专题21-USB驱动程序设计

一、USB总线介绍

1.1、USB发展史

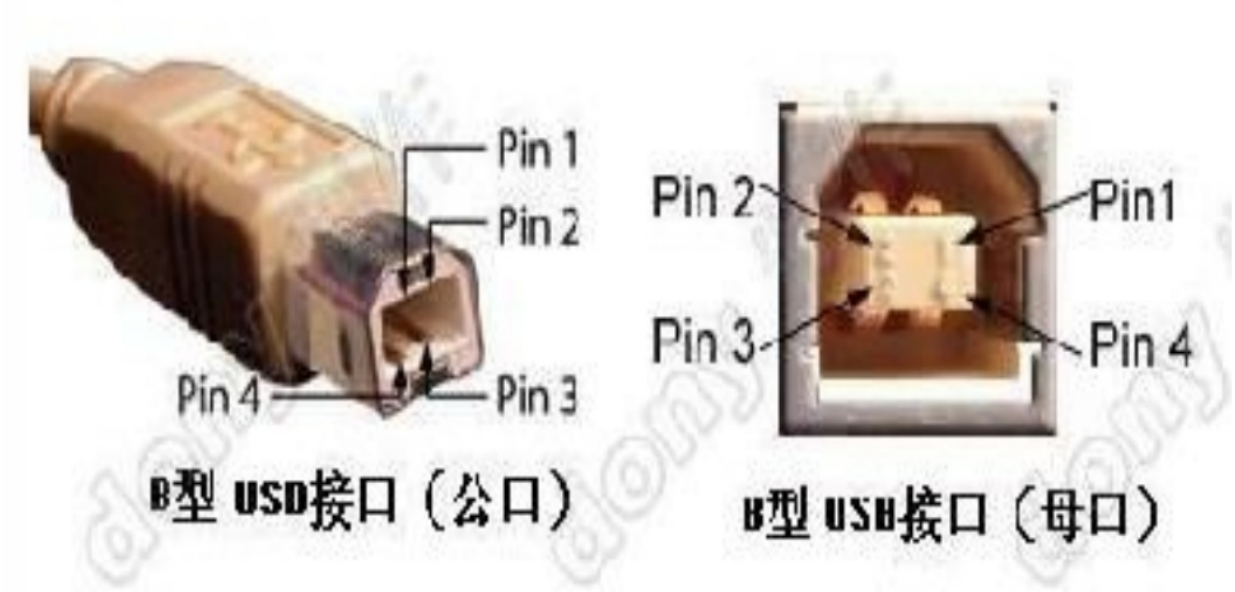
USB(Universal Serial Bus)，通用串行总 线，是一种外部总线标准，用于规范电脑与 外部设备的连接和通讯。USB是在1994年底由英特尔、康柏、IBM、Microsoft等多家公 司联合提出的，自1996年推出后，已成功替 代串口和并口，成为当今个人电脑和大量 智能设备的必配接口之一。

•USB 1.0出现在1996年的，速度只有1.5Mb/s ,1998年升级为USB 1.1，速度也提升到12Mb/s， 称之为" full speed"
•USB2.0规范是由USB1.1规范演变而来的。它的 传输速率达到了 480Mbps ，称之为 " high speed"
•USB3.0提供了十倍于USB 2.0的传输速度和更 高的节能效率，被称为" super speed"

1.2、USB硬件接口

1.2.1、标准A口

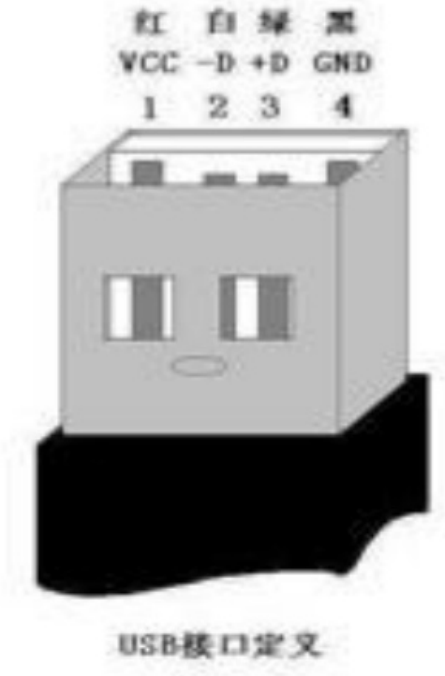
1.2.1、标准B口



1.2.1、mini-usb口

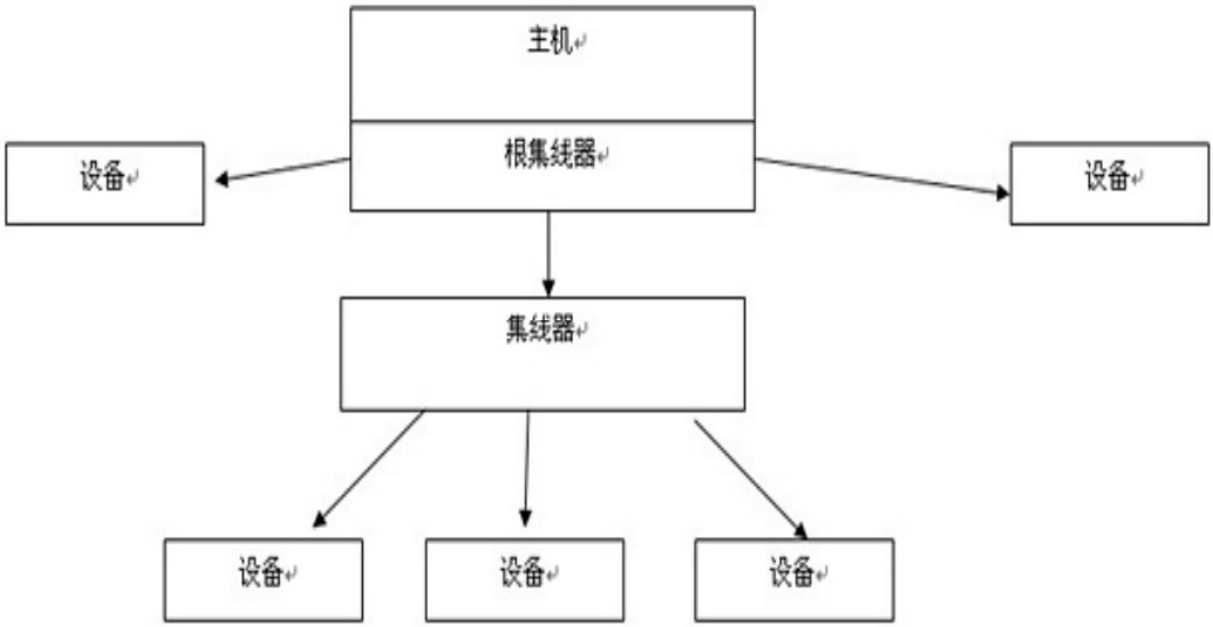
1.2.1、micro-usb口

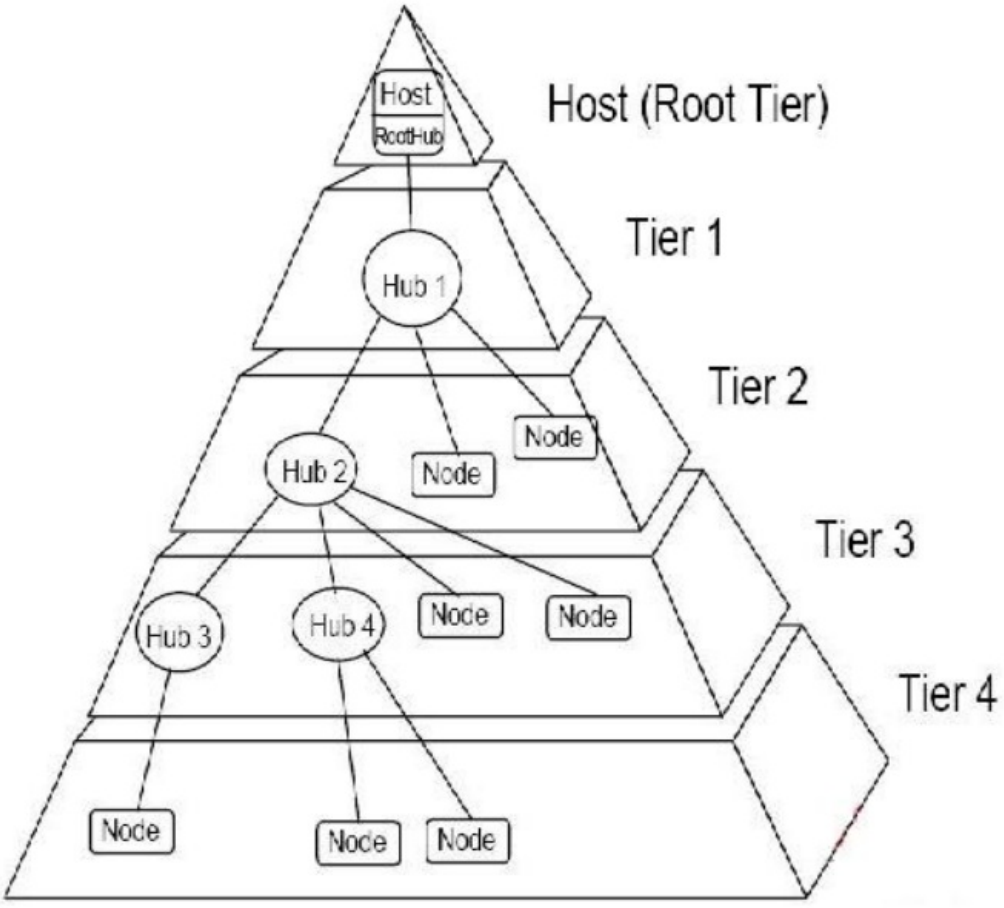
1.2.5、USB信号线



USB接口有4根线，两根电源线，两根信 号线。USB接口的输出电压和电流是： +5V 500mA
USB电源： 标有VCC、Power、5V、 5VSB字样
USB数据线：（正）DATA+、USBD+ 、PD+、USBDT+
USB数据线：（负）DATA-、USBD- 、PD-、USBDT-
地线： GND、Ground

1.3、USB系统拓扑结构





对于每个USB系统来说，都有一个称为主机 控制器的设备，该控制器和一个根Hub作为 一个整体。这个根Hub下可以接多级的Hub， 每个子Hub又可以接子Hub。每个USB设备 作为一个节点接在不同级别的Hub上。 每条 USB总线上最多可以接127个设备。

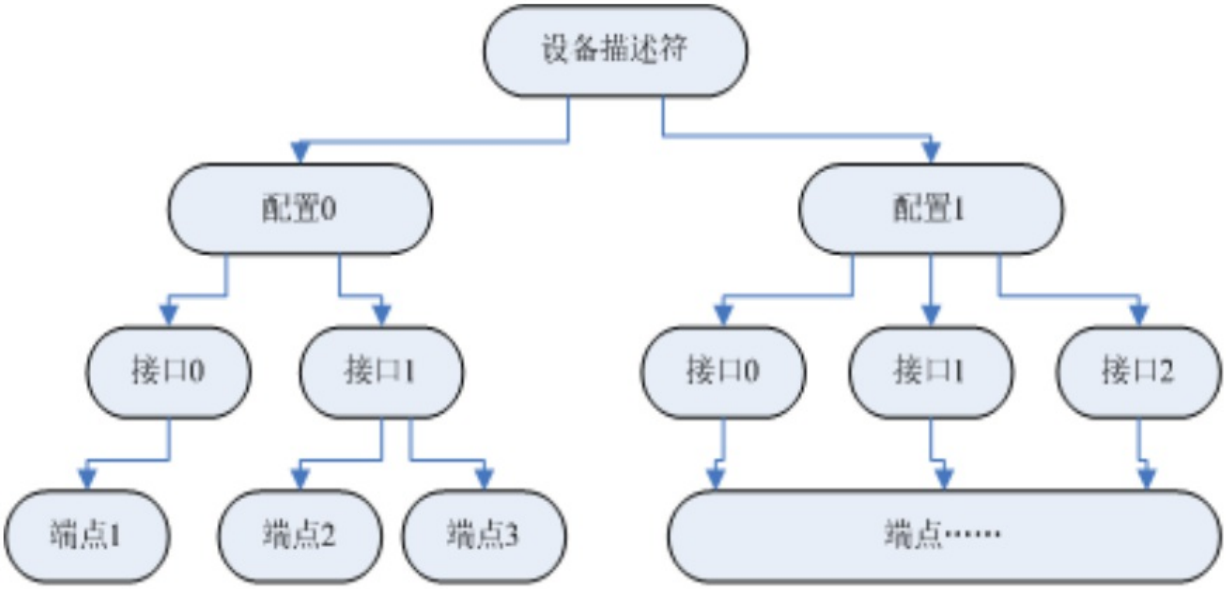
常见的USB主控制器规格有：
OHCI：主要是非PC系统上的USB芯片
UHCI：大多是Intel和Via主板上的USB控制 器芯片。他们都是由USB1.1规格的。
EHCI：是有Intel等几个厂商研发，兼容OHCI UHCI ，遵循USB2.0规范。

二、USB协议分析

2.1、USB设备逻辑结构

2.1.1、逻辑组织结构

在USB设备的逻辑组织中，包含设备、配置、接口和端点4个层次。设 备通常有一个或多个配置,配置通常有一个或多个接口,接口有零或多个 端点。



USB的设备、配置、接口、端点

每个USB设备都可以包含一个或多个配置，不同的配置使设备表 现出不同的功能组合，配置由多个接口组成。在USB协议中，接口代表一个基本的功能，一个功能复杂的USB设备可以具有多个 接口,而接口是端点的汇集。

一个USB播放器带有音频,视频功能,还有旋钮和按钮。
配置1: 音频(接口)+旋钮(接口)
配置2: 视频(接口)+旋钮(接口)
配置3: 音频(接口)+视频(接口)+按钮(接口)
音频接口,视频接口,按钮接口,旋钮接口均需要一个驱动程序。

USB设备中的唯一可寻址的部分是设备端点，端点 的作用类似于寄存器。每个端点在设备内部有唯一 的端点号，这个端点号是在设备设计时给定的。主 机和设备的通信最终都作用于设备上的各个端点。 每个端点所支持的操作都是单向的，要么只读，要 么只写。

2.1.2、描述符

Why?

当我们把USB设 备（如：USB鼠 标）插到我们的 PC时，主机能够自动识别出我们 的USB设备类型

在每一个USB设备内部，包含了固定格式的数 据，通过这些数据，USB主机就可以获取USB 设备的类型、生产厂商等信息。这些固定格式的 数据，我们就称之为USB描述符。标准的USB 设备有5种USB描述符：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。

2.1.2.1、设备描述符

一个USB设备只有 一个设备描述符， 设备描述符长度为 18个字节，格式如 左图（《USB specification :Ta ble-9.8》）

Table 9-8. Standard Device Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	DEVICE Descriptor Type
2	*bcdUSB*	2	BCD	USB Specification Release Number in Binary-Coded Decimal (i.e., 2.10 is 210H). This field identifies the release of the USB Specification with which the device and its descriptors are compliant.
4	*bDeviceClass*	1	Class	Class code (assigned by the USB-IF). If this field is reset to zero, each interface within a configuration specifies its own class information and the various interfaces operate independently. If this field is set to a value between 1 and FEH, the device supports different class specifications on different interfaces and the interfaces may not operate independently. This value identifies the class definition used for the aggregate interfaces. If this field is set to FFH, the device class is vendor-specific.
5	*bDeviceSubClass*	1	SubClass	Subclass code (assigned by the USB-IF). These codes are qualified by the value of the *bDeviceClass* field. If the *bDeviceClass* field is reset to zero, this field must also be reset to zero. If the *bDeviceClass* field is not set to FFH, all values are reserved for assignment by the USB-IF.

Table 9-8. Standard Device Descriptor (Continued)

Offset	Field	Size	Value	Description
6	*bDeviceProtocol*	1	Protocol	Protocol code (assigned by the USB-IF). These codes are qualified by the value of the *bDeviceClass* and the *bDeviceSubClass* fields. If a device supports class-specific protocols on a device basis as opposed to an interface basis, this code identifies the protocols that the device uses as defined by the specification of the device class. If this field is reset to zero, the device does not use class-specific protocols on a device basis. However, it may use class-specific protocols on an interface basis. If this field is set to FFH, the device uses a vendor-specific protocol on a device basis.
7	*bMaxPacketSize0*	1	Number	Maximum packet size for endpoint zero (only 8, 16, 32, or 64 are valid)
8	*idVendor*	2	ID	Vendor ID (assigned by the USB-IF)
10	*idProduct*	2	ID	Product ID (assigned by the manufacturer)
12	*bcdDevice*	2	BCD	Device release number in binary-coded decimal
14	*iManufacturer*	1	Index	Index of string descriptor describing manufacturer
15	*iProduct*	1	Index	Index of string descriptor describing product
16	*iSerialNumber*	1	Index	Index of string descriptor describing the device's serial number
17	*bNumConfigurations*	1	Number	Number of possible configurations

bLength : 描述符长度，固定为0x12。
bDescriptorType : 设备描述符类型，固定为0x01。
bcdUSB : USB 规范发布号。表示了本设备能适用于那种协议，如2.0=0200
bDeviceClass : 类型代码。
bDeviceSubClass : 子类型代码。
bDeviceProtocol : 协议代码。
bMaxPacketSize0 : 端点0最大分组大小。
idVendor : 供应商ID。
idProduct : 产品ID（由厂商分配）。
bcdDevice : 设备出产编码，由厂家自行设置。
iManufacturer : 厂商描述符字符串索引．索引到对应的字符串描述符。
iProduct : :产品描述符字符串索引。
iSerialNumber : 设备序列号字符串索引。
bNumConfigurations : 可能的配置数。

2.1.2.2、配置描述符

USB配置描述符长度为8个字节，格式如下图（《USB specification :Table-9.10》）

Table 9-10. Standard Configuration Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	CONFIGURATION Descriptor Type
2	*wTotalLength*	2	Number	Total length of data returned for this configuration. Includes the combined length of all descriptors (configuration, interface, endpoint, and class- or vendor-specific) returned for this configuration.
4	*bNumInterfaces*	1	Number	Number of interfaces supported by this configuration
5	*bConfigurationValue*	1	Number	Value to use as an argument to the SetConfiguration() request to select this configuration
6	*iConfiguration*	1	Index	Index of string descriptor describing this configuration

Table 9-10. Standard Configuration Descriptor (Continued)

Offset	Field	Size	Value	Description
7	*bmAttributes*	1	Bitmap	Configuration characteristics D7: Reserved (set to one) D6: Self-powered D5: Remote Wakeup D4...0: Reserved (reset to zero) D7 is reserved and must be set to one for historical reasons. A device configuration that uses power from the bus and a local source reports a non-zero value in *bMaxPower* to indicate the amount of bus power required and sets D6. The actual power source at runtime may be determined using the GetStatus(DEVICE) request (see Section 9.4.5). If a device configuration supports remote wakeup, D5 is set to one.
8	*bMaxPower*	1	mA	Maximum power consumption of the USB device from the bus in this specific configuration when the device is fully operational. Expressed in 2 mA units (i.e., 50 = 100 mA). Note: A device configuration reports whether the configuration is bus-powered or self-powered. Device status reports whether the device is currently self-powered. If a device is disconnected from its external power source, it updates device status to indicate that it is no longer self-powered. A device may not increase its power draw from the bus, when it loses its external power source, beyond the amount reported by its configuration. If a device can continue to operate when disconnected from its external power source, it continues to do so. If the device cannot continue to operate, it fails operations it can no longer support. The USB System Software may determine the cause of the failure by checking the status and noting the loss of the device's power source.

bLength : 描述符长度，固定为0x09。
bDescriptorType : 配置描述符类型，固定为0x02。
wTotalLength : 返回整个数据的长度，指此配置返回的配置描述符，接口描述符以及端点描述符的全部大小。
bNumInterfaces : 配置所支持的接口数，指该配置配备的接口数量，也表示该配置下接口描述符数量。
bConfigurationValue : 作为Set Configuration的一个参数选择配置值。
iConfiguration : 用于描述该配置字符串描述符的索引。
bmAttributes : 供电模式选择。Bit4-0保留，D7:总线供电，D6:自供电，D5:远程唤醒 .
MaxPower : 总线供电的USB设备的最大消耗电流，以2mA为单位。

2.1.2.3、接口描述符

USB接口描述符长 度为8个字节 《USB specification》 Table-9.12

Table 9-12. Standard Interface Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	INTERFACE Descriptor Type
2	*bInterfaceNumber*	1	Number	Number of this interface. Zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
3	*bAlternateSetting*	1	Number	Value used to select this alternate setting for the interface identified in the prior field
4	*bNumEndpoints*	1	Number	Number of endpoints used by this interface (excluding endpoint zero). If this value is zero, this interface only uses the Default Control Pipe.
5	*bInterfaceClass*	1	Class	Class code (assigned by the USB-IF). A value of zero is reserved for future standardization. If this field is set to FFH, the interface class is vendor-specific. All other values are reserved for assignment by the USB-IF.
6	*bInterfaceSubClass*	1	SubClass	Subclass code (assigned by the USB-IF). These codes are qualified by the value of the *bInterfaceClass* field. If the *bInterfaceClass* field is reset to zero, this field must also be reset to zero. If the *bInterfaceClass* field is not set to FFH, all values are reserved for assignment by the USB-IF.

Table 9-12. Standard Interface Descriptor (Continued)

Offset	Field	Size	Value	Description
7	*bInterfaceProtocol*	1	Protocol	Protocol code (assigned by the USB). These codes are qualified by the value of the *bInterfaceClass* and the *bInterfaceSubClass* fields. If an interface supports class-specific requests, this code identifies the protocols that the device uses as defined by the specification of the device class. If this field is reset to zero, the device does not use a class-specific protocol on this interface. If this field is set to FFH, the device uses a vendor-specific protocol for this interface.
8	*iInterface*	1	Index	Index of string descriptor describing this interface

bLength : 描述符长度，固定为0x09。
bDescriptorType : 接口描述符类型，固定为0x04。
bInterfaceNumber: 该接口的编号。
bAlternateSetting : 用于为上一个字段选择可供替换的设置。
bNumEndpoint : 使用的端点数目，端点0除外。
bInterfaceClass : 类型代码（由USB组织分配）。
bInterfaceSunClass : 子类型代码（由USB组织分配）。

bInterfaceProtocol : 协议代码（由USB组织分配）。
iInterface : 字符串描述符的索引。

2.1.2.4、端点描述符

USB端点描述符长度为7个字节，格式如下图（USB specification :Table-9.13）

Table 9-13. Standard Endpoint Descriptor

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	ENDPOINT Descriptor Type
2	*bEndpointAddress*	1	Endpoint	The address of the endpoint on the USB device described by this descriptor. The address is encoded as follows: Bit 3...0: The endpoint number Bit 6...4: Reserved, reset to zero Bit 7: Direction, ignored for control endpoints 0 = OUT endpoint 1 = IN endpoint

Table 9-13. Standard Endpoint Descriptor (Continued)

Offset	Field	Size	Value	Description
3	*bmAttributes*	1	Bitmap	This field describes the endpoint's attributes when it is configured using the *bConfigurationValue*. Bits 1..0: Transfer Type 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt If not an isochronous endpoint, bits 5..2 are reserved and must be set to zero. If isochronous, they are defined as follows: Bits 3..2: Synchronization Type 00 = No Synchronization 01 = Asynchronous 10 = Adaptive 11 = Synchronous Bits 5..4: Usage Type 00 = Data endpoint 01 = Feedback endpoint 10 = Implicit feedback Data endpoint 11 = Reserved Refer to Chapter 5 for more information. All other bits are reserved and must be reset to zero. Reserved bits must be ignored by the host.

Table 9-13. Standard Endpoint Descriptor (Continued)

Offset	Field	Size	Value	Description
4	*wMaxPacketSize*	2	Number	Maximum packet size this endpoint is capable of sending or receiving when this configuration is selected. For isochronous endpoints, this value is used to reserve the bus time in the schedule, required for the per-(micro)frame data payloads. The pipe may, on an ongoing basis, actually use less bandwidth than that reserved. The device reports, if necessary, the actual bandwidth used via its normal, non-USB defined mechanisms. For all endpoints, bits 10..0 specify the maximum packet size (in bytes). For high-speed isochronous and interrupt endpoints: Bits 12..11 specify the number of additional transaction opportunities per microframe: 00 = None (1 transaction per microframe) 01 = 1 additional (2 per microframe) 10 = 2 additional (3 per microframe) 11 = Reserved Bits 15..13 are reserved and must be set to zero. Refer to Chapter 5 for more information.
6	*bInterval*	1	Number	Interval for polling endpoint for data transfers. Expressed in frames or microframes depending on the device operating speed (i.e., either 1 millisecond or 125 µs units). For full-/high-speed isochronous endpoints, this value must be in the range from 1 to 16. The *bInterval* value is used as the exponent for a $2^{(bInterval-1)}$ value; e.g., a *bInterval* of 4 means a period of 8 ($2^{(4-1)}$). For full-/low-speed interrupt endpoints, the value of this field may be from 1 to 255. For high-speed interrupt endpoints, the *bInterval* value is used as the exponent for a $2^{(bInterval-1)}$ value; e.g., a *bInterval* of 4 means a period of 8 ($2^{(4-1)}$). This value must be from 1 to 16. For high-speed bulk/control OUT endpoints, the *bInterval* must specify the maximum NAK rate of the endpoint. A value of 0 indicates the endpoint never NAKs. Other values indicate at most 1 NAK each *bInterval* number of microframes. This value must be in the range from 0 to 255. See Chapter 5 description of periods for more detail.

bLength : 描述符大小，固定为0x07。
bDescriptorType : 接口描述符类型，固定为0x05。
bEndpointType : USB设备的端点地址。Bit7，方向，对于控制端点可以忽略，1/0:IN/OUT。Bit6-4，保留。BIt3-0 : 端点号 .
bmAttributes : 端点属性，Bit7-2，保留。BIt1-0：00控制，01同步，02批量，03中断。
wMaxPacketSize : 本端点接收或发送的最大信息包大小。
bInterval : 轮训数据传送端点的时间间隔 . 对于批量传送和控制传送的端点忽略 . 对于同步传送的端点，必须为１，对于中断传送的端点，范围为1-255

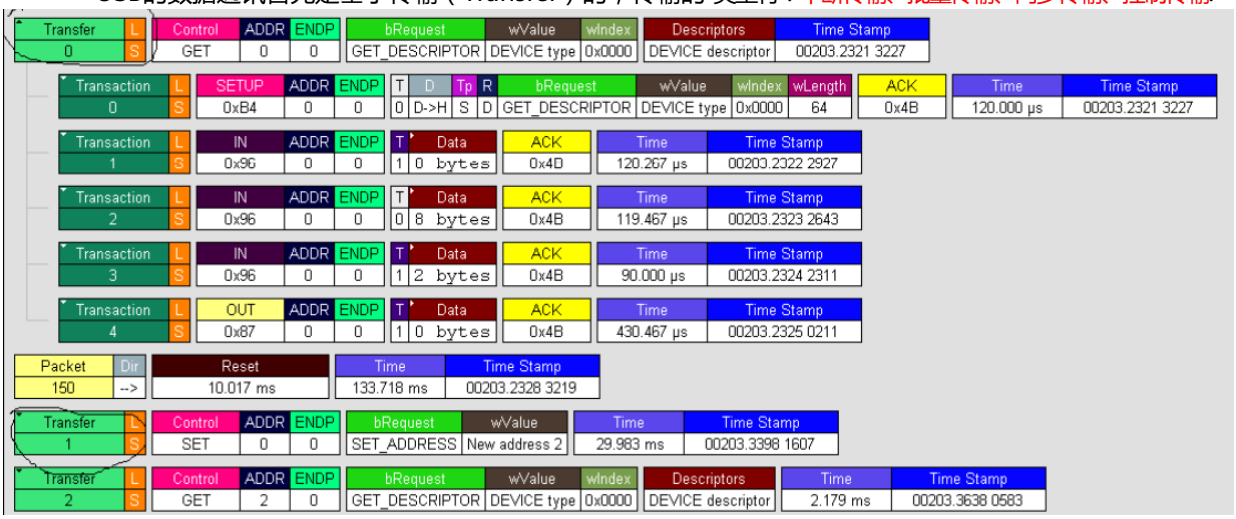
2.1.2.5、字符串描述符

2.2、USB数据通讯

2.2.1、通讯模型

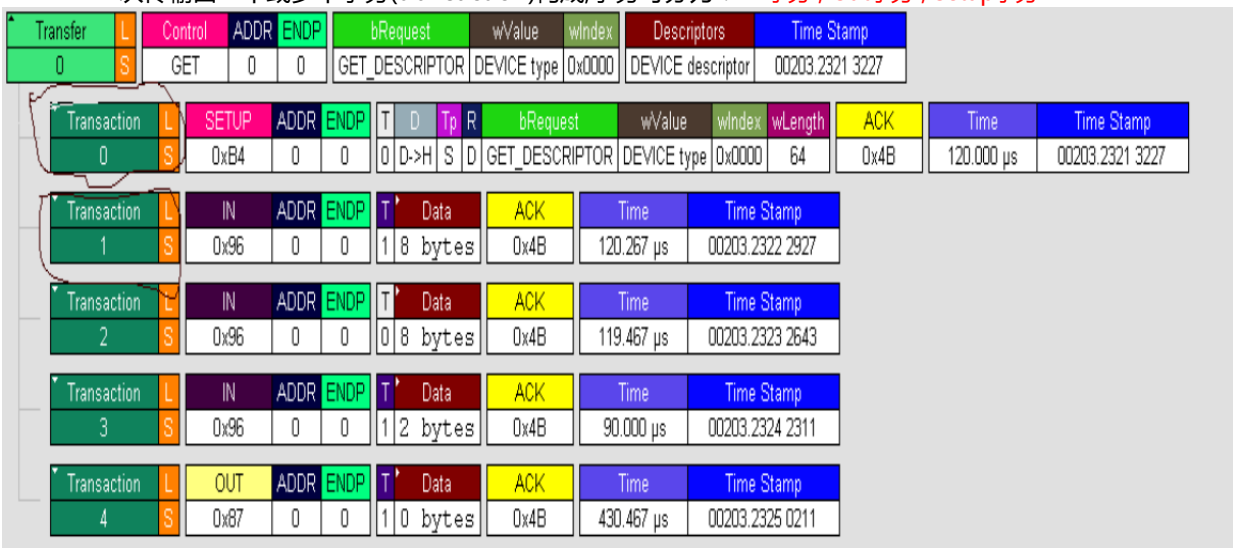
2.2.2、传输

USB的数据通讯首先是基于传输（Transfer）的，传输的 类型有：中断传输、批量传输、同步传输、控制传输.



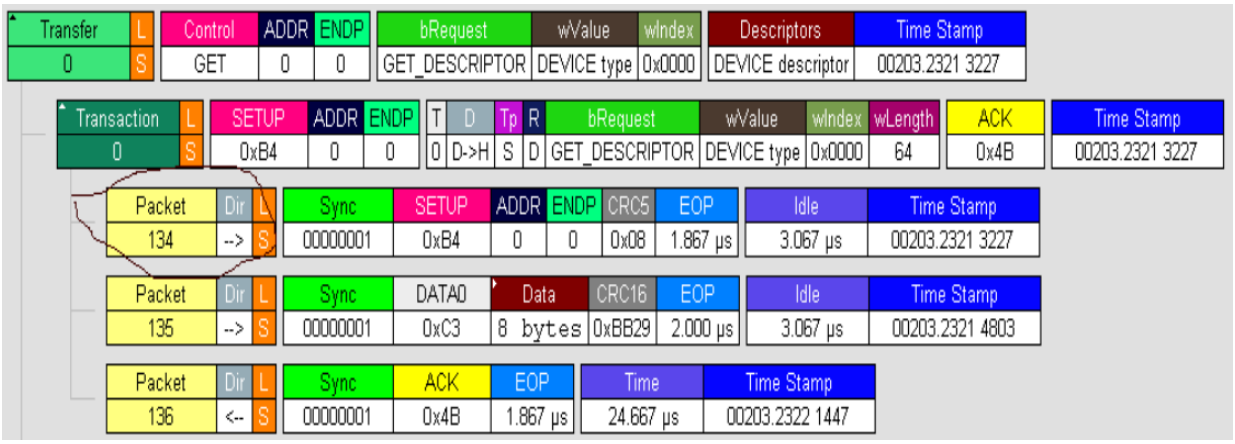
2.2.3、事务

一次传输由一个或多个事务(transaction)构成,事务可分为： In事务，Out事务，Setup事务



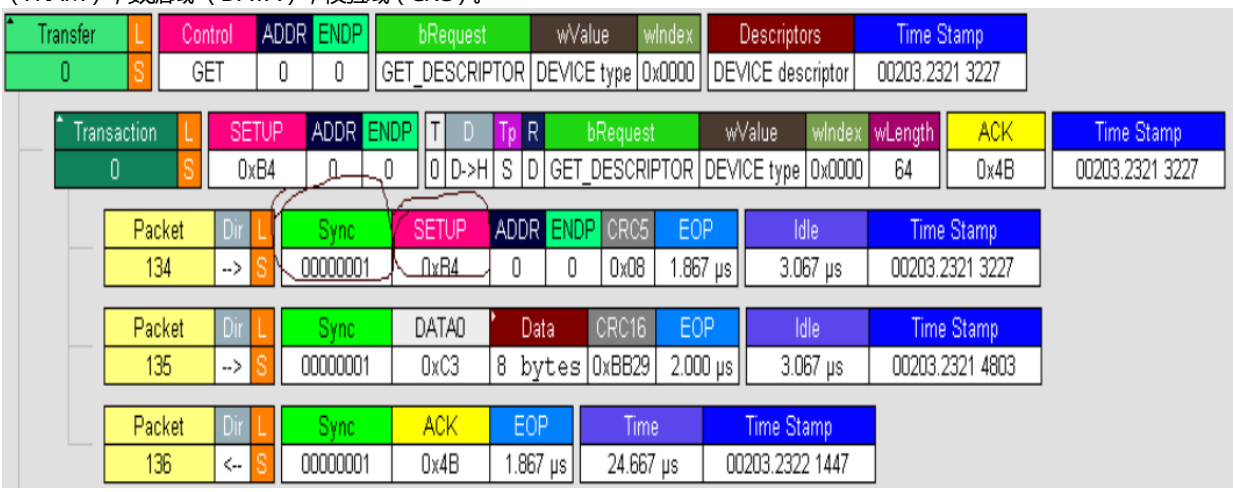
2.2.4、包

一个事务由一个或多个包(packet)构成,包可分为：令牌包 (setup)、数据包(data)、握手包(ACK)和特殊包



2.2.5、域

一个包由多个域构成,域可分为：同步域（SYNC），标识域（PID）， 地址域（ADDR），端点域（ENDP），帧号域（FRAM），数据域 （DATA），校验域（CRC）。



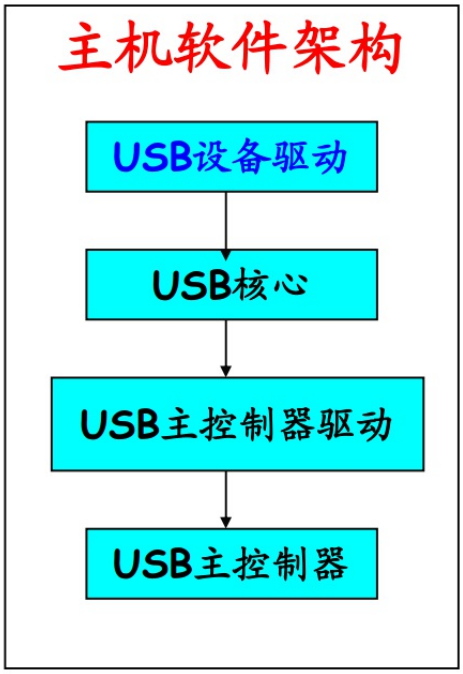
2.3、USB设备枚举

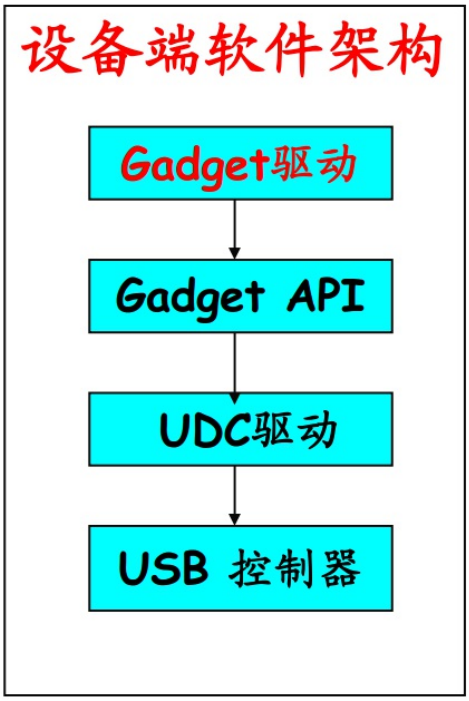
USB设备在正常工作以前, 第一件要做的事 就是枚举。枚举是让主机认得这个USB设 备, 并且为该设备准备资源，建立好主机和 设备之间的数据传递通道。

1. 获取设备描述符
2. 复位
3. 设置地址
4. 再次获取设备描述符
5. 获取配置描述符
6. 获取接口、端点描述符
7. 获取字符串描述符
8. 选择设备配置

三、Linux USB系统架构

3.1、软件系统架构

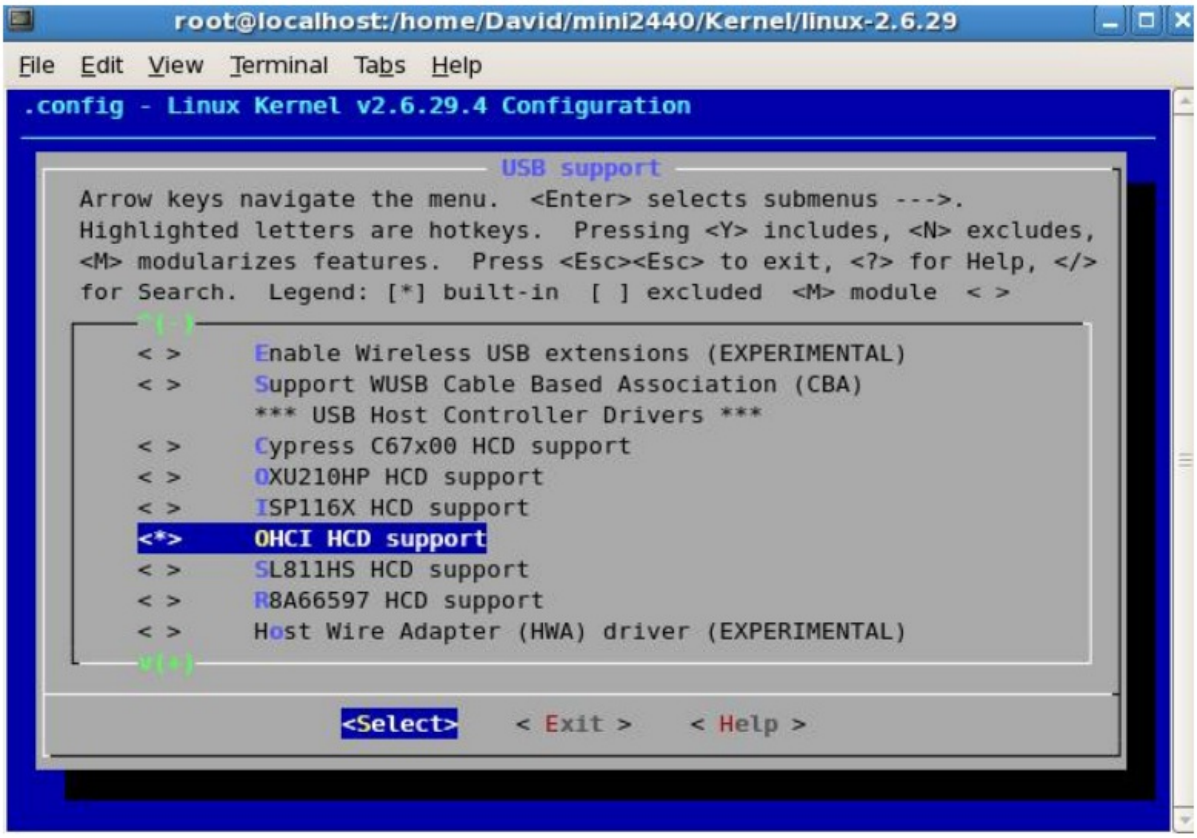




3.2、USB-MassStorage驱动体验

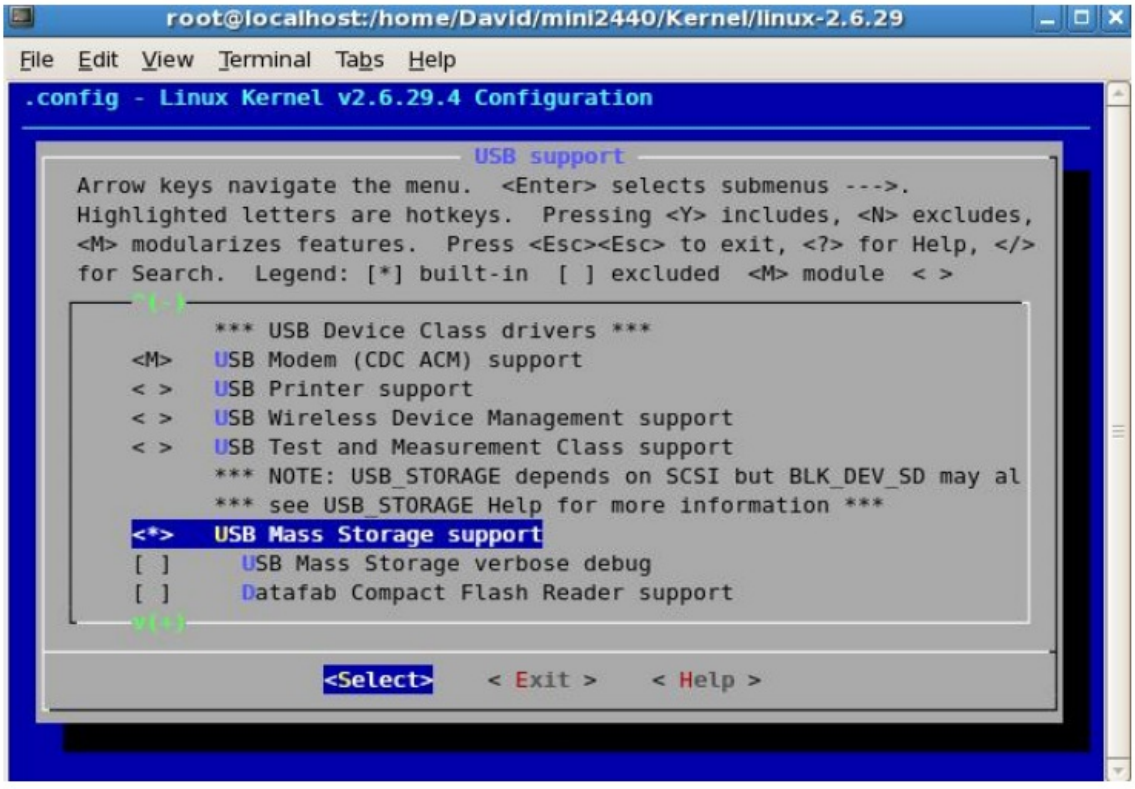
3.2.1、主控制器驱动

Device Drivers -> USB support ->support for host...



3.2.2、设备驱动

Device Drivers -> USB support ->support for host...

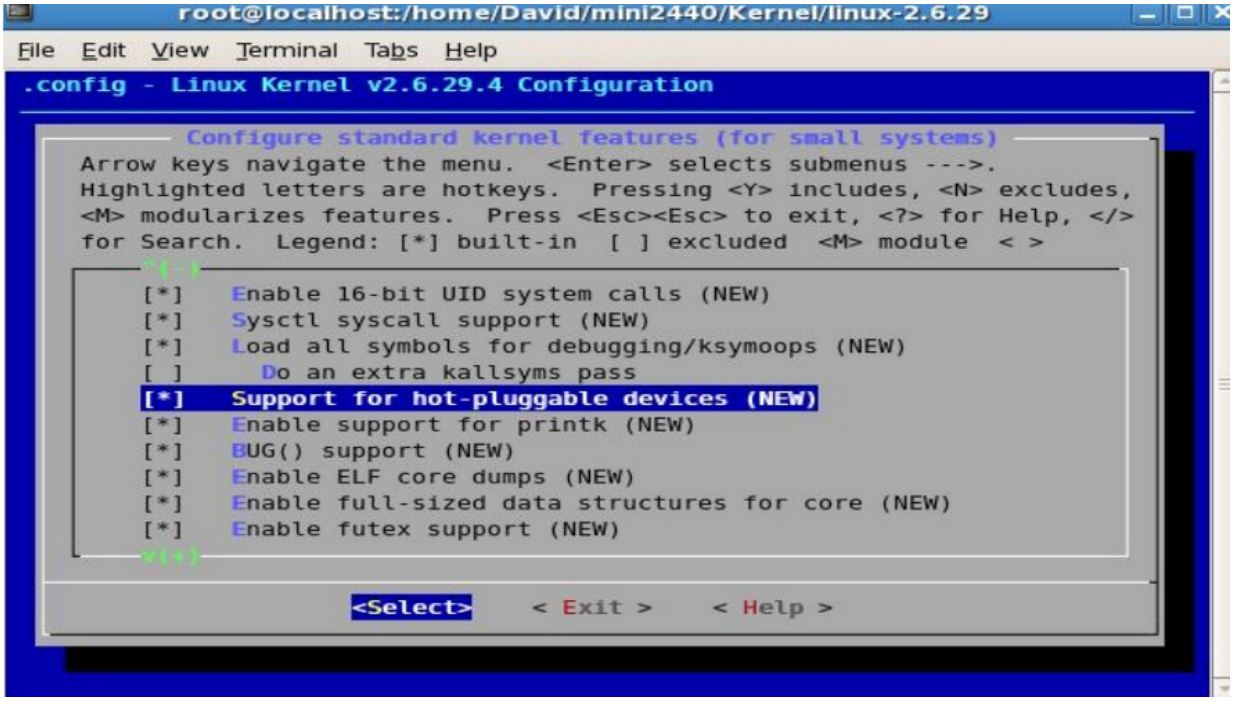


3.2.3、其他配置

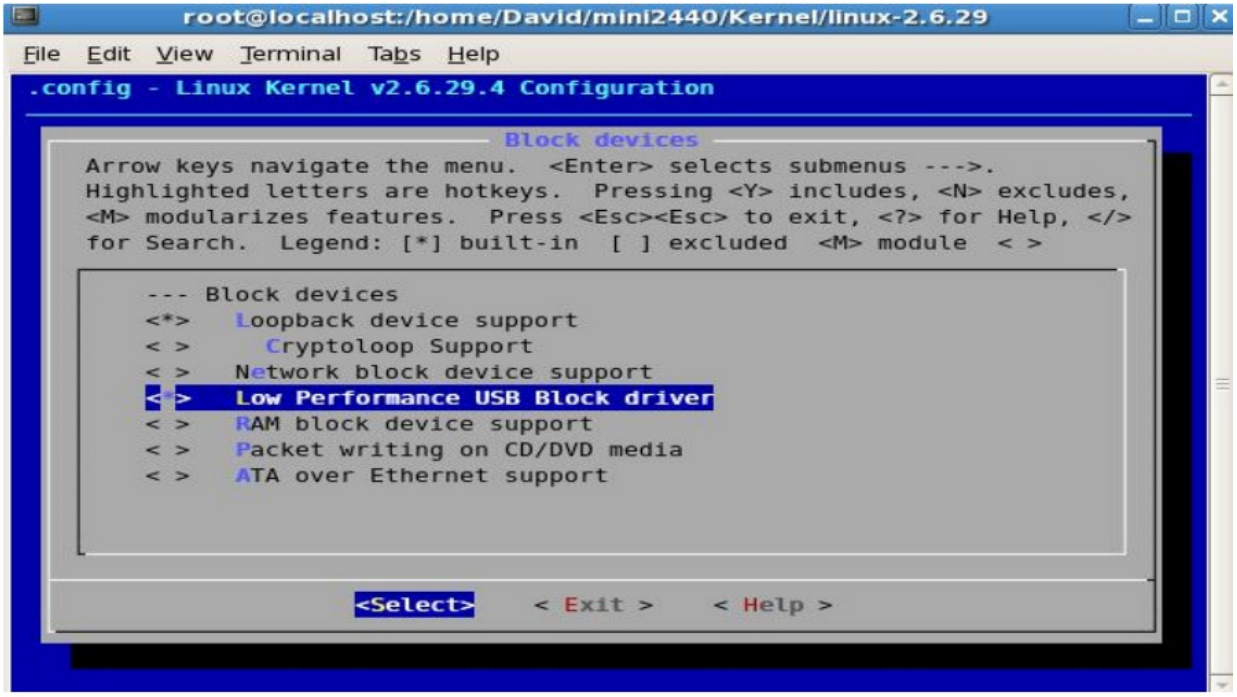
General setup --->

[*] Configure standard kernel features (for small systems) --->

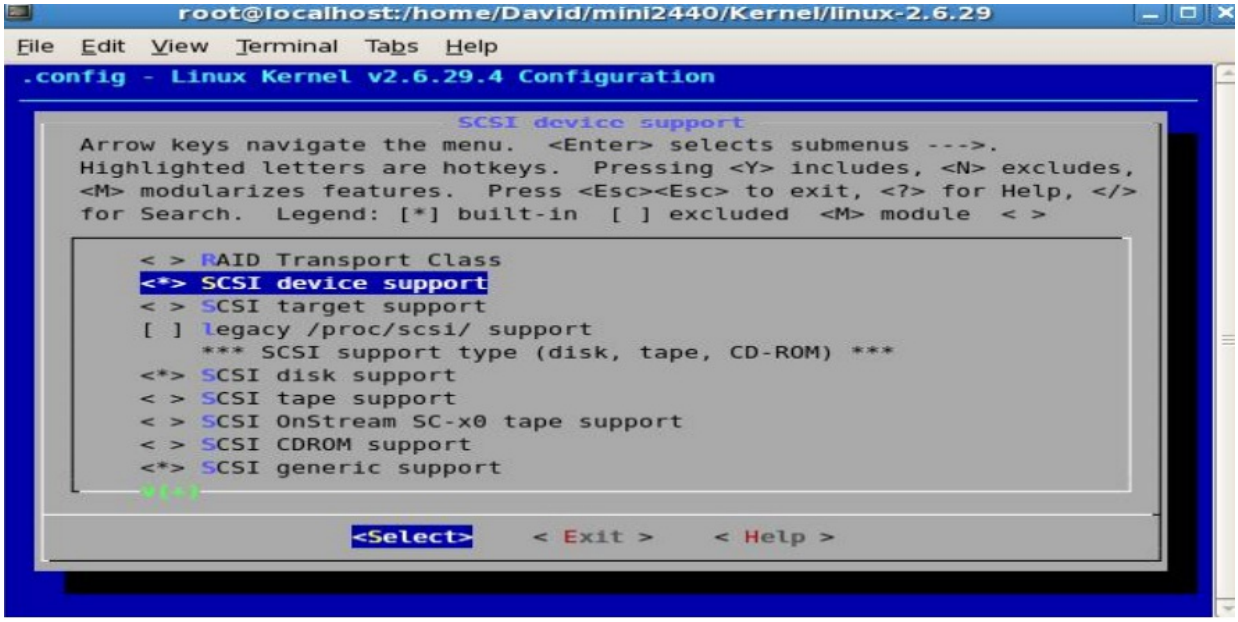
[*] Support for hot-pluggable devices (NEW)



Device Drivers --->
Block devices --->
<*> Low Performance USB Block driver



Device Drivers --->
SCSI device support --->
<*> SCSI device support
<*> SCSI disk support
<*> SCSI generic support

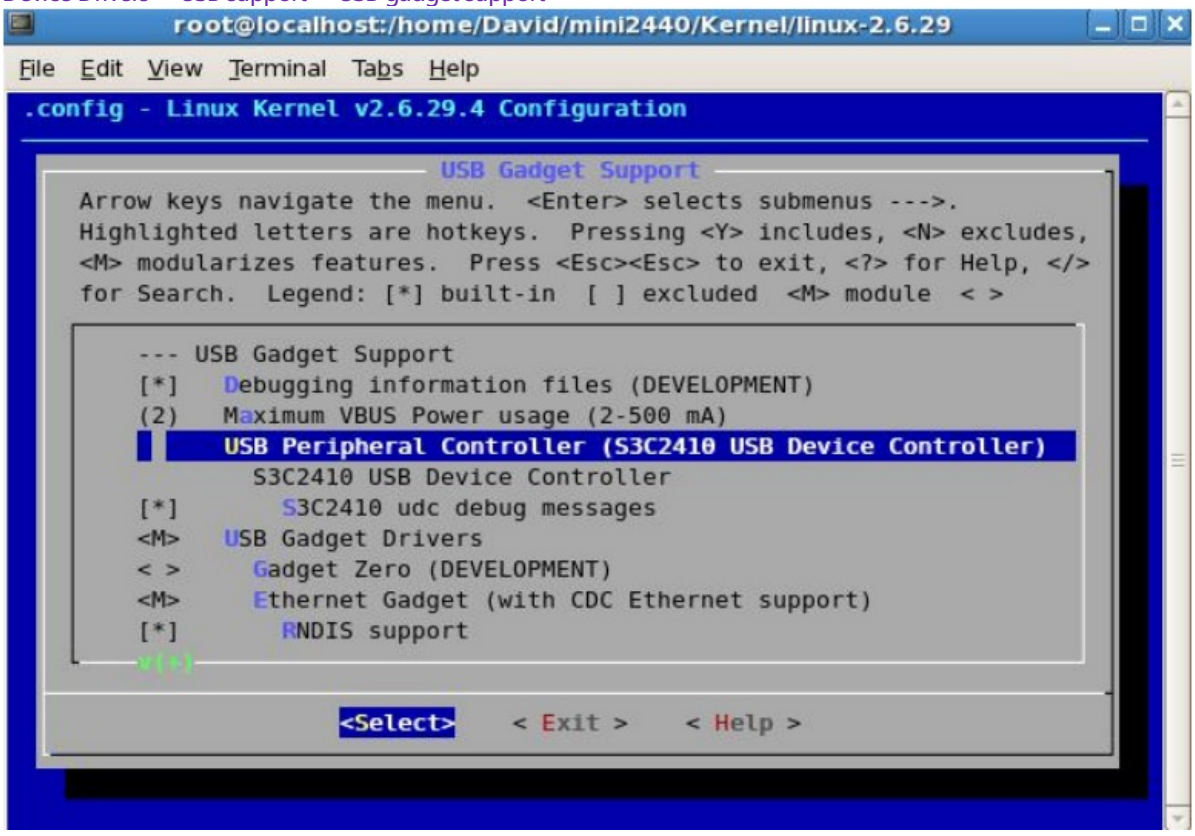


File systems --->
DOS/FAT/NT Filesystems --->
<*> MSDOS fs support
<*> VFAT (Windows-95) fs support
(936) Default codepage for FAT
(cp936) Default iocharset for FAT
Partition Types --->
[*] PC BIOS (MSDOS partition tables) support
Native Language Support --->
<*> Simplified Chinese charset (CP936, GB2312)
<*> NLS UTF8

3.3、USB-RANDIS驱动体验

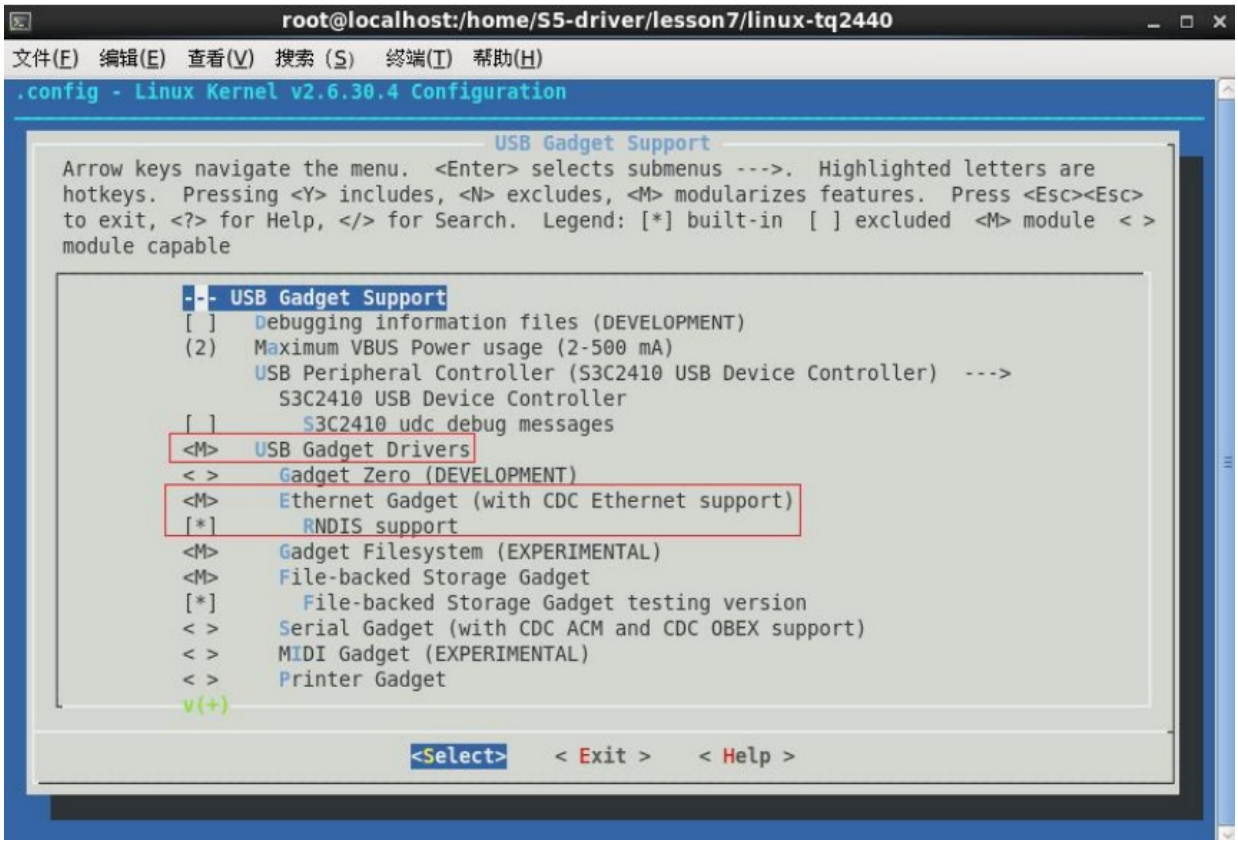
3.3.1、控制器驱动

Device Drivers ->USB support ->USB gadget support



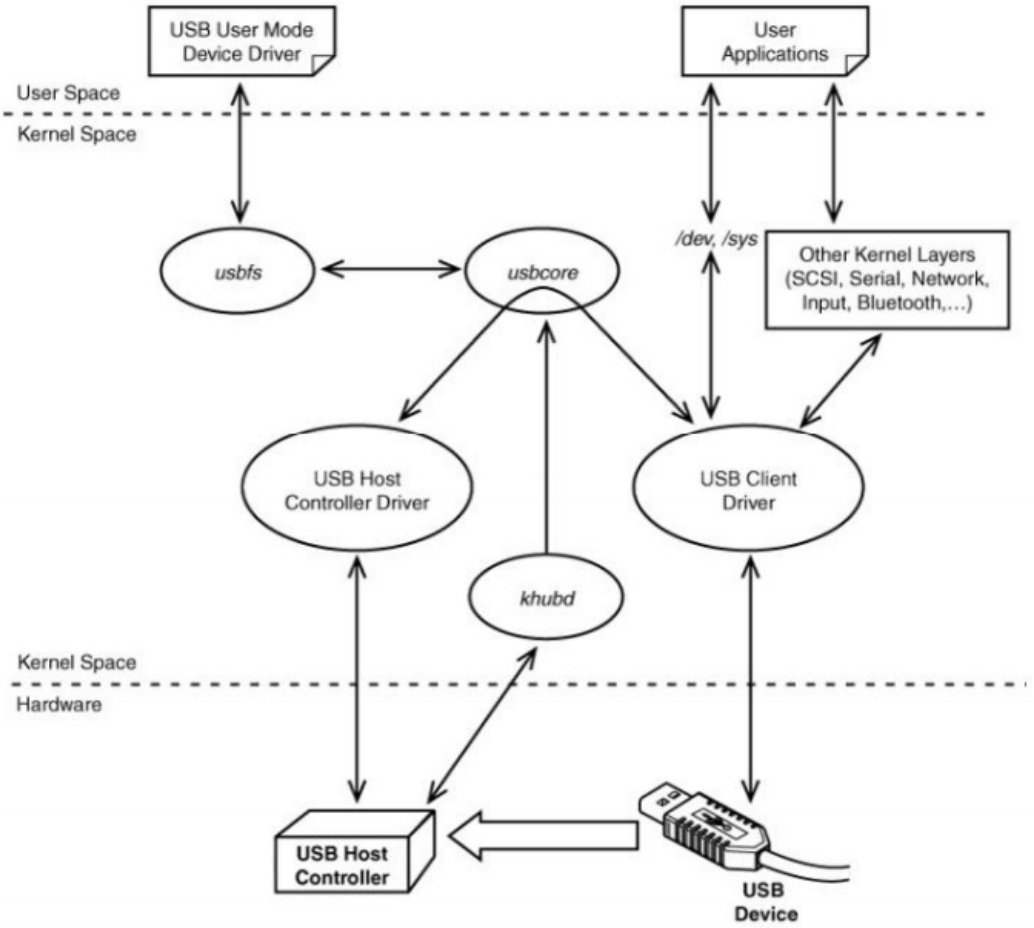
3.3.2、Gadget驱动

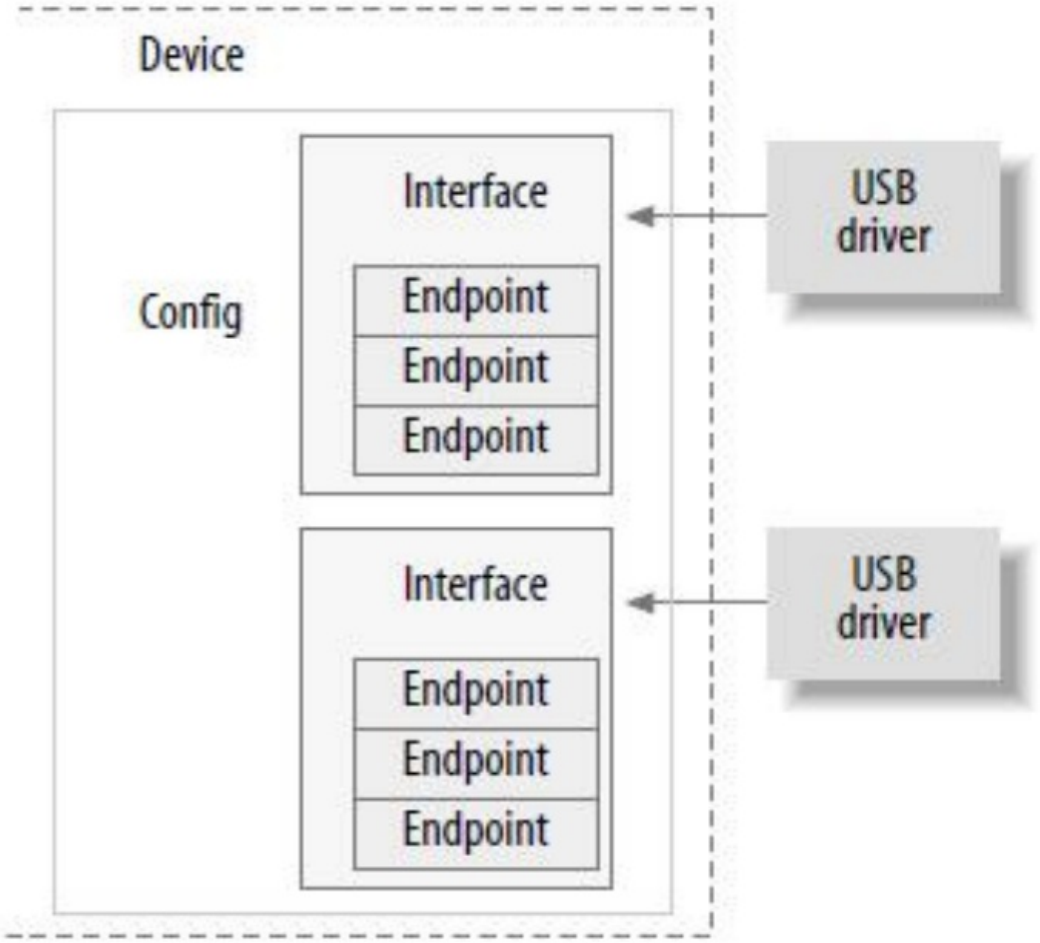
Device Drivers ->USB support ->USB gadget support



四、Linux USB设备驱动设计

4.1、USB驱动模型





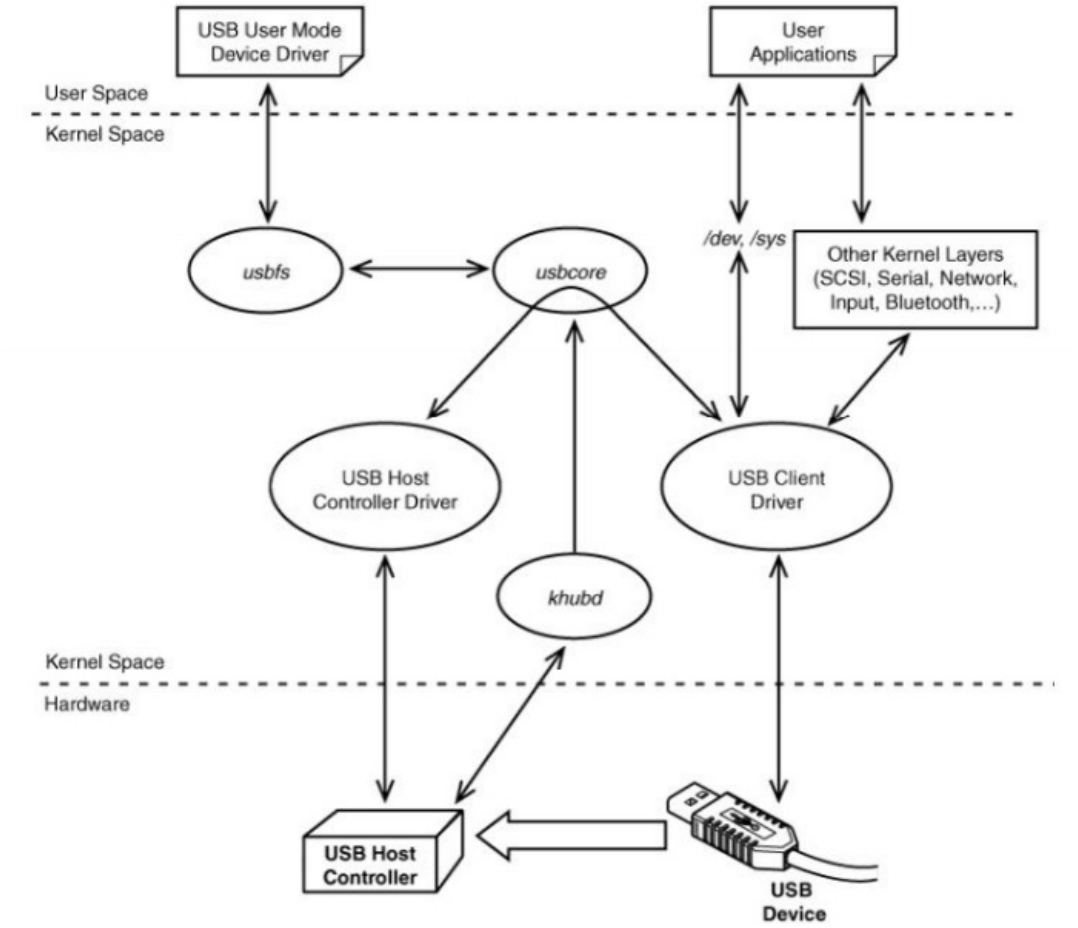
USB设备包括配置(configuration)、接口（interface）和端点(endpoint)，一个USB 设备驱动程序对应一个USB接口，而非整个USB设备。

在Linux内核中，使用struct usb_driver结构描述一个USB驱动。

```
struct usb_driver {
    const char *name; /*驱动程序名*/
    /* 当USB核心发现了该驱动能够处理的USB接口时，调用该函数 */
    int (*probe) (struct usb_interface *intf, const struct usb_device_id *id);
    /* 当相应的USB接口被移除时，调用该函数 */
    void (*disconnect) (struct usb_interface *intf);
    /* USB驱动能够处理的设备列表 */
    const struct usb_device_id *id_table;
}
```

4.2、URB

4.2.1、URB通讯模型



USB请求块（USB request block-URB）是 USB设备驱动中用来与USB设备通信所用的 基本载体和核心数据结构，非常类似于网络 设备驱动中的sk_buff结构体，是USB主 机与设备通信的“电波”。

1. USB 设备驱动程序创建并初始化一个访问特定端 点的urb，并提交给USB core；
2. USB core提交该urb到USB主控制器驱动程序；
3. USB 主控制器驱动程序根据该urb描述的信息，来 访问USB设备；
4. 当设备访问结束后，USB 主控制器驱动程序通知 USB 设备驱动程序。

4.2.2、创建URB

struct urb *usb_alloc_urb(int iso_packets, gfp_t mem_flags)

参数：

iso_packets：urb所包含的等时数据包的个数。

mem_flags：内存分配标识(如GFP_KERNEL)，参考kmalloc。

4.2.3、初始化URB

对于中断urb，使用usb_fill_int_urb函数来初始化

对于批量urb，使用usb_fill_bulk_urb函数来初始化

对于控制urb，使用usb_fill_control_urb函数来初始化

对于等时urb，只能手动地初始化urb。

```
static inline void usb_fill_int_urb(
    struct urb *urb, //待初始化的urb
    struct usb_device *dev, //urb所要访问的设备
    unsigned int pipe, //要访问的端点所对应的管道，
    void *transfer_buffer, //保存传输数据的buffer
    int buffer_length, //buffer长度
    usb_complete_t complete_fn, //urb完成时调用的函数
    void *context, //赋值到urb->context的数据
    int interval) //urb被调度的时间间隔
```

4.2.4、提交URB

在完成urb的创建和初始化后，USB驱动需要将urb提交 给USB核心.

int usb_submit_urb(struct urb *urb, gfp_t mem_flags)

参数：

urb：要提交urb的指针

mem_flags: 内存分配标识(如GFP_KERNEL)，参考kmalloc

URB被提交到USB核心后，USB核心指定usb主控制器驱动 程序来处理该urb，处理完之后，urb完成函数将被调用。

4.3、HID协议

HID(Human Interface Device)，属于人机交互类的设备，如USB鼠标，USB键盘，USB游戏操纵杆等。该类设备必须遵循HID设计规范。

、

4.4、鼠标驱动分析

五、USB下载线驱动设计