

如何编写应用程序与 USB HID 设备通讯（读写 USB HID 设备）

说明：本实例所使用的上位机程序开发工具为 Visual C++6.0。

一、修改下位机固件程序

我们如果想实现一个 USB 的 HID 类设备，不需要在 Windows 下开发自己的驱动程序。HID 不一定要是标准的外设类型，唯一的要求是交换的数据存储在报文的结构内，设备固件必须支持报文的格式。任何工作在该限制之内的设备都可以成为一个 HID，例如温度计、电压计、读卡机等。

报文的格式是由报告描述符决定的，所以只要修改描述符就能实现我们需要的报文格式。下面我们来实现一个简单的报文格式：上位机发送固定 64 字节数据给设备，这个数据可以是命令，也可能是数据，具体含义并不是由报告描述符来决定的，是由开发人员事先约定好的。设备返回的数据也是 64 个字节，同样这个数据流的每个字节（甚至每个位）的具体含义由开发人员事先约定好。

1、修改报告描述符

A、在 Descriptor.C 中找到以 MouseReportDescriptor 函数，将其内容修改如下：

```
1.  code char MouseReportDescriptor[29] = {  
2.      0x06,0x00,0xFF,           //USAGE_PAGE (Vendor Defined Page 1)  
3.      0x09,0x01,           //USAGE (Vendor Usage 1)  
4.      0xA1,0x01,           //COLLECTION (Application)  
5.  
6.      0x19,0x01,           //(Vendor Usage 1)  
7.      0x29,0x08,           //(Vendor Usage 1)  
8.      0x15,0x00,           //LOGICAL_MINIMUM (0)  
9.      0x26,0xFF,0x00,       //LOGICAL_MAXIMUM (255)  
10.     0x75,0x08,           //REPORT_SIZE (8)  
11.     0x95,0x40,           //REPORT_COUNT (64)  
12.     0x81,0x02,           //INPUT (Data,Var,Abs)  
13.  
14.     0x19,0x01,           //(Vendor Usage 1)  
15.     0x29,0x08,           //(Vendor Usage 1)  
16.     0x91,0x02,           //OUTPUT (Data,Var,Abs)  
17.  
18.     0xC0                 // END_COLLECTION  
19. };
```

此报告描述符定义了 64 个字节的输入输出数据。

B、将 Descriptor.C 中的如下代码

```
1.      0x66,0x03,          //设备制造商定的产品 ID
```

修改为

```
1.      0x66,0x06,          //设备制造商定的产品 ID
```

C、在 Descriptor.h 中，将以下代码

```
1.      extern code char MouseReportDescriptor[52];
```

修改为

```
1.      extern code char MouseReportDescriptor[29];
```

D、在 Main.C 中找到以下代码

```
1.      if(bEPPflags.bits.configuration)
2.      {
3.          //在这里添加端点操作代码
4.
5.      }
```

将其修改为


```
1.      if(bEPPflags.bits.configuration)
2.      {
3.          //在这里添加端点操作代码
4.
5.          if(bEPPflags.bits.ep2_rxdone ) //主端点接收到数据(从主机发往设备的数据)
6.          {
7.              bEPPflags.bits.ep2_rxdone      = 0;
8.
9.              D12_WriteEndpoint(5,EP2_PACKET_SIZE,EpBuf); //立即将收到的 64 个字节发送到 PC 机
```

```
10.    }
11.    }
```

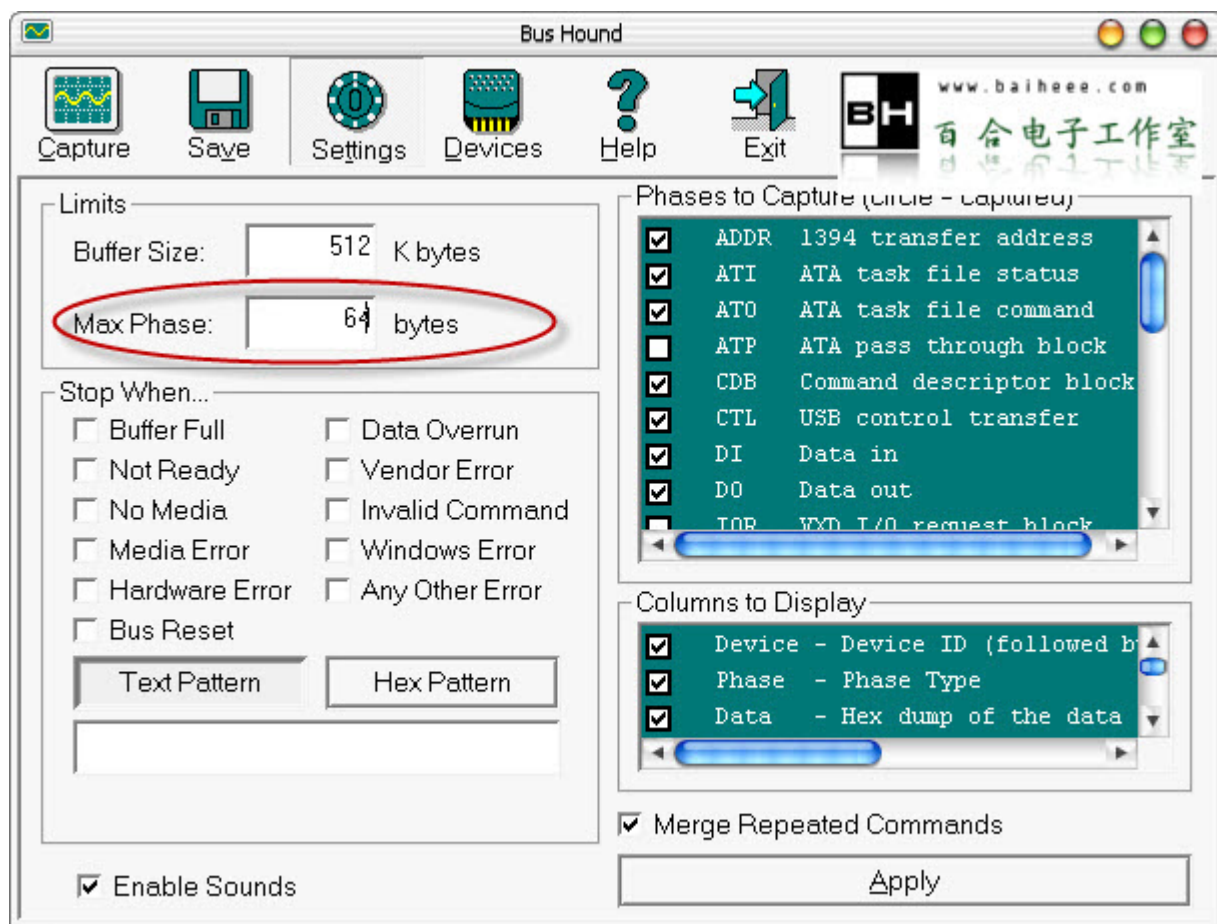
这一步的作用是立即将接到的数据返回给主机。函数 D12_WriteEndpoint 的定义位于 D12CI.C 中。

 [点击这里下载已修改好的源代码](#)

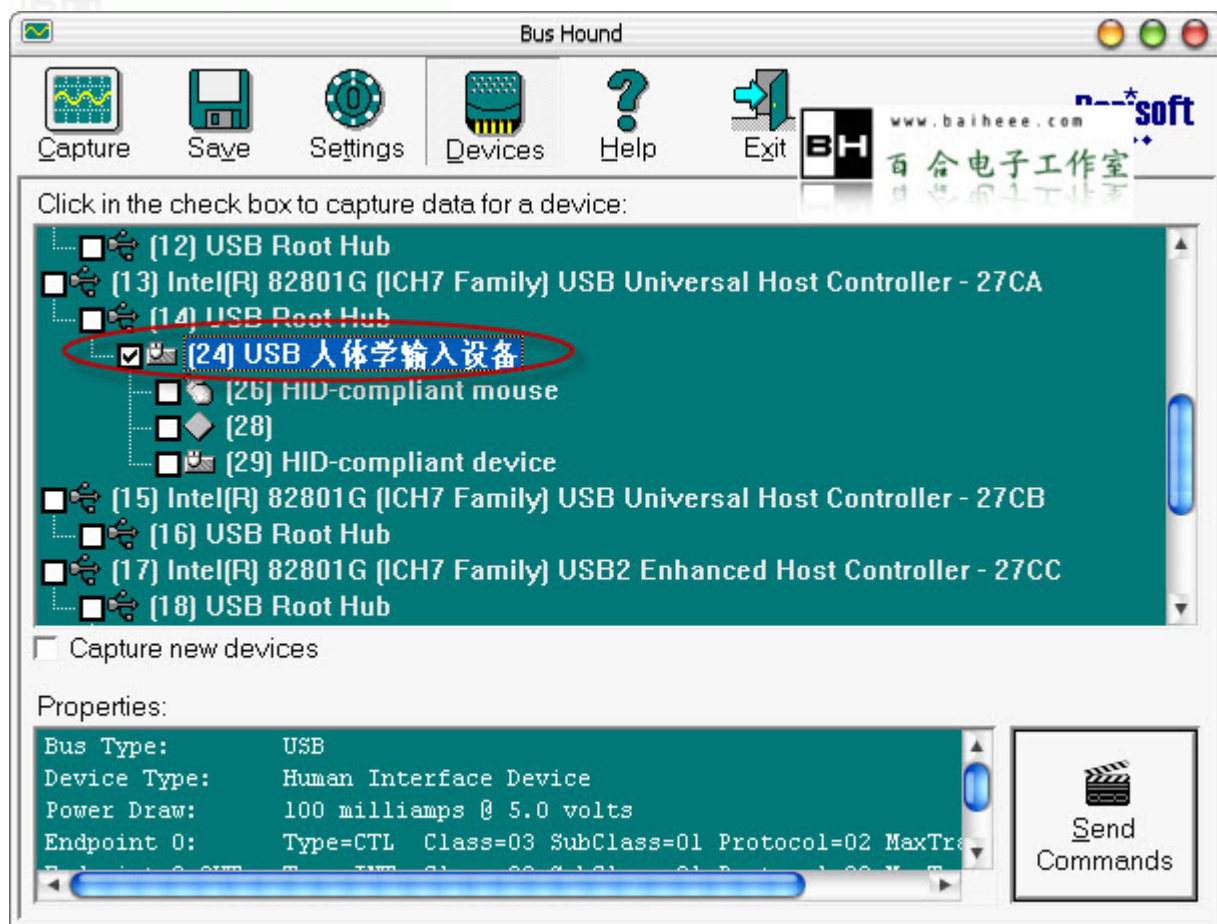
2、测试

我们可以利用一些 USB 调试软件在自己没有编写上位机程序的情况下先进行一些测试。这里我们要用到的工具是  **BUS HOUND**。

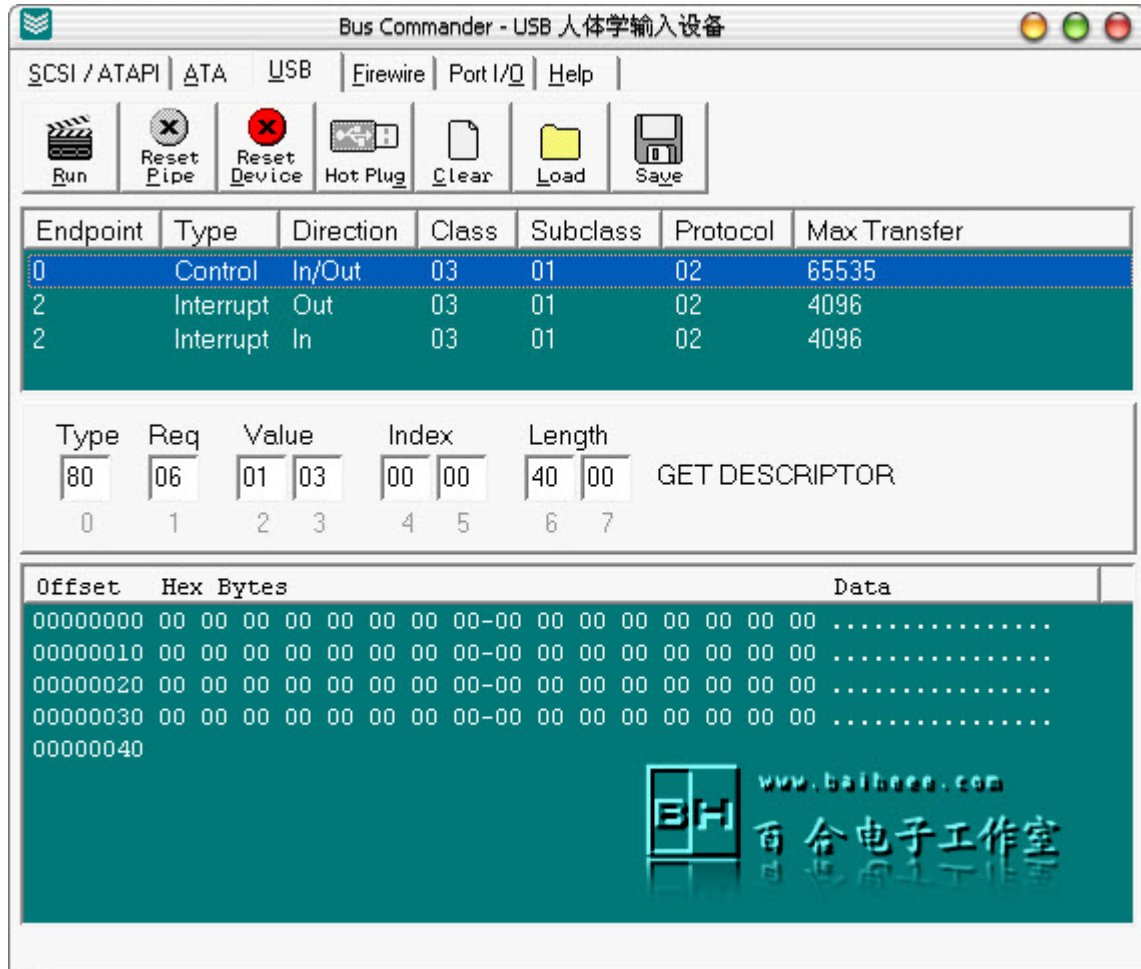
A、首先将第 1 步修改好的程序烧录到主控芯片中并将硬件插入电脑，然后打开 BUS HOUND，点“Settings”按钮切换到设置页，将“Max Phase”的值设为 64 并点“Apply”按钮，这样 BUS HOUND 才能监控最多 64 个字节的数据流。



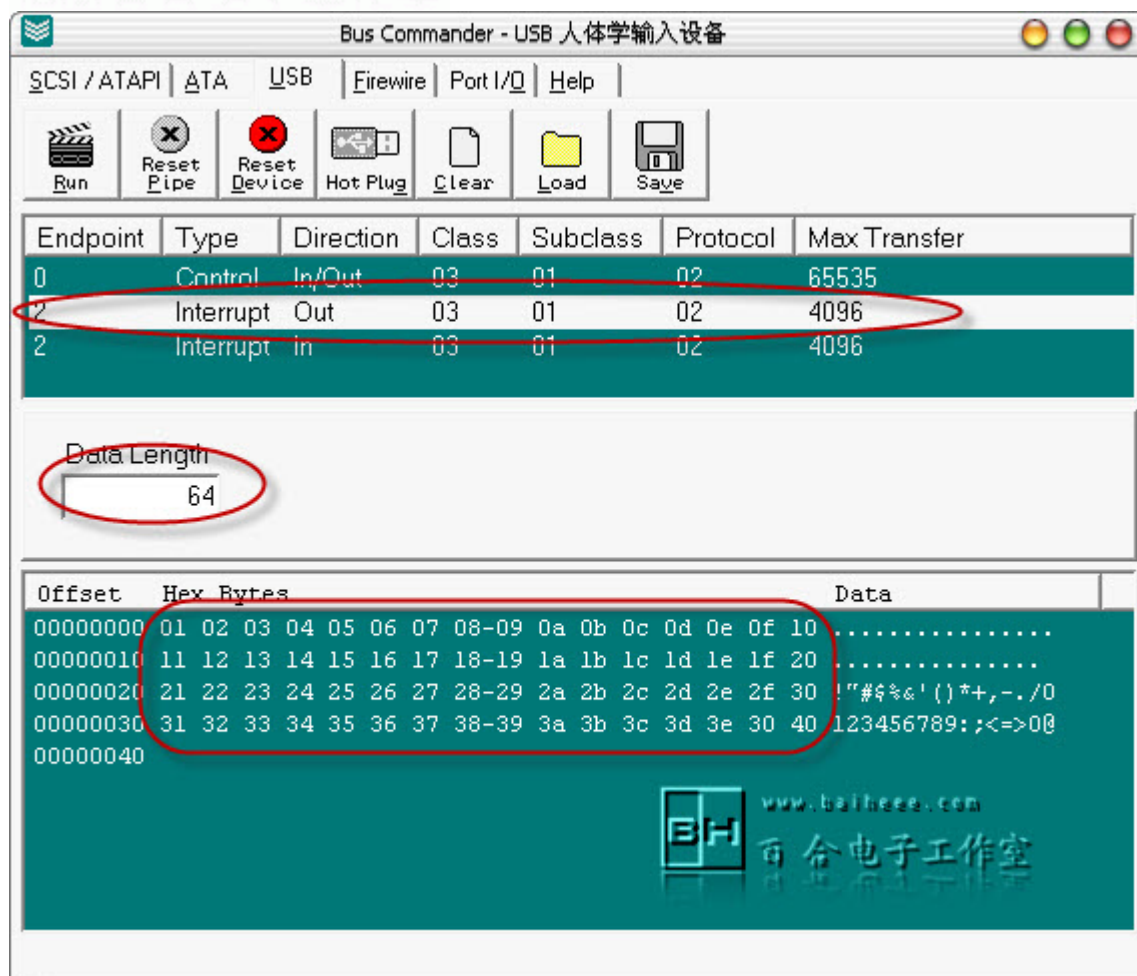
B、点“Devices”按钮切换到 Devices 页，选定我们刚插入的硬件（一定要选紧临“USB Root Hub”的下一级设备）。



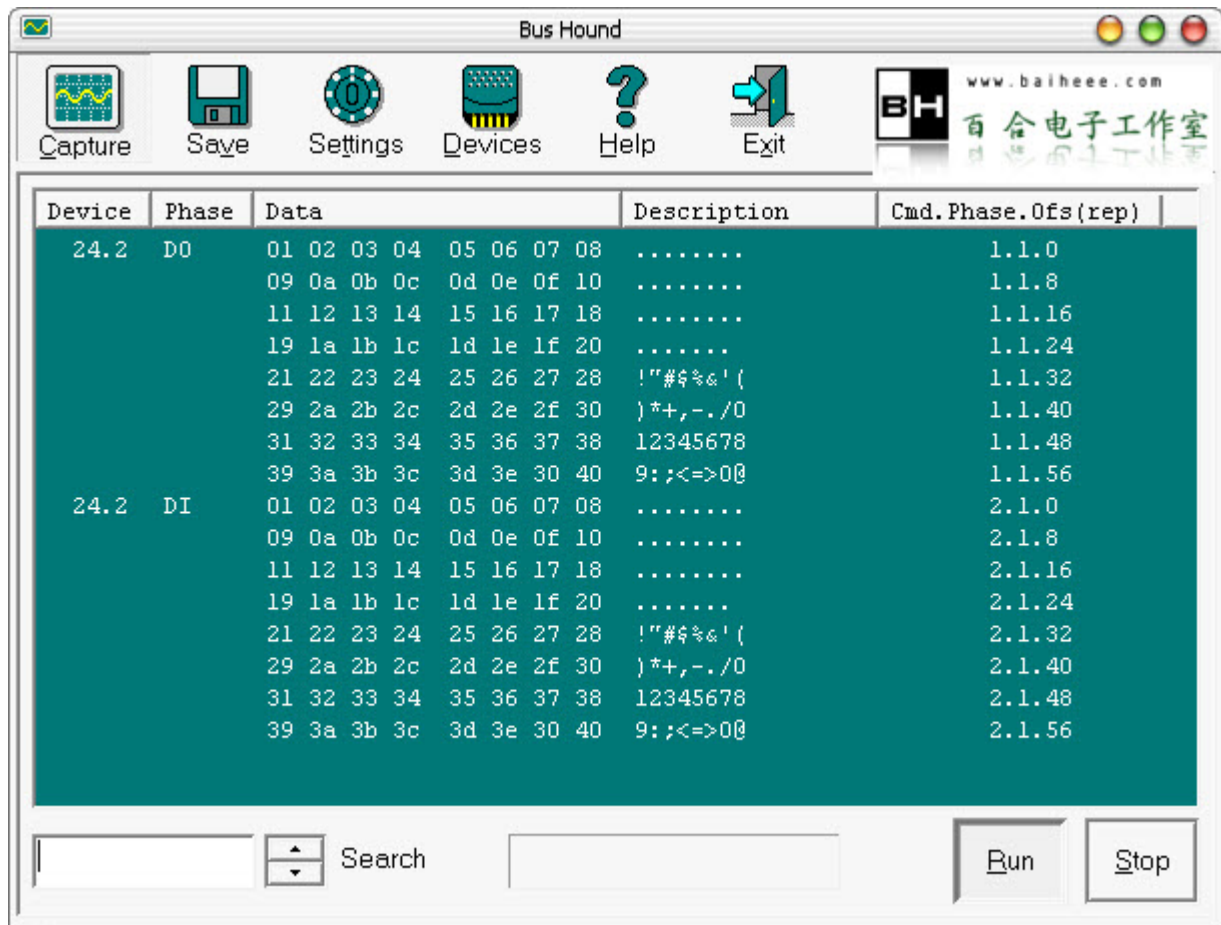
C、点“Send Commands”按钮，出现如下画面



D、选中“Endpoint”为 2，“Direction”为 Out 的那项，“Data Length”填 64，发送的数据随便填上一些数据，最后点击“Run”按钮



E、切换到 BUS HOUND 主界面，点“Capture”按钮切换到监控页面，如下图所示，BUS HOUND 捕捉到了一些数据。其中“Phase”列中的“DO”代表主机发往设备的数据，而“DI”代表设备发往主机的数据。我们看到设备返回的数据正好和主机发送的数据一样，证明我们的下位机程序已经修改成功。



三、读写 HID 设备的步骤

读写 HID 设备步骤如下:

- ①、得到系统 HID 设备结构数组指针
- ②、对设备进行遍历
- ③、得到指定 HID 设备的句柄
- ④、readfile/writefile 进行读写

下面分别对各步骤及其所涉及的相关 API 函数进行介绍。

1、得到设备句柄: 这步用到的两个主要 API 函数原型为:

A、通过以下函数

```
1. VOID HidD_GetHidGuid(OUT LPGUID HidGuid );
```

得到 HID 设备的 GUID。

B、再通过以下函数

```
1. HDEVINFO SetupDiGetClassDevs(  
2.     CONST LPGUID ClassGuid,  
3.     PCTSTR Enumerator,  
4.     HWND hwndParent,  
5.     DWORD Flags  
6. );
```

取得 HID 设备结构数组指针, 以便下步得用这个数组对所有 HID 设备进行遍历。

2、对设备进行遍历: 遍历过程如下

A、首先利用以下函数:

```
1. WINSETUPAPI BOOL WINAPI SetupDiEnumDeviceInterfaces(  
2.     IN HDEVINFO DeviceInfoSet,  
3.     IN PSP_DEVINFO_DATA DeviceInfoData OPTIONAL,  
4.     IN LPGUID InterfaceClassGuid,  
5.     IN DWORD MemberIndex,  
6.     OUT PSP_DEVICE_INTERFACE_DATA DeviceInterfaceData);
```

运行此函数的主要目的是取得第一个参数 DeviceInfoSet 的填充值, 又将此值作为以下函数

```
1. BOOL SetupDiGetDeviceInterfaceDetail(  
2.     HDEVINFO DeviceInfoSet,  
3.     PSP_DEVICE_INTERFACE_DATA DeviceInterfaceData,  
4.     PSP_DEVICE_INTERFACE_DETAIL_DATA DeviceInterfaceDetailData,  
5.     DWORD DeviceInterfaceDetailDataSize,  
6.     PDWORD RequiredSize,  
7.     PSP_DEVINFO_DATA DeviceInfoData);
```



的第一个参数,以便取得这个函数的第三个参数 `evicInterfaceDetailData` 的填充值,然后利用这个值传递给 `CreateFile` 函数,此时 `CreateFile` 会返回一个指向 HID 设备的句柄,再根据以下函数

```
1.  BOOLEAN  HidD_GetAttributes(  
2.      IN  HANDLE  HidDeviceObject,  
3.      OUT PHIDD_ATTRIBUTES  Attributes  
4.  );
```

取得此 HID 设备的属性(第二个参数的填充值),然后判断属性里的 PID

(`Attributes->ProductID`)和 VID(`Attributes->VendorID`)是否是我们要查找的设备的 PID 和 VID。PID 和 VID 在下位机固件代码的设备描述符里提供(设备描述里的 `idProduct` 域和 `idVendor`,参考百合电子工作室发表的文章《[USB 开发基础——USB 命令\(请求\)和 USB 描述符](#)》一文中表 4),当然您也可以通过一些工具查询得到 PID 和 VID,您可以到 USB 组织官方网站 www.usb.org 下载这类工具。

3、根据得到的设备句柄利用 `ReadFile` 和 `WriteFile` 对设备进行读写操作。

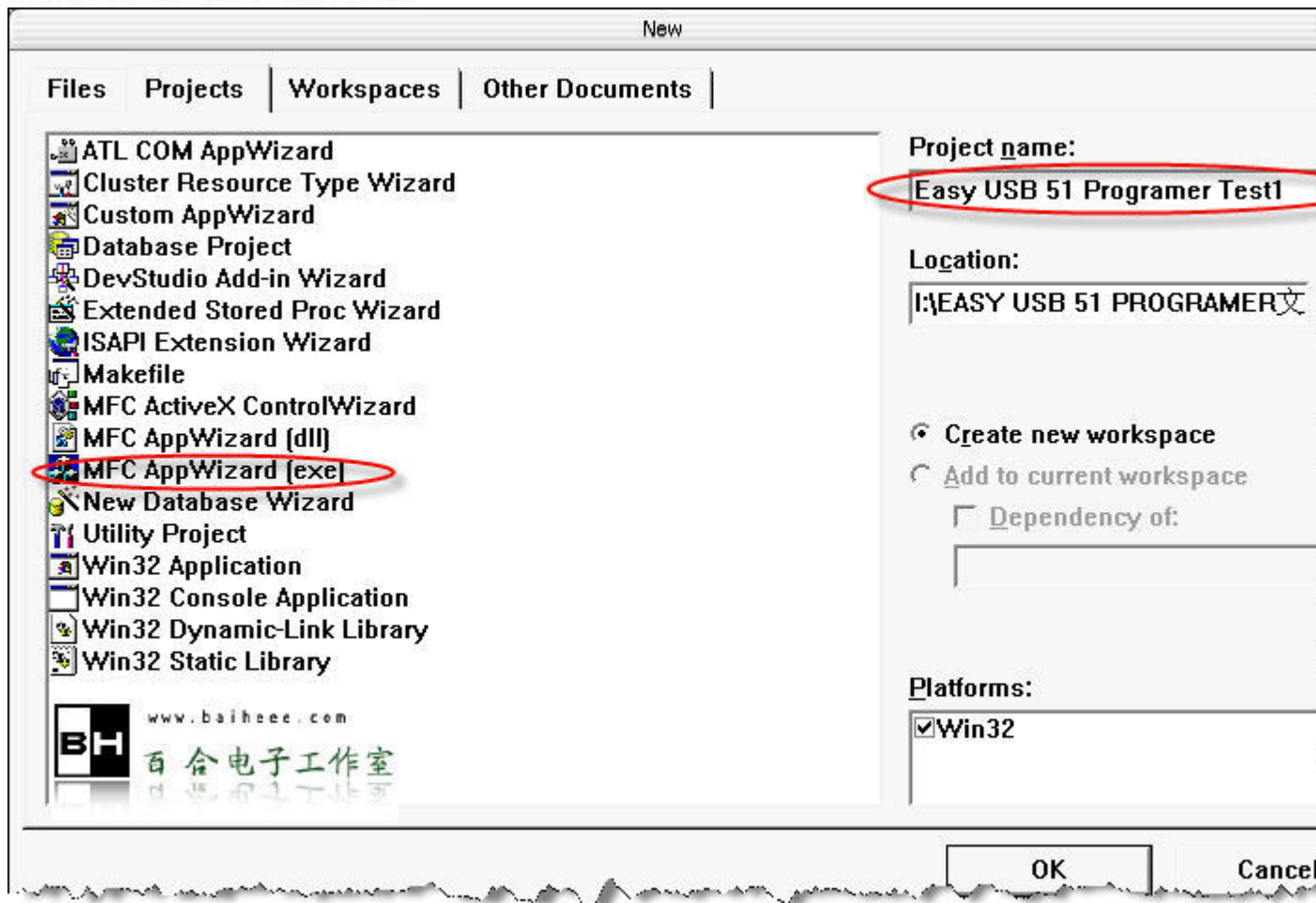
百合电子工作室已经将以上步骤封装成了一个 HID 类(参考了其它一些实例代码),它能实现对指定 PID 和 VID 设备的查找,并实现了数据收发功能,同时具有设备拨插检测通知功能。
[点击这里下载](#)

下面用实例说明如何使用这个类。

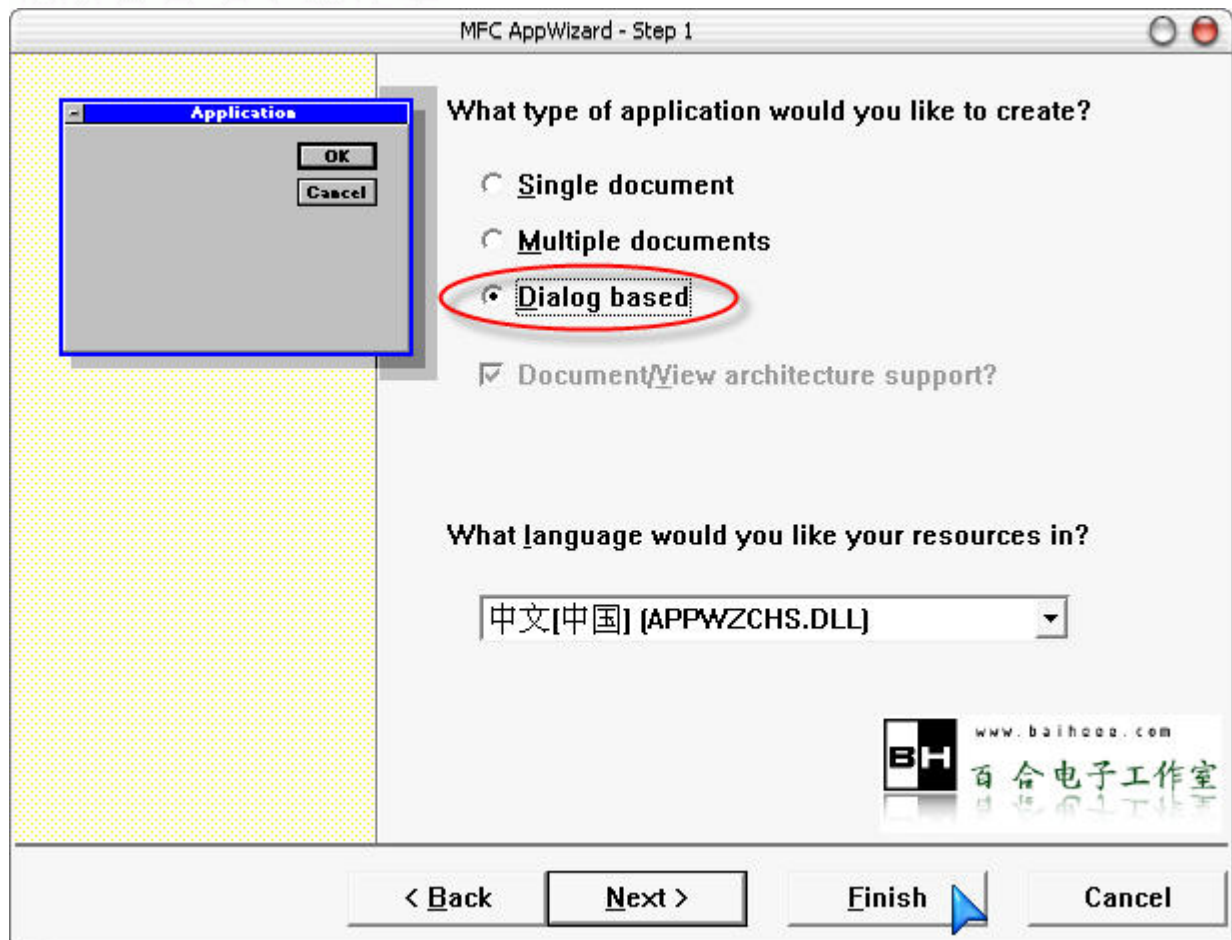
四、读写 HID 设备实例

实例 1

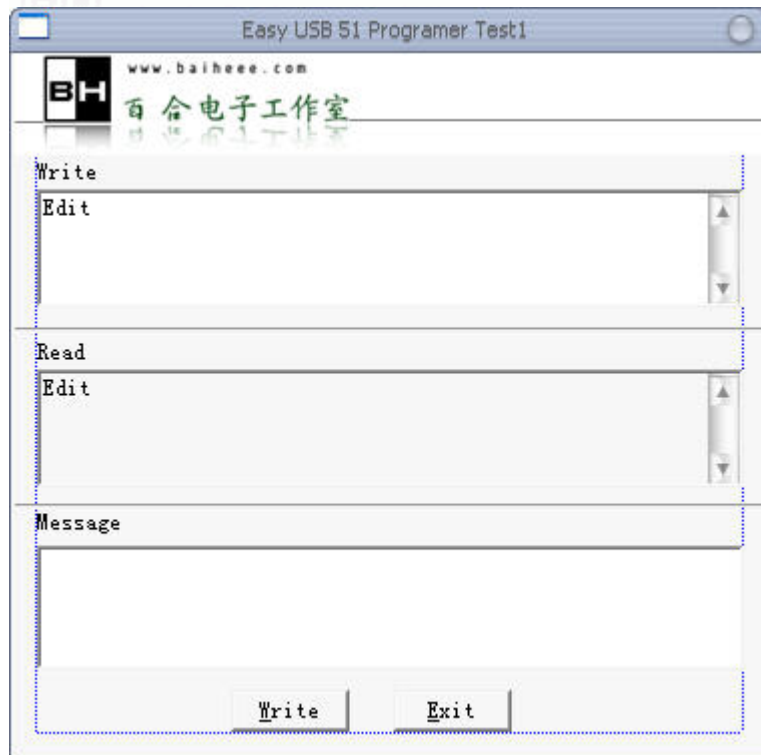
1、找开 Visual C++ 6.0,新建一基于 MFC 的工程名为: Easy USB 51 Programmer Test1。



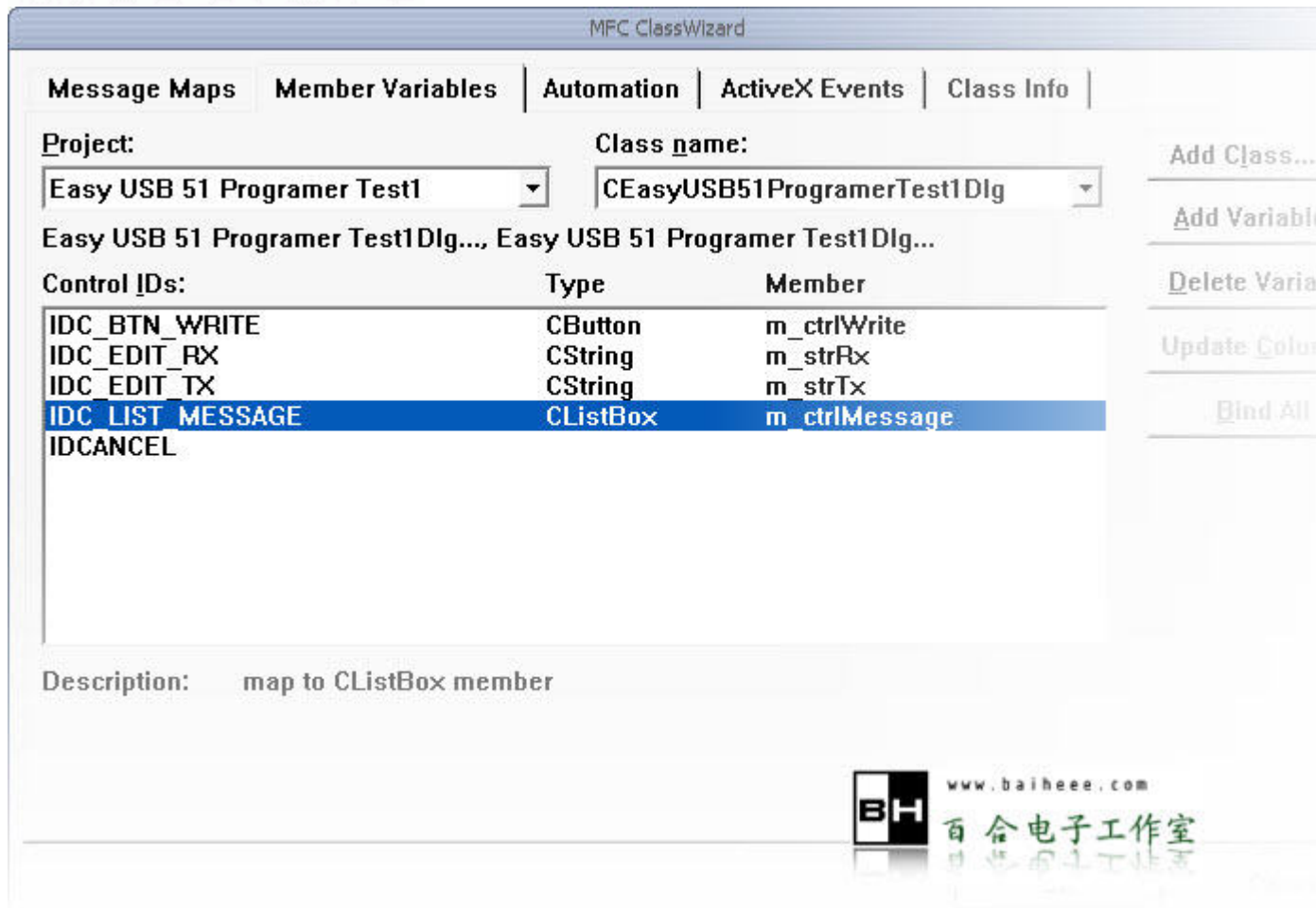
2、MFC AppWizard Step 1 对话框中选择基于对话框的应用程序，然后点“Finish”按钮，如图所示：



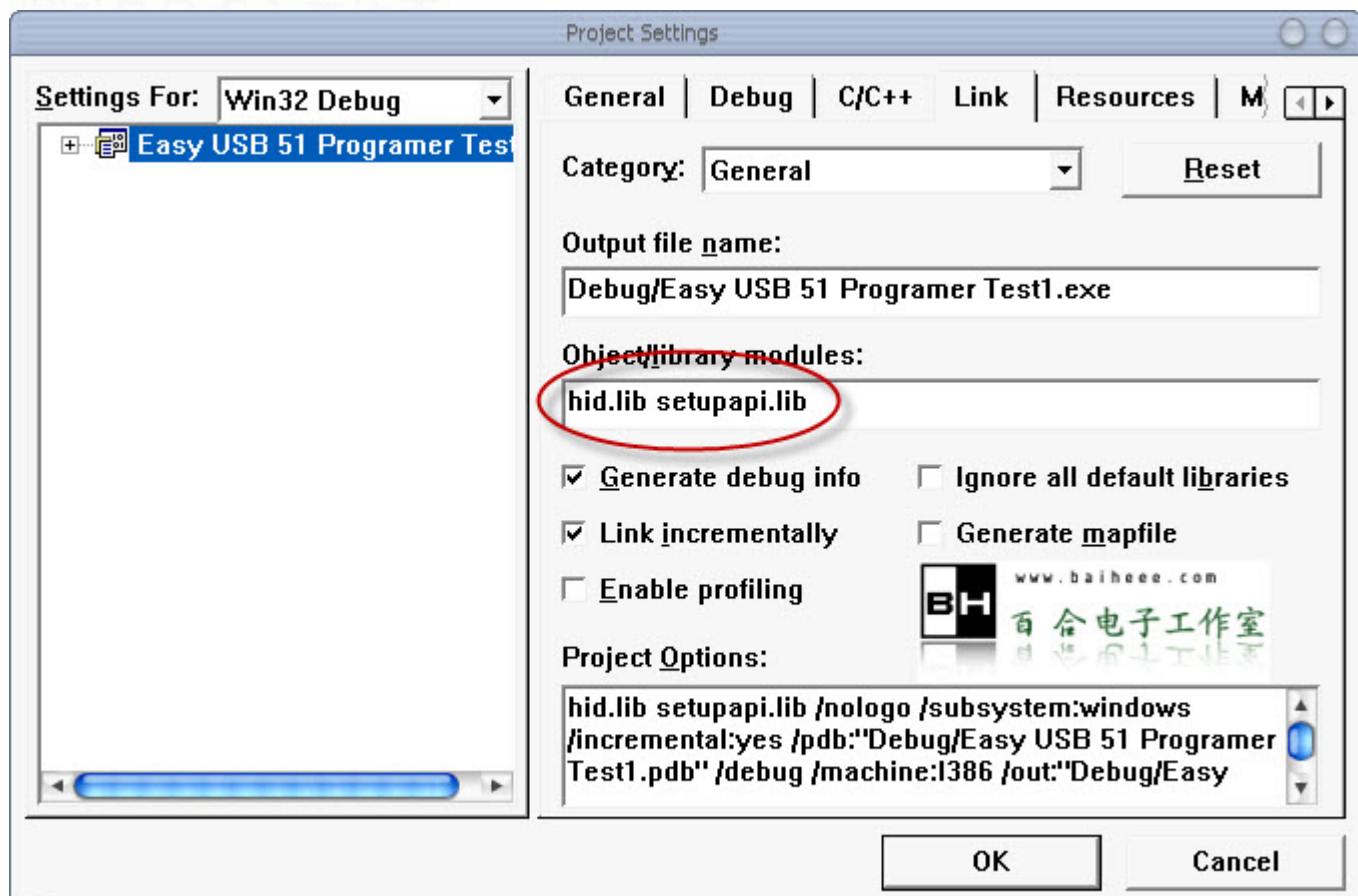
3、创建 3 个静态文本标签（Static Text），文本内容分别为：Write、Read 和 Message；创建两个文本框和一个列表框，ID 分别为：IDC_EDIT_TX、IDC_EDIT_RX 和 IDC_LIST_MESSAGE；两个按钮 ID 和文本分别为：IDC_BTN_WRITE（Write）和 IDCANCEL（Exit）。界面如下：



4、添加控件所对应的变量，如下图所示：



5、将 Hid.c 和 Hid.h 导入工程，并将“要用到的 windows ddk 里的几个文件”文件夹内的文件复制到工程所在目录，在 Proect->Settings->Link 页的“Object/Library moudles”设置中添加“hid.lib setupapi.lib”，如下图所示：



6、在 stdafx.h 文件中包含头文件语句前添加: #define WINVER 0x0500

7、修改 Hid.c 中的 PID 和 VID 宏定义来设置需要访问的 HID 设备, 此处的 PID 和 VID 值分别为 0x0666 和 0x0471:

```
1. #define VID 0x0471
2. #define PID 0x0666
```

8、在 CEasyUSB51ProgramerTest1Dlg 类中添加成员变量 m_MyHidDevice, 其定义如下

```
1. CHid m_MyHidDevice;
```

当然您得包含头文件 Hid.h。

9、在 CEasyUSB51ProgramerTest1Dlg 类的 OnInitDialog 函数中添加如下语句:


```
1. m_MyHidDevice.m_hParentWnd = (HANDLE*) this->GetSafeHwnd( );
2. if(m_MyHidDevice.FindHid()) //找到指定 HID 设备
3. {
4.     m_ctrlMessage.InsertString(-1,"My hid device detected");
5. }
6. else //没有找到指定 HID 设备
7. {
8.     m_ctrlMessage.InsertString(-1,"My hid device not detected");
9.     m_ctrlWrite.EnableWindow(FALSE); //禁用"write"按钮
10. }
```

10、在 CEasyUSB51ProgramerTest1Dlg 的消息映射中
("BEGIN_MESSAGE_MAP(CEasyUSB51ProgramerTest1Dlg, CDialog)" 与
"END_MESSAGE_MAP()"之间) 添加如下代码:

```
1. ON_MESSAGE(WM_DEVICECHANGE, OnDeviceChange)
```

11、在 CEasyUSB51ProgramerTest1Dlg 类中添加成员函数:

```
1. LRESULT OnDeviceChange(WPARAM wParam, LPARAM lParam);
```

12、成员函数 OnDeviceChange 的结构如下:

```
1. LRESULT CEasyUSB51ProgramerTest1Dlg::OnDeviceChange(WPARAM wParam, LPARAM lParam)
2. {
3.
4.     /* take the appropriate action for the message */
5.     switch(LOWORD(wParam))
6.     {
7.
8.         /* HID device arrival */
9.         case DBT_DEVICEARRIVAL:
10.        {
11.            /* try to open a handle to the device */
12.            if(m_MyHidDevice.FindHid()) /*此处必须调用 FindHid()判断是否是“我”的 HID
设备插入 USB 口*/
```

```

13.         {
14.             /*检测到指定的 HID 设备插入 USB 口*/
15.             /*您在这里可以添加其它功能代码*/
16.         }
17.         break;
18.     }
19.
20.     /* HID device removal */
21.     case DBT_DEVICEREMOVECOMPLETE:
22.     {
23.         if(!m_MyHidDevice.FindHid()) /*此处必须调用 FindHid()判断是否是“我”的
HID 设备拔出 USB 口*/
24.         {
25.             /*检测到指定的 HID 设备拔出 USB 口*/
26.             /*您在这里可以添加其它功能代码*/
27.         }
28.         break;
29.     }
30. }
31.
32. return true;
33. }

```

13、这里为了实现在 Message 信息框里显示 HID 设备的拨插操作，现对 OnDeviceChange 函数作如下填充：

```

1.  LRESULT CEasyUSB51ProgramerTest1Dlg::OnDeviceChange(WPARAM wParam, LPARAM lParam)
2.  {
3.
4.      /* take the appropriate action for the message */
5.      switch(LOWORD(wParam))
6.      {
7.
8.          /* HID device arrival */
9.          case DBT_DEVICEARRIVAL:
10.         {
11.             /* try to open a handle to the device */
12.             if(m_MyHidDevice.FindHid()) /*此处必须调用 FindHid()判断是否是“我”的 HID
设备插入 USB 口*/
13.             {
14.                 /*检测到指定的 HID 设备插入 USB 口*/

```

```

15.          /*您在这里可以添加其它功能代码*/
16.          unsigned short nIndex    = m_ctrlMessage.InsertString(-1,"My hid
           device detected");
17.          m_ctrlMessage.SetCurSel(nIndex);    //流动信息窗口
18.          m_ctrlWrite.EnableWindow(TRUE);    //启用"write"按钮
19.      }
20.      break;
21.  }
22.
23.      /* HID device removal */
24.      case DBT_DEVICEREMOVECOMPLETE:
25.      {
26.          if(!m_MyHidDevice.FindHid())    /*此处必须调用 FindHid()判断是否是“我”的
           HID 设备拔出 USB 口*/
27.          {
28.              /*检测到指定的 HID 设备拔出 USB 口*/
29.              /*您在这里可以添加其它功能代码*/
30.              unsigned short nIndex    = m_ctrlMessage.InsertString(-1,"My hid
           device removed");
31.              m_ctrlMessage.SetCurSel(nIndex);    //流动信息窗口
32.              m_ctrlWrite.EnableWindow(FALSE);    //禁用"write"按钮
33.          }
34.          break;
35.      }
36.  }
37.
38.      return true;
39.  }

```

14、对 HID 的读写可通过 Hid 类的成员函数 WriteHid 和 ReadHid。以下是"write"按钮的响应函数，实现对 HID 设备的读写操作：

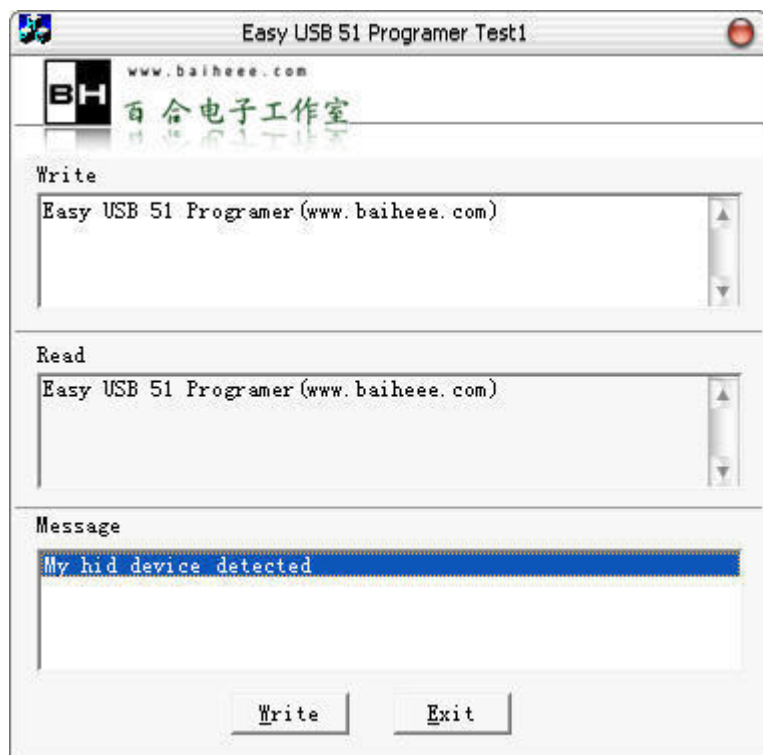
```

1.  void CEasyUSB51ProgramerTest1Dlg::OnBtnWrite()
2.  {
3.      unsigned char ucTxBuffer[64];    //发送缓冲
4.      unsigned char ucRxBuffer[64];    //接收缓冲
5.
6.      UpdateData(TRUE);
7.
8.      //判断发送框中内容是否超过 64 字节
9.      if(m_strTx.GetLength()>64)
10.     {

```

```
11.         AfxMessageBox( "发送字节数不能超过 64 个字节" );
12.     }
13.
14.     //准备发送缓冲区中的内容
15.     for(int i=0; i<64 ; i++)
16.     {
17.         if(i <= (m_strTx.GetLength()-1) )
18.             ucTxBuffer[i]  = m_strTx.GetAt(i);
19.         else
20.             ucTxBuffer[i]  = 0;
21.     }
22.
23.     //写操作
24.     m_MyHidDevice.WriteHid(ucTxBuffer,64);
25.     //读操作
26.     m_MyHidDevice.ReadHid(ucRxBuffer,64);
27.
28.     m_strRx      = ucRxBuffer;
29.     UpdateData(FALSE);
30. }
```

完成后的实际效果:



 下载程序

 下载源代码

实例 2

此例子需要用到扩展板: EXT-BOARD-A。实现功能为通过上位机设定 EXT-BOARD-A 上的 8 个发光二极管状态。

1、命令及数据定义

下位机已经规定了每帧数据的长度为 64 个字节,我们现在需要对每一个字节的含义作出定义。在这个实例中,我们可作如下规定:每帧数据前 5 个字节为命令,命令后面紧跟数据(从第 6 个字节开始)。命令为 ASCII 编码,在此实例中只有一个命令“ENLED”,代表设备 LED 状态。命令后面的一个字节为数据,代表 D0~D7 八个 LED 的状态,后面的字节无意义。

2、修改下位机程序

修改 main.c 文件,其内容如下:

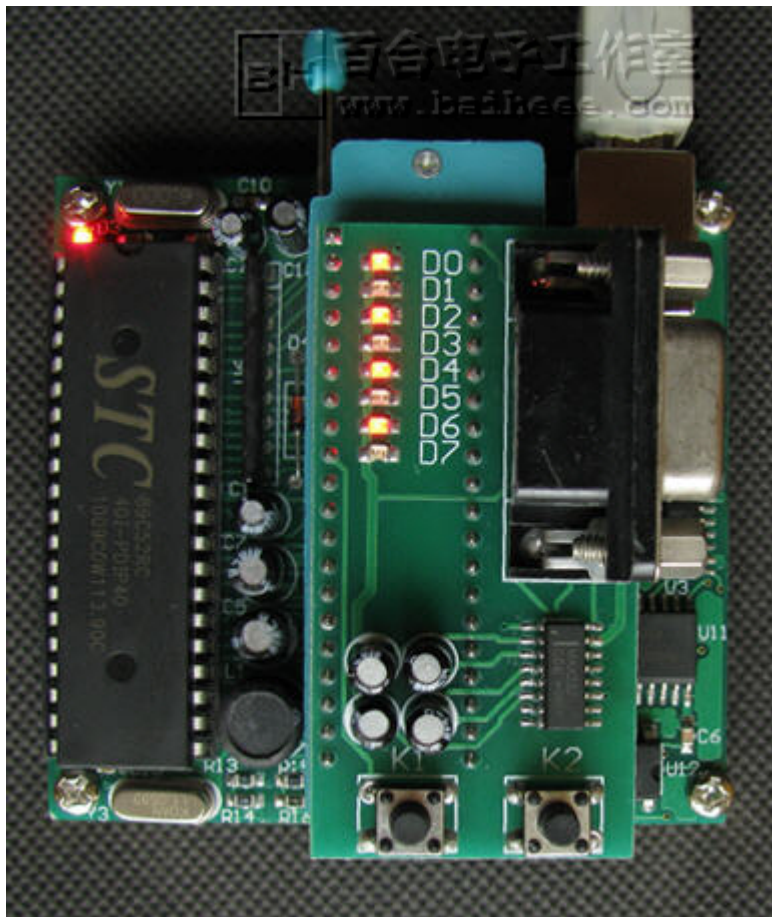
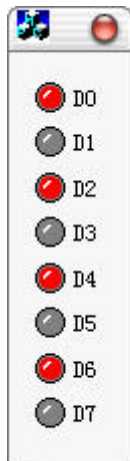
```
1.  /*****Copyright (c)*****/
   *****/
2.  **                               百合电子工作室
3.  **
4.  **
5.  **
6.  **                               http://www.baiheee.com
7.  **
8.  ** 文    件    名: main.c
9.  ** 最后修改日期: 2008 年 12 月 25 日
10. ** 描        述: 用户应用程序
11. ** 版            本: V6.0
12. *****/
   *****/
13.
14. // #include <at89x52.h>
15. #include <reg51.h>
16. #include "D12Config.h"
17. #include "Descriptor.h"
18. #include "Chap_9.h"
19. #include "D12Driver.h"
```

```
20. #include <string.h>
21.
22. main()
23. {
24.     unsigned char ucLedState;
25.
26.     if (Init_D12()!=0)                //初始化 D12
27.         return;                      //如果初始化不成功,返回
28.
29.     IT0 = 0;                          //外部中断 0 为电平触发方式
30.
31.     EX0 = 1;                          //开外部中断 0
32.     PX0 = 0;                          //设置外部中断 0 中断优先级
33.     EA = 1;                          //开 80C51 总中断
34.
35.     while(1)
36.     {
37.         usbserve();                  //处理 USB 事件
38.         if(bEPPflags.bits.configuration)
39.         {
40.             //在这里添加端点操作代码
41.
42.             if(bEPPflags.bits.ep2_rxdone ) //主端点接收到数据(从主机发往设备的数
               据)
43.             {
44.                 bEPPflags.bits.ep2_rxdone = 0;
45.
46.                 //判断是否是 ENLED 命令 (EpBuf 的前 5 个字节为"ENLED")
47.                 if(strncmp( "ENLED", EpBuf, 5)==0)
48.                 {
49.                     //取得 LED 状态设定值
50.                     ucLedState = EpBuf[5];
51.
52.                     //设定 LED 状态
53.                     P0 = ucLedState;
54.                 }
55.             }
56.         }
57.     }
58. }
```

 [点击这里下载修改好的源代码](#)

3、上位机程序

通过实例 1 的学习, 其实上位机程序的编写非常简单, 所以在这里只贴出源代码。



上位机界面

LED 的状态由上位机控制

 下载程序

 下载源代码