

实例讲解 USB 的枚举（配置）过程

所需要工具

- USB Monitor2.26
- 优盘一个

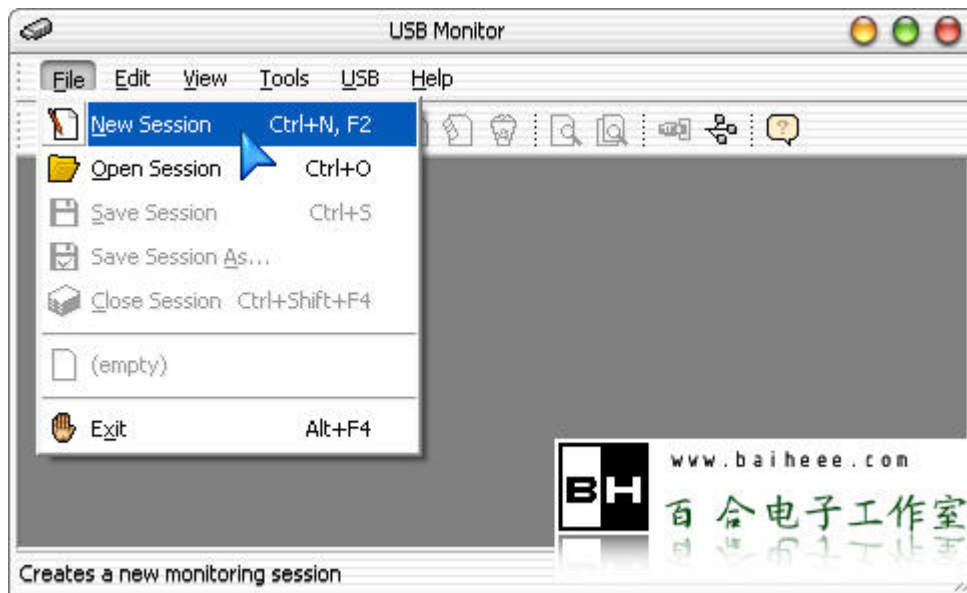
网上大量介绍用 bus hound 作监控软件，我们为什么不选 bus hound 而选 USB Monitor 呢，因为 bus hound 在 WindowsXP 环境不能监控 USB 枚举（配置）过程，它只有在 Windows2000 下才能实现这个功能。不过用 bus hound 做除枚举以外的数据分析还是比较好用的。USB Monitor 的唯一缺点是找不到它的破解版，只能试用大约一个月的时间。

USB Monitor 的安装

下载 USB Monitor2.26，安装一路 next 下去即可，安装完成后将会提示您重新启动系统。

启动您的 USB Monitor

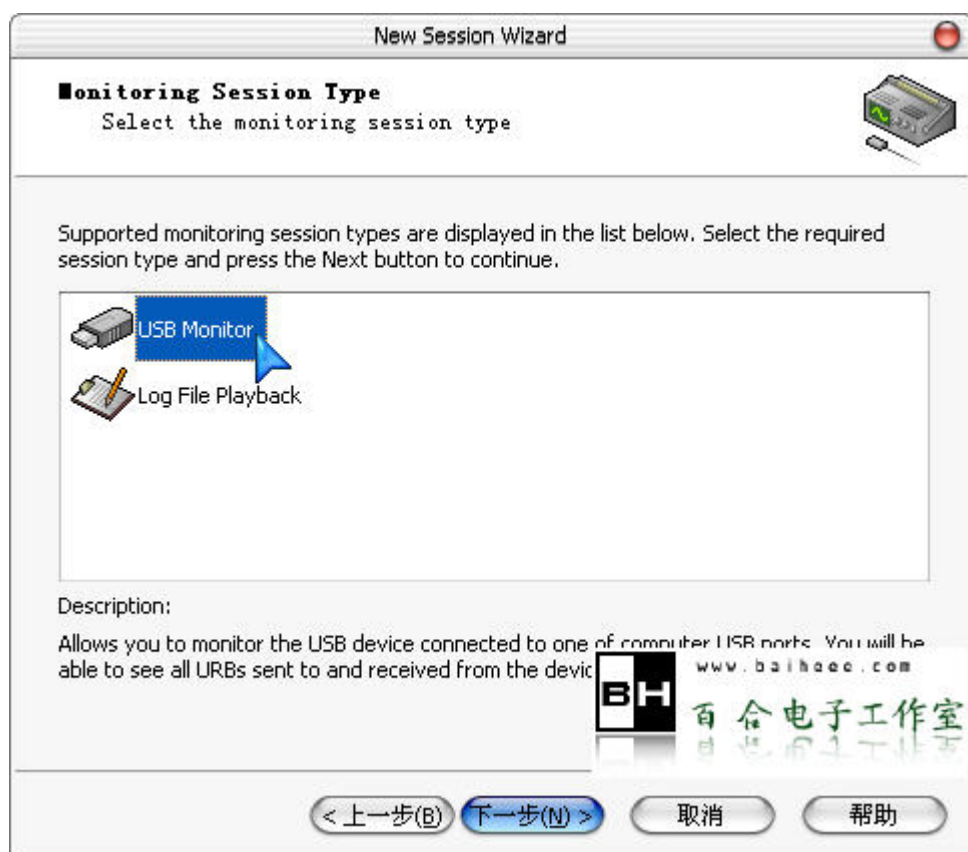
1、运行安装好后的 USB Monitor，点击“File”→“New Session”



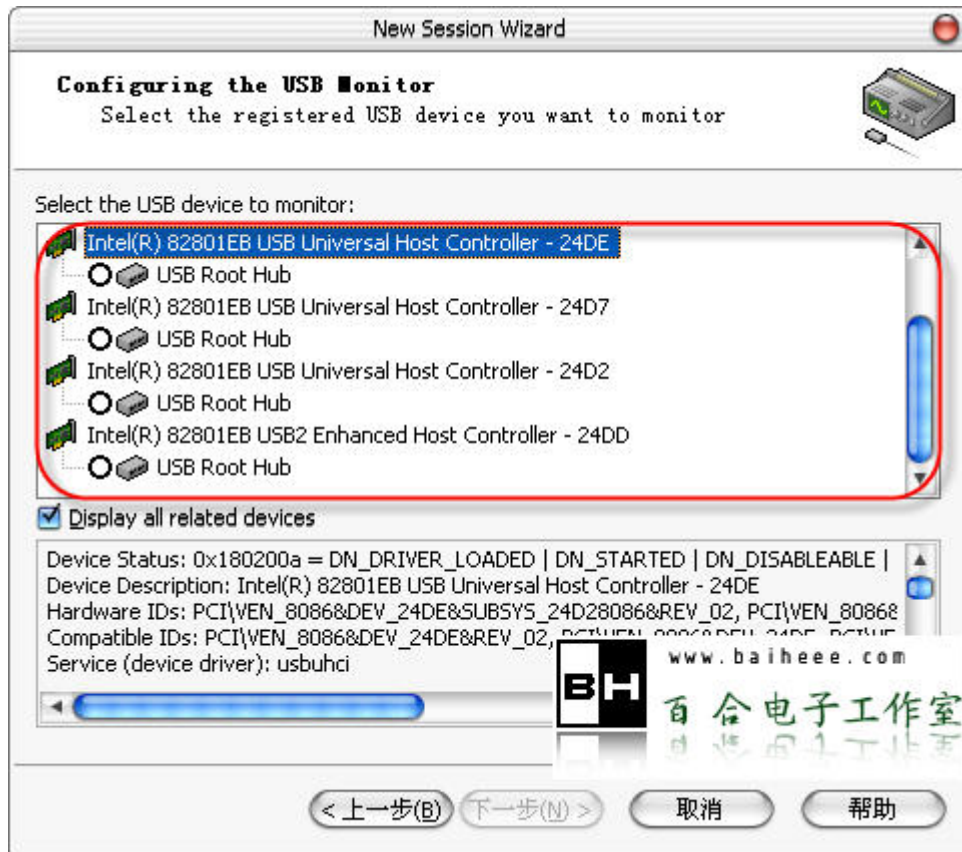
2、在弹出的对话框中点“下一步”



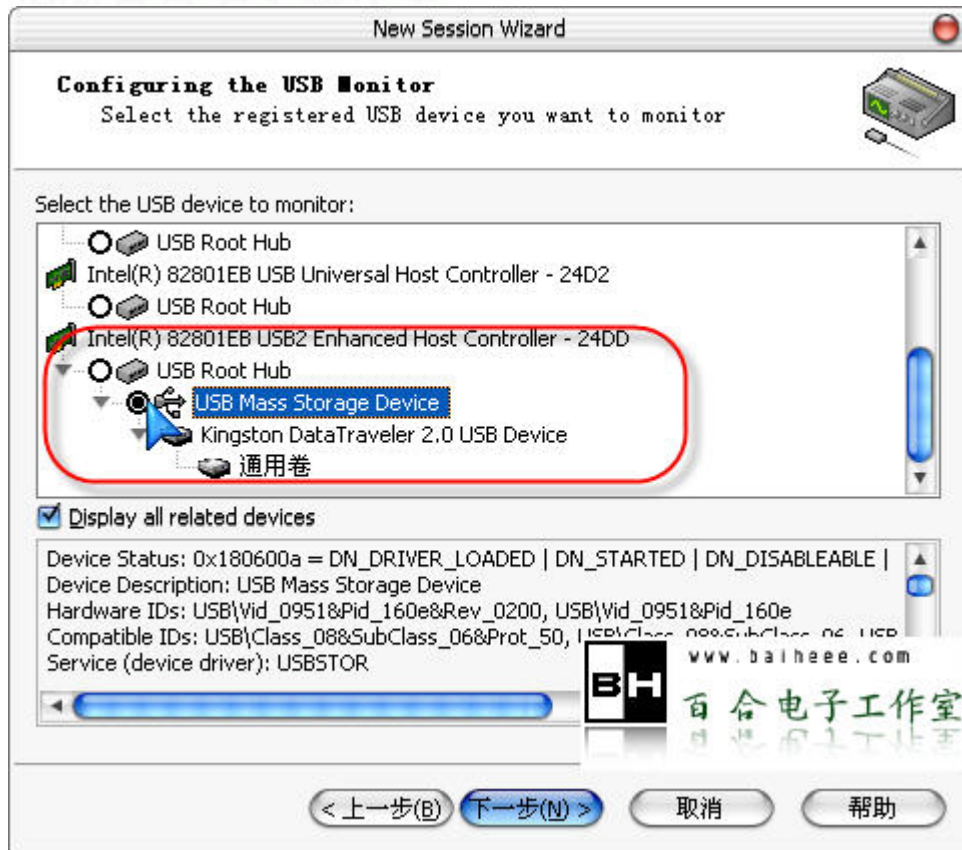
3、在“Monitoring Session Type”对话框中选择“USB Monitor”后点下一步



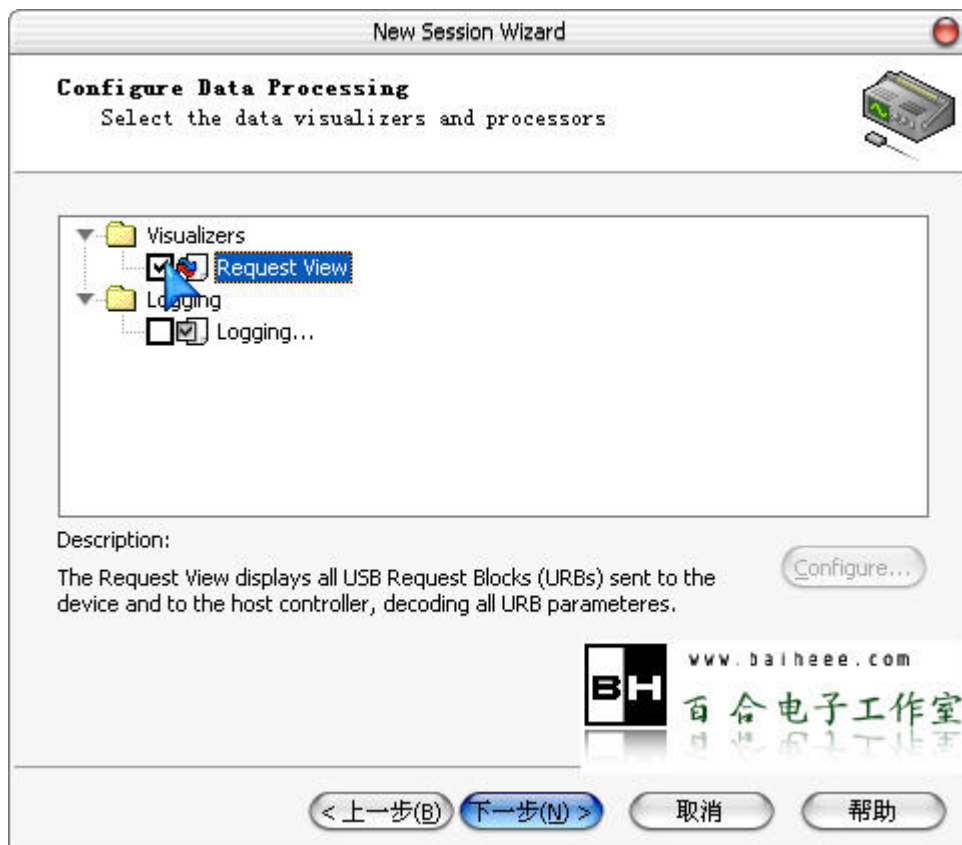
4、在“Configuring the USB Monitor”对话框中提示您选择哪一个 USB 设备需要监视，如果这时您还没有将任何 USB 设备插入主，将显示如下界面，我们怎么知道应该选择哪一项呢？请看第 5 步...



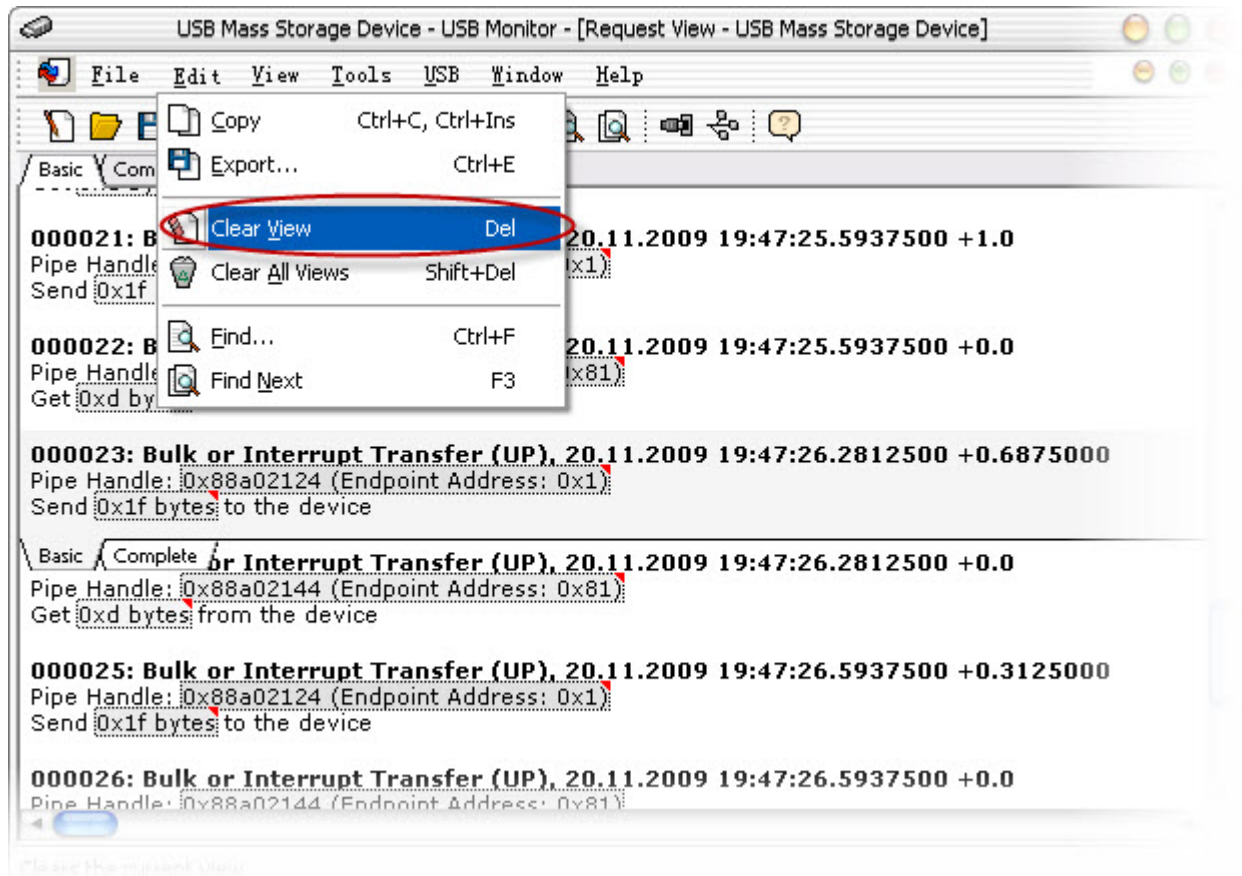
5、将您需要监控的 USB 设备插入主机 USB 端口，这时将会在项后面多出一些内容，如下图所示，我这里选择“USB Mass Storage”。小技巧：当我们要监控我们自己开发的设备时，可以先用一个优盘插入其中一个 USB 端口，在此步中选择我们插入的优盘，在实际监测过程中将我们开发的 USB 设备插入这个 USB 插口即可。



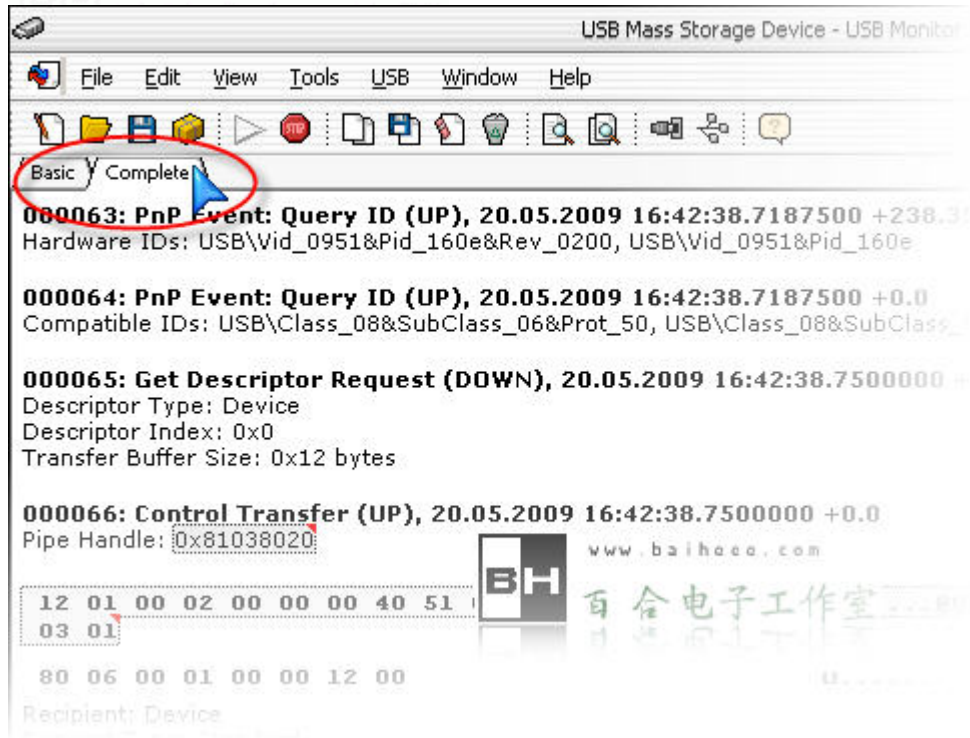
6、在“Configure Data Processing”对话框中选择“Request View”并点下一步



7、弹出您的优盘，然后在 USB Monitor 主界面里点“Edit”->“Clear View”，如下图所示



8、插入您的优盘，记得一定要插在同一个 USB 插口哟！此时 USB Monitor 会检测许多的数据流，我们切换到 Complete 标签，如下图所示：



分析 USB 设备的枚举过程

通过以上的操作后，我们现在可以对 USB Monitor 监测到的数据进行分析了。

- 1、按“Ctrl + F”，查找第一个“Descriptor Request”，从这里开始是对我们有用的信息。

002128: PnP Event: Query ID (UP), 21.05.2009 09:12:58.3593750 +11.625000
Hardware IDs: USB\VID_0951&PID_160E&Rev_0200, USB\VID_0951&PID_160E

002129: PnP Event: Query ID (UP), 21.05.2009 09:12:58.3593750 +0.0
Compatible IDs: USB\Class_08&SubClass_06&Prot_50, USB\Class_08&SubClass_06

002130: **Get Descriptor Request (DOWN)**, 21.05.2009 09:12:58.3750000 +0.0
Descriptor Type: Device
Descriptor Index: 0x0
Transfer Buffer Size: 0x12 bytes

002131: Control Transfer (UP), 21.05.2009 09:12:58.3750000 +0.0
Pipe Handle: 0x81020020

12 01 00 02 00 00 00 40 51 09 0E 16 00 02 01 02
03 01

Setup Packet

80 06 00 01 00 00 12 00

Recipient: Device
Request Type: Standard
Direction: Device->Host
Request: 0x6 (GET_CONFIGURAT...)
Value: 0x100
Index: 0x0
Length: 0x12

2、对 USB Monitor 显示的数据结构进行分析:

① 002130: Get Descriptor Request (DOWN) 21.05.2009 09:12:58.3750000 +0.0156250
Descriptor Type: Device
Descriptor Index: 0x0
Transfer Buffer Size: 0x12 bytes

① 002131: Control Transfer (UP) 21.05.2009 09:12:58.3750000 +0.0
Pipe Handle: 0x81020020

12 01 00 02 00 00 00 40 51 09 0E 16 00 02 01 02
03 01

Setup Packet
80 06 00 01 00 00 12 00

Recipient: Device
Request Type: Standard
Direction: Device->Host
Request: 0x6 (GET_CONFIGURATION)
Value: 0x100
Index: 0x0
Length: 0x12

① 002132: Get Descriptor Request (DOWN) 21.05.2009 09:12:58.3750000 +0.0
Descriptor Type: Configuration
Descriptor Index: 0x0
Transfer Buffer Size: 0x9 bytes

① 002133: Control Transfer (UP) 21.05.2009 09:12:58.3750000 +0.0
Pipe Handle: 0x81020020

09 02 20 00 01 01 00 80 32

Setup Packet
80 06 00 02 00 00 09 00

Recipient: Device
Request Type: Standard
Direction: Device->Host
Request: 0x6 (GET_CONFIGURATION)
Value: 0x200
Index: 0x0
Length: 0x9

- ①序列号，可理解为一个上行或下行数据流标号，无实际意义
- ②命令类型，括号里的“Down”代表输出数据，即主机到设备
- ③命令数据流，出现“Setup Packet”后面，十六进制
- ④对命令数据的简单分析
- ⑤传输类型，括号里的 UP 代表输入数据，即设备到主机
- ⑥设备返回的数据流，十六进制
- ⑦对命令（请求）数据的详略分析（其中对“Request”的分析应该有 BUG，0x6 应为 Get_Descriptor，而 USB Monitor 解释为 Get_Configuration）

以上没有看到对上行数据（设备返回的数据）的分析，我们只要将鼠标停留在设备返回数据流处（上图所示标记⑥），就会弹出对上行数据的分析画面，如下图所示：

002129: PnP Event: Query ID (UP), 21.05.2009 09:12:58.3593750 +0.0
Compatible IDs:

002130: Get De
Descriptor Type: Class: 0x0, Subclass: 0x0, Protocol: 0x0
Descriptor Index: Max. packet size: 64 bytes
Transfer Buffer S: Vendor ID: 0x951 (Unknown), Product ID: 0x160e, Device Version: 0x200
Manufacturer: 1, Product: 2, Serial Number: 3
Number of configurations: 1

002131: Control
Pipe Handle: 0x8

12 01 00 02 00 00 00 40 51 09 0E 16 00 02 01 02 03 01

Setup Packet
80 06 00 01 00 00 12 00

Recipient: Device
Request Type: Standard
Direction: Device->Host
Request: 0x6 (GET_CONFIGURATION)
Value: 0x100
Index: 0x0
Length: 0x12

Basic Complete **Descriptor Request (DOWN), 21.05.2009 09:12:58.3593750 +0.0**
Descriptor Type: Configuration
Descriptor Index: 0x0
Transfer Buffer Size: 0x9 bytes

002133: Control Transfer (UP), 21.05.2009
Pipe Handle: 0x81020020

09 02 20 00 01 01 00 80 32

3、枚举过程分析

1)、主机第一次发出请求描述符命令，数据流为：80 06 00 02 00 00 09 00，bRequest 的值为 0x06，代表 Get_Descriptor 命令，但为什么描述符呢？wValue 的高字节表示了描述符的类型，此处 wValue 的值为 0x0100，所以高字节为 0x01，代表设备描述符（见《USB 开发基础——USB 命令（请求）和 USB 描述符》中表 5）；设备返回的数据为：12 01 00 02 00 00 00 40 51 09 0E 16 00 02 01 02 03 01，bLength 的值为 0x12，表示此描述符的长度。bDescriptorType 的值为 0x01，代表设备描述符。bcdUSB 的值为 0x0200，代表 USB 协议的版本号，此处 2.0 版，如果为 0x0110 则表示 1.1 版。bDeviceClass 和 bDeviceSubClass 都为 0，表示设备类别由接口描述符指定。bDevicePortocol 的值为 0，但并不代表它不支持 USB 定义的标准设备类协议，因为此时可由接口描述符指明设备支持的协议。bMaxPacketSize0 的值为 0x40，表示端点 0 的数据包最大长度为 64 字节。iManufacturer、iProduct 和 iSerialNumber 的分别为 0x01、0x02 和 0x03，表示字符串索引，在主机读取字符串的命令中将这几个值来填充 wIndex 字段。

002129: Pin Event: Query ID (UP), 21.05.2009 09:12:58.3750000 +0.0
Compatible IDs: USB\Class_08&SubClass_06

002130: Get Descriptor Request (DOWN), 21.05.2009 09:12:58.3750000 +0.0
Descriptor Type: Device
Descriptor Index: 0x0
Transfer Buffer Size: 0x12 bytes

002131: Control Transfer (UP), 21.05.2009 09:12:58.3750000 +0.0
Pipe Handle: 0x81020020

Device Descriptor
Length: 0x12 bytes, USB Specification: 0x200
Class: 0x0, Subclass: 0x0, Protocol: 0x0
Max. packet size: 64 bytes
Vendor ID: 0x951 (Unknown), Product ID: 0x160e, Device Version: 0x0000
Manufacturer: 1, Product: 2, Serial Number: 3
Number of configurations: 1

12 01 00 02 00 00 00 40 51 09 01 16 00 02 01 02
03 01

Setup Packet
80 06 00 01 00 00 12 00

Recipient: Device
Request Type: Standard
Direction: Device->Host
Request: 0x6 (GET_CONFIGURATION)
Value: 0x100
Index: 0x0
Length: 0x12

命令数据流

设备返回的数据

Basic Complete /escriptor Request (DOWN), 21.05.2009 09:12:58.3750000 +0.0

请求设备描述符的数据流

2)、主机再次发出请求描述符指令, 数据流为: 80 06 00 02 00 00 09 00, wValue 的高字节为 0x02 表示配置描述符, 表示请求配置描述符, wLength 的值为 0x0009, 表示要求返回的数据长度为 9 个字节。这次设备返回的数据只有 9 个字节: 09 02 20 00 01 01 00 80 32, 第一位数为 bLeng 域, 其值为 0x09, 代表此描述符的长度。bDescriptorType 的值为 0x02, 表示配置描述符。wTotalLength 的值为 0x0020, 表示包括此配置描述符、接口描述符、端点描述符和设备类及厂商定义的描述符的总长为 32 个字节。bNumInterfaces 的值为 0x01, 表示该配置支持 1 个接口。MaxPower 的值为 0x32, 表示总线耗电量为 $50 \times 2 = 100\text{mA}$ 。

002132: Get Descriptor Request (DOWN), 21.05.2009 09:12:58.3750000 +0.0
Descriptor Type: Configuration
Descriptor Index: 0x0
Transfer Buffer Size: 0x9 bytes

002133: Control Transfer (UP), 21.05.2009 09:12:58.3750000 +0.0
Pipe Handle: 0x81020020

Configuration Descriptor
Length: 0x9 bytes, Total length: 0x20 bytes
Number of interfaces: 1, Configuration desc.: 0
Attributes: 0x80: [Bus-Powered]
Max Power: 0x32

09 02 20 00 01 01 00 80 32

Setup Packet
80 06 00 02 00 00 09 00

Recipient: Device
Request Type: Standard
Direction: Device->Host
Request: 0x6 (GET_CONFIGURATION)
Value: 0x200
Index: 0x0
Length: 0x9

命令数据流

设备返回的数据流

Basic Complete /escriptor Request (DOWN), 21.05.2009 09:12:58.3750000 +0.0

第一次请求配置描述符的数据流

3)、主机第一次请求设备的配置描述符后得知包括配置描述符、接口描述符、端点描述符等在内的描述总长为 32 个字节后, 这此主机再次发出请求配置描述指令, 这次命令与上一个命令只有 wLength 字段的值不一样, 这次 wLength 的值为 0x20, 表示要求设备返回 32 个字节, 正好和上次命令中设备返回的 wTotalLength 信息一致。设备返回的数据流为如下图所示, USB Monitor 已经将配置描述符、接口描述符、和端点描述符用不同的背景色区分开来, 鼠标停留在不同的区域将弹出对此描述符的具体分析。第一个描述符的数据为: 09 02 20 00 01 01 00 80 32, 和上一次请求配置描述符得到的数据一模一样, 这里不再分析。第二个描述符的数据为: 09 04 00 00 02 08 06 50 00, bLength 为 0x09 表示此描述符为 9 个字节, bDescriptorType 的值为 0x04 表示接口描述符, bInterfaceNumber 为 0x00 表示此配置接口的索引值。bNumEndpoint 的值为 0x02 表示此接口端点数为两个。bInterfaceClass 的值为 0x08 表示大数据存储类(见《USB 开发基础——USB 命令(请求)和 USB 描述符》中表 11), bInterfaceSubClass 的值为 0x06, 其含义因 bInterfaceClass 的不同而不同, 由于此处 bInterfaceClass 的值指定了此设备属大数据存储类, 所以需要查看大数据存储类(Mass storage class device)相关协议标准, 经查询得知 bInterfaceSubClass 为 0x06 时代表传输协议工业标准为 SCSI。bInterfaceProtocol 的值为 0x50, 大数据存储类协议规定其含义为通讯方式为批量传输(Bulk-only)。

第二次请求配置描述符的数据流

4)、在请求完配置描述符后, 主机发出请求字符串描述符指令。请求字符串描述符分两大步, 每大步又分两小步, 两大步为: 先请求字符串描述符语言 ID(此时 wIndex 的值设为零), 再请求 UNICODE 编码的字符串描述符。不管是请求语言 ID 还是 UNICODE 编码的字符串描述符, 都要分成两小步: 先请求设备返回相应字符串描述符的前两个字节, 第一个字节代表了此字符串描述符的长度, 主机根据这个长度再次请求相应字符串描述符, 这次得到整个字符串描述符。

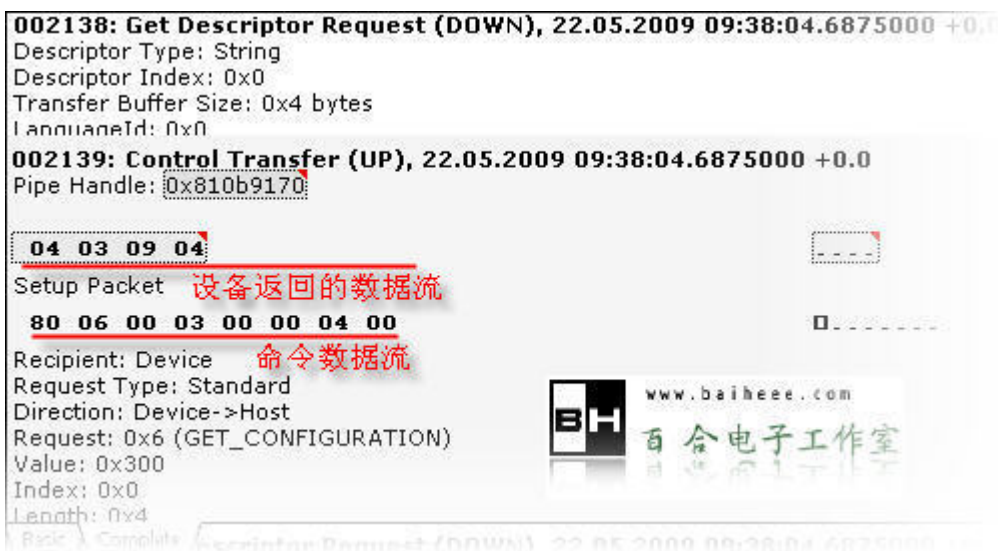
首先取得语言 ID 的前两个字节, 数据流为: 80 06 00 03 00 00 02 00, wValue 的高字节为 0x03, 表示字符串描述符。wIndex 的值为 0x0000, 表示读取语言 ID。wLength 为 0x0002 要求设备返回 2 个字节数据。设备返回的数据为 04 03, 第一个字节为 0x04 表示语言 ID 字符串描述符长度为 4 个字节。



002136: Get Descriptor Request (DOWN), 21.05.2009 09:12:58.3750000 +0.0
Descriptor Type: String
Descriptor Index: 0x0
Transfer Buffer Size: 0x2 bytes
LanguageId: 0x0
002137: Control Transfer (UP), 21.05.2009 09:12:58.3750000 +0.0
Pipe Handle: 0x81020020

04 03 设备返回的数据流
Setup Packet
80 06 00 03 00 00 02 00
Recipient: Device 命令数据流
Request Type: Standard
Direction: Device->Host
Request: 0x6 (GET_CONFIGURATION)
Value: 0x300
Index: 0x0
Length: 0x2

5)、主机根据上次读取的语言 ID 字符串描述符长再次读取此描述符, 这次读出全部语言 ID 描述符。主机发出的数据流为: 80 06 00 03 00 00 04 00, 其中 wValue 的高字节为 0x03, 表示字符串描述符, wIndex 的值为 0x00, 表示读取语言 ID。wLength 为 0x0004 要求设备返回 4 个字节数据。设备返回的数据流为: 04 03 09 04, bLength 的值为 0x04 表示此描述符长度为 4 个字节, bDescriptorType 的值为 0x03 表示字符串描述符。wLANGUID 的值为 0x0409, 表示英语。



002138: Get Descriptor Request (DOWN), 22.05.2009 09:38:04.6875000 +0.0
Descriptor Type: String
Descriptor Index: 0x0
Transfer Buffer Size: 0x4 bytes
LanguageId: 0x0
002139: Control Transfer (UP), 22.05.2009 09:38:04.6875000 +0.0
Pipe Handle: 0x810b9170

04 03 09 04 设备返回的数据流
Setup Packet
80 06 00 03 00 00 04 00
Recipient: Device 命令数据流
Request Type: Standard
Direction: Device->Host
Request: 0x6 (GET_CONFIGURATION)
Value: 0x300
Index: 0x0
Length: 0x4

6)、读取完字符串描述符支持的语言 ID 后将读取 UNICODE 编码的字符串描述符, 同读取语言 ID 一样, 主机同样会分两步读取。主机发出的命令数据流为: 80 06 03 03 09 04 02 00, wValue 的高字节为 0x03, 表示字符串描述符, 低字节为 0x03, 表示读取索引为 3 的字符串描述符(在第 1 步读取的设备描述中 iSerialNumber 字符串的索引为 3, 所以这里实际读取的是设备序列号)。wIndex 的值为 0x0409, 表示语言 ID。wLength 的值为 0x0002, 表示要求设备返回此描述符的前两个字节。设备返回的数据流为: 32 03, 第一个字节的值为 0x32 表示此字符串描述符的长度为 50 个字节。



7)、在取得 UNICODE 字符串描述符的长度后, 主机根据这个长度读取整个 UNICODE 字符串描述符。主机发出的命令数据流为: 80 06 03 03 09 04 32 00, wValue 的高字节为 0x03, 表示字符串描述符, 低字节为 0x03 表示读取索引为 3 的字符串描述符。wIndex 的值为 0x0409 表示语言 ID。wLength 的值为 0x0032, 表示要求设备返回 50 个字节的数据。设备返回的数据从第三个字节开始为 Unicode 编码的字符串, 这里返回的字符串是:

20070620000000005918B19E

002142: Get Descriptor Request (DOWN), 22.05.2009 09:38:04.6875000 +0.0

Descriptor Type: String
Descriptor Index: 0x3
Transfer Buffer Size: 0x32 bytes
LanguageId: 0x409

002143: Control Transfer (UP), 22.05.2009 09:38:04.6875000 +0.0

Pipe Handle: 0x810b9170

32 03 32 00 30 00 30 00 37 00 30 00 36 00 32 00	2.2.0.0.7.0.6.2
30 00 30 00 30 00 30 00 30 00 30 00 30 00 30 00	0.0.0.0.0.0.0.0
30 00 35 00 39 00 31 00 38 00 42 00 31 00 39 00	0.5.9.1.8.B.1.9
45 00	K

Setup Packet

80 06 03 03 09 04 32 00

设备返回的数据流

0. 2.

Recipient: Device 命令数据流
Request Type: Standard
Direction: Device->Host
Request: 0x6 (GET_CONFIGURATION)
Value: 0x303
Index: 0x409
Length: 0x32



Basic Complete Configuration (DOWN), 22.05.2009 09:38:04.6875000 +0.0

8)、在读取完字符串描述符后, 主机发出 Set_Configuration 选择配置索引以激活这个设备的一个配置, 然后设备对这一命令作出回应。(不知为何 USB Monitor 对这此命令的数据流没有显示出来)

002144: Select Configuration (DOWN), 21.05.2009 09:12:58.3750000 +0.0

Configuration Index: 1

002145: Select Configuration (UP), 21.05.2009 09:12:58.3750000 +0.0

Configuration Index: 1
Configuration Handle: 0x81264c68



9)、主机发出 Set_Interface 指令激活设备的某个接口, 然后设备对此指令作出回应。

002146: Select Interface (DOWN), 21.05.2009 09:12:58.4375000 +0.0

Configuration Handle: 0x81264c68, Interface Number: 0x0, Alternate Setting: 0x0

002147: Select Interface (UP), 21.05.2009 09:12:58.5000000 +0.0625000

Configuration Handle: 0x81264c68, Interface Number: 0x0, Alternate Setting: 0x0

10)、以上为所有 USB 设备枚举过程中都会经历的过程, 后续的配置过程将根据不同的设备分类 (见《USB 开发基础——USB 命令 (请求) 和 USB 描述符》中表 6) 而有所不同。下图所示为此实验所用优盘的后续配置过程的一部分。对于具体设备类相关协议规定的枚举过程就不作分析了, 请大家参考相关设备类协议。

002148: Class-Specific Request (DOWN), 22.05.2009 09:38:04.8125000 +0.0

Destination: Interface, Index 0

Reserved Bits: 0

Request: 0xfe

Value: 0x0

Get 0x1 bytes from the device

00

002149: Control Transfer (UP), 22.05.2009 09:38:04.8125000 +0.0

Pipe Handle: 0x810b9170

00

Setup Packet

A1 FE 00 00 00 00 01 00

Recipient: Interface

Request Type: Class

Direction: Device->Host

Request: 0xfe (Unknown)

Value: 0x0

Index: 0x0

Length: 0x1

002150: Bulk or Interrupt Transfer (UP), 22.05.2009 09:38:04.8125000 +0.0

Pipe Handle: 0x8124a744 (Endpoint Address: 0x2)

```
Send 0x1f bytes to the device:
```

```
55 53 42 43 D8 84 F4 80 24 00 00 00 80 00 06 12 USBSC接口设备...
00 00 00 24 00 00 00 00 00 00 00 00 00 00 00 00 www.haiheac.com
```

002151: Bulk or Interrupt Transfer (UP), 22

Pipe Handle: 0x8124a724 (Endpoint Address: 0x8

```
Get 0x24 bytes from the device;
```

```
00 80 02 02 1F 00 00 00 4B 69 6E 67 73 74 6F 6E .D.....King
44 61 74 61 54 72 61 76 65 6C 65 72 20 32 2E 30 DataTraveler
31 2E 30 30 .1.00
```