Rockchip Linux USB 开发指南

发布版本：1.2.2

作者邮箱：wulf@rock-chips.com、frank.wang@rock-chips.com、daniel.meng@rockchip.com

日期：2019-11-12

文档密级：公开资料

前言

概述

产品版本

芯片名称	内核版本
RK3399、RK3368、RK3366、RK3328、RK3288、RK312X、RK3188、RK30XX、RK3308、RK3326、PX30	Linux-4.4、Linux-4.19

读者对象

软件工程师，硬件工程师，FAE

修订记录

日期	版本	作者	修改说明
2017-12-22	v1.0	吴良峰、王明成、孟东阳	初始版本
2018-06-08	v1.1	吴良峰	support RK3308、RK3326、PX30 修正格式和错误
2019-01-09	V1.2	吴良峰	使用 markdownlint 修订格式
2019-03-20	V1.2.1	吴良峰	修正 USB OTG 控制器硬件电路说明 删除段首 Tab 空格符号
2019-11-12	V1.2.2	吴良峰	修改文档名称，支持Linux-4.19

---

1 概述

1.1 RK 平台 USB 控制器方案

Rockchip SOC 通常内置多个 USB 控制器，不同控制器互相独立，请在芯片 TRM 中获取详细信息。由于部分 USB 控制器有使用限制，所以请务必明确方案的需求及控制器限制后，再确定 USB 的使用方案。各芯片内置的 USB 控制器如表 1-1 所示：

表 1-1 RK 平台 USB 控制器列表

控制器 芯片	USB 2.0 HOST (EHCI&OHCI)	USB HSIC (EHCI)	USB 2.0/3.0 OTG (DWC3/XHCI)	USB 2.0 OTG (DWC2)
RK3399	×2	×1	×2	0
RK3368	×1	×1	0	×1
RK3366	×1	0	×1	×1
RK3328	×1	0	×1	×1
RK3288	0	×1	0	×2 (host+otg)
RK312X	×1	0	0	×1
RK3188	×1	×1	0	×1
RK30XX	×1	0	0	×1
RK3308	×1	0	0	x1
RK3326	0	0	0	x1
PX30	x1	0	0	x1

1.2 USB 2.0 Host

- Compatible Specification

Universal Serial Bus Specification, Revision 2.0

Enhanced Host Controller Interface Specification(EHCI), Revision 1.0

Open Host Controller Interface Specification(OHCI), Revision 1.0a

- Features

Support high-speed(480Mbps), full-speed(12Mbps) andlow-speed(1.5Mbps)

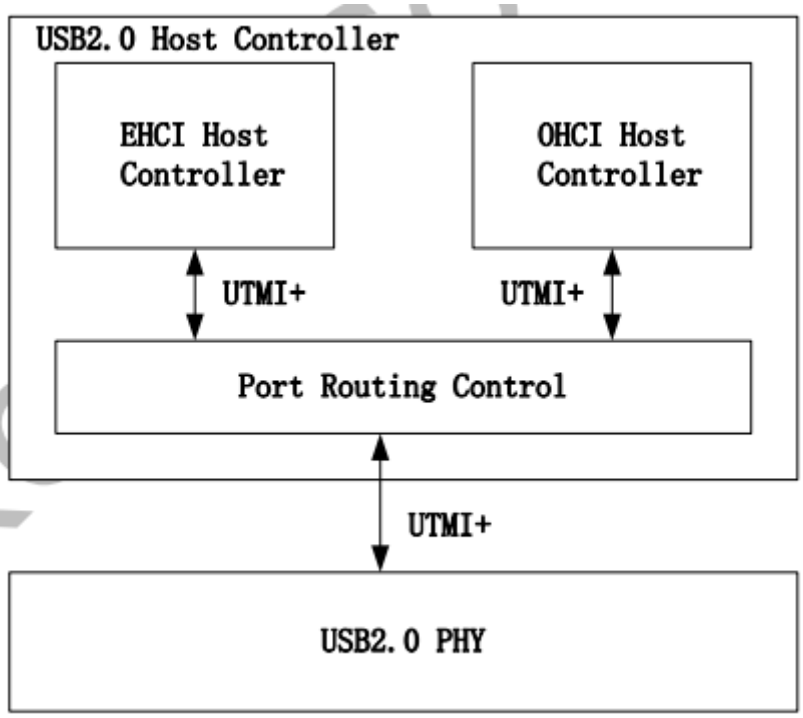

图 1-1 USB 2.0 Host Controller Block Diagram

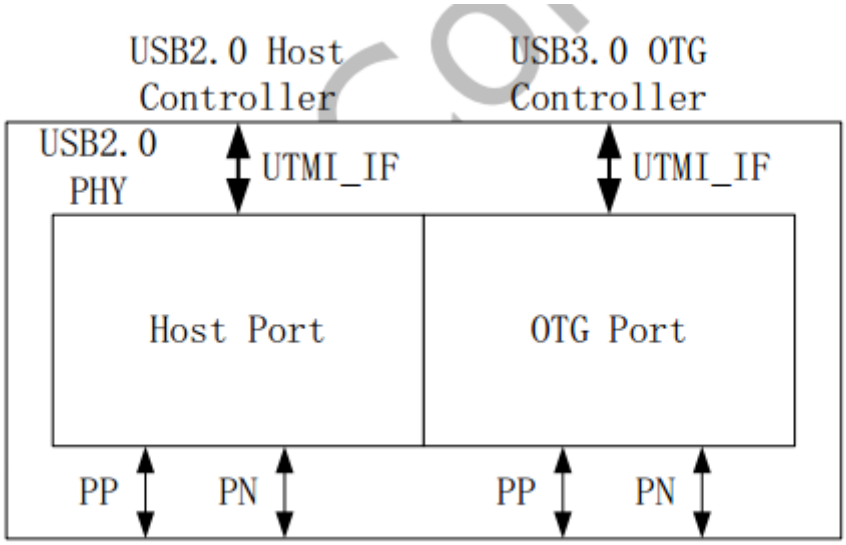

图 1-2USB 2.0 USB 2.0 PHY Block Diagram

1.3 USB 2.0 OTG

- Compatible Specification

Universal Serial Bus Specification, Revision 2.0

- Features

Operates in High-Speed and Full-Speed mode

Support 9 channels in host mode

9 Device mode endpoints in addition to control endpoint 0, 4 in, 3 out and 2 IN/OUT

Built-in one 1024x35 bits FIFO

Internal DMA with scatter/gather function

Supports packet-based, dynamic FIFO memory allocation for endpoints for flexible, efficient use of RAM

Support dynamic FIFO sizing

Support Battery Charge in device role

Support Uart Bypass Mode

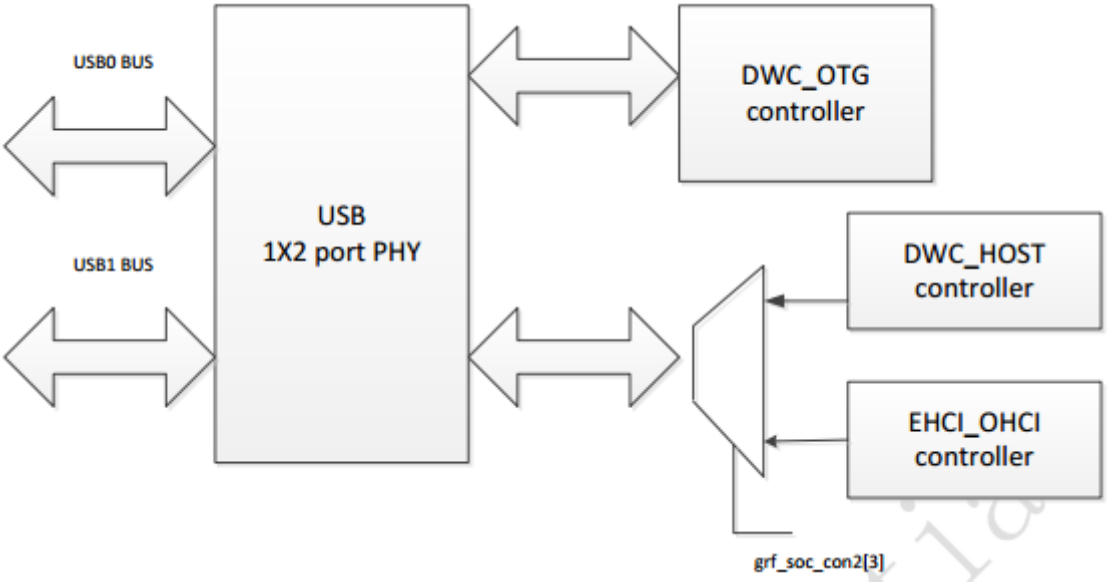

图 1-3 USB 2.0 OTG Block Diagram

1.4 USB 2.0 PHY

Host Port: used for USB 2.0 host controller

OTG Port: used for USB 3.0 OTG controller with TypeC PHY to compriseas fully feature TypeC

1.5 USB OTG 3.0

- Compatible Specification

Universal Serial Bus 3.0 Specification, Revision 1.0

Universal Serial Bus Specification, Revision 2.0

eXtensible Host Controller Interface for Universal Serial Bus(xHCI), Revision 1.1

- Features

DWC3 Features:

Support Control/Bulk(including stream)/Interrupt/IsochronousTransfer

Simultaneous IN and OUT transfer for USB 3.0, up to 8Gbps bandwidth

Descriptor Caching and Data Pre-fetching

USB 3.0 Device Features

Up to 7 IN endpoints, including control endpoint 0

Up to 6 OUT endpoints, including control endpoint 0

Up to 13 endpoint transfer resources, each one for each endpoint

Flexible endpoint configuration for multiple applications/USBset-configuration modes

Hardware handles ERDY and burst

Stream-based bulk endpoints with controller automatically initiatingdata movement

Isochronous endpoints with isochronous data in data buffers

Flexible Descriptor with rich set of features to support bufferinterrupt moderation, multiple transfers, isochronous, control, and scatteredbuffering support

USB 3.0 xHCI Host Features:

Support up to 64 devices

Support 1 interrupter

Support 1 USB 2.0 port and 1 Super-Speed port

Concurrent USB 3.0/USB 2.0 traffic, up to 8.48Gbps bandwidth

Support standard or open-source xHCI and class driver

Support xHCI Debug Capability

USB 3.0 Dual-Role Device (DRD) Features

Static Device operation

Static Host operation

USB 3.0/USB 2.0 OTG A device and B device basing on ID

UFP/DFP and Data Role Swap Defined in USB TypeC Specification

Not support USB 3.0/USB 2.0 OTG session request protocol(SRP), hostnegotiation protocol(HNP) and Role Swap Protocol(RSP)

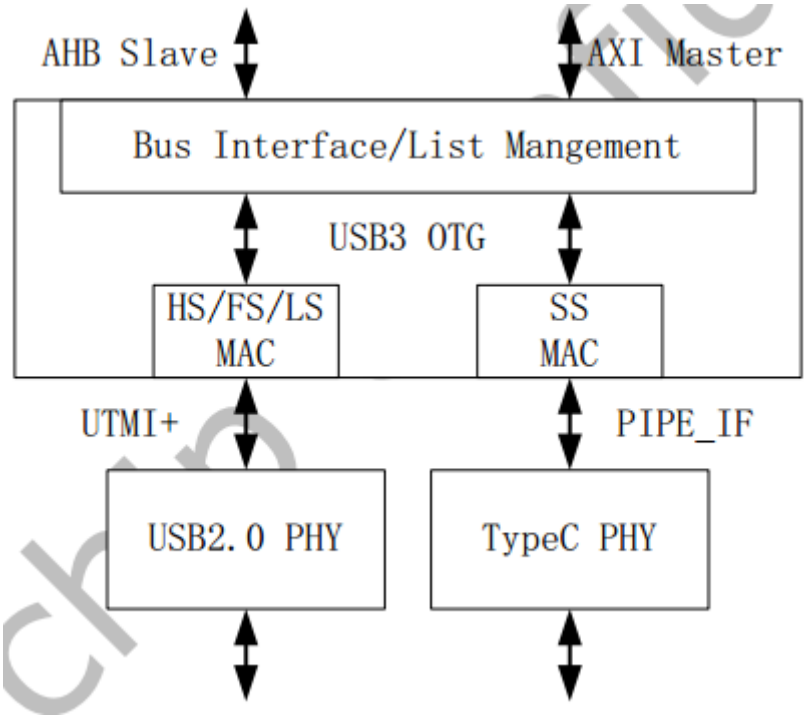

图 1-4 USB 3.0 OTG Block Diagram

1.6 TypeC PHY

Support USB 3.0 (SuperSpeed only)

Support DisplayPort 1.3 (RBR, HBR and HBR2 data rates only)

Support DisplayPort AUX channel

Support USB TypeC and DisplayPort Alt Mode

Support DisplayPort Alt Mode on TypeC A, B, C, D, E and F pinassignments

Support Normal and Flipped orientation

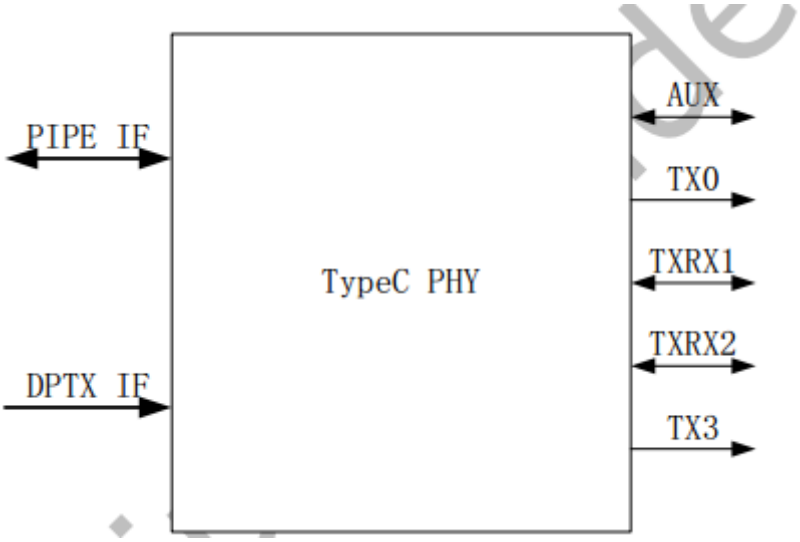

图 1-5 TypeC PHY Block Diagram

2 硬件电路及信号

2.1 USB HOST 控制器硬件电路

USB Host 控制器分别包含 USB 2.0 Host 和 HSIC，其硬件电路及信号分别说明如下：

2.1.1 USB 2.0 HOST 控制器硬件电路

USB 2.0 的工作时钟高达 480MHz，所以 layout 时需要特别注意，USB 走线宽度为 7-8MIL，做 90Ω阻抗差分走线，最好在表层走线并有包地，边上无干扰源，正对的上下层不能有其他信号走线。

USB HSIC（High Speed Inter Chip）使用 240MHz DDR 信号，传输速率与 USB 2.0 同为 480Mbps，典型的走线阻抗为 50Ω，建议最大走线长度不要超过 10cm。

USB 2.0 HOST 控制器硬件信号参考电路如图 2-1 所示，USB 2.0 HOST 的 VBUS 控制电路和接口电路如图 2-2 和图 2-3 所示：

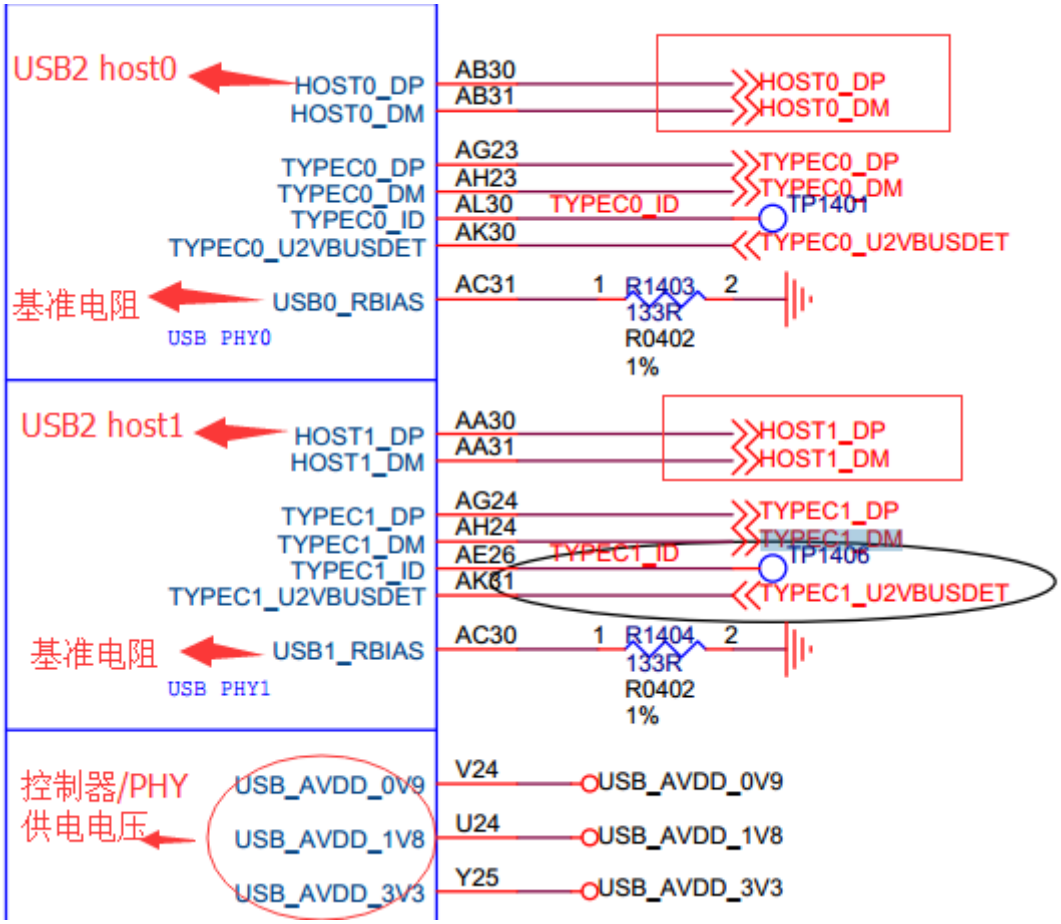

图 2-1 USB 2.0 HOST SoC 信号引脚

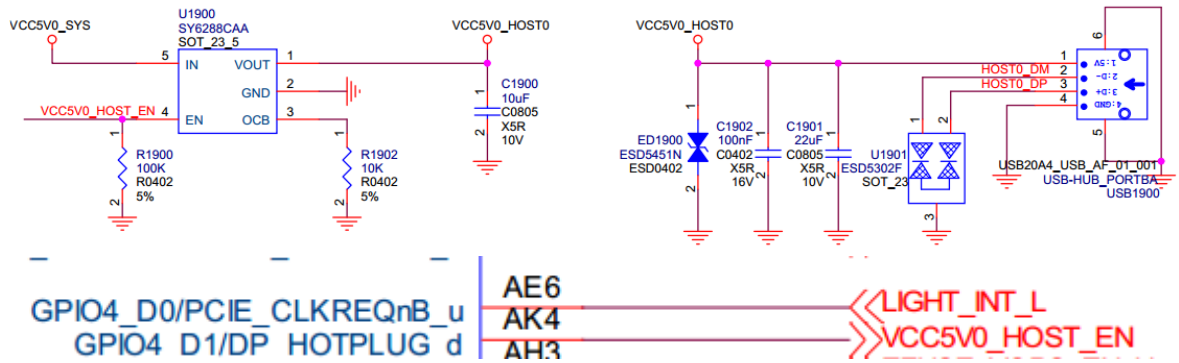

图 2-2 USB 2.0 HOST VBUS GPIO 控制脚

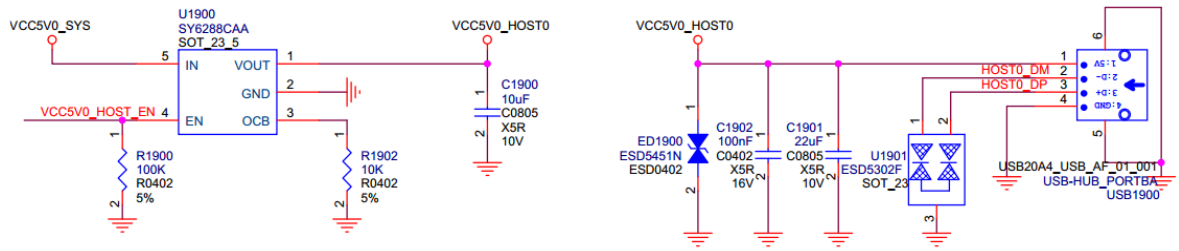

图 2-3 HSIC 控制器硬件电路

HSIC（High Speed Inter Chip）是具有与 USB 2.0 相同的带宽（480Mbps）的芯片间互连接口，如图 2-4 所示，USIC_STROBE 为 240MHz DDR 信号线，USIC_DATA 为数据线，供电电压为 0.9V 和 1.2V，信号传输的标准电压为 1.2V，降低了系统的功耗，最大的走线长度为 10cm（4 英寸）。

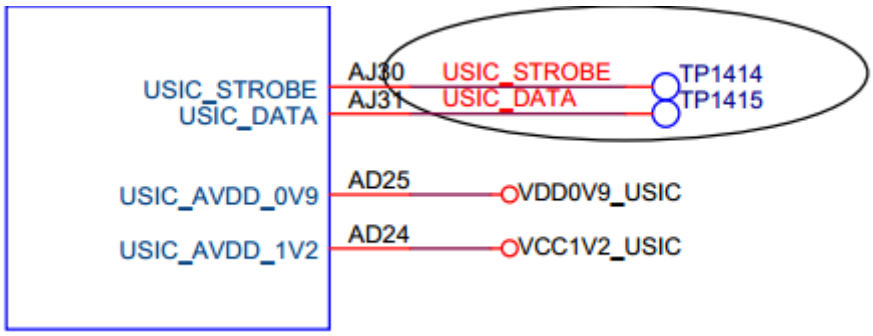

图 2-4 HSIC 硬件电路

2.2 USB OTG 控制器硬件电路

2.2.1 USB 2.0 OTG 控制器硬件电路

RK3399 没有独立的 USB 2.0 OTG 控制器，但有独立的 USB 3.0 OTG 控制器，并且可以向下兼容 USB 2.0 OTG 的完整功能，而 RK3368、RK3366、RK3328、RK3288、RK312X、RK3188、RK30XX 有独立的 USB 2.0 OTG 控制器。

完整的 USB 2.0 OTG 参考电路如图 2-5 ~ 图 2-8 所示：

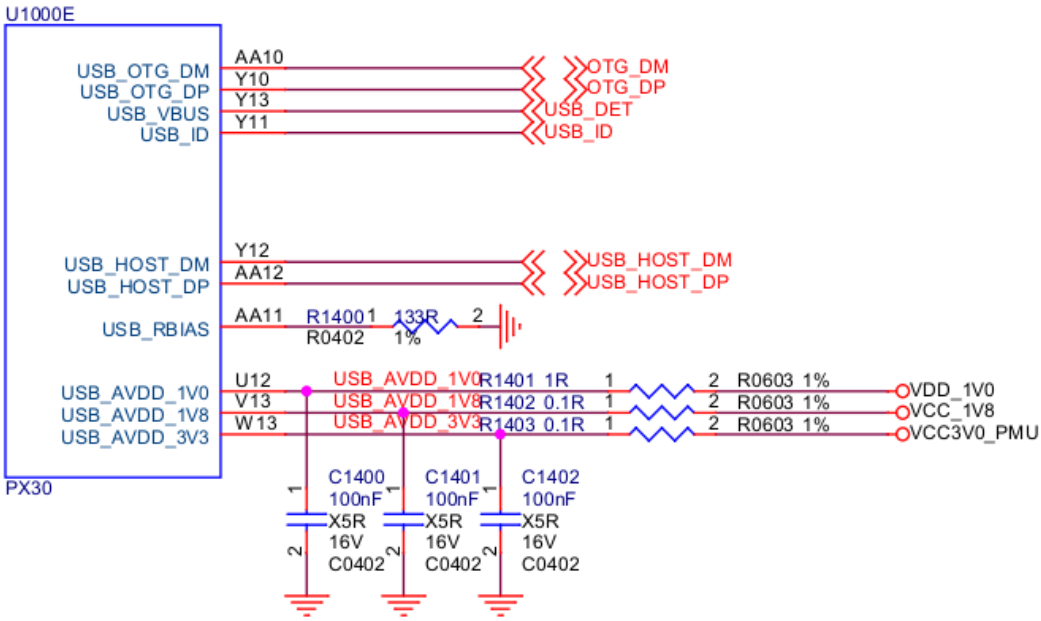

图 2-5 USB 2.0 控制器硬件信号

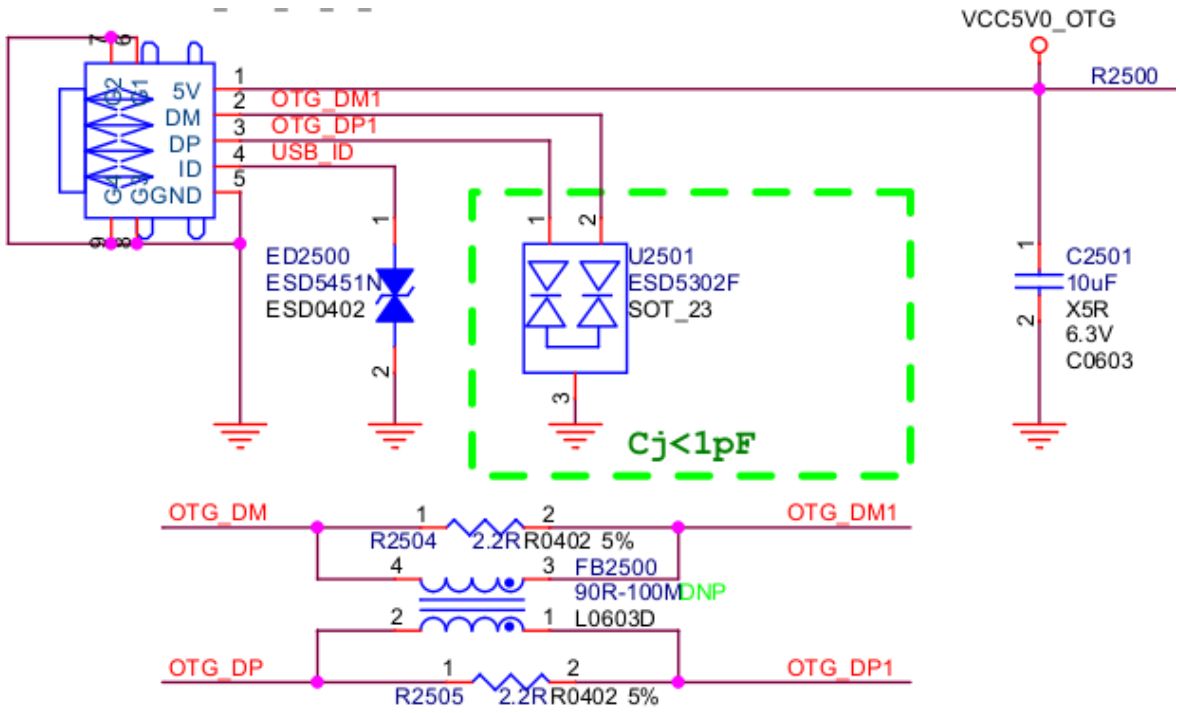

图 2-6 OTG PORT 电路图

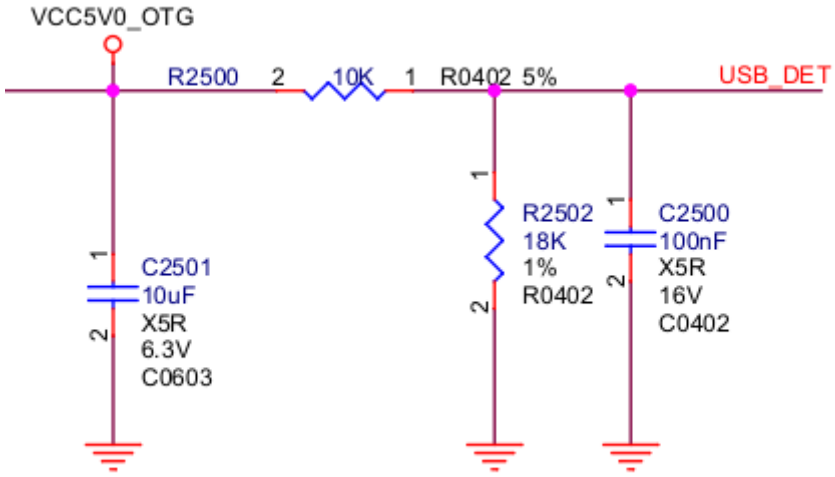

图 2-7 OTG DET 电路图

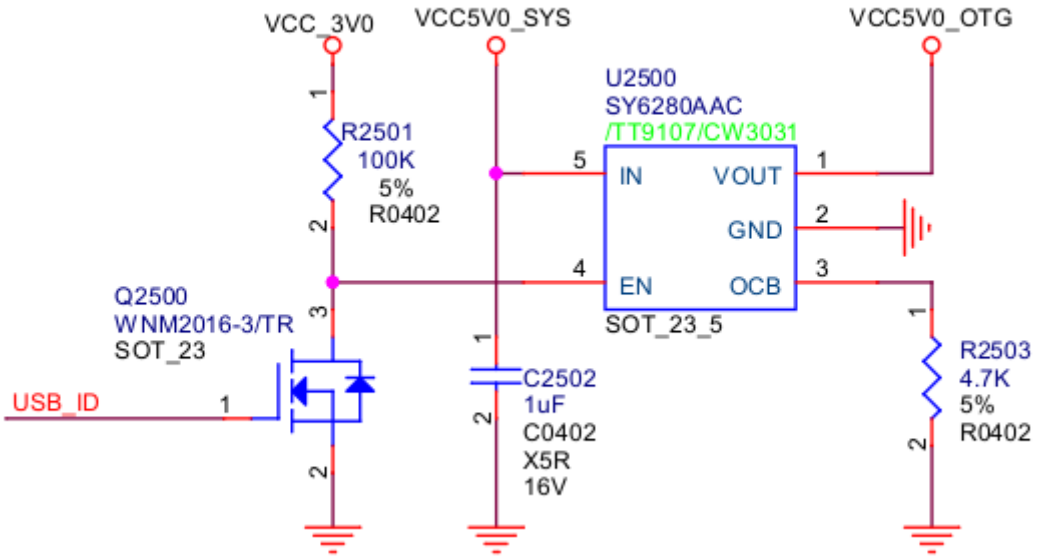

图 2-8 OTG DRV 电路图

- OTG_DP/OTG_DM: USB 差分信号 D+/D-，需要在每根信号线上串联 2.2Ω 的电阻。
- USB_DET: 输入信号，当 OTG 作为 Peripheral mode 时，用于检测 USB 是否连接到 Host（如: PC Host）或者 USB 充电器。默认为低电平 0V。当连接到 Host 或者 USB 充电器时，为高电平 3.0 ~ 3.2 V。
- USB_ID: 输入信号，用于判断切换为 Host mode 或者 Peripheral mode. 默认为高电平 1.8V（芯片内部拉高），OTG 作为 Peripheral mode。当插入 OTG-Host 线缆时，USB_ID 会被拉低到地，USB 驱动会根据 USB_ID 电平变化，将 OTG 切换为 Host mode。
- USB_RBIAS: USB 2.0 PHY 的外部基准电阻。该电阻的阻值会影响 USB 信号的幅值，所以请严格按照 SDK 参考原理图的阻值设计。
- VCC5V0_OTG: 当 OTG 工作于 Peripheral mode 时，VCC5V0_OTG 是 USB_DET 的输入源信号。当 OTG 工作于 Host mode 时，VCC5V0_OTG 输出 VBUS 5V 给 USB 外设。
- USB_AVDD_1V0/USB_AVDD_1V8/USB_AVDD_3V3: USB 2.0 PHY 的供电电源。

2.2.2 USB 3.0 OTG 控制器硬件电路

USB 3.0 OTG 具有 USB 3.0 OTG 功能，且向下兼容 USB 2.0 OTG 功能，最大传输速率为 5Gbps，物理接口为 Type-C，支持正反插。在传输线方面，USB 3.0 支持长达 3 米的四线差分信号线及 11 英寸 PCB。5Gbps 信号在长线缆上采用的是差分信号方式传输，从而避免信号被干扰及减少电磁干扰问题。

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
GND	TX1+	TX1−	VBUS	CC1	D+	D−	SBU1	VBUS	RX2−	RX2+	GND
GND	RX1+	RX1−	VBUS	SBU2	D−	D+	CC2	VBUS	TX2−	TX2+	GND
B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1

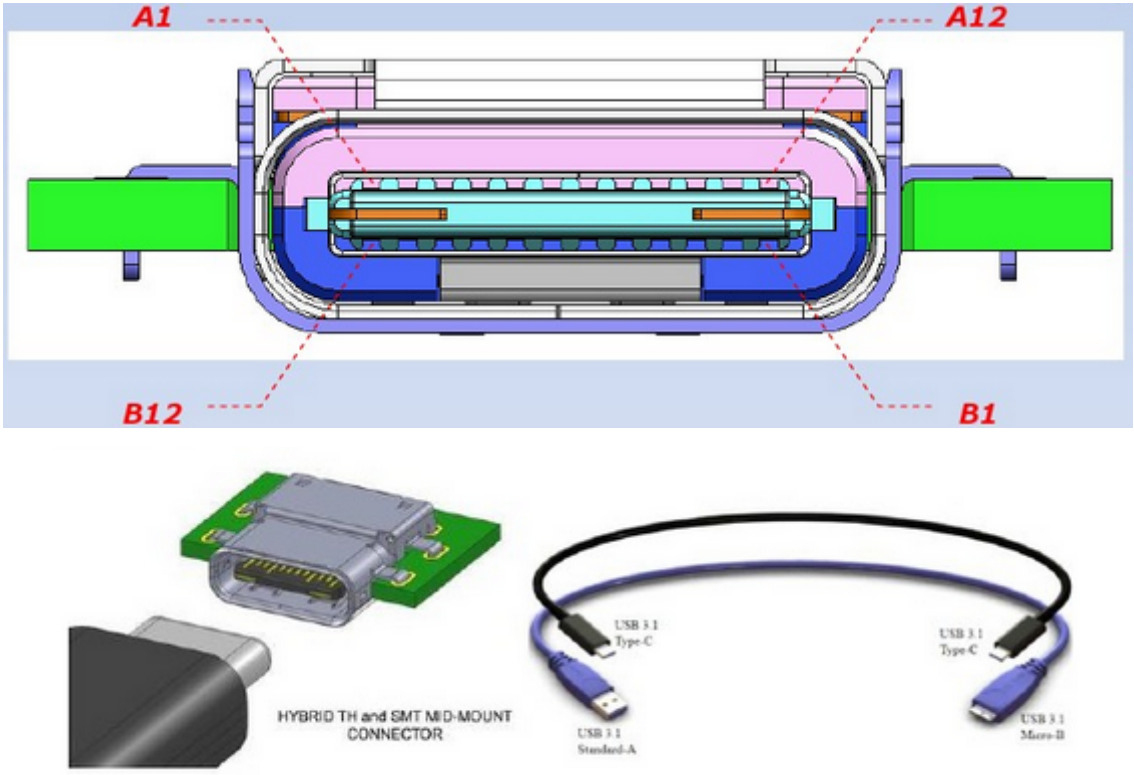

图 2-9 Type-C 接口定义

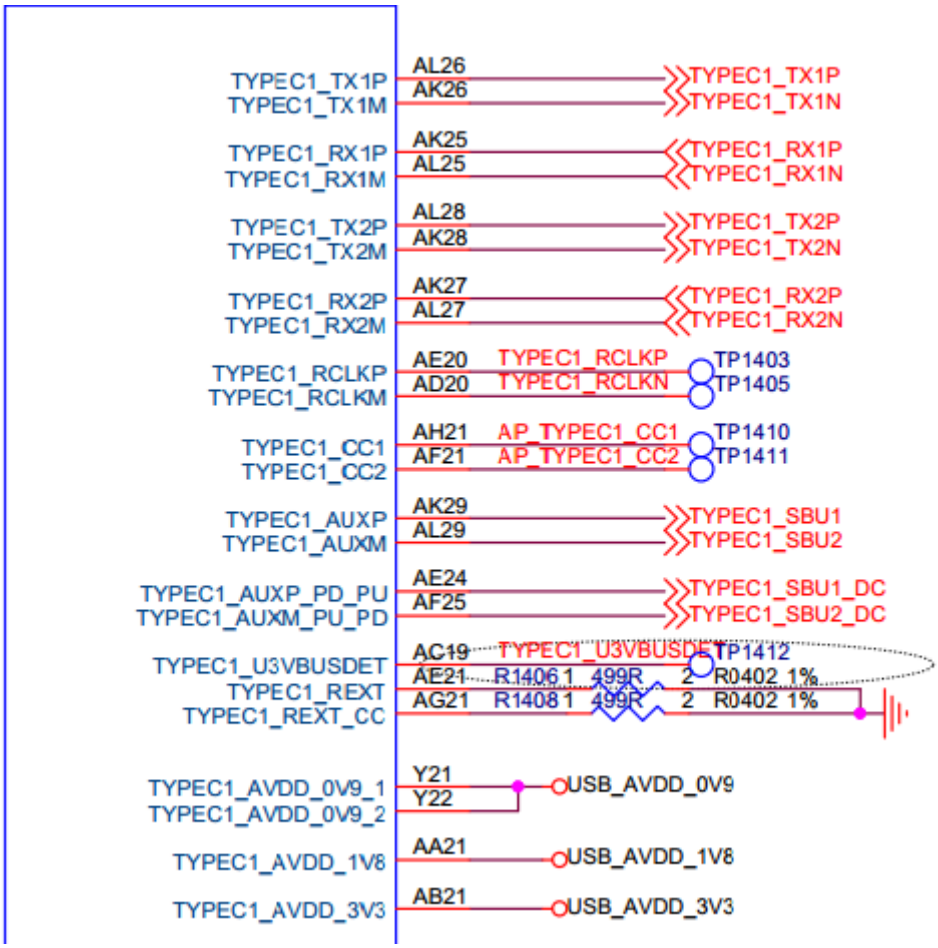

图 2-10 USB3 OTG 控制器 SoC 信号引脚

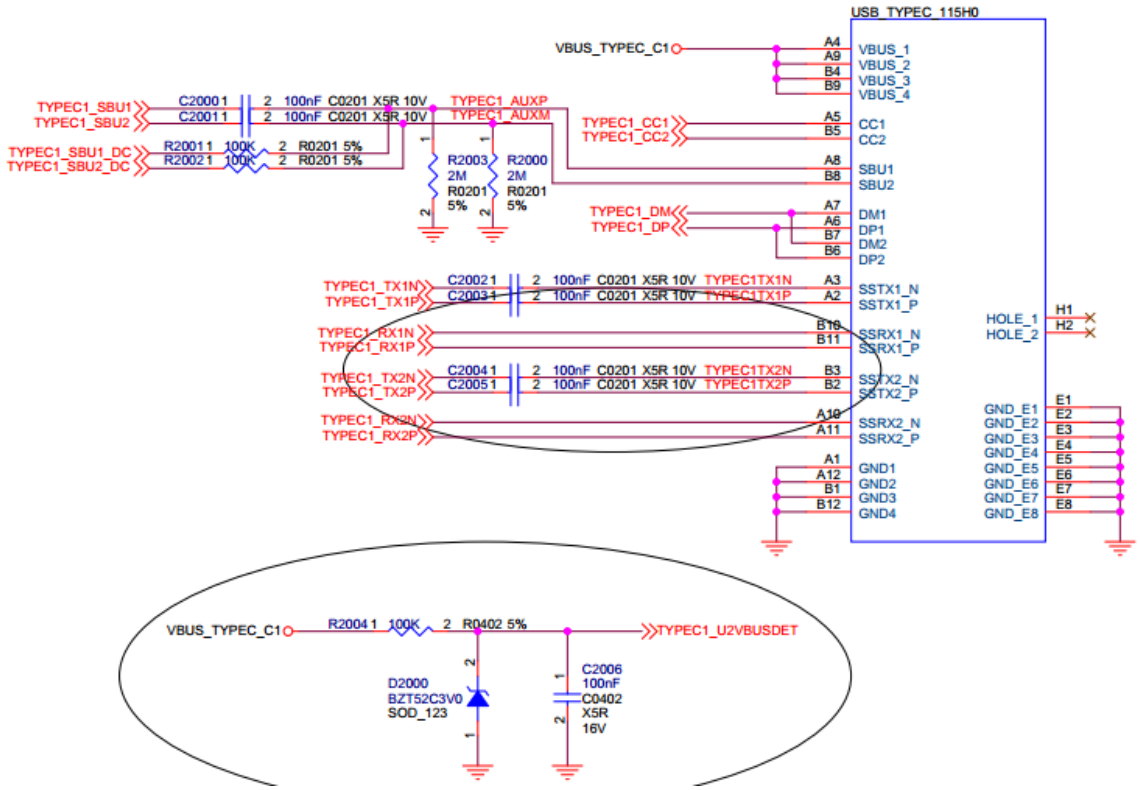

图 2-11 USB3OTG Type-C 接口

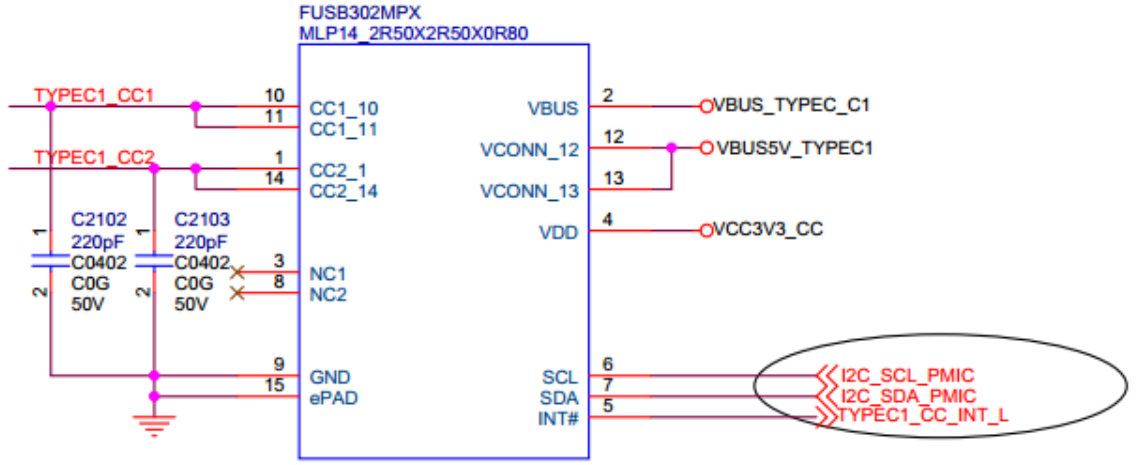

图 2-12 USB3Type-C pd/cc 电路（FUSB302）

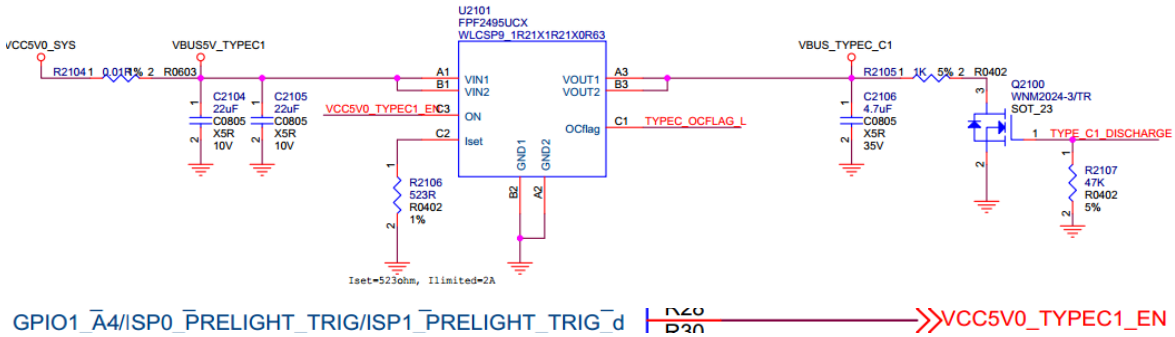

图 2-13 USB3 VBUS 控制电路-1（GPIO 控制电路输出 5V）

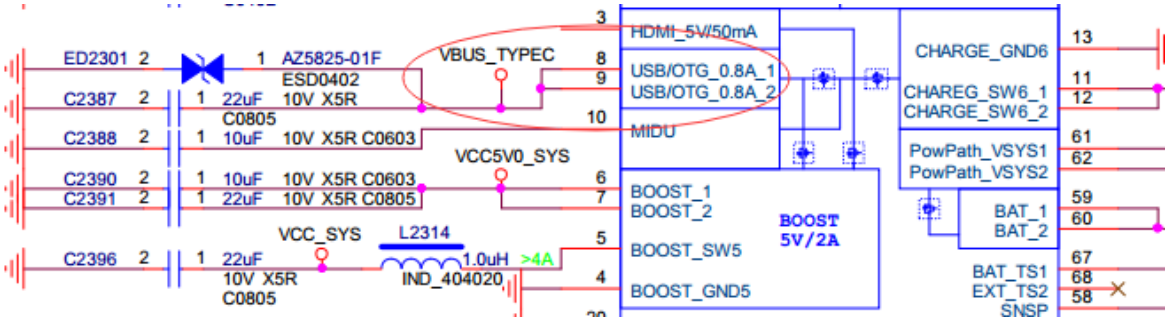

图 2-14 USB3 VBUS 控制电路-2（RK818 控制电路输出 5V）

3 Kernel USB CONFIG

USB 模块的配置及保存和其它内核模块的配置方法一样：

导入默认配置：

```
make ARCH=arm64 rockchip_defconfig
```

选择 Kernel 配置：

```
make ARCH=arm64 menuconfig
```

保存 default 配置：

```
make ARCH=arm64 savedefconfig
```

保存 default 配置，然后用 defconfig 替换 rockchip_defconfig。

3.1 USB PHY CONFIG

USB PHY 模块的配置位于

```
Device Drivers  --->
    PHY Subsystem  --->
        ...
        <*> Rockchip INNO USB2PHY Driver
        ...
        <*> Rockchip TYPEC PHY Driver
        ...
        <*> Rockchip INNO USB 3.0 PHY Driver
```

USB 2.0 PHY 使用的是 Innosilicon IP，所以应选择“Rockchip INNO USB2PHY Driver”。

RK3399 RK3366 USB 3.0 PHY 使用的是 Type-C，所以应选择“Rockchip TYPEC PHY Driver”。

RK3328 USB 3.0 PHY 使用的是 Innosilicon USB 3.0 PHY，所以应选择：

“Rockchip INNO USB 3.0 PHY Driver”。

3.2 USB HOST CONFIG

USB HOST 模块的配置位于

```
Device Drivers  --->
    -*- Support for Host-side USB
    [*] USB support  --->
        ...
        <*> xHCI HCD (USB 3.0) support
        -*- Generic xHCI driver for a platform device
        ...
        <*> EHCI HCD (USB 2.0) support
        [ ]     Root Hub Transaction Translators
        [*]     Improved Transaction Translator scheduling
        <*> Generic EHCI driver for a platform device
        ...
        <*> OHCI HCD (USB 1.1) support
        < >     OHCI support for PCI-bus USB controllers
        <*>     Generic OHCI driver for a platform device
```

必须选上 USB Support 项后才能支持 USB 模块并进行进一步的配置。

如果需要支持 USB HOST，首先需要选上 **<*>Supportfor Host-side USB** 项，然后会现如下的 HOST 相关的配置，其中，USB HOST 1.1 选择 OHCI Driver 配置，USB HOST 2.0 选择 EHCI Driver 配置，USB HOST 3.0 选择 XHCI Driver 配置。

需要注意的是 RK3308 的 default config（rk3308_linux_defconfig）为了裁剪内核，默认是 disable USB HOST，如果要支持 USB 2.0 HOST 和相关 USB 外设，需要先手动 enable EHCI Driver 配置，然后参考 3.5 USB 其它模块配置，使能对应的 USB 外设驱动。

3.3 USB OTG CONFIG

```
Device Drivers  --->
    -*- Support for Host-side USB
    [*] USB support  --->
        ...
        <*> DesignWare USB2 DRD Core Support
                DWC2 Mode Selection (Dual Role mode)
        ...
        <*> DesignWare USB3 DRD Core Support
                DWC3 Mode Selection (Dual Role mode)
```

3.4 USB Gadget CONFIG

USB Gadget 模块的配置位于

```
DeviceDrivers  --->
    [*]USB support  --->
        [*] USB Gadget Support  --->
            ...
            USBGadget Drivers (USB functions configurable through configfs)  --->
                [ ]       Generic serial bulk in/out
```

```
                [*]       Abstract Control Model (CDC ACM)
                [ ]       Object Exchange Model (CDC OBEX)
                [ ]       Network Control Model (CDC NCM)
                [ ]       Ethernet Control Model (CDC ECM)
                [ ]       Ethernet Control Model (CDC ECM) subset
                [*]       RNDIS
                [ ]       Ethernet Emulation Model (EEM)
                [*]       Mass storage
                [ ]       Loopback and sourcesink function (for testing)
                [*]       Function filesystem (FunctionFS)
                [*]       MTP gadget
                [*]         PTP gadget
                [*]       Accessory gadget
                [*]         Audio Source gadget
                [*]       Uevent notification of Gadget state
                [ ]       Audio Class 1.0
                [ ]       Audio Class 2.0
                [*]       MIDI function
                [ ]       HID function
                [ ]       USB Webcam function
                [ ]       Printer function
```

Rockchip 平台默认支持 RNDIS、MTP、PTP、Accessory、ADB、MIDI、Audio 等 Gadget 功能。开发者可以根据实际产品需求，enable 更多的 USB Gadget 功能，但同时，需要修改 init 文件（init.rk30board.usb.rc 和 init.usb.configfs.rc）。

3.5 USB 外设 CONFIG

3.5.1 Mass Storage Class CONFIG

U 盘属于 SCSI 设备，所以在配置 USB 模块之前需要配置 SCSI 选项。

```
Device Drivers --->
    SCSI device support --->
        <*> SCSI device support
        [ ] SCSI: use blk-mq I/O path by default
        [*] legacy /proc/scsi/ support
            *** SCSI support type (disk, tape, CD-ROM) ***
        <*> SCSI disk support
        < > SCSI tape support
        < > SCSI OnStream SC-x0 tape support
        < > SCSI CDROM support
        <*> SCSI generic support
        <*> SCSI media changer support
        [*] Verbose SCSI error reporting (kernel size +=75K)
        [*] SCSI logging facility
        [*] Asynchronous SCSI scanning
            SCSI Transports  --->
        [*] SCSI low-level drivers  --->
        [ ] PCMCIA SCSI adapter support  ----
        [ ] SCSI Device Handlers  ----
```

配置完 SCSI Device Support 后，可以在 USB Support 中找到如下选项，选上即可。

```
Device Driver --->
    [*] USB support  --->
        <*> USB Mass Storage support
```

3.5.2 USB Serial Converter CONFIG

- 支持 USB 3G Modem

  USB 3G Modem 使用的是 USB 转串口，使用时需要选上如下选项：

```
Device Driver --->
    [*] USB support  --->
        <*> USB Serial Converter support  --->
            <*>   USB driver for GSM and CDMA modems
```

此外，USB 3G Modem 还需要使能 PPP 拨号的相关配置项：

```
Device Driver --->
    [*] Network device support  --->
        <*>   PPP (point-to-point protocol) support
            <*>     PPP BSD-Compress compression
            <*>     PPP Deflate compression
            [*]     PPP filtering
            <*>     PPP MPPE compression (encryption)
            [*]     PPP multilink support
            <*>     PPP over Ethernet
            <*>     PPP over L2TP
            <*>     PPP on L2TP Access Concentrator
            <*>     PPP on PPTP Network Server
            <*>     PPP support for async serial ports
            <*>     PPP support for sync tty ports
```

- 支持 PL2303

  如果要使用 PL2303 输出数据到串口，需要选择如下选项，同时需要 disable “USB driver for GSM and CDMA modems”，否则，PL2303 可能会被误识别为 USB 3G modem。

```
Device Driver --->
    [*] USB support  --->
        <*> USB Serial Converter support  --->
            <*> USB Prolific 2303 Single Port Serial Driver
```

- 支持 USB GPS

  如果要支持 USB GPS，如 u-blox 6 - GPS Receiver 设备，需要选择如下选项：

```
Device Drivers  --->
    [*] USB support --->
        [*] USB Modem (CDC ACM) support
```

3.5.3 USB HID CONFIG

USB键鼠的配置选项如下：

```
Device Drivers  --->
    [*] HID support
        [*] USB HID transport layer
        [ ] PID device support
        [*] /dev/hiddev raw HID device support
```

3.5.4 USB Net CONFIG

- USB Bluetooth CONFIG

```
[*] Networking support  --->
    ...
    <*> Bluetooth subsystem support  --->
        Bluetooth device drivers  --->
            ...
            <*> HCI USB driver
            [*]   Broadcom protocol support (NEW)
            [*]   Realtek protocol support (NEW)
            ...
```

- USB Wifi CONFIG

  通常直接使用 Vendor 提供的驱动和配置。

- USB Ethernet CONFIG

```
Device Driver --->
    [*] Network device support  --->
        <*> USB Network Adapters  --->
            <*> USB CATC NetMate-based Ethernet device support
            <*> USB KLSI KL5USB101-based ethernet device support
            <*> USB Pegasus/Pegasus-II based ethernet device support
            <*> USB RTL8150 based ethernet device support
            <*> Realtek RTL8152/RTL8153 Based USB Ethernet Adapters
            < > Microchip LAN78XX Based USB Ethernet Adapters
            <*> Multi-purpose USB Networking Framework
            <*>     ASIX AX88xxx Based USB 2.0 Ethernet Adapters
            <*>     ASIX AX88179/178A USB 3.0/2.0 to Gigabit Ethernet
            -*- CDC Ethernet support (smart devices such as cable modems)
            <*> CDC EEM support
            -*- CDC NCM support
```

```
            < > Huawei NCM embedded AT channel support
            <*> CDC MBIM support
            <*> Davicom DM96xx based USB 10/100 ethernet devices
            < > CoreChip-sz SR9700 based USB 1.1 10/100 ethernet devices
            < > CoreChip-sz SR9800 based USB 2.0 10/100 ethernet devices
            <*> SMSC LAN75XX based USB 2.0 gigabit ethernet devices
            <*> SMSC LAN95XX based USB 2.0 10/100 ethernet devices
            <*> GeneSys GL620USB-A based cables
            <*> NetChip 1080 based cables (Laplink, ...)
            <*> Prolific PL-2301/2302/25A1/27A1 based cables
            <*> MosChip MCS7830 based Ethernet adapters
            <*> Host for RNDIS and ActiveSync devices
            <*> Simple USB Network Links (CDC Ethernet subset)
            [*]     ALi M5632 based 'USB 2.0 Data Link' cables
            [*]     AnchorChips 2720 based cables (Xircom PGUNET, ...)
            [*]     eTEK based host-to-host cables (Advance, Belkin, ...)
            [*]     Embedded ARM Linux links (iPaq, ...)
            [*]     Epson 2888 based firmware (DEVELOPMENT)
            [*]     KT Technology KC2190 based cables (InstaNet)
            <*> Sharp Zaurus (stock ROMs) and compatible
            <*> Conexant CX82310 USB ethernet port
            <*> Samsung Kalmia based LTE USB modem
            <*> QMI WWAN driver for Qualcomm MSM based 3G and LTE modems
            <*> Option USB High Speed Mobile Devices
            <*> Intellon PLC based usb adapter
            <*> Apple iPhone USB Ethernet driver
            <*> USB-to-WWAN Driver for Sierra Wireless modems
            < > LG VL600 modem dongle
            < > QingHeng CH9200 USB ethernet support
```

3.5.5 USB Camera CONFIG

```
Device Driver --->
    <*> Multimedia support  --->
        [*]   Media USB Adapters  --->
            *** Webcam devices ***
            <*> USB Video Class (UVC)
            [*] UVC input events device support
```

3.5.6 USB Audio CONFIG

```
Device Driver --->
    <*> Sound card support  --->
        <*> Advanced Linux Sound Architecture --->
            ...
            [*] USB sound devices   --->
                [*] USB Audio /MIDI driver
```

3.5.7 USB HUB CONFIG

如果要支持 USB HUB，请将“Disable external HUBs”配置选项去掉。

```
Device Drivers --->
    [*] USB support --->
        -*- Support for Host-side USB
        ...
        [*]     Disable external hubs
```

其他有可能用到的 USB 设备还有很多，如 GPS，Printer 等，有可能需要 Vendor 定制的驱动，也有可能是标准的 Class 驱动，如需支持，可直接在网络上搜索 Linux 对该设备支持要做的工作，RK 平台并无特殊要求，可直接参考。

---

4 Device Tree 开发

ARM Linux 内核在 Linux-3.x 内核取消了传统的设备文件而用设备树（DT）取代，因此，现在内核有关硬件描述的信息都需要放入 DT 中配置，下面对涉及到 USB 模块的 DT 开发做以详细说明。

4.1 USB PHY DTS

USB 2.0 PHY 的配置主要包括 PHY 的时钟、中断配置和 VBUS Supply 的配置。

USB 3.0 PHY 的配置主要包括 PHY 的时钟、中断配置、Reset 和 Type-CPHY 状态寄存器地址。

4.1.1 USB 2.0 PHY DTS

USB 2.0 PHY 详细配置可参考内核 Documentation/devicetree/bindings/phy 目录文档说明。

Innosilicon PHY 对应的文档为：

Documentation/devicetree/bindings/phy/phy-rockchip-inno-usb2.txt

具体分为 DTSI 和 DTS 两部分配置，下面以 RK3399 上一个 Host Port 的 PHY 为例说明。

下面所示为 DTSI 的配置，DTSI 主要配置 PHY 的公有属性。

```
grf: syscon@ff770000 {
    compatible = "rockchip,rk3399-grf", "syscon", "simple-mfd";
    reg = <0x0 0xff770000 0x0 0x10000>;
    #address-cells = <1>;
    #size-cells = <1>;

    u2phy0: usb2-phy@e450 {
        compatible = "rockchip,rk3399-usb2phy";
        reg = <0xe450 0x10>;
        clocks = <&cru SCLK_USB2PHY0_REF>;
        clock-names = "phyclk";
        #clock-cells = <0>;
        clock-output-names = "clk_usbphy0_480m";
        status = "disabled";

        u2phy0_host: host-port {
            #phy-cells = <0>;
            interrupts = <GIC_SPI 27 IRQ_TYPE_LEVEL_HIGH 0>;
            interrupt-names = "linestate";
            status = "disabled";
```

```
        };
};
```

首先，USB PHY Driver 中都是在操作 GRF，所以 USB PHY 的节点必须作为 GRF 的一个子节点。

其次，USB PHY 节点中包括 USB PHY 的硬件属性和 PHY port 的硬件属性，其中 PHY 的属性为所有 port 的共有属性，比如 Input 时钟；port 属性主要包括各个 port 所拥有的中断，比如 Linestate 中断、otg-id 中断，otg-bvalid 等。

最后，需要注意的是 port 的名称，HOST 对应的 port 要求命名为“host-port”，OTG 对应的命名为“otg-port”，因为 Driver 中根据这两个名称做不同 port 的初始化。下面所示为 DTS 的配置。

```
u2phy0_host: host-port {
    phy-supply = <&vcc5v0_host>;
    status = "okay";
};
```

DTS 的配置，主要根据不同的产品形态，配置 PHY 的私有属性。目前 SDK-DTS 的配置，主要包括 phy-port 的 Enable 以及 phy-Supply 即 Vbus Supply 的配置。

Vbus supply 的配置有两种方式，一种是配置成 GPIO 形式，直接在驱动中控制 GPIO，进而控制的供给；另外一种是目前内核比较通用的 Regulator 配置。

下面以 Host Vbus 的配置，详细讲述 regulator 的配置方法。其主要分为 Regulator 及 pinctrl 两个节点的配置。

```
vcc5v0_host: vcc5v0-host-regulator {
    compatible = "regulator-fixed";
    enable-active-high;
    gpio = <&gpio4 25 GPIO_ACTIVE_HIGH>;
    pinctrl-names = "default";
    pinctrl-0 = <&host_vbus_drv>;
    regulator-name = "vcc5v0_host";
    regulator-always-on;
};
```

如上面所示，这是一个 vbus-host-regulator 的配置实例，“enable-active-high”属性标识 GPIO 拉高使能；“pinctrl-0 = <&host_vbus_drv>;”Property 代表这个 regulator 所引用的 Pinctrl 中节点的名称，具体 regulator 的配置可参考 LinuxKernel 相关 regulator 的文档。通过对于 USB 模块而言，vbus-regulator 应该在 DTS 中（而不是 DTSI 中）做配置。

```
usb2 {
    host_vbus_drv: host-vbus-drv {
        rockchip,pins =
                <4 25 RK_FUNC_GPIO &pcfg_pull_none>;
    };
};
```

如上面所示，这是 hostvbus-drv 的 pinctrl 属性，rockchip,pins 属性即 GPIO 信息，需要从硬件原理图获知。这个节点作为 Pinctrl 的子节点，通过在 DTSI（而不是 DTS 中）做配置。

4.1.2 USB 3.0 PHY DTS

USB 3.0 PHY 为 Type-C PHY，详细的配置说明请查看：

Documentation/devicetree/bindings/phy/phy-rockchip-typec.txt

Example：

```
tcphy0: phy@ff7c0000 {
    compatible = "rockchip,rk3399-typec-phy";
    reg = <0x0 0xff7c0000 0x0 0x40000>;
    rockchip,grf = <&grf>;
    #phy-cells = <1>;
    clocks = <&cru SCLK_UPHY0_TCPDCORE>,
             <&cru SCLK_UPHY0_TCPDPHY_REF>;
    clock-names = "tcpdcore", "tcpdphy-ref";
    assigned-clocks = <&cru SCLK_UPHY0_TCPDCORE>;
    assigned-clock-rates = <50000000>;
    power-domains = <&power RK3399_PD_TCPD0>;
    resets = <&cru SRST_UPHY0>,
             <&cru SRST_UPHY0_PIPE_L00>,
             <&cru SRST_P_UPHY0_TCPHY>;
    reset-names = "uphy", "uphy-pipe", "uphy-tcphy";
    rockchip,typec-conn-dir = <0xe580 0 16>;
    rockchip,usb3tousb2-en = <0xe580 3 19>;
    rockchip,usb3-host-disable = <0x2434 0 16>;
    rockchip,usb3-host-port = <0x2434 12 28>;
    rockchip,external-psm = <0xe588 14 30>;
    rockchip,pipe-status = <0xe5c0 0 0>;
    rockchip,uphy-dp-sel = <0x6268 19 19>;
    status = "disabled";

    tcphy0_dp: dp-port {
        #phy-cells = <0>;
    };

    tcphy0_usb3: usb3-port {
        #phy-cells = <0>;
    };
};
```

4.2 USB 2.0 Controller DTS

USB 2.0 控制器主要包括 EHCI、OHCI、OTG。其中 EHCI 和 OHCI Rockchip 采用 Linux 内核 Generic 驱动，一般开发时只需要对 DT 作相应配置，即可正常工作。

4.2.1 USB 2.0 HOST Controller DTS

下面所示为一个 EHCI 控制器的典型配置，主要包括 register、interrupts、clocks 的配置。需要注意，EHCI 相关的时钟，通常需要配置 EHCI 控制器和 EHCI/OHCI 仲裁器两个时钟。此外，phys 直接配置对应 phy-port 的名称即可。

```
usb_host0_ehci: usb@fe380000 {
    compatible = "generic-ehci";
    reg = <0x0 0xfe380000 0x0 0x20000>;
    interrupts = <GIC_SPI 26 IRQ_TYPE_LEVEL_HIGH 0>;
    clocks = <&cru HCLK_HOST0>, <&cru HCLK_HOST0_ARB>,
             <&cru SCLK_USBPHY0_480M_SRC>;
    clock-names = "hclk_host0", "hclk_host0_arb", "usbphy0_480m";
    phys = <&u2phy0_host>;
    phy-names = "usb";
    power-domains = <&power RK3399_PD_PERIHP>;
    status = "disabled";
};
```

下面所示为一个 OHCI 控制器的配置，其内容基本跟 EHCI 相同。

```
usb_host0_ohci: usb@fe3a0000 {
    compatible = "generic-ohci";
    reg = <0x0 0xfe3a0000 0x0 0x20000>;
    interrupts = <GIC_SPI 28 IRQ_TYPE_LEVEL_HIGH 0>;
    clocks = <&cru HCLK_HOST0>, <&cru HCLK_HOST0_ARB>,
             <&cru SCLK_USBPHY0_480M_SRC>;
    clock-names = "hclk_host0", "hclk_host0_arb", "usbphy0_480m";
    phys = <&u2phy0_host>;
    phy-names = "usb";
    power-domains = <&power RK3399_PD_PERIHP>;
    status = "disabled";
};
```

4.2.2 USB 2.0 otg Controller DTS

如下所示，为一个 DWC2 控制器的典型配置，主要包括 register、interrupts、clocks 的配置。需要注意，DWC2 相关的时钟，通常需要配置 HCLK 和 PMUCLK 两个时钟。此外，phys 直接配置对应 phy-port 的名称即可。

详细的配置说明请查看：

Documentation/devicetree/bindings/usb/dwc2.txt

```
usb20_otg: usb@ff580000 {
    compatible = "rockchip,rk3328-usb", "rockchip,rk3066-usb",
                 "snps,dwc2";
    reg = <0x0 0xff580000 0x0 0x40000>;
    interrupts = <GIC_SPI 23 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&cru HCLK_OTG>, <&cru HCLK_OTG_PMU>;
    clock-names = "otg", "otg_pmu";
    dr_mode = "otg";
    g-np-tx-fifo-size = <16>;
    g-rx-fifo-size = <275>;
    g-tx-fifo-size = <256 128 128 64 64 32>;
    g-use-dma;
    phys = <&u2phy_otg>;
    phy-names = "usb2-phy";
```

```
    status = "disabled";
};
```

4.3 USB 3.0 Controller DTS

4.3.1 USB 3.0 HOST Controller DTS

USB 3.0 HOST 控制器为 XHCI，集成于 DWC3 OTG IP 中，所以不用单独配置 dts，只需要配置 DWC3，并且设置 DWC3 的 dr_mode 属性为 dr_mode = "otg"或者 dr_mode = "host"，即可以 enable XHCI 控制器。

4.3.2 USB 3.0 OTG Controller DTS

USB 3.0 OTG 的详细配置方法，请查看：

Documentation/devicetree/bindings/usb/dwc3-rockchip.txt

Example：

```
usbdrd3: usb@ff600000 {
    compatible = "rockchip,rk3328-dwc3";
    clocks = <&cru SCLK_USB3OTG_REF>, <&cru SCLK_USB3OTG_SUSPEND>,
             <&cru ACLK_USB3OTG>;
    clock-names = "ref_clk", "suspend_clk",
                  "bus_clk";
    #address-cells = <2>;
    #size-cells = <2>;
    ranges;
    status = "disabled";

    usbdrd_dwc3: dwc3@ff600000 {
        compatible = "snps,dwc3";
        reg = <0x0 0xff600000 0x0 0x100000>;
        interrupts = <GIC_SPI 67 IRQ_TYPE_LEVEL_HIGH>;
        dr_mode = "host";
        phys = <&u3phy_utmi>, <&u3phy_pipe>;
        phy-names = "usb2-phy", "usb3-phy";
        phy_type = "utmi_wide";
        snps,dis_enblslpm_quirk;
        snps,dis-u2-freeclk-exists-quirk;
        snps,dis_u2_susphy_quirk;
        snps,dis-u3-autosuspend-quirk;
        snps,dis_u3_susphy_quirk;
        snps,dis-del-phy-power-chg-quirk;
        snps,tx-ipgap-linecheck-dis-quirk;
        status = "disabled";
    };
};
```

5 驱动开发

目前，EHCI、OHCI 和 DWC3 均采用 Linux Upstream 的代码，因此驱动本身修改的可能性很少，只需要对 DT 做正确配置即可。DWC2 使用内部版和 Upstream 版两个版本，内部版使用的时间较久，稳定性相对较好，Upstream 版主要和 Upstream Mainline 同步更新。USB 2.0 PHY 的驱动也已经 upstream，后续开发仅需对芯片做适配即可。

5.1 USB PHY drivers

5.1.1 USB 2.0 PHY driver

Driver 代码路径：

`drivers/phy/phy-rockchip-inno-usb2.c`

USB 2.0 PHY 采用 Innosilicon IP，SoC 上有两个 USB 2.0 的 PHY，每个 PHY 有两个 port，一个 port 用于支持 USB 2.0 HOST 控制器，另一个 port 用于支持 USB 2.0 OTG 控制器。

对于 Innosilicon IP USB 2.0 PHY 特性，目前已开发并 upstream 了相应的 PHY 驱动代码，针对 host-port，主要涉及到 suspend/resume、sm_work 相关的配置；具体 register 说明可参考代码（drivers/phy/phy-rockchip-inno-usb2.c）中注释。

NOTE： 请参照代码阅读以下内容。

对于新功能的开发，首先应清楚该功能是针对 phy 还是 phy-port，然后对应操作 struct rockchip_usb2phy 和 struct rockchip_usb2phy_port 两个数据结构，第一个用于管理 phy 的成员属性；第二个用于管理 phy-port 的成员属性。

同时，配合上面两个数据结构，还有 struct rockchip_usb2phy_cfg 和 struct rockchip_usb2phy_port_cfg 两个用于配置的数据结构。如下是一个典型的 USB 2.0 host port 的配置。

```
static const struct rockchip_usb2phy_cfg rk3399_phy_cfgs[] = {
    {
        .reg        = 0xe450,
        .num_ports  = 2,
        .phy_tuning = rk3399_usb2phy_tuning,
        .clkout_ctl = { 0xe450, 4, 4, 1, 0 },
        .port_cfgs  = {
            [USB2PHY_PORT_HOST] = {
                .phy_sus    = { 0xe458, 1, 0, 0x2, 0x1 },
                .ls_det_en  = { 0xe3c0, 6, 6, 0, 1 },
                .ls_det_st  = { 0xe3e0, 6, 6, 0, 1 },
                .ls_det_clr = { 0xe3d0, 6, 6, 0, 1 },
                .utmi_ls    = { 0xe2ac, 22, 21, 0, 1 },
                .utmi_hstdet    = { 0xe2ac, 23, 23, 0, 1 }
            }
        },
```

下面是 USB 2.0 otg-port phy 配置参考，port_cfgs 主要用于插拔和 otg mode 检测，chg_det 主要用于充电类型检测：

```
static const struct rockchip_usb2phy_cfg rk3399_phy_cfgs[] = {
    {
        .reg        = 0xe450,
        .num_ports  = 2,
```

```
        .phy_tuning = rk3399_usb2phy_tuning,
        .clkout_ctl = { 0xe450, 4, 4, 1, 0 },
        .port_cfgs  = {
            [USB2PHY_PORT_OTG] = {
                .phy_sus = { 0xe454, 8, 0, 0x052, 0x1d1 },
                .bvalid_det_en  = { 0xe3c0, 3, 3, 0, 1 },
                .bvalid_det_st  = { 0xe3e0, 3, 3, 0, 1 },
                .bvalid_det_clr = { 0xe3d0, 3, 3, 0, 1 },
                .bypass_dm_en   = { 0xe450, 2, 2, 0, 1 },
                .bypass_sel = { 0xe450, 3, 3, 0, 1 },
                .idfall_det_en  = { 0xe3c0, 5, 5, 0, 1 },
                .idfall_det_st  = { 0xe3e0, 5, 5, 0, 1 },
                .idfall_det_clr = { 0xe3d0, 5, 5, 0, 1 },
                .idrise_det_en  = { 0xe3c0, 4, 4, 0, 1 },
                .idrise_det_st  = { 0xe3e0, 4, 4, 0, 1 },
                .idrise_det_clr = { 0xe3d0, 4, 4, 0, 1 },
                .ls_det_en  = { 0xe3c0, 2, 2, 0, 1 },
                .ls_det_st  = { 0xe3e0, 2, 2, 0, 1 },
                .ls_det_clr = { 0xe3d0, 2, 2, 0, 1 },
                .utmi_avalid    = { 0xe2ac, 7, 7, 0, 1 },
                .utmi_bvalid    = { 0xe2ac, 12, 12, 0, 1 },
                .utmi_iddig = { 0xe2ac, 8, 8, 0, 1 },
                .utmi_ls    = { 0xe2ac, 14, 13, 0, 1 },
                .vbus_det_en    = { 0x449c, 15, 15, 1, 0 },
            },
        },
        .chg_det = {
            .opmode     = { 0xe454, 3, 0, 5, 1 },
            .cp_det     = { 0xe2ac, 2, 2, 0, 1 },
            .dcp_det    = { 0xe2ac, 1, 1, 0, 1 },
            .dp_det     = { 0xe2ac, 0, 0, 0, 1 },
            .idm_sink_en    = { 0xe450, 8, 8, 0, 1 },
            .idp_sink_en    = { 0xe450, 7, 7, 0, 1 },
            .idp_src_en = { 0xe450, 9, 9, 0, 1 },
            .rdm_pdwn_en    = { 0xe450, 10, 10, 0, 1 },
            .vdm_src_en = { 0xe450, 12, 12, 0, 1 },
            .vdp_src_en = { 0xe450, 11, 11, 0, 1 },
        },
    },
```

USB 2.0 PHY 驱动在 sys/devices/platform/下建立了一个 otg_mode 节点，用于切换 dwc2 的 host 和 peripheral mode，以 RK3326 SoC 为例：

```
rk3326_evb:/sys/devices/platform/ff2c0000.syscon/ff2c0000.syscon:usb2-phy@100 #
ls
driver          extcon   of_node  phy   subsystem
driver_override modalias otg_mode power uevent
```

otg_mode: Show & store the current value of otg mode for otg port

For example, force mode for RK3326 board USB:

1. Force host mode

```
echo host > /sys/devices/platform/<u2phy dev name>/otg_mode
```

2. Force peripheral mode

```
echo peripheral > /sys/devices/platform/<u2phy dev name>/otg_mode
```

3. Force otg mode

```
echo otg > /sys/devices/platform/<u2phy dev name>/otg_mode
```

Legacy Usage:

1. Force host mode

```
echo 1 > /sys/devices/platform/<u2phy dev name>/otg_mode
```

2. Force peripheral mode

```
echo 2 > /sys/devices/platform/<u2phy dev name>/otg_mode
```

3. Force otg mode

```
echo 0 > /sys/devices/platform/<u2phy dev name>/otg_mode
```

5.1.2 USB 3.0 PHY driver

Driver 代码有两个版本，3228H、3328 的 USB 3.0 PHY 使用 Innosilicon IP。

代码路径：

`drivers/phy/rockchip/phy-rockchip-inno-usb3.c`

Innosilicon IP 只包含一个 USB 3.0 PHY 端口，支持 USB 3.0 底层数据传输、LFPS 通信和物理信号检测等功能。

对于 Innosilicon IP USB 3.0 PHY 特性，目前已开发了相应的 PHY 驱动代码，暂时还未 upstream，对端口的控制，主要涉及 suspend/resume、power 和 detective 等，具体 register 说明可参考代码（drivers/phy/phy-rockchip-inno-usb3.c）中注释。

NOTE： 请参照代码阅读以下内容。

对于新功能的开发，主要操作 struct rockchip_u3phy 和 struct rockchip_u3phy_port 两个数据结构，第一个用于管理 phy 的成员属性；第二个用于管理 phy-port 的成员属性。

同时，配合上面两个数据结构，还有 struct rockchip_u3phy_cfg 、struct rockchip_u3phy_grf_cfg 和 struct rockchip_u3phy_apbcfg 三个用于配置的数据结构。struct rockchip_u3phy_apbcfg 主要用于保存 USB 物理信号 tuning 的相关参数，在 tuning 函数中直接赋值给相关寄存器；struct rockchip_u3phy_cfg 和 struct rockchip_u2phy_grf_cfg 则是用于 USB 底层协议相关的功能性相关属性配置，如下是一个典型的 rockchip_u3phy_cfg 的配置。

```
static const struct rockchip_u3phy_cfg rk3328_u3phy_cfgs[] = {
    {
        .reg        = 0xff470000,
        .grfcfg     = {
```

```
            .um_suspend = { 0x0004, 15, 0, 0x1452, 0x15d1 },
            .u2_only_ctrl   = { 0x0020, 15, 15, 0, 1 },
            .um_ls      = { 0x0030, 5, 4, 0, 1 },
            .um_hstdct  = { 0x0030, 7, 7, 0, 1 },
            .ls_det_en  = { 0x0040, 0, 0, 0, 1 },
            .ls_det_st  = { 0x0044, 0, 0, 0, 1 },
            .pp_pwr_st  = { 0x0034, 14, 13, 0, 0},
            .pp_pwr_en  = { {0x0020, 14, 0, 0x0014, 0x0005},
                        {0x0020, 14, 0, 0x0014, 0x000d},
                        {0x0020, 14, 0, 0x0014, 0x0015},
                        {0x0020, 14, 0, 0x0014, 0x001d} },
            .u3_disable = { 0x04c4, 15, 0, 0x1100, 0x101},
        },
        .phy_pipe_power = rk3328_u3phy_pipe_power,
        .phy_tuning = rk3328_u3phy_tuning,
        .phy_cp_test    = rk322xh_u3phy_cp_test_enable,
    },
    { /* sentinel */ }
};
```

其余芯片的 USB 3.0 PHY 为 Cadence IP，

代码路径：

`drivers/phy/rockchip/phy-rockchip-typec.c`

Cadence IP 只有一个端口，支持 USB3.1、DisplayPort1.3，可以工作在 USB、DisplayPort、USB+DisplayPort 三种模式，DisplayPort 部分在显示部分的文档描述，这里我们只关心 USB 部分，Cadence IP 同样支持 USB 3.0 底层数据传输、LFPS 通信和物理信号检测等功能。

对于 Cadence IP USB 3.0 PHY 特性，目前已开发并 upstream 了相应的 PHY 驱动代码，针对端口的控制，主要涉及到 suspend/resume、power、DP USB 模式切换等，具体 register 说明可参考驱动文件（`drivers/phy/rockchip/phy-rockchip-inno-usb3.c`）中注释。

对于新功能的开发，主要操作 struct rockchip_typec 和 stuct rockchip_usb3phy_port_cfg 两个数据结构，第一个用于管理 phy 的成员属性；第二个用于管理 phy-port 的成员属性。
rockchip_usb3phy_port_cfg 的配置参数在驱动初始化时通过 DTS 传入，运行过程中通过调用 tcphy_get_mode 获取实际连接状态进行实时配置。

除此之外驱动中新建了 struct phy_reg usb3_pll_cfg 用于保存 PLL 相关的寄存器配置，常见的配置参数如下：

```
struct phy_reg usb3_pll_cfg[] = {
    { 0xf0,     CMN_PLL0_VCOCAL_INIT },
    { 0x18,     CMN_PLL0_VCOCAL_ITER },
    { 0xd0,     CMN_PLL0_INTDIV },
    { 0x4a4a,   CMN_PLL0_FRACDIV },
    { 0x34,     CMN_PLL0_HIGH_THR },
    { 0x1ee,    CMN_PLL0_SS_CTRL1 },
    { 0x7f03,   CMN_PLL0_SS_CTRL2 },
    { 0x20,     CMN_PLL0_DSM_DIAG },
    { 0,        CMN_DIAG_PLL0_OVRD },
    { 0,        CMN_DIAG_PLL0_FBH_OVRD },
    { 0,        CMN_DIAG_PLL0_FBL_OVRD },
    { 0x7,      CMN_DIAG_PLL0_V2I_TUNE },
    { 0x45,     CMN_DIAG_PLL0_CP_TUNE },
```

```
    { 0x8,      CMN_DIAG_PLL0_LF_PROG },
};
```

5.2 USB Controller drivers

5.2.1 USB 3.0 OTG drivers

5.2.1.1 USB 3.0 OTG 驱动说明

USB 3.0 OTG 采用 DWC3 控制器（内嵌 xHCI 控制器），支持 Peripheral 和 Host mode，并且可以动态切换 mode。

Driver 代码路径：
`drivers/usb/dwc3/*`（USB3.0 OTG Global core 和 Peripheral 相关驱动）
`drivers/usb/host/xhci*`（USB3.0 Host 相关驱动）

drivers/usb/dwc3 目录下的文件主要包括 DWC3 控制器 core 驱动、Device 驱动以及 Vendor Special 驱动。其中，文件名带 Vendor 名字的为 Vendor Special 驱动，Rcockchip 的驱动文件名为 dwc3-rockchip.c，core.c 主要负责 DWC3 控制器 core 的初始化，gadget.c 是 DWC3 Device 驱动文件，主要实现控制器 Device 初始化、中断处理和数据传输等功能。

重要的结构体：

```
static const struct usb_gadget_ops dwc3_gadget_ops = {
    .get_frame      = dwc3_gadget_get_frame,
    .wakeup         = dwc3_gadget_wakeup,
    .set_selfpowered    = dwc3_gadget_set_selfpowered,
    .pullup         = dwc3_gadget_pullup,
    .udc_start      = dwc3_gadget_start,
    .udc_stop       = dwc3_gadget_stop,
};
```

drivers/usb/host 目录下文件名含有“xhci”的为 XHCI 控制器相关驱动，其中 xhci-plat.c 是注册驱动的初始化文件，xhci.c 是控制器基础文件，实现 USB core 层 HCD 接口和 USB 传输控制的相关操作，xhci-ring.c 是控制器数据结构 TRB 以及传输机制相关的文件，实现具体的传输功能，xhci-hub.c 是控制器自带的 USB 3.0 root Hub 驱动。

重要的结构体：

```
static const struct hc_driver xhci_hc_driver = {
    .description =      "xhci-hcd",
    .product_desc =     "xHCI Host Controller",
    .hcd_priv_size =    sizeof(struct xhci_hcd *),
    /*
     * generic hardware linkage
     */
    .irq =          xhci_irq,
    .flags =        HCD_MEMORY | HCD_USB3 | HCD_SHARED,
    /*
     * basic lifecycle operations
     */
    .reset =        NULL, /* set in xhci_init_driver() */
    .start =        xhci_run,
```

```
    .stop =         xhci_stop,
    .shutdown =     xhci_shutdown,
    /*
     * managing i/o requests and associated device resources
     */
    .urb_enqueue =      xhci_urb_enqueue,
    .urb_dequeue =      xhci_urb_dequeue,
    .alloc_dev =        xhci_alloc_dev,
    .free_dev =     xhci_free_dev,
    .alloc_streams =    xhci_alloc_streams,
    .free_streams =     xhci_free_streams,
    .add_endpoint =     xhci_add_endpoint,
    .drop_endpoint =    xhci_drop_endpoint,
    .endpoint_reset =   xhci_endpoint_reset,
    .check_bandwidth =  xhci_check_bandwidth,
    .reset_bandwidth =  xhci_reset_bandwidth,
    .address_device =   xhci_address_device,
    .enable_device =    xhci_enable_device,
    .update_hub_device =    xhci_update_hub_device,
    .reset_device =     xhci_discover_or_reset_device,
    /*
     * scheduling support
     */
    .get_frame_number = xhci_get_frame,
    /*
     * root hub support
     */
    .hub_control =      xhci_hub_control,
    .hub_status_data =  xhci_hub_status_data,
    .bus_suspend =      xhci_bus_suspend,
    .bus_resume =       xhci_bus_resume,
    /*
     * call back when device connected and addressed
     */
    .update_device =        xhci_update_device,
    .set_usb2_hw_lpm =  xhci_set_usb2_hardware_lpm,
    .enable_usb3_lpm_timeout =  xhci_enable_usb3_lpm_timeout,
    .disable_usb3_lpm_timeout = xhci_disable_usb3_lpm_timeout,
    .find_raw_port_number = xhci_find_raw_port_number,
};
```

5.2.1.2 USB 3.0 OTG 调试接口

以 RK3399 SoC USB 3.0 OTG0 为例:

```
rk3399_box:/sys/kernel/debug/usb@fe800000 # ls
host_testmode     rk_usb_force_mode

rk3399_box:/sys/kernel/debug/fe800000.dwc3 # ls
ep0in  ep1in  ep2in  ep3in  ep4in  ep5in  ep6in      mode    testmode
ep0out ep1out ep2out ep3out ep4out ep5out link_state regdump

rk3399_box:/sys/kernel/debug/fe800000.dwc3/ep0in # ls
descriptor_fetch_queue rx_info_queue    trb_ring
event_queue            rx_request_queue tx_fifo_queue
rx_fifo_queue          transfer_type    tx_request_queue
```

- **host_testmode**

  Enables USB2/USB3 HOST Test Modes (U2: J, K SE0 NAK, Test_packet,Force Enable; U3: Compliance mode)

  For example， set testmodes for RK3399 board USB:

1. set Test packet for Type-C0 USB2 HOST:

```
echo test_packet > /sys/kernel/debug/usb@fe800000/host_testmode
```

2. set compliance mode for Type-C0 USB3 HOST normal orientation:

```
echo test_u3 > /sys/kernel/debug/usb@fe800000/host_testmode
```

3. set compliance mode for Type-C0 USB3 HOST flip orientation:

```
echo test_flip_u3 > /sys/kernel/debug/usb@fe800000/host_testmode
```

4. check the testmode status:

```
cat /sys/kernel/debug/usb@fe800000/host_testmode
```

The log maybe like this:
U2: test_packet /* means that U2 in test mode /
U3: compliance mode / means that U3 in test mode */

- **rk_usb_force_mode**

  force dr_mode of DWC3 controller

  (**Note: the dr_mode of DTS must be "otg" and extcon of DTS must be config to null**).

  For example， set force mode for RK3399 board USB:

1. Force host mode:

```
echo host > sys/kernel/debug/usb@fe800000/rk_usb_force_mode
```

2. Force peripheral mode:

```
echo peripheral > sys/kernel/debug/usb@fe800000/rk_usb_force_mode
```

3. Force otg mode:

```
echo otg > sys/kernel/debug/usb@fe800000/rk_usb_force_mode
```

- **ep*in/out:** Directory of EP debug files
- **mode:** dr_mode read or store
- **link_state:** Link state read or store
- **regdump:** Dump registers of DWC3
- **descriptor_fetch_queue:** Dump the available DescFetchQ space of EP
- **rx_info_queue:** Dump the available RXInfoQ space of EP
- **trb_ring:** Dump the TRB pool of EP
- **event_queue:** Dump the avaliable EventQ space of EP
- **rx_request_queue:** Dump the avaliable RxReqQ space of EP
- **tx_fifo_queue:** Dump the avaliable TxFIFO space of EP
- **rx_fifo_queue:** Dump the avaliable RxFIFO space of EP
- **transfer_type:** Print the Transfer Type of EP
- **tx_request_queue:** Dump the abaliable TxReqQ space of EP

  Note：

  XHCI 驱动没有文件系统下的 debug 接口，但是可以通过在内核编译的 config 文件中增加 CONFIG_DYNAMIC_DEBUG，打开 debug 信息。

5.2.2 USB 2.0 OTG drivers

5.2.2.1 USB 2.0 OTG 驱动说明

USB 2.0 OTG 使用的是 Synopsys 方案，即使用 DWC2 控制器同时实现 Host 和 Device 功能，DWC2 控制器通过检测 OTG 口上 ID 脚的电平切换模式，ID 脚的电平变化触发控制器 ID 脚中断，然后由软件切换到对应模式。

USB2.0 OTG 使用两种驱动版本，一个是 upstream 版，驱动在 dwc2 目录下，主要从 upstream 开源项目更新代码，另一个是内部版，驱动在 dwc_otg_310 目录下，由 RK 内部自行维护，还未 upstream。

Driver 代码路径：

upstream 版：`drivers/usb/dwc2/*`

内部版：`drivers/usb/dwc_otg_310/*`

drivers/usb/dwc2 目录下的文件可以分成三类，一类是文件名包含“hcd”的 Host 相关驱动，负责 Host 初始化、Host 中断处理和 Host 数据传输操作，一类是 Device 相关文件 gadget.c，负责 Device 初始化、中断处理、数据传输的工作，其余的文件是控制器 core 层和引导驱动，包括通用接口、通用中断处理和控制器初始化等功能。

drivers/usb/dwc_otg_310 目录下有多个 dwc_otg 开头的文件，分成三类，一类是文件名包含“hcd”的 host 相关驱动文件，主要负责 Host 初始化、Host 中断处理和 Host 数据传输相关操作，一类是文件名包含“pcd”的 Device 相关驱动，主要负责 Device 初始化、Device 中断处理和 Device 数据传输相关操作，还有一类是 Host Device 通用驱动，主要包括通用属性配置、通用中断处理、通用控制器接口、控制器初始化以及“usbdev”开头的 PHY 相关的设置文件。

5.2.2.2 USB 2.0 OTG 调试接口

1. dwc2 控制器驱动调试接口

   以 RK3328 SoC 为例：

```
rk3328_box:/sys/kernel/debug/ff580000.usb # ls
ep0   ep2out ep4out ep6out ep8in  ep9in  fifo    state
ep1in ep3in  ep5in  ep7in  ep8out ep9out regdump testmode
```

   ep*in/out: Shows the state of the given endpoint (one is registered for each available).

   fifo: Show the FIFO information for the overall fifo and all the periodic transmission FIFOs.

   state: shows the overall state of the hardware and some general information about each of the endpoints available to the system.

   regdump: Gets register values of core.

   testmode: Modify the current usb test mode.

2. dwc_otg_310 控制器驱动调试接口

   以 RK3328 SoC 为例：

```
rk3328_box:/sys/devices/platform/ff580000.usb # ls
busconnected  fr_interval gsnpsid   modalias       regoffset     uevent
buspower      gadget      guid      mode           regvalue      usb5
bussuspend    ggpio       gusbcfg   mode_ch_tim_en remote_wakeup wr_reg_test
devspeed      gnptxfsiz   hcd_frrem pools          spramdump
disconnect_us gotgctl     hcddump   power          subsystem
driver        gpvndctl    hprt0     rd_reg_test    test_sq
enumspeed     grxfsiz     hptxfsiz  regdump        udc

rk3328_box:/sys/devices/platform/ff580000.usb/driver # ls
bind       dwc_otg_conn_en force_usb_mode uevent vbus_status
debuglevel ff580000.usb    op_state       unbind versio
```

   busconnected: Gets or sets the Core Control Status Register.

   fr_interval: On read, shows the value of HFIR Frame Interval. On write, dynamically reload HFIR register during runtime. The application can write a value to this register only after the Port Enable bit of the Host Port Control and Status register (HPRT.PrtEnaPort) has been set.

   gsnpsid: Gets the value of the Synopsys ID Regester.

   regoffset: Sets the register offset for the next Register Access.

   buspower: Gets or sets the Power State of the bus (0 - Off or 1 - On).

   guid: Gets or sets the value of the User ID Register.

   regvalue: Gets or sets the value of the register at the offset in the regoffset attribute.

   bussuspend: Suspends the USB bus.

   ggpio: Gets the value in the lower 16-bits of the General Purpose IO Register or sets the upper 16 bits.

   gusbcfg: Gets or sets the Core USB Configuration Register.

mode_ch_tim_en: This bit is used to enable or disable the host core to wait for 200 PHY clock cycles at the end of Resume to change the opmode signal to the PHY to 00 after Suspend or LPM.

remote_wakeup: On read, shows the status of Remote Wakeup. On write, initiates a remote wakeup of the host. When bit 0 is 1 and Remote Wakeup is enabled, the Remote Wakeup signalling bit in the Device Control Register is set for 1 milli-second.

wr_reg_test: Displays the time required to write the GNPTXFSIZ register many times (the output shows the number of times the register is written).

devspeed: Gets or sets the device speed setting in the DCFG register.

gnptxfsiz: Gets or sets the non-periodic Transmit Size Register.

spramdump: Dumps the contents of core registers.

disconnect_us: On read, shows the status of disconnect_device_us. On write, sets disconnect_us which causes soft disconnect for 100us. Applicable only for device mode of operation.

gotgctl: Gets or sets the Core Control Status Register.

hcddump: Dumps the current HCD state.

gpvndctl: Gets or sets the PHY Vendor Control Register.

hprt0: Gets or sets the value in the Host Port Control and Status Register.

rd_reg_test: Displays the time required to read the GNPTXFSIZ register many times (the output shows the number of times the register is read).

test_sq: Gets or sets the usage of usb controler test_sq attribute.

enumspeed: Gets the device enumeration Speed.

grxfsiz: Gets or sets the Receive FIFO Size Register.

hptxfsiz: Gets the value of the Host Periodic Transmit FIFO.

regdump: Dumps the contents of core registers.

wc_otg_conn_en: Enable or disable connect to PC in device mode.

force_usb_mode: Force work mode of core (0 - Normal, 1 - Host, 2 - Device).

vbus_status: Gets the Voltage of VBUS.

debuglevel: Gets or sets the driver Debug Level.

op_state: Gets or sets the operational State, during transations (a_host>>a_peripherial and b_device=>b_host) this may not match the core but allows the software to determine transitions.

version: Gets the Driver Version.

5.2.3 USB 2.0 HOST drivers

5.2.3.1 USB 2.0 HOST 驱动说明

Driver 代码路径：

`drivers/usb/host/ehci*` （USB 2.0 Host 驱动）

`drivers/usb/host/ohci*` （USB1.1/1.0 Host 驱动）

EHCI 控制器使用的是 Upstream 版驱动，host 目录下文件名包含“ehci”的为 EHCI 控制器相关文件，其中文件名包含厂商名字的为产商引导文件，目前 RK 没有使用厂商引导文件，而是使用通用引导文件 ehci-platform.c 进行驱动的加载与初始化。ehci-hcd.c 负责控制整个控制器，实现 USB core 层 HCD 控制器接口，ehci-mem.c 与 ehci-sched.c 是控制器数据传输结构与传输调度的相关代码，ehci-hub.c 是 EHCI 控制器 root hub 驱动代码。

重要的结构体：

```
static const struct hc_driver ehci_hc_driver = {
    .description =      hcd_name,
    .product_desc =     "EHCI Host Controller",
    .hcd_priv_size =    sizeof(struct ehci_hcd),
    /*
     * generic hardware linkage
     */
    .irq =          ehci_irq,
    .flags =        HCD_MEMORY | HCD_USB2 | HCD_BH,
    /*
     * basic lifecycle operations
     */
    .reset =        ehci_setup,
    .start =        ehci_run,
    .stop =         ehci_stop,
    .shutdown =     ehci_shutdown,
    /*
     * managing i/o requests and associated device resources
     */
    .urb_enqueue =      ehci_urb_enqueue,
    .urb_dequeue =      ehci_urb_dequeue,
    .endpoint_disable = ehci_endpoint_disable,
    .endpoint_reset =   ehci_endpoint_reset,
    .clear_tt_buffer_complete = ehci_clear_tt_buffer_complete,
    /*
     * scheduling support
     */
    .get_frame_number = ehci_get_frame,
    /*
     * root hub support
     */
    .hub_status_data =  ehci_hub_status_data,
    .hub_control =      ehci_hub_control,
    .bus_suspend =      ehci_bus_suspend,
    .bus_resume =       ehci_bus_resume,
    .relinquish_port =  ehci_relinquish_port,
    .port_handed_over = ehci_port_handed_over,
    /*
     * device support
     */
    .free_dev =     ehci_remove_device,
};
```

OHCI 控制器使用的也是 Upstream 版驱动，host 目录下文件名包含“ohci”的是 OHCI 控制器相关文件，其中文件名包含厂商名字的为厂商引导文件，与 EHCI 一样，RK 使用 ohci-platform.c 进行驱动加载和初始化。类似的，ohci-hcd.c 实现 USB core 层的 HCD 控制器接口，ohci-mem.c 和 ohci-q.c 是传输数据结构和传输调度相关代码，ohci-hub.c 是 OHCI 控制器 root hub 驱动代码。

重要的结构体：

```
static const struct hc_driver ohci_hc_driver = {
    .description =          hcd_name,
    .product_desc =         "OHCI Host Controller",
    .hcd_priv_size =        sizeof(struct ohci_hcd),
    /*
     * generic hardware linkage
    */
    .irq =                  ohci_irq,
    .flags =                HCD_MEMORY | HCD_USB11,
    /*
    * basic lifecycle operations
    */
    .reset =                ohci_setup,
    .start =                ohci_start,
    .stop =                 ohci_stop,
    .shutdown =             ohci_shutdown,
    /*
     * managing i/o requests and associated device resources
    */
    .urb_enqueue =          ohci_urb_enqueue,
    .urb_dequeue =          ohci_urb_dequeue,
    .endpoint_disable =     ohci_endpoint_disable,
    /*
    * scheduling support
    */
    .get_frame_number =     ohci_get_frame,
    /*
    * root hub support
    */
    .hub_status_data =      ohci_hub_status_data,
    .hub_control =          ohci_hub_control,
#ifdef CONFIG_PM
    .bus_suspend =          ohci_bus_suspend,
    .bus_resume =           ohci_bus_resume,
#endif
    .start_port_reset = ohci_start_port_reset,
};
```

5.2.3.2 USB 2.0 HOST 调试接口

1. EHCI 驱动 debug 接口

（需要 enable CONFIG_DYNAMIC_DEBUG）

```
rk3399_box:/sys/kernel/debug/fe380000.usb # ls
async bandwidth periodic registers
```

async: Dump a snapshot of the Async Schedule.

bandwidth: Dump the HS Bandwidth Table.

periodic: Dump a snapshot of the Periodic Schedule.

registers: Dump Capability Registers, Interrupt Params and Operational Registers.

2. OHCI 驱动 debug 接口

（需要 enable CONFIG_DYNAMIC_DEBUG）

```
rk3399_box:/sys/kernel/debug/usb/ohci/fe3a0000.usb # ls
async periodic registers
```

async: Display Control and Bulk Lists together, for simplicity

periodic: Dump a snapshot of the Periodic Schedule (and load)

registers: Dump driver info, then registers in Spec order and other registers mostly affect Frame Timings

---

6 Android USB Gadget 配置

从Linux-3.11开始，USB Gadget 引入 ConfigFS 框架，同时内核也删除了 Gadget 目录下 android.c 文件。因此 Gadget 与之前配置方式有所差异。

关于如何使能 Android ConfigFS Gadgets 功能，请参考 Linaro 官网的说明：

https://wiki.linaro.org/LMG/Kernel/AndroidConfigFSGadgets

6.1 USB Gadget CONFIG

请参阅3.4 USB Gadget CONFIG

6.2 USB Gadget init script

Android 中与 USB 相关的 script 主要有：

```
init.usb.rc
init.usb.configfs.rc
init.rk30board.usb.rc
fstab.rk30board.bootmode.emmc
```

1. init.usb.rc：为 Android 标准 rc 文件，一般不需要改动。
2. fstab.rk30board.bootmode.emmc：为 Android fstab 文件，可以用于配置 sdcard、usb 的 mount 路径，Rockchip 平台的 vold 和 kernel 已经可以做到使用通配符来自动搜索和匹配 usb mount 路径，不需要再做修改

```
# for USB 2.0
/devices/platform/*.usb*          auto vfat defaults     voldmanaged=usb:auto
# for USB 3.0
/devices/platform/usb@*/*.dwc3*    auto vfat defaults     voldmanaged=usb:auto
```

3. init.rk30board.usb.rc 和 init.usb.configfs.rc：为我们平台 Gadget 功能的配置管理文件，其内容主要包括 usb gadget configfs 的创建、描述符的定义（VID/PID）、Gadget function 的配置等，如下所示：

```
on boot
    mkdir /dev/usb-ffs 0770 shell shell
    mkdir /dev/usb-ffs/adb 0770 shell shell
    mount configfs none /config
    mkdir /config/usb_gadget/g1 0770 shell shell
    write /config/usb_gadget/g1/idVendor 0x2207
    write /config/usb_gadget/g1/bcdDevice 0x0310
    write /config/usb_gadget/g1/bcdUSB 0x0200
    mkdir /config/usb_gadget/g1/strings/0x409 0770
    write /config/usb_gadget/g1/strings/0x409/serialnumber ${ro.serialno}
    write /config/usb_gadget/g1/strings/0x409/manufacturer
${ro.product.manufacturer}
    write /config/usb_gadget/g1/strings/0x409/product ${ro.product.model}
    mkdir /config/usb_gadget/g1/functions/accessory.gs2
    mkdir /config/usb_gadget/g1/functions/audio_source.gs3
    mkdir /config/usb_gadget/g1/functions/ffs.adb
    mkdir /config/usb_gadget/g1/functions/mtp.gs0
    mkdir /config/usb_gadget/g1/functions/ptp.gs1
    mkdir /config/usb_gadget/g1/functions/rndis.gs4
    write /config/usb_gadget/g1/functions/rndis.gs4/wceis 1
    mkdir /config/usb_gadget/g1/functions/midi.gs5
    mkdir /config/usb_gadget/g1/configs/b.1 0770 shell shell
    mkdir /config/usb_gadget/g1/configs/b.1/strings/0x409 0770 shell shell
    write /config/usb_gadget/g1/os_desc/b_vendor_code 0x1
    write /config/usb_gadget/g1/os_desc/qw_sign "MSFT100"
    write /config/usb_gadget/g1/configs/b.1/MaxPower 500
    symlink /config/usb_gadget/g1/configs/b.1 /config/usb_gadget/g1/os_desc/b.1
    mount functionfs adb /dev/usb-ffs/adb uid=2000,gid=2000
    setprop sys.usb.configfs 1
    setprop sys.usb.controller "fe800000.dwc3"

on property:sys.usb.config=none && property:sys.usb.configfs=1
    write /config/usb_gadget/g1/os_desc/use 0
    setprop sys.usb.ffs.ready 0

on property:init.svc.adbd=stopped
    setprop sys.usb.ffs.ready 0

on property:sys.usb.config=mtp && property:sys.usb.configfs=1
    write
/config/usb_gadget/g1/functions/mtp.gs0/os_desc/interface.MTP/compatible_id
"MTP"
    write /config/usb_gadget/g1/os_desc/use 1
    write /config/usb_gadget/g1/idProduct 0x0001
```

```
on property:sys.usb.config=mtp,adb && property:sys.usb.configfs=1
    write
/config/usb_gadget/g1/functions/mtp.gs0/os_desc/interface.MTP/compatible_id
"MTP"
    write /config/usb_gadget/g1/os_desc/use 1
    write /config/usb_gadget/g1/idProduct 0x0011
```

其中，Serialnumber、manufacturer、product 三个属性由 Android 配置。如果 Serialnumber 没有配置成功，可能会造成 ADB 无法使用。

setprop sys.usb.controller 用来使能 Gadget 对应的 USB 控制器，RK3399 有两个 OTG 控制器，都可以支持 USB Gadget 功能，但由于当前 USB Gadget driver 内核架构只支持一个 USB 控制器，所以需要根据实际的产品需求来配置使能对应的 USB 控制器，如 RK3399 Android SDK，默认使能 Type-C 0 port 的 USB Gadget 功能：

```
setprop sys.usb.controller "fe800000.usb"
```

如果要使能 Type-C 1 port 的 USB Gadget 功能，则修改为 init.rk30board.usb.rc 的 sys.usb.controller 为 fe900000.usb，参考修改如下：

```
setprop sys.usb.controller "fe900000.usb"
```

6.3 USB Gadget 调试接口

内核提供了设备节点来查看 USB Gadget 的关键配置信息，如下：

```
root@rk3399:/config/usb_gadget/g1 # ls
UDC          bDeviceProtocol bMaxPacketSize0 bcdUSB  functions idVendor strings
bDeviceClass bDeviceSubClass bcdDevice       configs idProduct os_desc
```

大部分节点的功能，可以直观地看出来，这里就不再赘述。

“UDC”可以确认当前 Gadget 对应的 usb controller, 也可以用于手动选择对应的 usb controller。如默认使用 Type-C 0 USB Controller，要切换为使用 Type-C 1 USB Controller，则手动执行如下的命令：

```
echo none > config/usb_gadget/g1/UDC
echo fe900000.dwc3 > config/usb_gadget/g1/UDC
```

7 常见问题分析

7.1 设备枚举日志

7.1.1 USB 2.0 OTG 正常开机

开机未连线，默认为 device 模式。

```
[    8.764441]otg id chg last id -1 currentid 67108864
[    8.764925] PortPower off
[    8.866923] Using Buffer DMA mode
[    8.867280] Periodic Transfer Interrupt Enhancement- disabled
[    8.867787] Multiprocessor InterruptEnhancement - disabled
[    8.868294] OTG VER PARAM: 0, OTG VER FLAG: 0
[    8.868700] ^^^^^^^^^^^^^^^^^Device Mode
```

7.1.2 USB 2.0 Device 连接

```
[  133.368479] ***vbusdetect*
[ 133.500590] Using Buffer DMA mode
[  133.500886] Periodic Transfer InterruptEnhancement - disabled
[  133.501391] Multiprocessor InterruptEnhancement - disabled
[  133.501875] OTG VER PARAM: 0, OTG VER FLAG: 0
[  133.502255] ^^^^^^^^^^^^^^^^^Device Mode
[  133.502630] ***************softconnect!!!********************
[  133.618581] USB RESET
[  133.710877] android_work: sent ueventUSB_STATE=CONNECTED
[  133.714269] USB RESET
[  133.947001] configfs-gadget gadget: high-speed config #1: b
[  133.947649] android_work: sent ueventUSB_STATE=CONFIGURED
[  133.995447] mtp_open
```

7.1.3 USB 2.0 Device 断开连接

```
[  187.085682] ********session end ,softdisconnect***********
[  187.086486] android_work: sent ueventUSB_STATE=DISCONNECTED
[  187.087217] mtp_release
```

7.1.4 USB 2.0 HOST-LS 设备

```
[  325.412454] usb 2-1: new low-speed USB device number 2 using ohci-platform
[  325.619507] usb 2-1: New USB device found,idVendor=046d, idProduct=c077
[  325.620116] usb 2-1: New USB device strings:Mfr=1, Product=2, SerialNumber=0
[  325.620809] usb 2-1: Product: USB OpticalMouse
[  325.621222] usb 2-1: Manufacturer: Logitech
```

7.1.5 USB 2.0 HOST-FS 设备

```
[  370.896519] usb 2-1: new full-speed USB device number 3 using ohci-platform
[  371.109574] usb 2-1: New USB device found,idVendor=1915, idProduct=0199
[  371.110183] usb 2-1: New USB device strings:Mfr=1, Product=2, SerialNumber=0
[  371.110832] usb 2-1: Product: Memsartcontroller
[  371.111251] usb 2-1: Manufacturer: Memsart
[  371.123172] input: Memsart Memsart controlleras /
```

7.1.6 USB 2.0 HOST-HS 设备

```
[  405.400521] usb 1-1: new high-speed USB device number 5 using ehci-platform
[  405.536569] usb 1-1: New USB device found,idVendor=0951, idProduct=1687
[  405.537178] usb 1-1: New USB device strings:Mfr=1, Product=2, SerialNumber=3
[  405.537815] usb 1-1: Product: DT R400
[  405.538151] usb 1-1: Manufacturer: Kingston
[  405.538533] usb 1-1: SerialNumber:0018F3D97D02BB91517E017D
[  405.541111] usb-storage 1-1:1.0: USB MassStorage device detected
[  405.542472] scsi host1: usb-storage 1-1:1.0
[  406.584573] scsi 1:0:0:0: Direct-AccessKingston DT R400 PMAP PQ: 0 ANSI: 0
CCS
[  406.586425] sd 1:0:0:0: Attached scsi genericsg0 type 0
[  408.171256] sd 1:0:0:0: [sda] 15646720512-byte logical blocks: (8.01 GB/7.46
GiB)
[  408.172788] sd 1:0:0:0: [sda] Write Protectis off
[  408.173970] sd 1:0:0:0: [sda] No Caching modepage found
[  408.174453] sd 1:0:0:0: [sda] Assuming drivecache: write through
[  408.223001] sda: sda1
[  408.229280] sd 1:0:0:0: [sda] Attached SCSIremovable disk
```

7.1.7 USB 2.0 HOST-LS/FS/HS 设备断开

```
[  443.151067] usb 1-1: USB disconnect, devicenumber 3
```

7.1.8 USB 3.0 Device 连接

```
[   72.310531] android_work: sent ueventUSB_STATE=CONNECTED
[   72.689120] configfs-gadget gadget: super-speed config #1: b
[   72.690110] android_work: sent ueventUSB_STATE=CONFIGURED
[   72.767950] mtp_open
```

7.1.9 USB 3.0 HOST-SS 设备

```
[   26.715320] usb 8-1: new SuperSpeed USB device number 2 using xhci-hcd
[   26.732190] usb 8-1: New USB device found,idVendor=0bc2, idProduct=2320
[   26.732812] usb 8-1: New USB device strings:Mfr=2, Product=3, SerialNumber=1
[   26.733515] usb 8-1: Product: Expansion
[   26.733885] usb 8-1: Manufacturer: Seagate
[   26.734263] usb 8-1: SerialNumber: NA45HT1K
```

```
[   26.738410] usb-storage 8-1:1.0: USB MassStorage device detected
[   26.740446] scsi host0: usb-storage 8-1:1.0
[   27.745028] scsi 0:0:0:0: Direct-Access     Seagate Expansion        0608 PQ:
0 ANSI:6
[   27.753066] sd 0:0:0:0: [sda] 1953525167512-byte logical blocks: (1.00 TB/932
GiB)
[   27.754245] sd 0:0:0:0: [sda] Write Protectis off
[   27.754982] sd 0:0:0:0: Attached scsi genericsg0 type 0
[   27.755281] sd 0:0:0:0: [sda] Write cache:enabled, read cache: enabled,
doesn't support DPO or FUA
[   27.783395] sda: sda1
[   27.791561] sd 0:0:0:0: [sda] Attached SCSIdisk
```

7.2 USB 常见问题分析

7.2.1 硬件电路

在同时使用多个控制器对应同一个 USB 口，或者一个控制器对应多个 USB 口时，可能会使用电子开关来切换 USB 信号及电源。需要确保不同控制器的电源控制是互相独立的，通过电子开关后，控制器与 USB 口之间的连接是有效的。

场景一:

1 个硬件 USB 口同时支持 host 和 device 功能，使用 USB 2.0 HOST 控制器作为 host 和 USB 2.0 OTG 控制器作为 device，通过硬件电子开关进行切换。

需要保证工作于 host 状态时，USB 信号是切换到 USB 2.0 HOST 控制器，而 VBUS 是由 HOST 供电电路提供，而不影响 device 的 VBUS 电平检测电路。工作于 device 状态时，USB 信号是切换到 USB 2.0 OTG 控制器，VBUS 由 PC 通过 USB 线提供。

场景二:

使用一个 USB 2.0 OTG 控制器，对应使用两个硬件 USB 口分别是 host 和 device。通过电子开关进行信号切换。

工作于 HOST 状态时，USB 2.0 OTG 的 DP/DM 信号线是切换到 HOST 口，且 HOST 口 VBUS 提供 5V 500MA 的供电；工作于 device 状态时 DP/DM 信号是切换到 device 口，VBUS 电平检测电路只检测 PC 提供的 5V 供电。

7.2.2 Device 功能异常分析

USB Device 正常连接至 PC 的现象主要有：

1. 串口输出正常 log，见7.1.2 USB 2.0 Device 连接；
2. PC 出现盘符，但默认不能访问；(Windows 7 和 MAC OS 可能只出现在设备管理器)；
3. 设备 UI 状态栏出现"USB 已连接"标识；
4. 打开 USB 已连接的提示窗口，默认为 charger only 模式，选择"MTP"或者"PTP"后，PC 可以访问盘符。

USB Device 异常现象的排查方法：

异常 1：连接 USB 时串口完全没有 log

排查方法：首先，通过调试接口，确认 USB 控制器是否工作在 device 状态；然后，检查 USB 硬件电路和信号；最后，测量 USB_DET 信号电压，USB 连接时应该由低到高。

异常 2：连接失败，PC 显示不可识别设备，log 一直重复打印：

```
[36.682587] DWC_OTG:
********softconnect!!!****************************************
[36.688603] DWC_OTG: USB SUSPEND
[36.807373] DWC_OTG: USB RESET
```

排查方法：没有正常 log 中的后面几条信息。一般为 USB 硬件信号差，无法完成枚举。

异常 3：连接 PC 后，kernel log 正常，并且设备为出现 USB 已连接"标识，但 PC 无法访问设备

排查方法：驱动工作正常，请先确认是否有选择 USB 为"MTP"或"PTP"，如果已选择，则可能是 Android 层异常，请截取 logcat 内容，并请负责维护 vold/mtpserver 代码的 Android 工程师帮忙 debug。

异常 4：连接 PC 正常，并能正常访问，拷贝文件过程中提示拷贝失败。

排查方法：首先，测试下 USB 眼图，并使用 USB 分析仪抓取数据流进行分析。其次，排查是否为 Flash/SD 卡读写超时，log 一般为连接 Window XP 时约 10S 出现一次重新连接的 log。最后，排斥是否为 Flash/SD 磁盘分区出错，导致每次拷贝到同一个点时失败。可使用命令检查并修复磁盘分区。假设挂载的磁盘分区为 E，则打开 Windows 命令提示符窗口，输入命令：chkdsk E: /f

异常 5：USB 线拔掉后 UI 状态栏仍然显示"USB 已连接"，或 USB 线拔掉时只有以下 log：

```
[25.330017] DWC_OTG: USB SUSPEND
```

而没有下面的 log：

```
[25.514407] DWC_OTG: session end intr, softdisconnect
```

排查方法：VBUS 异常，一直为高，会影响 USB 检测及系统休眠唤醒，请硬件工程师排查问题。

7.2.3 Host 功能异常分析

USB HOST 正常工作情况如下：

1. 首先 HOST 电路提供 5V，至少 500mA 的供电；
2. 如果有 USB 设备连接进来，串口首先会打印 HOST 枚举 USB 设备的 log(见7.1.4至7.1.7)，表明 USB 设备已经通过 HOST 的标准设备枚举；

USB HOST 异常现象的排查方法：

异常 1：HOST 口接入设备后，串口无任何打印：

排查方法：首先需要确认通过电子开关后的电路连接正确；然后，确认控制器工作于 HOST 状态，并确认供电电路正常。

异常 2：串口有 HOST 枚举 USB 设备内容，但是没有出现 class 驱动的打印信息。

排查方法：Kernel 没有加载 class 驱动，需要重新配置 kernel，加入对应 class 驱动支持。

异常 3：kernel 打印信息完整(USB 标准枚举信息及 CLASS 驱动信息)，已在 Linux 对应位置生成节点，但是 Android 层无法使用。

排查方法：Android 层支持不完善，如 U 盘在 kernel 挂载完成/dev/block/sda 节点后，需要 Android 层 vold 程序将可存储介质挂载到/udisk 提供媒体库，资源管理器等访问，同样鼠标键盘等 HID 设备也需要 Android 层程序支持。

U 盘枚举出现/dev/block/sda 后仍然无法使用，一般是 vold.fstab 中 U 盘的 mount 路径有问题，如果 vold.fstab 代码如下(系统起来后可直接 cat/system/etc/vold.fstab 查看)：

```
dev_mount udisk /mnt/udisk 1 /devices/platform/usb20_HOST/usb2
```

而实际的 device 路径可能是在 usb20_OTG 控制器下或者最后的字段为 usb1.

如果设备属于这种情况的无法正常使用，需要联系 Android 工程师帮忙 debug。

异常 4：串口一直打印如下提示字节没有对齐的类似 log：

```
DWC_OTG:dwc_otg_hcd_urb_enqueue urb->transfer_buffer address not align to 4-
byte0xd6eab00a
DWC_OTG:dwc_otg_hcd_urb_enqueue urb->transfer_buffer address not align to 4-
byte0xccf6140a
```

排查方法：RK 平台的 USB 驱动要求在提交 URB 传输请求时， URB 的成员 transfer_buffer 地址必须为四字节对齐，否则会提示上述错误 log。

如：函数 usb_control_msg 的 data 参数必须要四字节对齐。

异常 5：OTG 口作为 host 时，无法识别接入的设备

排查方法：首先，检查 Kernel 的 OTG 配置是否正确；然后，检查 OTG 电路的 ID 电平(作 host，为低电平)和 VBUS 5V 供电是否正常；最后，如果确认 1 和 2 都正常，仍无法识别设备，请提供设备插入后无法识别的错误 log 给我们。

7.2.4 USB Camera 异常分析

1. 使用 Camera 应用，无法打开 USB camera

   首先，检查/dev 目录下是否存在 camera 设备节点 video0 或 video1，如果不存在，请检查 kernel 的配置是否正确，如果存在节点，请确认 USB camera 是在系统开机前插入的，因为 RK 平台的 SDK，默认是不支持 USB camera 热拔插的。如果要支持 USB camera 热拔插，请联系负责 camera 的工程师修改 Android 相关代码，USB 驱动不需要做修改。
   如果仍无法解决，请提供 log 给负责 USB 驱动工程师或者负责 camera 的工程师，进一步分析。

2. 出现概率性闪屏、无图像以及 camera 应用异常退出的问题

   可能是 USB 驱动丢帧导致的。需要使用 USB 分析仪抓实际通信的数据进行分析，如果无法定位，请联系负责 USB 驱动的工程师。

7.2.5 USB 充电检测

USB2 PHY 支持 BC1.2 标准的充电检测，代码实现请参考 drivers/phy/rockchip/phy-rockchip-inno-usb2.c， 可以检测 SDP/CDP/标准 DCP(D+/D-短接)/非标准 DCP(D+/D-未短接)四种充电类型。

SDP —— Standard Downstream Port

根据 USB 2.0 规范，当 USB 外设处于未连接(un-connect)或休眠(suspend)的状态时，一个 Standard Downstream Port 可向该外设提供不超过 2.5mA 的平均电流;当外设处于已经连接并且未休眠的状态时，电流可以至最大 100mA(USB 3.0 150mA);而当外设已经配置(configured)并且未休眠时，最大可从 VBUS 获得 500mA(USB 3.0 900mA)电流。

CDP —— Charging Downstream Port

即兼容 USB 2.0 规范，又针对 USB 充电作出了优化的下行 USB 接口，提供最大 1.5A 的供电电流，满足大电流快速充电的需求。

DCP —— Dedicated Charging Port (USB Charger)

BC1.2 spec 要求将 USB Charger 中的 D+和 D-进行短接，以配合 USB 外设的识别动作，但它不具备和 USB 设备通信的能力。

USB 充电类型检测流程见下图所示：

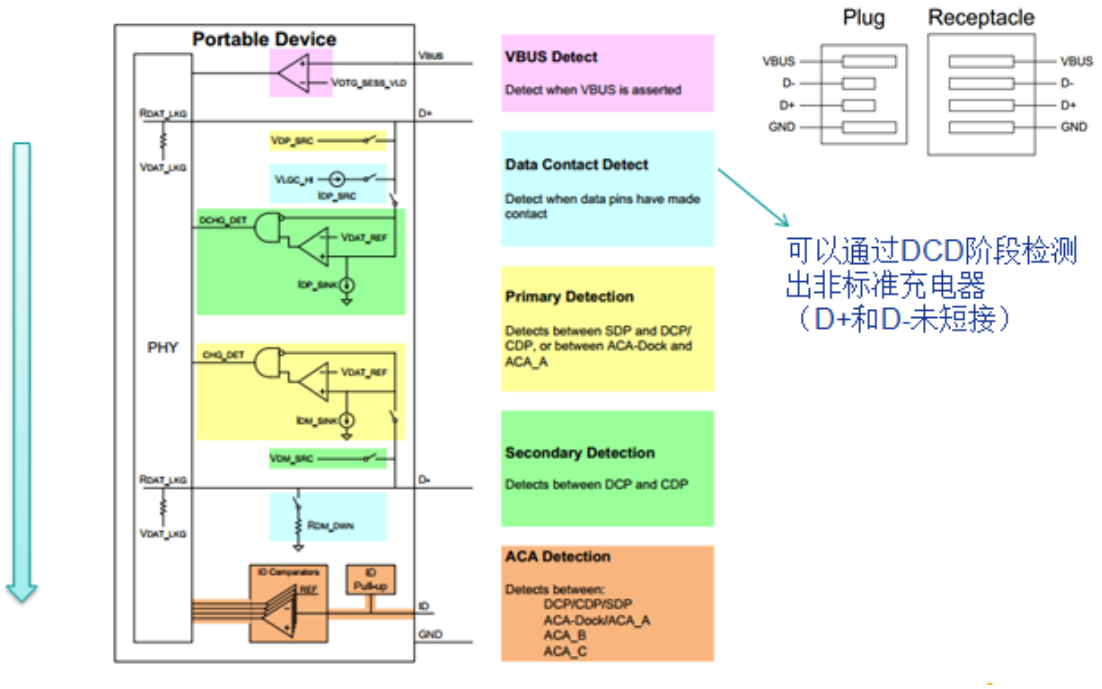

图 7-1 USB 充电检测流程

典型的 SDP 检测过程中，D+/D-波形如下图所示：

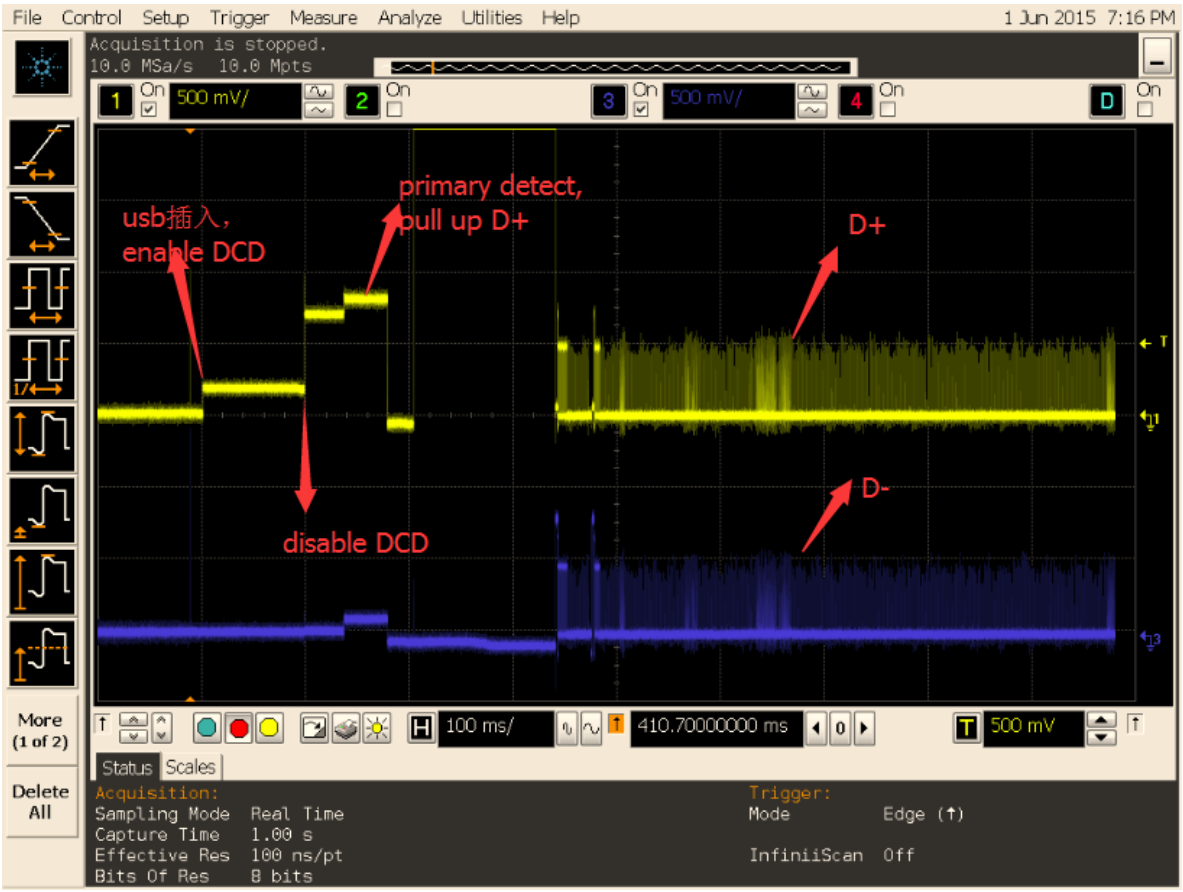

图 7-2 SDP 检测波形

典型的 DCP 检测过程中，D+/D-波形如下图所示：

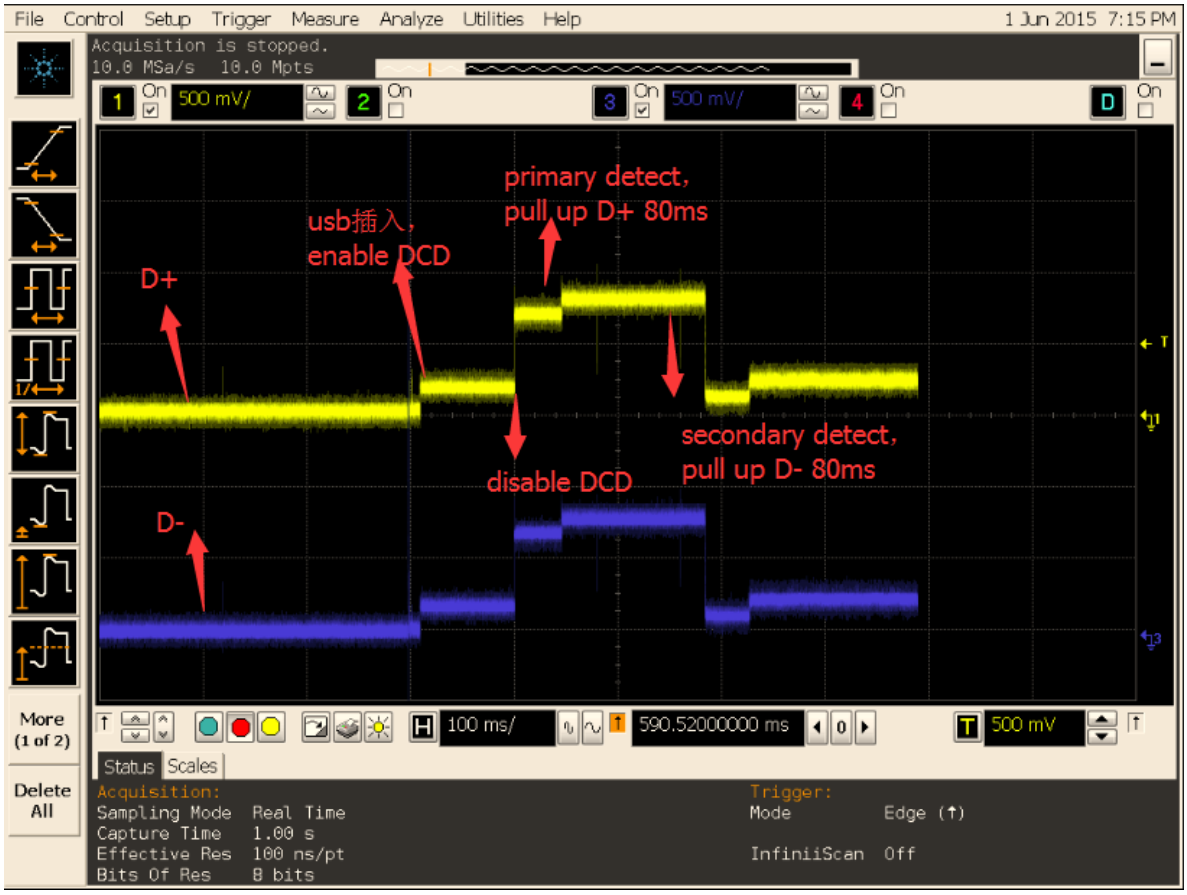

图 7-3 DCP 检测波形

如果连接 USB 充电器，发现充电慢，有可能是 DCP 被误检测为 SDP，导致充电电流被设置为 500mA。当 USB 线连接不稳定或者充电检测驱动出错，都可能会产生该问题。解决方法：

抓取 USB 充电器连接的 log，通过 log 的提示判断检测的充电类型，正常应为 DCP；

如果连接的是 USB 充电器，但 log 提示为 SDP，则表示发生了误检测。请先更换 USB 线测试，并使用万用表确认 D+/D-是否短接。如果仍无法解决，请将检测的 log 发给我们测试。同时，如果有条件，请使用示波器抓 USB 插入时的 D+/D-波形，并连同 log 一起发送给我们分析和定位问题。

如果连接的是 USB 充电器，并且 log 提示为 DCP，但充电仍然很慢，则表明软件检测正常，可能是充电 IC 或者电池的问题。

7.3 PC 驱动问题

所有 USB 设备要在 PC 上正常工作都是需要驱动的，有些驱动是标准且通用的，而有些驱动是需要额外安装的。对于 RK 的设备连接到 PC 后，需要安装驱动的情况有两种的设备，需要分别选择对应的驱动。

1. 生成后未烧写的裸片或者进入升级模式后的 RK 设备，会以 rockUSB 的模式连接到 PC，需要在 PC 端使用 RK 平台专门的驱动安装助手 DriverAssitant（RK3399 需要 v4.4 支持）安装驱动才能识别到 USB 设备；
2. RK 的设备正常运行时，在设置里面打开了 USB debugging 选项，连接时会以 ADB 的模式连接 PC，同样需要在 PC 端使用 RK 平台专门的驱动安装助手 DriverAssitant 安装 ADB 驱动后，才能正常识别到 ADB 设备。

8 USB 信号测试

[USB 2.0/3.0 信号测试方法及常见问题分析请参阅《RK USB Compliance Test NoteV1.2》