Rockchip
瑞芯微电子

Rockchip
USB 开发指南

发布版本:1.0

日期:2016.07

前言

概述

产品版本

芯片名称	内核版本
RK3399	Linux4.4

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
2016-07-04	v1.0	WLF WMC	

本文档适用范围：

RK3399 芯片。运行 Android 6.0，Linux Kernel4.4 及以上系统。

目录

插图目录

表格目录

1 概述

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。RK3399 芯片内置的 USB 控制器如表 1-1 所示：

表 1-1 RK3399 平台 USB 控制器列表

控制器 / 芯片	USB2.0 HOST（EHCI&OHCI）	USB HSIC（EHCI）	USB2.0/3.0 OTG（DWC3/XHCI）
RK3399	√	√	√

RK3399 SoC 包含 2 个 USB2.0 HOST 控制器，2 个 USB2.0/3.0 OTG 控制器，1 个 HSIC 控制器（只支持 USB2.0 speed）,各个控制器及 USB PHY 的具体硬件信息说明如下：

USB Host2.0

Embedded 2 USB Host 2.0 interfaces

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

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

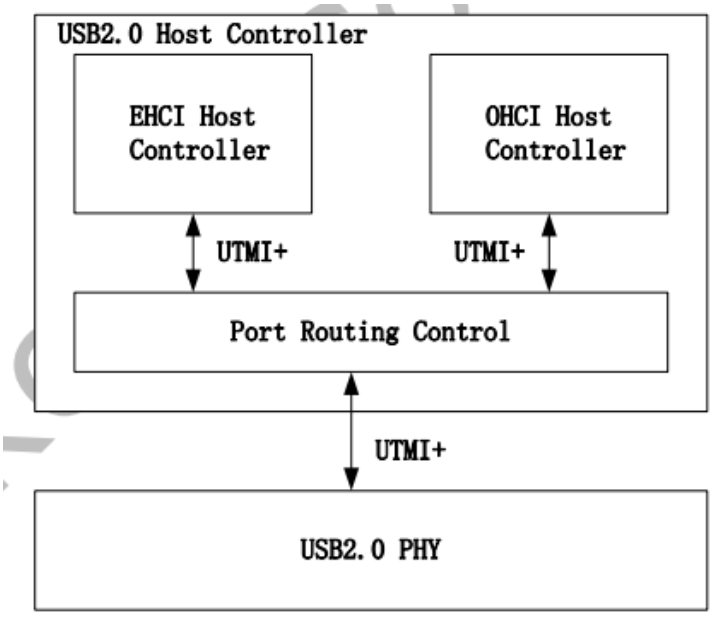


图 1-1 USB2.0 Host Controller Block Diagram

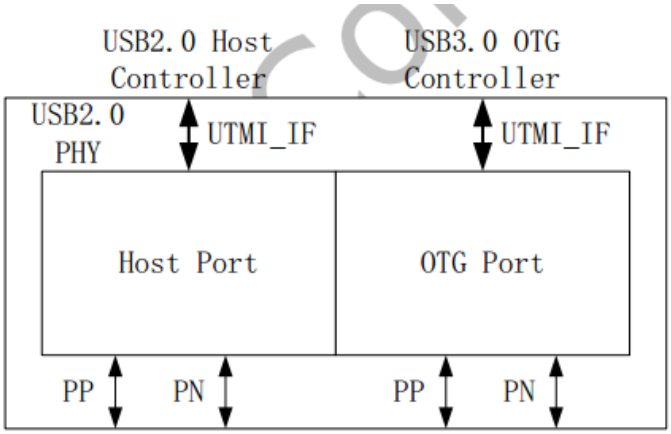


图 1-2 USB2.0 USB2.0 PHY Block Diagram

USB2.0 PHY

Host Port: used for USB2.0 host controller

OTG Port: used for USB3.0 OTG controller with TypeC PHY to comprise as fully feature TypeC

USB OTG3.0

Embedded 2 USB OTG3.0 interfaces

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

Support Control/Bulk(including stream)/Interrupt/Isochronous Transfer

Simultaneous IN and OUT transfer for USB3.0, up to 8Gbps bandwidth

Descriptor Caching and Data Pre-fetching

USB3.0 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/USB set-configuration modes

Hardware handles ERDY and burst

Stream-based bulk endpoints with controller automatically initiating data movement

Isochronous endpoints with isochronous data in data buffers

Flexible Descriptor with rich set of features to support buffer interrupt moderation, multiple transfers, isochronous, control, and scattered buffering support

USB 3.0 xHCI Host Features

Support up to 64 devices

Support 1 interrupter

Support 1 USB2.0 port and 1 Super-Speed port

Concurrent USB3.0/USB2.0 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

USB3.0/USB2.0 OTG A device and B device basing on ID

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

Not support USB3.0/USB2.0 OTG session request protocol(SRP), host negotiation protocol(HNP) and Role Swap Protocol(RSP)

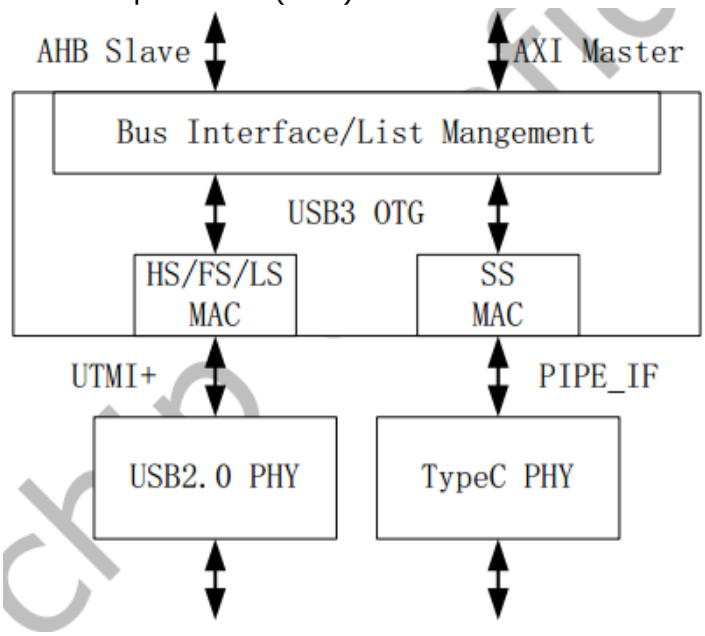


图 1-3 USB3.0 OTG Block Diagram

TypeC PHY

Support USB3.0 (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 pin assignments

Support Normal and Flipped orientation

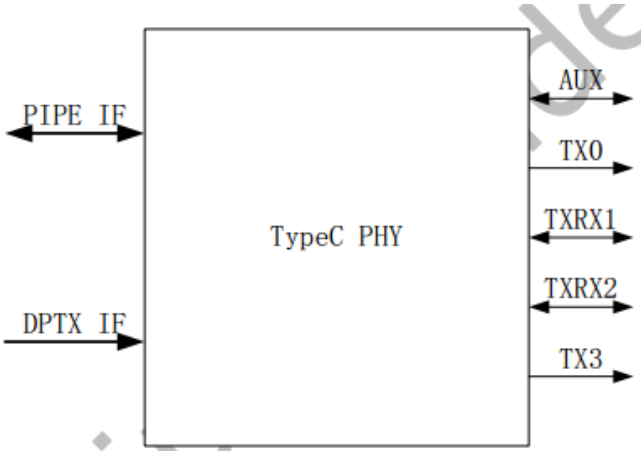


图 1-4 TypC PHY Block Diagram

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 USB2.0 Host 和 HSIC，其硬件电路及信号分别说明如下：

2.1.1 USB2.0 HOST 控制器硬件电路

USB2.0 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 使用 240MHz DDR 信号，传输速率与 USB2.0 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。

RK3399 USB2.0 HOST 控制器硬件信号如图 1-1 和图 1-2 所示，完整的 USB 2.0 HOST 电路如图 2-1 ，图 2-3 所示：

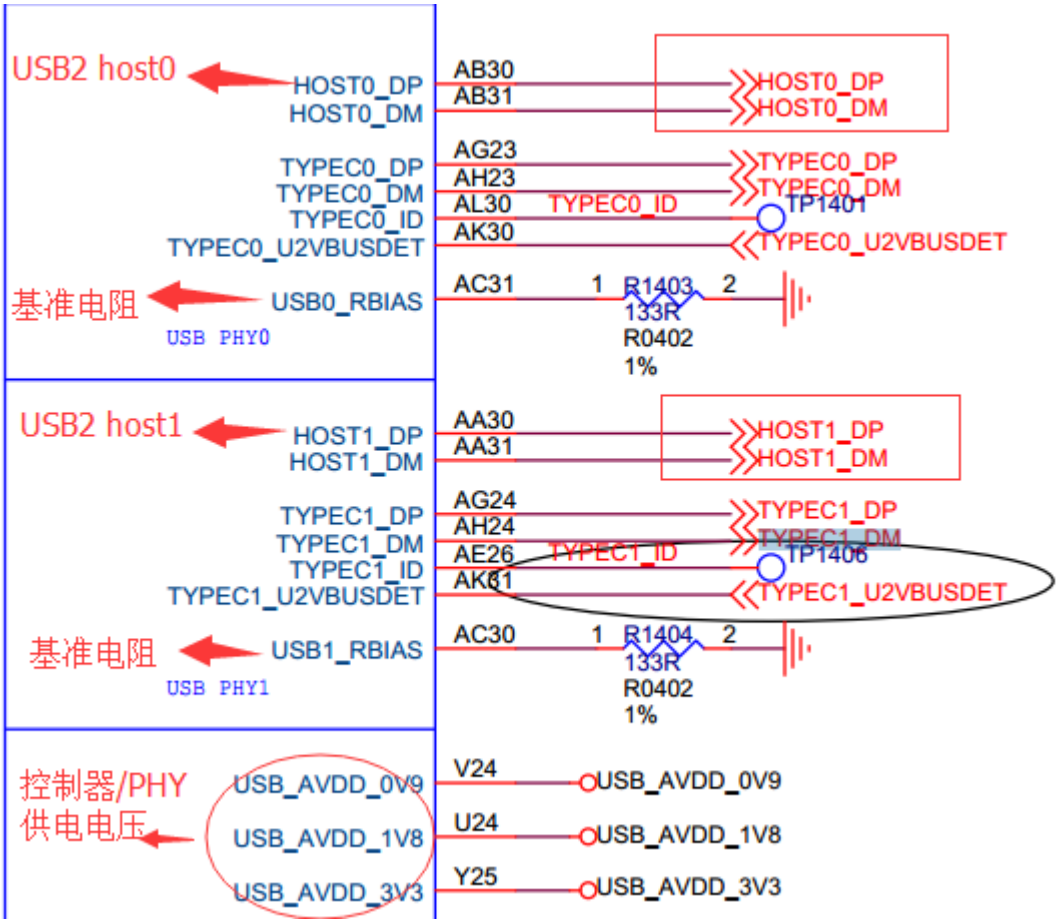


图 2-1 USB 2.0 HOST SoC 信号引脚

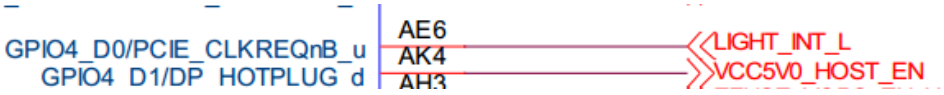


图 2-2 USB 2.0 HOST VBUS GPIO 控制脚

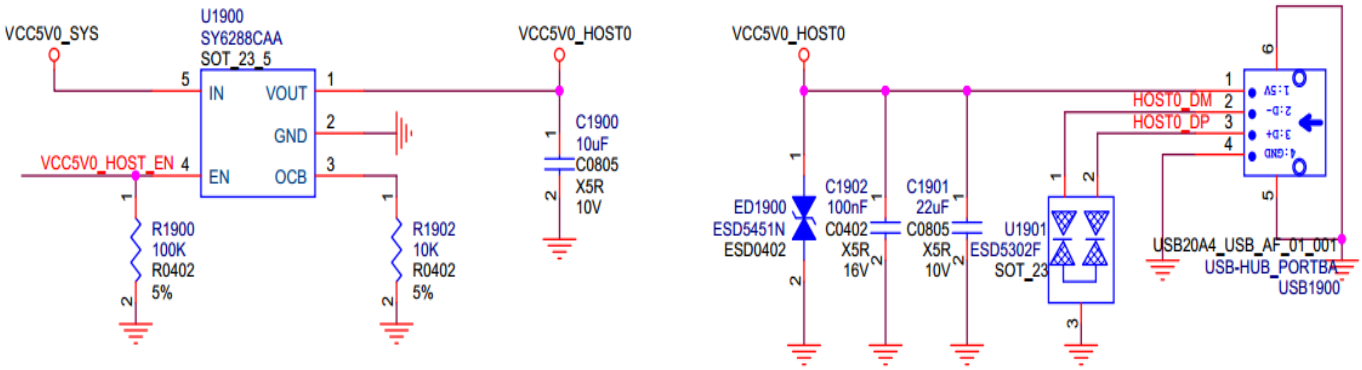


图 2-3 HSIC 控制器硬件电路

HSIC 是具有与 USB2.0 相同的带宽（480Mbps）的 2 引脚芯片间互连接口，HSIC 去除了为 USB2.0 设计的模拟收发器（PHY），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）。

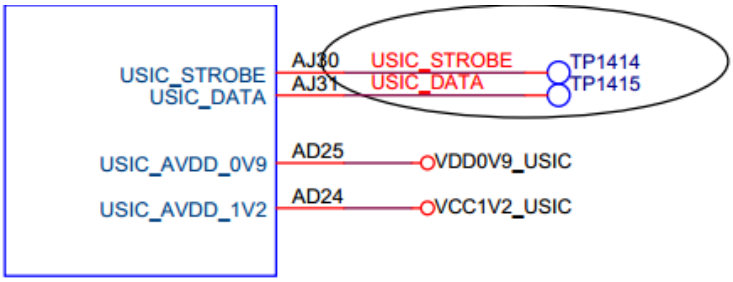


图 2-4 HSIC 硬件电路

2.2 USB OTG 控制器硬件电路

2.2.1 USB2.0 OTG 控制器硬件电路

RK3399 没有独立的 USB2.0 OTG 控制器，但有独立的 USB3.0 OTG 控制器，并且可以向下兼容 USB2.0 OTG 的完整功能。

2.2.2 USB3.0 OTG 控制器硬件电路

RK3399 支持 USB3.0 OTG 功能，且向下兼容 USB2.0 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，USB3.0 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

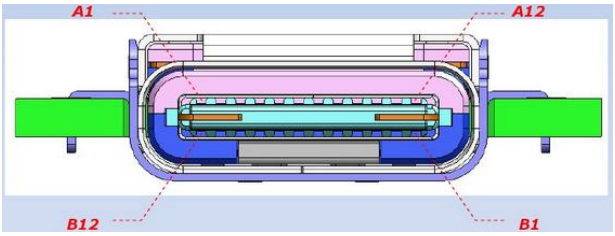


图 2-5 Type-C 接口定义

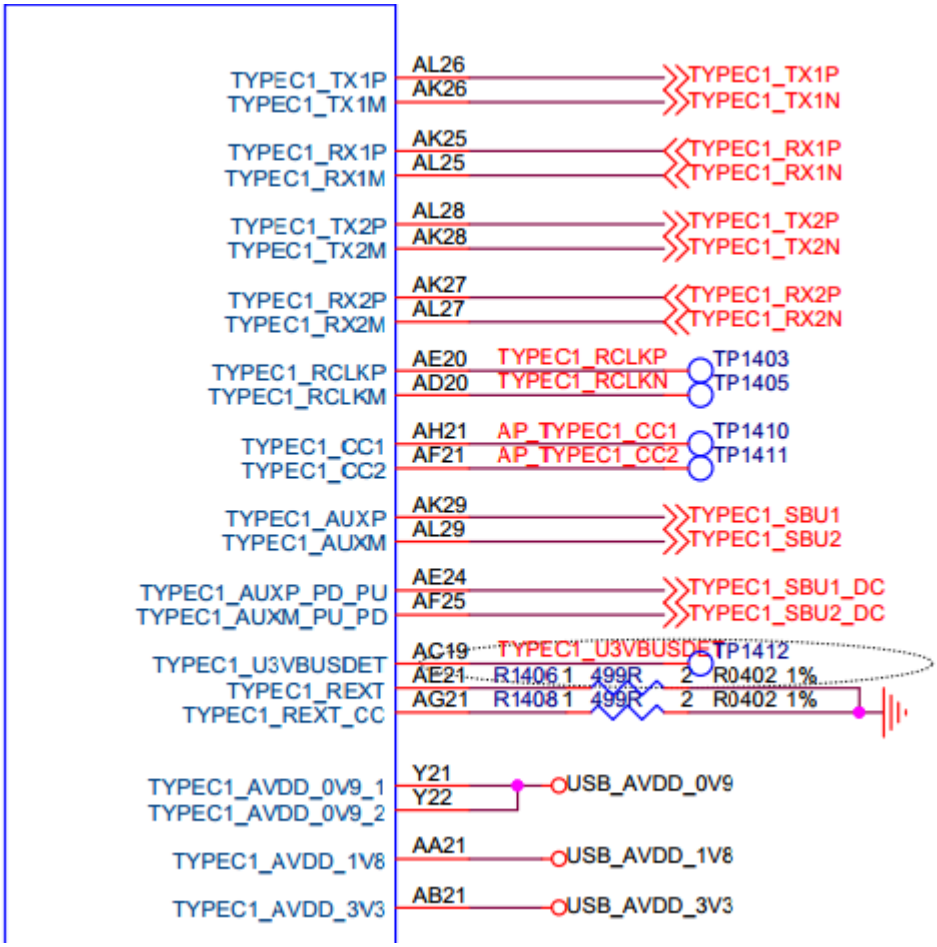


图 2-6 USB3 OTG 控制器 SoC 信号引脚

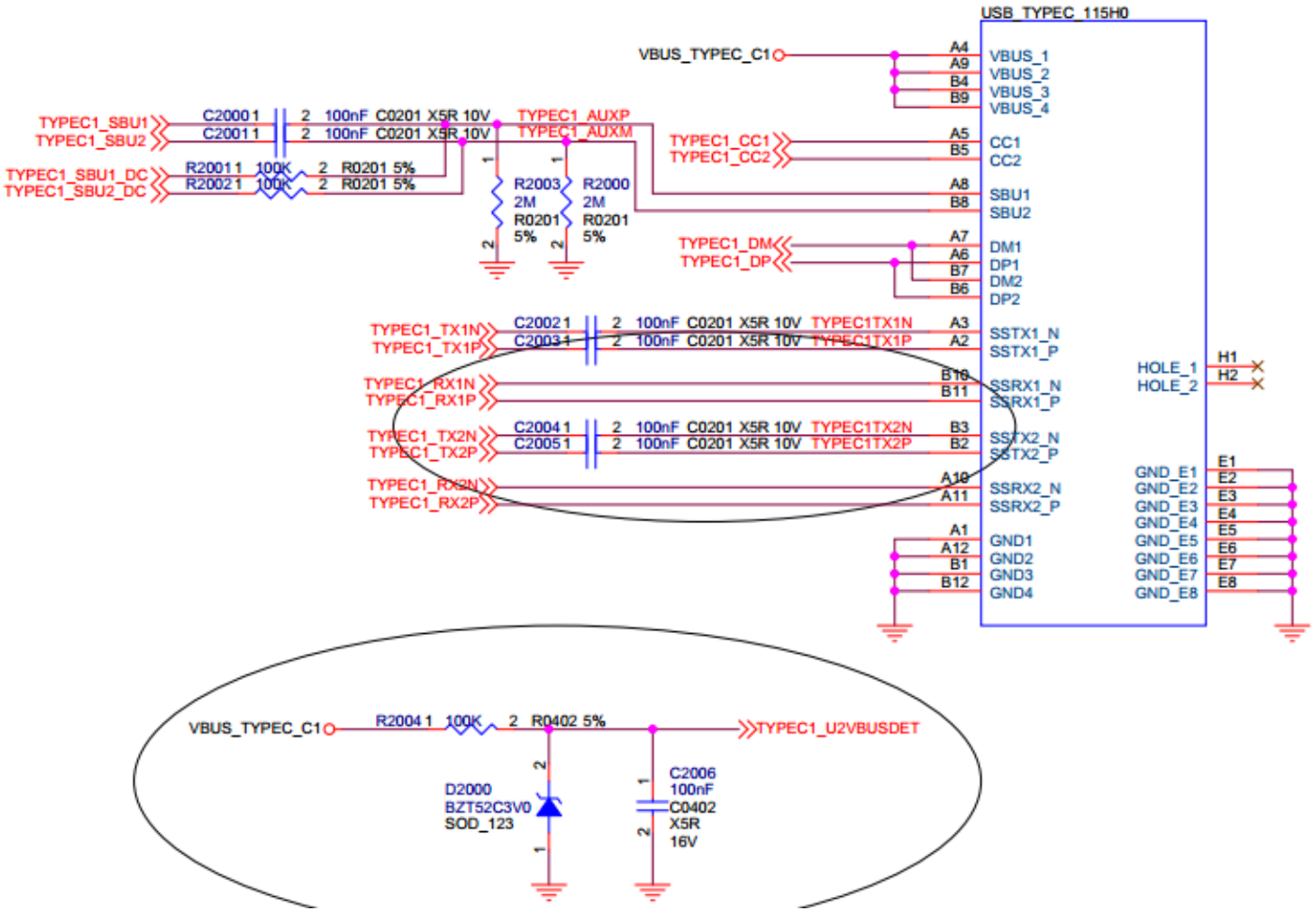


图 2-7 USB3 OTG Type-C 接口

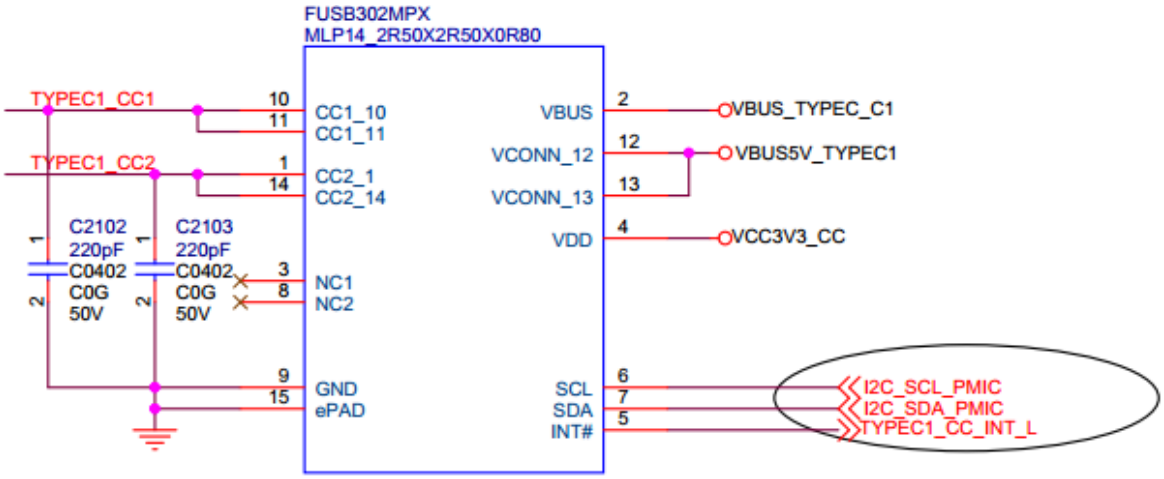


图 2-8 USB3 Type-C pd/cc 电路（FUSB302）

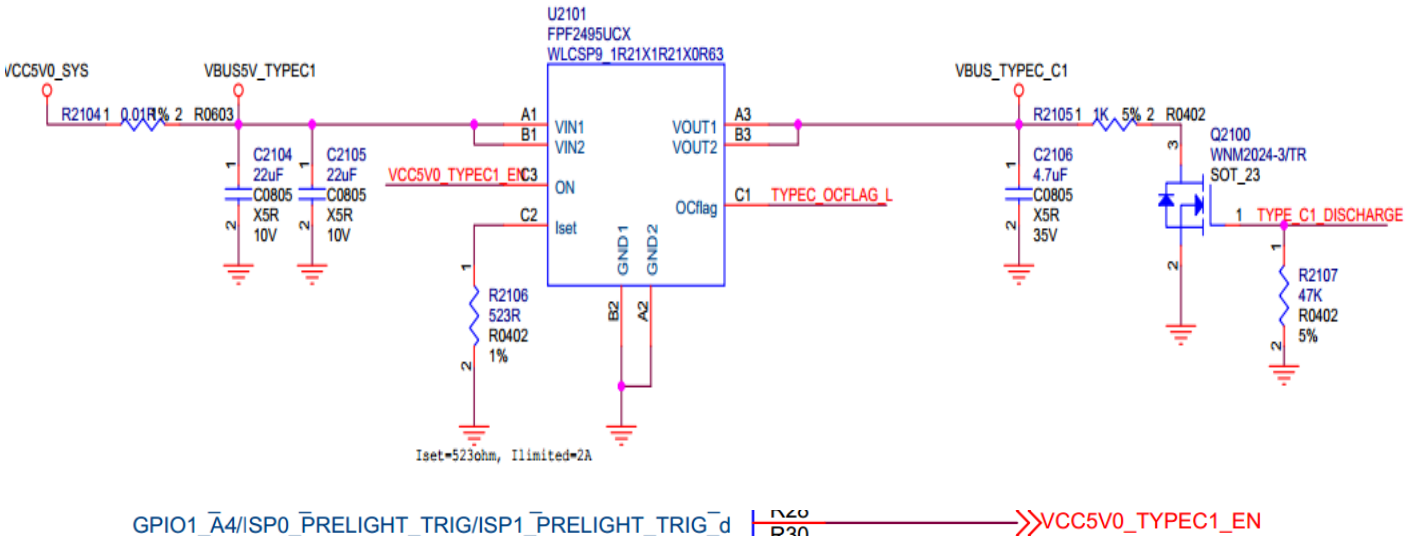


图 2-9 USB3 VBUS 控制电路-1（GPIO 控制电路输出 5V）

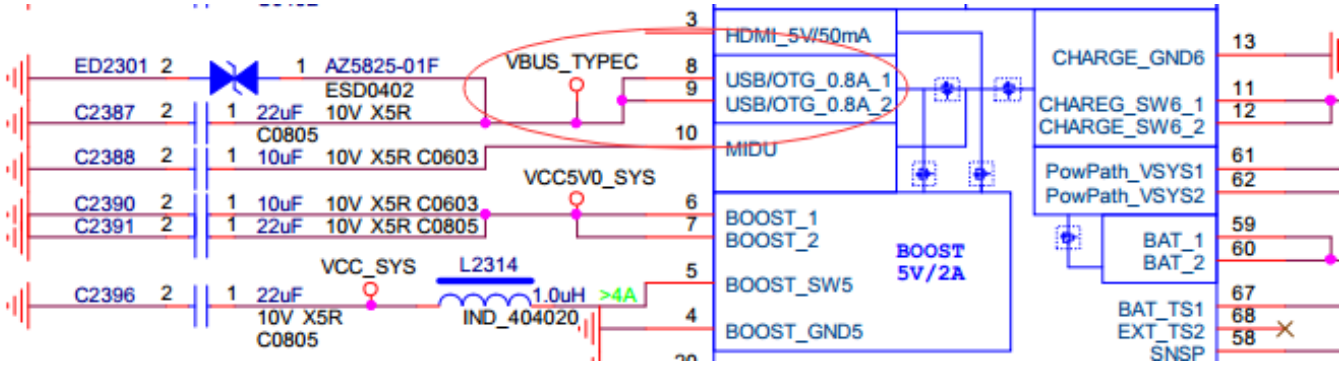


图 2-10 USB3 VBUS 控制电路-2（RK818 控制电路输出 5V）

3 Kernel 模块配置

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 相关配置

USB PHY 模块的配置位于

```
Device Drivers  --->
PHY Subsystem  --->
```

RK3399 USB2.0 PHY 使用的是 Innosilicon IP，所以应选择“Rockchip INNO USB2PHY Driver”。

RK3399 USB3.0 PHY 使用的是 Type-C，所以应选择“Rockchip TYPEC PHY Driver”。

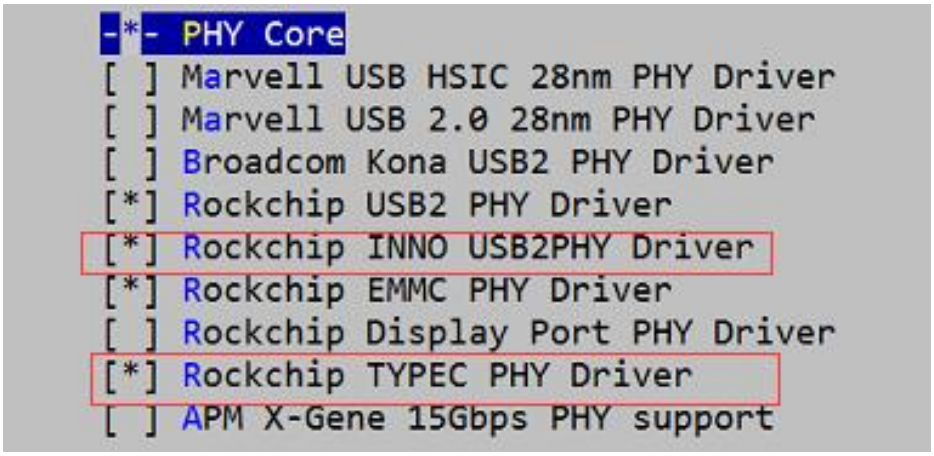


图 3-1 USB PHY Driver 配置示意图

3.2 USB HOST 相关配置

Make menuconfig 得到 kernel 配置界面后，USB 模块的配置位于

```
Device Drivers  --->
[*] USB support  --->
```

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

需要支持 USB HOST，首先需要选上<*>Support for Host-side USB 项，然后会现如下的 HOST 相关的配置，其中，HOST1.1 选择 OHCI Driver 配置，HOST2.0 选择 EHCI Driver 配置，HOST3.0 选择 XHCI Driver 配置。

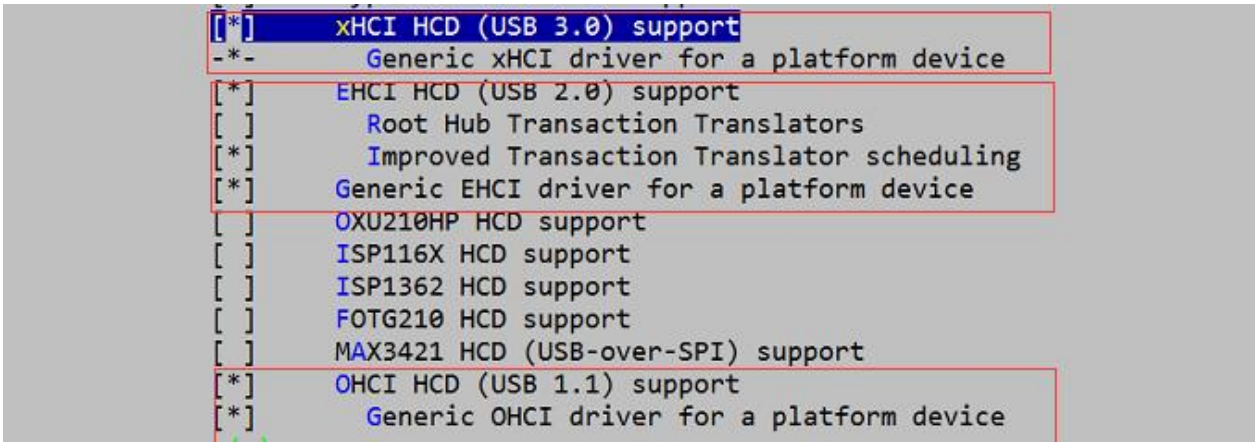


图 3-2 Host Driver 内核的配置选项

3.3 USB OTG 相关配置

Device Drivers --->

[*] USB support --->

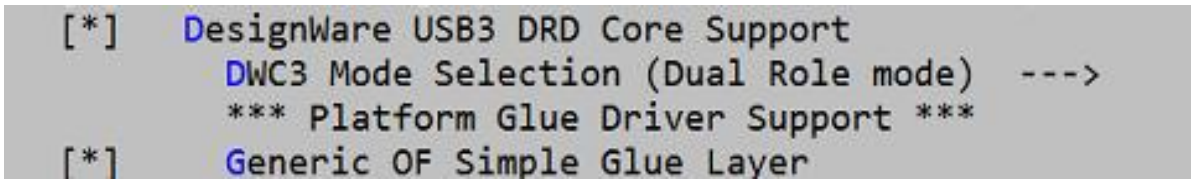


图 3-3 USB3 OTG 内核配置选项

3.4 USB Gadget 配置

Make menuconfig 得到 Kernel 配置界面后，USB 模块的配置位于

Device Drivers --->
[*] USB support --->
[*] USB Gadget Support --->
USB Gadget Drivers (USB functions configurable through configfs) --->

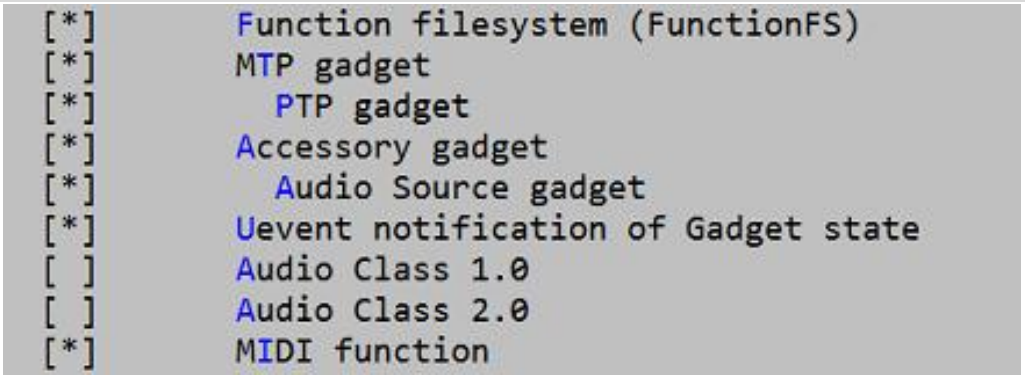


图 3-4 USB Gadget Driver 配置示意图

RK3399 目前支持 MTP、PTP、Accessory、ADB、MIDI、Audio 等 Gadget 功能。

3.5 USB 其它模块配置

3.5.1 Mass Storage Class（MSC）

U 盘属于 SCSI 设备，所以在配置 USB 模块之前需要配置 SCSI 选项（默认配置已经选上）。

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

图 3-5 SCSI 配置

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

```
[*]     USB Mass Storage support
[ ]       USB Mass Storage verbose debug
```

图 3-6 MSC 配置

3.5.2 USB Serial Converter

- 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：

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

- 支持 PL2303

如果要使用 PL2303，输出数据到串口，需要选择如下选项：

```
[*]   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

USB 键鼠的配置选项如下：

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

3.5.4 USB Net

- USB Bluetooth

```
[*] Networking support  --->
    [*]   Bluetooth subsystem support  --->
        Bluetooth device drivers  --->
        [*] HCI USB driver
```

- USB Wifi

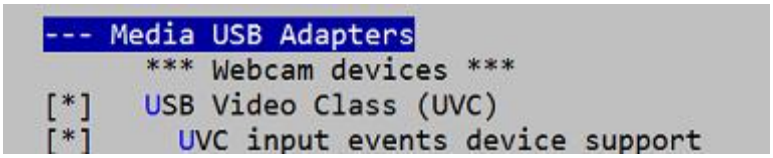
通常直接使用 Vendor 提供的驱动和配置。

- USB Ethernet

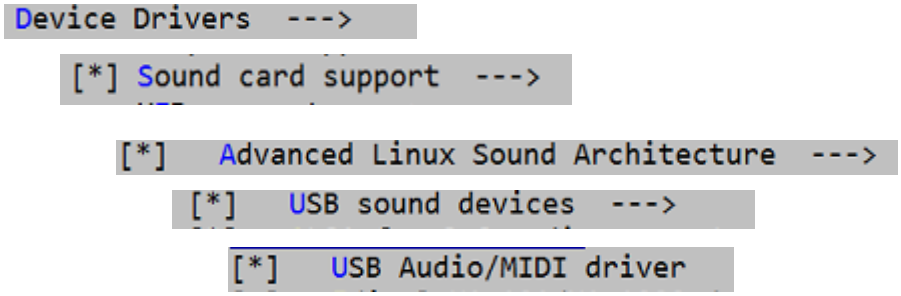
```
Device Drivers  --->
    [*] Network device support  --->
        [*]   USB Network Adapters  --->
[*]   USB CATC NetMate-based Ethernet device support
[*]   USB KLSI KL5USB101-based ethernet device support
[*]   USB Pegasus/Pegasus-II based ethernet device support
[*]   USB RTL8150 based ethernet 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
```

3.5.5 USB Camera

```
Device Drivers  --->
    [*] Multimedia support  --->
        [*]   Media USB Adapters  --->
```

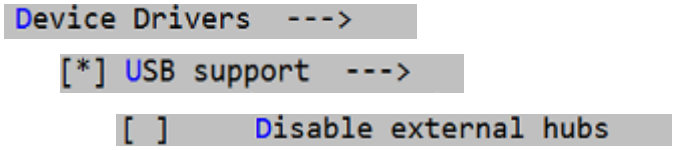


3.5.6 USB Audio



3.5.7 USB HUB

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



3.5.8 其他 USB 设备配置

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

USB2.0 PHY 的配置主要包括 PHY 的时钟、中断配置和 VBUS Supply 的配置。

USB3.0 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 USB2.0 PHY DTS

USB2.0 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。

如下图 4-1 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 = <0x0e450 0x10>;
                clocks = <&cru SCLK_USB2PHY0_REF>;
                clock-names = "phyclk";
                #clock-cells = <0>;
                clock-output-names = "clk_usbphy0_480m";

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

图 4-1 USB2.0 PHY DTSI 配置示意图

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。如下图 4-2 所示，为 DTS 的配置。


```
&u2phy0_host {
        phy-supply = <&vbus_host>;
        status = "okay";
};
```

图 4-2 USB2.0 PHY DTS 配置示意图

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。

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

图 4-3 USB2.0 PHY VBUS Regulator 配置示意图

如上图 3-3 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 Linux Kernel 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>;
        };
};
```

图 4-4 USB2.0 PHY VBUS Pinctrl 配置示意图

如上图 3-4 所示，这是 host vbus-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 USB3.0 PHY DTS

RK3399 USB3.0 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 = <0>;
        extcon = <&fusb0>;
        clocks = <&cru SCLK_UPHY0_TCPDCORE>,
                 <&cru SCLK_UPHY0_TCPDPHY_REF>;
        clock-names = "tcpdcore", "tcpdphy-ref";
        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,external-psm = <0xe588 14 30>;
        rockchip,pipe-status = <0xe5c0 0 0>;
        rockchip,uphy-dp-sel = <0x6268 19 19>;
};
```

图 4-5 USB3 Type-phy dts 配置

4.2 USB2.0 Controller DTS

USB2.0 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 USB2.0 HOST Controller DTS

如下图 3-6 所示，为 RK339 上一个 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>;
        clocks = <&cru HCLK_HOST0>, <&cru HCLK_HOST0_ARB>;
        clock-names = "hclk_host0", "hclk_host0_arb";
        phys = <&u2phy0_host>;
        phy-names = "usb2_phy0";
        status = "disabled";
};
```

图 4-6 EHCI DTS 配置示意图

如下图 3-7 所示，为 RK3399 上一个 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>;
        clocks = <&cru HCLK_HOST0>, <&cru HCLK_HOST0_ARB>;
        clock-names = "hclk_host0", "hclk_host0_arb";
        status = "disabled";
};
```

图 4-7 EHCI DTS 配置示意图

4.3 USB3.0 Controller DTS

4.3.1 USB3.0 HOST Controller DTS

USB3.0 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 USB3.0 OTG Controller DTS

USB3.0 OTG 的详细配置方法，请查看：

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

Example：

```
usbdrd3_0: usb@fe800000 {
        compatible = "rockchip,dwc3";
        clocks = <&cru SCLK_USB3OTG0_REF>, <&cru SCLK_USB3OTG0_SUSPEND>,
                 <&cru ACLK_USB3OTG0>, <&cru ACLK_USB3_RKSOC_AXI_PERF>,
                 <&cru ACLK_USB3>, <&cru ACLK_USB3_GRF>;
        clock-names = "clk_usb3otg0_ref", "clk_usb3otg0_suspend",
                      "aclk_usb3otg0", "aclk_usb3_rksoc_axi_perf",
                      "aclk_usb3", "aclk_usb3_grf";
        #address-cells = <2>;
        #size-cells = <2>;
        ranges;
        status = "disabled";
        usbdrd_dwc3_0: dwc3@fe800000 {
                compatible = "snps,dwc3";
                reg = <0x0 0xfe800000 0x0 0x100000>;
                interrupts = <GIC_SPI 105 IRQ_TYPE_LEVEL_HIGH>;
                dr_mode = "otg";
                phys = <&u2phy_otg>, <&tcphy0>;
                phy-names = "usb2-phy", "usb3-phy";
                snps,dis_enblslpm_quirk;
                snps,phyif_utmi_16_bits;
                snps,dis_u2_freeclk_exists_quirk;
                snps,dis_del_phy_power_chg_quirk;
                snps,xhci_slow_suspend_quirk;
                status = "disabled";
        };
```

图 4-8 USB3 OTG dts 配置

5 驱动开发

目前，EHCI、OHCI、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；此外，USB2.0 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 USB2.0 PHY driver

Driver 代码路径：drivers/phy/phy-rockchip-inno-usb2.c

RK3399 USB2.0 PHY 采用 Innosilicon IP，SoC 上有两个 USB2.0 的 PHY，每个 PHY 有两个 port，一个 port 用于支持 USB2.0 HOST 控制器，另一个 port 用于支持 USB2.0 OTG 控制器。

对于 Innosilicon IP USB2.0 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。如下图 4-1，是一个典型的 RK3399 USB2.0 host port 的配置。

```
static const struct rockchip_usb2phy_cfg rk3399_phy_cfgs[] = {
        {
                .reg = 0xe450,
                .num_ports      = 2,
                .clkout_ctl     = { 0xe450, 4, 4, 1, 0 },
                .port_cfgs      = {
                        [USB2PHY_PORT_HOST] = {
                                .phy_sus        = { 0xe458, 1, 0, 0x10, 0x01 },
                                .ls_det_en      = { 0xe3c0, 6, 6, 0, 1 },
                                .ls_det_st      = { 0xe3e0, 6, 6, 0, 1 },
                                .ls_det_clr     = { 0xe3d0, 6, 6, 0, 1 },
                        }
                },
        },
```

图 5-1 RK3399 USB2.0 PHY host-port 配置

图 4-2 为 RK3399 USB2.0 otg-port phy 配置参考：


```
static const struct rockchip_usb2phy_cfg rk3399_phy_cfgs[] = {
        {
                .reg            = 0xe450,
                .num_ports      = 2,
                .clkout_ctl     = { 0xe450, 4, 4, 1, 0 },
                .port_cfgs      = {
                        [USB2PHY_PORT_OTG] = {
                                .phy_sus        = { 0xe454, 1, 0, 2, 1 },
                                .bvalid_det_en  = { 0xe3c0, 3, 3, 0, 1 },
                                .bvalid_det_st  = { 0xe3e0, 3, 3, 0, 1 },
                                .bvalid_det_clr = { 0xe3d0, 3, 3, 0, 1 },
                                .utmi_bvalid    = { 0xe2ac, 7, 7, 0, 1 },
                        },
                },
                .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 },
```

图 5-2 RK3399 USB2.0 otg-port phy 配置

5.1.2 USB3.0 PHY driver

Driver 代码路径：drivers/phy/phy-rockchip-typec.c

// TODO:

5.2 USB3.0 OTG drivers

Driver 代码路径：

drivers/usb/dwc3/*

drivers/usb/host/xhci*

// TODO:

6 Android Gadget 配置

Linux Kernel4.0，Android5.0 及其后版本，Gadget 均采用 ConfigFs 配置，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 Gadget 驱动配置

请参阅 2.4 章节。

6.2 BOOT IMG 配置

在 Android boot.img 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、init.usb.configfs.rc 为 Android 标准 rc 文件，一般不需要改动。
2) init.rk30board.usb.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，Gadget 描述符的定义（VID/PID）、Gadget function 节点的定义等，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
```

图 6-1 Android Gadget init.rk30board.usb.rc

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"
```

内核提供了设备节点来查看 USB Gadget 的关键配置信息，在根目录如下：

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

大部分节点的功能，可以直观地看出来，这里就不在赘述。

“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
```

3) fstab.rk30board.bootmode.emmc 为 Android fstab 文件，可以用于配置 sdcard、usb 的 mount 路径，RK3399 平台的 vold 和 kernel 已经可以做到自动搜索和匹配 usb mount 路径，不需要再做修改。

```
    # for usb2.0
    /devices/platform/*.usb*              auto vfat defaults
voldmanaged=usb:auto
    # for usb3.0
    /devices/platform/usb@*/*.dwc3*       auto vfat defaults
voldmanaged=usb:auto
```

7 常见问题分析

7.1 设备枚举日志

7.1.1 USB2.0 OTG 正常开机日志

```
开机未连线，默认为 device 模式。
[    8.764441] [otg id chg] last id -1 current id 67108864
[    8.764925] PortPower off
[    8.866923] Using Buffer DMA mode
[    8.867280] Periodic Transfer Interrupt Enhancement - disabled
[    8.867787] Multiprocessor Interrupt Enhancement - disabled
[    8.868294] OTG VER PARAM: 0, OTG VER FLAG: 0
[    8.868700] ^^^^^^^^^^^^^^^^^^Device Mode
```

7.1.2 USB2.0 Device 连接

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

7.1.3 USB2.0 Device 断开连接

```
[  187.085682] ********session end ,soft disconnect***********
[  187.086486] android_work: sent uevent USB_STATE=DISCONNECTED
[  187.087217] mtp_release
```

7.1.4 USB2.0 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 Optical Mouse
[  325.621222] usb 2-1: Manufacturer: Logitech
```


7.1.5 USB2.0 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: Memsart controller
    [   371.111251] usb 2-1: Manufacturer: Memsart
    [   371.123172] input: Memsart Memsart controller as /
```

7.1.6 USB2.0 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 Mass Storage device detected
    [   405.542472] scsi host1: usb-storage 1-1:1.0
    [   406.584573] scsi 1:0:0:0: Direct-Access Kingston DT R400 PMAP PQ: 0 ANSI: 0
CCS
    [   406.586425] sd 1:0:0:0: Attached scsi generic sg0 type 0
    [   408.171256] sd 1:0:0:0: [sda] 15646720 512-byte logical blocks: (8.01
GB/7.46 GiB)
    [   408.172788] sd 1:0:0:0: [sda] Write Protect is off
    [   408.173970] sd 1:0:0:0: [sda] No Caching mode page found
    [   408.174453] sd 1:0:0:0: [sda] Assuming drive cache: write through
    [   408.223001]  sda: sda1
    [   408.229280] sd 1:0:0:0: [sda] Attached SCSI removable disk
```

7.1.7 USB2.0 HOST-LS/FS/HS 设备断开 log

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

7.1.8 USB3.0 Device 连接

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

7.1.9 USB3.0 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 Mass Storage 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] 1953525167 512-byte logical blocks: (1.00
TB/932 GiB)
    [    27.754245] sd 0:0:0:0: [sda] Write Protect is off
    [    27.754982] sd 0:0:0:0: Attached scsi generic sg0 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 SCSI disk
```

7.2 USB 常见问题分析

7.2.1 软件配置

首先必须明确项目中 USB 控制器是如何分配的，并确保 kernel 的配置是正确的，请参考第二章配置说明，需要根据项目的实际使用情况进行配置。主要注意下面几点：

1、如果使用 USB2.0 HOST 控制器，请配置对 EHCI/OHCI 配置，否则不支持。

2、OTG // TODO:

7.2.2 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 USB2.0 HOST 控制器作为 HOST 和 USB2.0 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 USB2.0 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 USB2.0 OTG 控制器，VBUS 由 PC 通过 USB 线提供。

场景二:

使用一个 USB2.0 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，USB2.0 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。

7.2.3 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 USB2.0 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。

常见异常排查：

1、 连接 USB 时串口完全没有 log：

(1) USB 硬件信号连接正确；

(2) USB 控制器确保工作在 device 状态；

(3) 测量 USB_DET 信号电压，USB 连接时应该由低到高。

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: ********soft connect!!!****************************************
[36.688603] DWC_OTG: USB SUSPEND
[36.807373] DWC_OTG: USB RESET
```

但是没有正常 log 中的后面几条信息。

一般为 USB 硬件信号差，无法完成枚举。

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

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。

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。

可能原因是：

(1) USB 信号质量差。可测试下 USB 眼图，并使用 USB 分析仪抓取数据流后分析。

(2) 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。

(3) 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**

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, soft disconnect**********************
```

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. 4Host 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；

常见异常及排查：

1. HOST 口接入设备后，串口无任何打印：

(1) 首先需要确认通过电子开关后的电路连接正确；

(2) 确认控制器工作于 HOST 状态，并确认供电电路正常。

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。

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

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。

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。

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-byte 0xd6eab00a
DWC_OTG: dwc_otg_hcd_urb_enqueue urb->transfer_buffer address not align to 4-byte 0xccf6140a
```

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。

5. OTG 口作为 host 时，无法识别接入的设备

(1) 检查 kernel 的 OTG 配置是否正确；

(2) 检查 OTG 电路的 ID 电平(作 host，为低电平)和 VBUS 5V 供电是否正常；

(3) 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。

7.2.5 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。

7.2.6 USB 充电检测

RK3399 USB2 PHY 支持 BC1.2 标准的充电检测，代码实现请参考 drivers/phy/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 USB2.0 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(USB3.0 150mA);而当外设已经配置(configured)并且未休眠时，最大可从 VBUS 获得 500mA(USB3.0 900mA)电流。

- CDP —— Charging Downstream Port

即兼容 USB2.0 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：

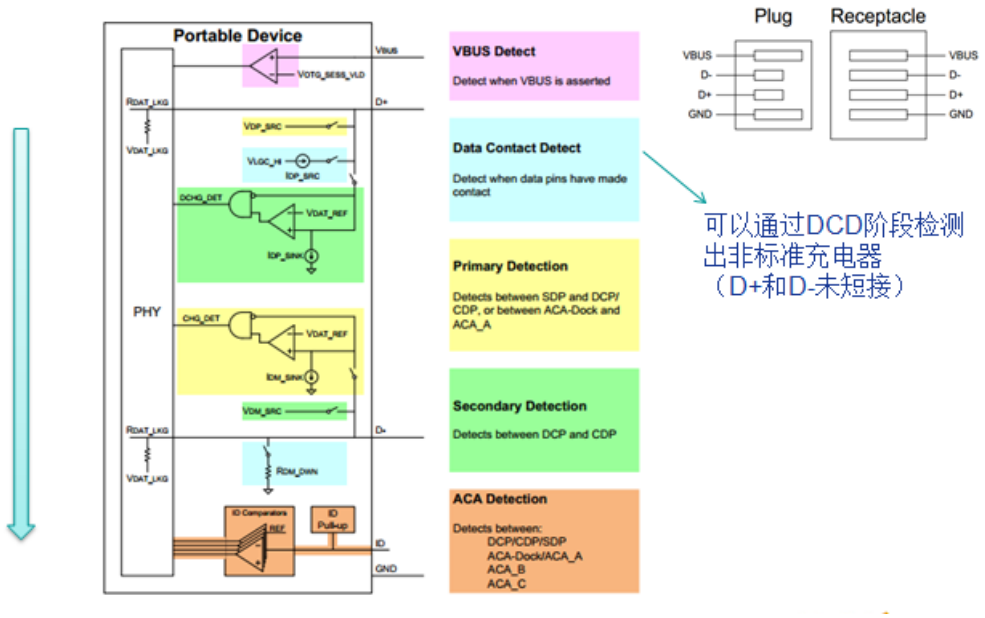


图 7-1 USB 充电检测流程

典型的 SDP 检测过程中，D+/D-波形如下图所示：

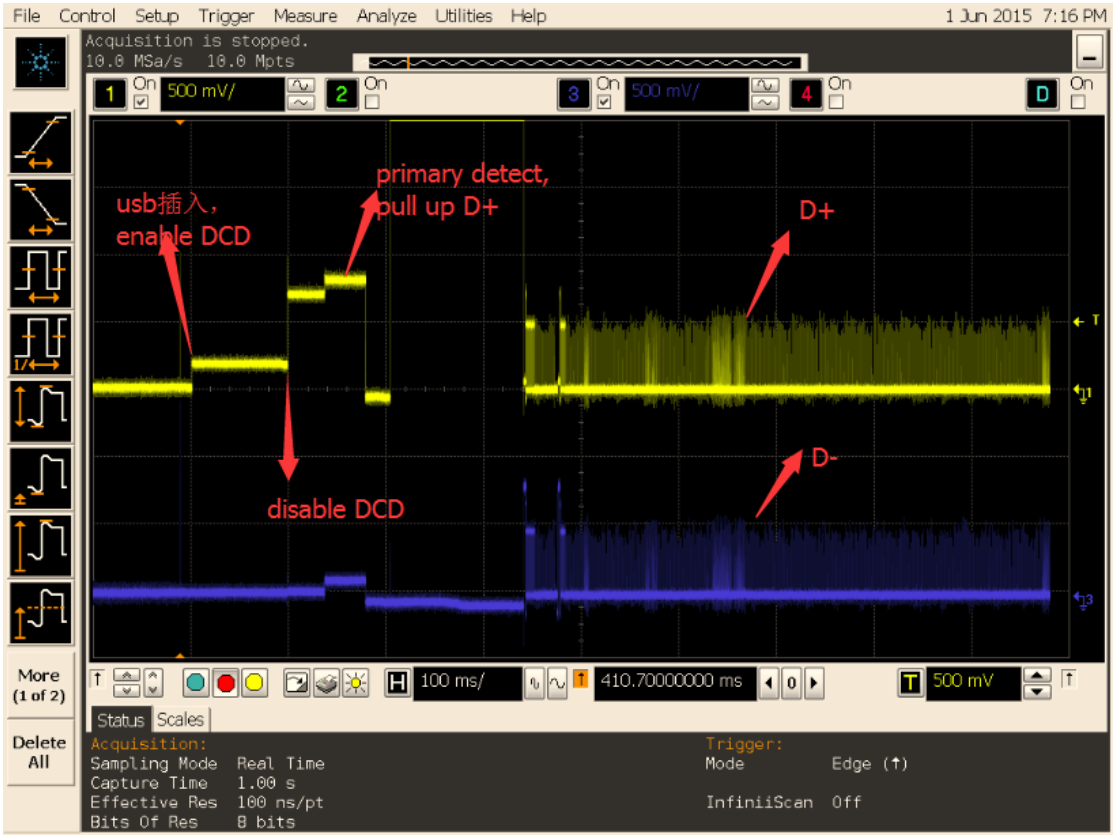


图 7-2 SDP 检测波形

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

USB2.0/3.0 信号测试方法及常见问题分析请参阅《RK USB Compliance Test Note V1.2》