

Rockchip RK356x USB 开发指南

文件标识: RK-SM-YF-142

发布版本: V1.4.0

日期: 2024-04-24

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自所有者所有。

版权所有© 2024瑞芯微电子股份有限公司

超越合理使用范畴，非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址: 福建省福州市铜盘路软件园A区18号

网址: www.rock-chips.com

客户服务电话: +86-4007-700-590

客户服务传真: +86-591-83951833

客户服务邮箱: fae@rock-chips.com

前言

概述

本文档提供 RK356x USB 模块的开发指南，目的是让工程师理解 RK356x USB 控制器和 USB PHY 的硬件设计和软件驱动设计，以便开发者根据产品的 USB 应用需求进行灵活设计和快速开发。

芯片名称	内核版本
RK3566、RK3568	Linux-4.19及以上版本

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

硬件开发工程师

修订记录

日期	版本	作者	修改说明
2021-01-18	V1.0.0	吴良峰	初始版本
2021-03-10	V1.1.0	吴良峰	1. 根据 Rockchip 文档编写规范，修改文件标识 2. 修改 USB 2.0 PHYs 功耗优化方法
2021-03-15	V1.2.0	吴良峰	增加 USB 2.0 PHYs 功耗优化的软件修改说明
2022-09-26	V1.3.0	吴良峰	1. 增加 Type-C USB PD 硬件电路说明 2. 增加 Linux-4.19/5.10 Type-C USB DTS 配置说明 3. 增加 Type-C 控制器芯片支持列表
2024-04-24	V1.4.0	吴良峰	新增用于 USB2 only 方案的 DTS 属性及说明

目录

Rockchip RK356x USB 开发指南

1. RK356x USB 控制器和 PHY 简介
2. RK356x USB 硬件电路设计
 - 2.1 RK356x USB 2.0/3.0 供电及功耗控制
 - 2.1.1 RK356x USB 2.0/3.0 控制器供电及功耗控制
 - 2.1.2 RK356x USB 2.0 PHYs 供电及功耗控制
 - 2.1.3 RK356x USB 3.0 PHYs 供电及功耗控制
 - 2.2 RK3566 USB 2.0 OTG Micro-B 接口硬件电路
 - 2.3 RK3566 USB 2.0 OTG Type-C 接口硬件电路
 - 2.3.1 RK3566 Type-C USB 2.0 only 硬件电路设计[No PD]
 - 2.3.2 RK3566 Type-C USB 2.0 + PD 硬件电路
 - 2.4 RK356x USB 2.0 Host Type-A 接口硬件电路
 - 2.5 RK3568 USB 3.0 OTG Type-A 接口硬件电路
 - 2.6 RK3568 USB 3.0 OTG Type-C 接口硬件电路
 - 2.7 RK356x USB 3.0 HOST Type-A 接口硬件电路
3. RK356x USB DTS 配置
 - 3.1 RK3568 USB 控制器和 PHY DTSI 节点
 - 3.2 RK3568 USB 3.0 OTG to USB 2.0 only 配置
 - 3.3 RK356x USB 3.0 Host to USB 2.0 only 配置
 - 3.4 RK356x Type-C USB DTS 配置
 - 3.4.1 Linux-4.19 RK3566 Type-C USB 2.0 + FUSB302 DTS 配置
 - 3.4.2 Linux-4.19 RK3566 Type-C USB 2.0 + HUSB311 DTS 配置
 - 3.4.3 Linux-5.10 RK3566 Type-C USB 2.0 + FUSB302 DTS 配置
 - 3.4.4 Linux-5.10 RK3566 Type-C USB 2.0 + HUSB311 DTS 配置
 - 3.5 USB VBUS 配置
 - 3.6 Linux USB DT 配置的注意点
 - 3.6.1 USB DT 重要属性说明
 - 3.6.1.1 USB 3.1 控制器 DT 属性
 - 3.6.1.2 USB2 PHY DT 属性
 - 3.6.1.3 USB 3.0/SATA/QSGMII ComboPHY DT 属性
4. RK356x USB OTG mode 切换命令
5. Type-C 控制器芯片支持列表
6. 参考文献

1. RK356x USB 控制器和 PHY 简介

RK356x USB 控制器支持列表如下表 1

表 1 RK356x USB 控制器列表

芯片/控制器	USB 2.0 HOST (EHCI/OHCI)	USB 3.0/2.0 OTG (DWC3/xHCI)
RK3566	2	2 (OTG 2.0 + Host 3.0)
RK3568	2	2 (OTG 3.0+ Host 3.0)

RK356x USB PHY 支持列表如下表 2

表 2 RK356x USB PHY 列表

芯片/PHY	USB 2.0 ComboPHY	USB 3.0/SATA/QSGMII ComboPHY
RK3566	2 [2 × port]	1
RK3568	2 [2 × port]	2

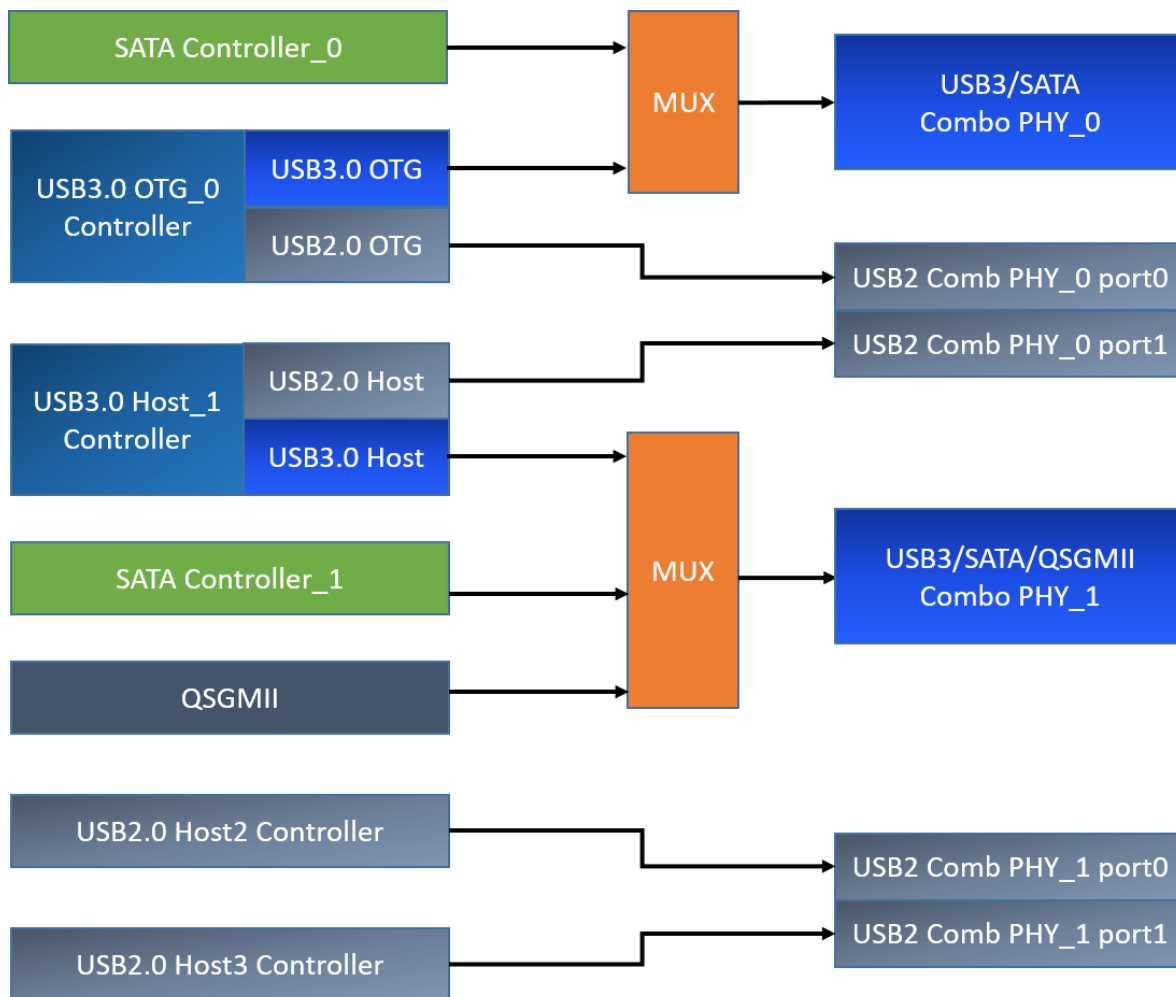
Note:

1. 表格中，数字 N 表示支持 N 个独立的 USB 控制器；
2. 表格中，[2 × ports] 表示一个 PHY 支持两个 USB port；
3. 表格中，“EHCI/OHCI”表示该 USB 控制器集成了 EHCI 控制器和 OHCI 控制器；“DWC3/xHCI”表示该 USB 控制器集成了 DWC3 控制器和 xHCI 控制器；
4. RK3566 的 DWC3 OTG 控制器只支持 OTG 2.0，不支持 OTG 3.0，也即最高只能支持 USB 2.0 480Mbps 传输；

RK3566 和 RK3568 的 USB 模块区别是，RK3568 支持 OTG 3.0，而 RK3566 只支持 OTG 2.0，但都支持 1 个 USB 3.0 Host，2 个 USB 2.0 Host。如下图 1-1 是 RK3568 USB 控制器和 PHY 的连接示意图。由图 1-1 可以看出：

1. USB 3.0 OTG 控制器与 SATA_0 控制器复用 USB3/SATA Combo PHY_0；
2. USB 3.0 Host_1 控制器与 SATA_1/QSGMII 控制器复用 USB3/SATA/QSGMII Combo PHY_1；
3. USB 3.0 OTG 控制器与 USB 3.0 Host_1 控制器分别使用 USB 2.0 Comb PHY_0 的 port0 和 port1；
4. USB 2.0 Host_2 控制器与 USB 2.0 Host_3 控制器分别使用 USB 2.0 Comb PHY_1 的 port0 和 port1；

需要注意的是，USB3/SATA Combo PHY_0 和 USB3/SATA/QSGMII Combo PHY_1 在同一时刻，只能支持一种工作模式，也即 USB3 与 SATA, QSGMII 接口是互斥的，开发者可以根据产品形态，灵活配置内核的 DTS, 使能对应的外设接口。USB 的 DTS 配置方法，请参考 [RK356x USB DTS 配置](#)。



RK3568 USB2.0/3.0 Controllers and PHYs hardware framework

图 1-1 RK3568 USB 控制器和 PHY 的连接示意图

2. RK356x USB 硬件电路设计

本章节主要说明 RK356x USB 在实际应用中，可以支持不同硬件电路设计方案。如下图 2-1 是 RK3568 USB 接口框图，可以看到，RK3568 USB 总共支持 4 个 USB 外设接口，包括 1 个 USB 3.0 OTG 接口，1 个 USB 3.0 Host 接口，以及 2 个 USB 2.0 Host 接口。而 RK3566 USB 同样支持 4 个 USB 外设接口，包括 1 个 USB 2.0 OTG 接口，1 个 USB 3.0 Host 接口，以及 2 个 USB 2.0 Host 接口。

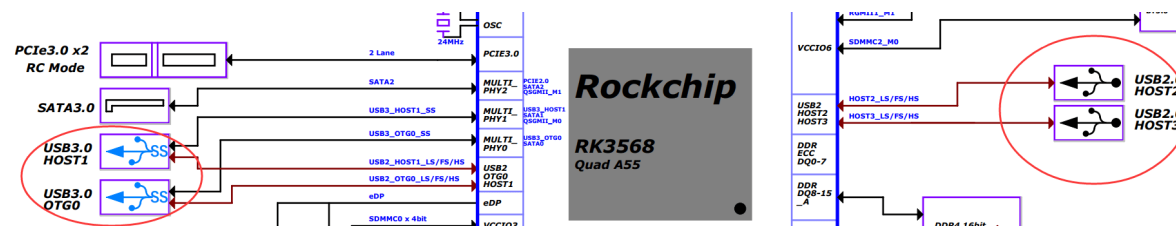


图 2-1 RK3568 原理图的USB 外设接口

2.1 RK356x USB 2.0/3.0 供电及功耗控制

功耗控制的原则是：

1. 硬件电路设计中，如果可以直接断开 USB 2.0 PHY或者 Combo PHY 的供电，则 PHY 功耗为0，PHY 对应的 DTS 节点配置为disable 即可；
2. 硬件电路设计中，如果必须保证 USB 2.0 PHY 的供电，则 **DTS** 中需要将 **PHY** 对应的两个 **port** 节点都配置为**enable** 状态，以便 PHY 驱动中设置不使用的 port 进入suspend mode；
3. 不使用的 USB 控制器，对应的控制器 DTS 节点配置为 disable 即可；
4. 不使用的 Combo PHY，对应的 Combo PHY DTS 节点配置为 disable 即可；

2.1.1 RK356x USB 2.0/3.0 控制器供电及功耗控制

RK356x USB 2.0/3.0 控制器的供电是 VDD_LOGIC，并且 USB 3.0 控制器与 PCIE/SATA/XPCS 一起使用芯片内部的 power domain PD_PIPE，可以根据 USB 3.0 接口的工作情况，动态控制 PD_PIPE 的开关，以降低 USB 3.0 控制器的功耗。但 USB 2.0 控制器没有 power domain。

需要注意的是，RK356x Linux 内核已经支持 USB 3.0 OTG 的 PD_PIPE 动态开关，以及支持 USB Host auto suspend功能（指 USB Host 接口不接任何外设时，Host 控制器自动进入 suspend 低功耗状态），因此开发者不用对 USB 2.0/3.0 控制器的功耗控制进行调试工作。

如果产品上不需要使用所有的 USB 接口，建议在内核中，将未使用的 USB 控制器的 DTS 配置为 disable，以降低 USB 控制器的功耗。如下是 disable USB 2.0 Host_2 和 USB 2.0 Host_3 的方法：

```
&usb_host0_ehci {
    status = "disabled";
};

&usb_host0_ohci {
    status = "disabled";
};

&usb_host1_ehci {
    status = "disabled";
};

&usb_host1_ohci {
    status = "disabled";
};
```

2.1.2 RK356x USB 2.0 PHYs 供电及功耗控制

RK356x 包含 2 个 USB 2.0 Combo PHY。其中，USB OTG 和 USB Host_1 使用USB 2.0 Comb PHY_0；USB Host_2 和 USB Host_3 使用 USB 2.0 Comb PHY_1。每一个 USB 2.0 Combo PHY 的供电有三路：3.3V，1.8V 和 0.9V，如下表 3 所示。

需要注意的是，实际电路中，这三路电压值超过规定的最大值或者低于规定的最小值（电压最大允许上下波动±10%），都可能会导致 USB 连接异常。

表 3 USB2.0 PHY power supplies

Supply Voltage	Min	Typ	Max	Unit
USB_AVDD_3V3	3.0	3.3	3.6	V
USB_AVDD_1V8	1.62	1.8	1.98	V
USB_AVDD_0V9	0.81	0.9	0.99	V

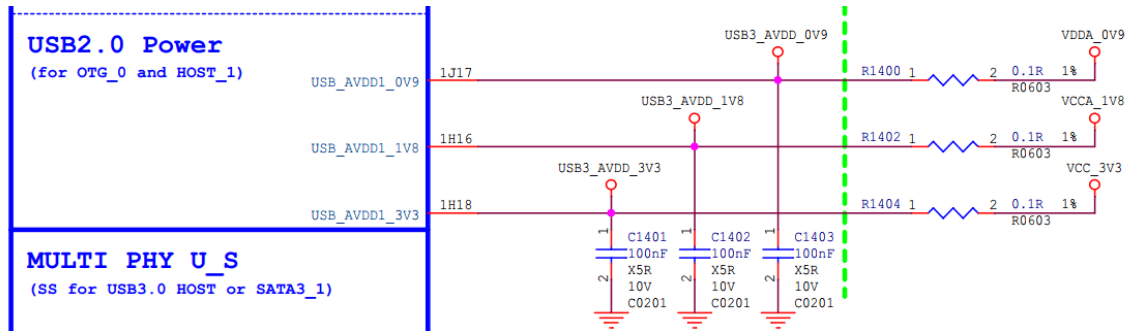


图 2-2 USB 2.0 Comb PHY_0 (for OTG and Host_1) 供电电路

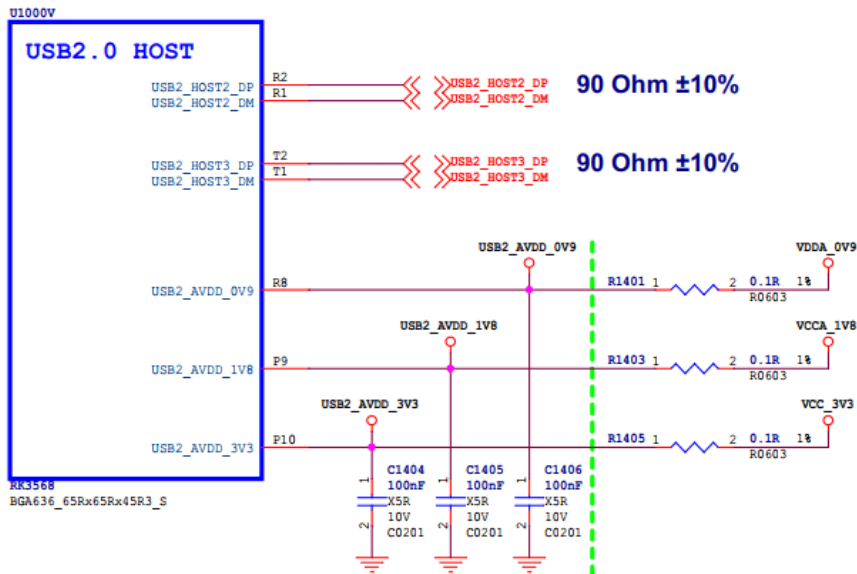


图 2-3 USB 2.0 Comb PHY_1 (for Host_2 and Host_3) 供电电路

上电后，USB 2.0 Comb PHY 的默认配置是处于正常工作的状态，因此需要在 USB PHY 驱动中做动态功耗控制（PHY 驱动代码已经支持），或者断开 USB PHY 的供电。需要注意的是，**USB 2.0 Comb PHY_0** 用于 **USB OTG**，因为至少需要支持 **USB** 下载固件的功能，所以 **USB 2.0 Comb PHY_0** 的供电一定不能断。

在 RK356x 实际的产品中，一种常见 USB 2.0 PHY 功耗优化场景是：

产品只需要支持 USB OTG 功能，不需要 USB 3.0 Host_1/USB 2.0 Host_2/USB 2.0 Host_3，建议 USB 2.0 PHY 功耗优化如下：

1. 在内核 DTS 中，disable u2phy0_host 节点

目的是，避免在 PHY 驱动的 probe 流程中，执行 u2phy0_host 的初始化，导致 u2phy0_host 功耗增加。

2. 断开 USB 2.0 Comb PHY_1 的供电

如下图 2-4 的硬件电路所示，将 Comb PHY_1 的三路供电接地。

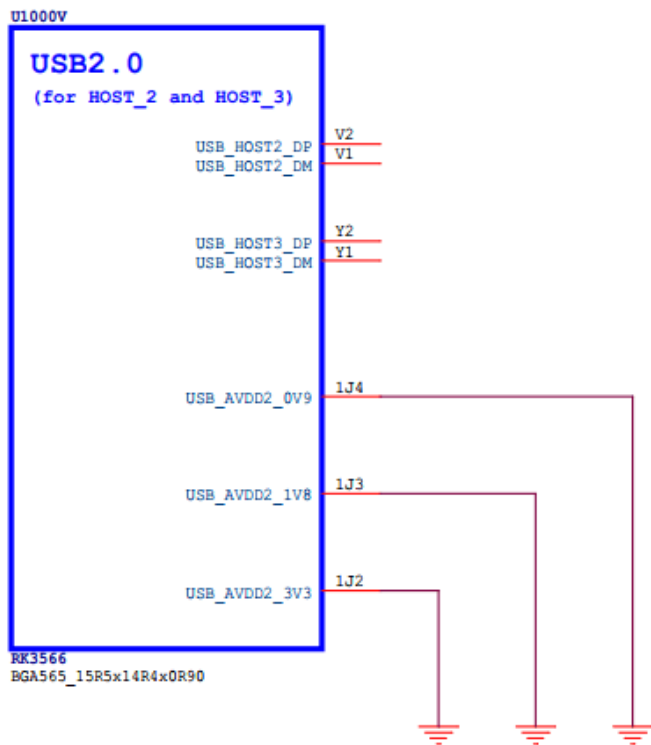


图 2-4 USB 2.0 Comb PHY_1 (for Host_2 and Host_3) 断开供电电路

因为 Comb PHY_1 对一个 USB 2.0 Host2 和 USB 2.0 Host3，所以，软件上需要相应修改内核 DTS，必须 disable 内核 DTS 中的 PHY 节点 u2phy1_host 和 u2phy1_otg，以及对应的 USB 控制器节点 usb_host1_ehci 和 usb_host1_ohci，参考方法如下：

```
&usb_host1_ehci {
    status = "disabled";
};

&usb_host1_ohci {
    status = "disabled";
};

&u2phy1_host {
    status = "disabled";
};

&u2phy1_otg {
    status = "disabled";
};
```

需要特别注意的是，在 USB PHY 未供电的情况下，如果没有 disable 内核 DTS 中 PHY 对应的 USB 控制器节点，将会导致内核初始化时，卡死在 USB 控制器的初始化，对应的典型 log 如下：

```
[ 1.240101] ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
[ 1.241896] ohci-platform: OHCI generic platform driver
[ 1.242698] ohci-platform fd8c0000.usb: Generic Platform OHCI controller
[ 1.243686] ohci-platform fd8c0000.usb: new USB bus registered, assigned bus
number 3
```

2.1.3 RK356x USB 3.0 PHYs 供电及功耗控制

RK3566 USB 3.0 Host_1 使用 USB3/SATA/QSGMII Combo PHY_1 (对应原理图中的 MULTI_PHY1);

RK3568 USB 3.0 OTG 使用 USB3/SATA Combo PHY_0 (对应原理图中的 MULTI_PHY0), USB 3.0 Host_1 使用 USB3/SATA/QSGMII Combo PHY_1 (对应原理图中的 MULTI_PHY1);

RK356x MULTI_PHY0/MULTI_PHY1/MULTI_PHY2 使用同样的供电电源, 如下图 2-5 所示, 有两路供电: 0.9V 和 1.8V。对这两路供电电源的电压要求, 请参考表 4。

表 4 USB 3.0 Combo PHY power supplies

Supply Voltage	Min	Typ	Max	Unit
MULTI_PHY_AVDD_1V8	1.62	1.8	1.98	V
MULTI_PHY_AVDD_0V9	0.81	0.9	0.99	V

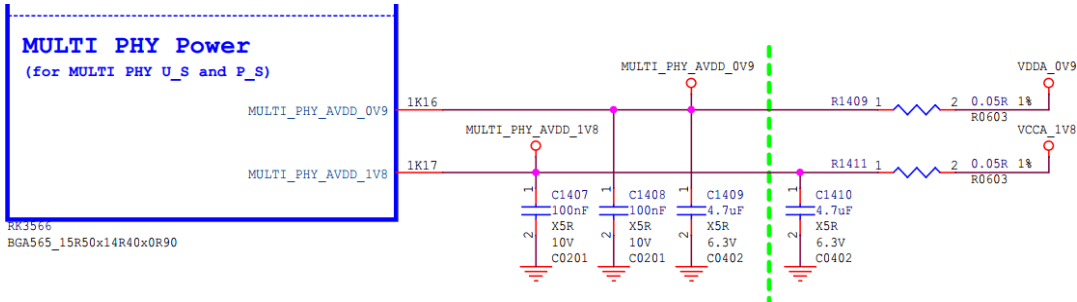


图 2-5 RK356x Multi PHYs 供电电路

芯片上电时, 三个 MULTI_PHY0/MULTI_PHY1/MULTI_PHY2 默认处于工作状态。在 UBoot 中, 会通过软件设置三个 PHY 处于 reset 状态, 以保持 PHY 处于最低功耗。进入内核后, USB 3.0/SATA/PCIE/QSGMII 控制器驱动会调用 rockchip_combphy_init() 函数释放 PHY 的 reset。

当 Combo PHY 工作在 USB 3.0 mode 时, PHY 的 PIPE state (P0/P1/P2/P3) 由 USB 3.0 控制器硬件自动控制, 根据不同的工作场景, 动态进入和退出 P0/P1/P2/P3 state。比如, USB 3.0 Host 未插入任何 USB 设备时, 则 PIPE 处于 P3 state; 插入 U3 disk 时, 则切换到 P0 state; 当接 USB 3.0 HUB 时, 只要 HUB 的下行端口没有接其他 USB 外设, 则 PIPE state 会自动进入 P3 state。当有 USB 外设插入 USB3 HUB, 则 PIPE state 切为 P0。(注: P0 为正常工作状态, P3 为最低功耗状态)

当 Combo PHY 未作任何功能使用时, 建议在内核 DTS 中, 将对应的 USB 3.0 控制器和 PHY 节点 disable, 以保证 PHY 处于最低功耗状态。DTS 的配置方法, 请参考章节 [RK356x USB DTS 配置](#)。

2.2 RK3566 USB 2.0 OTG Micro-B 接口硬件电路

RK3566 USB 2.0 OTG Micro-B 接口硬件电路如下图 2-6 ~ 图 2-8 所示, 与 Rockchip 其他平台的 SDK 中常见的 USB 2.0 OTG 电路设计类似, 因此, 这里不详细介绍电路设计原理。

有三点需要注意:

1. DP/DM 必须分别串接 2.2 Ω 电阻, 提高 DP/DM 的抗压/抗静电能力;
2. 如果要支持 USB Device 动态拔插检测和充电类型检测功能, 则 VBUSDET 脚一定要连接到 USB Micro接口;
3. 如果不需要支持 USB Device 动态拔插检测和充电类型检测功能, 可以允许 VBUSDET 脚固定拉高到 3.3V 或者悬空 (推荐拉高到 3.3V)。由于 RK356x Maskrom USB 的连接不依赖于 VBUSDET 信号的检测, 所以即使 VBUSDET 脚悬空, Maskrom USB 仍然可以正常工作并下载固件;

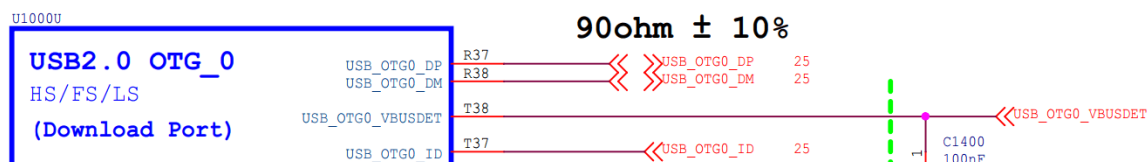


图 2-6 RK3566 USB2.0 OTG Pins

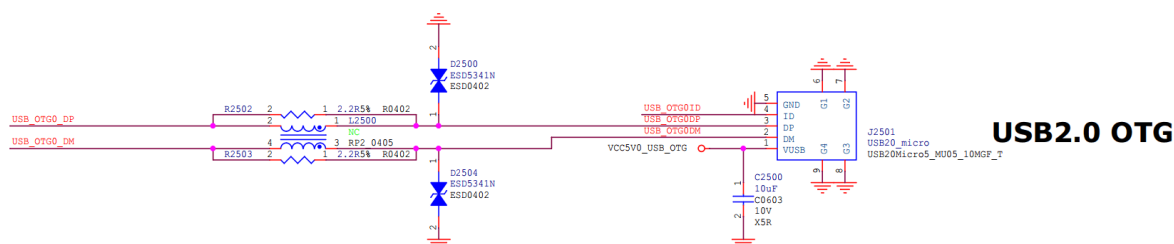


图 2-7 RK3566 USB2.0 OTG Micro-B 接口

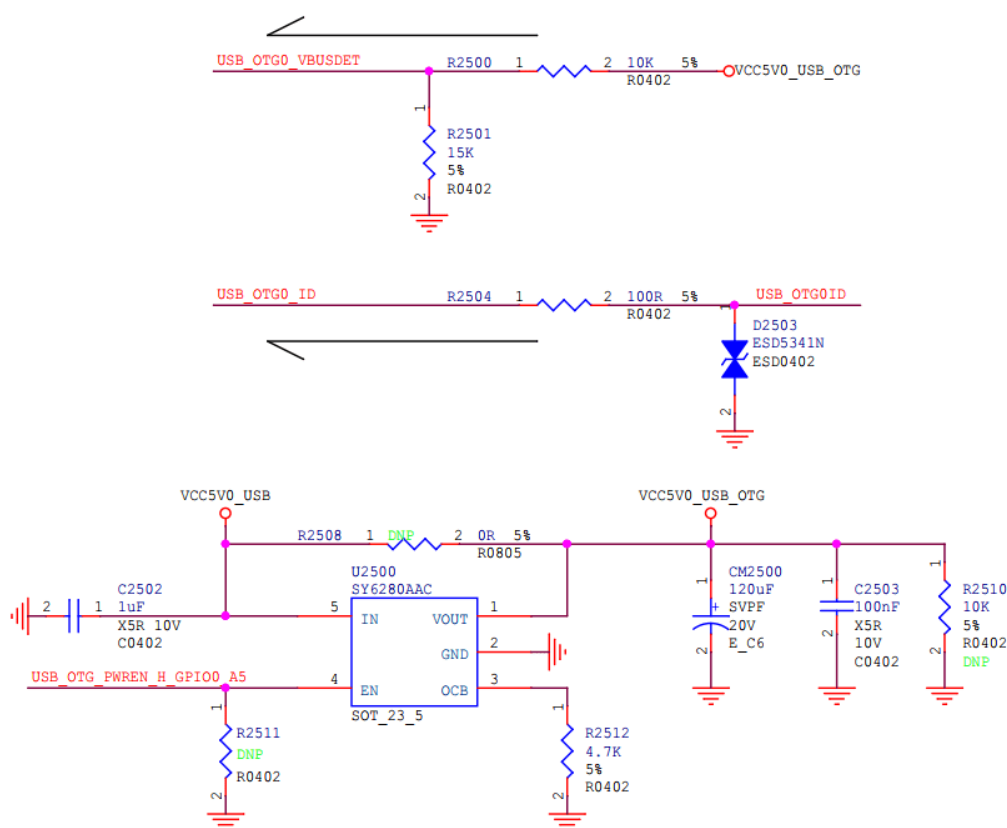


图 2-8 RK3566 USB2.0 OTG VBUS 和 OTG_ID 控制电路

2.3 RK3566 USB 2.0 OTG Type-C 接口硬件电路

2.3.1 RK3566 Type-C USB 2.0 only 硬件电路设计[No PD]

RK3566 Type-C USB2 only 硬件电路设计如下图 2-9 ~ 图 2-11 所示。该电路设计的特点是：

1. 支持 USB 2.0 OTG mode 检测和自动切换；
2. 不支持 Type-C PD (Power delivery)；
3. 不需要 Type-C 外置控制器芯片，节省硬件成本；

RK3566 Type-C USB2 + PD 的硬件电路设计如下图 2-12 ~ 图 2-14 所示。该电路设计的特点是：

1. 支持 USB 2.0 OTG mode 检测和自动切换；
2. 支持 Type-C PD (Power delivery)；
3. 需要 Type-C 外置控制器芯片；
4. 需要 Type-C VBUS 5V 稳压电路（为了支持 PD 高压大电流充电）；

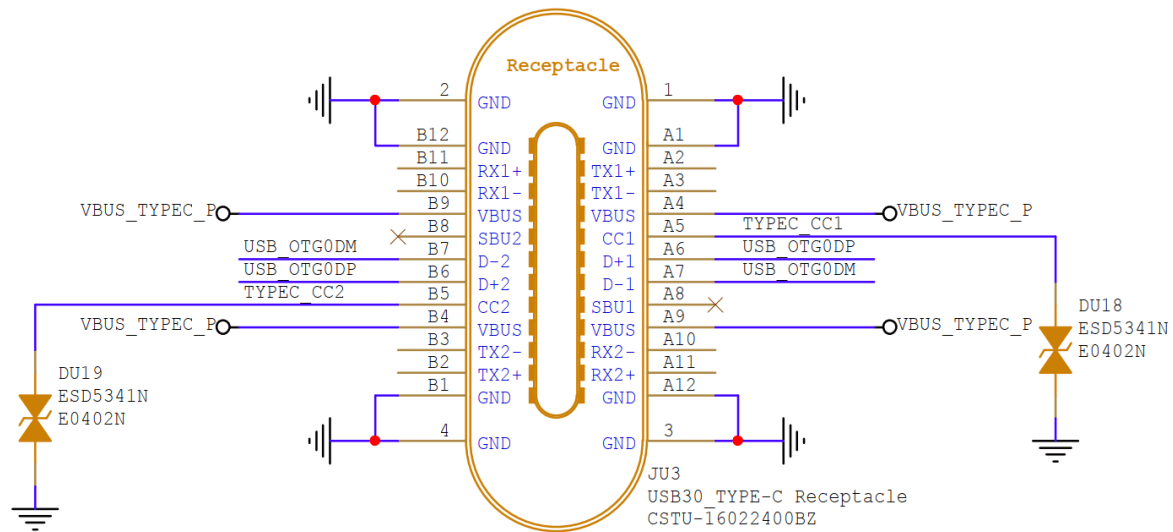


图 2-12 RK3566 Type-C USB 2.0 + PD 接口

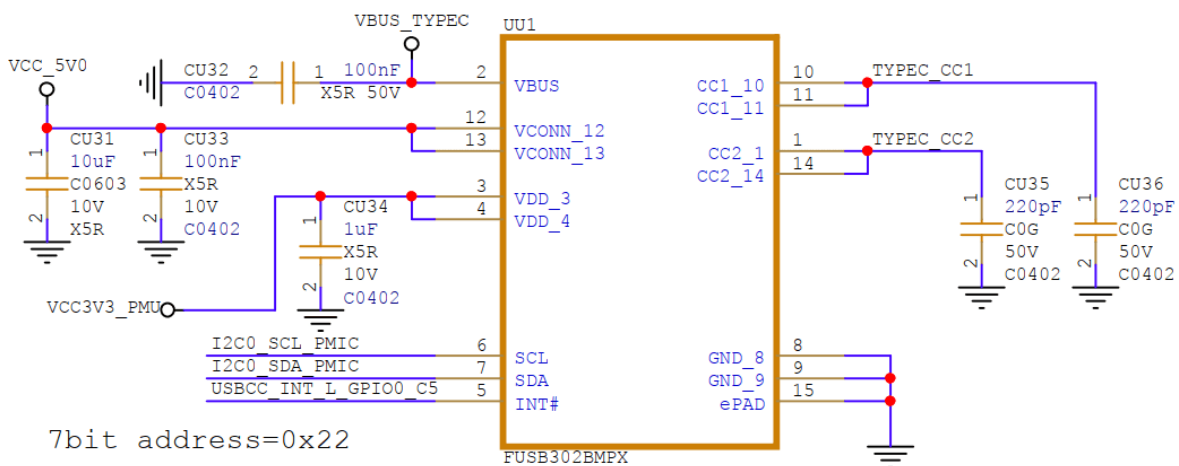


图 2-13 RK3566 Type-C FUSB302 电路

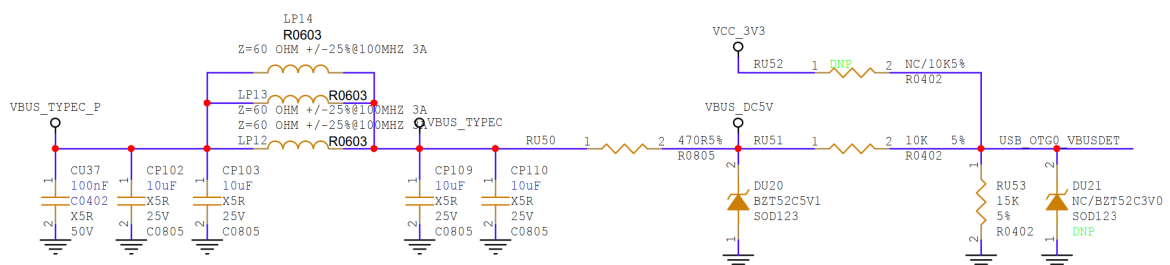


图 2-14 RK3566 Type-C VBUS 5V 稳压电路

2.4 RK356x USB 2.0 Host Type-A 接口硬件电路

RK356x USB 2.0 Host Type-A 接口硬件电路如下图 2-15 和图 2-16 所示，与 Rockchip 其他平台的 SDK 中常见的 USB 2.0 Host 电路设计类似，因此，这里不详细介绍电路设计原理。

需要注意的是，如果产品上不使用 USB 2.0 Host，建议断开 USB 2.0 Host PHY 的供电，具体方法参考章节[RK356x USB 2.0 PHYs 供电及功耗控制](#)和图 5 的电路设计。

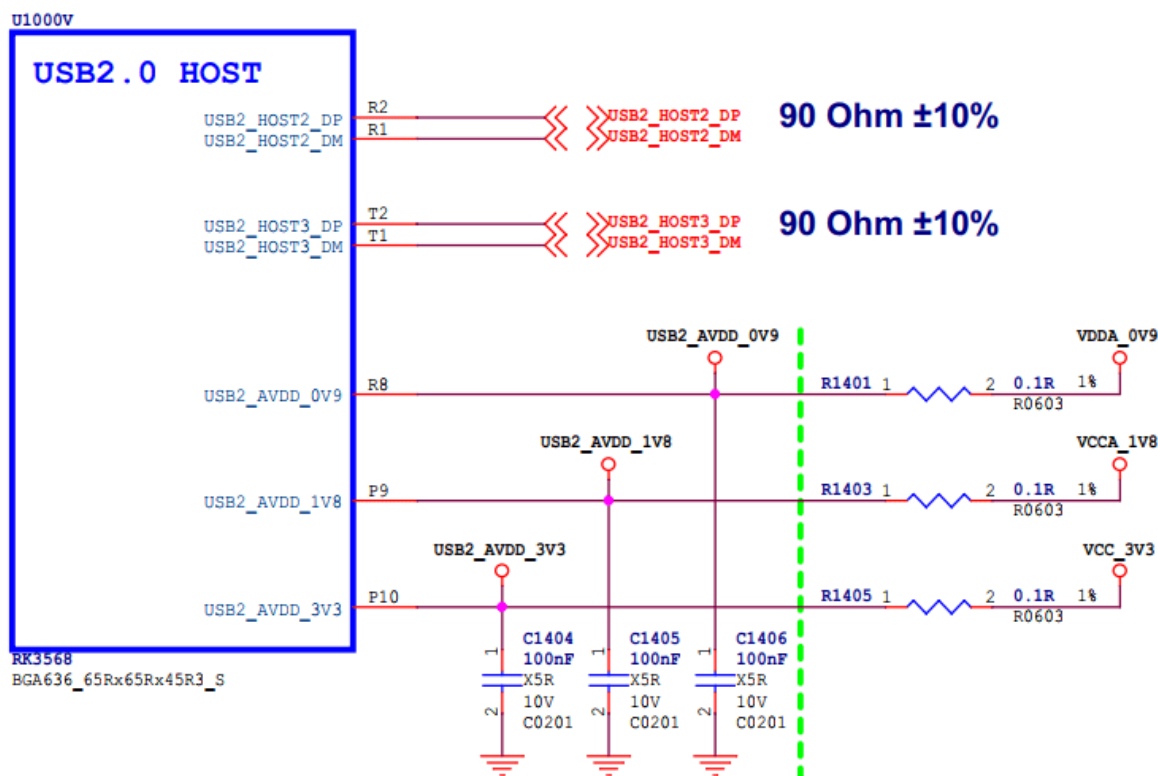


图 2-15 RK356x USB 2.0 Host pins

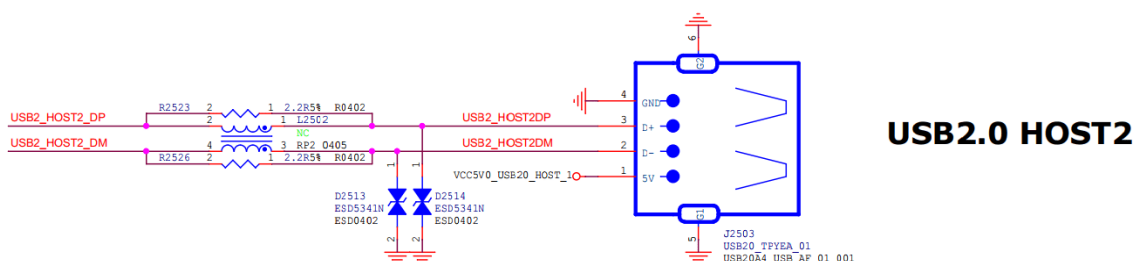


图 2-16 RK356x USB 2.0 Host 接口

2.5 RK3568 USB 3.0 OTG Type-A 接口硬件电路

RK3568 USB 3.0 OTG Type-A 的接口电路设计如图 2-17 ~ 图 2-19 所示。这种电路设计的优点是物理接口比 Micro USB 3.0 尺寸小，并且方便接 USB 外设。但缺点是，没有支持 USB3_OTG0_ID 的检测，因此无法支持硬件自动切换 OTG Host mode，需要通过软件切换 mode，切换命令参考章节[RK356x USB OTG mode 切换命令](#)。通常情况下，USB 3.0 OTG 默认是配置为 OTG mode，当作为 Device mode 连接 PC 时，USB 驱动支持自动切换为 Device mode。如果要连接 USB 外设（如 U 盘）时，则需要软件强制切换为 Host mode。当需要从 Host mode 切换回 Device mode 时，同样需要软件强制切换。

需要注意的是，有些 USB 3.0 外设（如 USB 3.0 机械硬盘和 USB 3.0 Camera）对工作电流要求较高，所以当 USB 3.0 OTG 作 Host mode 时，需要保证 VBUS 的供电电流达到 1A 以上，如图 2-19 所示，电源芯片 SY6280AAC 的输出限流配置为 1.446 A，满足大部分 USB 3.0 外设的工作电流需求。

有四点需要注意：

1. DP/DM 必须分别串接 $2.2\ \Omega$ 电阻，提高 DP/DM 的抗压/抗静电能力；
2. 如果要支持 USB Device 动态拔插检测和充电类型检测功能，则 VBUSDET 脚一定要连接到 USB Type-A 接口；
3. 如果不需要支持 USB Device 动态拔插检测和充电类型检测功能，可以允许 VBUSDET 脚固定拉高到 3.3V 或者悬空 (推荐拉高到 3.3V)。由于 RK356x Maskrom USB 的连接不依赖于 VBUSDET 信号的检测，所以即使 VBUSDET 脚悬空，Maskrom USB 仍然可以正常工作并下载固件；
4. USB3_OTG0_ID 未使用，悬空即可，芯片内部拉高到 1.8V；

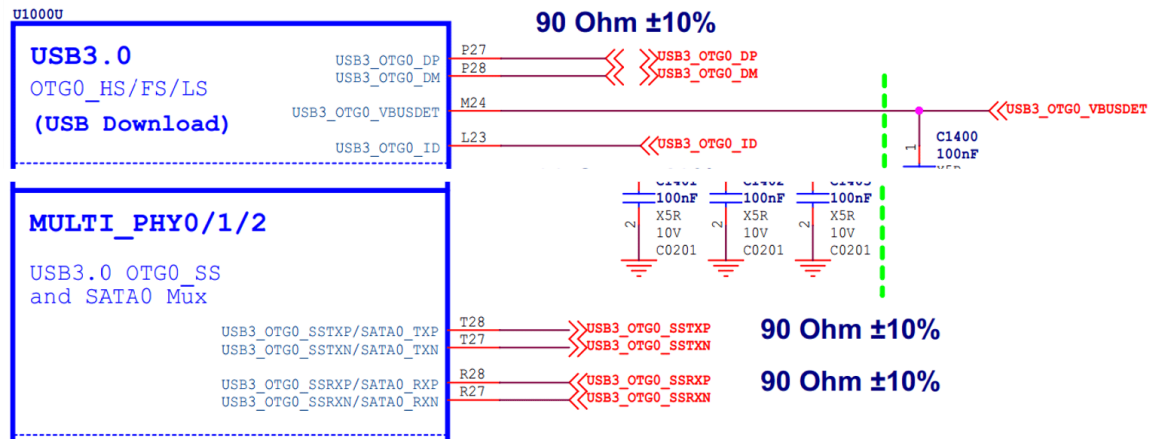


图 2-17 RK3568 USB 3.0 OTG Pins

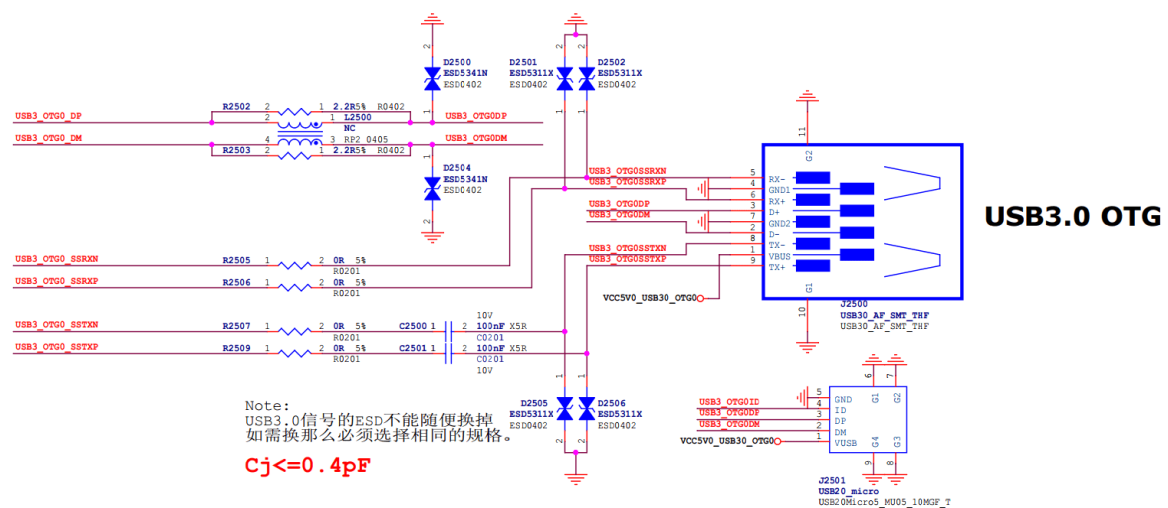


图 2-18 RK3568 USB 3.0 OTG Type-A 接口

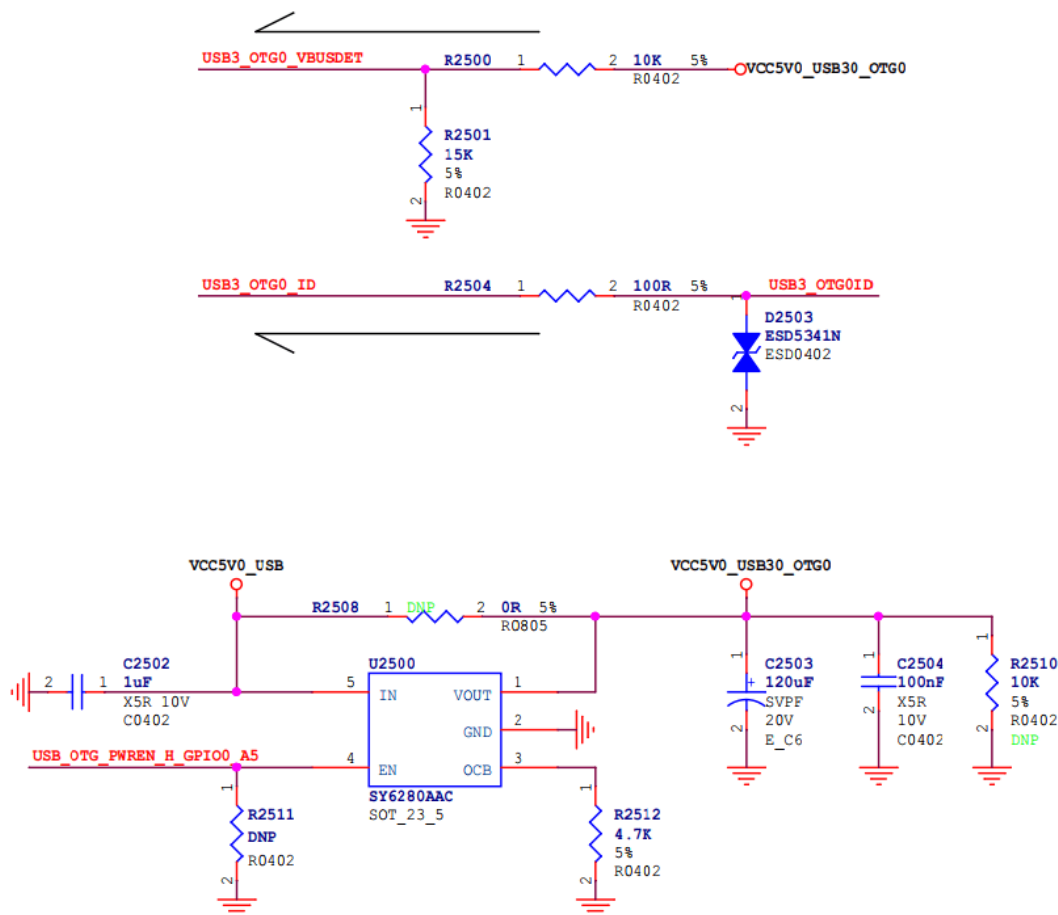


图 2-19 RK3568 USB 3.0 OTG VBUS 和 OTG ID 控制电路

2.6 RK3568 USB 3.0 OTG Type-C 接口硬件电路

RK3568 SDK 未提供 USB 3.0 OTG Type-C 接口的硬件参考电路。如果开发者要支持该接口，需要在 RK3568 USB 3.0 OTG 和 Type-C 接口中间增加一个 USB 3.1 Switch 芯片（如 FUSB340）和一个 CC 通信芯片（如 FUSB302）。

2.7 RK356x USB 3.0 HOST Type-A 接口硬件电路

RK3566/RK3568 USB 3.0 HOST Type-A 接口硬件电路与前面章节 [RK3568 USB 3.0 OTG Type-A 接口硬件电路](#) 基本一样，不同点在于 RK3566/RK3568 USB 3.0 HOST 没有 VBUSDET pin 和 ID pin。电路设计的注意事项，请参考 RK3568 USB 3.0 OTG Type-A 接口硬件电路。

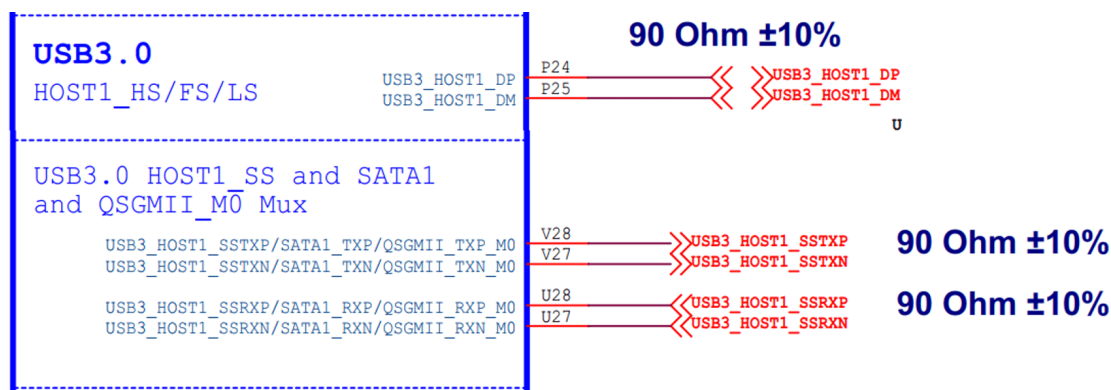


图 2-20 RK356x USB3.0 HOST1 Pins

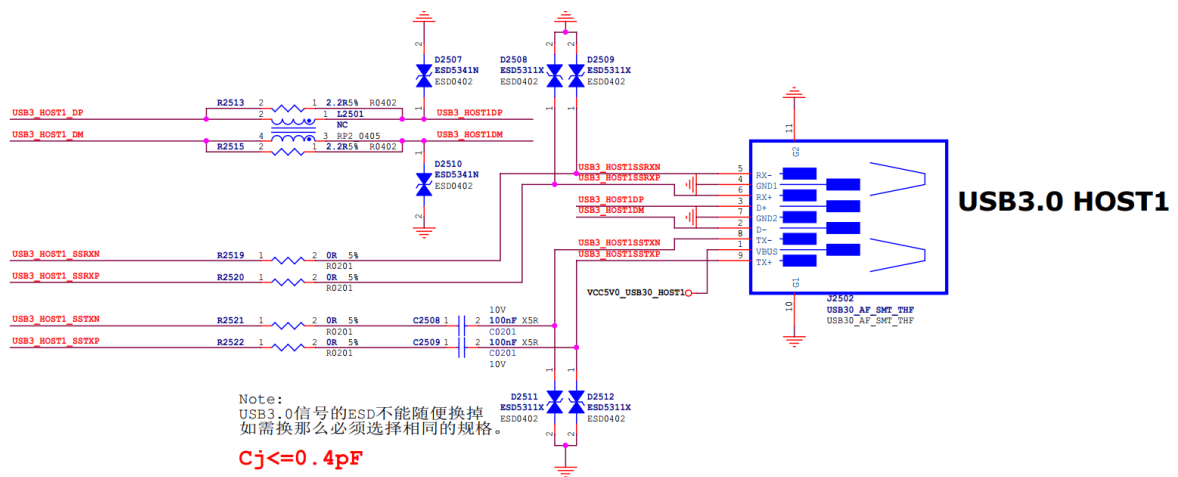


图 2-21 RK356x USB3.0 HOST1 Type-A 接口

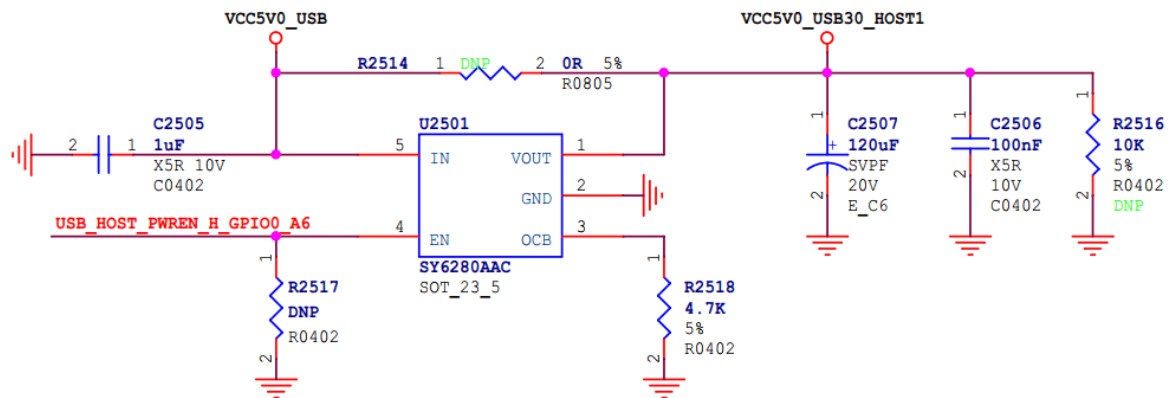


图 2-22 RK356x USB3.0 HOST1 VBUS 控制电路

3. RK356x USB DTS 配置

RK356x USB 硬件电路具有多样化的特点，尤其是灵活多变的 USB OTG 口的硬件电路，以及复杂的 USB 3.0/SATA/QSGMII Combo PHY 复用关系。因此，建议开发者要在理解硬件电路设计的基础上，正确配置 USB 相关的控制器和 PHY 的 DTS 节点。

本文档主要说明 USB DTS 的配置方法，没有说明 USB 控制器和 PHY 的驱动。如果开发者需要了解 USB 控制器和 PHY 的详细设计，请参考 SDK 的 USB 开发指南^[2]。

3.1 RK3568 USB 控制器和 PHY DTSI 节点

RK3568 USB 控制器和 PHY 的 DTS 详细配置方法，请参考内核文档^{[4]~[10]}。

如下代码是 DTSI 中 USB 控制器和 PHY 相关的主要节点。

```
/* USB 3.0 OTG/SATA Combo PHY_0 */
combphy0_us: phy@fe820000 {
    compatible = "rockchip,rk3568-naneng-combphy";
    reg = <0x0 0xfe820000 0x0 0x100>;
    #phy-cells = <1>;
    clocks = <&pmucru CLK_PCIEPHY0_REF>, <&cru PCLK_PIPEPHY0>,
```

```

        <&cru PCLK_PIPE>;
        clock-names = "refclk", "apbclk", "pipe_clk";
        assigned-clocks = <&pmucru CLK_PCIEPHY0_REF>;
        assigned-clock-rates = <24000000>;
        resets = <&cru SRST_P_PIPEPHY0>, <&cru SRST_PIPEPHY0>;
        reset-names = "combphy-apb", "combphy";
        rockchip,pipe-grf = <&pipegrf>;
        rockchip,pipe-phy-grf = <&pipe_phy_grf0>;
        status = "disabled";
};

/* USB 3.0 Host/SATA/QSGMII Combo PHY_1 */
combphy1_usq: phy@fe830000 {
    compatible = "rockchip,rk3568-naneng-combphy";
    reg = <0x0 0xfe830000 0x0 0x100>;
    #phy-cells = <1>;
    clocks = <&pmucru CLK_PCIEPHY1_REF>, <&cru PCLK_PIPEPHY1>,
        <&cru PCLK_PIPE>;
    clock-names = "refclk", "apbclk", "pipe_clk";
    assigned-clocks = <&pmucru CLK_PCIEPHY1_REF>;
    assigned-clock-rates = <24000000>;
    resets = <&cru SRST_P_PIPEPHY1>, <&cru SRST_PIPEPHY1>;
    reset-names = "combphy-apb", "combphy";
    rockchip,pipe-grf = <&pipegrf>;
    rockchip,pipe-phy-grf = <&pipe_phy_grf1>;
    status = "disabled";
};

/* USB OTG/USB Host_1 USB 2.0 Comb PHY_0 */
usb2phy0: usb2-phy@fe8a0000 {
    compatible = "rockchip,rk3568-usb2phy";
    reg = <0x0 0xfe8a0000 0x0 0x10000>;
    interrupts = <GIC_SPI 135 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&pmucru CLK_USBPHY0_REF>;
    clock-names = "phyclk";
    #clock-cells = <0>;
    assigned-clocks = <&cru USB480M>;
    assigned-clock-parents = <&usb2phy0>;
    clock-output-names = "usb480m_phy";
    rockchip,usbgrf = <&usb2phy0_grf>;
    status = "disabled";

    u2phy0_host: host-port {
        #phy-cells = <0>;
        status = "disabled";
    };

    u2phy0_otg: otg-port {
        #phy-cells = <0>;
        status = "disabled";
    };
};

/* USB Host_1/USB Host_2 USB 2.0 Comb PHY_0 */
usb2phy1: usb2-phy@fe8b0000 {
    compatible = "rockchip,rk3568-usb2phy";
    reg = <0x0 0xfe8b0000 0x0 0x10000>;
    interrupts = <GIC_SPI 136 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&pmucru CLK_USBPHY1_REF>;
    clock-names = "phyclk";
    #clock-cells = <0>;

```

```

        rockchip,usbgrf = <&usb2phy1_grf>;
        status = "disabled";

        u2phy1_host: host-port {
            #phy-cells = <0>;
            status = "disabled";
        };

        u2phy1_otg: otg-port {
            #phy-cells = <0>;
            status = "disabled";
        };
    };

    /* USB 3.0 OTG controller */
    usbdrd30: usbdrd {
        compatible = "rockchip,rk3568-dwc3", "rockchip,rk3399-dwc3";
        clocks = <&cru CLK_USB3OTG0_REF>, <&cru CLK_USB3OTG0_SUSPEND>,
            <&cru ACLK_USB3OTG0>, <&cru PCLK_PIPE>;
        clock-names = "ref_clk", "suspend_clk",
            "bus_clk", "pipe_clk";
        #address-cells = <2>;
        #size-cells = <2>;
        ranges;
        status = "disabled";

        usbdrd_dwc3: dwc3@fcc00000 {
            compatible = "snps,dwc3";
            reg = <0x0 0xfcc00000 0x0 0x400000>;
            interrupts = <GIC_SPI 169 IRQ_TYPE_LEVEL_HIGH>;
            dr_mode = "otg";
            phys = <&u2phy0_otg>, <&combphy0_us PHY_TYPE_USB3>;
            phy-names = "usb2-phy", "usb3-phy";
            phy_type = "utmi_wide";
            power-domains = <&power RK3568_PD_PIPE>;
            resets = <&cru SRST_USB3OTG0>;
            reset-names = "usb3-otg";
            snps,dis_enblslpm_quirk;
            snps,dis-u1-entry-quirk;
            snps,dis-u2-entry-quirk;
            snps,dis-u2-freeclk-exists-quirk;
            snps,dis-del-phy-power-chg-quirk;
            snps,dis-tx-ipgap-linecheck-quirk;
            snps,dis_rxdet_inp3_quirk;
            snps,xhci-trb-ent-quirk;
            status = "disabled";
        };
    };

    /* USB 3.0 Host_1 controller */
    usbhost30: usbhost {
        compatible = "rockchip,rk3568-dwc3", "rockchip,rk3399-dwc3";
        clocks = <&cru CLK_USB3OTG1_REF>, <&cru CLK_USB3OTG1_SUSPEND>,
            <&cru ACLK_USB3OTG1>, <&cru PCLK_PIPE>;
        clock-names = "ref_clk", "suspend_clk",
            "bus_clk", "pipe_clk";
        #address-cells = <2>;
        #size-cells = <2>;
        ranges;
        status = "disabled";
    };

```

```

usbhost_dwc3: dwc3@fd000000 {
    compatible = "snps,dwc3";
    reg = <0x0 0xfd000000 0x0 0x400000>;
    interrupts = <GIC_SPI 170 IRQ_TYPE_LEVEL_HIGH>;
    dr_mode = "host";
    phys = <&u2phy0_host>, <&combphy1_usq PHY_TYPE_USB3>;
    phy-names = "usb2-phy", "usb3-phy";
    phy_type = "utmi_wide";
    power-domains = <&power RK3568_PD_PIPE>;
    resets = <&cru SRST_USB30TG1>;
    reset-names = "usb3-host";
    snps,dis_enblslpm_quirk;
    snps,dis-u2-freeclk-exists-quirk;
    snps,dis-del-phy-power-chg-quirk;
    snps,dis-tx-ipgap-linecheck-quirk;
    snps,dis_rxdet_inp3_quirk;
    snps,xhci-trb-ent-quirk;
    status = "disabled";
};

/* USB 2.0 Host_2 EHCI controller for high speed */
usb_host0_ehci: usb@fd800000 {
    compatible = "generic-ehci";
    reg = <0x0 0xfd800000 0x0 0x40000>;
    interrupts = <GIC_SPI 130 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&cru HCLK_USB2HOST0>, <&cru HCLK_USB2HOST0_ARB>,
            <&cru PCLK_USB>, <&usb2phy1>;
    clock-names = "usbhost", "arbiter", "pclk", "utmi";
    phys = <&u2phy1_otg>;
    phy-names = "usb2-phy";
    status = "disabled";
};

/* USB 2.0 Host_2 OHCI controller for full/low speed */
usb_host0_ohci: usb@fd840000 {
    compatible = "generic-ohci";
    reg = <0x0 0xfd840000 0x0 0x40000>;
    interrupts = <GIC_SPI 131 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&cru HCLK_USB2HOST0>, <&cru HCLK_USB2HOST0_ARB>,
            <&cru PCLK_USB>, <&usb2phy1>;
    clock-names = "usbhost", "arbiter", "pclk", "utmi";
    phys = <&u2phy1_otg>;
    phy-names = "usb2-phy";
    status = "disabled";
};

/* USB 2.0 Host_3 EHCI controller for high speed */
usb_host1_ehci: usb@fd880000 {
    compatible = "generic-ehci";
    reg = <0x0 0xfd880000 0x0 0x40000>;
    interrupts = <GIC_SPI 133 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&cru HCLK_USB2HOST1>, <&cru HCLK_USB2HOST1_ARB>,
            <&cru PCLK_USB>, <&usb2phy1>;
    clock-names = "usbhost", "arbiter", "pclk", "utmi";
    phys = <&u2phy1_host>;
    phy-names = "usb2-phy";
    status = "disabled";
};

/* USB 2.0 Host_3 OHCI controller for full/low speed */

```

```
usb_host1_ohci: usb@fd8c0000 {
    compatible = "generic-ohci";
    reg = <0x0 0xfd8c0000 0x0 0x40000>;
    interrupts = <GIC_SPI 134 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&cru HCLK_USB2HOST1>, <&cru HCLK_USB2HOST1_ARB>,
            <&cru PCLK_USB>, <&usb2phy1>;
    clock-names = "usbhost", "arbiter", "pclk", "utmi";
    phys = <&u2phy1_host>;
    phy-names = "usb2-phy";
    status = "disabled";
};
```

3.2 RK3568 USB 3.0 OTG to USB 2.0 only 配置

USB 3.0 OTG 的 USB 3.0 PHY 与 SATA_0 控制器复用 Comb PHY_0，当 Comb PHY_0 配置给 SATA_0 用时，需要配置 USB 3.0 OTG 工作在 USB 2.0 only 模式，对应的 DTS 配置如下（控制器和PHY的驱动不用改动）：

```
&usbdrd_dwc3 {
    phys = <&u2phy0_otg>; /* 配置 phys 属性只引用 usb 2.0 phy 节点 */
    phy-names = "usb2-phy";
    extcon = <&usb2phy0>;
    maximum-speed = "high-speed"; /* 配置 dwc3 控制器最高支持 high speed */
    snps,dis_u2_susphy_quirk; /* 配置 dwc3 控制器不支持自动 suspend usb2 phy */
    snps,usb2-lpm-disable; /* 配置 dwc3 host mode 不支持 LPM 功能 */
};

&combphy0_us {
    rockchip,dis-u3otg0-port; /* 配置 dwc3_0 控制器最高支持 high speed */
    status = "okay";
};

&u2phy0_otg {
    rockchip,dis-u2-susphy; /* 可选项，用于关闭 PHY 动态进入 suspend 的功能，提高
USB 热拔插稳定性 */
    status = "okay";
};
```

3.3 RK356x USB 3.0 Host to USB 2.0 only 配置

USB 3.0 Host 的 USB 3.0 PHY 与 SATA_1/QSGMII 控制器复用 Comb PHY_1，当 Comb PHY_1 配置给 SATA_1 或者 QSGMII 用时，需要配置 USB 3.0 Host 工作在 USB 2.0 only 模式，对应的 DTS 配置如下（控制器和PHY的驱动不用改动）：

```

&usbhost_dwc3 {
    phys = <u2phy0_host>; /* 配置 phys 属性只引用 usb 2.0 phy 节点 */
    phy-names = "usb2-phy";
    maximum-speed = "high-speed"; /* 配置 dwc3 控制器最高支持 high speed */
    snps,dis_u2_susphy_quirk; /* 配置 dwc3 控制器不支持自动 suspend usb2 phy */
    snps,usb2-lpm-disable; /* 配置 dwc3 host mode 不支持 LPM 功能 */
    status = "okay";
};

&combphy1_usq {
    rockchip,dis-u3otgl-port; /* 配置 dwc3_1 控制器最高支持 high speed */
    status = "okay";
};

```

3.4 RK356x Type-C USB DTS 配置

常见的 RK356x Type-C 设计方案包括如下五种：

1. RK3566 Type-C USB 2.0 only;
2. RK3568 Type-C USB 2.0 only;
3. RK3566 Type-C USB 2.0 + PD;
4. RK3568 Type-C USB 2.0 + PD;
5. RK3568 Type-C USB 3.0 + PD;

上述方案 1 和 2，硬件电路可以参考章节[RK3566 Type-C USB 2.0 only 硬件电路设计\[No PD\]](#)，采用了 CC to OTG_ID 的硬件转换电路，芯片内部检测 OTG mode 的原理与 Micro-B USB 接口一样。因此，方案 1 和 2 的 USB DTS 配置请参考常见的 RK356x OTG Micro-USB /Type-A 接口设计即可。

方案 3 和 4，硬件电路可以参考章节[RK3566 Type-C USB 2.0 + PD 硬件电路](#)，采用外置 Type-C 控制器芯片以支持 PD 通信，这两种方案的 USB DTS 配置基本一样，本章节重点说明方案 3 (RK3566 Type-C USB 2.0 + PD) 的 USB DTS 配置方法，方案 4 只需要在此基础上，参考章节[RK3568 USB 3.0 OTG to USB 2.0 only 配置](#)，配置 USB 控制器工作在 USB 2.0 only 模式即可。

方案 5，考虑到 RK3568 USB3 PHY 不支持 Type-C 正反面切换，因此，硬件电路需要增加两个外置芯片，即：一个 USB3 Switch 芯片（如 FUSB340）和一个 Type-C 控制器芯片（如 FUSB302）。其中，Type-C 控制器芯片的 DTS 配置可以参考方案 3 和 4，而 USB 3.1 Switch 芯片的控制，需要根据硬件方案选型，自行开发控制驱动。

3.4.1 Linux-4.19 RK3566 Type-C USB 2.0 + FUSB302 DTS 配置

Linux-4.19 内核支持两套 FUSB302 驱动，分别是：

`drivers/mfd/fusb302.c` [Legacy]

`drivers/usb/typec/tcpm/fusb302.c` [New]

Linux-4.19 SDK 默认使用 Legacy 驱动，rockchip_defconfig 已经使能对应的驱动配置 CONFIG_FUSB_30X，并且，该驱动已经在 RK3399 平台上推广和量产，推荐优先使用。如果使用新驱动，需要参考[Linux-4.19 RK3566 Type-C USB 2.0 + HUSB311 DTS 配置](#)，更新内核 Type-C TCPM 系列补丁。

以适配 Legacy FUSB302 驱动为例，Linux-4.19 DTS 参考配置如下：

/ 根据 FUSB302 I2C 实际硬件设计进行修改 */*

```

&i2cx {
    fusb0: fusb30x@22 { /* 配置 FUSB302 芯片硬件信息*/
        compatible = "fairchild,fusb302";
        reg = <0x22>;
        pinctrl-names = "default";
        pinctrl-0 = <&fusb0_int>;
        int-n-gpios = <&gpiox RK_Pxx GPIO_ACTIVE_HIGH>; /* 根据 FUSB302
INT 实际连接的 GPIO 进行修改 */
        vbus-5v-gpios = <&gpiox RK_Pxx GPIO_ACTIVE_HIGH>; /* 根据 Type-C
VBUS 实际连接的 GPIO 进行修改 */
        status = "okay";
    };
};

/* 使能 USB2.0 PHY */
&usb2phy0 {
    status = "okay";
    extcon = <&fusb0>; /* 配置 extcon 属性, 用于接收 FUSB302 驱动的 Type-C DRD
notifier*/
};

/* 使能 USB2.0 PHY OTG port */
&u2phy0_otg {
    rockchip,dis-u2-susphy; /* 可选项, 用于关闭 PHY 动态进入 suspend 的功能, 提高
USB 热拔插稳定性 */
    status = "okay";
};

/* 使能 USB3 控制器父节点 */
&usbd3 {
    status = "okay";
};

/* 使能 USB3 控制器子节点 */
&usbd3_dwc3 {
    extcon = <&fusb0>; /* 配置 extcon 属性, 用于接收 FUSB302 驱动的 Type-C DRD
notifier*/
    status = "okay";
};

&pinctrl {
    fusb30x { /* 配置 FUSB302 驱动的中断和 vbus gpio pinctrl */
        fusb0_int: fusb0-int {
            rockchip,pins = <gpiox RK_Pxx RK_FUNC_GPIO
&pcfg_pull_up>,
                                <gpiox RK_Pxx RK_FUNC_GPIO
&pcfg_pull_up>;
        };
    };
};

```

3.4.2 Linux-4.19 RK3566 Type-C USB 2.0 + HUSB311 DTS 配置

RK3568 最新的 Linux-4.19 内核已经可以支持 HUSB311 和 ET7303 这两款 Type-C 控制器芯片，考虑到 HUSB311 和 ET7303 的 I2C 设备地址都为 4E，DTS 配置基本一样，只需要修改节点名字和 "compatible" 属性，因此，本章节以 HUSB311 为例进行说明。

HUSB311 和 ET7303 这两款芯片都是基于内核 Type-C 驱动 TCPM 框架，因此，要求 Linux-4.19 同步到 develop-4.19 最新代码，如果无法同步完整的内核代码，至少要同步如下目录：

```
kernel/include/linux/usb
kernel/drivers/usb/typec
kernel/drivers/usb/dwc3
```

同时，还需要手动更新如下补丁：

```
0001-usb-typec-tcpm-add-vdm-retry-mechanism.patch
0002-arm64-rockchip_defconfig-enable-tcpm-and-husb311-and.patch
```

请下载完整的参考资料："RK356x_Linux-4.19_TypeC-HUSB311-ET7303-参考设计"，进行修改。

下载地址：<https://redmine.rockchip.com.cn/documents/113>

以 RK3566 适配 HUSB311 驱动为例，Linux-4.19 DTS 参考配置如下：

```
/* 需要包含头文件 */
#include "dt-bindings/usb/pd.h"

/* 可选配置，用于 vbus 通过 gpio 控制的硬件方案 */
vcc_otg_vbus: otg-vbus-regulator {
    compatible = "regulator-fixed";
    gpio = <&gpiox RK_PXX GPIO_ACTIVE_HIGH>; /* 根据 Type-C VBUS 实际连接的
GPIO 进行修改 */
    pinctrl-names = "default";
    pinctrl-0 = <&otg_vbus_drv>;
    regulator-name = "vcc_otg_vbus";
    regulator-min-microvolt = <5000000>;
    regulator-max-microvolt = <5000000>;
    enable-active-high;
};

/* 根据 HUSB311 I2C 实际硬件设计进行修改 */
&i2cx {
    /* 对于 et7303，可以修改为 usbc0: et7303@4e */
    usbc0: husb311@4e {
        /* 对于 et7303，compatible = "etek,et7303"*/
        compatible = "hynetek,husb311";
        reg = <0x4e>;
        /* 根据 HUSB311 INT 实际连接的 GPIO 进行修改 */
        interrupt-parent = <&gpiox>;
        interrupts = <X IRQ_TYPE_LEVEL_LOW>;
        pinctrl-names = "default";
        pinctrl-0 = <&husb311_int>;
        /*
        * vbus 配置
        * 若 vbus 是通过 gpio 控制，则需要 vbus-supply = <&vcc_otg_vbus> 配
置；
        * 若 vbus 是 RK817/RK818/bq25700/其他 charge ic 输出，则无需 vbus-
supply = <&vcc_otg_vbus> 配置；
        */
    };
};
```

```

        */
        vbus-supply = <&vcc_otg_vbus>;
        status = "okay";

        usb_con: connector {
            data-role = "dual";
            power-role = "dual";
            try-power-role = "sink";
            /*
            * source-pdos: PDO_FIXED(电压(mV), 电流(mA),
PDO_FIXED_USB_COMM)
            * 主板端作为 Source 时, 主板 vbus 可以输出的电压/电流列表, 对于
RK平台采用默认值即可
            */
            source-pdos = <PDO_FIXED(5000, 2000,
PDO_FIXED_USB_COMM)>;
            /*
            * sink-pdos: <PDO_FIXED(电压(mV), 电流(mA),
PDO_FIXED_USB_COMM)>
            * 主板端作为 Sink 时, 主板可以支持的电压/电流列表.
            * 若有 PD 快充需求, sink-pdos 配置依赖于 charger ic 充电输入
最大电压/电流和主板承受的能力
            * 若电压太大可能会烧主板, 这个细节请与硬件工程师沟通.
            * 下面是主板可支持最大充电电压(20v)/电流(3A)配置, 请根据硬件设计
和实际需求进行适当修改.
            * 比如主板只支持最大充电电压(12v)/电流(3A), 必须需要去掉 15v/3A
和 20V/3A.
            */
            sink-pdos = <PDO_FIXED(5000, 3000, PDO_FIXED_USB_COMM)
                PDO_FIXED(9000, 3000, PDO_FIXED_USB_COMM)
                PDO_FIXED(12000, 3000, PDO_FIXED_USB_COMM)
                PDO_FIXED(15000, 3000, PDO_FIXED_USB_COMM)
                PDO_FIXED(20000, 3000,
PDO_FIXED_USB_COMM)>;
            /* 若没有 PD 快充需求, sink-pdos 配置如下即可. */
            sink-pdos = <PDO_FIXED(5000, 2000, PDO_FIXED_USB_COMM)>;
            /*充电功率(op-sink-microwatt) = max_vol(mV) * max_cur(mA),
使用默认即可 */
            op-sink-microwatt = <10000000>;
        };
    };

    &pinctrl {
        husb311 {
            husb311_int: husb311-int {
                rockchip,pins = <X RK_PXX RK_FUNC_GPIO &pcfg_pull_up>;
            };
        };

        usb {
            otg_vbus_drv: otg-vbus-drv {
                /* 可选项, 如果无 vcc_otg_vbus, 就不需要该配置 */
                rockchip,pins = <RK_XX RK_FUNC_GPIO &pcfg_pull_none>;
            };
        };
    };
};

```

```

/* 使能 PMIC */
bq25700: bq25700@6b {
    compatible = "ti,bq25703";
    reg = <0x6b>;
    extcon = <&usbc0>; /* 配置 extcon 属性，用于接收 TCPM 驱动的 PD notifier */
    .....
};

/* 使能 USB2.0 PHY */
&u2phy0 {
    status = "okay";
    extcon = <&usbc0>; /* 配置 extcon 属性，用于接收 TCPM 驱动的 Type-C DRD
notifier*/
};

/* 使能 USB2.0 PHY OTG port */
&u2phy0_otg {
    rockchip,dis-u2-susphy; /* 可选项，用于关闭 PHY 动态进入 suspend 的功能，提高
USB 热拔插稳定性 */
    status = "okay";
};

/* 使能 USB3 控制器父节点 */
&usbdrd30 {
    status = "okay";
};

/* 使能 USB3 控制器子节点 */
&usbdrd_dwc3 {
    status = "okay";
    extcon = <&usbc0>; /* 配置 extcon 属性，用于接收 TCPM 驱动的 Type-C DRD
notifier */
};

```

USB VBUS 通过 RK817/RK818/bq25700 等 PMIC 控制的 DTS 配置说明

```

/* rk818 */
rk818: pmic@1c {
    extcon = <&usbc0>;
};

/* rk817 */
rk817: pmic@20 {
    charger {
        extcon = <&usbc0>;
    };
};

/* bq25700 */
bq25700: bq25700@09 {
    extcon = <&usbc0>;
};

```

3.4.3 Linux-5.10 RK3566 Type-C USB 2.0 + FUSB302 DTS 配置

Linux-5.10 Type-C 驱动采用标准的 TCPM 框架，与 Linux-4.19 及更早的内核版本，最大的区别在于：Linux-5.10 Type-C 驱动采用 remote-endpoint 和注册回调函数的机制，替代之前的 extcon 机制。因此，Linux-5.10 Type-C 相关的 DTS 配置中，不再需要 "extcon" 属性。

以 RK3566 适配 FUSB302 驱动为例，Linux-5.10 DTS 参考配置如下：

```
/* 配置 VBUS gpio regulator, 用于 Type-C 作 USB Host mode 时 VBUS 5V 输出控制 */
vbus_typec: vbus-typec-regulator {
    compatible = "regulator-fixed";
    enable-active-high;
    gpio = <&gpiox RK_PXX GPIO_ACTIVE_HIGH>;
    pinctrl-names = "default";
    pinctrl-0 = <&vcc5v0_typec0_en>;
    regulator-name = "vbus_typec";
    vin-supply = <&vcc5v0_sys>;
};

/* 配置 FUSB302 芯片硬件信息*/
&i2cX { /* 根据 FUSB302 I2C 实际硬件设计进行修改 */
    usbc0: fusb302@22 {
        compatible = "fcs,fusb302";
        reg = <0x22>;
        interrupt-parent = <&gpiox>; /* 根据 FUSB302 INT 实际连接的 GPIO 进
进行修改 */

        interrupts = <RK_PXX IRQ_TYPE_LEVEL_LOW>;
        pinctrl-names = "default";
        pinctrl-0 = <&usbc0_int>;
        vbus-supply = <&vbus_typec>;
        status = "okay";

        ports {
            #address-cells = <1>;
            #size-cells = <0>;
            /* 用于关联 USB 控制器, 实现 USB 角色切换的功能 */
            port@0 {
                reg = <0>;
                usbc0_role_sw: endpoint@0 {
                    remote-endpoint = <&dwc3_role_switch>;
                };
            };
        };

        usb_con: connector { /* 用于描述 PD 协议相关信息 */
            compatible = "usb-c-connector";
            label = "USB-C";
            data-role = "dual";
            power-role = "dual";
            try-power-role = "sink";
            op-sink-microwatt = <1000000>;
            sink-pdos =
                <PDO_FIXED(5000, 2500, PDO_FIXED_USB_COMM)>;
            source-pdos =
                <PDO_FIXED(5000, 1500, PDO_FIXED_USB_COMM)>;
        };
    };
};
```

```

/* 使能 Type-C0 USB2.0 PHY */
&u2phy0 {
    status = "okay";
};

&u2phy0_otg {
    rockchip,dis-u2-susphy; /* 可选项，用于关闭 PHY 动态进入 suspend 的功能，提高
USB 热拔插稳定性 */
    status = "okay";
};

/* 使能 USB3 控制器父节点 */
&usbdrd30 {
    status = "okay";
};

/* 使能 USB3 控制器子节点 */
&usbdrd_dwc3 {
    status = "okay";
    usb-role-switch; /* 用于使能 DWC3 驱动注册 USB Role Switch 回调函数 */
    port {
        #address-cells = <1>;
        #size-cells = <0>;
        dwc3_role_switch: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&usb0_role_sw>; /* 关联 FUSB302，实现
USB 角色切换的功能 */
        };
    };
};

&pinctrl {
    usb-typec {
        usbc0_int: usbc0-int { /* 配置 FUSB302 驱动的中断 gpio pinctrl */
            rockchip,pins = <X RK_PXX RK_FUNC_GPIO &pcfg_pull_up>;
        };

        vcc5v0_type0_en: vcc5v0-typec0-en { /* 配置 FUSB302 驱动的 VBUS
gpio pinctrl */
            rockchip,pins =
                <X RK_PXX RK_FUNC_GPIO &pcfg_pull_none>;
        };
    };
};

```

3.4.4 Linux-5.10 RK3566 Type-C USB 2.0 + HUSB311 DTS 配置

使用 HUSB311 芯片替换 FUSB302 芯片，只需要基于[Linux-5.10 RK3566 Type-C USB 2.0 + FUSB302 DTS 配置](#)进行简单修改即可，参考修改：

```

/* 配置外置 Type-C 控制器芯片 HUSB311 */
&i2cx {
    usbc0: husb311@4e {
        compatible = "hynetek,husb311";
        reg = <0x4e>;
        .....
    };
};

```

3.5 USB VBUS 配置

Rockchip 平台的 USB VBUS 控制电路，主要有三种方案^[3]:

1. 使用 GPIO 控制电源稳压芯片输出 Vbus 5V 供电电压;
2. 使用 PMIC (如RK809/RK817/RK818) 输出 Vbus 5V 供电电压;
3. 开机后, 硬件直接输出 Vbus 5V 供电电压, 不需要软件控制, 一般用于 USB Host 接口;

RK356x SDK 主要使用的是第 1 种方案, 以 RK3568 EVB 的 OTG 口和 HOST 口的 VBUS 配置为例:

```

vcc5v0_host: vcc5v0-host-regulator {
    compatible = "regulator-fixed";
    enable-active-high;
    gpio = <&gpio0 RK_PA6 GPIO_ACTIVE_HIGH>;
    pinctrl-names = "default";
    pinctrl-0 = <&vcc5v0_host_en>;
    regulator-name = "vcc5v0_host";
    regulator-always-on; /* Host Vbus 为常供电, 所以配置属性为 regulator-always-
on */
};

vcc5v0_otg: vcc5v0-otg-regulator {
    compatible = "regulator-fixed";
    enable-active-high;
    gpio = <&gpio0 RK_PA5 GPIO_ACTIVE_HIGH>;
    pinctrl-names = "default";
    pinctrl-0 = <&vcc5v0_otg_en>;
    regulator-name = "vcc5v0_otg";
};

&pinctrl {
    usb {
        vcc5v0_host_en: vcc5v0-host-en {
            rockchip,pins = <0 RK_PA6 RK_FUNC_GPIO &pcfg_pull_none>;
        };

        vcc5v0_otg_en: vcc5v0-otg-en {
            rockchip,pins = <0 RK_PA5 RK_FUNC_GPIO &pcfg_pull_none>;
        };
    };
};

/* 注意: host 控制 vbus 的属性是 phy-supply */
&u2phy0_host {
    phy-supply = <&vcc5v0_host>;
    status = "okay";
};

```

```
};

/* 注意: otg 控制 vbus 的属性是 vbus-supply */
&u2phy0_otg {
    vbus-supply = <&vcc5v0_otg>;
    status = "okay";
};

&u2phy1_host {
    phy-supply = <&vcc5v0_host>;
    status = "okay";
};

&u2phy1_otg {
    phy-supply = <&vcc5v0_host>;
    status = "okay";
};
```

3.6 Linux USB DT 配置的注意点

3.6.1 USB DT 重要属性说明

3.6.1.1 USB 3.1 控制器 DT 属性

1. "snps,dis_rxdet_inp3_quirk": RK356x USB 3.1 控制器的特有属性，用于关闭 P3 state 下的 receiver dtection (Rx-detect) 功能，即增加该属性后，在执行 receiver dtection 时，会先将 USB3 PHY 状态从 P3 切回 P2，待 receiver dtection (耗时 30us 左右) 结束后，再将 PHY 状态切回 P3。该属性可以解决部分 USB3 盘无法识别的兼容性问题。
2. "snps,dis_u2_susphy_quirk": 配置 USB 3.1 控制器不支持自动 suspend USB2 PHY。在 USB 3.1 force 为 USB 2.0 使用的场景，必须加该属性，否则，可能导致 USB2 功能异常。其他场景，加该属性，不会影响 USB 功能，但会增加 USB autosuspend 场景的功耗。
3. "snps,usb2-lpm-disable": 配置 USB 3.1 控制器关闭 USB2 LPM 的功能。在 USB 3.1 force 为 USB 2.0 使用的场景，必须加该属性，以兼容不支持 USB2 LPM 的 U 盘。
4. "usb-role-switch": 仅用于 Linux-5.10 且 USB 物理接口为标准 Type-C 接口（带有 PD 控制器芯片），同时须配置 dr_mode="otg" 属性；如果 dr_mode 为非“otg”模式，请勿配置 "usb-role-switch"；
5. "extcon": 主要功能之一是动态切换 OTG mode，适用于非标准 Type-C 接口（带有 PD 控制器芯片）的硬件电路设计，如：USB Micro 接口或者 Type-A 接口，同时控制器配置为“otg”模式的方案中，实现 OTG 模式动态切换。

3.6.1.2 USB2 PHY DT 属性

1. "vbus-supply" 和 "phy-supply": 这两个属性都是用于控制 VBUS 输出 5V，但两者又有使用区别。"vbus-supply" 用于 OTG 口，支持动态开关 VBUS。"phy-supply" 用于 USB2 HOST 口，系统上电后，VBUS 5V 常开；
2. "rockchip,vbus-always-on": 用于 OTG 接口且 VBUS 常供电的场景。增加该属性，可以解决 USB Device（如：USB ADB）无法识别的问题。但同时会禁止 USB2 PHY autosuspend 的功能，增加动态运行时的 USB2 PHY 功耗；
3. "rockchip,dis-u2-susphy": 关闭 USB2 PHY 驱动动态进入 suspend mode 的功能，主要用于 USB 2.0 only 的方案，保持 USB2 PHY 输出时钟给 USB 控制器。

3.6.1.3 USB 3.0/SATA/QSGMII ComboPHY DT 属性

1. "rockchip,dis-u3otg0-port": 关闭 USB3_OTG0 的 USB3 root port, 用于 USB3_OTG0 force 为 USB 2.0 使用的场景;
2. "rockchip,dis-u3otg1-port": 关闭 USB3_HOST1 的 USB3 root port, 用于 USB3_HOST1 force 为 USB 2.0 使用的场景;

4. RK356x USB OTG mode 切换命令

RK356x SDK 支持通过软件方法, 强制设置 USB 2.0/3.0 OTG 切换到 Host mode 或者 Peripheral mode, 而不受 USB 硬件电路的 OTG ID 电平或者 Type-C 接口的影响。

- Linux-4.19/5.10 USB OTG mode 切换命令 (Legacy)

```
#1.Force host mode
echo host > /sys/devices/platform/fe8a0000.usb2-phy/otg_mode
#2.Force peripheral mode
echo peripheral > /sys/devices/platform/fe8a0000.usb2-phy/otg_mode
#3.Force otg mode
echo otg > /sys/devices/platform/fe8a0000.usb2-phy/otg_mode
```

- Linux-5.10 内核新增切换命令

```
#1.Force host mode
echo host > /sys/kernel/debug/usb/fcc00000.usb/mode
#2.Force peripheral mode
echo device > /sys/kernel/debug/usb/fcc00000.usb/mode
```

Note:

Linux-5.10 内核新增的切换命令, 可以解决旧的切换命令的使用限制 (依赖于 USB DTS 的 "extcon" 属性的正确配置)。

5. Type-C 控制器芯片支持列表

表 5-1 Type-C 控制器芯片支持列表

Type-C控制器 芯片型号	Linux- 4.4	Linux- 4.19	Linux- 5.10	说明
FUSB302	支持	支持	支持	RK平台最常用
ET7301B	支持	支持	支持	软硬件完全兼容 FUSB302 ^{note1}
ET7303	不支持	支持	支持	硬件兼容 FUSB302，软件驱动与 RT1711 高度相似 ^{note2}
HUSB311	不支持	支持	支持	推荐优先使用 硬件兼容 FUSB302，但软件驱动不兼容 ^{note3}
RT1711H	不支持	支持	支持	硬件兼容 FUSB302，软件驱动与 ET7303 高度相似 ^{note4}
ANX7411	不支持	不支持	调试 中	RK3588 适配中
WUSB3801	SDK 不 支持， 个别项目 使用	不支持	不支持	自定义的单线通信机制，误码率高，无 法保证通信稳定。

note1.

- Linux-4.4 因为不支持 TCPM 软件框架，所有只能支持 FUSB302/ET7301B，两者可以直接替换使用，不需要修改软硬件；
- Linux-4.19/5.10 支持 TCPM 软件框架和 TCPCI 协议，理论上可以兼容所有基于 TCPCI 标准设计的 Type-C 控制器芯片（如：ET7303/HUSB311/RT1711H）；

note2.

- 小封装的 ET7303 (据了解原厂目前没有提供大封装) 已经在 RK 平台验证通过。内核需要单独使能 CONFIG_TYPEC_ET7303，DTS 详细配置请参考章节[Linux-4.19 RK3566 Type-C USB 2.0 + HUSB311 DTS 配置](#)和章节[Linux-5.10 RK3566 Type-C USB 2.0 + HUSB311 DTS 配置](#)，因为 HUSB311 和 ET7303 的 I2C 设备地址都为 0x4E，DTS 配置基本一样，只需要修改节点名字和 compatible = "etek,et7303" 属性。

note3.

- 内核需要单独使能 CONFIG_TYPEC_HUSB311，DTS 详细配置请参考章节[Linux-4.19 RK3566 Type-C USB 2.0 + HUSB311 DTS 配置](#)和章节[Linux-5.10 RK3566 Type-C USB 2.0 + HUSB311 DTS 配](#)

置。

note4.

- RT1711H 硬件兼容 FUSB302/ET7303，同时软件驱动与 ET7303 高度相似。内核需要单独使能 CONFIG_TYPEC_RT1711H，DTS 配置与 HUSB311/ET7303 基本一样，I2C 设备地址都为 0x4E，只需要修改节点名字和 compatible = "richtek,rt1711h" 属性。

6. 参考文献

1. 《Universal Serial Bus Type-C Cable and Connector Specification》
2. 《Rockchip_Developer_Guide_USB_CN》
3. 《Rockchip_RK3399_Developer_Guide_USB_CN》
4. kernel/Documentation/devicetree/bindings/usb/generic.txt
5. kernel/Documentation/devicetree/bindings/usb/dwc3.txt
6. kernel/Documentation/devicetree/bindings/usb/rockchip,dwc3.txt
7. kernel/Documentation/devicetree/bindings/usb/usb-ehci.txt
8. kernel/Documentation/devicetree/bindings/usb/usb-ohci.txt
9. kernel/Documentation/devicetree/bindings/phy/phy-rockchip-naneng-combphy.txt
10. kernel/Documentation/devicetree/bindings/phy/phy-rockchip-inno-usb2.txt