[USB 2.0总线协议 2](#_Toc477445957)

[介绍 2](#_Toc477445958)

[名词解释 3](#_Toc477445959)

[功能 9](#_Toc477445960)

[应用分类 9](#_Toc477445961)

[USB体系 10](#_Toc477445962)

[物理层 11](#_Toc477445963)

[物理连接 11](#_Toc477445964)

[总线结构（总线拓扑） 12](#_Toc477445965)

[USB主机 13](#_Toc477445966)

[协议层 14](#_Toc477445967)

[USB传输数据结构 14](#_Toc477445968)

[概述 14](#_Toc477445969)

[数据包 15](#_Toc477445970)

[数据帧 18](#_Toc477445971)

[错误检测与处理 19](#_Toc477445972)

[CRC计算 19](#_Toc477445973)

[USB传输数据过程 20](#_Toc477445974)

[批量事务传输 20](#_Toc477445975)

[控制传输 21](#_Toc477445976)

[中断传输 22](#_Toc477445977)

[同步传输 23](#_Toc477445978)

[USB OTG总线协议 24](#_Toc477445979)

[介绍 24](#_Toc477445980)

[主机协商协议 25](#_Toc477445981)

[对话请求协议（SRP） 26](#_Toc477445982)

[OTG功能构建 26](#_Toc477445983)

USB 2.0总线协议

# 介绍

Universal Serial Bus

## 名词解释

|  |  |
| --- | --- |
| **Byte** | 字节（8位）. |
| **b/s** | 数据传输速率—位/秒 |
| **B/s** | 数据传输速率—字节/秒 |
| ACK | 应答信号 |
| Active Device | 正常运作的设备 |
| Asynchronous Data | 异步数据 |
| **Asynchronous RA** | The incoming data rate, Fsi, and the outgoing data rate, Fso, of the RA processare independent (i.e., there is no shared master clock). See also rateadaptation. |
| **Asynchronous SRC** | The incoming sample rate, Fsi, and outgoing sample rate, Fso, of the SRCprocess are independent (i.e., there is no shared master clock). See also samplerate conversion. |
| **AWG#** | 电线很界面测量标准 |
| **Babble** | Unexpected bus activity that persists beyond a specified point in a(micro)frame. |
| **Bandwidth** | 带宽：单位时间单元传输的数据量 单位：b/s（bit每秒） B/s（字节/每秒） |
| **Big Endian** | 存储数据时，将数据的最低有效字节存储在最高位的地址，最高有效字节放在最低位的地址 |
| **Bit** | 位（1或0） |
| **Phase Locked Loop**  **(PLL)** | 锁相环电路  保持振荡器的频率和输入信号的频率相同 |
| **Bit Stuffing** | **位填充**  Insertion of a “0” bit into a data stream to cause an electrical transition on the  data wires, allowing a PLL to remain locked. |
| **Buffer** | 缓存  设备时间传输数据时，补偿 数据传输速率不同 或 事件发生的时间不同 |
| **Transfer Type** | Determines the characteristics of the data flow between a software client and its function. Four standard transfer types are defined: control, interrupt, bulk, and isochronous. |
| **Bulk Transfer** | **USB转换模式之一**  **非周期**  **用于任意带宽**  One of the four USB transfer types. Bulk transfers are non-periodic, large bursty communication typically used for a transfer that can use any available bandwidth and can also be delayed until bandwidth is available. See also transfer type. |
| **Control Transfer** | **USB转换模式之一**  One of the four USB transfer types. Control transfers support  configuration/command/status type communications between client and  function. See also transfer type. |
| **Interrupt Transfer** | **USB转换模式之一**  One of the four USB transfer types. Interrupt transfer characteristics are small  data, non-periodic, low-frequency, and bounded-latency. Interrupt transfers  are typically used to handle service needs. See also transfer type. |
| Isochronous **Transfer** | **USB转换模式之一**  **用于处理同步数据**  One of the four USB transfer types. Isochronous transfers are used when  working with isochronous data. Isochronous transfers provide periodic,  continuous communication between host and device. See also transfer type. |
| **Bus Enumeration** | Detecting and identifying USB devices. |
| **Capabilities** | 用户定义的USB设备的属性 |
| **Characteristics** | **USB设备的固定属性**  Those qualities of a USB device that are unchangeable; for example, the device  class is a device characteristic. |
| **Client** | Software resident on the host that interacts with the USB System Software to  arrange data transfer between a function and the host. The client is often the  data provider and consumer for transferred data. |
| **Configuring Software** | Software resident on the host software that is responsible for configuring a  USB device. This may be a system configurator or software specific to the  device. |
| **Control Endpoint** | A pair of device endpoints with the same endpoint number that are used by a  control pipe. Control endpoints transfer data in both directions and, therefore,  use both endpoint directions of a device address and endpoint number  combination. Thus, each control endpoint consumes two endpoint addresses. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Control Pipe** Same as a message pipe.

**CRC** See Cyclic Redundancy Check.

**CTI** Computer Telephony Integration.

**Cyclic Redundancy**

**Check (CRC)**

A check performed on data to see if an error has occurred in transmitting,

reading, or writing the data. The result of a CRC is typically stored or

transmitted with the checked data. The stored or transmitted result is

compared to a CRC calculated for the data to determine if an error has

occurred.

**Default Address** An address defined by the USB Specification and used by a USB device when

it is first powered or reset. The default address is 00H.

**Default Pipe** The message pipe created by the USB System Software to pass control and

status information between the host and a USB device’s endpoint zero.

**Device** A logical or physical entity that performs a function. The actual entity

described depends on the context of the reference. At the lowest level, device

may refer to a single hardware component, as in a memory device. At a higher

level, it may refer to a collection of hardware components that perform a

particular function, such as a USB interface device. At an even higher level,

device may refer to the function performed by an entity attached to the USB;

for example, a data/FAX modem device. Devices may be physical, electrical,

addressable, and logical.

When used as a non-specific reference, a USB device is either a hub or a

function.

**Device Address** A seven-bit value representing the address of a device on the USB. The device

address is the default address (00H) when the USB device is first powered or

the device is reset. Devices are assigned a unique device address by the USB

System Software.

**Device Endpoint** A uniquely addressable portion of a USB device that is the source or sink of

information in a communication flow between the host and device. See also

endpoint address.

**Device Resources** Resources provided by USB devices, such as buffer space and endpoints. See

also Host Resources and Universal Serial Bus Resources.

**Device Software** Software that is responsible for using a USB device. This software may or

may not also be responsible for configuring the device for use.

**Downstream** The direction of data flow from the host or away from the host. A downstream

port is the port on a hub electrically farthest from the host that generates

downstream data traffic from the hub. Downstream ports receive upstream

data traffic.

**Driver** When referring to hardware, an I/O pad that drives an external load. When

referring to software, a program responsible for interfacing to a hardware

device, that is, a device driver.

**DWORD** Double word. A data element that is two words (i.e., four bytes or 32 bits) in

size.

**Dynamic Insertion**

**and Removal**

The ability to attach and remove devices while the host is in operation.

**E2PROM** See Electrically Erasable Programmable Read Only Memory.

**EEPROM** See Electrically Erasable Programmable Read Only Memory.

**Electrically**

**Erasable**

**Programmable**

**Read Only Memory**

**(EEPROM)**

Non-volatile rewritable memory storage technology.

**End User** The user of a host.

**Endpoint** See device endpoint.

**Endpoint Address** The combination of an endpoint number and an endpoint direction on a USB

device. Each endpoint address supports data transfer in one direction.

**Endpoint Direction** The direction of data transfer on the USB. The direction can be either IN or

OUT. IN refers to transfers to the host; OUT refers to transfers from the host.

**Endpoint Number** A four-bit value between 0H and FH, inclusive, associated with an endpoint on

a USB device.

**Envelope detector** An electronic circuit inside a USB device that monitors the USB data lines and

detects certain voltage related signal characteristics.

**EOF** End-of-(micro)Frame.

**EOP** End-of-Packet.

**External Port** See port.

**Eye pattern** A representation of USB signaling that provides minimum and maximum

voltage levels as well as signal jitter.

**False EOP** A spurious, usually noise-induced event that is interpreted by a packet receiver

as an EOP.

**Frame** A 1 millisecond time base established on full-/low-speed buses.

**Frame Pattern** A sequence of frames that exhibit a repeating pattern in the number of samples

transmitted per frame. For a 44.1 kHz audio transfer, the frame pattern could

be nine frames containing 44 samples followed by one frame containing

45 samples.

**Fs** See sample rate.

**Full-duplex** Computer data transmission occurring in both directions simultaneously.

**Full-speed** USB operation at 12 Mb/s. See also low-speed and high-speed.

**Function** A USB device that provides a capability to the host, such as an ISDN

connection, a digital microphone, or speakers.

**Handshake Packet** A packet that acknowledges or rejects a specific condition. For examples, see

ACK and NAK.

**High-bandwidth**

**endpoint**

A high-speed device endpoint that transfers more than 1024 bytes and less than

3073 bytes per microframe.

**High-speed** USB operation at 480 Mb/s. See also low-speed and full-speed.

**Host** The host computer system where the USB Host Controller is installed. This

includes the host hardware platform (CPU, bus, etc.) and the operating system

in use.

**Host Controller** The host’s USB interface.

**Host Controller**

**Driver (HCD)**

The USB software layer that abstracts the Host Controller hardware. The Host

Controller Driver provides an SPI for interaction with a Host Controller. The

Host Controller Driver hides the specifics of the Host Controller hardware

implementation.

**Host Resources** Resources provided by the host, such as buffer space and interrupts. See also

Device Resources and Universal Serial Bus Resources.

**Hub** A USB device that provides additional connections to the USB.

**Hub Tier** One plus the number of USB links in a communication path between the host

and a function. See Figure 4-1.

**Interrupt Request**

**(IRQ)**

A hardware signal that allows a device to request attention from a host. The

host typically invokes an interrupt service routine to handle the condition that

caused the request.

**I/O Request Packet** An identifiable request by a software client to move data between itself (on the

host) and an endpoint of a device in an appropriate direction.

**IRP** See I/O Request Packet.

**IRQ** See Interrupt Request.

**Isochronous Data** A stream of data whose timing is implied by its delivery rate.

**Isochronous Device** An entity with isochronous endpoints, as defined in the USB Specification, that

sources or sinks sampled analog streams or synchronous data streams.

**Isochronous Sink**

**Endpoint**

An endpoint that is capable of consuming an isochronous data stream that is

sent by the host.

**Isochronous Source**

**Endpoint**

An endpoint that is capable of producing an isochronous data stream and

sending it to the host.

**Isochronous**

**Jitter** A tendency toward lack of synchronization caused by mechanical or electrical

changes. More specifically, the phase shift of digital pulses over a

transmission medium.

**kb/s** Transmission rate expressed in kilobits per second.

**kB/s** Transmission rate expressed in kilobytes per second.

**Little Endian** Method of storing data that places the least significant byte of multiple-byte

values at lower storage addresses. For example, a 16-bit integer stored in little

endian format places the least significant byte at the lower address and the

most significant byte at the next address. See also big endian.

**LOA** Loss of bus activity characterized by an SOP without a corresponding EOP.

**Low-speed** USB operation at 1.5 Mb/s. See also full-speed and high-speed.

**LSb** Least significant bit.

**LSB** Least significant byte.

**Mb/s** Transmission rate expressed in megabits per second.

**MB/s** Transmission rate expressed in megabytes per second.

**Message Pipe** A bi-directional pipe that transfers data using a request/data/status paradigm.

The data has an imposed structure that allows requests to be reliably identified

and communicated.

**Microframe** A 125 microsecond time base established on high-speed buses.

**MSb** Most significant bit.

**MSB** Most significant byte.

**NAK** Handshake packet indicating a negative acknowledgment.

**Non Return to Zero**

**Invert (NRZI)**

A method of encoding serial data in which ones and zeroes are represented by

opposite and alternating high and low voltages where there is no return to zero

(reference) voltage between encoded bits. Eliminates the need for clock

pulses.

**NRZI** See Non Return to Zero Invert.

**Object** Host software or data structure representing a USB entity.

**Packet** A bundle of data organized in a group for transmission. Packets typically

contain three elements: control information (e.g., source, destination, and

length), the data to be transferred, and error detection and correction bits.

**Packet Buffer** The logical buffer used by a USB device for sending or receiving a single

packet. This determines the maximum packet size the device can send or

receive.

**Packet ID (PID)** A field in a USB packet that indicates the type of packet, and by inference, the

format of the packet and the type of error detection applied to the packet.

**Phase** A token, data, or handshake packet. A transaction has three phases.

**Physical Device** A device that has a physical implementation; e.g., speakers, microphones, and

CD players.

**PID** See Packet ID.

**Pipe** A logical abstraction representing the association between an endpoint on a

device and software on the host. A pipe has several attributes; for example, a

pipe may transfer data as streams (stream pipe) or messages (message pipe).

See also stream pipe and message pipe.

**PLL** See Phase Locked Loop.

**Polling** Asking multiple devices, one at a time, if they have any data to transmit.

**POR** See Power On Reset.

**Port** Point of access to or from a system or circuit. For the USB, the point where a

USB device is attached.

**Power On Reset**

**(POR)**

Restoring a storage device, register, or memory to a predetermined state when

power is applied.

**Programmable**

**Data Rate**

Either a fixed data rate (single-frequency endpoints), a limited number of data

rates (32 kHz, 44.1 kHz, 48 kHz, …), or a continuously programmable data

rate. The exact programming capabilities of an endpoint must be reported in

the appropriate class-specific endpoint descriptors.

**Protocol** A specific set of rules, procedures, or conventions relating to format and timing

of data transmission between two devices.

**RA** See rate adaptation.

**Rate Adaptation** The process by which an incoming data stream, sampled at Fsi, is converted to

an outgoing data stream, sampled at Fso,with a certain loss of quality,

determined by the rate adaptation algorithm. Error control mechanisms are

required for the process. Fsi and Fso can be different and asynchronous. Fsi is

the input data rate of the RA; Fso is the output data rate of the RA.

**Request** A request made to a USB device contained within the data portion of a SETUP

packet.

**Retire** The action of completing service for a transfer and notifying the appropriate

software client of the completion.

**Root Hub** A USB hub directly attached to the Host Controller. This hub (tier 1) is

attached to the host.

**Root Port** The downstream port on a Root Hub.

**Sample** The smallest unit of data on which an endpoint operates; a property of an

endpoint.

**Sample Rate (Fs)** The number of samples per second, expressed in Hertz (Hz).

**Sample Rate**

**Conversion (SRC)**

A dedicated implementation of the RA process for use on sampled analog data

streams. The error control mechanism is replaced by interpolating techniques.

**Service** A procedure provided by a System Programming Interface (SPI).

**Service Interval** The period between consecutive requests to a USB endpoint to send or receive

data.

**Service Jitter** The deviation of service delivery from its scheduled delivery time.

**Service Rate** The number of services to a given endpoint per unit time.

**SOF** See Start-of-Frame.

**SOP** Start-of-Packet.

**SPI** See System Programming Interface.

**Split transaction** A transaction type supported by host controllers and hubs. This transaction

type allows full- and low-speed devices to be attached to hubs operating at

high-speed.

**SRC** See Sample Rate Conversion.

**Stage** One part of the sequence composing a control transfer; stages include the Setup

stage, the Data stage, and the Status stage.

**Start-of-Frame**

**(SOF)**

The first transaction in each (micro)frame. An SOF allows endpoints to

identify the start of the (micro)frame and synchronize internal endpoint clocks

to the host.

**Stream Pipe** A pipe that transfers data as a stream of samples with no defined USB

structure.

**Synchronization**

**Type**

A classification that characterizes an isochronous endpoint’s capability to

connect to other isochronous endpoints.

**Synchronous RA** The incoming data rate, Fsi, and the outgoing data rate, Fso, of the RA process

are derived from the same master clock. There is a fixed relation between Fsi

and Fso.

**Synchronous SRC** The incoming sample rate, Fsi, and outgoing sample rate, Fso, of the SRC

process are derived from the same master clock. There is a fixed relation

between Fsi and Fso.

**System**

**Programming**

**Interface (SPI)**

A defined interface to services provided by system software.

**TDM** See Time Division Multiplexing.

**TDR** See Time Domain Reflectometer.

**Termination** Passive components attached at the end of cables to prevent signals from being

reflected or echoed.

**Time Division**

**Multiplexing**

**(TDM)**

A method of transmitting multiple signals (data, voice, and/or video)

simultaneously over one communications medium by interleaving a piece of

each signal one after another.

**Time Domain**

**Reflectometer**

**(TDR)**

An instrument capable of measuring impedance characteristics of the USB

signal lines.

**Timeout** The detection of a lack of bus activity for some predetermined interval.

**Token Packet** A type of packet that identifies what transaction is to be performed on the bus.

**Transaction** The delivery of service to an endpoint; consists of a token packet, optional data

packet, and optional handshake packet. Specific packets are allowed/required

based on the transaction type.

**Transaction**

**translator**

A functional component of a USB hub. The Transaction Translator responds to

special high-speed transactions and translates them to full/low-speed

transactions with full/low-speed devices attached on downstream facing ports.

**Transfer** One or more bus transactions to move information between a software client

and its function.

**Turn-around Time** The time a device needs to wait to begin transmitting a packet after a packet

has been received to prevent collisions on the USB. This time is based on the

length and propagation delay characteristics of the cable and the location of the

transmitting device in relation to other devices on the USB.

**Universal Serial**

**Bus Driver (USBD)**

The host resident software entity responsible for providing common services to

clients that are manipulating one or more functions on one or more Host

Controllers.

**Universal Serial**

**Bus Resources**

Resources provided by the USB, such as bandwidth and power. See also

Device Resources and Host Resources.

**Upstream** The direction of data flow towards the host. An upstream port is the port on a

device electrically closest to the host that generates upstream data traffic from

the hub. Upstream ports receive downstream data traffic.

**USBD** See Universal Serial Bus Driver.

**USB-IF** USB Implementers Forum, Inc. is a nonprofit corporation formed to facilitate

the development of USB compliant products and promote the technology.

**Virtual Device** A device that is represented by a software interface layer. An example of a

virtual device is a hard disk with its associated device driver and client

software that makes it able to reproduce an audio .WAV file.

**Word** A data element that is two bytes (16 bits) in size.

## 功能

USB主要是用于个人电脑PC的功能扩展

The USB is specified to be an industry-standard extension to the PC architecture with a focus on PC

peripherals that enable consumer and business applications. The following criteria were applied in defining

the architecture for the USB:

Ease-of-use for PC peripheral expansion

Low-cost solution that supports transfer rates up to 480 Mb/s

Full support for real-time data for voice, audio, and video

Protocol flexibility for mixed-mode isochronous data transfers and asynchronous messaging

Integration in commodity device technology

Comprehension of various PC configurations and form factors

Provision of a standard interface capable of quick diffusion into product

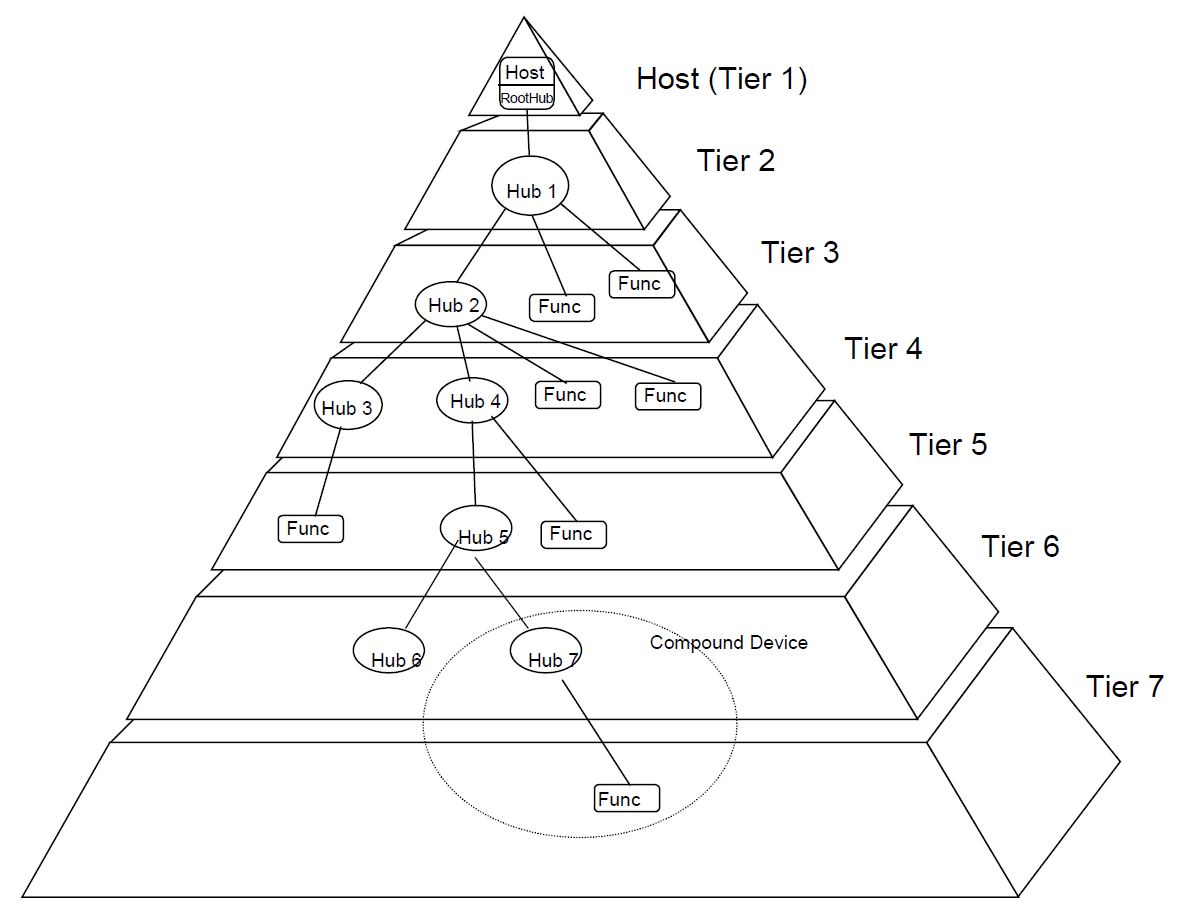
Enabling new classes of devices that augment the PC’s capability

Full backward compatibility of USB 2.0 for devices built to previous versions of the specification

## 应用分类

|  |  |  |
| --- | --- | --- |
|  | 应用 | 属性 |
| 低速USB | 键盘、鼠标、游戏外设、VR | 10~100 Kb/s |
| 全速USB | 宽频信号、麦克风 | 500Kb/s ~ 10Mb/s |
| 高速USB | 视频设备、存储设备、成像、宽频信号 | 25~400 Mb/s |

## USB体系



USB 体系包括“主机”、“设备”以及“物理连接”三个部分。

主机是一个提供USB接口及接口管理能力的硬件、软件及固件的复合体，可以是PC，也可以是OTG设备。一个USB 系统中仅有一个USB主机；

设备包括 USB功能设备和 USB HUB，最多支持 127个设备；

物理连接即指的是USB 的传输线。

在USB 2.0系统中，要求使用屏蔽的双绞线。

USB 体系采用分层的星型拓扑来连接所有USB设备

以 HOST-ROOT HUB为起点，最多支持 7 层（Tier），也就是说任何一个USB 系统中最多可以允许 5个 USB HUB 级联。

一个复合设备（Compound Device）将同时占据两层或更多的层。

ROOT HUB 是一个特殊的 USB HUB，它集成在主机控制器里，不占用地址。

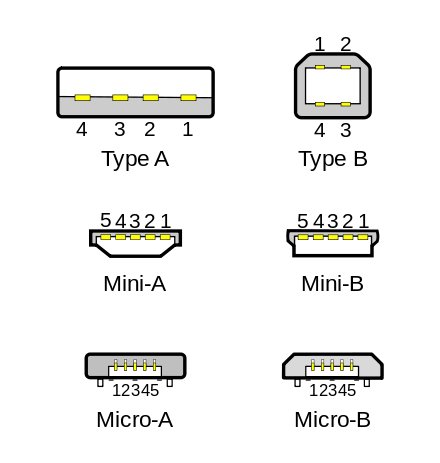
“复合设备（Compound Device）”可以占用多个地址。所谓复合设备其实就是把多个功能设备通过内置的 USB HUB 组合而成的设备，比如带录音话筒的 USB 摄像头等。

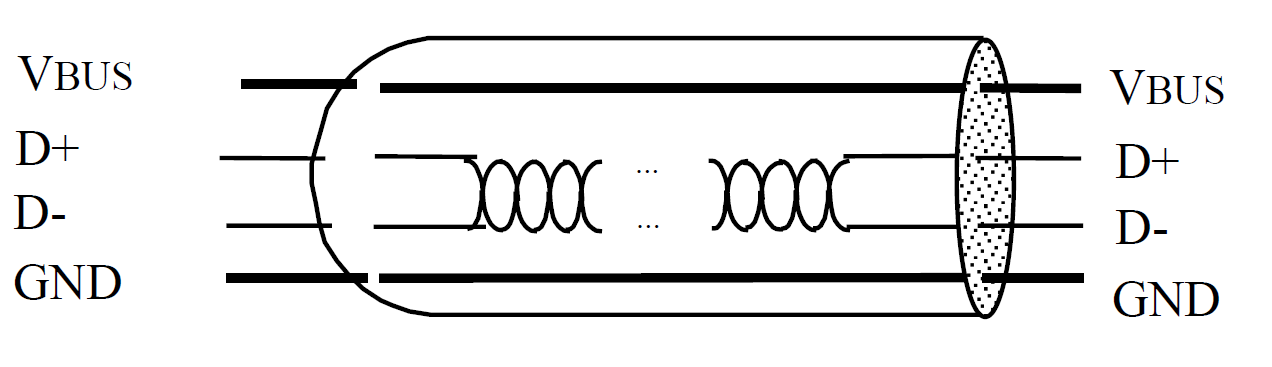
一个 USB HOST 最多可以同时支持 128 个地址，地址 0 作为默认地址，只在设备枚举期间临时使用，而不能被分配给任何一个设备，因此一个 USB HOST 最多可以同时支持 127 个地址，如果一个设备只占用一个地址，那么可最多支持 127个 USB 设备（含USB HUB）。

物理层

## 物理连接

USB接口类型：





如图所示是两个设备相连接的示意图。

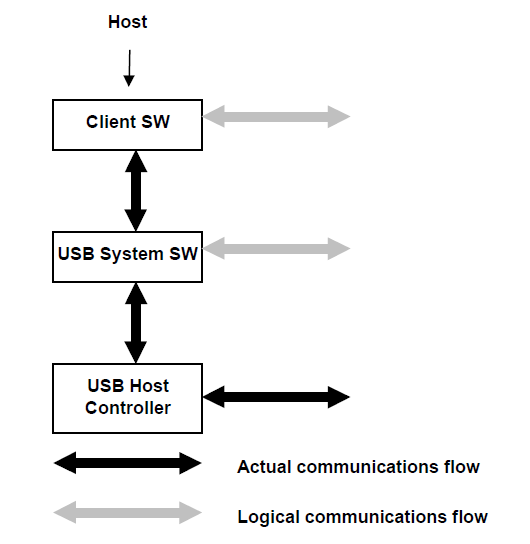
一个USB接口有4个引脚：2个电源引脚和2个信号引脚

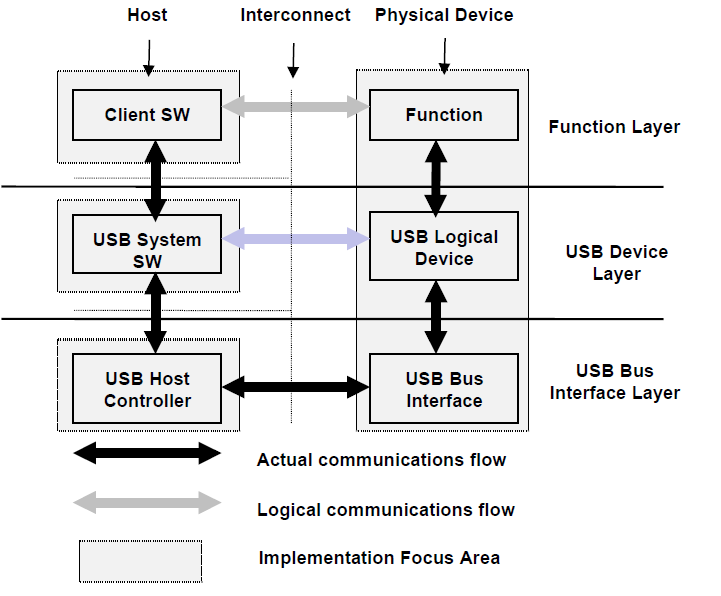
|  |  |  |
| --- | --- | --- |
| 引脚 |  |  |
| VBUS |  |  |
| D+ |  |  |
| D- |  |  |
| GND |  |  |

## 总线结构（总线拓扑）

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| USB主机 | USB主控制器 |  | 在USB总线结构中只有一个主机  主机包含一个根集线器（root hub）  The Host Controller may be implemented in a combination of hardware, firmware, or  software. A root hub is integrated within the host system to provide one or more attachment points |
| USB系统软件 | 包括：USB驱动、USB主控制器驱动、主机软件 |
| 客户端 |  |
| USB从机 |  |  | 在USB总线结构中可以有多个从机  从机可以是：  USB devices are one of the following:  Hubs, which provide additional attachment points to the USB  Functions, which provide capabilities to the system, such as an ISDN connection, a digital joystick, or  speakers  USB devices present a standard USB interface in terms of the following:  Their comprehension of the USB protocol  Their response to standard USB operations, such as configuration and reset  Their standard capability descriptive information |
| 分层星形拓扑结构 |  |  | 每一层表示一个设备，第一层是USB主机，下面的都是USB从机  The USB connects USB devices with the USB host. The USB physical interconnect is a tiered star  topology. A hub is at the center of each star. Each wire segment is a point-to-point connection between the  host and a hub or function, or a hub connected to another hub or function. Figure 4-1 illustrates the  topology of the USB.  Due to timing constraints allowed for hub and cable propagation times, the maximum number of tiers  allowed is seven (including the root tier). Note that in seven tiers, five non-root hubs maximum can be  supported in a communication path between the host and any device. A compound device (see Figure 4-1)  occupies two tiers; therefore, it cannot be enabled if attached at tier level seven. Only functions can be  enabled in tier seven. |
|  |  |  |  |

## USB主机





|  |  |
| --- | --- |
| 功能层 | 主机通过客户端软件层  逻辑通信 |
| USB设备层 | 软件控制USB设备  逻辑通信 |
| USB总线接口层 | 主机与从机之间，物理、信号、数据包的连接p  实际通信 |

协议层

# USB传输数据结构

## 概述

发送数据时，先发送低位，由低向高进行发送。

接收数据时，收到的数据先放在存储空间的低地址位，由低向高进行放置。

这样发送的数据和接收的数据就是一致的。

## 数据包

数据在 USB总线上的传输以包为单位，若干个包组成一帧，一帧表示一个完整数据。

有4种类型的数据包：令牌包(Token)、数据包(Data)、握手包(HandShake)、特殊包(Special)

数据包的一般格式：

从左到右的顺序是数据位由低到高的顺序

不同的数据包格式可能不同

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fields | PID | ADDR | ENDP | DATA | CRC |
| FrameNumber | |
| 位宽 | 4+4 | 7 | 4 | N\*8  （N=0,1,2,……1024） | 5/16 |
| 11 | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PID | 一共8位  低4位用于指明该数据包的类型，高四位是数据包检测位 | | | |
| 数据位（3:0） | 数据包类型 | 具体类型 |  |
| 0001 | 令牌包(Token) | OUT | 只有主机可以发送令牌包 |
| 1001 | 令牌包(Token) | IN |
| 0101 | 令牌包(Token) | SOF |
| 1101 | 令牌包(Token) | SRTUP |
| 0011 | 数据包(Data) | DATA0 |  |
| 1011 | 数据包(Data) | DATA1 |  |
| 0111 | 数据包(Data) | DATA2 |  |
| 1111 | 数据包(Data) | MDATA |  |
| 0010 | 握手(HandShake) | ACK |  |
| 1010 | 握手(HandShake) | NAK |  |
| 1110 | 握手(HandShake) | STALL |  |
| 0110 | 握手(HandShake) | NYET |  |
| 1100 | 特殊包(Special) | PRE |  |
| 1100 | 特殊包(Special) | ERR |  |
| 1000 | 特殊包(Special) | SPLIT |  |
| 0100 | 特殊包(Special) | PING |  |
| 0000 | 特殊包(Special) | RESERVED |  |
| 数据包检测位 （7:4）  这四位用于检验数据包类型位的有效性，方法是：  数据包检测位设置为数据包类型位的反码。如  OUT令牌包的PID为：1110 0001  ACK握手包的PID为：1101 0010  接收器对收到的数据包会进行检测，如果发现数据包的PID的高4位不是低4位的反码，那么这个数据包无效 | | | |
| ADDR | 7位地址位  一共有128个值，每一个值指定一项功能  默认为000 0000 | | | |
| ENDP | 4位端点号  一共16个值，每一个值指定一项功能  默认为0000 | | | |
| FrameNumber | 11位帧编号  只出现在SOF令牌包中  帧编号的值会被主机增加，每帧加1，达到最大值7FFH后，返回为0 | | | |
| DATA | 8位数据域  只有数据包有数据域。  数据域取值是 0到1023，0000 0000 ~~ FFFF FFFF | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OUT、IN、SRTUP 令牌包 | | | | |
| Fields | PID | ADDR | ENDP | CRC |
| Bits | 4+4 | 7 | 4 | 5 |
| 说明 | For OUT and SETUP transactions, the address and endpoint fields uniquely  identify the endpoint that will receive the subsequent Data packet. For IN transactions, these fields uniquely  identify which endpoint should transmit a Data packet.  An IN PID defines a Data transaction from a function to the host. OUT and SETUP PIDs define Data  transactions from the host to a function. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| SOF 令牌包 | | | |
| Fields | PID | FrameNumber | CRC |
| Bits | 1010 0101 | 11 | 5 |
| 说明 | SOF是一类特殊的令牌包，用于表示一个数据帧的起始标志。  PID 后跟的是11 位的帧编号。 | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PING 特殊令牌包 | | | | |
| Fields | PID | ADDR | ENDP | CRC |
| Bits | xx11 xx00 | 7 | 4 | 5 |
| 说明 | For PING transactions, these fields uniquely  identify which endpoint will respond with a handshake packet.  A PING PID defines a handshake transaction from the function to  the host. | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SPLIT 特殊令牌包 | | | | | | | | |
| Fields | PID | ADDR | SC | PORT | S | E | ET | CRC |
| Bits | xx11 xx00 | 7 | 1 | 7 | 1 | 1 | 2 | 5 |
| 说明 | 相比于一般的令牌包只有3个字节，SPLIT有个字节  start-split transaction (SSPLIT) → SC=0，  complete-split transaction(CSPLIT) → SC=1， | | | | | | | |
|  |  | | | | | | | |
|  |  | | | | | | | |
|  |  | | | | | | | |
|  |  | | | | | | | |
|  |  | | | | | | | |
|  |  | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 数据包 | | | |
| Fields | PID | DATA | CRC |
| Bits | xx00 xx11 | N\*8  （N=0,1,2,……1024） | 16 |
| 说明 | 对于数据包来说，PID 之后直接跟数据域，数据域的长度为N字节，数据域后以 16 位的 CRC 校验和结束; | | |

|  |  |
| --- | --- |
| 握手包 | |
| Fields | PID |
| Bits | xx01 xx10 |
| 说明 | 握手包仅有PID 域，没有数据也没有校验和 |

|  |  |
| --- | --- |
| 特殊包 | |
| Fields | PID |
| Bits | xx11 xx00 |
| 说明 | 握手包仅有PID 域，没有数据也没有校验和 |

## 数据帧

|  |  |
| --- | --- |
| SYNC | SYNC处于数据包的最低的位置处，所以发送数据时最先发送的是SYNC  对于高速USB，SYNC为00……01，共31个0，最低位是1，即发送整个数据包时最先发送一个逻辑“1”  对于全速和低速USB，SYNC为00……01，共7个0，最低位是1，即发送整个数据包时最先发送一个逻辑“1”  实际接收到的 SYNC长度由于USB HUB 的关系，可能会小于该值  SYNC的功能是用于同步 |

|  |
| --- |
| SYNC |
| 高速USB 总线的SYNC为32位  全速和低速USB的SYNC为 8位 |

数据在 USB总线上的传输以包为单位，若干个包组成一帧，一帧表示一个完整数据。

高速USB 总线的帧周期为125uS，全速和低速 USB 总线的帧周期为 1ms。

|  |  |  |  |
| --- | --- | --- | --- |
| 一帧数据 | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Token and SOF packets are delimited by an EOP

after three bytes of packet field data. If a packet decodes as an otherwise valid token or SOF but does not

terminate with an EOP after three bytes, it must be considered invalid and ignored by the receiver.

帧的起始由一个特定的包（SOF 包）表示，帧尾为 EOF。EOF不是一个包，而是一种电平状态，EOF期间不允许有数据传输。

注意：虽然高速 USB总线和全速/低速 USB总线的帧周期不一样，当时 SOF包中帧编号的增加速度是一样的，因为在高速 USB系统中，SOF包中帧编号实际上取得是计数器的高 11位，最低三位作为微帧编号没有使用，因此其帧编号的增加周期也为 1mS。

# 错误检测与处理

## CRC计算

循环冗余检测位，在令牌包中占5位，在数据包中占16位。自动生成，不需要用户写

PID域有自己的检测位，而其他的域则需要使用CRC来进行检测。

CRC位处于数据包的最高位的位置，是最晚传输的，如果检测不正确，那么数据接收者会舍弃整个数据包

计算CRC位的方法：

以CRC8校验为例，即生成8位CRC码：

例如：  信息字段代码为: 00000001 00000010    ，生成多项式为：g(x)=X8+X5+X4+1

那么  信息字段代码对应的多项式为m(x)=X8+X  ，生成多项式对应的二进制代码为：1 0011 0001

现在我要对2字节数据0x0102生成CRC8校验码，并最终将生成的1字节CRC校验码跟在0x0102的后面，即 0x01 02 ##，（##即8为CRC码），最终生成的3字节数据就是经CRC8校验生成的数据。

先计算X8 \* m(x)=X16+X9，对应的2进制数为：100000010 00000000  。

可以看到这样运算所得到的结果其实就是将信息字段代码的数左移8位。因为最终要将生成的8位CRC8校验码附在信息字段的后面，所以要将信息字段的数左移8位，右边补出的8位用来存放CRC。

用x8 \* m(x)得到的二进制数 对 生成多项式g(x)进行模二运算，最终的余数就是所要的CRC8校验码。

 除数与被除数高位对齐；

 ^是异或符号；

100000010 00000000     ^ 1 0011 0001    =  000110011 00000000     ，去掉高位的0

110011 00000000   ^ 1 0011 0001   = 010101  00100000         ，去掉高位的0

10101  00100000 ^   10011 0001   = 00110 00110000         ，去掉高位的0

110 00110000  ^    100 110001   =  010 11110100            ，去掉高位的0

10 11110100   ^ 10 0110001    =  00 10010110         ，去掉高位的0

10010110

这个8位的二进制数就是CRC8校验码。

所以，经CRC8校验后研发送的数据就是0x010296。

再举几个例子：

信息字段11011100 的CRC为 01111001 (0x79).

信息字段01101000 00111010 的CRC为  01111100 (0x7C).

信息字段01001110 10000101  的CRC为  01101011 (0x6B).

令牌包：

信息字段是ADDR ENDP 共11位待检测位；

（注意发送数据时，是从低到高发送的，比如ADDR=100 0101 EDRP=0001

那么发送的顺序是：1-0-1-0-0-0-1 - 1-0-0-0

我们在计算CRC时使用的  信息字段代码为0001 100 0101）；

使用多项式为G(X) = X5 + X2 + 1；

需要生成5位CRC位；

数据包：

信息字段是DATA 共8位待检测位；

使用多项式为G(X) = X16 + X15 + X2 + 1

需要生成11位CRC位

# USB传输数据过程

## 批量事务传输

批量传输是可靠的传输，需要握手包来表明传输的结果。

若数据量比较大，将采用多次批量事务传输来完成全部数据的传输，传输过程中数据包的PID 按照 DATA0-DATA1-DATA0-…的方式翻转，以保证发送端和接收端的同步。

USB 允许连续 3次以下的传输错误，会重试该传输，若成功则将错误次数计数器清零，否则累加该计数器。超过三次后，HOST 认为该端点功能错误（STALL），放弃该端点的传输任务。

一次批量传输（Transfer）由 1 次到多次批量事务传输（Transaction）组成。

翻转同步：发送端按照 DATA0-DATA1-DATA0-…的顺序发送数据包，只有成功的事务传输才会导致 PID 翻转，也就是说发送段只有在接收到 ACK 后才会翻转 PID，发送下一个数据包，否则会重试本次事务传输。同样，若在接收端发现接收到到的数据包不是按照此顺序翻转的，比如连续收到两个 DATA0，那么接收端认为第二个 DATA0 是前一个 DATA0 的重传。

## 控制传输

一次控制传输分为三（或两个）个阶段：

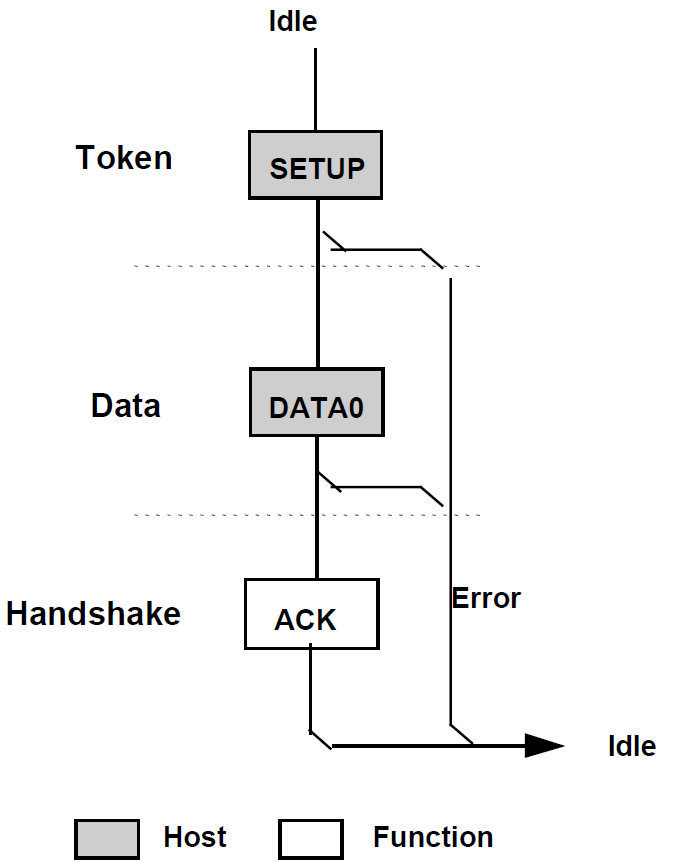
建立（Setup）

数据（DATA）（可能没有）

状态（Status）

每个阶段都由一次或多次（数据阶段）事务传输组成（Transaction）

下图为建立阶段的事务传输流程图。可以看出：与批量传输相比，在流程上并没有多大区别，区别只在于该事务传输发生的端点不一样、支持的最大包长度不一样、优先级不一样等这样一些对用户来说透明的东西。



建立阶段过后，可能会有数据阶段，这个阶段将会通过一次或多次控制传输事务，完成数据的传输。同样也会采用PID翻转的机制。建立阶段，Device只能返回 ACK包，或者不返回任何包。

最后是状态阶段，通过一次方向与前一次相反的控制事务传输来表明传输的成功与否。如果成功会返回一个长度为 0 的数据包，否则返回 NAK或 STALL。下图为整个控制传输的示意图：

## 中断传输

中断传输在流程上除不支持PING 之外，其他的跟批量传输是一样的。

他们之间的区别也仅在于事务传输发生的端点不一样、支持的最大包长度不一样、优先级不一样等这样一些对用户来说透明的东西。

主机在排定中断传输任务时，会根据对应中断端点描述符中指定的查询间隔发起中断传输。中断传输有较高的优先级，仅次于同步传输。

同样中断传输也采用PID翻转的机制来保证收发端数据同步。下图为中断传输的流程图。

## 同步传输

同步传输是不可靠的传输，所以它没有握手包，也不支持PID翻转。主机在排定事务传输时，同步传输有最高的优先级。

USB OTG总线协议

# 介绍

OTG是On-The-Go的缩写，自1996 年USB1.0规范以后，USB-IF(Universal Serial Bus Implementers Forums)又陆续公布了USB2.0和 USB OTG等几个规范，其中USB2.0的传输带宽达到480Mbps，而USB OTG更使USB装置摆脱了原来主从架构的限制，实现了端对端的传输模式。

USB OTG标准在完全兼容USB2.0标准的基础上，增添了电源管理（节省功耗）功能，它允许设备既可作为主机，也可作为外设操作（两用OTG）。

USB OTG标准支持主机通令协议（[HNP](http://baike.baidu.com/view/3301671.htm)）和对话请求协议（[SRP](http://baike.baidu.com/view/1545205.htm)）。

## 主机协商协议

在USB OTG中，**初始**主机设备采用A型接口，称为A类设备(A-[Device](http://baike.baidu.com/view/1542026.htm))；从机采用B型接口，称为B类设备(B-Device)。

1个双角色设备（DRD）既可以作为主机，也可以作为从机外设。

2个DRD互连时，使用一种新的接口——微型AB插座(mini-AB receptacle)以及微型A插头(mini-A plug)和微型B插头(mini-B plug)。

在微型AB插座、微型A插头和微型B插头中增加了1个引脚——ID引脚。

mini-A插头中的ID引脚接地，mini-B插头中的ID引脚浮空。

当OTG设备检测到接地的ID引脚时，表示默认的是A设备（主机），而检测到ID引脚浮着的设备则认为是B设备（外设）。

系统一旦连接后，初始的状态决定了哪个设备是A设备，哪个设备是B设备，那么在这之后，这个属性将不会改变，之后可以通过主机协商协议，允许设备互换主机和外设的角色，而不必拔下连接线调换插头方向，重新连接。

A设备作为默认主机并提供VBUS电源，并在检测到有设备连接时复位总线、枚举并配置B设备。

在OTG中电源一直都是由A类设备(连接mini-Aplug的DRD)提供的。

由于主机协商协议，A类设备也可能作为外设使用，此时，电源仍然是由A类设备提供。

OTG收发器一般用在嵌入式设备中，这类设备普遍采用电池供电，对功耗要求很严。为了节省电源，在OTG标准中，当电源总线没有使用时，允许A类设备挂起电源总线。

## 对话请求协议（[SRP](http://baike.baidu.com/view/1545205.htm)）

OTG标准为USB增添的第二个新协议称为对话请求协议（[SRP](http://baike.baidu.com/view/1545205.htm)）。

当1个B类设备要工作时，它必须通过某种方法通知A类设备向电源总线供电。为了实现这一功能，在OTG中提出了会话请求协议(SRP)。

在OTG中，1个会话定义为A类设备向电源总线VBUS有效供电的时间。

当A类设备挂起VBUS后，B类设备进入休眠状态。当B类设备需要再次工作时，它可以通过向数据线发送1个脉冲信号(Data-linePulsing)或向电源总线发送一个脉冲信号(VBUSPulsing)来请求A类设备向电源总线供电。

OTG要求无论是DRD设备还是普通的B类设备，都必须具有发送会话请求的功能；同时，普通的A类设备或者DRD设备都必须能够响应1个会话请求。

也可通过A设备关闭VBUS电源来结束一次会话以节省功耗，这在电池供电产品中是非常重要的。

例如，在两台[蜂窝电话](http://baike.baidu.com/view/800226.htm)通过连接互相交换信息时，一台连接在费电的mini-A端，是A设备，默认为主机。另一台是B设备，默认为外设。当在不需要USB通信时，A设备可以关闭VBUS线，此时B设备就会检测到该状态并进入低功耗模式。

## OTG功能构建

构建OTG功能时需要在基础USB外设上添加电路，电路中的[通用串行总线](http://baike.baidu.com/view/474321.htm)控制器可以是一个微处理器和USB SIE（串口引擎），也可以是集成的μP/USB芯片或与USB收发器相连的ASIC。为总线提供电源的外部设备需要一路3.3V稳压输出供电电压，以便为逻辑电路和连接在D+、D-引脚的1500Ω电阻提供电源。

通过D+、D-引脚上的上拉电阻可向主机发出设备已连接的信号，并指示设备的工作速度。电阻上拉至D+表示全速运行，电阻上拉至D-表示低速运行。

其它端点（包括D+和D-的15kΩ下拉电阻）用于检测上拉电阻的状态。

由于USB设计需要提供热插拔功能。因此，其[ESD](http://baike.baidu.com/view/176445.htm)保护电路主要用于为D+、D-和VBUS引脚提供保护。

为了增加OTG的两用功能，必须扩充收发器功能来使OTG设备既可作为[主机](http://baike.baidu.com/view/23880.htm)使用，也可以作为外设使用。

要实现上述功能，就需要添加D+和D-端的15kΩ下拉电阻并为VBUS提供供电电源。此外，收发器还需要具备以下三个条件：

（1）可切换D+/D-线上的上拉和下拉电阻，以提供外设和主机功能。

（2）作为A设备时，需要具有VBUS监视和供电电路；作为B设备初始化SRP时，需要监视和触发VBUS。

（3）具有ID输入引脚。

作为两用OTG设备，[ASIC](http://baike.baidu.com/view/111601.htm)、[DSP](http://baike.baidu.com/subview/1192/10810131.htm)或其它与收发器连接的电路必须具备充当外设和主机的功能，并应按照HNP协议转换其角色。

收发器所需添加的大多数电路用于VBUS[引脚](http://baike.baidu.com/view/641241.htm)的管理。作为主机，它必须能够提供5V、输出电流可达8mA的电源。

ASIC和控制器还必须包含USB主机逻辑控制功能，包括

发送SOF（帧启动）包

发送配置\u36755输入\u36755输出数据包

在USB 1 msec帧内确定传输进程

发送USB[复位信号](http://baike.baidu.com/view/4507848.htm)

提供USB电源管理等。