**OTG Module Description Document**

**1. Introduction**

The following abbreviations are used in this document:

LS Low Speed

FS Full Speed

HS High Speed

MAC Media Access Controller

OTG On-The-Go

PFC Packet FIFO Controller

PHY Physical Layer

USB Universal Serial Bus

UTMI USB Transceiver Macro Unit Controller

ULPI UTMI+ interface with fewer pins

Reference Documents:

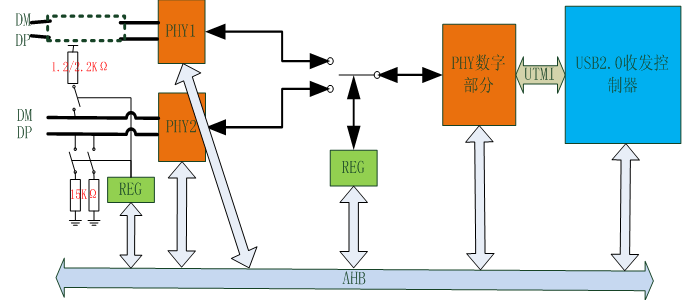
URL: www.usb.org

Universal Serial Bus Revision 2.0 Specification

**2. Introduction**

The OTG module integrates 1 USB 2.0 controller and 2 USB 1.1 full-speed PHY interfaces, does not support external PHYs, and supports both HOST and DEVICE roles. In HOST mode, it supports full-speed (FS, 12 Mb/s) and low-speed (LS, 1.5 Mb/s) transfers. In DEVICE mode, it only supports full-speed (FS, 12 Mb/s) transfers. It does not require an ID line to identify inserted A-B class devices and can dynamically switch between host and device roles.

The OTG hardware block diagram is as follows:



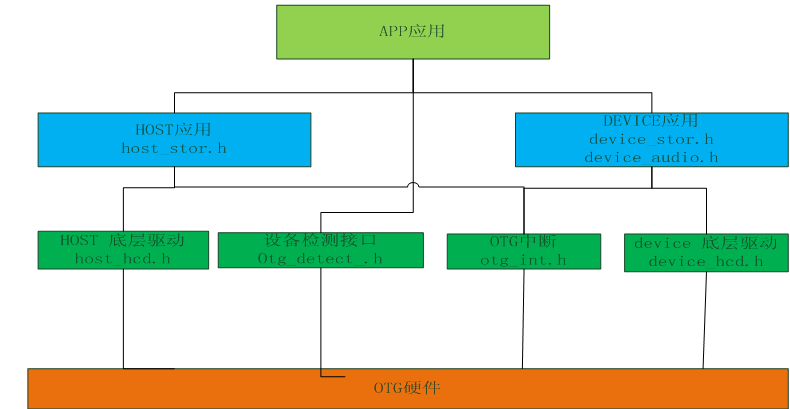
Translation：

PHY数字部分=PHY digital section

USB2.0收发控制器=USB2.0 transceiver controller

Figure 1 OTG hardware block diagram

OTG software code framework:



Translation：

APP应用= APP application

HOST应用= HOST application DEVICE应用= DEVICE application

HOST底层驱动= HOST low-level driver

设备检测接口= Device detection interface

OTG中断= OTG interrupt

DEVICE底层驱动= DEVICE low-level driver

OTG硬件= OTG hardware

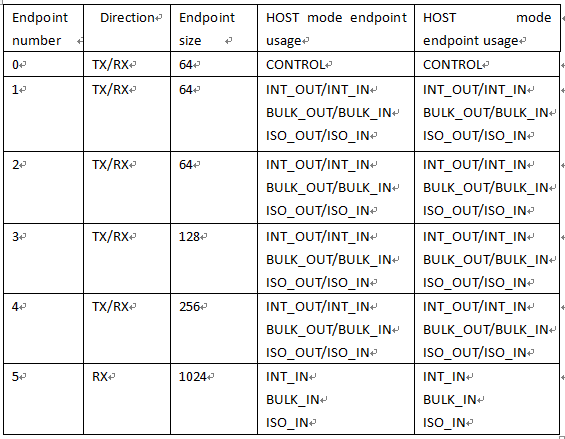
Figure 2 OTG software block diagram

Note: All layers below the APP application layer are provided in lib format.

**3. Main Features**

USB has endpoint resources internally

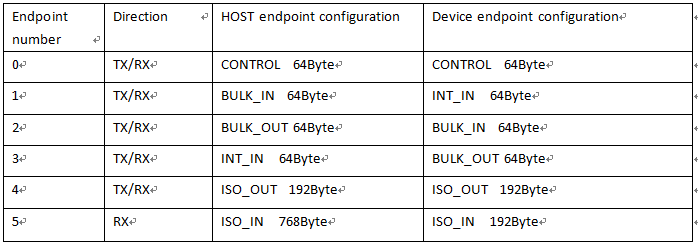
Table 1 USB Endpoint Resources



Note: If the ISO endpoint in the protocol is less than or equal to 1024 bytes, then ISO transmission sending data greater than 256 bytes will not meet the application requirements. All other endpoint resources comply with the USB 1.1 protocol standard.

AP80 chip endpoint default configuration:

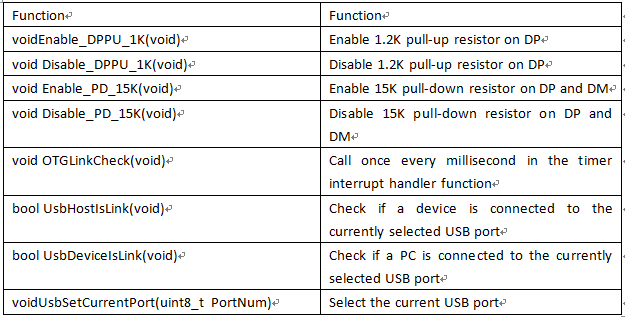
Table 2 USB endpoint default configuration



**4. Function Description**

**4.1. Device Detection**

Table 3 Device Detection Related Functions



Device Detection Application

//Interrupt-driven 1ms periodic call

OTGLinkCheck();

//Application layer checks connection status

UsbSetCurrentPort(PORT\_NUM);

Status = UsbHostIsLink()

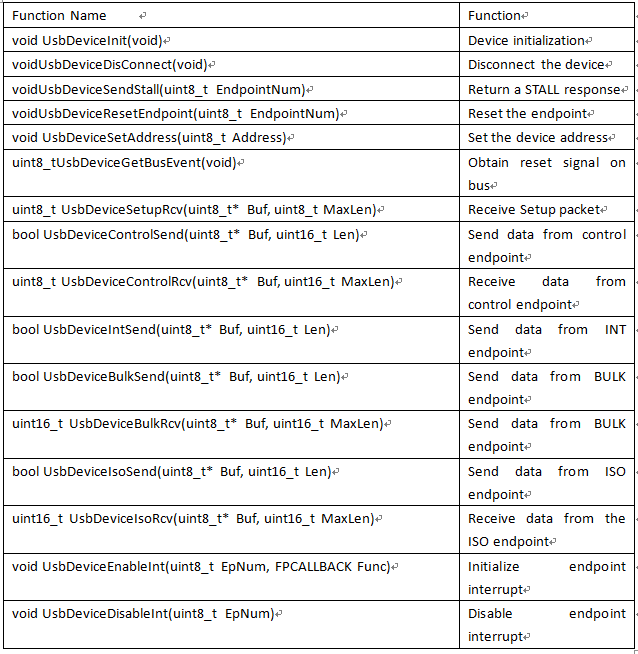
Status = UsbDeviceIsLink()

Note: Detecting whether a USB device (such as a USB flash drive) is connected is based on the status obtained from the DM and DP lines, after debouncing. Detecting whether a connection exists with the host (such as a PC) is based on the connection feedback from actual data communication. This solution cannot detect USB LS devices, such as mice, keyboards, etc., but the AP80 chip itself supports LS devices.

**4.2. Device Application**

**4.2.1. Commonly Used API Function Interfaces**

Table 4 DEVICE Application-Related Functions



**4.2.2. Usage**

1. Initialize USB device

UsbSetCurrentPort(PORT\_NUM);

UsbDeviceInit();

After executing the above functions, the corresponding port can accept data sent by the USB host.

2. Data packet transmission and reception processing

Perform device standard enumeration according to the USB 2.0 protocol

Perform class enumeration according to the corresponding USB class protocol (HID Class, CDC Class, etc.)

The enumeration process uses the following APIs to transmit and receive data:

UsbDeviceSetupRcv

UsbDeviceControlSend

UsbDeviceControlRcv

UsbDeviceIntSend

UsbDeviceBulkSend

UsbDeviceBulkRcv

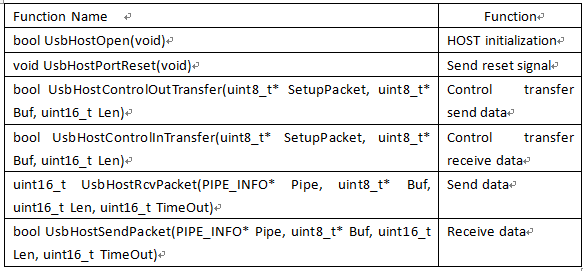
UsbDeviceIsoSend

UsbDeviceIsoRcv

**4.3. HOST Application**

**4.3.1. Commonly Used API Function Interfaces**

Table 5 HOST-Related Functions



**4.3.2. Usage**

1. Initialize the USB HSOT controller

UsbSetCurrentPort(PORT\_NUM);

UsbHostOpen();

UsbHostPortReset();

UsbHostOpen();

Execute the above code to fully open the controller and complete initialization.

2. Enumerate according to USB-related protocols. The enumeration process requires the following APIs:

bool UsbHostControlOutTransfer(uint8\_t\* SetupPacket, uint8\_t\* Buf, uint16\_t Len)

bool UsbHostControlInTransfer(uint8\_t\* SetupPacket, uint8\_t\* Buf, uint16\_t Len)

uint16\_t UsbHostRcvPacket(PIPE\_INFO\* Pipe, uint8\_t\* Buf, uint16\_t Len, uint16\_t TimeOut)

bool UsbHostSendPacket(PIPE\_INFO\* Pipe, uint8\_t\* Buf, uint16\_t Len, uint16\_t TimeOut)