, rectSmall 为 CRect 类型，用于存储窗口尺寸。
- ResourceView 视图列出界面信息。双击 Dialog 中的 IDD_USB_HID_TEST_DIALOG 可以看到规划的界面。
- FileView 视图显示代码文件。主在 USB_HID_TESTDlg.cpp 中编写。

2.1.2 与 HID 编程有关的准备

需要用到以下 HID API 函数：

- HidD_GetHidGuid(); **à** 获得 GUID
- SetupDiGetClassDevs(); **à** 根据 GUID 查询 HID 的 USB 设备，返回设备信息结构的句柄。参考信息如下网站：http://msdn.microsoft.com/en-us/library/ms792959.aspx
  http://baike.baidu.com/view/1854691.html?fromTaglist
- SetupDiEnumDeviceInterfaces(); **à** 列举设备信息结构中的接口信息，返回 true/false,及接口信息
  http://msdn.microsoft.com/en-us/library/ms791242.aspx
- SetupDiGetDeviceInterfaceDetail(); **à** 获取接口的详细信息，包括路径。返回 true/false .
  http://msdn.microsoft.com/en-us/library/ms792901.aspx
- CreateFile(); **à** 打开设备，建立通信
  - HidD_GetAttributes(); **à** 获得 ID
  - WriteFile(); **à** 给设备发送数据
  - ReadFile(); **à** 从设备中读取数据

由于要用到这些 HID 的 API 函数，因此需要有 hid.lib setupapi.lib 这两个库，头文件 hidsdi.h setupapi.h。

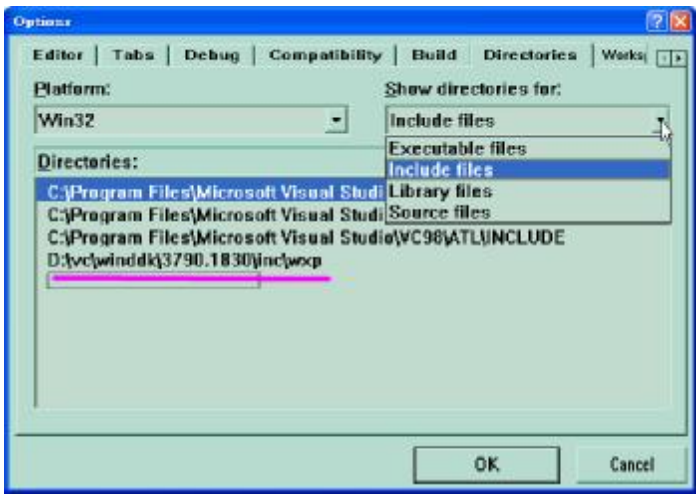
然：获得 winddk 后，在程序中，应如下引用：

```
extern "C"
{
#include "hidsdi.h"
#include "setupapi.h"
}
```

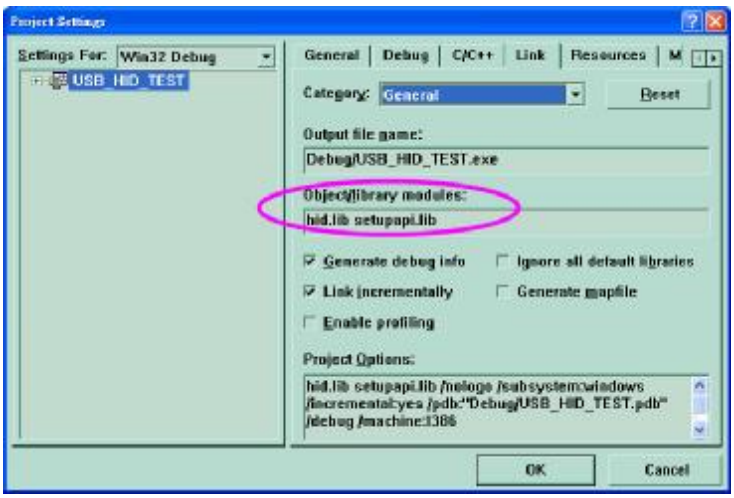
原因：借用网上朋友的说明“（可能该库是 C 风格的，不是 C++风格的。C++风格的可以支持函数重载，所以，编译期间的 C++函数名会加上类型修饰部分，看起来有很多@?这样的字符，而 C 风格的，不支持函数重载，编译期间的函数名与原来的函数名一样。）”

另：添加库位置，tool à options à Directories à show directoryes for:的 include files/library files 处添加对应的文件所在位置。如下图 2.1.2-1。

在 Project à Settings… à Link à Object/library modules:中添加 hid.lib setupapi.lib ,用空格来分隔，如下图 2.1.2-2 所示。



2.1.2-1 添加库/头文件所在目录



2.1.2-2 连接 hid.lib setupapi.lib 库

2.2 程序编写

在 USB_HID_TESTDlg.cpp 文件中的下面位置，添加红色的代码。

```
BEGIN_MESSAGE_MAP(CUSB_HID_TESTDlg, CDialog)
        //{{AFX_MSG_MAP(CUSB_HID_TESTDlg)
        ON_WM_SYSCOMMAND()
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_BN_CLICKED(IDC_FIND, ShowHIDAll)
        ON_BN_CLICKED(IDC_SEND, SendData)
        ON_BN_CLICKED(IDC_GET, GetData)
        ON_CBN_SELCHANGE(IDC_COMBO_LINK, CommunctionHID)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

其中 ON_BN_CLICKED(IDC_FIND, ShowHIDAll)为“Find”按键按下时响应的消息，函数为 ShowHIDAll；其他同理。

在 USB_HID_TESTDlg.h 的 CUSB_HID_TESTDlg 类中添加如下红色代码，

```
protected:
        HICON m_hIcon;
        // Generated message map functions
        //{{AFX_MSG(CUSB_HID_TESTDlg)
        virtual BOOL OnInitDialog();
        afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
        afx_msg void OnPaint();
        afx_msg HCURSOR OnQueryDragIcon();
        afx_msg void ShowHIDAll();
        afx_msg void SendData();
        afx_msg void GetData();
        afx_msg void CommunctionHID();
```

2.2.1 void ShowHIDAll()

void CUSB_HID_TESTDlg::ShowHIDAll()

{

以下分块对此函数经行说明(“n**à**”表示第 n 行)。

```
1 à     m_strLog.Empty();
        //IDC_COMBO_LINK reset
2 à     ((CComboBox*)GetDlgItem(IDC_COMBO_LINK))->ResetContent();

        //initialization text box
3 à     SetDlgItemText(IDC_EDIT_VID,"");
4 à     SetDlgItemText(IDC_EDIT_PID,"");
5 à     SetDlgItemText(IDC_EDIT_VER,"");
6 à     SetDlgItemText(IDC_EDIT_SEND,"");
7 à     SetDlgItemText(IDC_EDIT_GET,"");
        //隐藏 Communction 块的窗口
8 à     SetWindowPos(NULL,0,0,rectSmall.Width (),rectSmall.Height (),
                        SWP_NOMOVE|SWP_NOZORDER);
```

1:清变量 m_strLog

2:清 Combo box 内容

3-7：清文本框内容

8：隐藏 Communction 块的视图界面，因为只有建立起连接要进行通信后才会出现此界面。

```
9 à     m_strLog = _T("Find the GUID of HID...\r\n");
10 à    SetDlgItemText(IDC_EDIT_REPORT,m_strLog);

        // HID  GUID
11 à    GUID guidHID;
12 à    HidD_GetHidGuid(&guidHID);
13 à    m_strLog += _T("HID'GUID:\r\n");

14 à    CString strShow;
15 à    strShow.Format("%08x-%04x-%04x-%02x-%02x-%02x-%02x-%02x-%02x-%02x-%02x\r\n",
                guidHID.Data1,guidHID.Data2,guidHID.Data3,guidHID.Data4[0],
                guidHID.Data4[1],guidHID.Data4[2],guidHID.Data4[3],guidHID.Data4[4],
                guidHID.Data4[5],guidHID.Data4[6],guidHID.Data4[7]);
        // GUID
16 à    m_strLog += strShow;
17 à    SetDlgItemText(IDC_EDIT_REPORT,m_strLog);

        //
18 à    HDEVINFO hDevInfo = SetupDiGetClassDevs(&guidHID,NULL,0,
                DIGCF_PRESENT|DIGCF_DEVICEINTERFACE);
19 à    if(hDevInfo==INVALID_HANDLE_VALUE)
        {
20 à            m_strLog += _T("Find USB device error     \r\n");
21 à            SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
22 à            return;
        }

//      UINT nIndex = 0;
23 à    m_strLog += _T("Finding useful USB device...\r\n");
24 à    SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
```

9-10:在 IDC_EDIT_REPORT 文本框中显示 Find the GUID of HID...

11-12：获得 GUID

13-17：将 GUID 以十六进制形式显示在 IDC_EDIT_REPORT 中

18：根据 GUID 查询 HID 的 USB 设备，返回设备信息结构的句柄，存于 HdevInfo 中

19-22：判断是否获得有效的 USB 设备句柄，如果没有则退出

23-24：否则找到有用的 USB

```
25 à    SP_DEVICE_INTERFACE_DATA strtInterfaceData;
26 à    strtInterfaceData.cbSize=sizeof(SP_DEVICE_INTERFACE_DATA);

27 à    BOOL bSuccess;
        UINT index = 1;
        //
28 à    for(int i=0;i<10;i++)
        {
29 à            bSuccess = SetupDiEnumDeviceInterfaces(hDevInfo,NULL,&guidHID,index, &strtInterfaceData);

30 à            if (!bSuccess)
                {
31 à                    m_strLog += _T("Can't find useful USB device.\r\n");
32 à                    SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
33 à                    SetupDiDestroyDeviceInfoList(hDevInfo);
34 à                    break;
                }
```

25：定义一个 SP_DEVICE_INTERFACE_DATA 类型的变量 strtInterfaceData

26：赋值 strtInterfaceData.cbSize

28：循环查找设备。假设设备不多于 10 个。

29：列举设备信息结构中的接口信息，调用此函数前，应先执行第 26 行“The caller must set *DeviceInterfaceData*.cbSize to sizeof(SP_DEVICE_INTERFACE_DATA) before calling this function.”

30-34：如果没有找到设备接口的消息则退出。

```
35 à   else
36 à           {
37 à                   if(strtInterfaceData.Flags==SPINT_ACTIVE )
                       {
38 à                           strShow.Format("\r\n%d link. \r\n",index);
39 à                           m_strLog +=strShow;
40 à                           SetDlgItemText(IDC_EDIT_REPORT,m_strLog);

41 à                           PSP_DEVICE_INTERFACE_DETAIL_DATA strtDetailData;

42 à                           DWORD strSzie=0,requiesize=0;
43 à                           SetupDiGetDeviceInterfaceDetail(hDevInfo,&strtInterfaceData,NULL,0,
                                                                       &strSzie,NULL);
44 à                           requiesize=strSzie;

45 à                           strtDetailData=(PSP_DEVICE_INTERFACE_DETAIL_DATA)malloc(requiesize);
46 à                           strtDetailData->cbSize=sizeof(SP_DEVICE_INTERFACE_DETAIL_DATA);

47 à                           if (!SetupDiGetDeviceInterfaceDetail(hDevInfo,&strtInterfaceData,
                                                       strtDetailData,strSzie,&requiesize,NULL))
                               {
48 à                                   m_strLog +=_T("Find USB device's path error!\r\n");
49 à                                   SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
50 à                                   SetupDiDestroyDeviceInfoList(hDevInfo);
51 à                                   break;
                               }


                               //
52 à                           strShow.Format("device's path:\r\n%s\r\n\r\n",strtDetailData->DevicePath);
53 à                           m_strLog += strShow;
54 à                           SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
```

35: 否则则找到有效的接口信息

43：获取设备信息的大小，这里应用此函数主要为获得 strSzie 的值，真正调用此函数在 47 行。另此函数的第 3 个参数“**If this parameter is specified, the caller must set *DeviceInterfaceDetailData*.cbSize to sizeof(SP_DEVICE_INTERFACE_DETAIL_DATA) before calling this function**”

47：获取设备路径信息

52-54：显示出设备路径信息

```
55 à                    m_strLog += _T("Open the path...\r\n");
56 à                    SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
57 à
58 à                    //HANDLE hCom = CreateFile(
59 à                     hCom[index] = CreateFile(
60 à                                 strtDetailData->DevicePath,
                                    GENERIC_READ | GENERIC_WRITE,
                                    FILE_SHARE_READ | FILE_SHARE_WRITE,
                                    NULL,
                                    OPEN_EXISTING,
                                    0,//FILE_ATTRIBUTE_NORMAL,
                                    NULL);
                        //
61 à                    if (hCom[index] == INVALID_HANDLE_VALUE)
                        {
                            m_strLog +=_T("Can't open device!\r\n");
                            SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
                            SetupDiDestroyDeviceInfoList(hDevInfo);
                            break;
                        }

                        m_strLog += _T("Open the path is OK!\r\n");
                        SetDlgItemText(IDC_EDIT_REPORT,m_strLog);

                        //
62 à                    HIDD_ATTRIBUTES strtAttrib;
                        m_strLog += _T("Reading the device's ID...\r\n");
                        SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
63 à                    if (!HidD_GetAttributes(hCom[index],&strtAttrib))
                        {
                            m_strLog +=_T("Find the device' ID error!\r\n");
                            SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
                            CloseHandle(hCom[index]);
                            SetupDiDestroyDeviceInfoList(hDevInfo);
                            break;
                        }

                        m_strLog += _T("Read ID OK, as follow:\r\n");
                        SetDlgItemText(IDC_EDIT_REPORT,m_strLog);

                        //
64 à                    strShow.Format("VendorID:0x%X\r\nProductID:0x%X\r\nVersionNumber:0x%X\r\n",
                                    strtAttrib.VendorID,strtAttrib.ProductID,strtAttrib.VersionNumber);
                        m_strLog +=strShow;
                        SetDlgItemText(IDC_EDIT_REPORT,m_strLog);

                        strShow.Format("link %d",index);
65 à                    ((CComboBox*)GetDlgItem(IDC_COMBO_LINK))->AddString(strShow);

66 à                    index++;
                    }
                }
            }
```

59：应用 createfile 函数建立连接，获得通信所用的句柄，以后对设备的读写就通过这个句柄进行。此句柄可以保存起来，在其他任何地方都可以只通过这个句柄进行访问。函数定义见附录。

61：判断是否建立起连接

62-65：建立起连接后，就可以对设备进行读写的。首先获取设备的 PID,VID,VER 资讯。

66：查询下一个设备

}

2.2.2 void CUSB_HID_TESTDlg::CommunctionHID()

{

```
1 à   CString strTemp,strLink;

2 à   m_link.GetLBText(m_link.GetCurSel(),strTemp);

3 à   if(strTemp == "link 1")
      {
            HIDD_ATTRIBUTES strtAttrib;
4 à         if (!HidD_GetAttributes(hCom[1],&strtAttrib))
            {
5 à               CloseHandle(hCom);
                  return;
            }
            CString strShow;
6 à         strShow.Format("0x%X",strtAttrib.VendorID);
7 à         SetDlgItemText(IDC_EDIT_VID,strShow);

8 à         strShow.Format("0x%X",strtAttrib.ProductID);
9 à         SetDlgItemText(IDC_EDIT_PID,strShow);

10 à        strShow.Format("0x%X",strtAttrib.VersionNumber);
11 à        SetDlgItemText(IDC_EDIT_VER,strShow);

            //>><<

12 à        SetWindowPos(NULL,0,0,rectLarge.Width (),rectLarge.Height (),
                  SWP_NOMOVE|SWP_NOZORDER);
      }
```

此函数在组合框选择项目后被运行。

1：

2：获取组合框当前显示的字符存放到 CString 定义的变量中

3：判断是不是选择了要通信的设备（这里固定为 link 1）

4：根据 createfile 函数获得的句柄，再测获得设备的 ID。

5：如果无法获取，则关闭此设备，退出

6-11：把获得的 ID 显示在文本框 IDC_EDIT_VID、IDC_EDIT_PID、IDC_EDIT_VER

12：显示通信的窗口，以便建立数据通信

}

2.2.3 void CUSB_HID_TESTDlg::SendData()

{

发送信息程序，在按了 Send 按键后，运行此程序。此程序把文本框中的数据发送给设备。此与设备的 USB 设置有关系。

```
1 à    CString strVID,strPID,strSend;
2 à    char string[20]={0,0,};

3 à    GetDlgItemText(IDC_EDIT_VID,strVID);
4 à    GetDlgItemText(IDC_EDIT_PID,strPID);
5 à    if ((strVID=="0x6655") && (strPID="0x800A") )
       {
            BOOL bResult=0;
            unsigned long nBytesRead=0;
            unsigned char bOrder[19]={0,0,5,'a','b','c','d','e',};

6 à         GetDlgItemText(IDC_EDIT_SEND,strSend);

7 à         if(strSend.GetLength()<17)
            {
8 à              string[2]=strSend.GetLength();
                 strcpy(&string[3],strSend);
9 à              bResult = WriteFile(hCom[1],string,19,&nBytesRead,NULL);

10 à             if(!bResult && nBytesRead==0)
                 {
                      m_strLog += _T("Send data error!\r\n");
                      SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
                      return;
                 }
11 à             m_strLog += _T("\r\nSend data:");
12 à             SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
13 à             m_strLog += strSend;
14 à             SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
15 à             m_strLog += _T("\r\nSend data OK!\r\n");
16 à             SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
            }
17 à        else
            {
18 à             m_strLog += _T("\r\nMore than 17 character!\r\nPlease input again!!!!\r\n");
                 SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
            }
            SetDlgItemText(IDC_EDIT_SEND,"");
       }
```

5：判断是否是要进行通信的设备（根据 ID 判断）

6：获得要发送的数据

9：应用 writerfile()函数发送客户数据给设备

10：判断发送数据是否成功

11-16：如果发送数据成功，则也把发送的数据显示出来

}

2.2.4 void CUSB_HID_TESTDlg::GetData()

{

在点击 Get 按键后运行此函数，此函数用于获取设备的信息。

```
    CString strVID,strPID,strGet;
    GetDlgItemText(IDC_EDIT_VID,strVID);
    GetDlgItemText(IDC_EDIT_PID,strPID);
1 à if ((strVID=="0x6655") && (strPID=="0x800A") )
    {
        unsigned char bOrder[20];
        BOOL bResult=0;
        unsigned long nBytesRead=0;
2 à     bResult = ReadFile(hCom[1],bOrder,8,&nBytesRead,NULL);
3 à     if(!bResult && nBytesRead==0)
        {
            m_strLog += _T("Get data error!\r\n");
            SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
            return;
        }
        bOrder[nBytesRead]=0;
        strGet=bOrder+1;
        SetDlgItemText(IDC_EDIT_GET,strGet);

        m_strLog += _T("\r\nGet data OK!\r\n");
        SetDlgItemText(IDC_EDIT_REPORT,m_strLog);
    }
```

2：此函数主在这一行，使用 readfile()函数获得设备的信息，其他与 send 类似。

}

3 F/W 程序设计

F/W 为使用 C8051F340 单片机，自定义了一个 HID 设备。其 report 表如下所示：

```
const hid_report_descriptor code HIDREPORTDESC =
{
    0x06,0x00,0x0FF,                    // Usage Page (Generic Desktop)
    0x09, 0x01,                         // Usage (Mouse)
    0xA1, 0x01,                         // Collection (Application)
 //INPUT
      0x09,0x01,
    0x15, 0x00,                         //     Logical Minimum (0)
    0x26,0x0FF,0x00 ,                   //     Logical Maximum (255)
    0x95, 0x07,                         //     Report Count (7)
    0x75, 0x08,                         //     Report Size (8)
    0x81, 0x02,                         //     Input (Data, Variable, Absolute)
 //OUTPUT
      0x09,0x01,
      0x15, 0x00,                                   //     Logical Minimum (0)
      0x26,0x0FF,0x00,                              //     Logical Maximum (255)
      0x75, 0x08,                                   //     Report Size (8)
      0x95, 0x12,                                   //     Report Count (17)
      0x91, 0x02,                                   //     Output (Data, Variable, Absolute)
    0xC0                                    // End Collection (Application)

};
```

注意：设定的 report count 值，必须与 S/W 中使用 readfile(),writefile()给出接收发送数据个数一致，否则接收或发送不了数据。

4 问题：

4.1 一些宏定义

```
BEGIN_MESSAGE_MAP(CUSB_HID_TESTDlg, CDialog)
    //{{AFX_MSG_MAP(CUSB_HID_TESTDlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_FIND, ShowHIDAll)
    ON_BN_CLICKED(IDC_SEND, SendData)
    ON_BN_CLICKED(IDC_GET, GetData)
    ON_CBN_SELCHANGE(IDC_COMBO_LINK, CommunctionHID)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

4.2 combo box 使用

此程序中用到的函数：

```
//清除内容
((CComboBox*)GetDlgItem(IDC_COMBO_LINK))->ResetContent();
//往 combo box 中添加信息
((CComboBox*)GetDlgItem(IDC_COMBO_LINK))->AddString(strShow);
//获得 combo box 当前的信息，存放到 CString 类型的 strTemp 中。
// CComboBox      m_link;的定义
m_link.GetLBText(m_link.GetCurSel(),strTemp);
//函数响应
ON_CBN_SELCHANGE(IDC_COMBO_LINK, CommunctionHID)
```

更多其用法可以从网络获得：

http://www.cppblog.com/Lee7/archive/2008/02/21/43066.aspx

4.3 CString 与 char 的互换

```
char    chartemp[10];
CString    stringtemp;
stringtemp    =    chartemp;          //char-->CString
strcpy(chartemp,stringtemp    )      //CString-->char
```

4.4 字符串中 0 表示结束

如有一个数组 char chartemp[5]={0,3,..},则使用 CString stringtemp= chartemp;得到的 stringtemp 为空。可以 stringtemp = chartemp +1。

4.5 xp 下却不能获得 USB 鼠标的信息

引自网上的说明“是因为鼠标和键盘在系统下是独占的，CreateFile 是不能打开的。”

另：在查找到设备的通道后，要与此设备进行通信，就要根据此通道使用 createfile 函数，获得此设备的句柄。以后，与此设备的通信，就根据 createfile 函数返回的句柄来访问。

5 附录

5.1 SetupDiEnumDeviceInterfaces()

The SetupDiEnumDeviceInterfaces function enumerates the device interfaces that are contained in a device information set.

```
WINSETUPAPI BOOL WINAPI
  SetupDiEnumDeviceInterfaces(
    IN HDEVINFO  DeviceInfoSet,
    IN PSP_DEVINFO_DATA  DeviceInfoData  OPTIONAL,
    IN LPGUID  InterfaceClassGuid,
    IN DWORD  MemberIndex,
    OUT PSP_DEVICE_INTERFACE_DATA  DeviceInterfaceData
    );
```

Parameters

DeviceInfoSet

A pointer to a device information set that contains the device interfaces for which to return information. This handle is typically returned by SetupDiGetClassDevs.

DeviceInfoData

A pointer to an SP_DEVINFO_DATA structure that specifies a device information element in *DeviceInfoSet*. This parameter is optional and can be NULL. If this parameter is specified, SetupDiEnumDeviceInterfaces constrains the enumeration to the interfaces that are supported by the specified device. If this parameter is NULL, repeated calls to SetupDiEnumDeviceInterfaces return information about the interfaces that are associated with all the device information elements in *DeviceInfoSet*. This pointer is typically returned by SetupDiEnumDeviceInfo.

InterfaceClassGuid

A pointer to a GUID that specifies the device interface class for the requested interface.

MemberIndex

A zero-based index into the list of interfaces in the device information set. The caller should call this function first with *MemberIndex* set to zero to obtain the first interface. Then, repeatedly increment *MemberIndex* and retrieve an interface until this function fails and GetLastError returns ERROR_NO_MORE_ITEMS.

If *DeviceInfoData* specifies a particular device, the *MemberIndex* is relative to only the interfaces exposed by that device.

DeviceInterfaceData

A pointer to a caller-allocated buffer that contains, on successful return, a completed SP_DEVICE_INTERFACE_DATA structure that identifies an interface that meets the search parameters. The caller must set *DeviceInterfaceData*.cbSize to sizeof(SP_DEVICE_INTERFACE_DATA) before calling this function.

Return Value

SetupDiEnumDeviceInterfaces returns TRUE if the function completed without error. If the function completed with an error, FALSE is returned and the error code for the failure can be retrieved by calling GetLastError.

5.2 SetupDiGetDeviceInterfaceDetail()

The SetupDiGetDeviceInterfaceDetail function returns details about a device interface.

```
WINSETUPAPI BOOL WINAPI
  SetupDiGetDeviceInterfaceDetail(
    IN HDEVINFO  DeviceInfoSet,
    IN PSP_DEVICE_INTERFACE_DATA  DeviceInterfaceData,
    OUT PSP_DEVICE_INTERFACE_DETAIL_DATA  DeviceInterfaceDetailData..OPTIONAL,
    IN DWORD  DeviceInterfaceDetailDataSize,
    OUT PDWORD  RequiredSize..OPTIONAL,
    OUT PSP_DEVINFO_DATA  DeviceInfoData  OPTIONAL
    );
```

Parameters

DeviceInfoSet

A pointer to the device information set that contains the interface for which to retrieve details. This handle is typically returned by SetupDiGetClassDevs.

DeviceInterfaceData

A pointer to an SP_DEVICE_INTERFACE_DATA structure that specifies the interface in *DeviceInfoSet* for which to retrieve details. A pointer of this type is typically returned by SetupDiEnumDeviceInterfaces.

DeviceInterfaceDetailData

A pointer to an SP_DEVICE_INTERFACE_DETAIL_DATA structure to receive information about the specified interface. This parameter is optional and can be NULL. This parameter must be NULL if *DeviceInterfaceDetailSize* is zero. If this parameter is specified, the caller must set *DeviceInterfaceDetailData*.cbSize to sizeof(SP_DEVICE_INTERFACE_DETAIL_DATA) before calling this function. The cbSize member always contains the size of the fixed part of the data structure, not a size reflecting the variable-length string at the end.

DeviceInterfaceDetailDataSize

The size of the *DeviceInterfaceDetailData* buffer. The buffer must be at least (offsetof(SP_DEVICE_INTERFACE_DETAIL_DATA, DevicePath) + sizeof(TCHAR)) bytes, to contain the fixed part of the structure and a single NULL to terminate an empty MULTI_SZ string.

This parameter must be zero if *DeviceInterfaceDetailData* is NULL.

RequiredSize

A pointer to a variable of type DWORD that receives the required size of the *DeviceInterfaceDetailData* buffer. This size includes the size of the fixed part of the structure plus the number of bytes required for the variable-length device path string. This parameter is optional and can be NULL.

DeviceInfoData

A pointer to a buffer that receives information about the device that supports the requested interface. The caller must set *DeviceInfoData*.cbSize to sizeof(SP_DEVINFO_DATA). This parameter is optional and can be NULL.

Return Value

SetupDiGetDeviceInterfaceDetail returns TRUE if the function completed without error. If the function completed with an error, FALSE is returned and the error code for the failure can be retrieved by calling GetLastError.

5.3 CreateFile()

```
HANDLE CreateFile(
  LPCTSTR lpFileName,          // pointer to name of the file
  DWORD dwDesiredAccess,       // access (read-write) mode
  DWORD dwShareMode,           // share mode
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
                               // pointer to security attributes
  DWORD dwCreationDisposition,  // how to create
  DWORD dwFlagsAndAttributes,  // file attributes
  HANDLE hTemplateFile         // handle to file with attributes to
                               // copy
);
```

5.4 HidD_GetAttributes()

The HidD_GetAttributes routine returns the attributes of a specified top-level collection.

```
BOOLEAN
  HidD_GetAttributes(
    IN HANDLE  HidDeviceObject,
    OUT PHIDD_ATTRIBUTES  Attributes
    );
```

Parameters

HidDeviceObject

Specifies an open handle to a top-level collection.

Attributes

Pointer to a caller-allocated HIDD_ATTRIBUTES structure that returns the attributes of the collection specified by *HidDeviceObject*.

Return Value

HidD_GetAttributes returns TRUE if succeeds; otherwise, it returns FALSE.

5.5 WriteFile()

The WriteFile function writes data to a file and is designed for both synchronous and asynchronous operation. The function starts writing data to the file at the position indicated by the file pointer. After the write operation has been completed, the file pointer is adjusted by the number of bytes actually written, except when the file is opened with FILE_FLAG_OVERLAPPED. If the file handle was created for overlapped input and output (I/O), the application must adjust the position of the file pointer after the write operation is finished.

```
BOOL WriteFile(
  HANDLE hFile,                // handle to file to write to
  LPCVOID lpBuffer,            // pointer to data to write to file
  DWORD nNumberOfBytesToWrite, // number of bytes to write
  LPDWORD lpNumberOfBytesWritten, // pointer to number of bytes written
  LPOVERLAPPED lpOverlapped    // pointer to structure for overlapped I/O
);
```

Parameters

hFile

Handle to the file to be written to. The file handle must have been created with GENERIC_WRITE access to the file.

lpBuffer

Pointer to the buffer containing the data to be written to the file.

nNumberOfBytesToWrite

Number of bytes to write to the file.

A value of zero specifies a null write operation. A null write operation does not write any bytes but does cause the time stamp to change.

Named pipe write operations across a network are limited to 65,535 bytes.

lpNumberOfBytesWritten

Pointer to the number of bytes written by this function call. WriteFile sets this value to zero before doing any work or error checking.

lpOverlapped

Pointer to an OVERLAPPED structure. This structure is required if *hFile* was opened with FILE_FLAG_OVERLAPPED.

If *hFile* was opened with FILE_FLAG_OVERLAPPED, the *lpOverlapped* parameter must not be NULL. It must point to a valid OVERLAPPED structure. If *hFile* was opened with FILE_FLAG_OVERLAPPED and *lpOverlapped* is NULL, the function can incorrectly report that the write operation is complete.

If *hFile* was opened with FILE_FLAG_OVERLAPPED and *lpOverlapped* is not NULL, the write operation starts at the offset specified in the OVERLAPPED structure and WriteFile may return before the write operation has been completed. In this case, WriteFile returns FALSE and the GetLastError function returns ERROR_IO_PENDING. This allows the calling process to continue processing while the write operation is being completed. The event specified in the OVERLAPPED structure is set to the signaled state upon completion of the write operation.

If *hFile* was not opened with FILE_FLAG_OVERLAPPED and *lpOverlapped* is NULL, the write operation starts at the current file position and WriteFile does not return until the operation has been completed.

If *hFile* was not opened with FILE_FLAG_OVERLAPPED and *lpOverlapped* is not NULL, the write operation starts at the offset specified in the OVERLAPPED structure and WriteFile does not return until the write operation has been completed.

5.6 ReadFile()

The ReadFile function reads data from a file, starting at the position indicated by the file pointer. After the read operation has been completed, the file pointer is adjusted by the number of bytes actually read, unless the file handle is created with the overlapped attribute. If the file handle is created for overlapped input and output (I/O), the application must adjust the position of the file pointer after the read operation.

```
BOOL ReadFile(
  HANDLE hFile,                // handle of file to read
  LPVOID lpBuffer,             // pointer to buffer that receives data
  DWORD nNumberOfBytesToRead,  // number of bytes to read
  LPDWORD lpNumberOfBytesRead, // pointer to number of bytes read
  LPOVERLAPPED lpOverlapped    // pointer to structure for data
);
```

6 程序

VC++ USB HID 程序：

USB_HID_TEST.zip

C8051F34X USB 自定义 hid 程序：

Usb_hid.zip

7 参考资料：

1) 《用 VC++编写 USB 接口通信程序.pdf》 “programming of USB Interface Communication in VC++” ; 作者：陈俊 李治（成都西南交通大学电气工程学院）

2) MSDN ver4.73

3) http://msdn.microsoft.com/en-us/library/default.aspx

4) 还有些网上的资料

5) 孙鑫的《VC++从入门到精通开发详解视频教程》