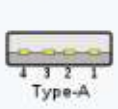
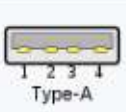
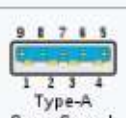
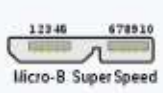
**基于stm32讲USB(描述符)**

# 一、什么是USB

USB接口是我们日常生活中最常见到的一种接口了，在电脑，手机，键盘，鼠标上都会见到。现常用的是 USB2.0 和 USB3.0 规格的。

| 類別 | | Type-A | Type-B | Type-C | Mini-A | Mini-B | Mini-AB | Micro-A | Micro-B | Micro-AB |
|---|---|---|---|---|---|---|---|---|---|---|
| 插頭（公頭） | 2.0 |  Type-A |  Type-B | 不存在 |  Mini-A |  Mini-B | 不存在 |  Micro-A |  Micro-B | 不存在 |
| | 3.2 |  Type-A SuperSpeed |  Type-B SuperSpeed |  | 已棄用 | 已棄用 | | 已棄用 |  Micro-B SuperSpeed | |
| 插座（母頭） | 2.0 |  Type-A |  Type-B | 不存在 |  Mini-A |  Mini-B |  Mini-AB | 不存在 |  Micro-B |  Micro-AB |
| | 3.2 |  Type-A SuperSpeed |  Type-B SuperSpeed |  Type-C | 已棄用 | 已棄用 | 已棄用 | |  Micro-B SuperSpeed | 已棄用 |
| 應用範圍 | | 電腦 | 掃描器、印表機等 | 新式電腦、行動電話、平板電腦等 | | 舊式可攜式裝置 | 僅作為萬能接頭 | | 行動電話、平板電腦等 | 僅作為萬能接頭 |



VCC(一般+5V)、GND、D+、D-。而D+、D-是两个数据线，学过模电的我们都知道差分电路的好处是可以抑制共模信号也就是抑制干扰，以保证信号传输的质量。而USB就是采用差分传输模式，所以 D+ 、 D- 也叫 USB-DP 、 USB-DM 。

接入主机的设备又有分为 低速 、 全速 和 高速 ，可用来控制功耗等，那么主机怎么去判断接入的是什么设备呢。

# 二、USB 接口的识别

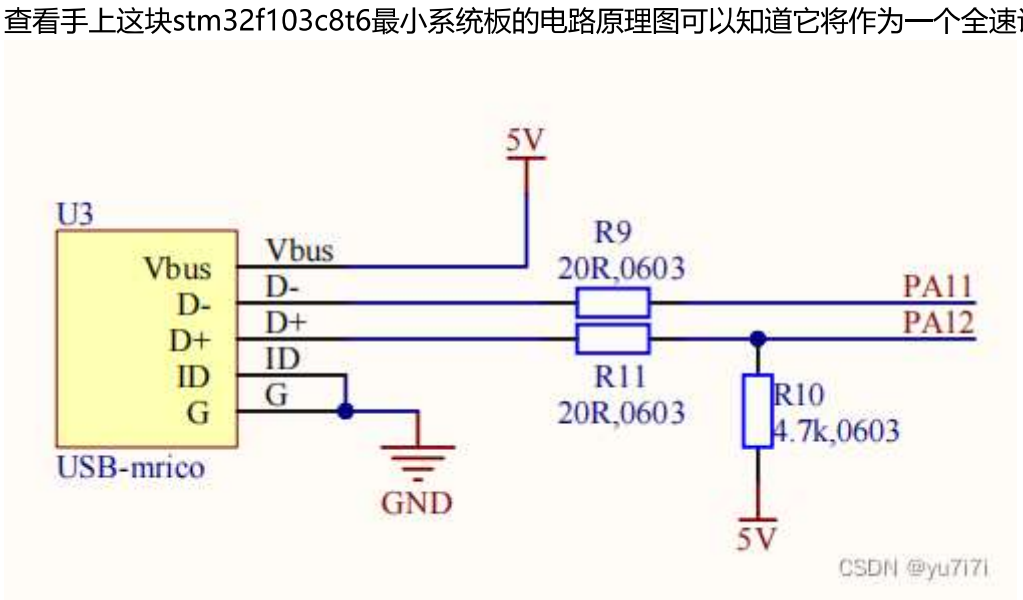
主机端 的USB接口的D+和D-都接有15K 下拉电阻 。

- 全速 USB设备的数据线D+接有1.5K的 上拉 电阻，一旦接入主机，主机的D+被拉高。
- 低速 USB设备的数据线D-接有1.5K的 上拉 电阻，一旦接入主机，主机的D-会被拉高。

因此，主机就可以根据检测到自己的D+为高还是D-为高，从而判断接入的设备是一个全速还是低速设备。

全速（Full Speed）和低速（Low Speed）很好区分。而 高速 设备初始是以一个全速设备的身份出现的，即和全速设备一样，D+线上有一个1.5k的上拉电阻。USB2.0的hub把它当作一个全速设备，之后，hub和设备通过一系列握手信号确认双方的身份。在这里对速度的检测是双向的，比如高速的hub需要检测所挂上来的设备是高速、全速还是低速， 高速 的设备需要检测所连上的hub是USB2.0的还是1.x的，如果是前者，就进行一系列动作切到高速模式工作，如果是后者，就以全速模式工作。(整个识别的过程有些复杂非本文重点，有兴趣可以自查资料不再赘述)

查看手上这块stm32f103c8t6最小系统板的电路原理图可以知道它将作为一个全速设备去连接主机。




# 三、USB设备类型定义

| Base Class | Descriptor Usage | Description |
| --- | --- | --- |
| 00h | Device | Use class information in the Interface Descriptors |
| 01h | Interface | Audio USB音频，参见UAC规范 |
| 02h | Both | Communications and CDC Control 网卡、调制解调器、串列端口 |
| 03h | Interface | HID (Human Interface Device)，键盘、鼠标 参见HID规范 |
| 05h | Interface | Physical 控制杆 |
| 06h | Interface | Image 影像扫描仪、Picture Transfer Protocol |
| 07h | Interface | Printer 打印机 |
| 08h | Interface | Mass Storage U盘、移动硬盘、存储卡读卡器、数字相机 |
| 09h | Device | Hub 集线器 |
| 0Ah | Interface | CDC-Data 调制解调器、网络卡、ISDN、传真 |
| 0Bh | Interface | Smart Card 读卡器 |
| 0Dh | Interface | Content Security |
| 0Eh | Interface | Video USB视频，UVC摄像头，参见UVC规范 |
| 0Fh | Interface | Personal Healthcare |
| 10h | Interface | Audio/Video Devices |
| 11h | Device | Billboard Device Class |
| 12h | Interface | USB Type-C Bridge Class |
| DCh | Both | Diagnostic Device |
| E0h | Interface | Wireless Controller 蓝牙 |
| EFh | Both | Miscellaneous |
| FEh | Interface | Application Specific 红外线资料桥接器 |
| FFh | Both | Vendor Specific 自定义USB设备 |



依附在总线上的设备可以是需要特定的驱动程序的完全定制的设备，也可能属于 某个设备类别 。这些类别定义设备的 行为 和 接口描述符 ，这样一个驱动程序可能用于所有此种类别的设备。一般操作系统都支持这些设备类别，为其提供 通用驱动程序 。

```
0x09,          /*bLength: Interface Descriptor size*/                  //设备描述符的字节数大小，为0x09
USB_INTERFACE_DESCRIPTOR_TYPE,/*bDescriptorType: Interface descriptor type*/ //描述符类型编号，为0x04
0x00,          /*bInterfaceNumber: Number of Interface*/               //接口的编号
0x00,          /*bAlternateSetting: Alternate setting*/                //备用的接口描述符编号
0x01,          /*bNumEndpoints*/                                       //该接口使用端点数，不包括端点0
0x03,          /*bInterfaceClass: HID*/                                //接口类型
0x01,          /*bInterfaceSubClass : 1=BOOT, 0=no boot*/              //接口子类型
0x02,          /*nInterfaceProtocol : 0=none, 1=keyboard, 2=mouse*/    //接口所遵循的协议
0,             /*iInterface: Index of string descriptor*/              //描述该接口的字符串索引值
/********************* Descriptor of Joystick Mouse HID ********************/ //操纵杆鼠标HID描述符
/* 18 */
```



上面是**鼠标**的一个设备分类，由上面表格可知鼠标属于 HID 设备，对应 0x03h 。

```
/********************* Descriptor of Mass Storage interface ********************/
/* 09 */
0x09,   /* bLength: Interface Descriptor size */
0x04,   /* bDescriptorType: */
/*      Interface descriptor type */
0x00,   /* bInterfaceNumber: Number of Interface */
0x00,   /* bAlternateSetting: Alternate setting */
0x02,   /* bNumEndpoints*/
0x08,   /* bInterfaceClass: MASS STORAGE Class */   //接口类:海量存储类
0x06,   /* bInterfaceSubClass : SCSI transparent*/
0x50,   /* nInterfaceProtocol */
4,          /* iInterface: */
/* 18 */
```



上面是**U盘**的一个设备分类，由上面表格可知鼠标属于 Mass Storage 设备，对应 0x08h 。

那么为什么要分设备类型呢？区分设备类型一个重要原因就是因为不同设备需要的 数据传输模式 不同。

> USB支持**四种**基本的**数据传输模式**： 控制传输 ， 等时传输 ， 中断传输 及数据块传输。每种传输模式应用到具有相同名字的终端，则具有不同的性质。

> 控制传输类型 ：
>
> 支持外设与主机之间的控制，状态，配置等信息的传输，为外设与主机之间提供一个控制通道。每种外设都支持控制传输类型，这样主机与外设之间就可以传送配置和命令/状态信息。
>
> 等时（Isochronous）传输类型（或称同步传输） ：
>
> 支持有周期性，有限的时延和带宽且数据传输速率不变的外设与主机间的数据传输。该类型无差错校验，故不能保证正确的数据传输，支持像计算机 - 电话集成系统（CTI）和音频系统与主机的数据传输。
>
> 中断传输类型 ：
>
> 支持像游戏手柄，鼠标和键盘等输入设备，这些设备与主机间数据传输量小，无周期性，但对响应时间敏感，要求马上响应。
>
> 数据块（Bulk）传输类型 ：
>
> 支持打印机，扫描仪，数码相机等外设，这些外设与主机间传输的数据量大，USB在满足带宽的情况下才进行该类型的数据传输。USB采用分块带宽分配方案，若外设超过当前带宽分配或潜在的要求，则不能进入该设备。同步和中断传输类型的终端保留带宽，并保证数据按一定的速率传送。集中和控制终端按可用的**最佳**带宽来传输传输数据。

# 四、描述符详解

> USB是通过USB描述符来对USB设备进行属性的说明，包括使用的协议、接口数目、端点和传输方式等等。当USB设备插入主机后，主机要对其进行总线枚举，配置该设备所需的驱动等信息。主机通过标准请求Get Descriptor来读取USB的描述符，从而得到设备的相关信息，根据这些信息，然后建立通信。因此说，只有正确设置USB的描述符，才能使USB设备正常工作起来。

标准的USB设备有5种USB描述符： 设备描述符 ， 配置描述符 ， 字符串描述符 、 接口描述符 ， 端点描述符 。下面详解：

## 1、设备描述符

一个设备只有一个设备描述符。

```
typedef struct _USB_DEVICE_DESCRIPTOR_
{
    BYTE        bLength,            描述符大小．固定为0x12．
    BYTE        bDescriptorType,    设备描述符类型固定为0x01．
    WORD        bcdUSB,             规范发布号．表示了本设备能适用于哪种协议，如2.0=0200，1.1=0110等．
    BYTE        bDeviceClass,       类型代码（由USB指定）
    BTYE        bDeviceSubClass,    子类型代码（由USB分配）
    BYTE        bDeviceProtol,      协议代码（由USB分配）
    BYTE        bMaxPacketSize0,    端点0最大分组大小
    WORD        idVenderI,          供应商ID（由USB分配）
    WORD        idProduct,          产品ID（由厂商分配）
    WORD        bcdDevice,          设备出产编码．
    BYTE        iManufacturer,      厂商描述符字符串索引
    BYTE        iProduct,           产品描述符字符串索引．
    BYTE        iSerialNumber,      设备序列号字符串索引
    BYTE        iNumConfiguations   可能的配置数．指配置字符串的个数
}USB_DEVICE_DESCRIPTOR;
```

以下是摘取的STM32的鼠标(HID)设备里的实例代码里的一个 设备描述符 ：

```
/* USB Standard Device Descriptor USB标准设备描述符 */
const uint8_t Joystick_DeviceDescriptor[JOYSTICK_SIZ_DEVICE_DESC] =
  {
    0x12,                       /*bLength 设备描述符的字节数大小*/
    USB_DEVICE_DESCRIPTOR_TYPE, /*bDescriptorType 设备描述符类型编号 /端点描述符类型编号*/
    0x00,                       /*bcdUSB */
    0x02,
    0x00,                       /*bDeviceClass*/
    0x00,                       /*bDeviceSubClass*/
    0x00,                       /*bDeviceProtocol*/
    0x40,                       /*bMaxPacketSize 64*/
    0x83,                       /*idVendor (0x0483)*/
    0x04,
    0x10,                       /*idProduct = 0x5710*/
    0x57,
    0x00,                       /*bcdDevice rel. 2.00*/
    0x02,
    1,                          /*Index of string descriptor describing
                                                manufacturer */
    2,                          /*Index of string descriptor describing
                                               product*/
    3,                          /*Index of string descriptor describing the
                                               device serial number */
    0x01                        /*bNumConfigurations*/
  }
  ; /* Joystick_DeviceDescriptorca 操纵杆设备描述符 */
```



## 2、配置描述符

配置描述符定义了设备的配置信息，一个设备可以有**多个** 配置描述符 。

```
typedef struct _USB_CONFIGURATION_DESCRIPTOR_
{
    BYTE      bLength,              描述符大小．固定为0x09．
    BYTE      bDescriptorType,      配置描述符类型．固定为0x02．
    WORD      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
    BYTE      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．
    BYTE      bConfigurationValue,  作为Set Configuration的一个参数选择配置值．
    BYTE      iConfiguration,       用于描述该配置字符串描述符的索引．
    BYTE      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．
    BYTE      MaxPower              总线供电的USB设备的最大消耗电流．以2mA为单位．
}USB_CONFIGURATION_DESCRIPTOR;
```

以下是摘取的STM32的鼠标(HID)设备里的实例代码里的一个 配置描述符 ：

```
0x09, /* bLength: Configuration Descriptor size */                    //设备描述符的字节数大小，为0x09
USB_CONFIGURATION_DESCRIPTOR_TYPE, /* bDescriptorType: Configuration */ //描述符类型编号，为0x02
JOYSTICK_SIZ_CONFIG_DESC,
/* wTotalLength: Bytes returned */
0x00,                                                                  //接口的编号
0x01,         /*bNumInterfaces: 1 interface*/
0x01,         /*bConfigurationValue: Configuration value*/
0x00,         /*iConfiguration: Index of string descriptor describing     bConfigurationValue:字符串描述符的索引配置
                the configuration*/
0xE0,         /*bmAttributes: Self powered */                          //bm属性:自供电
0x32,         /*MaxPower 100 mA: this current is used for detecting Vbus*/ //最大功率100ma:此电流用于检测Vbus
```



## 3、接口描述符

一个配置支持至少一个接口。接口只要定义了实现功能的硬件的集合。每一个能够与USB实现数据交换的硬件叫做端点。也就是说接口是端点的集合。接口描述符说明了接口所提供的配置，一个配置所拥有的接口数量通过配置描述符的 bNumInterfaces 决定。

```
typedef struct _USB_INTERFACE_DESCRIPTOR_
{
    BYTE      bLength,              描述符大小．固定为0x09。
    BYTE      bDescriptorType,      接口描述符类型．固定为0x04 。
    BYTE      bInterfaceNumber,     该接口的编号。
    BYTE      bAlternateSetting,    用于为上一个字段选择可供替换的位置。即备用的接口描述符标号。
    BYTE      bNumEndpoint,         使用的端点数目．端点０除外。
    BYTE      bInterfaceClass,      类型代码（由USB分配）。
    BYTE      bInterfaceSubClass,   子类型代码（由USB分配）。
    BYTE      bInterfaceProtocol,   协议代码（由USB分配）。
    BYTE      iInterface            字符串描述符的索引。
}USB_INTERFACE_DESCRIPTOR;
```

以下是摘取的STM32的鼠标(HID)设备里的实例代码里的一个 接口描述符 ：

```
/************** Descriptor of Joystick Mouse interface ****************/    //操纵杆鼠标接口描述符
/* 09 */
0x09,         /*bLength: Interface Descriptor size*/                       //设备描述符的字节数大小，为0x09
USB_INTERFACE_DESCRIPTOR_TYPE,/*bDescriptorType: Interface descriptor type*/ //描述符类型编号，为0x04
0x00,         /*bInterfaceNumber: Number of Interface*/                    //接口的编号
0x00,         /*bAlternateSetting: Alternate setting*/                     //备用的接口描述符编号
0x01,         /*bNumEndpoints*/                                            //该接口使用端点数，不包括端点0
0x03,         /*bInterfaceClass: HID*/                                     //接口类型
0x01,         /*bInterfaceSubClass : 1=BOOT, 0=no boot*/                   //接口子类型
0x02,         /*nInterfaceProtocol : 0=none, 1=keyboard, 2=mouse*/         //接口所遵循的协议
0,            /*iInterface: Index of string descriptor*/                   //描述该接口的字符串索引值
```



## 4、端点描述符

每一个能够与USB实现数据交换的硬件叫做端点。USB设备中的每个端点都有自己的端点描述符，由接口描述符中的 bNumEndpoint 决定其数量。

```
typedef struct _USB_ENDPOINT_DESCRIPTOR_
{
    BYTE        bLength,              描述符大小．固定为0x07．
    BYTE        bDescriptorType,      接口描述符类型．固定为0x05．
    BYTE        bEndpointAddress,     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。
    BYTE        bmAttributes,         端点属性．Bit7-2，保留．BIt1-0: 00控制，01同步，02批量，03中断。
    WORD        wMaxPacketSize,       本端点接收或发送的最大信息包大小。
    BYTE        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．
}USB_ENDPOINT_DESCRIPTOR
```

以下是摘取的STM32的鼠标(HID)设备里的实例代码里的一个 端点描述符 ：

```
/********************* Descriptor of Joystick Mouse endpoint *********************///操纵杆鼠标端点的描述符
/* 27 */
0x07,          /*bLength: Endpoint Descriptor size*/
USB_ENDPOINT_DESCRIPTOR_TYPE, /*bDescriptorType:*/

0x81,          /*bEndpointAddress: Endpoint Address (IN)*/
0x03,          /*bmAttributes: Interrupt endpoint*/
0x04,          /*wMaxPacketSize: 4 Byte max */
0x00,
0x20,          /*bInterval: Polling Interval (32 ms)*/
/* 34 */
```



## 5、字符串描述符

其中字符串描述符是可选的．如果不支持字符串描述符，其设备，配置，接口描述符内的所有字符串描述符索引都必须为 0 。

```
typedef struct _USB_STRING_DESCRIPTION_
{
    BYTE      bLength,              描述符大小．由整个字符串的长度加上bLength和bDescriptorType的长度决定。
    BYTE      bDescriptionType,     接口描述符类型．固定为0x03。
    BYTE      bString[1];           Unicode编码字符串．
}USB_STRING_DESCRIPTION;
```

以下是摘取的STM32的鼠标(HID)设备里的实例代码里的一个 字符串描述符 ：

```
/* USB String Descriptors (optional) USB字符串描述符(可选)*/
const uint8_t Joystick_StringLangID[JOYSTICK_SIZ_STRING_LANGID] =
  {
    JOYSTICK_SIZ_STRING_LANGID,
    USB_STRING_DESCRIPTOR_TYPE,
    0x09,
    0x04
  }
  ; /* LangID = 0x0409: U.S. English */
```