Rockchip RK3399 USB 开发指南

文件标识：RK-SM-YF-101

发布版本：V2.0.0

日期：2022-09-20

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 RK3399 USB 模块在不同 Linux 内核版本的 DTS 配置方法和使用方法。RK3399 支持两个 Type-C 接口（支持 USB 3.1 Gen1 和 DisplayPort）和两个 USB 2.0 Host 接口，并且，两个 Type-C 接口分别采用独立的 USB 3.1 Gen1 OTG 控制器，都可以支持 OTG 功能（Peripheral 和 Host）。同时，Type-C USB 3.1 Gen1 接口向下兼容 USB2.0/1.1/1.0。在实际的产品设计中，Type-C 接口可以根据应用需求，将物理接口简化设计为 Type-A USB 3.0/2.0 + DP 接口、Micro USB 3.0/2.0 等多种接口类型，内核 USB 驱动已经兼容常见的各种 USB 接口类型的硬件设计，开发者只需要根据实际的硬件电路设计正确配置 Linux 内核 USB DTS，就可以支持相应的 USB 接口。

产品版本

芯片名称	内核版本
RK3399	Linux-4.4 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：

软件工程师

技术支持工程师

修订记录

日期	版本	作者	修改说明
2018-03-01	V1.0	吴良峰	初始版本
2019-01-09	V1.1	吴良峰	使用 markdownlint 修订格式
2019-06-25	V1.2	吴良峰	1. 增加 Type-C to Type-A USB 2.0 说明 2. 增加 VBUS 供电说明 3. 更新文档目录名称 4. 参考示例由 EVB 改为 Sapphire Excavator Board 5. 修订一些错误
2019-12-20	V1.3	吴良峰	1. 增加 Type-C to Type-A USB 3.0 OTG DTS 的配置说明 2. 增加章节 《OTG 切换命令》 3. 增加章节 《Linux-4.4 与 4.19 USB 3.0 DTS配置的差异点》 4. 修订一些排版格式
2020-02-19	V1.3.1	吴良峰	增加免责声明，商标声明以及版权声明
2021-01-15	V1.3.2	吴良峰	1. 修订文档名称 2. 修订超链接问题 3. 修订一些格式错误
2022-09-20	V2.0.0	吴良峰 王明成	1. 重构文档，提高可读性 2. 增加 RK3399 USB 控制器和 PHY 简介 3. 增加 RK3399 Type-C USB 使用注意点 4. 增加 Linux-5.10 内核 USB DTS 配置说明 5. 增加 USB DTS 配置的注意点 6. 增加 Type-C 控制器芯片支持列表

目录

RK3399 USB 控制器和 PHY 简介

RK3599 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 2.0 HOST HSIC 控制器。

表 1-1 RK3399 USB 控制器列表

芯片/控制器	USB 2.0 HOST (EHCI&OHCI)	USB 3.1 OTG (DWC3&xHCI)	USB 2.0 Host HSIC(EHCI)
RK3399	2	2	1

RK3399 支持 5 个 USB PHY，包括：2 个 USB 2.0 Combo PHY，2 个 USB 3.1/DP Combo PHY，1 个 HSIC PHY。

表 1-2 RK3399 USB PHY 支持列表

芯片/PHY	USB 2.0 Combo PHY	USB3.1/DP Combo PHY	HSIC PHY
RK3399	2 [2 × ports]	2	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，[2 × ports] 表示一个 PHY 只支持 2 个 USB port，即对应 2 个 USB 接口；
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；

RK3399 USB 控制器和 PHY 的内部连接关系（未包含 HSIC），以及对应的常见 USB 物理接口如下图 1-1 所示。

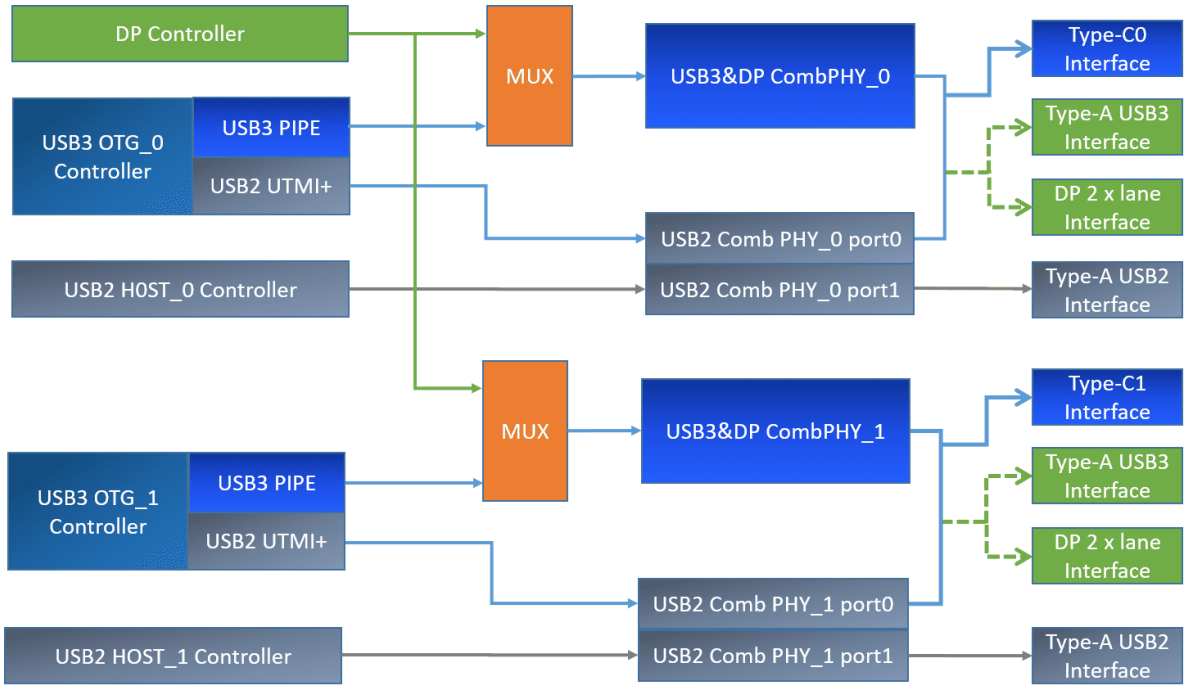

图 1-1 RK3399 USB 控制器和 PHY 的连接示意图

Note:

1. RK3399 最多可以同时支持 2 个 Type-C 接口，2 个 Type-A USB 2.0 接口，1 个 USB 2.0 HSIC 接口；
2. RK3399 只支持 1 个 DP 控制器，SDK 默认设计 DP 控制器连接到 Type-C0，实际产品应用时，可由软件选择切换到 Type-C1，但只能二选一。
3. 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]）；
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，这两部分 PHY 在芯片内部的硬件模块是独立的，供电也是独立的。其中，USB 3.1/DP Combo PHY0 固定与 USB2 Combo PHY0 的 port0 组合，USB 3.1/DP Combo PHY1 固定与 USB2 Combo PHY1 的 port0 组合，这种组合关系在 Linux USB DTS 中会体现出来；
5. 需要注意的是，RK3399 USB 支持的接口类型并不局限于图 1-1 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RK3399 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 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 RK3399 USB DTS 配置；

RK3399 USB 支持的接口类型

RK3399 USB 可以支持如下图 2-1 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。

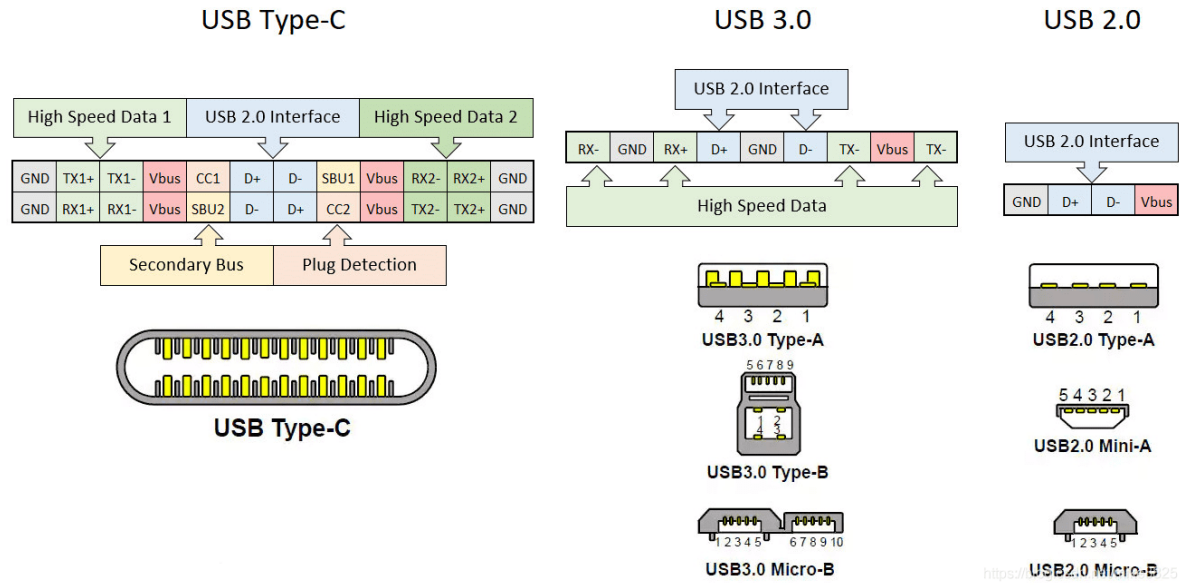

图 2-1 RK3399 可支持的 USB 接口类型

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，如下图 2-2 所示，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

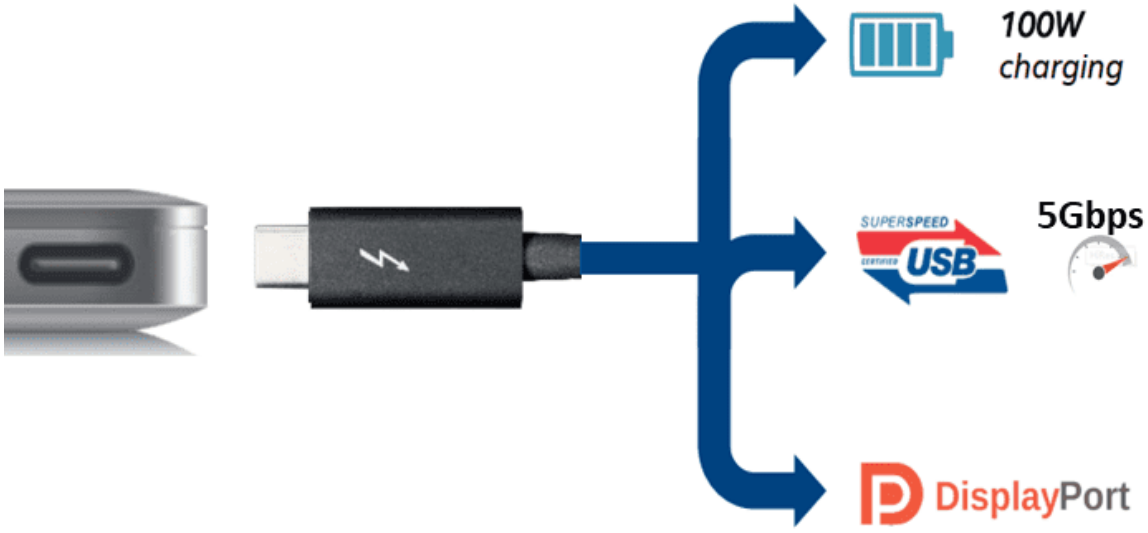

图 2-2 Type-C USB 3.1/DP 接口

Type-C 接口的公头和母头的引脚完整定义如下图 2-3 所示。

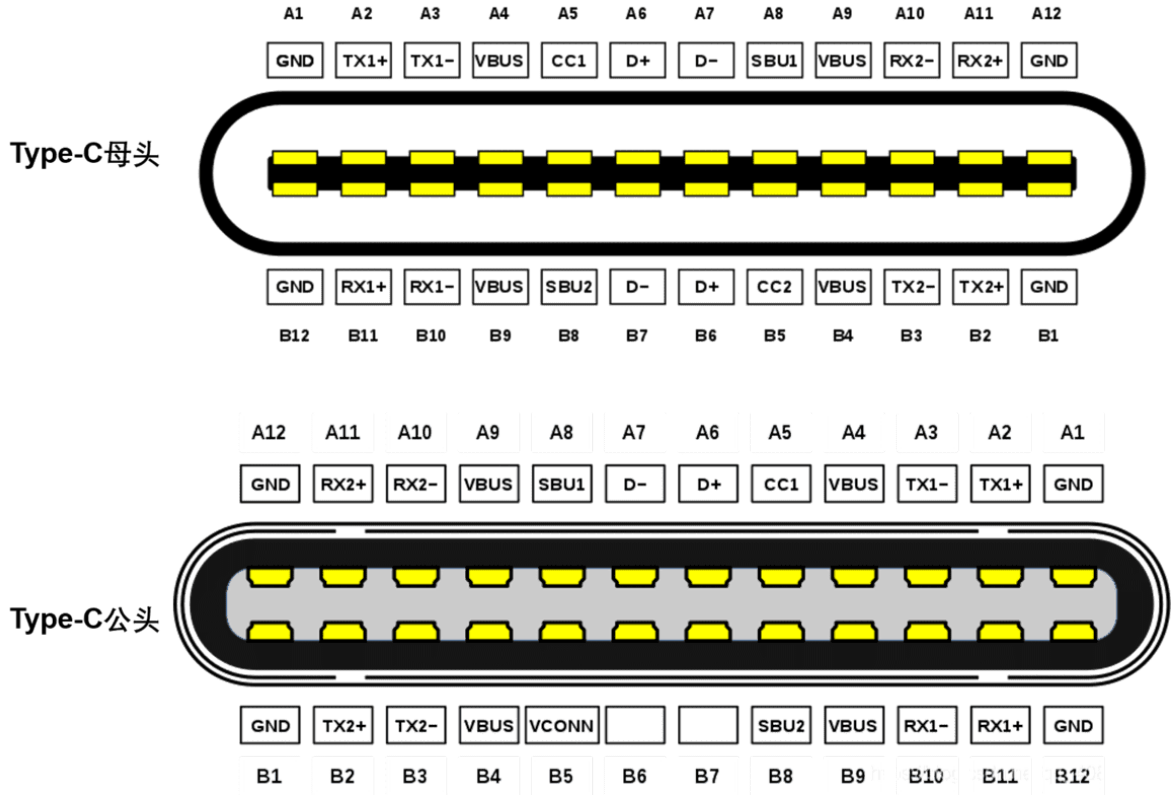

图 2-3 Type-C 接口的引脚定义

表 2-1 Type-C 接口的引脚描述

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 2-2 RK3399 Type-C0 与 Type-C 接口的连接关系

RK3399 Type-C0 Pin	Type-C 接口 Pin	关系描述
TYPEC0_TX1P/TX1M	A2/A3	可用于 USB 3.1 Tx1 或者 DP Tx lane0
TYPEC0_RX1P/RX1M	B11/B10	可用于 USB 3.1 Rx1 或者 DP Tx lane1
TYPEC0_RX2P/RX2M	A11/A10	可用于 USB 3.1 Rx2 或者 DP Tx lane2
TYPEC0_TX2P/TX2M	B2/B3	可用于 USB 3.1 Tx2 或者 DP Tx lane3
TYPEC0_DP/DM	A6/A7, B6/B7	连接到 RK3399 TYPEC0_OTG_DP/DM，其中，A6 和 B6并联，A7 和 B7 并联
TYPEC0_SBU1/SBU2	A8/B8	连接到 RK3399 USBDP PHY 的 AUX，只用于 DP Alternate Mode
TYPEC0_SBU1_DC/SBU2_DC	A8/B8	用于控制 DP AUX 传输时的上拉（注意：不需要连接到 GPIO）
TYPEC0_CC1/CC2	A5/B5	连接到外置 Type-C 控制器芯片 (HUSB311/FUSB302)，不需要连接到 R3399 SoC 的 CC1/CC2
TYPEC0_U2VBUSDET	A4/A9/B4/B9	Type-C 接口的 VBUS 通过电压转换电路，将 VBUS 降压为 3V，再连接的 RK3399 SoC 的 VBUSDET

Note：

1. RK3399 芯片的 CC1/CC2 不使用，硬件设计时，默认悬空即可；
2. RK3399 芯片内部 USBDP PHY 可以支持 TYPEC0_SBU1_DC/SBU2_DC 的上下拉控制，不需要额外占用 GPIO，这点与 RK3588 不同；
3. RK3399 芯片内部只支持 1 个 DP 控制器，Type-C0 和 Type-C1 不能同时支持 DP 功能，这点与 RK3588 不同（RK3588 支持 2 个 独立的 DP 控制器）；
4. RK3399 Type-C 软件驱动未支持 TYPEC0_U2VBUSDET 常供电的方案，因此，要求硬件电路设计时，需要将 TYPEC0_U2VBUSDET 通过电压转换电路连接到 Type-C 接口的 VBUS，以支持 USB Deviec mode 热拔插和 BC 1.2 充电检测功能，这点与 RK3588 不同；
5. RK3399 与 RK3588 均支持 Type-C USB&DP 全功能接口，但它们的 USBDP PHY lane 的复用关系有所区别，如下表 2-3 所示。

表 2-3 RK3399/RK3588 USBDP PHY lane 复用关系

USBDP PHY lane	RK3399 USB	RK3399 DP	RK3588 USB	RK3588 DP
lane0	USB3 Tx1	DP TX0	USB3 Rx1	DP TX0
lane1	USB3 Rx1	DP TX1	USB3 Tx1	DP TX1
lane2	USB3 Rx2	DP TX2	USB3 Rx2	DP TX2
lane3	USB3 Tx2	DP TX3	USB3 Tx2	DP TX3

Type-C to Type-A USB&DP 接口

RK3399 Type-C0 可以拆分为独立的 Type-A USB 3.1 接口和 DP 接口使用，例如：

1. Type-A USB 3.1 (使用 USBDP PHY0 的 lane0/1) + DP 1.3 (使用 USBDP PHY0 的 lane2/3);
2. Type-A USB 3.1 (使用 USBDP PHY0 的 lane0/1) + DP to VGA (使用 USBDP PHY0 的 lane2/3);

RK3588 Type-C0 还可以拆分为独立的 Type-A USB 2.0 接口和 DP（4 x lanes）接口使用，例如：

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

Note:

1. Rockchip 发布的 SDK Linux-4.4/4.19 内核 Type-C 驱动默认不支持 USB 和 DP 拆分使用的方式，软件修改请参考Type-C to Type-A USB&DP DTS 配置;
2. 考虑到 RK3399 只支持 1 个 DP 控制器，Type-C0 和 Type-C1 不能同时支持 DP 功能。所以硬件设计时，优先考虑 Type-C0 支持 USB&DP 功能，Type-C1 支持 USB only 的组合方式，这样做，可以减少软件开发的工作量。
3. 理论上 RK3399 还可以支持 Type-A USB 3.1 使用 lane2/lane3 + DP 使用 lane0/lane1 的组合方式，但需要软件开发者自行修改内核 DTS 配置，会增加开发工作量。因此，如果没有特殊的硬件设计考虑，建议优先采用上面提供的组合 1 和 组合 2 两种常规方式。

Type-C USB 2.0 only 接口

RK3399 可以简化为 Type-C USB 2.0 only 接口。如下图 2-4 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。

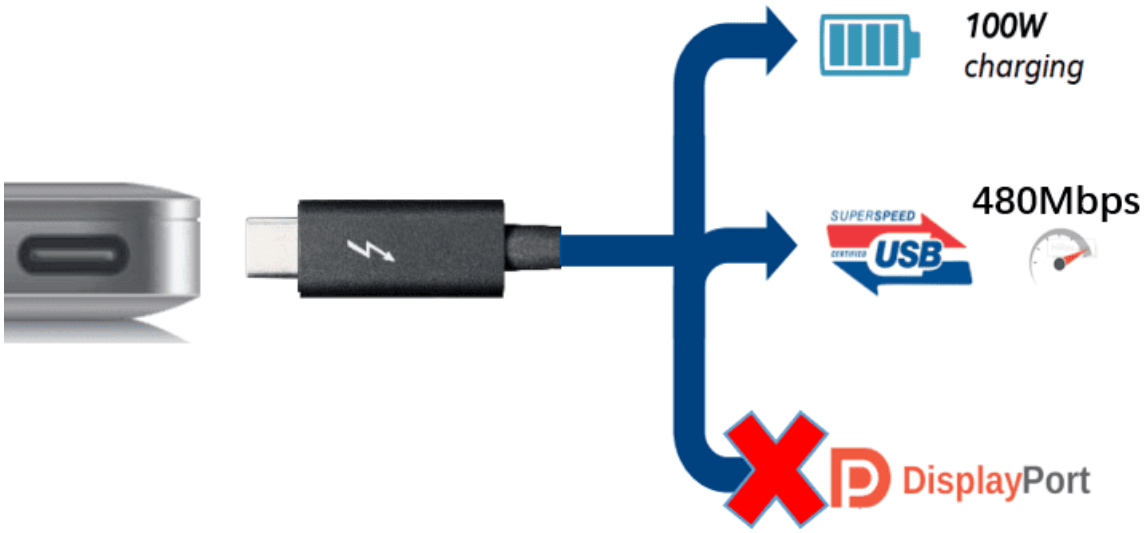

图 2-4 Type-C USB 2.0 only 接口

Type-A 接口类型

Type-A USB 3.1 接口

RK3399 最多可以支持 2 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

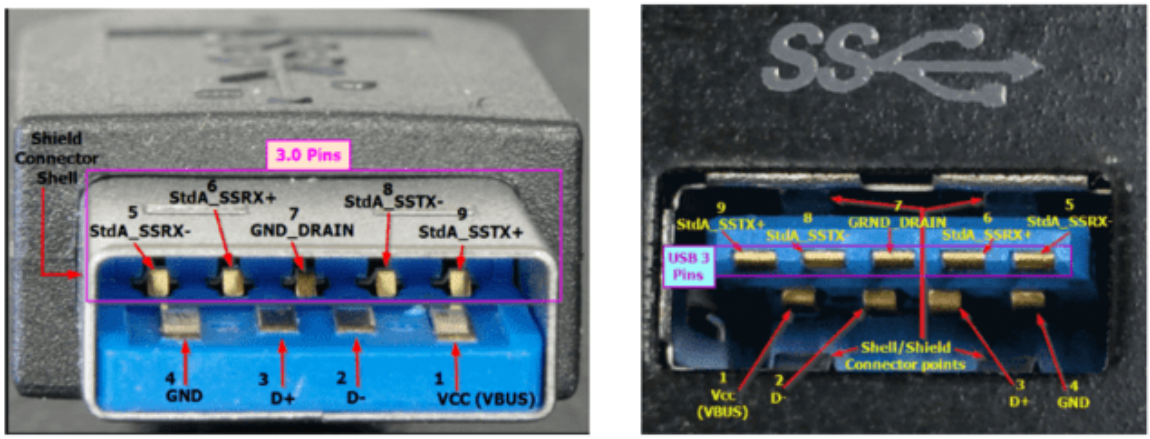

图 2-5 Type-A USB 3.1 接口

Type-A USB 2.0 接口

RK3399 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

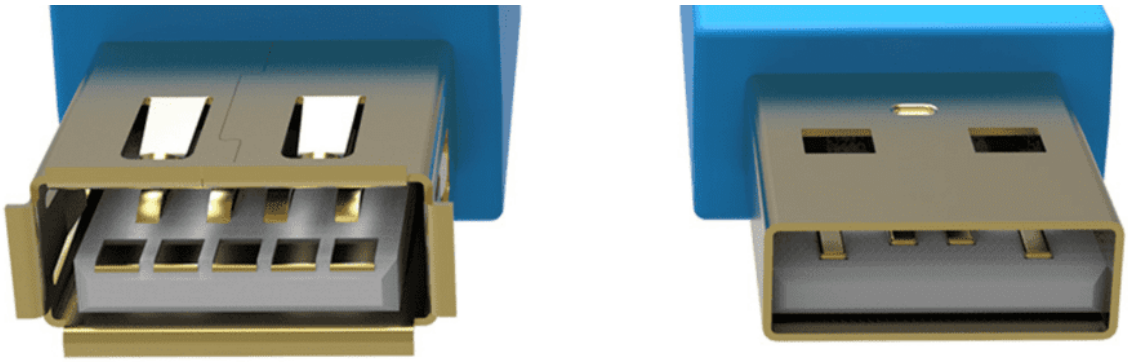

图 2-6 Type-A USB 2.0 接口

Micro 接口类型

Micro USB 3.1 接口

RK3399 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 软件自动切换 OTG Device/HOST mode。OTG ID 脚需要连接到 RK3399 TYPEC0_ID pin 脚（Type_C1 对应 TYPEC1_ID ），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。

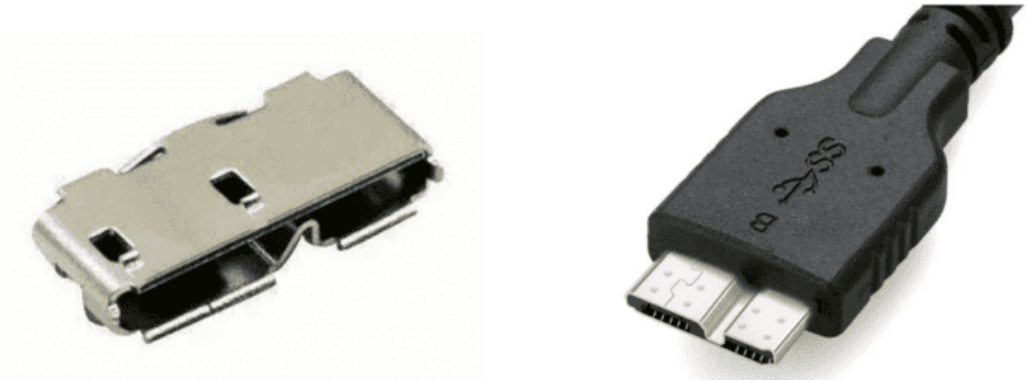

图 2-7 Micro USB 3.1 接口

Micro USB 2.0 接口

RK3399 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 RK3399 TYPEC0_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 2-8 Micro USB 2.0 接口

RK3399 Type-C USB 使用注意点

1. RK3399 Type-C USB 3.1 拆分为 USB 2.0 port 和 USB 3.1 port 的使用注意点

   RK3399 USB 3.1 控制器的特点是：

   (1) Device 和 Host 功能不支持同时使用；

   (2) Host mode 的 USB 2.0 Host Port 和 USB 3.1 Host Port 可以独立且同时使用；

   (3) Device mode 不支持 USB 2.0 Device 和 USB 3.1 Device 同时工作。

   基于 USB 3.1 控制器的上述设计特点，在产品应用上，如果明确 USB 3.1 拆分后，只有 Host 应用需求，没有 Device 应用需求，才适合拆分使用。否则，建议不要拆分 USB 3.1 的 USB 2.0 port 和 USB 3.1 port，避免 Device 功能使用限制。

2. RK3399 Type-C USB 3.1 & DP 拆分为独立接口的使用注意点

   RK3399 Type-C 可以拆分为独立的 Type-A USB 3.1 接口和 DP 接口使用，主要应用场景包括：USB 3.1 + DP 2 x lanes 和 USB 2.0 + DP 4 x lanes 两种组合方式。Rockchip 平台发布的 SDK 软件方案，未完善支持USB & DP 拆分使用的方式，具体使用注意点请参考章节Type-C to Type-A USB&DP DTS 配置。

3. RK3399 TYPEC0_U2VBUSDET 硬件设计注意点

   TYPEC0_U2VBUSDET 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：

(1) 为了支持 RK3399 Maskrom USB 下载固件的功能，必须保证系统上电时，TYPEC0_U2VBUSDET 默认外部拉高，不能悬空；

(2) RK3399 芯片输入端 TYPEC0_U2VBUSDET 的高电平范围在 **[1.4V ~ 3.3V]**；

(3) 软件上（包括：Linux-4.4/4.19/5.10）尚未支持 TYPEC0_U2VBUSDET 常供电（固定上拉）的 Type-C 接口方案。 因此，要求硬件设计时，需要通过电压转换电路将 TYPEC0_U2VBUSDET 连接到 Type-C 接口的 VBUS 引脚。如果 TYPEC0_U2VBUSDET 常供电，会导致 USB Device 热拔插功能异常以及 BC 1.2 充电检测异常。

4. RK3399 双 Type-C 接口的使用注意点

   RK3399 Type-C0/1 USB 控制器硬件都支持 USB 3.1 OTG 功能，并且向下兼容 USB 2.0/1.1/1.0。但由于内核的 USB Gadget 框架限制，只支持一个 USB 控制器作为 Device 功能，所以 SDK 默认配置 Type-C0 支持 OTG mode，而 Type-C1 仅支持 Host mode。

   在实际应用上：

   (1) 下载固件只支持使用 Type-C0 接口，无法更改；

   (2) ADB 默认使用 Type-C0，如果要更改为使用 Type-C1，请参考章节Type-C1 支持 OTG mode 的修改方法；

   (3) 理论上，可以支持双 Type-C USB OTG 的功能。但 Rockchip 平台尚未支持 OTG0/1 动态切换 Device mode 的软件解决方案。因此，建议产品上固定使用 Type-C0 或者 Type-C1 作 Device mode。如果产品上，一定要支持双 Type-C OTG 的方案，需要开发者实现该功能，大致思路：应用层实时获取内核 Type-C0/C1 的 USB mode，再根据 USB mode 来动态配置内核节点：/config/usb_gadget/g1/UDC （作用是绑定 USB Gadget 驱动和 对应的 Type-C USB 控制器驱动）。

RK3399 USB DTS 配置

RK3399 USB DTS 配置，包括：芯片级 USB 控制器/PHY DTSI 配置和板级 DTS 配置。

其中，芯片级配置，主要负责配置 USB 控制器和 PHY 的公共资源和属性，包括：reg、clock、reset、grf、interrupts 以及控制器特有的属性等。板级配置，主要配置 VBUS 控制、Type-C 控制器芯片（如：FUSB302）、以及与硬件电路相关的特殊属性。

USB 芯片级 DTSI 配置

RK3399 DTSI 文件中 USB 控制器和 PHY 相关的主要节点如下表所示，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。

表 4-1 RK3399 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_otg tcphy0_usb3
TYPEC1	usbdrd3_1 usbdrd_dwc3_1	u2phy1_otg tcphy1_usb3
USB20_HOST0	usb_host0_ehci usb_host0_ohci	u2phy0_host
USB20_HOST1	usb_host1_ehci usb_host1_ohci	u2phy1_host

USB 控制器 DTSI 节点如下（以 Linux-4.19 内核为例）

arch/arm64/boot/dts/rockchip/rk3399.dtsi

```
/* USB3.1 OTG0 DWC3 Controller, support USB3.1/2.0/1.1/1.0 */
usbdrd3_0: usb@fe800000 {
        compatible = "rockchip,rk3399-dwc3";
        ......
        usbdrd_dwc3_0: usb@fe800000 {
                compatible = "snps,dwc3";
                ......
                dr_mode = "otg"; /* Type-C0 默认支持 OTG mode */
                phys = <&u2phy0_otg>, <&tcphy0_usb3>;  /* 配置 USB2.0 和 USB3.1
PHY 属性 */
                phy-names = "usb2-phy", "usb3-phy";
                ......
        };
};

/* USB3.1 OTG1 DWC3 Controller, support USB3.1/2.0/1.1/1.0 */
usbdrd3_1: usb@fe900000 {
        compatible = "rockchip,rk3399-dwc3";
        ......
        usbdrd_dwc3_1: usb@fe900000 {
                compatible = "snps,dwc3";
                ......
                dr_mode = "host"; /* Type-C1 默认只支持 Host mode */
                phys = <&u2phy1_otg>, <&tcphy1_usb3>; /* 配置 USB2.0 和 USB3.1
PHY 属性 */
                phy-names = "usb2-phy", "usb3-phy";
                ......
        };
};

/* USB2.0 HOST0 EHCI Controller, support USB2.0 only */
usb_host0_ehci: usb@fe380000 {
        compatible = "generic-ehci";
        ......
        phys = <&u2phy0_host>;
        ......
};

/* USB2.0 Host0 OHCI Controller, support USB1.1/1.0*/
usb_host0_ohci: usb@fe3a0000 {
        compatible = "generic-ohci";
        ......
        phys = <&u2phy0_host>;
        ......
};

/* USB2.0 HOST1 EHCI Controller, support USB2.0 only */
usb_host1_ehci: usb@fe3c0000 {
        compatible = "generic-ehci";
        ......
        phys = <&u2phy1_host>;
        ......
};

/* usb2.0 HOST1 OHCI Controller, support USB1.1/1.0 */
```

```
usb_host1_ohci: usb@fe3e0000 {
        compatible = "generic-ohci";
        ......
        phys = <&u2phy1_host>;
        ....
};
```

USB PHY DTSI 节点如下（以 Linux-4.19 内核为例）

`arch/arm64/boot/dts/rockchip/rk3399.dtsi`

```
/* USB2.0 Combo PHY0/1 */
grf: syscon@ff770000 {
        compatible = "rockchip,rk3399-grf", "syscon", "simple-mfd";
        ......
        u2phy0: usb2-phy@e450 {
                compatible = "rockchip,rk3399-usb2phy";
                ......
                u2phy0_host: host-port { /* 用于 USB2.0 HOST0 */
                        #phy-cells = <0>;
                        ......
                };

                u2phy0_otg: otg-port { /* 用于 Type-C0 OTG 的 USB2 port */
                        #phy-cells = <0>;
                        ......
                };
        };

        u2phy1: usb2-phy@e460 {
                compatible = "rockchip,rk3399-usb2phy";
                ......
                u2phy1_host: host-port { /* 用于 USB2.0 HOST1 */
                        #phy-cells = <0>;
                        ......
                };

                u2phy1_otg: otg-port { /* 用于 Type-C1 OTG 的 USB2 port */
                        #phy-cells = <0>;
                        ......
                };
        };
};

/* USB3.1/DP Combo PHY0 */
tcphy0: phy@ff7c0000 {
        compatible = "rockchip,rk3399-typec-phy";
        ......
        tcphy0_dp: dp-port { /* 用于 Type-C0 的 Display 功能 */
                #phy-cells = <0>;
        };

        tcphy0_usb3: usb3-port { /* 用于 Type-C0 的 USB3.1 功能 */
                #phy-cells = <0>;
        };
};
```

```
/* USB3.1/DP Combo PHY1 */
tcphy1: phy@ff800000 {
        compatible = "rockchip,rk3399-typec-phy";
        ......
        tcphy1_dp: dp-port {  /* 用于 Type-C1 的 Display 功能 */
                #phy-cells = <0>;
        };

        tcphy1_usb3: usb3-port { /* 用于 Type-C1 的 USB3.1 功能 */
                #phy-cells = <0>;
        };
};
```

Note:

1. 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 RK3399 USB 控制器和 PHY 简介。在DTSI 节点中，通过 USB 控制器节点的 "phys" 属性关联对应的 USB PHY；
2. 从 Linux-4.19 开始，USB DWC3 控制器驱动进行了较大的升级，所以，不同内核版本的 DWC3 控制器芯片级配置有所有不同，主要体现在：Linux-4.4 内核rk3399.dtsi 中 DWC3 的 “power-domains”属性和 “resets” 属性是放在 DWC3 控制器的父节点 usbdrd3_x，而 Linux-4.19 以及更新的内核版本，这两个属性移到 DWC3 控制器的子节点 usbdrd_dwc3_x（对于大部分 USB 应用开发者，可以不用关注这个变动）；
3. Rockchip 平台发布的 SDK，默认使用 Type-C0 作为 OTG mode，支持动态切换 Device/Host mode，同时，将 Type-C1 限制为 Host mode。如果产品上需要使用 Type-C1 作为 OTG mode，比如需要支持 Type-C1 USB Device ADB 功能，请参考章节RK3399 Type-C USB 使用注意点和章节Type-C1 支持 OTG mode 的修改方法，详细了解使用限制和软件修改方法。

Type-C USB 3.1/DP 全功能 DTS 配置

Type-C USB 3.1/DP 全功能接口，需要配合外置的 Type-C 控制器芯片，以实现正反面检测、PD（Power Delivery）、USB 角色切换、DP Alternate Mode等功能。为了支持不同的 Type-C 控制器芯片，需要正确配置 Type-C 控制器芯片的相关 DTS 节点。早期发布的 SDK Linux-4.4/Linux-4.19，因为不支持标准 TCPM 软件框架和 TCPCI 协议，所以只能支持 FUSB302。最新的 Linux-4.19 及以上的内核版本，已经支持标准 TCPM 软件框架和 TCPCI 协议，因此可以支持更多的基于 TCPCI 标准设计的 Type-C 控制器芯片（如：HUSB311）。不同内核版本，Type-C 控制器芯片的 DTS 配置方式有所区别，下面分别说明 Rockchip 平台主要适配的 Type-C 控制器芯片 FUSB302 和 HUSB311 在不同 Linux 内核版本的 DTS 配置方法。

Linux-4.4 Type-C USB 3.1&DP 全功能 DTS 配置[FUSB302]

以 RK3399 IND EVB 的 Type-C0 接口为例：

`arch/arm64/boot/dts/rockchip/rk3399-evb-ind.dtsi`

```
&i2c4 {
        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 = <&gpio1 2 GPIO_ACTIVE_HIGH>;
                vbus-5v-gpios = <&gpio1 18 GPIO_ACTIVE_HIGH>;
                status = "okay";
        };
};
```

```
&tcphy0 { /* 使能 Type-C0 USB3.1 PHY */
        extcon = <&fusb0>; /* 配置 extcon 属性，用于接收 FUSB302 驱动的 Type-C
notifier*/
        status = "okay";
};

&u2phy0 { /* 使能 Type-C0 USB2.0 PHY */
        status = "okay";
        extcon = <&fusb0>; /* 配置 extcon 属性，用于接收 FUSB302 驱动的 Type-C DRD
notifier*/
        u2phy0_otg: otg-port {
                status = "okay";
        };
};

&usbdrd3_0 { /* 使能 USB3 控制器父节点 */
        status = "okay";
        extcon = <&fusb0>; /* 配置 extcon 属性，用于接收 FUSB302 驱动的 Type-C DRD
notifier*/
};

&usbdrd_dwc3_0 { /* 使能 USB3 控制器子节点 */
        status = "okay";
};

&pinctrl {
    fusb30x { /* 配置 FUSB302 驱动的中断和 vbus gpio pinctrl */
                fusb0_int: fusb0-int {
                        rockchip,pins = <1 2 RK_FUNC_GPIO &pcfg_pull_up>,
                                        <1 18 RK_FUNC_GPIO &pcfg_pull_up>;
                };
        };
};
```

Linux-4.19 Type-C USB 3.1&DP 全功能 DTS 配置[FUSB302]

Linux-4.19 Type-C DTS 配置[FUSB302] 与 Linux-4.4 基本一致，唯一的区别是： DWC3 控制器节点引用 “extcon” 属性的位置不同。

```
&usbdrd3_0 {
        status = "okay";
};

&usbdrd_dwc3_0 {
        status = "okay";
        extcon = <&fusb0>; /* Linux-4.19 的 extcon 属性放在 DWC3 子节点 */
};
```

Linux-4.19 Type-C USB 3.1&DP 全功能 DTS 配置[HUSB311]

RK3399 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。

硬件设计建议：RK3399 EVB IND 的 FUSB302 可直接替换为 HUSB311 即可。

软件修改注意点：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-dts-rockchip-rk3399-ind-support-type-c-port-co.patch
0003-arm64-rockchip_defconfig-enable-tcpm-and-husb311-and.patch
```

请下载完整的参考资料： "RK3399_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 RK3399 IND EVB 的 Type-C0 适配 HUSB311 为例：

`arch/arm64/boot/dts/rockchip/rk3399-evb-ind.dtsi`

```
/* 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 = <&gpio1 RK_PC2 GPIO_ACTIVE_HIGH>;
        pinctrl-names = "default";
        pinctrl-0 = <&otg_vbus_drv>;
        regulator-name = "vcc_otg_vbus";
        regulator-min-microvolt = <5000000>;
        regulator-max-microvolt = <5000000>;
        enable-active-high;
};

&i2c4 {
        /* 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>;
                /* 中断 gpio 配置, 请根据原理图设计进行相应修改 */
                interrupt-parent = <&gpio1>;
                interrupts = <2 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>;

                        /* rk3399 平台才需要配置 dp altmodes, 默认配置即可 */
                        altmodes {
                                #address-cells = <1>;
                                #size-cells = <0>;

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

&pinctrl {
        husb311 {
                husb311_int: husb311-int {
                        rockchip,pins = <1 RK_PA2 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 = <1 RK_PC2 RK_FUNC_GPIO &pcfg_pull_none>;
                };
        };
};

&cdn_dp {
        status = "okay";
        extcon = <&husb0>;
        phys = <&tcphy0_dp>;
};

bq25700: bq25700@6b {
        compatible = "ti,bq25703";
        reg = <0x6b>;
        extcon = <&husb0>;
        ......
};

&tcphy0 {
        extcon = <&husb0>;
        status = "okay";
};

&u2phy0 {
        status = "okay";
        extcon = <&husb0>;
};

&usbdrd_dwc3_0 {
        status = "okay";
        extcon = <&husb0>;
};
```

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>;
};
```

Linux-5.10 Type-C USB 3.1&DP 全功能 DTS 配置[FUSB302]

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。

同时，需要注意的是，Linux-5.10 RK3399 Type-C 接口的 DP 功能尚未完善，因此，Linux-5.10 tcphy0 节点和 usb_con 节点中，未添加 dp_altmode 节点配置。

以 RK3399 IND EVB 的 Type-C0 接口为例：

`arch/arm64/boot/dts/rockchip/rk3399-evb-ind.dtsi`

```
/* 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 = <&gpio1 18 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*/
&i2c4 {
        usbc0: fusb302@22 {
                compatible = "fcs,fusb302";
                reg = <0x22>;
                interrupt-parent = <&gpio1>;
                interrupts = <RK_PA2 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_0_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)>;
```

```
                        ports {
                                #address-cells = <1>;
                                #size-cells = <0>;
                                /* 用于关联 Type-C PHY，实现正反面切换的功能 */
                                port@0 {
                                        reg = <0>;
                                        usbc0_orien_sw: endpoint {
                                                remote-endpoint =
<&tcphy0_orientation_switch>;
                                        };
                                };
                                /* TODO 关联 Type-C PHY，实现 dp_altmode 功能 */
                        };
                };
        };
};

/* 使能 Type-C0 USB3.1 PHY */
&tcphy0 {
        status = "okay";
        orientation-switch; /* 用于使能 Type-C PHY 驱动注册 orientation switch 回调
函数*/
        port {
                #address-cells = <1>;
                #size-cells = <0>;
                tcphy0_orientation_switch: endpoint@0 {
                        reg = <0>;
                        remote-endpoint = <&usbc0_orien_sw>; /* 关联 FUSB302，实现
正反面切换的功能 */
                };
        };
};

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

        u2phy0_otg: otg-port {
                status = "okay";
        };
};

/* 使能 USB3 控制器父节点 */
&usbdrd3_0 {
        status = "okay";
};

/* 使能 USB3 控制器子节点 */
&usbdrd_dwc3_0 {
        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_0_role_switch: endpoint@0 {
                        reg = <0>;
                        remote-endpoint = <&usbc0_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 = <1 RK_PA2 RK_FUNC_GPIO &pcfg_pull_up>;
                };

                vcc5v0_typec0_en: vcc5v0-typec0-en { /* 配置 FUSB302 驱动的 VBUS
gpio pinctrl */
                        rockchip,pins =
                                <1 RK_PC2 RK_FUNC_GPIO &pcfg_pull_none>;
                };
        };
};
```

Linux-5.10 Type-C USB 3.1&DP 全功能 DTS 配置[HUSB311]

使用 HUSB311 芯片替换 FUSB302 芯片，只需要基于Linux-5.10 Type-C USB 3.1&DP 全功能 DTS 配置[FUSB302]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&DP DTS 配置

Type-C to Type-A USB&DP 的硬件设计方案，有以下特点：

- Type-C 拆分为 USB 和 DP 两个独立的接口；
- 支持 USB 3.1 + DP 2 x lanes；
- 支持 USB 2.0 + DP 4 x lanes；

具体可以细化为两种不同的实现方案，分别是：

1. Type-C to Type-A USB 3.1&DP DTS 配置
2. Type-C to Type-A USB 2.0&DP DTS 配置

Note：

Rockchip 发布的 RK3399 SDK Linux-4.4/4.19 内核 Type-C 驱动默认不支持 USB 和 DP 拆分使用的方式，需要**手动添加特殊的驱动** `drivers/extcon/extcon-pd-virtual.c`，该驱动的作用是替代 Type-C 控制器芯片驱动，发通知给 Type-C PHY 驱动、DP 控制器驱动以及 USB 控制器驱动。如果有该功能需求，请提交需求到 Rockchip 的 Redmine 平台。

Rockchip 发布的 RK3399 SDK Linux-5.10 内核尚未支持 USB 和 DP 拆分使用的方式，计划不再采用添加特殊驱动的方法，功能正在开发中。

RK3399 TypeC to USB2.0 + DP 4xlanes 的软硬件参考设计的下载地址：

https://redmine.rockchip.com.cn/documents/113

Type-C to Type-A USB 3.1&DP DTS 配置

该方案支持 Type-A USB 3.1 (使用 Type-C PHY 的 lane0/1) + DP 1.3 (使用 Type-C PHY 的 lane2/3)。

Linux-4.4/4.19 Type-C0 to Type-A USB 3.1&DP DTS 配置 DTS 参考配置:

```
/* 配置虚拟 pd 驱动，用于发送通知给 Type-C PHY驱动和DP驱动 */
vpd0:virtual-pd0{
        compatible = "linux,extcon-pd-virtual";
        pinctrl-names = "default";
        pinctrl-0 = <&vpd0_int>;
        dp-det-gpios = <&gpio4 25 GPIO_ACTIVE_LOW>; /* 配置 DP 热拔插检测 GPIO */
        hdmi-5v-gpios = <&gpio4 29 GPIO_ACTIVE_LOW>; /* 配置 HDMI 5V 控制 GPIO */

        /* 0: positive, 1: negative*/
        vpd,init-flip = <0>;
        /* 0: u2, 1: u3*/
        vpd,init-ss = <1>;
        /* 0: dfp, 1: ufp, 2: dp, 3: dp&ufp */
        vpd,init-mode = <3>;
        status = "okay";
};

&pinctrl {
        vpd {
                vpd0_int: vpd0-int { /* 配置 GPIO pinctrl */
                        rockchip,pins =
                                <4 25 RK_FUNC_GPIO &pcfg_pull_up>;
                };
        };
};

/* 使能 DP 控制器 */
&cdn_dp {
        status = "okay";
        extcon = <&vpd0>; /* 配置 extcon 属性，用于接收虚拟 PD 驱动的通知 */
        dp_vop_sel = <1>;
};

/* 使能 Type-C0 PHY */
&tcphy0 {
        extcon = <&vpd0>; /* 配置 extcon 属性，用于接收虚拟 PD 驱动的通知 */
        status = "okay";
};

/* 使能 USB2 PHY */
&u2phy0 {
        status = "okay";
        /* 这里不需要配置extcon属性 */

        u2phy0_otg: otg-port {
                status = "okay";
        };
};

/* 使能 USB 控制器父节点 */
&usbdrd3_0 {
```

```
        extcon = <&u2phy0>;  /* extcon 是可选项，只用于 Linux-4.4 且为 Micro 接口。
如果用 Type-A 接口，不用配置，见 Note1 */
        status = "okay";
};

/* 使能 USB 控制器子节点 */
&usbdrd_dwc3_0 {
        dr_mode = "otg"; /* 配置为 otg mode */
        phys = <&u2phy0_otg>, <&tcphy0_usb3>; /* 必须要配置U2和U3 PHY */
        phy-names = "usb2-phy", "usb3-phy";
        extcon = <&u2phy0>;  /* extcon 只用于 Linux-4.19 */
        status = "okay";
};
```

Type-C to Type-A USB 2.0&DP DTS 配置

该方案支持 Type-A USB 2.0 (不使用 Type-C PHY) + DP 4 x lanes to HDMI2.0 (使用 Type-C PHY 的 lane0/1/2/3)

Linux-4.4/4.19 Type-C0 to Type-A USB 2.0&DP DTS 参考配置：

```
/* 配置虚拟 pd 驱动，用于发送通知给 Type-C PHY驱动和DP驱动 */
vpd0:virtual-pd0{
        compatible = "linux,extcon-pd-virtual";
        pinctrl-names = "default";
        pinctrl-0 = <&vpd0_int>;
        dp-det-gpios = <&gpio4 25 GPIO_ACTIVE_LOW>; /* 配置 DP 热拔插检测 GPIO */
        hdmi-5v-gpios = <&gpio4 28 GPIO_ACTIVE_LOW>; /* 配置 HDMI 5V 控制 GPIO */

        /* 0: positive, 1: negative*/
        vpd,init-flip = <0>;
        /* 0: u2, 1: u3*/
        vpd,init-ss = <0>;
        /* 0: dfp, 1: ufp, 2: dp, 3: dp&ufp */
        vpd,init-mode = <2>;
        status = "okay";
};

&pinctrl {
        vpd {
                vpd0_int: vpd0-int { /* 配置 GPIO pinctrl */
                        rockchip,pins =
                                <4 25 RK_FUNC_GPIO &pcfg_pull_up>;
                };
        };
};

/* 使能 DP 控制器 */
&cdn_dp {
        status = "okay";
        extcon = <&vpd0>; /* 配置 extcon 属性，用于接收虚拟 PD 驱动的通知 */
        dp_vop_sel = <1>;
};

/* 使能 Type-C0 PHY */
&tcphy0 {
        extcon = <&vpd0>; /* 配置 extcon 属性，用于接收虚拟 PD 驱动的通知 */
```

```
        status = "okay";
};

/* 使能 USB2 PHY */
&u2phy0 {
        status = "okay";
        /* 这里不需要配置extcon属性 */

        u2phy0_otg: otg-port {
                status = "okay";
        };
};

/* 使能 USB 控制器父节点 */
&usbdrd3_0 {
        extcon = <&u2phy0>;  /* extcon 是可选项，只用于 Linux-4.4 且为 Micro 接口。
如果用 Type-A 接口，不用配置，见 Note1 */
        status = "okay";
};

/* 使能 USB 控制器子节点 */
&usbdrd_dwc3_0 {
        dr_mode = "otg"; /* 配置为 otg mode */
        maximum-speed = "high-speed"; /* 配置最高支持 high-speed */
        phys = <&u2phy0_otg>, <&tcphy0_usb3>; /* 必须要配置U2和U3 PHY */
        phy-names = "usb2-phy", "usb3-phy";
        extcon = <&u2phy0>;  /* extcon 只用于 Linux-4.19 */
        status = "okay";
};
```

Note1：

如果用 Type-A 接口，系统启动后，需要应用层通过内核提供的 OTG mode 切换节点，配置 USB 控制器工作 Peripheral mode 或者 Host mode。配置方法参考RK3399 USB OTG mode 切换命令。

Type-C to Type-A USB only DTS 配置

Type-C to Type-A USB only 的硬件设计方案，有以下特点：

- 只支持 USB3.1/2.0 功能，不支持 DP 功能；
- 不支持 OTG mode 自动切换的功能，但支持软件强制切换 OTG mode；
- 不需要外置的 Type-C 控制器芯片；

具体可以细化为四种不同的实现方案，分别是：

1. Type-C to Type-A USB 3.1 OTG DTS 配置
2. Type-C to Type-A USB 3.1 Host only DTS 配置
3. Type-C to Type-A USB 2.0 OTG DTS 配置
4. Type-C to Type-A USB 2.0 Host only DTS 配置

Type-C to Type-A USB 3.1 OTG DTS 配置

Type-C to Type-A USB 3.1 OTG 的方案，通常应用于 RK3399 平台的 Type-C0 USB，支持 Type-A USB 3.1 OTG 功能，但不支持 DP 功能。系统启功后，USB 控制器默认作为 OTG mode 或者 Device mode，并且无法自动切换 Device/Host mode。如果方案设计中，要求开机后 USB 默认作 Host 功能，则需要应用层通过内核提供的接口，主动设置 USB 控制器工作于 Host mode，软件切换 OTG mode 的方法请参考章节RK3399 USB OTG mode 切换命令 。

硬件设计建议：VBUS 可以设计为常供电或者通过 GPIO/PMIC 进行控制，参考章节USB Host VBUS 的控制方法，并且 Type-C 的三路供电需要正常开启。

Linux-4.4 软件配置 DTS 的注意点：

- Type-C 控制器芯片（如：FUSB302/HUSB311）节点不要配置，因为 Type-A USB 3.1 不需要外置芯片；
- Linux-4.4 内核 tcphy 和 u2phy 节点、USB 控制器父节点 usbdrd3 都不要配置 extcon 属性；
- Linux-4.4 内核 USB 控制器的 "dr_mode" 属性要配置为 "otg"；
- Linux-4.4 内核 u2phy0_otg 节点中，配置 "vbus-supply" 属性（可选项，用于 OTG 作 Host mode 输出 5V）；

Linux-4.19 软件配置 DTS 的注意点：

- Type-C 控制器芯片（如：FUSB302/HUSB311）节点不要配置，因为 Type-A USB 3.1 不需要外置芯片；
- Linux-4.19 内核 tcphy 和 u2phy 节点都不要配置 "extcon" 属性；
- Linux-4.19 内核 USB 控制器子节点 usbdrd_dwc3_0 中增加 "extcon" 属性，用于支持软件切换 OTG mode；
- Linux-4.19 内核 USB 控制器的 "dr_mode" 属性要配置为"otg"；
- Linux-4.19 内核 u2phy0_otg 节点中，配置 "vbus-supply" 属性（可选项，用于 OTG 作 Host mode 输出 5V）；

Linux-5.10 软件配置 DTS 的注意点：

- Type-C 控制器芯片（如：FUSB302/HUSB311）节点不要配置，因为 Type-A USB 3.1 不需要外置芯片；
- Linux-5.10 内核 tcphy 节点不要配置 "orientation-switch" 及其相关属性；
- Linux-5.10 内核 USB 控制器子节点 usbdrd_dwc3_0 中 不要配置 "usb-role-switch" 及其相关属性，但要增加 "extcon" 属性；
- Linux-5.10 内核 USB 控制器的 "dr_mode" 属性要配置为"otg"；
- Linux-5.10 内核 u2phy0_otg 节点中，配置 "vbus-supply"（可选项，用于 OTG 作 Host mode 输出 5V）；

以 Type-C0 USB 配置为 Type-C to Type-A USB 3.1 OTG mode 为例（USB VBUS 通过 GPIO3_PC6 控制）

```
/* 配置 VBUS gpio regulator，用于 OTG USB Host mode 控制 VBUS 5V 输出 */
vcc_otg_vbus: otg-vbus-regulator {
                compatible = "regulator-fixed";
                gpio = <&gpio3 RK_PC6 GPIO_ACTIVE_HIGH>;
                pinctrl-names = "default";
                pinctrl-0 = <&otg_vbus_drv>;
                regulator-name = "vcc_otg_vbus";
                regulator-min-microvolt = <5000000>;
                regulator-max-microvolt = <5000000>;
                enable-active-high;
};

&pinctrl {
        usb {
                otg_vbus_drv: otg-vbus-drv { /* 配置 VBUS gpio pinctrl */
                        rockchip,pins = <3 RK_PC6 RK_FUNC_GPIO &pcfg_pull_none>;
                };
        };
};
```

```
/* 使能 Type-C0 USB3.1 PHY */
&tcphy0 {
        /* 不要配置 extcon 属性，且不要配置 orientation-switch 及其相关属性 */
        status = "okay";
};

/* 使能 Type-C0 USB2.0 PHY */
&u2phy0 {
        /* 不要配置 extcon 属性 */
        status = "okay";

        u2phy0_otg: otg-port {
                vbus-supply = <&vcc_otg_vbus>; /* 可选配置，用于 PHY 驱动控制 OTG
VBUS 输出 5V */
                status = "okay";
        };
};

/* 使能 Type-C0 USB3.1 控制器父节点 */
&usbdrd3_0 {
        /* 不要配置父节点的 extcon 属性 */
        status = "okay";
};

/* 使能 Type-C0 USB3.1 控制器子节点 */
&usbdrd_dwc3_0 {
        dr_mode = "otg"; /* 用于支持 OTG device/host mode 切换 */
        extcon = <&u2phy0>; /* 可选配置，只用于 Linux-4.19/5.10 支持软件切换 OTG mode
*/
        status = "okay";
};
```

Note:

VBUS GPIO 为可选配置。RK3399 平台可以支持 VBUS 为常供电的硬件设计（即系统开机后，VBUS 一直为高）。对于常供电方案，则不需要配置 u2phy0_otg 节点的 "vbus-supply" 属性，但需要增加 "rockchip,vbus-always-on" 属性，否则，可能会出现 USB ADB 无法正常连接的情况。

以 Type-C0 USB 为例，DTS 配置如下（如果是 Type-C1 USB， 应修改对应的 u2phy1_otg 节点）：

```
&u2phy0_otg {
        rockchip,vbus-always-on;
};
```

Type-C to Type-A USB 3.1 Host only DTS 配置

Type-C to Type-A USB 3.1 Host 的方案，在 RK3399 平台的 Type-C1 USB 上广泛应用。比如，RK3399 IND EVB 平台的 Type-C1 USB 默认设计为 Type-A USB 3.1 Host 接口。

硬件设计建议：对应的 USB VBUS 5V 可以为常供电或者 GPIO 控制，同时，要求 Type-C1 的三路供电需要正常开启，才能支持 USB 3.1 Super-speed 功能。

Linux-4.4/4.19 软件配置 DTS 的注意点：

- Type-C 控制器芯片（如：FUSB302/HUSB311）节点不要配置，因为 Type-A USB 3.1 不需要外置芯片；

- Linux-4.4/4.19 内核 tcphy 和 u2phy 节点、USB 控制器父节点 usbdrd3 都不要配置 extcon 属性；
- Linux-4.4/4.19 内核 USB 控制器的 "dr_mode" 属性要配置为"host"；
- Linux-4.4/4.19 内核 u2phy 节点中，配置 "phy-supply" 属性（可选项，用于控制 USB VBUS 输出 5V）；

Linux-5.10 软件配置 DTS 的注意点：

- Type-C 控制器芯片（如：FUSB302/HUSB311）节点不要配置，因为 Type-A USB 3.1 不需要外置芯片；
- Linux-5.10 内核 tcphy 节点不要配置 "orientation-switch" 及其相关属性；
- Linux-5.10 内核 USB 控制器子节点 usbdrd_dwc3_0 中 不要配置 "usb-role-switch" 及其相关属性，且不要配置 "extcon" 属性；
- Linux-5.10 内核 USB 控制器的 "dr_mode" 属性要配置为"host"；
- Linux-5.10 内核 u2phy 节点中，配置 "phy-supply" 属性（可选项，用于控制 USB VBUS 输出 5V）；

以 RK3399 IND EVB 平台 Type-C1 to Type-A USB 3.1 Host 接口配置为例：

`arch/arm64/boot/dts/rockchip/rk3399-evb-ind.dtsi`

```
/* 使能 Type-C1 USB3.1 PHY */
&tcphy1 {
        /* 不要配置 extcon 属性 */
        status = "okay";
};

/* 使能 Type-C1 USB2.0 PHY */
&u2phy1 {
        status = "okay";
        /* 不要配置 extcon 属性 */

        u2phy1_otg: otg-port {
                status = "okay";
                /* phy-supply 属性为可选项 */
        };
};

/* 使能 Type-C0 USB3.1 控制器父节点 */
&usbdrd3_1 {
        status = "okay";
};

/* 使能 Type-C0 USB3.1 控制器子节点 */
&usbdrd_dwc3_1 {
        dr_mode = "host"; /* 配置 dr_mode 为 host，表示只支持 Host only mode */
        status = "okay";
        /* 不要配置 extcon 属性，且不要配置 usb-role-switch 及其相关属性 */
};
```

Type-C to Type-A USB 2.0 OTG DTS 配置

Type-C to Type-A USB 2.0 OTG 的方案，通常应用于 RK3399 平台的 Type-C0 USB，可以认为是Type-C to Type-A USB 3.1 OTG的简化版，只支持 Type-A USB 2.0 OTG 功能。系统启功后，USB 控制器默认作为 OTG mode 或者 Device mode，并且无法自动切换 Device/Host mode。如果方案设计中，要求开机后 USB 默认作 Host 功能，则需要应用层通过内核提供的接口，主动设置 USB 控制器工作于

Host mode，软件切换 OTG mode 的方法请参考章节RK3399 USB OTG mode 切换命令 。

硬件设计建议：VBUS 可以设计为常供电或者通过 GPIO/PMIC 进行控制，参考章节USB Host VBUS 的控制方法，并且 Type-C 的三路供电应该关闭，以降低 Type-C PHY 的功耗。

软件配置 DTS 的注意点，请参考Type-C to Type-A USB 3.1 OTG DTS 配置，此外还需要特别注意：

1. Disable tcphy 节点；
2. USB 控制器节点需要配置 "dr_mode" 属性、"maximum-speed" 属性和 "phys" 属性；

以 Type-C0 USB 配置为Type-C to Type-A USB 2.0 OTG 为例（USB VBUS 通过 GPIO3_PC6 控制）

```
/* 配置 VBUS gpio regulator，用于 OTG USB Host mode 控制 VBUS 5V 输出 */
vcc_otg_vbus: otg-vbus-regulator {
                compatible = "regulator-fixed";
                gpio = <&gpio3 RK_PC6 GPIO_ACTIVE_HIGH>;
                pinctrl-names = "default";
                pinctrl-0 = <&otg_vbus_drv>;
                regulator-name = "vcc_otg_vbus";
                regulator-min-microvolt = <5000000>;
                regulator-max-microvolt = <5000000>;
                enable-active-high;
};

&pinctrl {
        usb {
                otg_vbus_drv: otg-vbus-drv { /* 配置 VBUS gpio pinctrl */
                        rockchip,pins = <3 RK_PC6 RK_FUNC_GPIO &pcfg_pull_none>;
                };
        };
};

/* Disable Type-C0 USB3.1 PHY */
&tcphy0 {
        status = "disabled";
};

/* 使能 Type-C0 USB2.0 PHY */
&u2phy0 {
        /* 不要配置 extcon 属性 */
        status = "okay";

        u2phy0_otg: otg-port {
                vbus-supply = <&vcc_otg_vbus>; /* 可选配置，用于 PHY 驱动控制 OTG
VBUS 输出 5V */
                status = "okay";
        };
};

/* 使能 Type-C0 USB3.1 控制器父节点 */
&usbdrd3_0 {
        /* 不要配置父节点的 extcon 属性 */
        status = "okay";
};

/* 使能 Type-C0 USB3.1 控制器子节点 */
&usbdrd_dwc3_0 {
        dr_mode = "otg"; /* 用于支持 OTG device/host mode 切换 */
```

```
        maximum-speed = “high-speed”; /* maximum-speed 属性配置为 high-speed */
        phys = <&u2phy0_otg>; /* phys 属性只引用 USB2 PHY节点 */
        phy-names = "usb2-phy";
        extcon = <&u2phy0>; /* 可选配置，只用于 Linux-4.19/5.10 支持软件切换 OTG mode
*/
        status = "okay";
};
```

Type-C to Type-A USB 2.0 Host only DTS 配置

Type-C to Type-A USB 2.0 Host only 方案的特点是，进入系统后，只支持 Host 功能，不支持 Device 功能，但可以支持固件烧录。考虑到实际产品中，通常都至少需要 USB Device ADB 功能，因此，这种方案在实际产品中较少使用。

硬件设计建议：对应的 USB VBUS 5V 可以为常供电或者 GPIO 控制，同时，Type-C 的三路供电应该关闭，以降低 Type-C PHY 的功耗。

软件配置 DTS 的注意点，请参考Type-C to Type-A USB 3.1 OTG DTS 配置，此外还需要特别注意：

1. Disable tcphy 节点；
2. USB 控制器节点需要配置 "dr_mode" 属性、"maximum-speed" 属性和 "phys" 属性；

以 Type-C0 USB 配置为 Type-C to Type-A USB 2.0 Host only 为例（USB VBUS 为常供电设计）：

```
/* Disable Type-C0 USB3.1 PHY */
&tcphy0 {
        status = "disabled";
};

/* 使能 Type-C0 USB2.0 PHY */
&u2phy0 {
        /* 不要配置 extcon 属性 */
        status = "okay";

        u2phy0_otg: otg-port {
                /* vbus-supply 属性为可选配置，VBUS 常供电方案不用配置 */
                status = "okay";
        };
};

/* 使能 Type-C0 USB3.1 控制器父节点 */
&usbdrd3_0 {
        /* 不要配置父节点的 extcon 属性 */
        status = "okay";
};

/* 使能 Type-C0 USB3.1 控制器子节点 */
&usbdrd_dwc3_0 {
        dr_mode = "host"; /* 用于支持 OTG device/host mode 切换 */
        maximum-speed = “high-speed”; /* maximum-speed 属性配置为 high-speed */
        phys = <&u2phy0_otg>; /* phys 属性只引用 USB2 PHY节点 */
        phy-names = "usb2-phy";
        /* extcon 属性不用配置，因为不需要支持 OTG mode 切换的功能*/
        status = "okay";
};
```

Type-C to Micro USB DTS 配置

Type-C to Micro USB 3.1 OTG Mode DTS 配置

Micro USB 3.1 接口可以支持硬件自动检测 OTG cable 以及软件自动切换 OTG device/host mode，同时，不需要外置的 Type-C 控制器芯片。因此，相比 Type-C 接口，Micro USB 3.1 接口的硬件设计成本较低，但由于 Micro USB 3.1 的物理接口相比 Type-C 接口大很多，不利于大部分嵌入式设备的设计，因此，在实际产品中使用较少。

硬件设计建议：对应的 USB VBUS 5V 由 GPIO 控制；RK3399 TYPEC0_ID 连接到 Micro USB 3.1 接口的 ID 脚，用于检测 ID 脚的电平变化和切换 OTG mode。

Type-C to Micro USB 3.1 OTG Mode DTS 配置基本与Type-C to Type-A USB 3.1 OTG DTS 配置一样，主要存在三个不同点：

1. Linux-4.4 内核 USB 控制器父节点 usbdrd3 需要配置 extcon 属性，参考如下：

```
/* 使能 Type-C0 USB3.1 控制器父节点 */
&usbdrd3_0 {
        extcon = <&u2phy0>; /* 只用于 Linux-4.4 OTG mode 切换 */
        status = "okay";
};

/* 使能 Type-C0 USB3.1 控制器子节点 */
&usbdrd_dwc3_0 {
        dr_mode = "otg"; /* 用于支持 OTG device/host mode */
        status = "okay";
};
```

2. Linux-4.19/5.10 内核USB 控制器子节点 usbdrd_dwc3 需要配置 extcon 属性，参考如下：

```
/* 使能 Type-C0 USB3.1 控制器父节点 */
&usbdrd3_0 {
        status = "okay";
};

/* 使能 Type-C0 USB3.1 控制器子节点 */
&usbdrd_dwc3_0 {
        dr_mode = "otg"; /* 用于支持 OTG device/host mode */
        extcon = <&u2phy0>; /* 只用于 Linux-4.19/5.10 OTG mode 切换 */
        status = "okay";
};
```

3. u2phy0_otg 节点中，"vbus-supply" 属性为必选项，用于控制 VBUS 5V 的开关。

Type-C to Micro USB 2.0 OTG Mode DTS 配置

Micro USB 2.0 接口可以支持硬件自动检测 OTG cable 以及软件自动切换 OTG device/host mode，同时，不需要外置的 Type-C 控制器芯片。因此，相比 Type-C 接口，Micro USB 2.0 接口的硬件设计成本较低，但最高速率只能支持 USB 2.0 480Mbps。

硬件设计建议：对应的 USB VBUS 5V 由 GPIO 控制；RK3399 TYPEC0_ID 连接到 Micro USB 2.0 接口的 ID 脚，用于检测 ID 脚的电平变化和切换 OTG mode；Type-C 的三路供电应该关闭，以降低 Type-C PHY 的功耗。

Type-C to Micro USB 2.0 OTG Mode DTS 配置基本与Type-C to Type-A USB 2.0 OTG DTS 配置一样，主要存在三个不同点：

1. Linux-4.4 内核 USB 控制器父节点 usbdrd3 需要配置 extcon 属性，参考如下：

```
/* 使能 Type-C0 USB3.1 控制器父节点 */
&usbdrd3_0 {
        extcon = <&u2phy0>; /* 只用于 Linux-4.4 OTG mode 切换 */
        status = "okay";
};

/* 使能 Type-C0 USB3.1 控制器子节点 */
&usbdrd_dwc3_0 {
        dr_mode = "otg"; /* 用于支持 OTG device/host mode 切换 */
        maximum-speed = “high-speed”; /* maximum-speed 属性配置为 high-speed
*/
        phys = <&u2phy0_otg>; /* phys 属性只引用 USB2 PHY节点 */
        phy-names = "usb2-phy";
        status = "okay";
};
```

2. Linux-4.19/5.10 内核USB 控制器子节点 usbdrd_dwc3 需要配置 extcon 属性，参考如下：

```
/* 使能 Type-C0 USB3.1 控制器父节点 */
&usbdrd3_0 {
        status = "okay";
};

/* 使能 Type-C0 USB3.1 控制器子节点 */
&usbdrd_dwc3_0 {
        dr_mode = "otg"; /* 用于支持 OTG device/host mode 切换 */
        maximum-speed = “high-speed”; /* maximum-speed 属性配置为 high-speed
*/
        phys = <&u2phy0_otg>; /* phys 属性只引用 USB2 PHY节点 */
        phy-names = "usb2-phy";
        extcon = <&u2phy0>; /* 只用于 Linux-4.19/5.10 OTG mode 切换 */
        status = "okay";
};
```

3. u2phy0_otg 节点中， "vbus-supply" 属性为必选项，用于控制 VBUS 5V 的开关。

Type-A USB 2.0 Host DTS 配置

RK3399 支持两个 USB2.0 Host 接口，对应的 USB2.0 控制器为 EHCI&OHCI，相比 Type-C 接口的多种硬件设计方案，USB2.0 Host 的接口一般只有一种设计方案，即 Type-A USB2.0 Host 接口，对应的 DTS 配置，包括控制器 DTS 配置和 PHY DTS 配置。

实际方案中，开发者只要根据硬件电路设计，同时参考章节USB 芯片级 DTSI 配置的表4-1，使能对应的 USB2.0 控制器节点和 PHY 节点，并且正确配置 PHY 节点的 “phy-supply ” 属性，该属性用于 PHY 驱动控制 VBUS 5V 输出。Rockchip 平台支持多种 VBUS 控制方法，请参考章节USB Host VBUS 的控制方法。

Type-A USB 2.0 Host DTS 在不同内核版本，配置方法一样。

以 RK3399 IND EVB 的 USB20_HOST0 接口为例：

`arch/arm64/boot/dts/rockchip/rk3399-evb-ind.dtsi`

```
rk809: pmic@20 {
        ....
        regulators {
```

```
                vcc5v0_usb: SWITCH_REG1 { /* 配置 USB VBUS regulator */
                        regulator-always-on;
                        regulator-boot-on;
                        regulator-name = "vcc5v0_usb";
                        regulator-state-mem {
                                regulator-on-in-suspend;
                        };
                };
        };
};

&usb_host0_ehci { /* 使能 USB2.0 EHCI Controller, 支持 USB2.0 only */
        status = "okay";
};

&usb_host0_ohci { /* 使能 USB2.0 OHCI Controller, 支持 USB1.1/1.0 */
        status = "okay";
};

&u2phy0 {
        status = "okay";

        u2phy0_host: host-port {
                phy-supply = <&vcc5v0_usb>; /* 用于 PHY 驱动控制 USB VBUS 输出 5V
*/
                status = "okay";
        };
};
```

Note:

“phy-supply” 属性用于 PHY 驱动控制 USB VBUS regulator 的开关，而不是用于 PHY 本身的供电控制。

RK3399 USB 3.1 force to USB 2.0

该功能是指在 USB 3.0 Tx/Rx 连接的情况下，要强制让 USB 运行在 USB 2.0 的速率。这种应用场景，一般用于硬件设计问题导致 USB 3.0 工作异常或者某些特殊的场景需求，需要去掉 USB 3.0 功能，只要支持 USB 2.0。 由于这不是常规功能，所以 SDK 驱动默认没有支持该功能。Rockchip 以独立的补丁形式，发布给有这类需求的客户。如果有该功能需求，请提交需求到 Rockchip 的 Redmine 平台。

RK3399 USB OTG mode 切换命令

RK3399 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。

- Linux-4.4 USB OTG mode 切换命令

  旧的接口使用方法：

  ```
  #1.Force host mode
    echo host > /sys/kernel/debug/usb@fe800000/rk_usb_force_mode
  #2.Force peripheral mode
    echo peripheral > /sys/kernel/debug/usb@fe800000/rk_usb_force_mode
  ```

新的接口使用方法：

```
#1.Force host mode
  echo host > /sys/devices/platform/usb0/dwc3_mode
#2.Force peripheral mode
  echo peripheral > /sys/devices/platform/usb0/dwc3_mode
```

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

```
#1.Force host mode
  echo host > /sys/devices/platform/ff770000.syscon/ff770000.syscon:usb2-
phy@e450/otg_mode
#2.Force peripheral mode
  echo peripheral >
/sys/devices/platform/ff770000.syscon/ff770000.syscon:usb2-phy@e450/otg_mode
```

- Linux-5.10 内核新增切换命令

```
#1.Force host mode
  echo host > /sys/kernel/debug/usb/fe800000.usb/mode
#2.Force peripheral mode
  echo device > /sys/kernel/debug/usb/fe800000.usb/mode
```

Note：

1. OTG mode 切换命令，都只适用于 RK3399 Type-C0 USB 控制器。假如开发者需要使用 Type-C1 作为 OTG（不建议这么使用），只需将切换命令中 USB 节点名称替换为 Type-C1 对应的 USB 节点即可；
2. 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 的正确配置，以及只能用于非 Type-C 接口的硬件电路设计）。

RK3399 USB DTS 配置的注意点

Linux-4.4 与 Linux-4.19 USB 3.0 DTS配置的差异点

Linux-4.19 USB DWC3 控制器驱动相比 Linux-4.4 进行了较大的升级，所以，USB DWC3 对应的 DTS 配置也有所改动。Linux-4.4 和 Linux-4.19 之间 DWC3 DTS 存在差异点，主要如下：

1. DWC3 的 "power-domains" 属性，"resets" 属性，"extcon" 属性 引用位置不同。在Linux-4.4内核，这三个属性是放在 DWC3 控制器的父节点（usbdrd3），而在Linux-4.19内核，这三个属性移到了 DWC3 控制器的子节点（usbdrd_dwc3）；
2. 在配置 Type-C to Type-A USB DTS 时，Linux-4.19 内核需要在USB 控制器子节点（usbdrd_dwc3）中增加 "extcon" 属性的配置，才能支持软件切换 OTG 模式，而 Linux-4.4 内核无此要求。

Linux-5.10 较之前内核 Type-C DTS 配置的差异点

Linux-5.10 Type-C 驱动采用全新 TCPM 框架，其控制器驱动与之前内核有较大差异，因此 DTS 配置也不同，TCPM 与 USB PHY/Cotroller 采用回调机制通信，不同于之前内核的 extcon 通信。DTS 配置主要差异如下：

1. Type-C 控制器芯片（如：FUSB302）新增 ports 子节点，用于关联 usb-role-switch ;
2. Type-C 控制器芯片（如：FUSB302）新增 connector 子节点，用于描述 Type-C data-role、powr-role、sink-pdos、source-pdos 及 alt-modes 相关信息。具体可参考Linux-5.10 Type-C

USB 3.1&DP 全功能 DTS 配置[FUSB302]。

Linux USB DTS 重要属性说明

1. USB 控制器的属性 "usb-role-switch" 仅用于标准 Type-C 接口（带有 PD 控制器芯片），且需要和 dr_mode = "otg" 配合使用。如果 dr_mode 属性配置为 "host"，需要同时将 "usb-role-switch" 属性删除掉，否则可能导致 DP 功能异常；
2. USB2 PHY 的属性 "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；
3. "extcon" 属性主要功能之一是动态切换 OTG mode。Linux-4.19 内核需要在USB 控制器子节点（usbdrd_dwc3）中增加 "extcon" 属性的配置，才能支持软件切换 OTG 模式；
4. 当 USB3.1 force 为 USB2.0 时，需要增加如下两个 USB 控制器属性

```
"snps,usb2-lpm-disable"   /* 兼容不支持 U2 LPM 的U盘 */
"snps,dis_u2_susphy_quirk" /* 关闭 USB 控制器自动 suspend usb2 phy 的功能 */
```

Type-C1 支持 OTG mode 的修改方法

在章节RK3399 Type-C USB 使用注意点中已经提到，由于内核 USB Gadget 驱动框架只能支持一个 USB 控制器作为 Device 功能，所以 RK3399 SDK 默认配置 Type-C0 作为 OTG mode 支持 USB Peripheral 功能，而 Type-C1 只支持 Host mode。实际产品中，可以根据应用需求，配置 Type-C1 为 OTG mode，支持 USB Peripheral 功能。

Note:

1. 硬件设计上，需要将 TYPEC1_U2VBUSDET 连接到 USB 接口的 VBUS 脚，否则，OTG1 Device 功能无法正常使用；
2. 下载固件仍然使用 Type-C0 接口，无法更改；
3. 双 Type-C USB OTG 的方案尚未支持。因此，如果要支持 Type-C1 OTG mode，请同时将 Type-C0 对应的 USB 控制器节点的 "dr_mode" 属性配置为 "host"。

Linux-4.4/4.19 Type-C1 OTG DTS 修改方法:

```
&usbdrd_dwc3_0 {
        status = "okay";
        dr_mode = "host";  /* 配置 Type-C0 USB 控制器为 host mode */
        extcon = <&fusb1>; /* 注意：extcon 属性要根据实际的硬件电路设计来配置 */
};

&usbdrd_dwc3_1 {
        status = "okay";
        dr_mode = "otg";  /* 配置 Type-C1 USB 控制器为 OTG mode */
        extcon = <&fusb1>; /* 注意：extcon 属性要根据实际的硬件电路设计来配置 */
};
```

Linux-5.10 Type-C1 OTG DTS 修改方法:

```
&usbdrd_dwc3_0 {
        status = "okay";
        /delete-property/ usb-role-switch; /* 需要删除  usb-role-switch 属性，否则会导致 DP 异常 */
        dr_mode = "host";  /* 配置 Type-C0 USB 控制器为 host mode */
};
```

```
&usbdrd_dwc3_1 {
        status = "okay";
        dr_mode = "otg";  /* 配置 Type-C1 USB 控制器为 OTG mode */
        usb-role-switch;  /* 配置 Type-C 相关属性 */
        port {
                #address-cells = <1>;
                #size-cells = <0>;
                dwc3_1_role_switch: endpoint@0 {
                        reg = <0>;
                        remote-endpoint = <&usbc1_role_sw>;
                };
        };
};
```

USB Host VBUS 的控制方法

RK3399 平台的 USB VBUS 供电硬件电路设计，主要有三种方案：

1. 使用 GPIO 控制电源稳压芯片输出 VBUS 5V 供电电压；

   VBUS GPIO 控制是 Rockchip 平台常用的方案，可以用于 Type-C USB 接口，Type-A USB 接口，Micro USB 接口等，不同接口，对应的 DTS 配置方案不同，具体如下：

   （1）Type-C USB 接口的 VBUS GPIO DTS 配置，请参考Type-C USB 3.1/DP 全功能 DTS 配置；

   （2）Type-A USB 接口的 VBUS GPIO DTS 配置，请参考Type-C to Type-A USB only DTS 配置；

   （3）Micro USB 接口的 VBUS GPIO DTS 配置，请参考Type-C to Micro USB DTS 配置；

2. 使用 PMIC（如 RK817/RK818）输出 VBUS 5V 供电电压；

   （1）如果 PMIC 使用的是 RK8xx（RK809 除外），DTS 不用配置 VBUS 的属性，如果配置，反而可能会导致 VBUS 供电异常。这种方案，驱动会通过发送 EXTCON_USB_VBUS_EN 的通知给 PMIC 驱动，以控制 VBUS 的供电。

   （2）如果 PMIC 使用的是其他 Vendor 的芯片，请参考驱动 drivers/power/rk818_charger.c 实现接收 EXTCON_USB_VBUS_EN 通知的逻辑。

   （3）如果 PMIC 使用的是 RK809，由于该 PMIC 只有 VBUS 输出 5V 的供电功能，没有充电功能，所以不适用于使用发送 EXTCON_USB_VBUS_EN 的通知的方法，而应该配置为 USB2 PHY 节点的 "vbus-supply"，由 USB2 PHY 控制 RK809 的 VBUS 开关，参考配置如下：

```
rk809: pmic@20 {
        regulators {
                vcc5v0_host: SWITCH_REG1 {
                        regulator-name = "vcc5v0_host";
                };
        };
};
&u2phy {
        status = "okay";
        u2phy_host: host-port {
                status = "okay";
        };
        u2phy_otg: otg-port {
                vbus-supply = <&vcc5v0_host>;
                status = "okay";
        };
```

```
};
```

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；

Type-C 控制器芯片支持列表

表 9-1 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 FUSB302note1
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 Type-C USB 3.1&DP 全功能 DTS 配置[HUSB311]和 章节Linux-5.10 Type-C USB 3.1&DP 全功能 DTS 配置[HUSB311]，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.

- RK3399 挖掘机 FUSB302 电路设计不适用 HUSB311，需要飞线。RK3399 IND 板子 FUSB302 可直接替换为 HUSB311。内核需要单独使用 CONFIG_TYPEC_HUSB311，DTS 详细配置请参考章节 Linux-4.19 Type-C USB 3.1&DP 全功能 DTS 配置[HUSB311]和 章节Linux-5.10 Type-C USB 3.1&DP 全功能 DTS 配置[HUSB311]。

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。

参考文档

1. kernel/Documentation/devicetree/bindings/usb/generic.txt
2. kernel/Documentation/devicetree/bindings/usb/dwc3.txt
3. kernel/Documentation/devicetree/bindings/usb/rockchip,dwc3.txt
4. kernel/Documentation/devicetree/bindings/usb/usb-ehci.txt
5. kernel/Documentation/devicetree/bindings/usb/usb-ohci.txt
6. kernel/Documentation/devicetree/bindings/phy/phy-rockchip-typec.txt
7. kernel/Documentation/devicetree/bindings/phy/phy-rockchip-inno-usb2.txt
8. kernel/Documentation/devicetree/bindings/connector/usb-connector.yaml
9. kernel/Documentation/devicetree/bindings/usb/fcs,fusb302.txt