Rockchip RK3588 USB 开发指南

文件标识：RK-SM-YF-450

发布版本：V1.1.0

日期：2022-09-22

文件密级：□绝密 □秘密 □内部资料 ■公开

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 RK3588 USB 模块的开发指南，目的是让开发者理解 RK3588 USB 控制器和 PHY 的硬件电路设计和软件 DTS 配置，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
RK3588、RK3588S	Linux-5.10

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
2022-09-22	V1.1.0	吴良峰 王明成	新增 Micro 接口配置说明 新增 DT 重要属性说明

目录

RK3588 USB 控制器和 PHY 简介

RK3588 支持 5 个独立的 USB 控制器，包括：2 个 USB 2.0 HOST 控制器，2 个 USB 3.1 OTG 控制器，1 个 USB 3.1 HOST 控制器。RK3588S 相比 RK3588 少了 1 个 USB 3.1 OTG 控制器。USB 控制器的具体类型如下表 1 所示，如果要了解更详细的 USB 控制器特性，请参考 RK3588 datasheet。

表 1 RK3588 USB 控制器列表

芯片/控制器	USB 2.0 HOST (EHCI&OHCI)	USB 3.1 OTG (DWC3&xHCI)	USB 3.1 Host(xHCI)
RK3588	2	2	1
RK3588S	2	1	1

RK3588 支持 7 个独立的 USB PHY，包括：4 个 USB 2.0 PHY，2 个 USB 3.1/DP Combo PHY，1 个 USB 3.1/SATA/PCIe Combo PHY。RK3588S 相比 RK3588 少了 1 个 USB 2.0 PHY 和 1 个 USB 3.1/DP Combo PHY。

表 2 RK3588 USB PHY 支持列表

芯片/PHY	USB 2.0 PHY	USB3.1/DP ComboPHY	USB 3.1/SATA/PCIe ComboPHY
RK3588	4 [1 × port]	2	1
RK3588S	3 [1 × port]	1	1

Note:

1. 表格中，数字 N 表示支持 N 个独立的 USB 控制器和 USB PHY；
2. 表格中，[1 × ports] 表示一个 PHY 只支持 1 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. USB 3.1 Gen1 物理层传输速率为 5Gbps，USB 2.0 物理层传输速率为 480Mbps;
5. USB 3.1/DP Combo PHY 支持 4 x lanes，可以同时支持 USB 3.1 + DP 2 x lanes；
6. USB 3.1/SATA/PCIe Combo PHY 在同一时刻，只能支持一种工作模式，也即 USB3.1 与 SATA/PCIe 接口是互斥的；

表 3 RK3588 USB 控制器和 PHY 的连接关系

USB 接口名称(原理图)	USB 控制器	USB PHY
TYPEC0	OTG0 (DWC3&xHCI)	USB3.1/DP ComboPHY0 + USB2.0 PHY0
TYPEC1	OTG1 (DWC3&xHCI)	USB3.1/DP ComboPHY1 + USB2.0 PHY1
USB20_HOST0	USB2.0 HOST0 (EHCI&OHCI)	USB2.0 PHY2
USB20_HOST1	USB2.0 HOST1 (EHCI&OHCI)	USB2.0 PHY3
USB30_2	USB3.1 HOST2 (xHCI)	USB3.1/SATA/PCIe ComboPHY2

RK3588 USB 控制器和芯片端 USB 传输数据的 pin 脚的对应关系如下表 4 所示。

表 4 RK3588 USB 控制器和 USB pin 脚的对应关系

USB控制器/Pin脚	RK3588 USB data pin
USB 2.0 HOST0	USB20_HOST0_DP/USB20_HOST0_DM
USB 2.0 HOST1	USB20_HOST1_DP/USB20_HOST1_DM
USB 3.1 OTG0	TYPEC0_USB20_OTG_DP/TYPEC0_USB20_OTG_DM, TYPEC0_SSRX1P/TYPEC0_SSRX1N, TYPEC0_SSTX1P/TYPEC0_SSTX1N, TYPEC0_SSRX2P/TYPEC0_SSRX2N, TYPEC0_SSTX2P/TYPEC0_SSTX2N
USB 3.1 OTG1	TYPEC1_USB20_OTG_DP/TYPEC1_USB20_OTG_DM, TYPEC1_SSRX1P/TYPEC1_SSRX1N, TYPEC1_SSTX1P/TYPEC1_SSTX1N TYPEC1_SSRX2P/TYPEC1_SSRX2N, TYPEC1_SSTX2P/TYPEC1_SSTX2N
USB 3.1 HOST2	USB30_2_SSTXP/USB30_2_SSTXN USB30_2_SSRXP/USB30_2_SSRXN

RK3588 USB 控制器和 PHY 的内部连接关系，以及对应的常见 USB 物理接口如下图 1 所示。

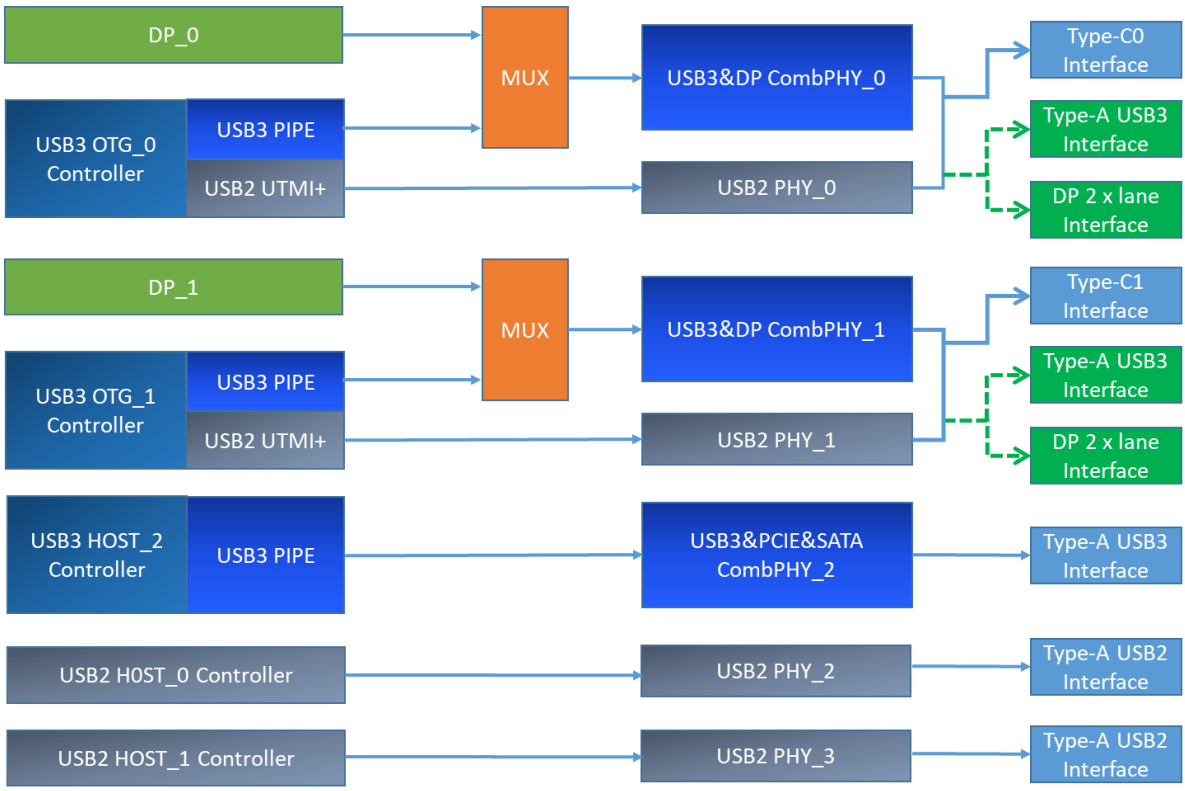

图 1 RK3588 USB 控制器和 PHY 的连接示意图

由图 1 可以看出：

1. RK3588 最多可以同时支持 2 个全功能的 Type-C 接口，2 个 Type-A USB 2.0 接口，1 个 Type-A USB 3.1 only 接口（不向下兼容 USB 2.0）；
2. USB 3.1 OTG 控制器 与 DP 控制器复用 USB3.1/DP Combo PHY，可组成全功能的 Type-C 接口，也可以拆分独立使用（如：常见的 Type-A USB 3.1 接口 + DP 接口[2 x lanes]）；
3. USB 3.1 HOST_2 控制器只支持 USB 3.1 功能，无法支持 USB 2.0（因为没有带 USB 2.0 PHY）。可以将 USB 3.1 HOST2 与 USB 2.0 HOST_0 或 USB 2.0 HOST_1 组合，作为一个完整的 Type-A USB 3.1 接口；
4. USB 3.1/DP Combo PHY 只能支持 USB 3.1 Gen1，不向下兼容 USB 2.0。所以，它们在芯片内部设计时，实际是与 USB 2.0 PHY 组合，以支持完整的 USB 3.1 协议功能。其中，USB 3.1/DP Combo PHY0 固定与 USB2 PHY0 组合，USB 3.1/DP Combo PHY1 固定与 USB2 PHY1 组合；
5. USB 3.1/SATA/PCIe Combo PHY 只能支持 USB 3.1 Gen1，不向下兼容 USB 2.0。并且，在芯片内部设计时，没有与 USB2 PHY 组合。因此，USB3 HOST_2 需要与 USB2 HOST_0/1 接口（二选一）组合，以支持完整的 USB 3.1 协议功能；
6. RK3588 和 RK3588S 的 USB 模块区别是：RK3588S 相比 RK3588 少了一组 Type-C1 (即：1 个 USB 3.1 OTG controller + 1 个 DP controller + 1 个 USB3.1/DP Combo PHY + 1 个 USB 2.0 PHY);

需要注意的是，RK3588 USB 支持的接口类型并不局限于图 1 所描述的 Type-C/A USB 接口类型，而是可以支持所有常见的 USB 接口，包括 Type-C USB 2.0/3.1，Type-A USB 2.0/3.1，Micro USB 2.0/3.1 等，具体信息请参考RK3588 USB 支持的接口类型。为了适配不同的 USB 电路设计和接口类型，Linux-5.10 内核 USB 驱动已经做了软件兼容，开发者只需要根据产品的 USB 硬件电路，对 Linux USB DTS 进行正确配置，即可使能对应的 USB 接口功能。详细的 USB DTS 配置方法，请参考 RK3588USB DTS 配置。

RK3588 USB 支持的接口类型

RK3588 USB 可以支持如下图 2 常见的 USB 接口类型，产品设计时，可以根据实际的应用场景需求，灵活设计 USB 硬件电路，同时，只要对 Linux USB DTS 进行适配即可。

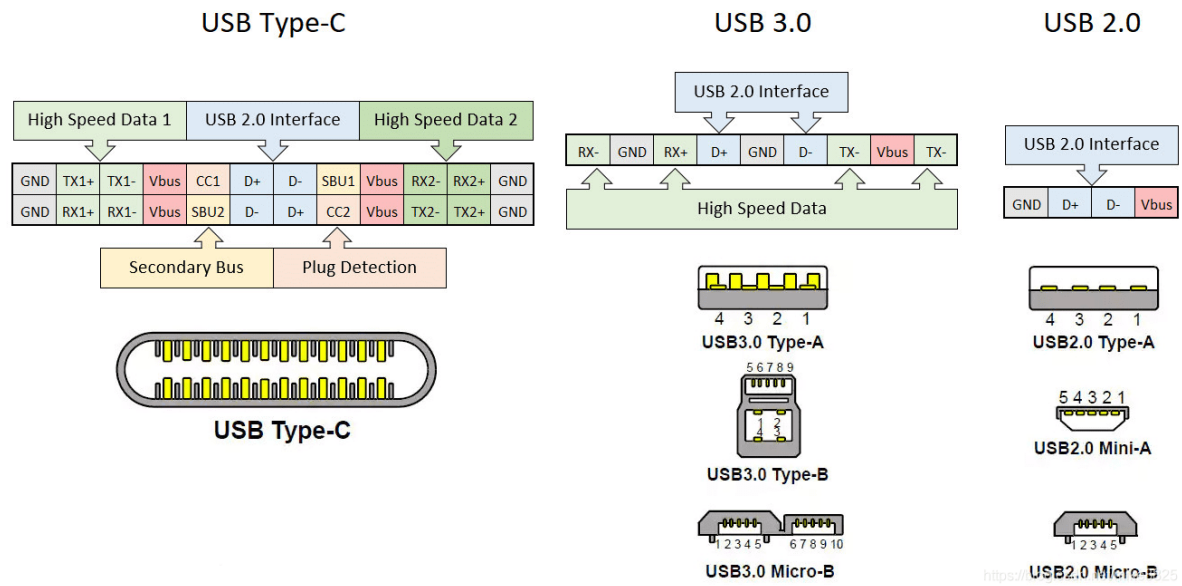

图 2 USB 接口类型

Type-C 接口类型

Type-C USB 3.1/DP 全功能接口

RK3588 Type-C0/1 可以支持全功能的 Type-C 接口功能[1]，如下图 3 所示，具体的硬件电路设计，请参考 Type-C USB 3.1/DP 全功能硬件电路。主要支持的功能如下：

- 支持 Type-C PD （需要配合外置 Type-C 控制器芯片）
- 支持 USB 3.1 Gen1 5Gbps 数据传输
- 支持 DP Alternate Mode

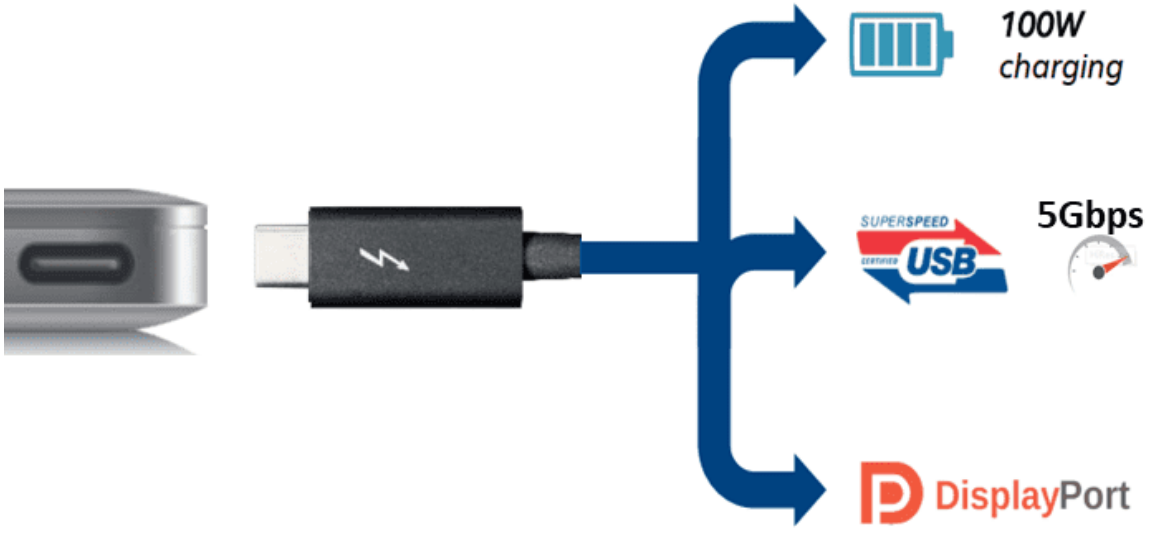

图 3 Type-C USB 3.1/DP 接口

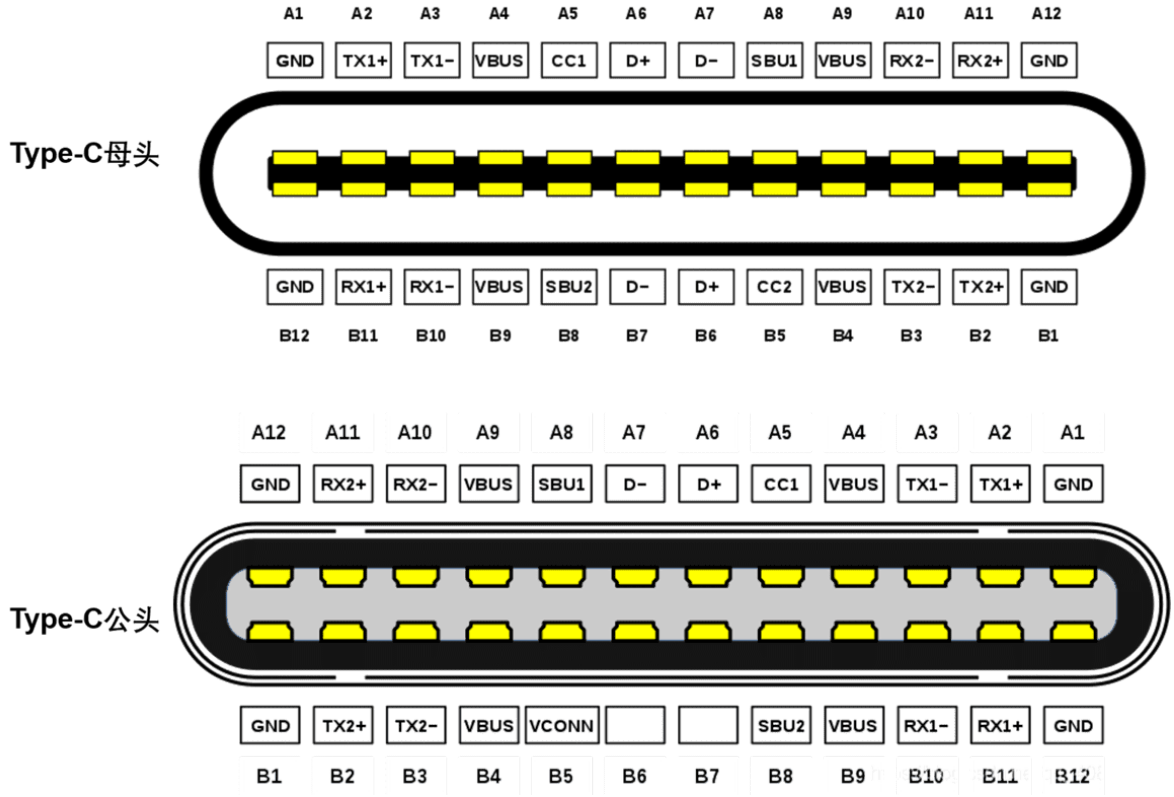

图 4 Type-C 接口引脚定义

表 5 Type-C 接口描述

Pin	名称	描述	Pin	名称	描述
A1	GND	接地	B12	GND	接地
A2	SSTXp1	SuperSpeed 差分信号 TX1+	B11	SSRXp1	SuperSpeed 差分信号 RX1+
A3	SSTXn1	SuperSpeed 差分信号 TX1-	B10	SSRXn1	SuperSpeed差分信号 RX1-
A4	VBUS	USB 总线电源	B9	VBUS	USB 总线电源
A5	CC1	Configuration channel	B8	SBU2	Sideband use (SBU)
A6	Dp1	USB 2.0 差分信号 D1+	B7	Dn2	USB 2.0 差分信号 D2-
A7	Dn1	USB 2.0 差分信号 D1-	B6	Dp2	USB 2.0 差分信号 D2+
A8	SBU1	Sideband use (SBU)	B5	CC2	Configuration channel
A9	VBUS	USB 总线电源	B4	VBUS	USB 总线电源
A10	SSRXn2	SuperSpeed 差分信号 RX2-	B3	SSTXn2	SuperSpeed差分信号 TX2-
A11	SSRXp2	SuperSpeed 差分信号 RX2+	B2	SSTXp2	SuperSpeed差分信号 TX2+
A12	GND	接地	B1	GND	接地

表 6 RK3588 Type-C0 与 Type-C 接口的连接关系

RK3588 Type-C0 Pin	Type-C 接口 Pin	关系描述
TYPEC0_SSRX1P/DP0_TX0P TYPEC0_SSRX1N/DP0_TX0N	B10/B11	连接到 RK3588 USBDP PHY 的 lane0，可用于 USB 3.1 Rx 或者 DP Tx
TYPEC0_SSTX1P/DP0_TX1P TYPEC0_SSTX1N/DP0_TX1N	A2/A3	连接到 RK3588 USBDP PHY 的 lane1，可用于 USB 3.1 Tx 或者 DP Tx
TYPEC0_SSRX2P/DP0_TX2P TYPEC0_SSRX2N/DP0_TX2N	A10/A11	连接到 RK3588 USBDP PHY 的 lane2，可用于 USB 3.1 Rx 或者 DP Tx
TYPEC0_SSTX2P/DP0_TX3P TYPEC0_SSTX2N/DP0_TX3N	B2/B3	连接到 RK3588 USBDP PHY 的 lane3，可用于 USB 3.1 Tx 或者 DP Tx
TYPEC0_OTG_DP/DM	A6/A7, B6/B7	连接到 RK3588 TYPEC0_OTG_DP/DM，其中，A6 和 B6并联，A7 和 B7 并联。
TYPEC0_SBU1/TYPEC0_SBU2	A8/B8	连接到 RK3588 USBDP PHY 的 AUX，只用于 DP Alternate Mode
TYPEC0_SBU1_DC/TYPEC0_SBU2_DC	A8/B8	连接到 RK3588 GPIO，用于软件控制 DP AUX 传输时的上拉。对 GPIO 默认的上下拉方式没要求
TYPEC0_CC1/TYPEC0_CC2	A5/B5	连接到外置 Type-C 控制器芯片 (HUSB311/FUSB302)，未连接到 R3588 SoC

Type-C to Type-A USB 3.1/DP 接口

RK3588 Type-C0/1 可以拆分为独立的 Type-A USB 3.1 接口和 DP 接口使用。

具体的硬件电路设计，请参考 Type-C to Type-A USB 3.1/DP 硬件电路。

以 RK3588 EVB2 的 Type-C to Type-A USB 3.1/DP 接口设计为例：

Type-C0：Type-A USB 3.1 (使用 USBDP PHY0 的 lane0/1) + DP 1.4 (使用 USBDP PHY0 的 lane2/3);

Type-C1：Type-A USB 3.1 (使用 USBDP PHY1 的 lane0/1) + DP to VGA (使用 USBDP PHY1 的 lane2/3);

Note:

理论上，硬件可以分配 Type-A USB 3.1 使用 lane2/3，DP 使用 lane0/1，同时，软件只要修改 Linux usbdp_phy 节点的属性 `rockchip,dp-lane-mux` 进行适配。

Type-C to Type-A USB 2.0/DP 接口

RK3588 Type-C0/1 可以拆分为独立的 Type-A USB 2.0 接口和 DP（4 x Lane）接口使用。

具体的硬件电路设计，请参考 Type-C to Type-A USB 2.0/DP 硬件电路。

以 RK3588 NVR DEMO board 的 Type-C1 to Type-A USB 2.0/DP 接口设计为例：

Type-C1：Type-A USB 2.0 (未使用 USBDP PHY) + DP 4 x lane to HDMI2.0 (使用 USBDP PHY1 的 lane0/1/2/3)

Type-C USB 2.0 only 接口

RK3588 Type-C0/1 可以简化为 Type-C USB 2.0 only 接口。如下图 5 所示：

- 支持 Type-C PD （需要配合外置 Type-C 控制器芯片）
- 支持 USB 2.0 480Mbps 数据传输
- 不支持 DP Alternate Mode

这种设计方式，主要目的是为了简化硬件电路设计，但会降低 USB 最大传输速率。同时，为了适配这种接口设计，需要对 Linux USB DTS 进行较大的修改，请参考Type-C USB 2.0 only DTS 配置。

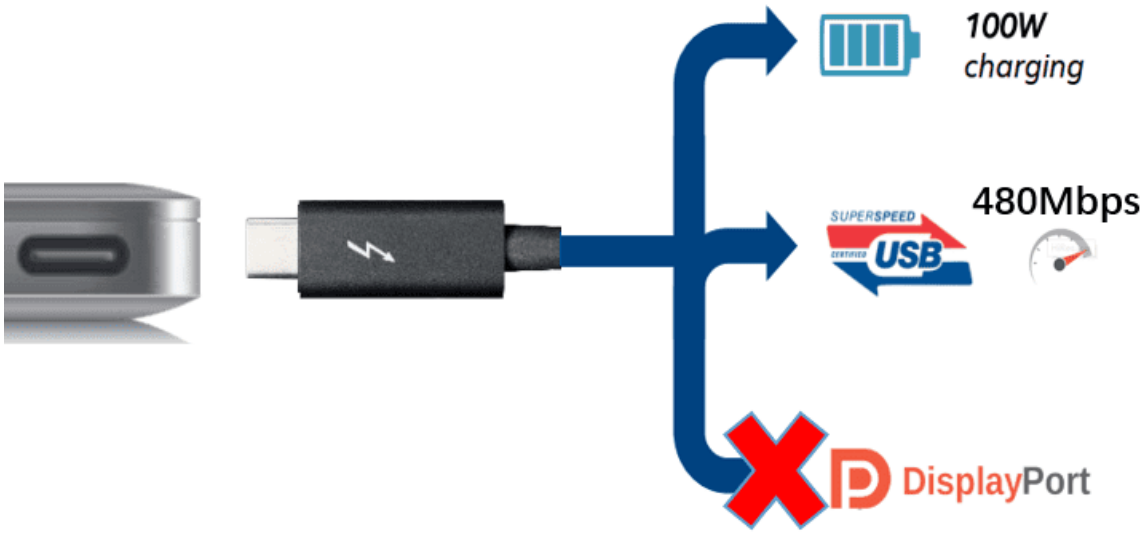

图 5 Type-C USB 2.0 only 接口

Type-A 接口类型

Type-A USB 3.1 接口

RK3588 最多可以支持 3 个 Type-A USB 3.1 接口，包括：

- Type-C0 to Type-A USB 3.1
- Type-C1 to Type-A USB 3.1
- USB3_HOST2 + USB 2.0 HOST0/1

具体的硬件电路设计，请参考 Type-A USB 3.1 硬件电路。

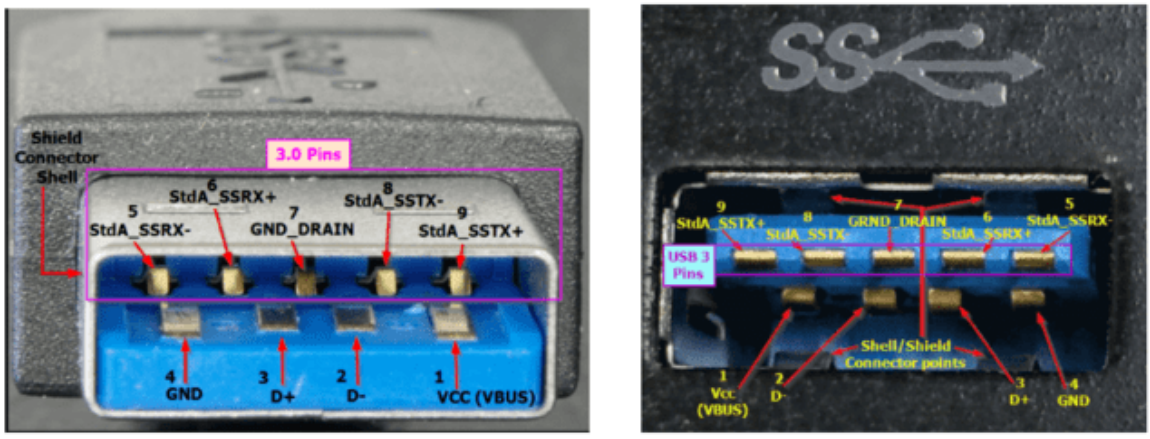

图 6 Type-A USB 3.1 接口

Type-A USB 2.0 接口

RK3588 最多可以支持 4 个 Type-A USB 2.0 接口，包括：

- Type-C0 to Type-A USB 2.0
- Type-C1 to Type-A USB 2.0
- USB 2.0 HOST0
- USB 2.0 HOST1

具体的硬件电路设计，请参考 Type-A USB 2.0 硬件电路。

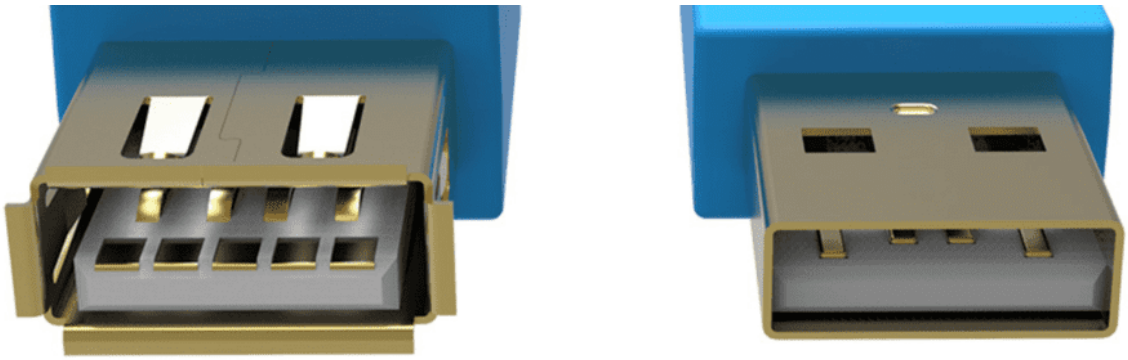

图 7 Type-A USB 2.0 接口

Micro 接口类型

Micro USB 3.1 接口

RK3588 Type-C0/1 可以支持 Micro USB 3.1 的接口设计。但考虑到 Micro USB 3.1 接口占用的 PCB 面积较大，目前产品上使用较少。

Micro USB 3.1 与 Type-A USB 3.1的引脚区别，主要是增加了 OTG ID 脚，用于硬件自动检测 ID 电平和切换 OTG Device/HOST mode。OTG ID 脚需要连接到 RK3588 USB20_OTG_ID pin 脚，在芯片内部已经默认上拉 ID 到高电平 1.8V，外部电路不需要上拉。

图 8 Micro USB 3.1 接口

Micro USB 2.0 接口

RK3588 Type-C0 可以支持 Micro USB 2.0 的接口设计。这种设计方式，主要目的是为了简化硬件电路设计，但会降低 USB 最大传输速率。

Micro USB 2.0 与 Type-A USB 2.0 的引脚区别，主要是增加了 OTG ID 脚，用于硬件自动检测 ID 电平和切换 OTG Device/HOST mode。OTG ID 脚需要连接到 RK3588 USB20_OTG_ID pin 脚，在芯片内部已经默认上拉 ID 到高电平 1.8V，外部电路不需要上拉。

图 9 Micro USB 2.0 接口

RK3588 USB Config Map

RK3588 的 5 个独立的 USB 控制器和 7 个 独立的 USB PHY，可以支持如下图 10 所列出的配置方式。

Type-C0/1 可以支持 4 中配置：

Config0: Type-C0 with DP function

Config1: USB 2.0 OTG + DP 4 x Lane (Swap off)

Config2: USB 2.0 OTG + DP 4 x Lane (Swap on)

Config3: USB 3.1 OTG + DP 2 x Lane (Swap on)

Config4: USB 3.1 OTG + DP 2 x Lane (Swap off)

USB 2.0 HOST0/1 和 USB3_HOST2 支持的配置：

Config0: USB 2.0 HOST0 + USB 2.0 HOST1 (USB3_HOST2 not used)

Config1: USB 2.0 HOST0 + USB 2.0 HOST1 Combo with USB3_HOST2

Config2: USB 2.0 HOST0 Combo with USB3_HOST2 + USB 2.0 HOST1

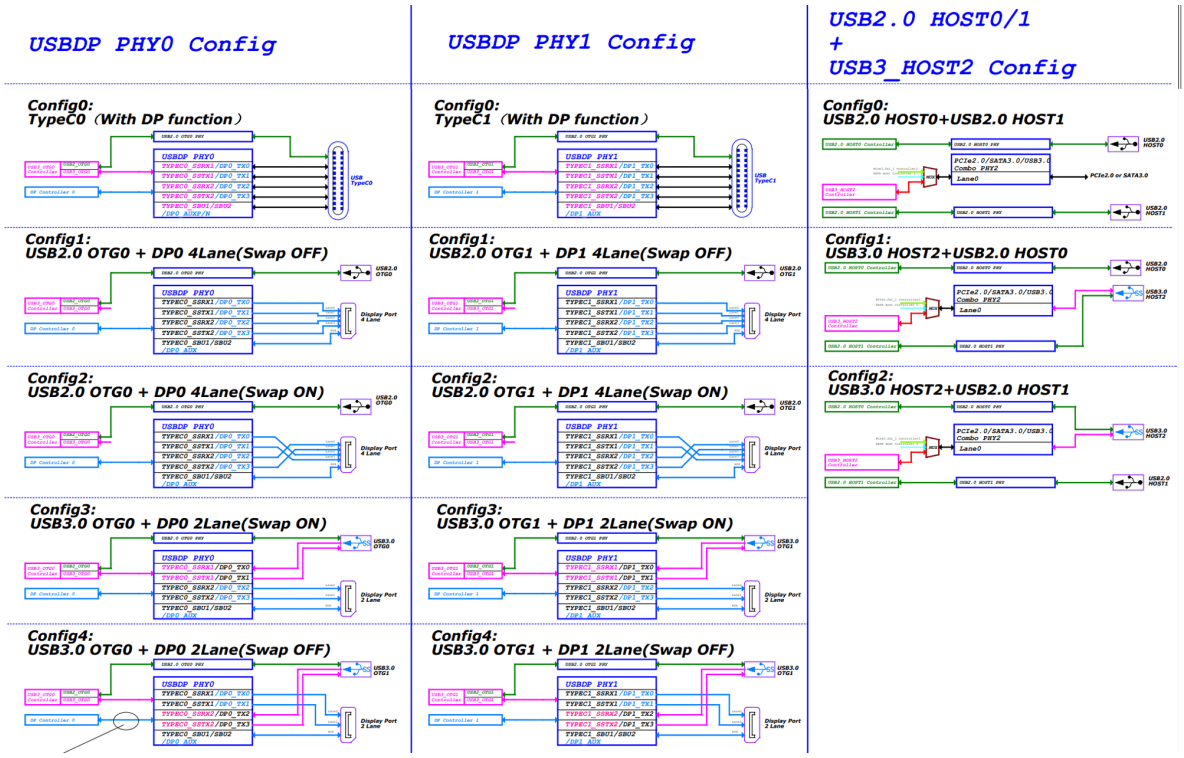

图 10 RK3588 USB Config Map

如果要了解更详细的 USB 配置表，请参考 SDK EVB 参考原理图的 USB Controller Configure Table。

需要注意的是，DP 控制器和 USBDP PHY 的 4 条 lane 支持任意映射，如下图 11 所示，只要修改 USBDP PHY 的 DTS 进行适配即可。如果开发者不清楚如何进行软件适配，建议参考图 10 列出的常规配置进行硬件电路设计。

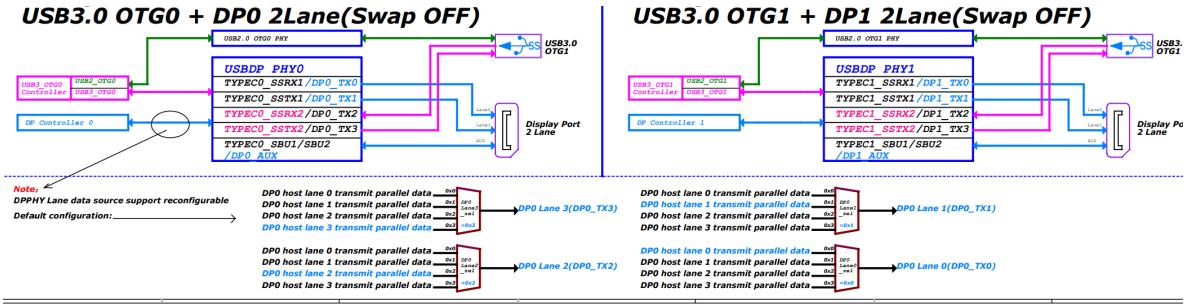

图 11 RK3588 DP Lane Map

RK3588 USB 硬件电路设计

本章节主要说明 RK3588 USB 在实际应用中，可以支持的各种硬件电路设计方案。如下图 12 是 RK3588 USB 接口框图，由图中可以看出，RK3588 可以支持的接口如下：

- USB20 HOST0
- USB20 HOST1
- USB30/DP1.4 MULTI0
- USB30/DP1.4 MULTI1
- USB30/PCIE2.0/SATA30 MULTI2

RK3588S 少了 USB30/DP1.4 MULTI1 这组接口。

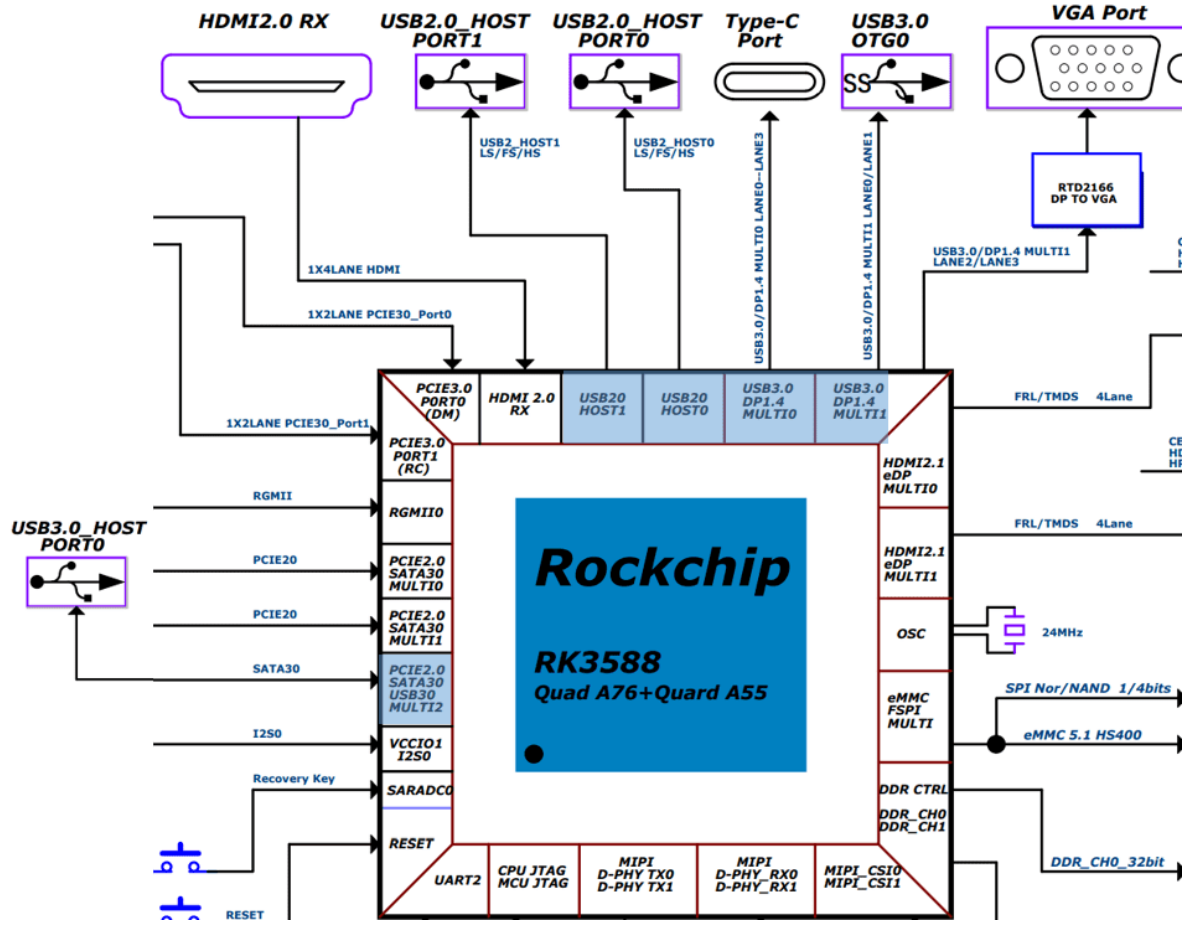

图 12 RK3588 USB 接口框图

USB 控制器供电及功耗管理

RK3588 USB 控制器的供电电源是 VDD_LOGIC。同时，芯片内部有设计 USB 控制器专用的 power domain，各个 USB 控制器对应的 Power Domain 如下表 7 所示。

表 7 USB 控制器和 PD 的对应关系

USB 控制器	Power Domain
USB 2.0 HOST0/1	PD_USB
USB 3.1 OTG0/1	PD_USB
USB 3.1 HOST2	PD_PHP

在实际使用场景中，Linux USB 控制器驱动会根据 USB 接口的工作情况，基于 Linux PM Runtime 机制，动态开关 USB 控制器的 PD，以降低 USB 控制器的功耗。而当系统进入二级待机时，为了达到最优功耗的目的，软件会强制关闭 USB 控制器的所有 PD。因此，在实际产品的应用场景中，如果需要在二级待机时，保持 USB 控制器的寄存器工作状态，则需要在 USB 控制器驱动中调用函数 `device_init_wakeup`，避免二级待机时关闭 USB 控制器的 PD。

USB 控制器的功耗控制策略如下：

1. 对于不使用的 USB 控制器，需要将对应的控制器 DTS 节点配置为 disabled；
2. 对于内核已启用的 USB 控制器，内核 USB 驱动已经支持 USB 控制器 Auto suspend 功能 (当 USB HOST 接口不接任何外设时，控制器自动进入 suspend 低功耗状态)，因此，开发者不需要对 USB 控制器的动态功耗管理进行调试。

内核 disable USB 2.0 HOST0/1 的方法如下：

```
# Disable USB 2.0 HOST0
&usb_host0_ehci {
        status = "disabled";
};

&usb_host0_ohci {
        status = "disabled";
};

# Disable USB 2.0 HOST1
&usb_host1_ehci {
        status = "disabled";
};

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

USB PHY 供电及功耗管理

USB 2.0 PHY 供电及功耗管理

RK3588 支持 4 个 独立的 USB 2.0 PHY。RK3588S 相比 RK3588 少了 1 个 USB 2.0 PHY1 。在芯片内部，所有 USB 2.0 PHY 都属于 PD_BUS (Alive)，并且，所有 USB 2.0 PHY 共用如下图 13 所示的 3 路供电电源。因此，在系统运行时，无法通过硬件断电和关闭 PD 的简单方法，来降低 USB 2.0 PHY 的功耗。

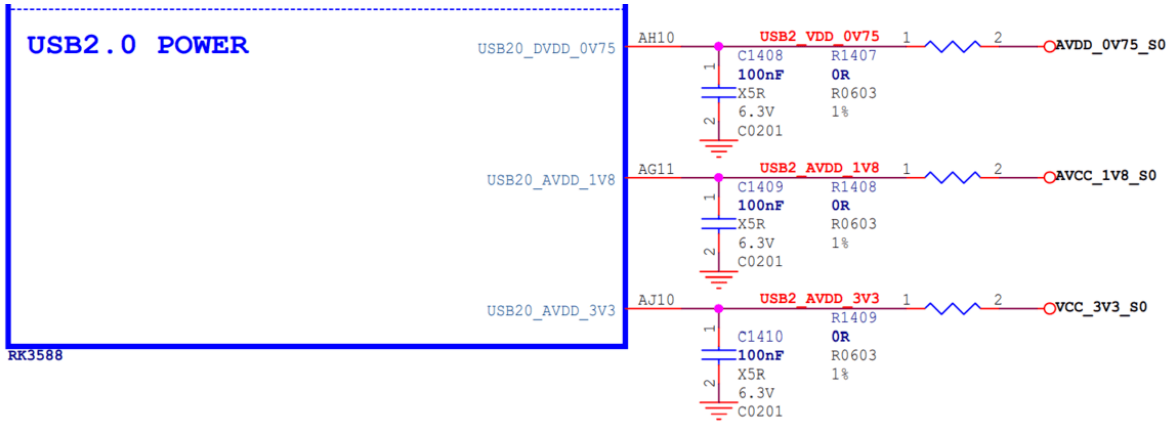

图 13 USB 2.0 PHY 供电电源

需要注意的是，在实际电路中，USB 2.0 PHY 的供电电压值超过规定的最大值或者低于规定的最小值，可能会导致 USB 连接异常。

表 8 USB 2.0 PHY 供电电压要求

供电电源	最小	正常	最大	Unit
USB20_DVDD_0V75	0.6975	0.75	0.825	V
USB20_AVDD_1V8	1.674	1.8	1.98	V
USB20_AVDD_3V3	3.069	3.3	3.63	V

USB 2.0 PHY 的功耗控制策略如下：

1. 为了支持 Maskrom USB 下载固件的功能，必须保证 USB 2.0 PHY 的供电正常；
2. 系统上电后，所有 USB 2.0 PHY 默认处于 Normal mode，软件在 U-Boot SPL 阶段，配置 USB 2.0 PHY1/PHY2/PHY3 处于最低功耗 IDDQ mode（SDK 已经支持），在进入系统后，内核 USB 驱动会根据应用需求，设置对应的 USB 2.0 PHY 退出 IDDQ mode；
3. 对于不使用的 USB 2.0 PHY，需要将对应的 USB 2.0 PHY DTS 节点配置为 disabled (参考表 9)；
4. 对于内核已启用的 USB 2.0 PHY，内核 USB 2.0 PHY 驱动会自动对 PHY 进行动态功耗控制，当检测到有设备插入时，自动设置 USB 2.0 PHY 处于 Normal mode，当检测到没有设备插入时，自动设置 USB 2.0 PHY 处于 Suspend mode；

USB 2.0 PHY 处于不同工作模式的功耗数据如下表 9 所示。

表 9 USB 2.0 PHY 功耗数据

(统计单个 USB 2.0 PHY 的功耗)

供电电源	读写数据	动态休眠	PHY disabled	二级待机	Unit
USB20_DVDD_0V75	7.1	2.6	0.05	0	mA
USB20_AVDD_1V8	17.8	3.1	0.05	0	mA
USB20_AVDD_3V3	3.3	0.05	0.05	0	mA

Note:

- 读写数据功耗的测试场景：接 U2 盘拷贝数据，PHY 处于 Normal mode；
- 动态休眠功耗的测试场景：USB 2.0 PHY 的 DTS enable，但不接 USB 外设，PHY 处于 Suspend mode；
- PHY disabled 功耗的测试场景：USB 2.0 PHY 的 DTS diabled，PHY 处于 IDDQ mode；

- 二级待机功耗的测试场景：USB 2.0 PHY 的三路供电电源全部关闭；

表 10 USB 2.0 PHY 和 USB 控制器的连接关系

USB 2.0 PHY	USB 控制器
USB 2.0 PHY0	USB 3.1 OTG0
USB 2.0 PHY1	USB 3.1 OTG1
USB 2.0 PHY2	USB 2.0 HOST0
USB 2.0 PHY3	USB 2.0 HOST1

内核 disable USB 2.0 PHY2/3 的方法如下：

```
&u2phy2 {
        status = "disabled";
};

&u2phy3 {
        status = "disabled";
};

&u2phy2_host {
        status = "disabled";
};

&u2phy3_host {
        status = "disabled";
};
```

USB 3.1 PHY 供电及功耗管理

RK3588 支持两种 USB 3.1 Combo PHY：

1. USB 3.1/DP Combo PHY
2. USB 3.1/SATA/PCIe Combo PHY

表 11 USB 3.1 Combo PHY 和 USB 控制器的连接关系

USB 3.1 Combo PHY	USB 控制器
USB 3.1/DP Combo PHY0	USB 3.1 OTG0
USB 3.1/DP Combo PHY1	USB 3.1 OTG1
USB 3.1/SATA/PCIe Combo PHY2	USB 3.1 HOST2

这两种 USB 3.1 Combo PHY 对应的供电电源和功耗控制方式都不一样，下面分别进行说明。

USB 3.1/DP Combo PHY

RK3588 支持两个独立的 USB 3.1/DP Combo PHY 。RK3588S 相比 RK3588 少了 1 个 USB 3.1/DP Combo PHY1。在芯片内部，两个 USB 3.1/DP Combo PHY 都属于 PD_BUS (Alive)，在芯片外部，两个 PHY 有独立的供电电源 pin，如图 14 所示。

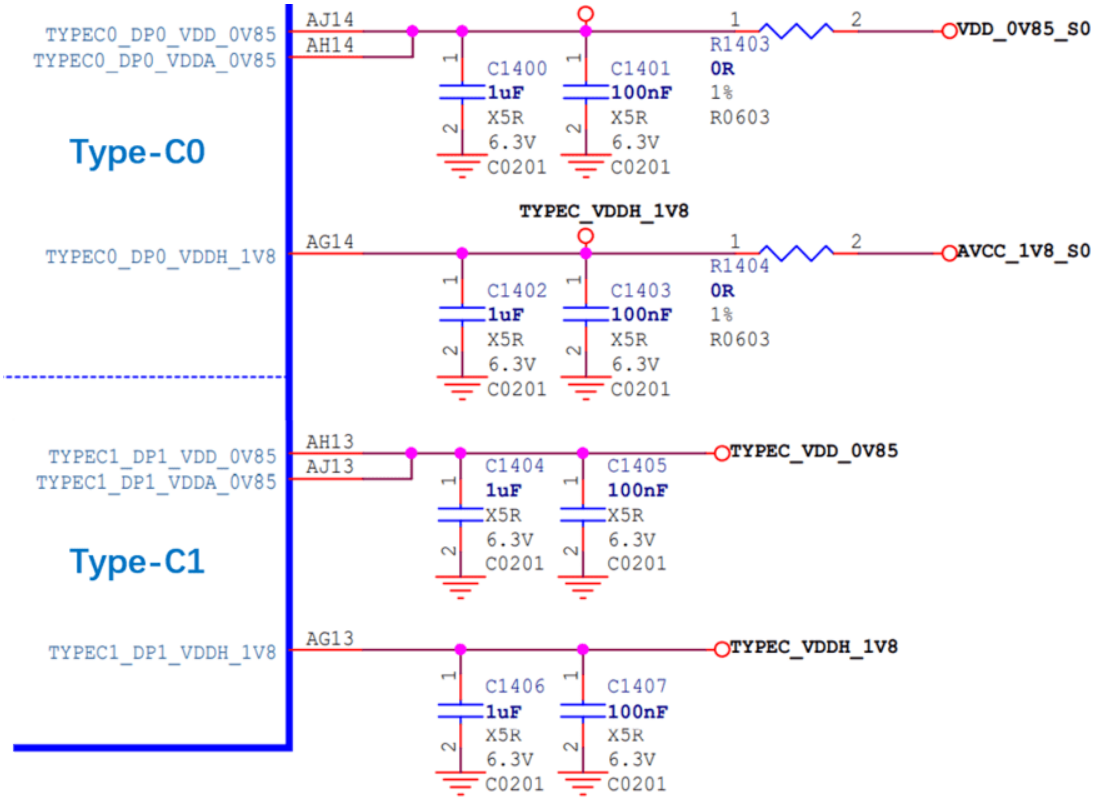

图 14 RK3588 USB 3.1/DP Combo PHY 供电电源

表 12 USB 3.1/DP Combo PHY 供电电压要求

供电电源	最小	正常	最大	Unit
VDD_0V85/VDDA_0V85	0.8075	0.85	0.8925	V
VDDH_1V8	1.71	1.8	1.89	V

USB 3.1/DP Combo PHY 的功耗控制策略如下：

1. **为了支持 Maskrom USB 下载固件的功能，必须保证 Type-C0 USBDP PHY0 的供电正常**；
2. 当 Type-C1 接口不使用时，对应的 Type-C1 USBDP PHY1 可以不供电；
3. 系统上电后，USBDP PHY 处于未初始化状态时的功耗最低；
4. 对于不使用的 USBDP PHY，需要将对应的 USBDP PHY DTS 节点配置为 disabled，也即让 PHY 处于未初始化状态，功耗最低；
5. 对于内核已启用的 USBDP PHY，内核 USBDP PHY 驱动会自动对 PHY 进行动态功耗控制，当检测到有设备插入时，自动设置 USBDP PHY 处于 P0 State，当检测到没有设备插入时，自动设置 USBDP PHY 处于 P3 State (应用于 Type-A 接口)或者处于 reset state (应用于 Type-C 接口)；

表 13 USB 3.1/DP Combo PHY 功耗数据

(统计单个 USB 3.1/DP Combo PHY 的功耗)

供电电源	读写数据	动态休眠[1]	动态休眠[2]	PHY disabled	二级待机	Unit
VDD_0V85/VDDA_0V85	115.7	6.8	7.9	2	0	mA
VDDH_1V8	29.4	0	2	0	0	mA

Note：

- 读写数据功耗的测试场景：接 U3 盘拷贝数据，PHY 处于 P0 state；
- 动态休眠功耗[1]的测试场景：Type-C 接口，不接 USB 外设，PHY 处于 reset state；
- 动态休眠功耗[2]的测试场景：Type-A 接口，不接 USB 外设，PHY 处于 P3 state
- PHY disabled 功耗的测试场景：PHY 的 DTS 节点配置为 disabled，PHY 处于未初始化状态，功耗最低；
- 二级待机功耗的测试场景：PHY 的两路供电电源全部关闭；

内核 disable USBDP PHY1 的方法如下：

```
&usbdp_phy1 {
        status = "disabled";
};

&usbdp_phy1_dp {
        status = "disabled";
};

&usbdp_phy1_u3 {
        status = "disabled";
};
```

USB 3.1/SATA/PCIe Combo PHY

RK3588 支持 1 个 USB3.1/SATA/PCIe Combo PHY。在芯片内部，这个 PHY 属于 PD_BUS (Alive)，在芯片外部，PHY 有独立的供电电源，如图 15 所示。

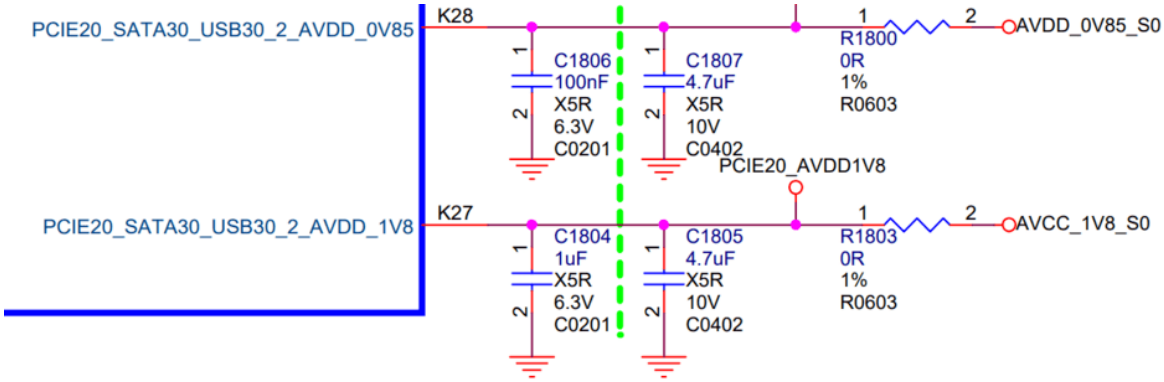

图 15 RK3588 USB 3.1/SATA/PCIe Combo PHY 供电电源

表 14 USB 3.1/SATA/PCIe Combo PHY 供电电压要求

供电电源	最小	正常	最大	Unit
AVDD_0V85	0.8	0.85	0.935	V
AVDD_1V8	1.62	1.8	1.98	V

USB 3.1/SATA/PCIe Combo PHY 的功耗控制策略如下：

1. 芯片上电时，USB 3.1/SATA/PCIe Combo PHY 默认处于工作状态。软件在 U-Boot SPL 阶段，配置设置 PHY 处于 reset state，以保持 PHY 处于最低功耗。进入内核后，USB 控制器驱动会通过调用 `rockchip_combphy_init()` 函数释放 PHY 的 reset。
2. PHY 的动态功耗控制：当 Combo PHY 工作在 USB mode 时，PHY 的 PIPE state (P0/P1/P2/P3) 由 USB 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，未插入任何 USB 设备时，PIPE 处于 P3 state；插入 U3 disk 时，则切换到 P0 state；当接 U3 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 低

功耗。当有 USB 外设插入U3 HUB，则 PIPE state 自动切换为 P0。(注：P0 为正常工作状态，P3 为最低功耗状态)
3. 当明确不使用 USB 3.1 HOST2 接口时，对应的 USB 3.1/SATA/PCIe Combo PHY 可以不供电；
4. 在 PHY 供电的情况下，如果不使用这个PHY，需要将对应的 PHY DTS 节点配置为 disabled，也即让 PHY 处于 reset state，功耗最低；

表 15 USB 3.1/SATA/PCIe Combo PHY 功耗数据

供电电源	读写数据	动态休眠	PHY disabled	二级待机	Unit
AVDD_0V85	43.4	43.3	0.4	0	mA
AVDD_1V8	4.3	4.1	0.2	0	mA

Note:

- 读写数据功耗的测试场景：接 U3 盘拷贝数据，PHY 处于 P0 state。
- 动态休眠的测试场景：PHY DTS enable，但不接 USB 外设，PHY 处于 P3 State;
- PHY disabled 的测试场景：PHY 的 DTS 节点配置为 disabled，PHY 处于 reset state；
- 二级待机功耗的测试场景：PHY 的两路供电电源全部关闭；

内核 disable USB 3.1/SATA/PCIe Combo PHY 的方法如下：

```
&combphy2_psu {
        status = "disabled";
};
```

USB 硬件电路设计

TYPEC0_USB20_VBUSDET 电路设计

TYPEC0_USB20_VBUSDET 用于 USB Device 的使用场景，检测 USB Device 的连接和断开。
TYPEC0_USB20_VBUSDET 的设计注意点如下：

- 当 Type-C0 设计为支持 PD 功能的 Type-C 接口时，即支持外置 Type-C 控制器芯片（如：FUSB302 或者HUSB311），则参考 RK3588 EVB1 Type-C0 的电路设计即可（TYPEC0_USB20_VBUSDET 固定上拉到 VCC_3V3_S0），软件驱动可以通过 Type-C 控制器芯片的 CC 检测 USB Device 的连接和断开;
- 对于其他没有支持外置 Type-C 控制器芯片的电路设计方案（如Type-C0 USB 2.0 only，Type-A USB 3.1，Micro USB 2.0/3.1），要求 TYPEC0_USB20_VBUSDET 仍然按照传统的分压电路设计，连接到 USB 接口的VBUS 引脚，VBUSDET 不作常供电的设计（如果有作 USB HOST 的需求，需要独立的 GPIO 或者 PMIC VBUS 控制，不与其他 USB HOST 接口复用）；
- 要求芯片输入端 TYPEC0_USB20_VBUSDET 的高电平范围在 **[0.9V ~ 3.3V]**;

Maskrom USB 电路设计

RK3588 Maskrom 固定使用 Type-C0 USB 2.0 作为下载固件的功能。相关的电路设计如下图 16 所示。
为了保证 Maskrom USB 下载功能正常，要求如下：

- TYPEC0_USB20_VBUSDET 要能支持外部拉高，不能悬空；
- TypeC0 USBDP PHY 的供电电源（TYPEC0_DP0_VDD_0V85/TYPEC0_DP0_VDDA_0V85/TYPEC0_DP0_VDDH_1V8）和外部参考电阻（TYPEC0_DP0_REXT）都要正常连接，进系统后，软件会对 TypeC0 USBDP PHY 进行低功耗处理。

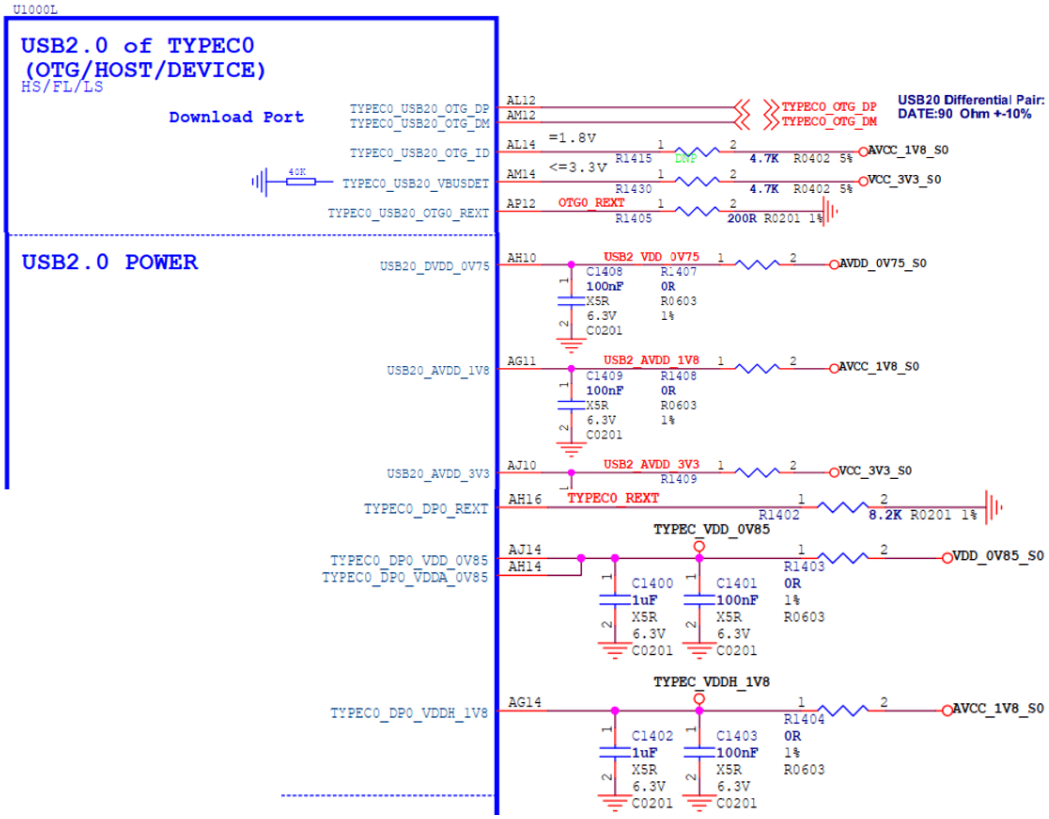

图 16 RK3588 Maskrom USB2 固件下载口电路

Type-C USB 3.1/DP 全功能硬件电路

该方案适用于 RK3588 Type-C0 和 Type-C1。

以 RK3588 EVB1 Type-C0 硬件电路设计为例。

1. RK3588 Type-C0 各个 Pin 与 Type-C 接口的详细连接关系，请参考 Type-C USB 3.1/DP 全功能接口 的表 5；
2. Type-C 电路需要配合外置的 Type-C 控制器芯片，才能实现 Type-C 的完整功能（包括：正反面检测、高电压大电流的 PD 协议交互等）。RK3588 可支持大部分常用的 Type-C 控制器芯片，具体请参考Type-C 控制器芯片支持列表；
3. 为了支持高压充电功能，同时降低硬件电路的风险，TYPEC0_USB20_VBUSDET 不要连接 Type-C 接口的 VBUS，固定上拉到 3.3V 即可（如图17 TYPEC0_USB20_VBUSDET 连接到 VCC_3V3_S0），但不能悬空；
4. TYPEC0_USB20_OTG_ID 只用于 Micro 接口类型的 OTG 功能，Type-C 电路不需要使用，悬空即可。
5. TYPEC0_SBU1/TYPEC0_SBU2 只用于 DP Alternate Mode 的 AUX 通信。按照 AUX 的协议要求，需要根据 Type-C 插入的正反面，对 SBU1/SBU2 进行相应的电平上拉操作。 因为 RK3588 芯片内部没有实现 SBU1/SBU2 的自动上拉，所以要求硬件外部电路增加两个 GPIO 控制（对应图 19 中的TYPEC0_SBU1_DC/TYPEC0_SBU2_DC）。对 GPIO 的默认上下拉方式没要求，可以选择任意的 GPIO。软件上，需要修改 usbdp_phy 节点的属性 `sbu1-dc-gpios` 和 `sbu2-dc-gpios` 进行适配；
6. Type-C USB DTS 的软件配置，请参考 Type-C USB 3.1/DP 全功能 DTS 配置；

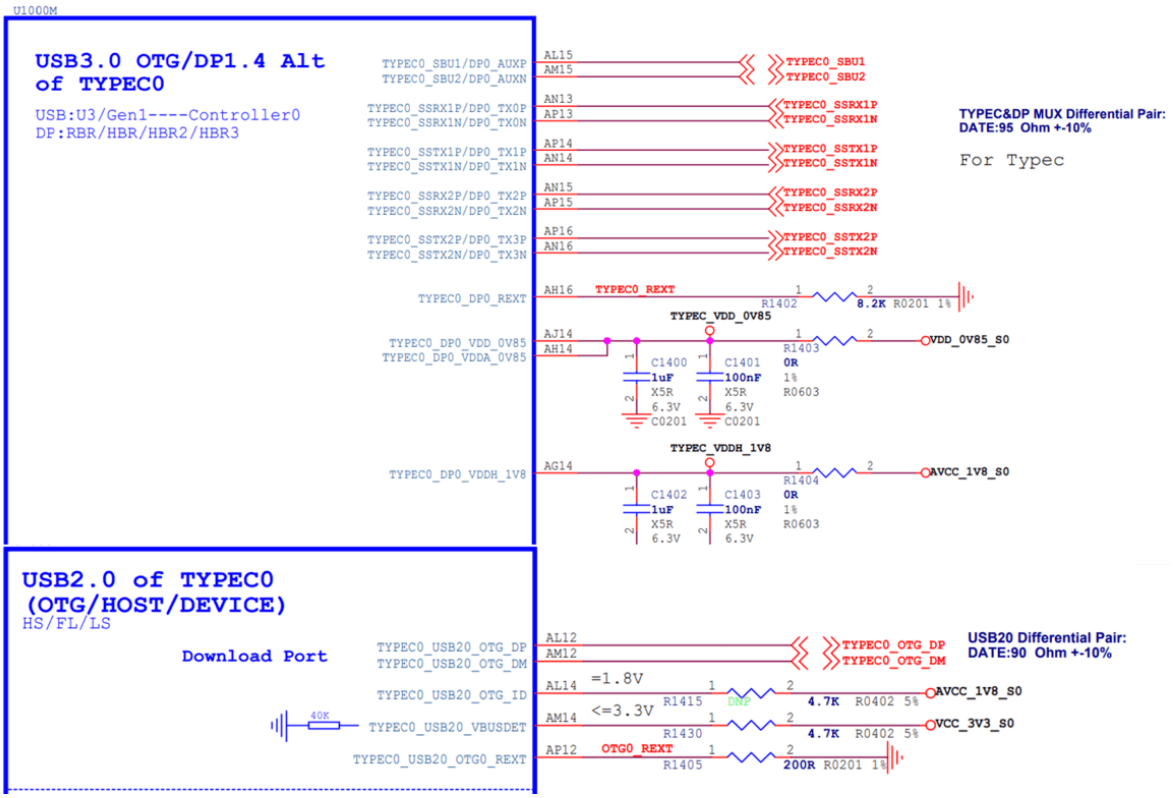

图 17 RK3588 Type-C0 电路

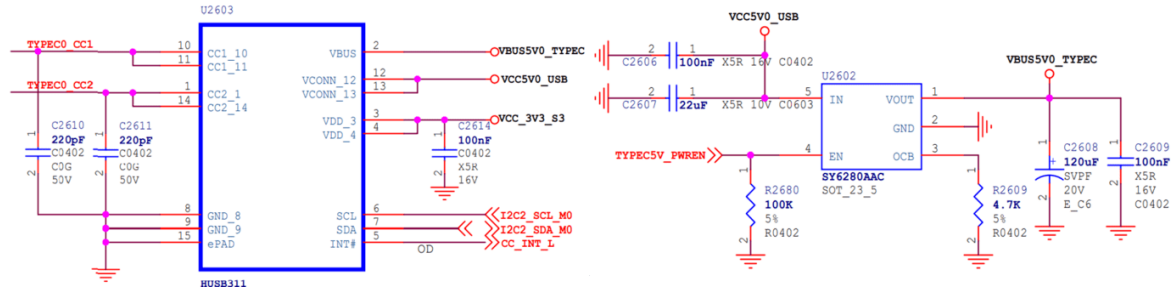

图 18 RK3588 Type-C0 PD 控制器电路和VBUS控制电路

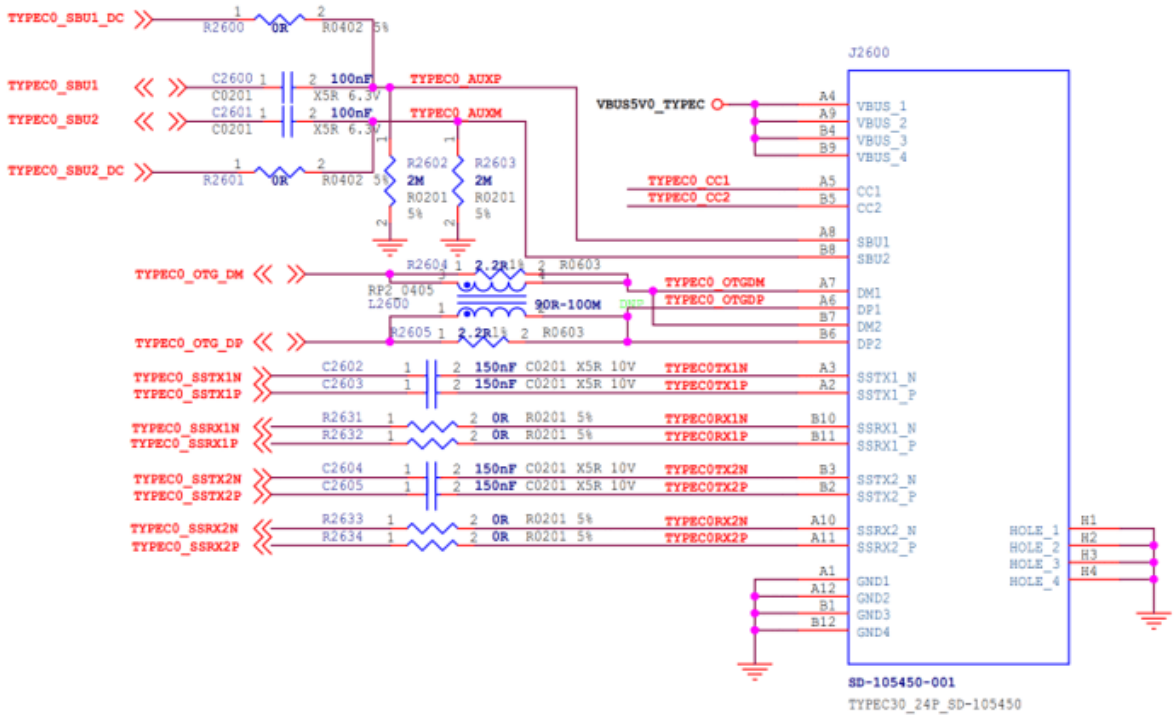

图 19 RK3588 Type-C0 接口

Type-C to Type-A USB 3.1/DP 硬件电路

该方案适用于 RK3588 Type-C0 和 Type-C1，可以拆分为独立的 Type-A USB 3.1 接口和 DP（2 x Lane）接口使用。

以 RK3588 EVB2 Type-C0 to Type-A USB 3.1/DP 硬件电路设计为例。

1. Type-A USB 3.1 使用 TYPEC0_SSRX1P/N 和 TYPEC0_SSTX1P/N (对应芯片内部 USBDP PHY 的 lane0/1)，而 DP 使用 DP0_TX2P/N 和 DP0_TX3P/N (对应芯片内部 USBDP PHY 的 lane2/3)；
2. 如果 USB OTG 有作 Device/HOST 的应用场景，建议 TYPEC0_OTG_VBUSDET 通过 30KΩ 的电阻串联到 Type-A USB 接口的 VBUS；
3. Type-A VBUS 的供电电源 (VCC5V0_USB30_HOST2) 由 GPIO 控制，当 OTG 作 Device mode，关闭 VBUS 输出。当 OTG 作 HOST mode，打开 VBUS 输出。此外，稳压芯片 SY6280AAC 的输出电流由 OCB pin 连接的电阻决定，最大电流 Ilim(A)=6800/Rset(ohm)，如下图 21 所示，VBUS 输出限流为 1A；
4. Type-C to Type-A USB 3.1/DP 对应的 DTS 配置，请参考 Type-C to Type-A USB 3.1/DP DTS 配置；

Note：

理论上，硬件电路也可以设计为 Type-A USB 3.1 使用 lane2/3，DP 使用 lane0/1，但软件需要对 usbdp_phy 节点的属性 `rockchip,dp-lane-mux` 进行修改，以适配硬件设计。

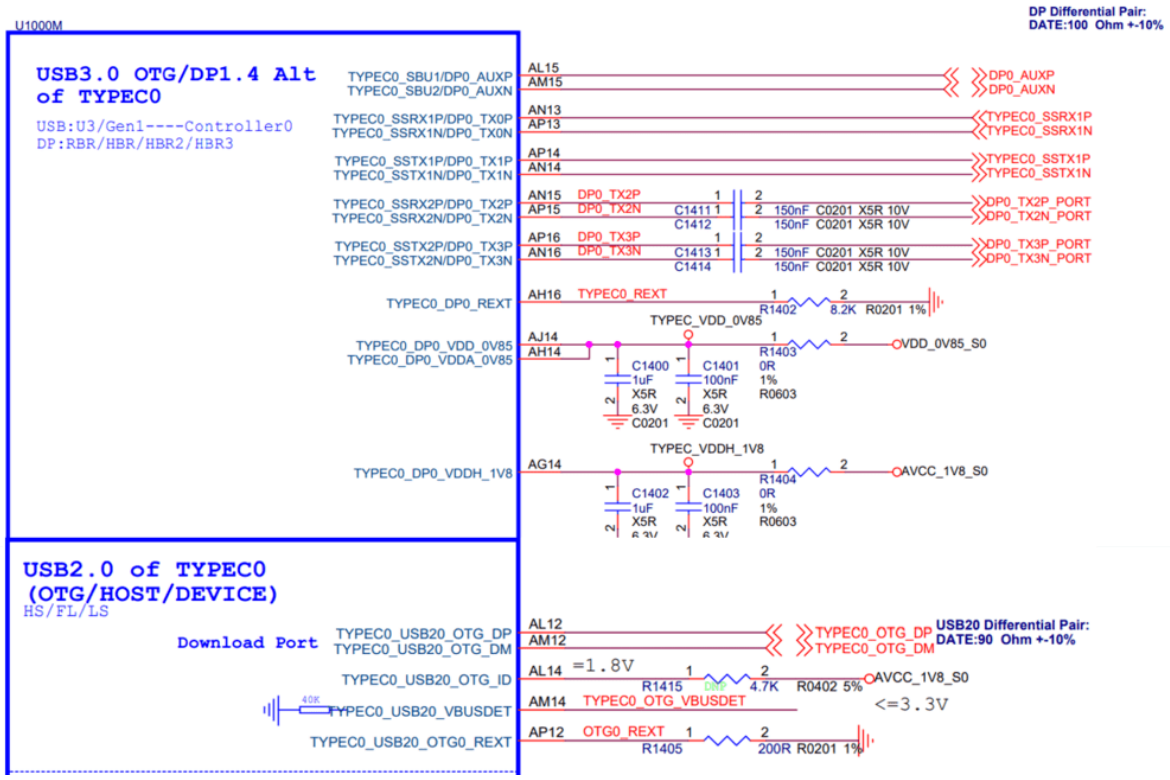

图 20 RK3588 Type-A USB 3.1/DP 电路

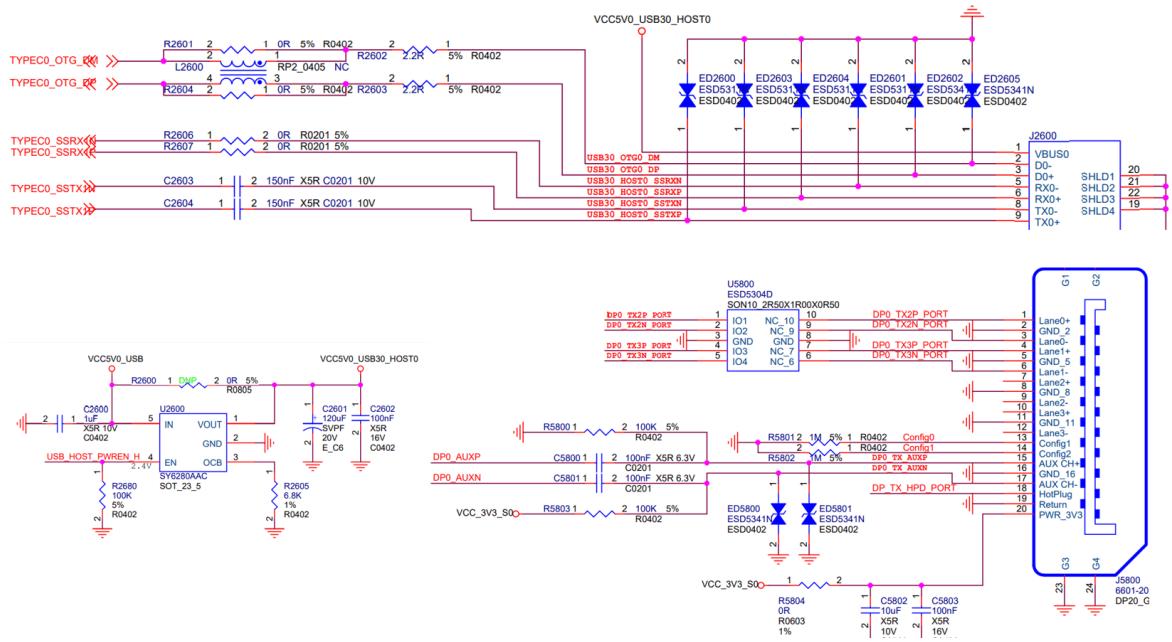

图 21 RK3588 Type-A USB3.1/DP 接口

Type-C to Type-A USB 2.0/DP 硬件电路

该方案适用于 RK3588 Type-C0 和 Type-C1，可以拆分为独立的 Type-A USB 2.0 接口和 DP（4 x Lane）接口使用。

以 RK3588 NVR Demo Board Type-C1 to Type-A USB 2.0/DP 硬件电路设计为例。

1. TYPEC1 USBDP PHY 的 4 x Lane 全部给 DP 接口使用，USB 不使用 USBDP PHY；
2. TYPEC1 USB 只作 HOST mode 使用时，TYPEC1_USB20_OTG_ID 和 TYPEC1_USB20_VBUSDET 悬空即可；
3. Type-A VBUS 的供电电源 (VCC5V0_USB20_HOST) 由 GPIO 控制，当 OTG 作 HOST mode，打开 VBUS 输出。此外，稳压芯片 SY6280AAC 的输出电流由 OCB pin 连接的电阻决定，最大电流 Ilim(A)=6800/Rset(ohm)，如下图 23 所示，VBUS 输出限流为 1.45A；
4. Type-C to Type-A USB 2.0/DP 对应的 DTS 配置，请参考 Type-C to Type-A USB 2.0/DP DTS 配置；

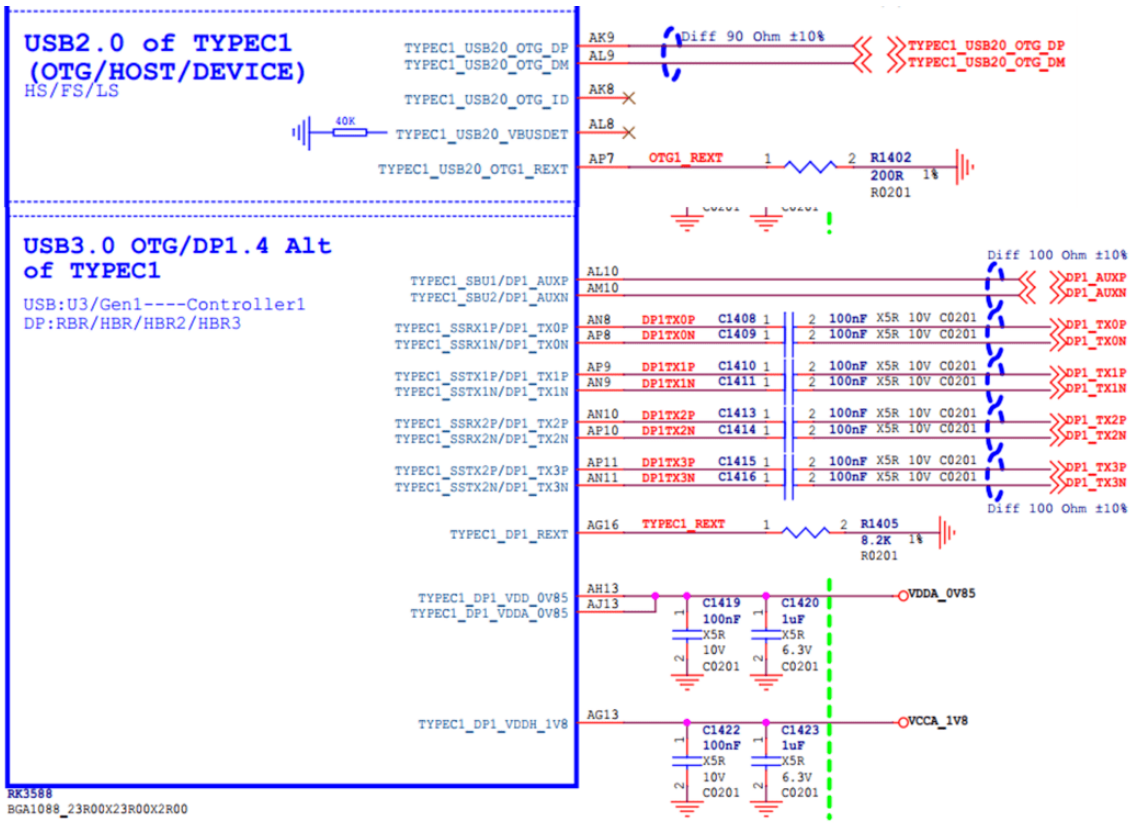

图 22 RK3588 Type-A USB 2.0/DP 电路

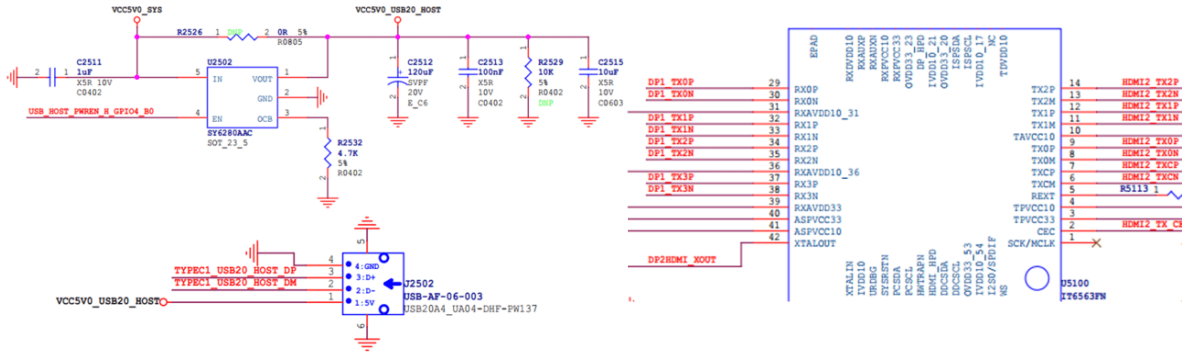

图 23 RK3588 Type-A USB2/DP to HDMI 接口

Type-A USB 3.1 硬件电路

该方案适用于 RK3588 Type-C0、Type-C1 和 USB 3.1 HOST2。其中，Type-C0/C1 对应的 Type-A USB 3.1 硬件电路设计，请参考Type-A USB 3.1/DP 硬件电路的 USB 3.1 电路设计部分即可。

本章节重点说明 USB 3.1 HOST2 对应的 Type-A USB 3.1 硬件电路设计。

在本开发指南的前面章节 RK3588 USB 控制器和 PHY 简介中，已经提到 USB 3.1 HOST_2 (也即 USB30_2) 控制器没有带 USB 2.0 PHY，所以 USB 3.1 HOST_2 需要与 USB 2.0 HOST0/1 组成完整的 USB 3.1 接口功能。

以 RK3588 EVB2 USB30_2 HOST 硬件电路设计为例。

1. USB 3.1 HOST2 与 USB 2.0 HOST1组成完整的 USB 3.1 接口；
2. 按照 RK3588 的芯片设计，USB 3.1 HOST2 同样可与 Type-C0/1 的 USB 2.0 组合，但考虑到 Type-C0/C1 还可能作为 USB Device 使用，且 USB 3.1 OTG 控制器的 Device 和 Host 功能无法同时使用[2]，所以，为了简化软硬件的设计，建议 USB 3.1 HOST2 只与 USB 2.0 HOST0/1 组合；
3. Type-A USB 3.1 对应的 DTS 配置请参考 Type-A USB 3.1 DTS 配置；

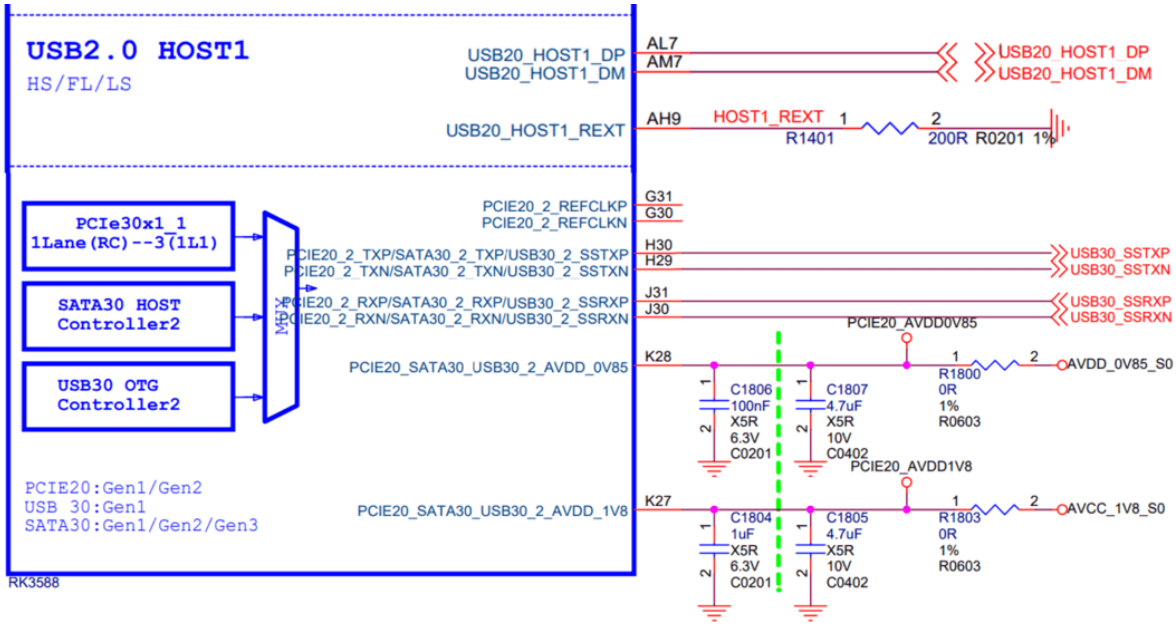

图 24 RK3588 USB 3.1 HOST2 电路

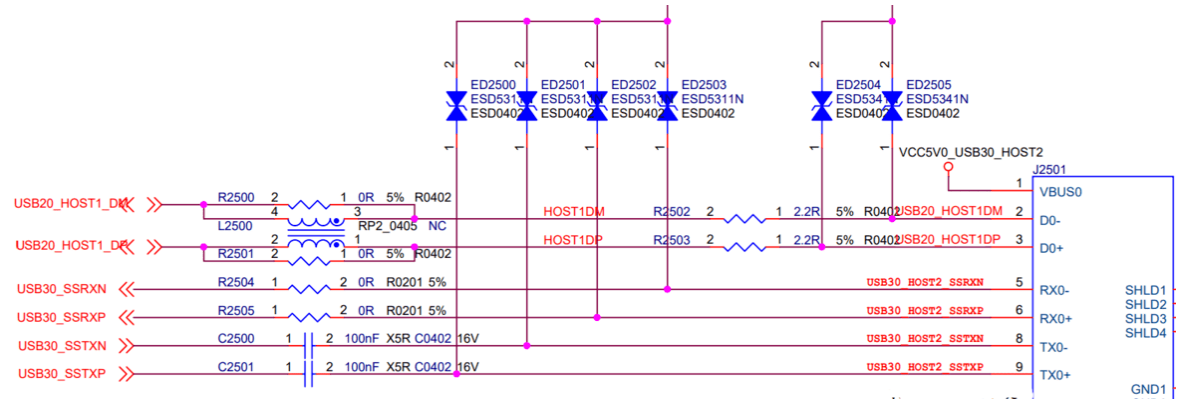

图 25 RK3588 USB 3.1 HOST2 接口

Type-A USB 2.0 硬件电路

该方案适用于 RK3588 USB 2.0 HOST0/1。

以 RK3588 EVB1 USB 2.0 HOST0/1 硬件电路设计为例。

1. USB 2.0 HOST0/1 共用 VBUS 供电电源（VCC5V0_USB20_HOST），由 GPIO 控制稳压芯片 SY6280AAC 输出 VBUS。最大输出电流由稳压芯片的 OCB pin 连接的电阻决定，最大电流 Ilim(A)=6800/Rset(ohm)，如下图 26 所示，VBUS 输出限流为 1.45A；
2. Type-A USB 2.0 对应的 DTS 配置请参考 Type-A USB 2.0 DTS 配置；

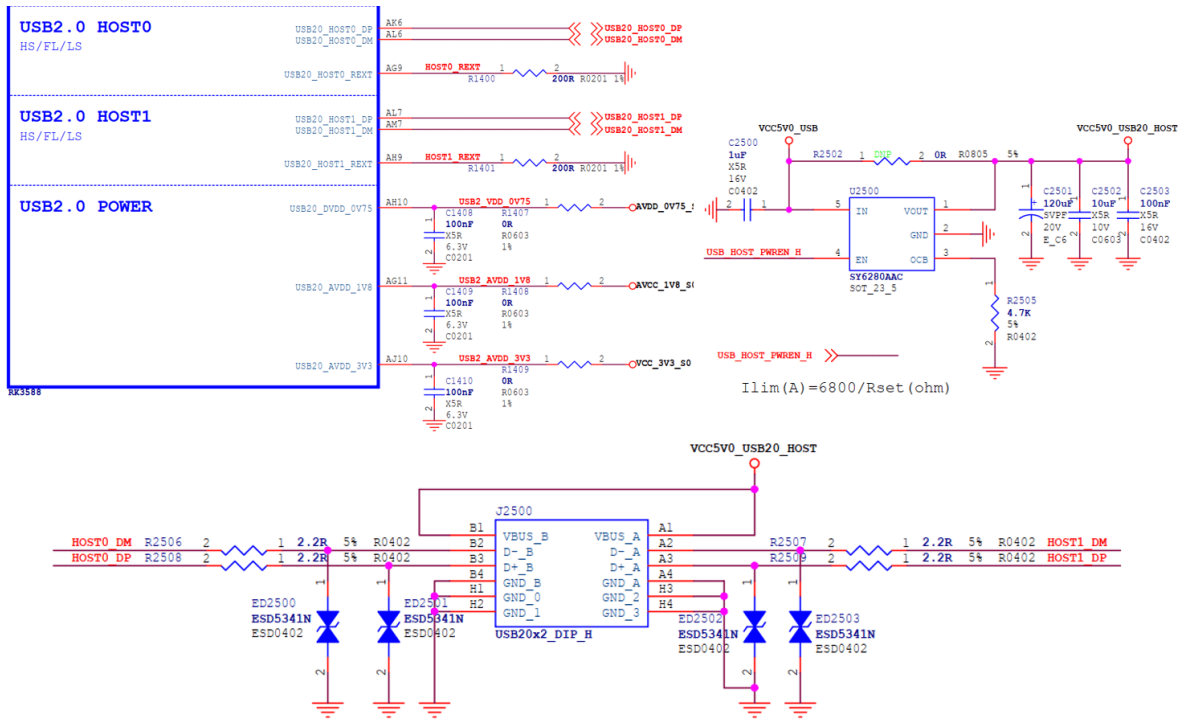

图 26 RK3588 Type-A USB 2.0 HOST 电路

RK3588 USB DTS 配置

RK3588 USB DTS 配置，包括：芯片级 USB 控制器/PHY DTSI 配置和板级 DTS 配置。

详细配置方法，请参考内核文档：

1. kernel/Documentation/devicetree/bindings/usb/snps,dwc3.yaml
2. kernel/Documentation/devicetree/bindings/usb/generic-ohci.yaml
3. kernel/Documentation/devicetree/bindings/usb/generic-ehci.yaml
4. kernel/Documentation/devicetree/bindings/connector/usb-connector.yaml
5. kernel/Documentation/devicetree/bindings/phy/phy-rockchip-inno-usb2.yaml
6. kernel/Documentation/devicetree/bindings/phy/phy-rockchip-usbdp.yaml
7. kernel/Documentation/devicetree/bindings/phy/phy/phy-rockchip-naneng-combphy.txt

USB 芯片级 DTSI 配置

RK3588 DTSI 文件中 USB 控制器和 PHY 相关的主要节点如下所示，因为 USB DTSI 节点配置的是 USB 控制器和 PHY 的公共资源和属性，建议开发者不要改动。

USB 3.1 OTG0、USB 2.0 HOST0/1、USB 3.1 HOST2 的 DTSI配置放在 rk3588s-evb.dtsi

USB 3.1 OTG1 的 DTSI 配置放在 rk3588-evb.dts

对应的 DTSI 完整路径如下：

`arch/arm64/boot/dts/rockchip/rk3588s.dtsi`

`arch/arm64/boot/dts/rockchip/rk3588.dtsi`

Note：

RK3588/RK3588S 的所有 USB 控制器和 PHY，在 rk3588s-evb.dtsi 和 rk3588-evb.dtsi 中，全部配置为 status = "okay"，如果产品的板级 DTS 文件有 include 这两个 EVB DTSI 文件，则只需要在板级 DTS 文件中，将不使用的 USB 节点配置为 "disabled" 即可。

USB 接口和 USB DTS 节点的对应关系如下表 16 所示。

表 16 RK3588 USB 接口和 USB DTS 节点的对应关系

USB 接口名称(原理图)	USB 控制器 DTS 节点	USB PHY DTS 节点
TYPEC0	usbdrd3_0 usbdrd_dwc3_0	u2phy0 u2phy0_otg usbdp_phy0 usbdp_phy0_u3
TYPEC1	usbdrd3_1 usbdrd_dwc3_1	u2phy1 u2phy1_otg usbdp_phy1 usbdp_phy1_u3
USB20_HOST0	usb_host0_ehci usb_host0_ohci	u2phy2 u2phy2_host
USB20_HOST1	usb_host1_ehci usb_host1_ohci	u2phy3 u2phy3_host
USB30_2	usbhost3_0 usbhost_dwc3_0	combphy2_psu

USB 控制器 DTSI 节点如下：

```
#USB3.1 OTG0 Controller
usbdrd3_0: usbdrd3_0 {
        compatible = "rockchip,rk3588-dwc3", "rockchip,rk3399-dwc3";
        ......
        usbdrd_dwc3_0: usb@fc000000 {
                compatible = "snps,dwc3";
                ......
        };
};

#USB2.0 HOST0 Controller
usb_host0_ehci: usb@fc800000 {
        compatible = "generic-ehci";
        ......
};

usb_host0_ohci: usb@fc840000 {
        compatible = "generic-ohci";
        ......
};

#USB2.0 HOST1 Controller
usb_host1_ehci: usb@fc880000 {
        compatible = "generic-ehci";
        ......
};

usb_host1_ohci: usb@fc8c0000 {
        compatible = "generic-ohci";
        ......
};
```

```
#USB3.1 HOST2 Controller
usbhost3_0: usbhost3_0 {
        compatible = "rockchip,rk3588-dwc3", "rockchip,rk3399-dwc3";
        ......
        usbhost_dwc3_0: usb@fcd00000 {
                compatible = "snps,dwc3";
                ......
        };
};

#USB3.1 OTG1 Controller
usbdrd3_1: usbdrd3_1 {
        compatible = "rockchip,rk3588-dwc3", "rockchip,rk3399-dwc3";
        ......
        usbdrd_dwc3_1: usb@fc400000 {
                compatible = "snps,dwc3";
                ......
        };
};
```

USB PHY DTSI 节点如下：

注意：USB PHY 和 USB 控制器具有一一对应的关系，需要成对配置。在芯片内部，USB PHY 和 控制器的连接关系，请参考 RK3588 USB 控制器和 PHY 简介的图 1 和 表 16。在DTSI 节点中，通过 USB 控制器节点的 "phys" 属性关联对应的 USB PHY。

```
#USB2.0 PHY0
usb2phy0_grf: syscon@fd5d0000 {
                compatible = "rockchip,rk3588-usb2phy-grf", "syscon",
                             "simple-mfd";
                ......
                u2phy0: usb2-phy@0 {
                        compatible = "rockchip,rk3588-usb2phy";
                        ......
                        u2phy0_otg: otg-port {
                                #phy-cells = <0>;
                                status = "disabled";
                        };
                };
        };

#USB2.0 PHY1
usb2phy1_grf: syscon@fd5d4000 {
    ......
};

#USB2.0 PHY2
usb2phy2_grf: syscon@fd5d8000 {
    ......
};

#USB2.0 PHY3
usb2phy3_grf: syscon@fd5dc000 {
    ......
};
```

```
#USB3.1/DP Combo PHY0
usbdp_phy0: phy@fed80000 {
        compatible = "rockchip,rk3588-usbdp-phy";
        ......
        usbdp_phy0_dp: dp-port {
                #phy-cells = <0>;
                status = "disabled";
        };

        usbdp_phy0_u3: u3-port {
                #phy-cells = <0>;
                status = "disabled";
        };
};

#USB3.1/DP Combo PHY1
usbdp_phy1: phy@fed90000 {
    ......
};

#USB3.1/SATA/PCIe PHY2
combphy2_psu: phy@fee20000 {
        compatible = "rockchip,rk3588-naneng-combphy";
        ......
};
```

Type-C USB 3.1/DP 全功能 DTS 配置

参考 `arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi` Type-C0 接口的 DTS 配置。

```
#USB2.0 PHY配置属性"rockchip,typec-vbus-det"，表示支持Type-C VBUS_DET常拉高的硬件设计
&u2phy0_otg {
        rockchip,typec-vbus-det;
};

#USB3.1/DP PHY0，需要根据硬件设计，配置属性"sbu1-dc-gpios"和"sbu2-dc-gpios"
&usbdp_phy0 {
        orientation-switch;
        svid = <0xff01>;
        sbu1-dc-gpios = <&gpio4 RK_PA6 GPIO_ACTIVE_HIGH>;
        sbu2-dc-gpios = <&gpio4 RK_PA7 GPIO_ACTIVE_HIGH>;

        port {
                #address-cells = <1>;
                #size-cells = <0>;
                usbdp_phy0_orientation_switch: endpoint@0 {
                        reg = <0>;
                        remote-endpoint = <&usbc0_orien_sw>;
                };

                usbdp_phy0_dp_altmode_mux: endpoint@1 {
                        reg = <1>;
                        remote-endpoint = <&dp_altmode_mux>;
                };
        };
};
```

```
#USB3.1 OTG0 Controller
&usbdrd_dwc3_0 {
        dr_mode = "otg";
        usb-role-switch;
        port {
                #address-cells = <1>;
                #size-cells = <0>;
                dwc3_0_role_switch: endpoint@0 {
                        reg = <0>;
                        remote-endpoint = <&usbc0_role_sw>;
                };
        };
};

#VBUS GPIO配置，在Type-C控制器芯片驱动中控制该GPIO
vbus5v0_typec: vbus5v0-typec {
        compatible = "regulator-fixed";
        regulator-name = "vbus5v0_typec";
        regulator-min-microvolt = <5000000>;
        regulator-max-microvolt = <5000000>;
        enable-active-high;
        gpio = <&gpio4 RK_PD0 GPIO_ACTIVE_HIGH>;
        vin-supply = <&vcc5v0_usb>;
        pinctrl-names = "default";
        pinctrl-0 = <&typec5v_pwren>;
};

#配置外置Type-C控制器芯片FUSB302
#需要根据实际的硬件设计，配置"I2C/interrupts/vbus-supply/usb_con"的属性
&i2c2 {
        status = "okay";
        usbc0: fusb302@22 {
                compatible = "fcs,fusb302";
                reg = <0x22>;
                interrupt-parent = <&gpio3>;
                interrupts = <RK_PB4 IRQ_TYPE_LEVEL_LOW>;
                pinctrl-names = "default";
                pinctrl-0 = <&usbc0_int>;
                vbus-supply = <&vbus5v0_typec>;
                status = "okay";

                ports {
                        #address-cells = <1>;
                        #size-cells = <0>;

                        port@0 {
                                reg = <0>;
                                usbc0_role_sw: endpoint@0 {
                                        remote-endpoint = <&dwc3_0_role_switch>;
                                };
                        };
                };

                usb_con: connector {
                        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, 1000, PDO_FIXED_USB_COMM)>;
                        source-pdos =
                                <PDO_FIXED(5000, 3000, PDO_FIXED_USB_COMM)>;

                        altmodes {
                                #address-cells = <1>;
                                #size-cells = <0>;

                                altmode@0 {
                                        reg = <0>;
                                        svid = <0xff01>;
                                        vdo = <0xffffffff>;
                                };
                        };

                        ports {
                            ......
                        };
                };
        };
```

Note:

如果使用 HUSB311 芯片替换 FUSB302 芯片，只需要基于 FUSB302 的 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
&i2c2 {
        usbc0: husb311@4e {
                compatible = "hynetek,husb311";
                reg = <0x4e>;
                ......
        };
};
```

Type-C to Type-A USB 3.1/DP DTS 配置

参考 `arch/arm64/boot/dts/rockchip/rk3588-evb2-lp4.dtsi` Type-C0 to Type-A USB 3.1/DP 的 DTS 配置。

```
#USB2.0 PHY0配置"phy-supply"属性，用于控制VBUS输出5V
#注意：使用phy-supply，无法实现VBUS的动态开关。如果OTG独占GPIO，不与其他HOST共用，并且OTG需要支持Device/HOST,则应该配置为"vbus-supply = <&vcc5v0_otg>",才能实现VBUS动态开关。
&u2phy0_otg {
        phy-supply = <&vcc5v0_host>;
};

#VBUS GPIO配置，在USB2.0 PHY驱动中控制该GPIO
vcc5v0_host: vcc5v0-host {
        compatible = "regulator-fixed";
        regulator-name = "vcc5v0_host";
        regulator-boot-on;
        regulator-always-on;
```

```
        regulator-min-microvolt = <5000000>;
        regulator-max-microvolt = <5000000>;
        enable-active-high;
        gpio = <&gpio4 RK_PA1 GPIO_ACTIVE_HIGH>;
        vin-supply = <&vcc5v0_usb>;
        pinctrl-names = "default";
        pinctrl-0 = <&vcc5v0_host_en>;
};

#USB3.1/DP PHY0，只需配置DP使用lane2/3,驱动会自动分配lane0/1给USB3.1 Rx/Tx
#如果硬件设计DP使用lane0/1，则此处应配置"rockchip,dp-lane-mux = <0 1>"
&usbdp_phy0 {
        rockchip,dp-lane-mux = <2 3>;
};

#USB3.1 OTG0 Controller
#配置"dr_mode"为"otg"，同时配置"extcon"属性，才能支持软件切换Device/Host mode
&usbdrd_dwc3_0 {
        dr_mode = "otg";
        extcon = <&u2phy0>;
        status = "okay";
};
```

Type-C to Type-A USB 2.0/DP DTS 配置

参考 `arch/arm64/boot/dts/rockchip/rk3588-nvr-demo.dtsi` Type-C1 to Type-A USB 2.0/DP 的 DTS 配置。

```
#USB2.0 PHY1配置"phy-supply"属性，用于控制VBUS输出5V
&u2phy1_otg {
        phy-supply = <&vcc5v0_host>;
        status = "okay";
};

#VBUS GPIO配置，在USB2.0 PHY驱动中控制该GPIO
vcc5v0_host: vcc5v0-host-regulator {
        compatible = "regulator-fixed";
        regulator-name = "vcc5v0_host";
        regulator-boot-on;
        regulator-always-on;
        regulator-min-microvolt = <5000000>;
        regulator-max-microvolt = <5000000>;
        enable-active-high;
        gpio = <&gpio4 RK_PB0 GPIO_ACTIVE_HIGH>;
        vin-supply = <&vcc5v0_sys>;
        pinctrl-names = "default";
        pinctrl-0 = <&vcc5v0_host_en>;
};

#USB3.1/DP PHY1，配置DP使用lane0/1/2/3
#需要根据实际的硬件设计，配置属性"rockchip,dp-lane-mux"
&usbdp_phy1 {
        rockchip,dp-lane-mux = < 0 1 2 3 >;
        status = "okay";
};

&usbdp_phy1_dp {
```

```
        status = "okay";
};

#配置属性"maximum-speed"，通知USBDP驱动将USB限制为USB2.0 only
&usbdp_phy1_u3 {
        maximum-speed = "high-speed";
        status = "okay";
};

#配置属性"maximum-speed"，通知DWC3驱动将USB限制为USB2.0 only
&usbdrd_dwc3_1 {
        dr_mode = "host";
        maximum-speed = "high-speed";
        status = "okay";
};
```

Type-C USB 2.0 only DTS 配置

配置1. 硬件电路带外置 Type-C 控制器芯片，支持 PD

参考 `arch/arm64/boot/dts/rockchip/rk3588s-tablet-rk806-single.dtsi` Type-C0 USB 2.0 OTG 的 DTS 配置

```
#USB2.0 PHY0注册typec orientation switch，用于与TCPM子系统交互，获取USB热拔插的信息
&u2phy0 {
        orientation-switch;
        status = "okay";

        port {
                #address-cells = <1>;
                #size-cells = <0>;
                u2phy0_orientation_switch: endpoint@0 {
                        reg = <0>;
                        remote-endpoint = <&usbc0_orien_sw>;
                };
        };
};

#USB2.0 PHY0 OTG配置
#配置属性"rockchip,sel-pipe-phystatus",表示选择GRF控制pipe phystatus，替代USBDP PHY的控制
#配置属性"rockchip,typec-vbus-det"，表示支持Type-C VBUS_DET常拉高的硬件设计
&u2phy0_otg {
        rockchip,sel-pipe-phystatus;
        rockchip,typec-vbus-det;
        status = "okay";
};

#disable USBDP PHY0的所有相关节点，让USBDP PHY0处于未初始化状态，达到最低功耗的目的
&usbdp_phy0 {
        status = "disabled";
};

&usbdp_phy0_dp {
        status = "disabled";
};
```

```
&usbdp_phy0_u3 {
        status = "disabled";
};

&dp0 {
        status = "disabled";
};

#配置USB3.1 OTG0 Controller
#配置"phys = <&u2phy0_otg>",即不引用USBDP PHY
#配置maximum-speed = "high-speed"，通知DWC3驱动将USB限制为USB2.0 only
&usbdrd3_0 {
        status = "okay";
}

&usbdrd_dwc3_0 {
        dr_mode = "otg";
        status = "okay";

        maximum-speed = "high-speed";
        phys = <&u2phy0_otg>;
        phy-names = "usb2-phy";
        usb-role-switch;
        port {
                #address-cells = <1>;
                #size-cells = <0>;
                dwc3_0_role_switch: endpoint@0 {
                        reg = <0>;
                        remote-endpoint = <&usbc0_role_sw>;
                };
        };
};

#配置外置Type-C控制器芯片FUSB302
#需要根据实际的硬件设计，配置"I2C/interrupts/vbus-supply/usb_con"的属性
#需要配置usbc0_orien_sw的属性remote-endpoint = <&u2phy0_orientation_switch>
&i2c8 {
        status = "okay";
        pinctrl-names = "default";
        pinctrl-0 = <&i2c8m2_xfer>;

        usbc0: fusb302@22 {
                compatible = "fcs,fusb302";
                reg = <0x22>;
                interrupt-parent = <&gpio0>;
                interrupts = <RK_PC4 IRQ_TYPE_LEVEL_LOW>;
                pinctrl-names = "default";
                pinctrl-0 = <&usbc0_int>;
                vbus-supply = <&vbus5v0_typec>;
                status = "okay";

                ports {
                        #address-cells = <1>;
                        #size-cells = <0>;

                        port@0 {
                                reg = <0>;
                                usbc0_role_sw: endpoint@0 {
```

```
                                        remote-endpoint = <&dwc3_0_role_switch>;
                                };
                        };
                };

                usb_con: connector {
                        compatible = "usb-c-connector";
                        label = "USB-C";
                        data-role = "dual";
                        power-role = "dual";
                        ......
                        ports {
                                #address-cells = <1>;
                                #size-cells = <0>;

                                port@0 {
                                        reg = <0>;
                                        usbc0_orien_sw: endpoint {
                                                remote-endpoint =
<&u2phy0_orientation_switch>;
                                        };
                                };
                        };

                };
        };
```

配置2. 硬件电路不带外置 Type-C 控制器芯片，支持 Device only

参考 `arch/arm64/boot/dts/rockchip/rk3588-evb5-lp4.dtsi` Type-C0 USB 2.0 Device 的 DTS 配置

```
#USB2.0 PHY0 OTG配置
#配置属性"rockchip,sel-pipe-phystatus",表示选择GRF控制pipe phystatus，替代USBDP PHY的
控制
&u2phy0_otg {
        rockchip,sel-pipe-phystatus;
        status = "okay";
};

#disable USBDP PHY0的所有相关节点，让USBDP PHY0处于未初始化状态，达到最低功耗的目的
&usbdp_phy0 {
        status = "disabled";
};

&usbdp_phy0_dp {
        status = "disabled";
};

&usbdp_phy0_u3 {
        status = "disabled";
}

#配置USB3.1 OTG0 Controller
#配置dr_mode = "peripheral"，通知DWC3驱动初始化为Device only mode
#配置"phys = <&u2phy0_otg>",即不引用USBDP PHY
#配置maximum-speed = "high-speed"，通知DWC3驱动将USB限制为USB2.0 only
```

```
&usbdrd_dwc3_0 {
        dr_mode = "peripheral";
        phys = <&u2phy0_otg>;
        phy-names = "usb2-phy";
        maximum-speed = "high-speed";
};
```

配置3. 硬件电路不带外置 Type-C 控制器芯片，支持 OTG（**需要增加 CC to ID 电平转换电路**）

```
#USB2.0 PHY0 OTG配置
#配置属性"rockchip,sel-pipe-phystatus",表示选择GRF控制pipe phystatus，替代USBDP PHY的控制
&u2phy0_otg {
        rockchip,sel-pipe-phystatus;
        status = "okay";
};

#disable USBDP PHY0的所有相关节点，让USBDP PHY0处于未初始化状态，达到最低功耗的目的
&usbdp_phy0 {
        status = "disabled";
};

&usbdp_phy0_dp {
        status = "disabled";
};

&usbdp_phy0_u3 {
        status = "disabled";
}

#配置USB3.1 OTG0 Controller
#配置dr_mode = "otg"
#配置"phys = <&u2phy0_otg>",即不引用USBDP PHY
#配置maximum-speed = "high-speed"，通知DWC3驱动将USB限制为USB2.0 only
#配置"extcon"属性，才能支持自动切换Device/Host mode
&usbdrd_dwc3_0 {
        dr_mode = "otg";
        phys = <&u2phy0_otg>;
        phy-names = "usb2-phy";
        maximum-speed = "high-speed";
        extcon = <&u2phy0>;
};
```

Type-A USB 3.1 DTS 配置

参考 `arch/arm64/boot/dts/rockchip/rk3588-evb2-lp4.dtsi` USB30_2 HOST 的 DTS 配置

```
#USB2.0 PHY3配置"phy-supply"属性，用于控制VBUS输出5V
&u2phy3_host {
        phy-supply = <&vcc5v0_host>;
}

#VBUS GPIO配置，在USB2.0 PHY驱动中控制该GPIO
vcc5v0_host: vcc5v0-host {
        compatible = "regulator-fixed";
        regulator-name = "vcc5v0_host";
```

```
        regulator-boot-on;
        regulator-always-on;
        regulator-min-microvolt = <5000000>;
        regulator-max-microvolt = <5000000>;
        enable-active-high;
        gpio = <&gpio4 RK_PA1 GPIO_ACTIVE_HIGH>;
        vin-supply = <&vcc5v0_usb>;
        pinctrl-names = "default";
        pinctrl-0 = <&vcc5v0_host_en>;
};

#使能USB3.1/SATA/PCIe Combo PHY
&combphy2_psu {
        status = "okay";
};

#配置USB3.1 HOST2 Controller
&usbhost3_0 {
        status = "okay";
};

&usbhost_dwc3_0 {
        dr_mode = "host";
        status = "okay";
};
```

Type-A USB 2.0 DTS 配置

参考 `arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi` USB 2.0 HOST0/1 的 DTS 配置。

```
#USB2.0 PHY2/3配置"phy-supply"属性，用于控制VBUS输出5V
&u2phy2_host {
        phy-supply = <&vcc5v0_host>;
};

&u2phy3_host {
        phy-supply = <&vcc5v0_host>;
};

#VBUS GPIO配置，在USB2.0 PHY驱动中控制该GPIO
vcc5v0_host: vcc5v0-host {
        compatible = "regulator-fixed";
        regulator-name = "vcc5v0_host";
        regulator-boot-on;
        regulator-always-on;
        regulator-min-microvolt = <5000000>;
        regulator-max-microvolt = <5000000>;
        enable-active-high;
        gpio = <&gpio4 RK_PB0 GPIO_ACTIVE_HIGH>;
        vin-supply = <&vcc5v0_usb>;
        pinctrl-names = "default";
        pinctrl-0 = <&vcc5v0_host_en>;
};

#USB2.0 HOST0/1 Controller
&usb_host0_ehci {
        status = "okay";
```

```
};

&usb_host0_ohci {
        status = "okay";
};

&usb_host1_ehci {
        status = "okay";
};

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

Micro USB DTS 配置

RK3588 OTG 可以支持 Micro USB 2.0 的接口设计，以 arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi 文件为例。

在 DT 配置中要删除如下节点及属性：

1. usbc0 节点及其子节点；
2. usbdp_phy0 节点中 orientation-switch 属性和 port 子节点；
3. usbdrd_dwc3_0 节点中 usb-role-switch 属性和 port 子节点；
4. u2phy0_otg 节点中 rockchip,typec-vbus-det 属性；
5. pinctrl 节点中 usb-typec 子节点。

在 DT 配置中要新增如下节点及属性：

1. vcc5v0_otg 节点用于控制 vbus 供给；
2. u2phy0_otg 节点添加 vbus-supply 属性；
3. usbdrd_dwc3_0 节点中添加 extcon = <&u2phy0>; 属性。

```
[...]
# VBUS GPIO 配置，在USB2.0 PHY驱动中控制该 GPIO
vcc5v0_otg: vcc5v0-otg {
    compatible = "regulator-fixed";
    regulator-name = "vcc5v0_otg";
    regulator-min-microvolt = <5000000>;
    regulator-max-microvolt = <5000000>;
    enable-active-high;
    gpio = <&gpio4 RK_PA7 GPIO_ACTIVE_HIGH>;
    vin-supply = <&vcc5v0_sys>;
    pinctrl-names = "default";
    pinctrl-0 = <&vcc5v0_otg_en>;
};

&pinctrl {
    usb {
        [...]
        vcc5v0_otg_en: vcc5v0-otg-en {
            rockchip,pins = <4 RK_PA7 RK_FUNC_GPIO &pcfg_pull_up>;
        };
    };
};

[...]
```

```
# 按实际设计配置 status. 如果 USB3 和 DP 都不使用，建议关闭 USBDP PHY0 的所有相关节点，以降低功耗
&usbdp_phy0 {
    status = "okay";
};

&usbdp_phy0_dp {
    status = "okay";
};

&usbdp_phy0_u3 {
    maximum-speed = "high-speed";
    status = "okay";
};

&u2phy0 {
    status = "okay";
};

&u2phy0_otg {
    vbus-supply = <&vcc5v0_otg>;
    rockchip,sel-pipe-phystatus;
    status = "okay";
};

&usbdrd_dwc3_0 {
    status = "okay";
    dr_mode = "otg";
    maximum-speed = "high-speed";
    extcon = <&u2phy0>;
};
```

Linux USB DT 配置的注意点

USB DT 重要属性说明

USB 控制器

1. "usb-role-switch" 仅用于标准 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" ；
2. "extcon" 属性主要功能之一是动态切换 OTG mode，使用 USB Micro 接口，同时控制器配置为 “otg” 模式的方案中，需要增加 "extcon" ，实现 OTG 模式切换。

USB2 PHY

1. "vbus-supply" 和 "phy-supply"，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,sel-pipe-phystatus"，该属性用于配置 GRF USB 控制寄存器，选择 GRF 控制 pipe phystatus，替代 USBDP PHY 的控制。主要用于 USB 2.O only 的方案，如果 USBDP PHY 没有使能，必须增加该属性，否则，USB device 无法正常工作；
3. "rockchip,typec-vbus-det"，用于支持 Type-C VBUS_DET 常拉高的硬件设计；

USBDP Combo PHY

1. "rockchip,dp-lane-mux" ，非全功能 Type-C 方案中，配置 DP 映射的 Lane number。DP 支持 2 条或 4 条 lane，如 "rockchip,dp-lane-mux = <2, 3>;" 表示 DP Lane0 mapping 至 USBDP PHY 的 Lane2，DP Lane1 mapping 至 USBDP PHY 的 Lane3；同理，"rockchip,dp-lane-mux = <0, 1, 2, 3>;" 表示 DP Lane0 mapping 至 USBDP PHY 的 Lane0 等等，依次类推。

RK3588 USB OTG mode 切换命令

RK3588 SDK 支持通过软件方法，强制设置 USB 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。

RK3588 Linux-5.10 内核切换 USB OTG 控制器工作在 Peripheral mode 或 Host mode，有如下两种方式。

注意：方式 1 依赖于 USB DTS 的正确配置，只能用于非 Type-C 接口的硬件电路设计，方式 2 没有限制。因此，在不确定软硬件是否正确适配时，推荐优先使用方式 2。

方式1. [Legacy]

```
#1.Force host mode
  echo host > /sys/devices/platform/fd5d0000.syscon/fd5d0000.syscon:usb2-
phy@0/otg_mode
#2.Force peripheral mode
  echo peripheral > /sys/devices/platform/fd5d0000.syscon/fd5d0000.syscon:usb2-
phy@0/otg_mode
```

方式2. [New]

```
#1.Force host mode
  echo host > /sys/kernel/debug/usb/fc000000.usb/mode
#2.Force peripheral mode
  echo device > /sys/kernel/debug/usb/fc000000.usb/mode
```

Type-C 控制器芯片支持列表

表 17 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 Type-C USB 3.1/DP 全功能 DTS 配置 中 usbc0 节点，只需要修改该节点名字、reg 地址和 compatible 属性即可。

note3.

- FUSB302 可直接替换为 HUSB311。内核需要单独使用 CONFIG_TYPEC_HUSB311，DTS 配置注意点同上述 note2。

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 配置注意点同上述 note2。

参考文档

1. 《Universal Serial Bus Type-C Cable and Connector Specification》
2. 《Rockchip_Developer_Guide_USB_CN》