实用通信知识

|  |  |
| --- | --- |
| 作者 | 刘文强 |
| 时间 | 2014/3/18 |
|  |  |

目录

[实用通信知识 1](#_Toc382927002)

[1 物理层 3](#_Toc382927003)

[1.1 RS485 3](#_Toc382927004)

[1.2 SPI 4](#_Toc382927005)

[1.3 I2C 4](#_Toc382927006)

[2 软件协议层 5](#_Toc382927007)

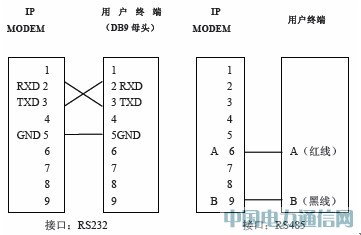
[2.1 MODBUS 5](#_Toc382927008)

[2.2 PROFIBUS 5](#_Toc382927009)

# 物理层

## RS485

适合多点布网



 什么是RS-485接口？它比RS-232-C接口相比有何特点？  
    答：由于RS-232-C接口标准出现较早，难免有不足之处，主要有以下四点：  
（1） 接口的信号电平值较高，易损坏接口电路的芯片，又因为与TTL 电平不兼容故需使用电平转换电路方能与TTL电路连接。  
（2） 传输速率较低，在异步传输时，波特率为20Kbps。  
（3） 接口使用一根信号线和一根信号返回线而构成共地的传输形式，这种共地传输容易产生共模干扰，所以抗噪声干扰性弱。  
（4） 传输距离有限，最大传输距离标准值为50英尺，实际上也只能用在50米左右。  
  1. RS-485的电气特性：逻辑"1"以两线间的电压差为+（2-6） V表示；逻辑"0"以两线间的电压差为-（2-6）V表示。接口信号电平比RS-232-C降低了，就不易损坏接口电路的芯片， 且该电平与TTL电平兼容，可方便与TTL电路连接。   
  2. RS-485的数据最高传输速率为10Mbps  
  3. RS-485接口是采用平衡驱动器和差分接收器的组合，抗共模干能力增强，即抗噪声干扰性好。   
  4. RS-485接口的最大传输距离标准值为4000英尺，实际上可达3000米，另外RS-232-C接口在总线上只允许连接1个收发器， 即单站能力。而RS-485接口在总线上是允许连接多达128个收发器。即具有多站能力,这样用户可以利用单一的RS-485接口方便地建立起设备网络。因RS-485接口具有良好的抗噪声干扰性，长的传输距离和多站能力等上述优点就使其成为首选的串行接口。 因为RS485接口组成的半双工网络，一般只需二根连线，所以RS485接口均采用屏蔽双绞线传输。RS485接口连接器采用DB-9的9芯插头座，与智能终端RS485接口采用DB-9（孔），与键盘连接的键盘接口RS485采用DB-9（针）

   RS485已经是网络技术了，一条总线上可以挂接200多个节点，再加上扩展技术；而RS232只能在2点之间通信。软件上管理相差甚远。

         在485上跑，必是主从方式；一定有一个主节点进行指挥的，它负责向各个从节点分发令牌。只有接收到令牌的从节点才能占有总线。——这么复杂的通信协议，软件上比RS232高级多了。

基于RS485，控制网络发展了集中式主控网，分布式控制网络，……软件上发展了分布式操作系统。与RS232上跑的通信软件天壤之别了。

## SPI

一般适用FLASH

## I2C

一般适用EEPROM

## USB

### USB通用介绍

己对USB的几点认识（主要写枚举过程相关知识）：

1．端点

从硬件的角度来看，端点其实就是USB中一系列实际的物理数据缓冲区，发送和接收的数据都存在这里。一个设备可以有很多种传输方式来与主机进行数据通信，每一种传输中都可以有特定的端点。从设备的角度来说，端点一般都直接由USB接口芯片来提供，功能较强的芯片都会提供多个具有一定容量的端点，开发人员在设计USB程序时一任务就是要合理分配这些端点，而每一次USB的数据传输都是在某一个特定的端点和主机之间进行的。因此，端点号也是第一次USB数据传输非常重要的参数。

USB协议规定，低速设备只能定义两个端点，即端点0和端点1，此外除端点0以外，任何一个端点都可以定义为IN端点或是OUT端点，因此一个全速设备则能定义多个个端点（最多32个）。

       USB系统中，每一个端点都有惟一的地址，这是由设备地址和端点号共同决定的。而设备的大小、属性等在设备出场时由厂家定义。所以，每一个USB设备在主机看来就是一系列端点的集合，主机通过端点与设备进行通信。

       端点的特性，主要有数据传输方式（用于IN事务的端点、OUT事务的端点和SETUP事务的端点）、总线访问频率、带宽、端口号（由USB接口芯片定义）和数据包最大容量等（也由芯片硬件决定）。

   除了端点0（用作控制传输端占）外，端点必须在设备被主机配置后才能使用。

2． 管道

管道并不像端点一样有实在的意义。它只是一种逻辑上的概念。上面说到端点就是数据缓冲区，那么就要以想到：管道就是主机与设备端点之间的连接。管道主是数据传输的管道，代表主机的数据缓冲区与设备端点之间交换数据的能力。设备被配置后，端点就可以使用了，因此管道也就是存在了。

作为USB即插即用特点的典型体现，只要设备连接到主机上，端点就可以被访问，即与之相应的管道也就存在了。管道的概念主要用于PC上驱动程序和用户程序的编写，在设计USB设备时一般不会涉及到。

3． 描述符

描述符是一个完整的数据结构，可以通过C语言等编程实现，并存储在USB设备中，用于描述一个USB的所有属性。USB主机是通过一系列的命令来要求设备发送这些信息的。

USB设备的属性包括很多内容，USB1.1协议将这些信息做了多种分类，定义为多种描述符，其中标准的描述符包括以下几种：

1)       设备描述符 (Device Descriptor)

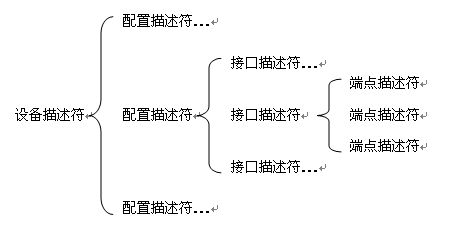
2)       配置描述符 (Cofngiuration Descriptor)

3)       接口描述符 (Interface Descriptor)

4)       端点描述符 (Endpoint Descriptor)

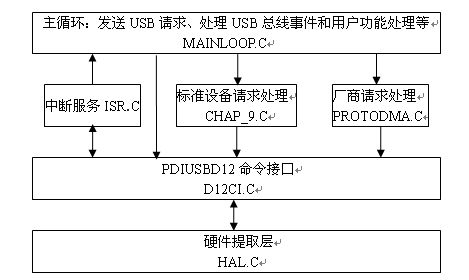
5)       字符串描述符 (String Descriptor)

描述符之间有一定的关系，设备描述符是最高级别的描述符，而端点描述符是最低级的描述符，每一个设备只有一个设备描述符，但设备描述符可以包含多个配置描述符；而一个配置描述符又可以包含多个接口描述符；一个接口使用了几个端点，就有几个端点描述符；字符串描述符是可选的。如下图：

[](http://img.blog.163.com/photo/ePVNik-sa7WUfHZt8wHt8A==/1487032301962941752.jpg)

4．USB固件接口：

   USB固件的积木式结构如图示：

[](http://img.blog.163.com/photo/WaSvFv_sFCbbkZuVBayJwQ==/2324701832653686752.jpg)

主要包括：

硬件提取层：对单片机I/O、数据总线等进行操作。

D12命令接口层：对PDIUSBD12器件进行操作。

中断服务程序：D12向单片机发出中断请求时读取中断传输

数据并设定事件标志位和Setup包数据缓冲区传输给主循环程序。

标准请求处理：对USB的标准请求进行处理。

厂商请求处理：对用户添加的厂商请求进行处理。

      主循环程序：发送USB请求、处理USB事件和用户功能。

5．USB设备与主机的通信

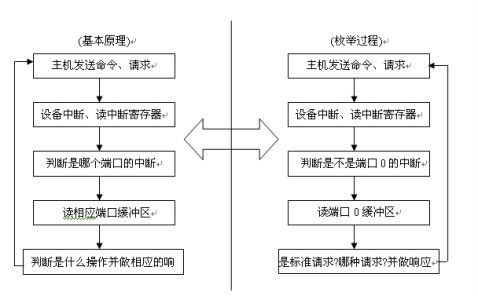
  1).通信的实现：

     初学USB时，很难理解USB设备和主机的通信是怎样实现的，主机和设备的通信包括：命令、数据和握手等步骤。这些步骤是如何实现的呢？其实很简单，可以这样说，通信过程中，命令和数据内容的发送由协议层和HAL层完成，至于扬包和命令包数据包中的其他部分则是由硬件来完成的，用户可以不去考虑。

  2).顺序(枚举顺序)：

主机对USB设备的枚举是USB设备开发的开始，也是整个程序的开始，但对这部分的高度不能通过单步运行的方式来进行调试，因为主机对USB设备的枚举是在很短的时间内完成的，单步调试不能满足时间的要求。所以刚开始时很难理解程序的执行顺序，初学者可以在程序中加入相应的串口输出程序在串口调试助手中来观察程序的执行顺序。

主机和USB设备通信的基本原理（这里用枚举过程来说明）：

[](http://img.blog.163.com/photo/_oykwUp86M_iLIO3Mz4FTA==/1487032301962941823.jpg)

6．枚举：

当USB设备第一次连接在主机上时，要接受主机的枚举和配置，目的就是让主机知道该设备具有什么功能，是哪一类USB设备，需要占用多少的USB资源，使用了哪些传输方式以及传输的数据量多大等等。只有完全确认了这些信息之后，设备才能真正开始工作。

对于U盘来说，枚举还要提供给主机批量传输输出BULK\_OUT端点和输入BULK\_IN端点和端点号以及相应的最大包尺寸。然后主机在后来的批量传输中才能正确地进行一系列操作，所以设备的枚举过程是非常重要的。

枚举过程：

1)       主机通过缺省的地址，端点0来获取设备描述符

2)       主机分配地址给设备

3)       主机将USB复位，进而就可以正式通过地址与设备通信了

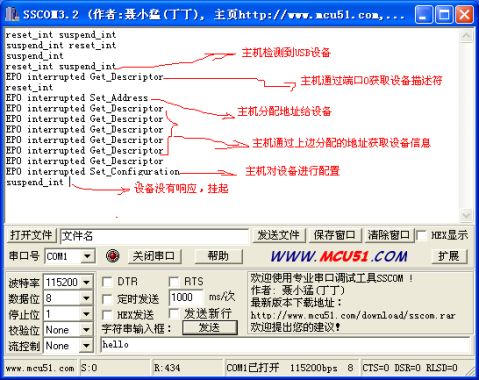
4)       接着主机通过2)中分配给设备的地址获取其他描述符

5)       主机进一步分析设备所有端点的属性后主基本完成了能设备的枚举，最后主机设置配置请求来选择相应的配置，主机和设备在枚举阶段的所有通信工作就完成了

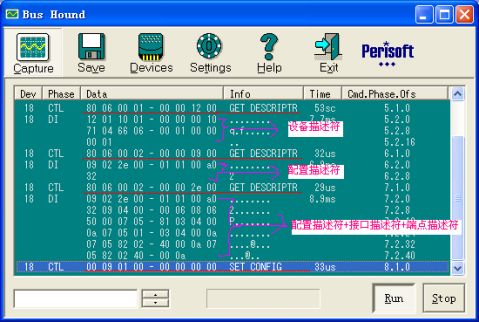
当主机对USB设备枚举完成之后，就可以根据设备的特性与设备进行通信了。在U盘设计中，接下来主机将通过枚举过程中获取的设备信息来对设备进行Mass Storage类相关命令，关于Mass Storage 相关部分我还没做，将在这周内学习。

（一）、USB的枚举过程：

1)用串口调试助手检测程序的执行顺序：

[](http://img.blog.163.com/photo/61EGYOiHgQEpL8dRhtEpUg==/3716595592487750488.jpg)

2)用Bus Hound检测主机USB接口数据：

[](http://img.blog.163.com/photo/fEaF4lmKpXfnhzgzH8cgbA==/3716595592487750498.jpg)

（二）、将USB描述符相关部分设置为Mass Storage相关信息，观察到的结果：

(

设备描述符中：

bDeviceClass字段：0x00，表示Mass Storage类设备;

bDeviceSubClass字段：0x00  àMass Storage 类

bDeviceProtlcol字段：0x00;   à Mass Storage 类

idProduct字段：0x0888;

接口描述符中：

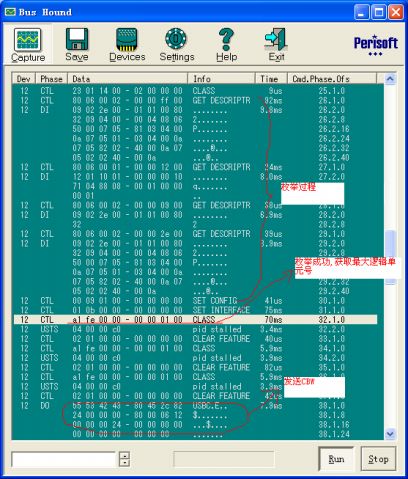
bInterfaceClass字段：0x08;  àMass Storage 类

bInterfaceSubClss字段：0x04; àUFI子类

binterfaceProtocol字段：0x50; à单批量传输

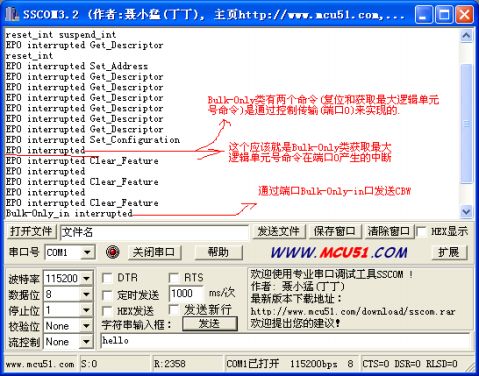
)

在bus hound中观察到：

[](http://img.blog.163.com/photo/lqDG3q0I7G8xe3v-y3tK8g==/2324701832653686895.jpg)

看见主机发送获取最大LUN号和CBW包，就说明枚举成功了。

在串口调试助手中观察到的程序执行顺序：

[](http://img.blog.163.com/photo/c8Fa_hzXR6F33KlIgy3jsw==/1487032301962941938.jpg)

看见Bulk-Only\_in中断说明主机对设备已经枚举成功了，并开始通过端口2和设备进行通信了。

### PC机与USB设备初始阶段总结

1. 集线器检测新设备  
   主机集线器监视着每个端口的信号电压，当有新设备接入时便可觉察。（集线器端口的两根信号线的每一根都有15kΩ的下拉电阻，而每一个设备在D+都有一个1.5kΩ的上拉电阻。当用USB线将PC和设备接通后，设备的上拉电阻使信号线的电位升高，因此被主机集线器检测到。）  
   （2）主机知道了新设备连接后  
   每个集线器用中断传输来报告在集线器上的事件。当主机知道了这个事件，它给集线器发送一个Get\_Status请求来了解更多的消息。返回的消息告诉主机一个设备是什么时候连接的。  
   （3）集线器重新设置这个新设备  
   当主机知道有一个新的设备时，主机给集线器发送一个Set\_Feature请求，请求集线器来重新设置端口。集线器使得设备的USB数据线处于重启（RESET）状态至少10ms。  
   （4）集线器在设备和主机之间建立一个信号通路  
   主机发送一个Get\_Status请求来验证设备是否激起重启状态。返回的数据有一位表示设备仍然处于重启状态。当集线器释放了重启状态，设备就处于默认状态了，即设备已经准备好通过Endpoint 0 的默认流程响应控制传输。即设备现在使用默认地址0x0与主机通信。  
   （5）集线器检测设备速度   
   集线器通过测定那根信号线（D+或D-）在空闲时有更高的电压来检测设备是低速设备还是全速设备。（全速和高速设备D+有上拉电阻，低速设备D-有上拉电阻）。  
   以下，需要USB的firmware进行干预  
   （6）获取最大数据包长度   
   PC向address 0发送USB协议规定的Get\_Device\_Descriptor命令，以取得却缺省控制管道所支持的最大数据包长度，并在有限的时间内等待USB设备的响应，该长度包含在设备描述符的bMaxPacketSize0字段中，其地址偏移量为7，所以这时主机只需读取该描述符的前8个字节。注意，主机一次只能列举一个USB设备，所以同一时刻只能有一个USB设备使用缺省地址0。  
   以下操作雷同，不同操作系统设定时延是不一样的，比如说win2k大概是几毫秒，如果没有反应就再发送一次命令，重复三次。   
   （7）主机分配一个新的地址给设备  
   主机通过发送一个Set\_Address请求来分配一个唯一的地址给设备。设备读取这个请求，返回一个确认，并保存新的地址。从此开始所有通信都使用这个新地址。  
   （8）主机向新地址重新发送Get\_Device\_Descriptor命令，此次读取其设备描述符的全部字段，以了解该设备的总体信息，如VID，PID。  
   （9）主机向设备循环发送Get\_Device\_Configuration命令，要求USB设备回答，以读取全部配置信息。  
   （10）主机发送Get\_Device\_String命令，获得字符集描述（unicode），比如产商、产品描述、型号等等。  
   （11）此时主机将会弹出窗口，展示发现新设备的信息，产商、产品描述、型号等。  
   （12）根据Device\_Descriptor和Device\_Configuration应答，PC判断是否能够提供USB的Driver，一般win2k能提供几大类的设备，如游戏操作杆、存储、打印机、扫描仪等，操作就在后台运行。但是Win98却不可以，所以在此时将会弹出对话框，索要USB的Driver。  
   （13）加载了USB设备驱动以后，主机发送Set\_Configuration（x）命令请求为该设备选择一个合适的配置(x代表非0的配置值)。如果配置成功，USB设备进入“配置”状态，并可以和客户软件进行数据传输。  
   此时，常规的USB完成了其必须进行的配置和连接工作。查看注册表，能够发现相应的项目已经添加完毕，至此设备应当可以开始使用。不过，USB协议还提供了一些用户可选的协议，设备如果不应答，也不会出错，但是会影响到系统的功能。

### USB协议命令以及描述符定义

**一、USB命令**

在USB规范里，对命令一词提供的单词为“Request”，但这里为了更好的理解主机与设备之间的主从关系，将它定义成“命令”。

　　所有的USB设备都要求对主机发给自己的控制命令作出响应，USB规范定义了11个标准命令，它们分别是：Clear\_Feature、Get\_Configuration、Get\_Descriptor、Get\_Interface、Get\_Status、Set\_Address、Set\_Configuration、Set\_Descriptor、Set\_Interface、Set\_Feature、Synch\_Frame。所有USB设备都必须支持这些命令（个别命令除外，如Set\_Descriptor、Synch\_Frame）。

　　不同的命令虽然有不同的数据和使用目的，但所有的USB命令结构是一样的。下表所示为USB命令的结构：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表1、USB命令的结构** | | | | |
| 偏移量 | 域 | 长度（字节） | 值 | 描述 |
| 0 | bmRequestType | 1 | 位图 | 请求特征： D7：传输方向 0=主机至设备 1=设备至主机 D6..5：种类 0=标准 1=类 2=厂商 3=保留 D4..0：接受者 0=设备 1=接口 2=端点 3=其他 4..31 保留 |
| 1 | bRequest | 1 | 值 | 命令类型编码值（见表3） |
| 2 | wValue | 2 | 值 | 根据不同的命令，含义也不同 |
| 4 | wIndex | 2 | 索引或偏移 | 根据不同的命令，含义也不同，主要用于传送索引或偏　移 |
| 6 | wLength | 2 |  | 如有数据传送阶段，此为数据字节数。 |

下表列出了USB的11种标准命令

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **表2、USB的11种标准命令** | | | | | | |
| 命令 | bmRequestType | bRequest | wValue | wIndex | wLength | Data |
| Clear\_Feature | 00000000B 00000001B 00000010B | CLEAR\_FEATURE | 特性选择符 | 零 接口号 端点号 | 零 | 无 |
| Get\_Configuration | 10000000B | GET\_CONFIGURATION | 零 | 零 | 一 | 配置值 |
| Get\_Descriptor | 10000000B | GET\_DESCRIPTOR | 描述表种类(高字节，见表5）和索引（低字节） | 零或语言标志 | 描述表长 | 描述表 |
| Get\_Interface | 10000001B | GET\_INTERFACE | 零 | 接口号 | 一 | 可选设置 |
| Get\_Status | 10000000B 10000001B 10000010B | GET\_STATUS | 零 | 零（返回设备状态） 接口号（对像时接口时） 端点号（对象是端点时） | 二 | 设备, 接口 ,或 端点状态 |
| Set\_Address | 00000000B | SET\_ADDRESS | 设备地址 | 零 | 零 | 无 |
| Set\_Configuration | 00000000B | SET\_CONFIGURATION | 配置值（高字节为0，低字节表示要设置的配置值） | 零 | 零 | 无 |
| Set\_Descriptor | 00000000B | SET\_DESCRIPTOR | 描述表种类（高字节，见表5）和索引（低字节） | 零或语言标志 | 描述表长 | 描述表 |
| Set\_Feature | 00000000B 00000001B 00000010B | SET\_FEATURE | 特性选择符(1表示设备，0表示端点） | 零 接口号 端点号 | 零 | 无 |
| Set\_Interface | 00000001B | SET\_INTERFACE | 可选设置 | 接口号 | 零 | 无 |
| Synch\_Frame | 100000010B | SYNCH\_FRAME | 零 | 端点号 | 二 | 帧号 |

其中bRequest为命令编码值，含意见表3：

|  |  |
| --- | --- |
| **表3、USB标准命令的编码值** | |
| bRequest | Value |
| GET\_STATUS | 0 |
| CLEAR\_FEATURE | 1 |
| 为将来保留 | 2 |
| SET\_FEATURE | 3 |
| 为将来保留 | 4 |
| SET\_ADDRESS | 5 |
| GET\_DESCRIPTOR | 6 |
| SET\_DESCRIPTOR | 7 |
| GET\_CONFIGURATION | 8 |
| SET\_CONFIGURATION | 9 |
| GET\_INTERFACE | 10 |
| SET\_INTERFACE | 11 |
| SYNCH\_FRAME | 12 |

**二、USB描述符**

USB协议为USB设备定义了一套描述设备功能和属性的有固定结构的描述符，包括标准的描述符即设备描述符、配置描述符、接口描述符、端点描述符和字符串描述符，还有百标准描述符，如类描述符。USB设备通过这些描述符向USB主机汇报设备的各种各样属性，主机通过对这些描述符的访问对设备进行类型识别、配置并为其提供相应的客户端驱动程序。

　　USB设备通过描述符反映自己的设备特性。USB描述符是由特定格式排列的一组数据结构组成。

　　在USB设备枚举过程中，主机端的协义软件需要解析从USB设备读取的所有描述符信息。在USB主向设备发送读取描述符的请求后，USB设备将所有的描述符以连续的数据流方式传输给USB主机。主机从第一个读到的字符开始，根据双方规定好的数据格式，顺序地解析读到的数据流。

　　USB描述符包含标准描述符、类描述符和厂商特定描述３种形式。任何一种设备必须USB标准描述符（队字符串描述符可选外）。

　　在USB1.X中，规定了5种标准描述符：设备描述符（Device Descriptor)、配置描述符（Configuration Descriptor）、接口描述符（Interface Descriptor）、端点描述符（Endpoint Descriptor）和字符串描述符（String Descriptor）。

　　每个USB设备只有一个设备描述符，而一个设备中可包含一个或多个配置描述符，即USB设备可以有多种配置。设备的每一个配置中又可以包含一个或多个接口描述符，即USB设备可以支持多种功能（接口），接口的特性通过描述符提供。

　　在USB主机访问USB设备的描述符时，USB设备依照设备描述符、配置描述符、接口描述符、端点描述符、字符串描述符顺序将所有描述符传给主机。一设备至少要包含设备描述符、配置描述符和接口描述符，如果USB设备没有端点描述符，则它仅仅用默认管道与主机进行数据传输。

１、设备描述符

　　设备描述符给出了USB设备的一般信息，包括对设备及在设备配置中起全程作用的信息，包括制造商标识号ID、产品序列号、所属设备类号、默认端点的最大包长度和配置描述符的个数等。一个USB设备必须有且仅有一个设备描述符。设备描述符是设备连接到总线上时USB主机所读取的第一个描述符，它包含了14个字段，结构如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表4、USB设备描述符的结构** | | | | |
| 偏移量 | 域 | 大小 | 值 | 描述 |
| 0 | bLength | 1 | 数字 | 此描述表的字节数 |
| 1 | bDecriptorType | 1 | 常量 | 描述符的类型（此处应为0x01，即设备描述符） |
| 2 | bcdUSB | 2 | BCD码 | 此设备与描述表兼容的USB设备说明版本号（BCD 码） |
| 4 | bDeviceClass | 1 | 类 | 设备类码： 如果此域的值为0则一个设置下每个接口指出它自己的类，各个接口各自独立工作。 如果此域的值处于1~FEH之间，则设备在不同的接口上支持不同的类。并这些接口可能不能独立工作。此值指出了这些接口集体的类定义。 如果此域设为FFH，则此设备的类由厂商定义。 |
| 5 | bDeviceSubClass | 1 | 子类 | 子类挖码 这些码值的具体含义根据bDeviceClass 域来看。 如bDeviceClass 域为零，此域也须为零 如bDeviceClass 域为FFH，此域的所有值保留。 |
| 6 | bDevicePortocol | 1 | 协议 | 协议码 这些码的值视bDeviceClass 和 bDeviceSubClass 的值而定。 如果设备支持设备类相关的协议，此码标志了设备类的值。如果此域的值为零，则此设备不支持设备类相关的协议，然而，可能它的接口支持设备类相关的协议。如果此域的值为FFH，此设备使用厂商定义的协议。 |
| 7 | bMaxPacketSize0 | 1 | 数字 | 端点0的最大包大小（仅8,16,32,64 为合法值） |
| 8 | idVendor | 2 | ID | 厂商标志（由USB-IF组织赋值） |
| 10 | idProduct | 2 | ID | 产品标志（由厂商赋值） |
| 12 | bcdDevice | 2 | BCD 码 | 设备发行号（BCD 码） |
| 14 | iManufacturer | 1 | 索引 | 描述厂商信息的字符串描述符的索引值。 |
| 15 | iProduct | 1 | 索引 | 描述产品信息的字串描述符的索引值。 |
| 16 | iSerialNumber | 1 | 索引 | 描述设备序列号信息的字串描述符的索引值。 |
| 17 | bNumConfigurations | 1 | 数字 | 可能的配置描述符数目 |

其中bDescriptorType为描述符的类型，其含义可查下表（此表也适用于标准命令Get\_Descriptor中wValue域高字节的取值含义）：

|  |  |  |
| --- | --- | --- |
| **表5、USB描述符的类型值** | | |
| 类型 | 描述符 | 描述符值 |
| 标准描述符 | 设备描述符（Device Descriptor) | 0x01 |
| 配置描述符（Configuration Descriptor） | 0x02 |
| 字符串描述符（String Descriptor） | 0x03 |
| 接口描述符（Interface Descriptor） | 0x04 |
| 端点描述符（EndPont Descriptor） | 0x05 |
| 类描述符 | 集线器类描述符（Hub Descriptor） | 0x29 |
| 人机接口类描述符（HID） | 0x21 |
| 厂商定义的描述符 |  | 0xFF |

设备类代码bDeviceClass可查下表：

|  |  |  |
| --- | --- | --- |
| **表6、设备的类别（bDeviceClass）** | | |
| 值（十进制） | 值（十六进制） | 说明 |
| 0 | 0x00 | 接口描述符中提供类的值 |
| 2 | 0x02 | 通信类 |
| 9 | 0x09 | 集线器类 |
| 220 | 0xDC | 用于诊断用途的设备类 |
| 224 | 0xE0 | 无线通信设备类 |
| 255 | 0xFF | 厂商定义的设备类 |

下表列出了一个USB鼠标的设备描述符的例子，供大家分析一下：

|  |  |
| --- | --- |
| **表7、一种鼠标的设备描述符示例** | |
| 字段 | 描述符值（十六制） |
| bLength | 0x12 |
| bDecriptorType | 0x01 |
| bcdUSB | x0110 |
| bDeviceClass | 0x00 |
| bDeviceSubClass | 0x00 |
| bDevicePortocol | 0x00 |
| bMaxPacketSize0 | 0x08 |
| idVendor | 0x045E(Microsoft Corporation） |
| idProduct | 0x0047 |
| bcdDevice | 0x300 |
| iManufacturer | 0x01 |
| iProduct | 0x03 |
| iSerialNumber | 0x00 |
| bNumConfigurations | 0x01 |

2、配置描述符

　　配置描述符中包括了描述符的长度（属于此描述符的所有接口描述符和端点描述符的长度的和）、供电方式（自供电/总线供电）、最大耗电量等。主果主机发出USB标准命令Get\_Descriptor要求得到设备的某个配置描述符，那么除了此配置描述符以外，此配置包含的所有接口描述符与端点描述符都将提供给USB主机。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表8、USB配置描述符的结构** | | | | |
| 偏移量 | 域 | 大小 | 值 | 描述 |
| 0 | bLength | 1 | 数字 | 此描述表的字节数长度。 |
| 1 | bDescriptorType | 1 | 常量 | 配置描述表类型（此处为0x02） |
| 2 | wTotalLength | 2 | 数字 | 此配置信息的总长（包括配置，接口，端点和设备类及厂商定义的描述符） |
| 4 | bNumInterfaces | 1 | 数字 | 此配置所支持的接口个数 |
| 5 | bCongfigurationValue | 1 | 数字 | 在SetConfiguration（）请求中用作参数来选定此配置。 |
| 6 | iConfiguration | 1 | 索引 | 描述此配置的字串描述表索引 |
| 7 | bmAttributes | 1 | 位图 | 配置特性： D7： 保留（设为一） D6： 自给电源 D5： 远程唤醒 D4..0：保留（设为一） 一个既用总线电源又有自给电源的设备会在MaxPower域指出需要从总线取的电量。并设置D6为一。运行时期的实际电源模式可由GetStatus(DEVICE) 请求得到。 |
| 8 | MaxPower | 1 | mA | 在此配置下的总线电源耗费量。以 2mA 为一个单位。 |

下面是一种硬盘的配置描述符示例：

|  |  |
| --- | --- |
| **表9、一种硬盘的配置描述符示例** | |
| 字段 | 描述符值（十六进制） |
| bLength | 0x09 |
| bDescriptorType | 0x02 |
| wTotalLength | 0x01F |
| bNumInterfaces | 0x01 |
| bCongfigurationValue | 0x01 |
| iConfiguration | 0x00 |
| bmAttributes | 0x0C |
| MaxPower | 0x32 |

3、接口描述符

　　配置描述符中包含了一个或多个接口描述符，这里的“接口”并不是指物理存在的接口，在这里把它称之为“功能”更易理解些，例如一个设备既有录音的功能又有扬声器的功能，则这个设备至少就有两个“接口”。

　　如果一个配置描述符不止支持一个接口描述符，并且每个接口描述符都有一个或多个端点描述符，那么在响应USB主机的配置描述符命令时，USB设备的端点描述符总是紧跟着相关的接口描述符后面，作为配置描述符的一部分被返回。接口描述符不可直接用Set\_Descriptor和Get\_Descriptor来存取。

　　如果一个接口仅使用端点0，则接口描述符以后就不再返回端点描述符，并且此接口表现的是一个控制接口的特性，它使用与端点0相关联的默认管道进行数据传输。在这种情况下bNumberEndpoints域应被设置成0。接口描述符在说明端点个数并不把端点0计算在内。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表10、USB接口描述符的结构** | | | | |
| **偏移量** | **域** | **大小** | **值** | **说明** |
| 0 | bLength | 1 | 数字 | 此表的字节数 |
| 1 | bDescriptorType | 1 | 常量 | 接口描述表类（此处应为0x04） |
| 2 | bInterfaceNumber | 1 | 数字 | 接口号，当前配置支持的接口数组索引（从零开始）。 |
| 3 | bAlternateSetting | 1 | 数字 | 可选设置的索引值。 |
| 4 | bNumEndpoints | 1 | 数字 | 此接口用的端点数量，如果是零则说明此接口只用缺省控制管道。 |
| 5 | bInterfaceClass | 1 | 类 | 接口所属的类值： 零值为将来的标准保留。 如果此域的值设为FFH，则此接口类由厂商说明。 所有其它的值由USB 说明保留。 |
| 6 | bInterfaceSubClass | 1 | 子类 | 子类码 这些值的定义视bInterfaceClass域而定。 如果bInterfaceClass域的值为零则此域的值必须为零。 bInterfaceClass域不为FFH则所有值由USB 所保留。 |
| 7 | bInterfaceProtocol | 1 | 协议 | 协议码：bInterfaceClass 和bInterfaceSubClass 域的值而定.如果一个接口支持设备类相关的请求此域的值指出了设备类说明中所定义的协议. |
| 8 | iInterface | 1 | 索引 | 描述此接口的字串描述表的索引值。 |

对于bInterfaceClass字段，表示接口所属的类别，USB协议根据功能将不同的接口划分成不的类，其具体含义如下表所示：

|  |  |
| --- | --- |
| **表11、USB协议定义的接口类别（bInterfaceClass）** | |
| 值（十六进制） | 类别 |
| 0x01 | 音频类 |
| 0x02 | CDC控制类 |
| 0x03 | 人机接口类（HID） |
| 0x05 | 物理类 |
| 0x06 | 图像类 |
| 0x07 | 打印机类 |
| 0x08 | 大数据存储类 |
| 0x09 | 集线器类 |
| 0x0A | CDC数据类 |
| 0x0B | 智能卡类 |
| 0x0D | 安全类 |
| 0xDC | 诊断设备类 |
| 0xE0 | 无线控制器类 |
| 0xFE | 特定应用类（包括红外的桥接器等） |
| 0xFF | 厂商定义的设备 |

4、端点描述符

　　端点是设备与主机之间进行数据传输的逻辑接口，除配置使用的端点0（控制端点，一般一个设备只有一个控制端点）为双向端口外，其它均为单向。端点描述符描述了数据的传输类型、传输方向、数据包大小和端点号（也可称为端点地址）等。

　　除了描述符中描述的端点外，每个设备必须要有一个默认的控制型端点，地址为0，它的数据传输为双向，而且没有专门的描述符，只是在设备描述符中定义了它的最大包长度。主机通过此端点向设备发送命令，获得设备的各种描述符的信息，并通过它来配置设备。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表12、USB端点描述符的结构** | | | | |
| **偏移量** | **域** | **大小** | **值** | **说明** |
| 0 | bLength | 1 | 数字 | 此描述表的字节数长度 |
| 1 | bDescriptorType | 1 | 常量 | 端点描述表类（此处应为0x05） |
| 2 | bEndpointAddress | 1 | 端点 | 此描述表所描述的端点的地址、方向： Bit 3..0 : 端点号. Bit 6..4 : 保留,为零 Bit 7: 方向,如果控制端点则略。 0：输出端点（主机到设备） 1：输入端点（设备到主机） |
| 3 | bmAttributes | 1 | 位图 | 此域的值描述的是在bConfigurationValue域所指的配置下端点的特性。 Bit 1..0 :传送类型 00=控制传送 01=同步传送 10=批传送 11=中断传送 所有其它的位都保留。 |
| 4 | wMaxPacketSize | 2 | 数字 | 当前配置下此端点能够接收或发送的最大数据包的大小。 对于实进传输，此值用于为每帧的数据净负荷预留时间。在实际运行时，管道可能不完全需要预留的带宽，实际带宽可由设备通过一种非USB定义的机制汇报给主机。对于中断传输，批量传输和控制传输，端点可能发送比之短的数据包 |
| 6 | bInterval | 1 | 数字 | 周期数据传输端点的时间间隙。 此域的值对于批传送的端点及控制传送的端点无意义。对于同步传送的端点此域必需为1，表示周期为1ms。对于中断传送的端点此域值的范围为1ms到255ms。 |

下表是一种鼠标的端点描述符的示例，该端点是一个中断端点：

|  |  |
| --- | --- |
| **表13、一种鼠标的端点描述符示例** | |
| 域 | 值（十六进制） |
| bLength | 0x07 |
| bDescriptorType | 0x05 |
| bEndpointAddress | 0x81 |
| bmAttributes | 0x03 |
| wMaxPacketSize | 0x04 |
| bInterval | 0x0A |

5、字符串描述符

　　字符串描述符是一种可选的USB标准描述符，描述了如制商、设备名称或序列号等信息。如果一个设备无字符串描述符，则其它描述符中与字符串有关的索引值都必须为0。字符串使用的是Unicode编码。

　　主机请示得到某个字符串描述符时一般分成两步：首先主机向设备发出USB标准命令Get\_Descriptor，其中所使用的字符串的索引值为0，设备返回一个字符串描述符，此描述符的结构如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表14、USB字符串描述符（响应主机请求时返回的表示语言ID的字符串描述符）** | | | | |
| **偏移量** | **域** | **大小** | **值** | **描述** |
| 0 | bLength | 1 | N+2 | 此描述表的字节数 |
| 1 | bDescriptorType | 1 | 常量 | 字串描述表类型（此处应为0x03） |
| 2 | wLANGID[0] | 2 | 数字 | 语言标识（LANGID） 码0 |
| **…** | **…** | **…** | **…** | **…** |
| N | wLANGID[x] | 2 | 数字 | 语言标识（LANGID） 码X |

该字符串描述符双字节的语言ID的数组，wLANGID[0]~wLANGID[x]指明了设备支持的语言，具体含义可查看[USB\_LANGIDs.pdf](http://www.baiheee.com/Documents/090518/090518112619/USB_LANGIDs.pdf)。

　　主机根据自己需要的语言，再次向设备发出USB标准命令Get\_Descriptor，指明所要求得到的字符串的索引值和语言。这次设备所返回的是Unicode编号的字符串描述符，其结构如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表15、Unicode字符串描述符（响应主机请求时真正表示字符串编码的字符串描述符）** | | | | |
| **偏移量** | **域** | **大小** | **值** | **描述** |
| 0 | bLength | 1 | 数字 | 此描述表的字节数（bString域的数值N＋2） |
| 1 | bDescriptorType | 1 | 常量 | 字串描述表类型（此处应为0x03） |
| 2 | bString | N | 数字 | UNICODE 编码的字串 |

# 软件协议层

## MODBUS

物理层一般可以使用RS485

## PROFIBUS