

Rockchip DisplayPort 软件开发指南

文件标识: RK-KF-YF-466

发布版本: V1.2.0

日期: 2024-03-25

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

免责声明

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

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

商标声明

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

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

版权所有 © 2024 瑞芯微电子股份有限公司

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

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址: 福建省福州市铜盘路软件园A区18号

网址: www.rock-chips.com

客户服务电话: +86-4007-700-590

客户服务传真: +86-591-83951833

客户服务邮箱: fae@rock-chips.com

前言

本文主要介绍 Rockchip 平台 DP 接口的使用与调试方法。

产品版本

芯片名称	内核版本
RK3576	LINUX Kernel 6.1
RK3588	LINUX Kernel 5.10/6.1

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

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	张玉炳	2022-05-26	初始版本
V1.1.0	张玉炳	2024-01-15	添加功能配置说明
V1.2.0	张玉炳	2024-03-25	添加RK3576配置说明

目录

Rockchip DisplayPort 软件开发指南

- Rockchip 平台 DisplayPort 简介
 - 功能特性
 - DP 与 VOP 连接关系
 - DP 输出
 - 代码路径
 - 驱动加载
- 功能配置
 - 使能 DP
 - DP Alt Mode(Type-C)
 - DP Legacy Mode
 - DP 接 Panel 外设
 - DP 开机 logo
 - DP connector-split mode
 - HDR
 - HDCP
- 常用 DEBUG 方法
 - 查看 connector 状态
 - 强制使能/禁用 DP
 - DPCP 读写
 - Type-C 接口 Debug
 - 查看 DP 寄存器
 - 查看 VOP 状态
 - 查看当前显示时钟
 - 调整 DRM log 等级
 - 查看 DP MST 信息
 - MST Port Info
 - Atomic state info
 - DPCD Info
 - Connector Path Info
- FAQ
 - 插入 DP 无显示或显示异常
 - DP Link Training 成功
 - DP connected
 - DP disconnected
 - Type-C 接口连接异常
 - AUX_CH 异常
 - aux16m clk 值异常
 - phy power on/off 流程异常
 - DP dual mode 转接线导致异常
 - 信号干扰导致异常

- 硬件异常
- 4K 120Hz 输出配置
- DP 带宽计算
 - SST 模式带宽计算
 - MST 模式带宽计算
- DP timing 限制
- MST 模式使用限制
 - 能力限制
 - 分辨率过滤

Rockchip 平台 DisplayPort 简介

功能特性

Rochchip RK3576 和 RK3588 DP 接口功能参数如下表格：

功能	RK3576	RK3588
Version	1.4a	1.4a
SST	Support	Support
MST	Support	Not support
DSC	Not support	Not support
Max resolution	4K@120Hz	8K@30Hz
Main-Link lanes	1/2/4 lanes	1/2/4 lanes
Main-Link rate	8.1/5.4/2.7/1.62 Gbps/lane	8.1/5.4/2.7/1.62 Gbps/lane
AUX_CH	1 M	1 M

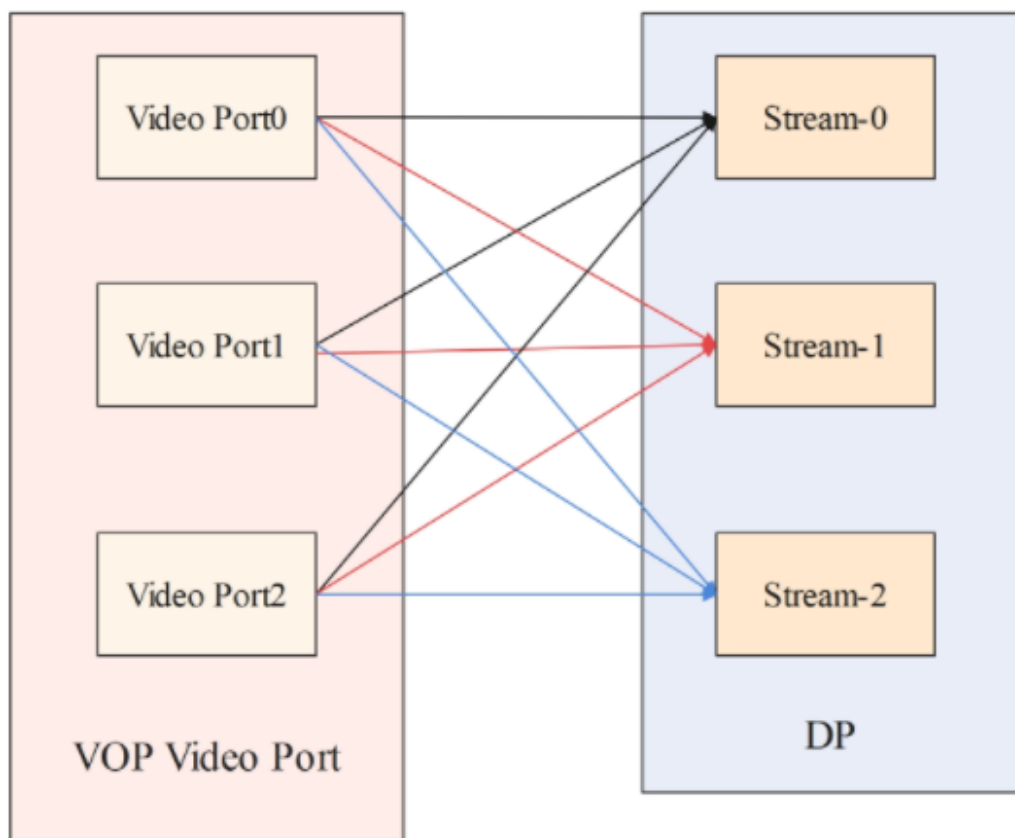
功能	RK3576	RK3588
Color Format	RGB/YUV444/YUV422/YUV420	RGB/YUV444/YUV422/YUV420
Color Depth	8/10 bit(6bit just for RGB)	8/10 bit(6bit just for RGB)
Display Split Mode	Support	Support
HDCP	HDCP2.2/HDCP1.3	HDCP2.2/HDCP1.3
Type-C support	DP Alternate Mode	DP Alternate Mode
I2S	Support	Support
SPDIF	Support	Support
HDR	Support	Support

RK3576 只有一个物理 DP 接口，但在 MST 模式下内部能接受3 路显示数据流(为区分物理接口，用 Stream-0, Stream-1, Stream-2 表示)。每路的最大输出能力如下：

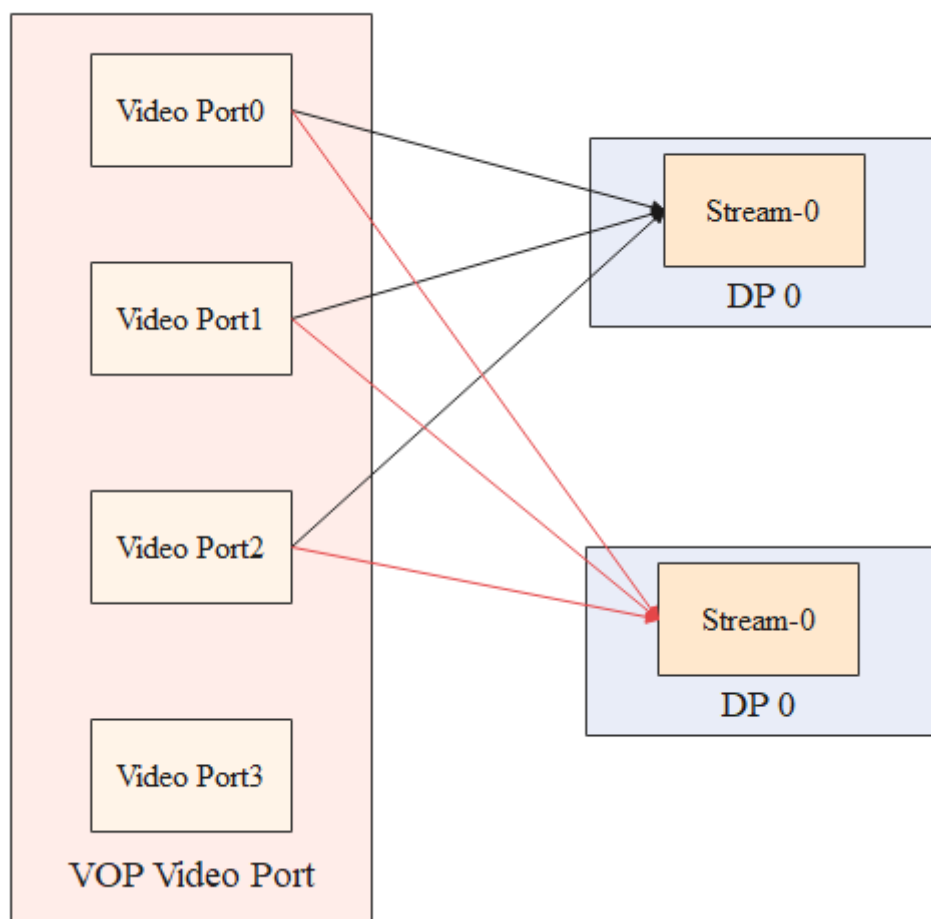
DP Stream Channel	max width	max height	max pixel clock
Stream-0	4096	2160	1188MHz
Stream-1	2560	1440	300MHz
Stream-2	1920	1080	150MHz

DP 与 VOP 连接关系

RK3576 的 VOP 有三个 Video Port, 一个 DP 控制器。在 MST 模式下，DP 控制器支持从 VOP 最多接收 3 路的显示数据流。Stream-0/1/2 均可接收来自 Video Port0/1/2的显示数据。其中，当工作在 SST 模式下时，只能使用 DP 控制器中的 Stream-0。工作在 MST 模式下时，Stream-0/1/2都可以使用。



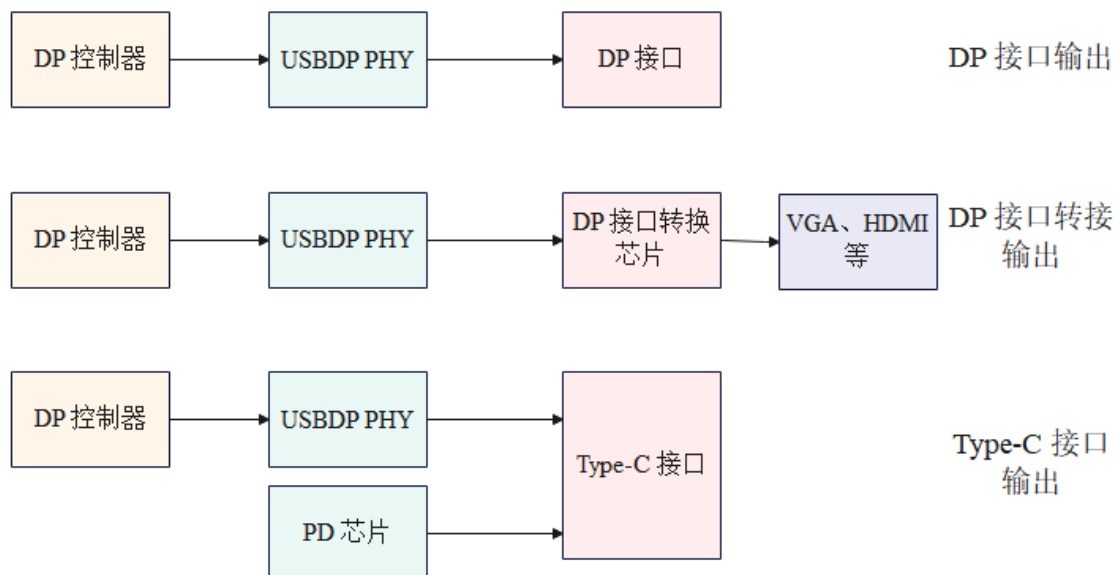
RK3588 的 VOP 有四个 Video Port, 两个 DP 控制器, 其中只有 Video Port 0/1/2 可以输出到 DP0/1, 如下图。



如 RK3588 两个 DP 接口不支持 MST 模式，并且内部只能接收一路显示数据 Stream-0。对于这种不支持 MST 的平台，默认 Video Port 输出输出到 DP 接口的 Stream-0。

DP 输出

根据应用场景的不同，可以设计不同的 DP 输出方式：Type-C 接口输出、DP 标准接口输出、通过其他转接芯片转接输出。



RK3576 在 MST 模式下，最多可以接 3 台显示器，可以通过 MST 显示器通过菊花链的方式串联，如下：

通过菊花链连接的显示器，只有最后一台显示器可以接 SST 显示器，其他的需要 MST 显示器。

另一种方式，可以通过 MST HUB 进行连接，如下：

通过 MST HUB 连接时，DP 显示器可以是 SST 显示器，也可以是 MST 显示器。

代码路径

U-Boot 驱动代码：

```
drivers/video/drm/dw-dp.c
drivers/phy/phy-rockchip-usbdp.c
```

Kernel 驱动代码：

```
drivers/gpu/drm/rockchip/dw-dp.c
drivers/phy/rockchip/phy-rockchip-usbdp.c
```

RK3576 参考 DTS 配置：

```
arch/arm64/boot/dts/rockchip/rk3576-evb1.dtsi
arch/arm64/boot/dts/rockchip/rk3576-test2.dtsi
```

RK3588 参考 DTS 配置：

```
arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
arch/arm64/boot/dts/rockchip/rk3588-evb2-lp4.dtsi
arch/arm64/boot/dts/rockchip/rk3588-evb3-lp5.dtsi
arch/arm64/boot/dts/rockchip/rk3588-nvr-demo.dtsi
```

驱动加载

通过下面的log，判断驱动加载是否完成：

```
RK3576:
[1.991964] rockchip-drm display-subsystem: bound 27e40000.dp (ops
0xffffffffc0094a1570) //DP 驱动加载完成
RK3588:
[2.472282] rockchip-drm display-subsystem: bound fde50000.dp (ops
dw_dp_component_ops) //DP0 驱动加载完成
[2.472319] rockchip-drm display-subsystem: bound fde60000.dp (ops
dw_dp_component_ops) //DP1 驱动加载完成
```

功能配置

对于 DP 接口，支持 MST 和 不支持 MST 的平台 DTS 节点的基础配置存在差异。

不支持 MST 的平台，如 RK3588，一个 DP 控制器只支持一路 DP 输出，只需定义一个 ports 子节点描述这路 DP 可以支持的显示通路即可，DP 节点描述如下：

```
dp0: dp@fde50000 {
    compatible = "rockchip,rk3588-dp";
    ...

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

        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            dp0_in_vp0: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&vp0_out_dp0>;
                status = "disabled";
            };

            dp0_in_vp1: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&vp1_out_dp0>;
                status = "disabled";
            };

            dp0_in_vp2: endpoint@2 {
                reg = <2>;
                remote-endpoint = <&vp2_out_dp0>;
                status = "disabled";
            };
        };
    };
};
```

```
};
...
};
};
```

对于支持 MST 的平台，如 RK3576，一个 DP 控制器要支持 3 路显示数据流输出，一个 ports 节点无法描述多个 DP 通道的显示通路，需要通过多个子节点描述，配置如下：

```
dp: dp@27e40000 {
    compatible = "rockchip,rk3576-dp";
    ...
    dp0: dp0 {
        status = "disabled";

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

            port@0 {
                reg = <0>;
                #address-cells = <1>;
                #size-cells = <0>;

                dp0_in_vp0: endpoint@0 {
                    reg = <0>;
                    remote-endpoint = <&vp0_out_dp0>;
                    status = "disabled";
                };

                dp0_in_vp1: endpoint@1 {
                    reg = <1>;
                    remote-endpoint = <&vp1_out_dp0>;
                    status = "disabled";
                };

                dp0_in_vp2: endpoint@2 {
                    reg = <2>;
                    remote-endpoint = <&vp2_out_dp0>;
                    status = "disabled";
                };
            };
        };
    };

    dp1: dp1 {
        status = "disabled";

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

            port@0 {
                reg = <0>;
                #address-cells = <1>;
                #size-cells = <0>;

                dp1_in_vp0: endpoint@0 {
```

```

        reg = <0>;
        remote-endpoint = <&vp0_out_dp1>;
        status = "disabled";
    };

    dp1_in_vp1: endpoint@1 {
        reg = <1>;
        remote-endpoint = <&vp1_out_dp1>;
        status = "disabled";
    };

    dp1_in_vp2: endpoint@2 {
        reg = <2>;
        remote-endpoint = <&vp2_out_dp1>;
        status = "disabled";
    };
};

};

dp2: dp2 {
    status = "disabled";

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

            dp2_in_vp0: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&vp0_out_dp2>;
                status = "disabled";
            };

            dp2_in_vp1: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&vp1_out_dp2>;
                status = "disabled";
            };

            dp2_in_vp2: endpoint@2 {
                reg = <2>;
                remote-endpoint = <&vp2_out_dp2>;
                status = "disabled";
            };
        };
    };
};
};
};

```

上述的 dp0/1/2 子节点，分别描述 DP 控制器中 Stream-0/1/2 可以支持的显示通路。

对比 DTS 的配置，支持 MST 的平台上多了一层 DP 通道的子节点。

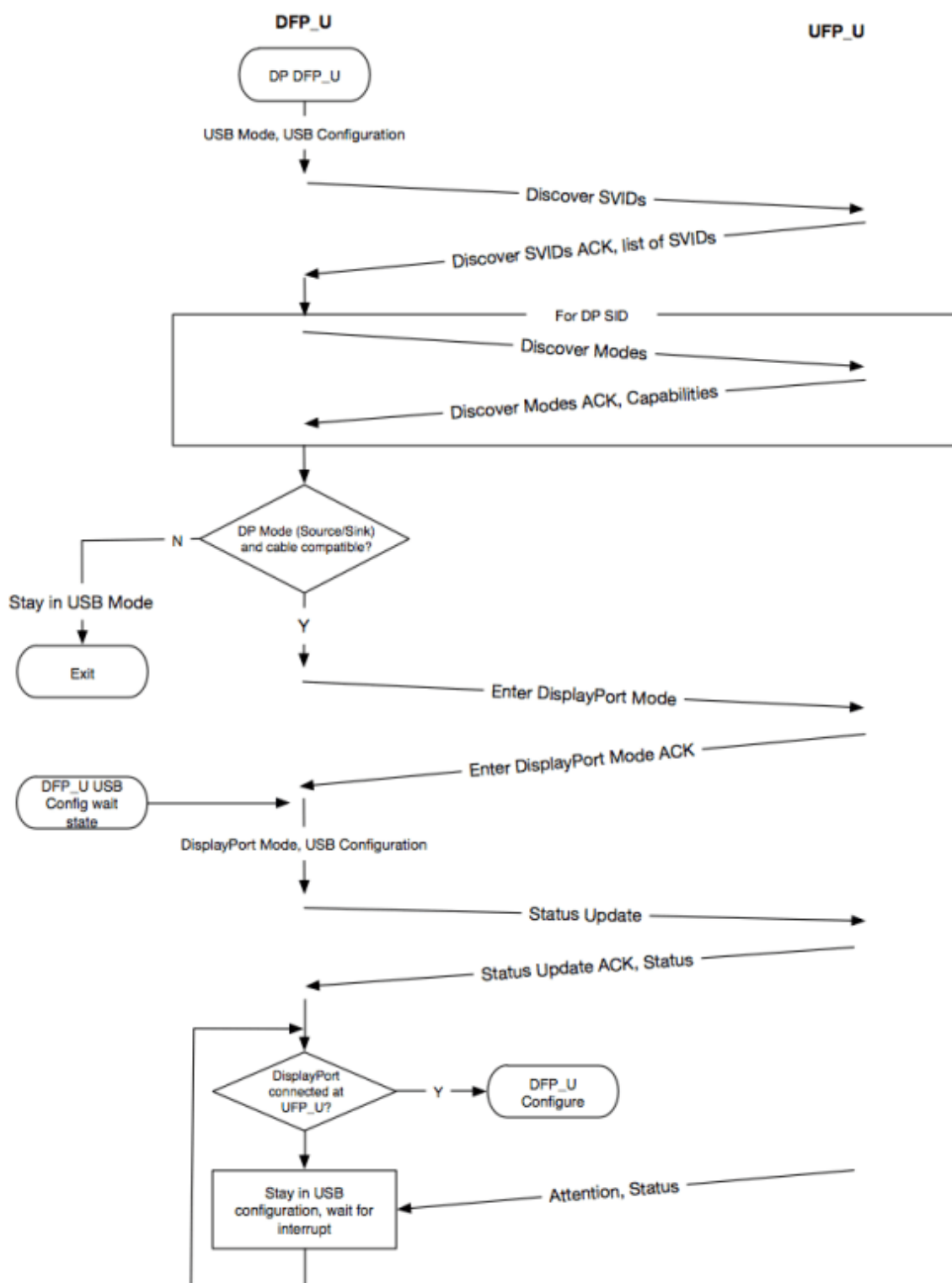
使能 DP

DP 和 USB3.0 共用 PHY，PHY lane 的配置根据接口的不同有两种方式，Type-C 模式和非 Type-C 模式。

DP Alt Mode(Type-C)

根据 DisplayPort Alt Mode 协议，通过 PD (Power Delivery) 的状态机和显示器进行通信，进行 lane 的映射和 HPD 信息的传递。通过 PD 协议进入 DP Mode 并通过 attention 指令传递 HPD 信息的流程主要如下图所示。

Figure 5-2 DFP_U DisplayPort Alternate Mode Discovery and Entry



不支持 MST 的平台，如 RK3588，配置如下：

```
&dp0 {
    status = "okay";
};

&dp0_in_vp2 {
    status = "okay";
};
```

在上面的配置中，使能了 DP0 接口，并把 DP0 绑定到 VOP 的 Video Port2，这只是一种参考配置，实际使用过程中，可以根据实际的需求，使能 DP0 或 DP1，并把 DP0 或 DP1 绑定到期望的 Video Port(0/1/2) 上。

支持 MST 的平台，如 RK3576，配置如下：

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

&dp0 {
    status = "okay";
};

&dp0_in_vp2 {
    status = "okay";
};
```

可以看到，支持 MST 的平台，需要使能 DP 设备节点，要开启的 DP Stream 通道，以及该通道要绑定的 VOP 上的 Video Port。上述的配置中，即使了 DP 接口的 Stream-0，并把 Stream-0 绑定到 VOP 的 Video Port2。

需要注意的是，支持 MST 的平台，因为 SST 模式下一定要使用 DP Steam-0，所以 dp0 节点是一定要使能的。dp1 和 dp2 根据使用情况进行配置。

PHY 配置如下，支持 MST 和不支持 MST 的平台无差异，参考如下 RK3588 usbdp phy0 的配置：

```
&usbdp_phy0 {
    status = "okay";
    orientation-switch;
    /* DP related config */
    svid = <0xff01>;
    sbu1-dc-gpios = <&gpio4 RK_PA6 GPIO_ACTIVE_HIGH>;
    sbu2-dc-gpios = <&gpio4 RK_PA7 GPIO_ACTIVE_HIGH>;
    /* DP related config */

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

        /* DP related config */
        usbdp_phy0_dp_altmux_mux: endpoint@1 {
```

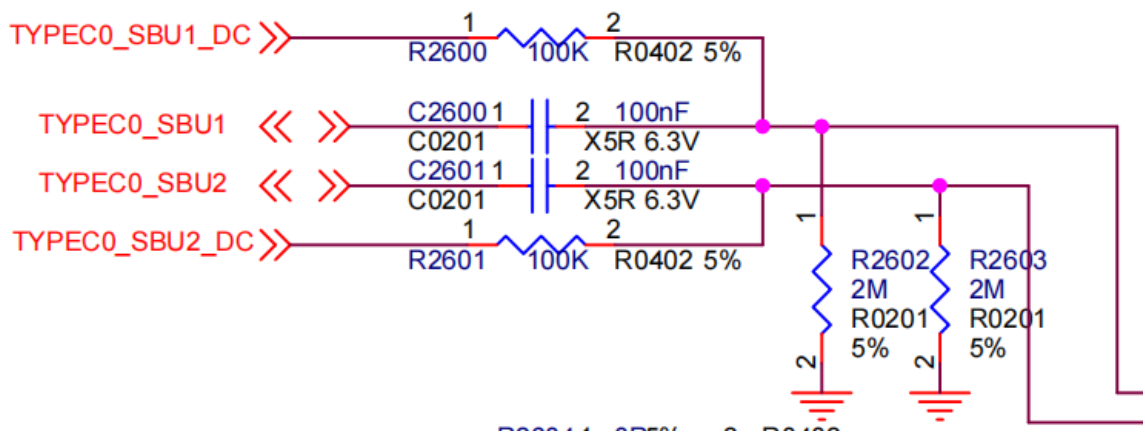
```

        reg = <1>;
        remote-endpoint = <&dp_altmode_mux>;
    };
    /* DP related config */
};
};
};

```

sbu1-dc-gpios 和 **sbu2-dc-gpios** :

Type-C 的 SBU1 和 SBU2 引脚是和 DP 的 AUX_CH 复用的, 在 Type-C 正插时, AUX_CH_P 复用 SBU1, AUX_CH_N 复用 SUB2。在 Type-C 反插时, AUX_CH_P 复用 SBU2, AUX_CH_N 复用 SBU1。根据 DP 协议要求, AUX_CH_P 需要配置为下拉状态, AUX_CH_N 需要配置成上拉状态。Type-C 不同的插入状态(正插和反插) AUX_CH_N 和 AUX_CH_P 的复用配置是不一样的, 在 RK 方案上, 是通过两个 GPIO 来分别控制 SBU1 和 SBU2 的上下拉状态, 即 dts 中的 sbu1-dc-gpios 和 sbu2-dc-gpios。因此, 在配置 PHY 时, 需要配置 sbu1-dc-gpios 和 sbu2-dc-gpios (实际配置这两个 GPIO 的时候要参照硬件设计的原理图, 例如下图的 TYPEC0_SBU1_DC 和 TYPEC0_SBU2_DC), PHY 驱动会根据当前的 Type-C 正反插状态去调整 GPIO 输出的电平。



svid:

对 DP 来说是固定值 0xff01。

Type-C 接口需要通过 Type-C 的 CC 检测和 PD 协商来配置 lane 和 HPD 的状态, 所以还需要配置 PD 芯片:

```

&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 = <&dw3_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)>;

    /* DP related config */
    altmodes {
        #address-cells = <1>;
        #size-cells = <0>;

        altmode@0 {
            reg = <0>;
            svid = <0xff01>;
            vdo = <0xffffffff>;
        };
    };
    /* DP related config */

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

        port@0 {
            reg = <0>;
            usbc0_orien_sw: endpoint {
                remote-endpoint = <&usbdp_phy0_orientation_switch>;
            };
        };

        /* DP related config */
        port@1 {
            reg = <1>;
            dp_altmode_mux: endpoint {
                remote-endpoint = <&usbdp_phy0_dp_altmode_mux>;
            };
        };
        /* DP related config */
    };
};
};
};
};

```

altmode@0 节点中, svid 固定配置为 0xff01, vdo 固定配置为 0xffffffff。

Note: 当前支持的 PD 芯片为 fusb302, hub311。fusb302 对应的驱动为/drivers/usb/typec/tcpm/fusb302.c, hub311 对应的驱动为/drivers/usb/typec/tcpm/tcpci_husb311.c。

DP Legacy Mode

非 Type-C 接口输出，无论是 DP 接口，还是通过其他的转接芯片输出，配置流程基本一致，并且都需要配置 HPD Pin。在实际分配 IO 引脚的时候，可以使用 DP_HPDP 专用引脚，这种情况按 IOMUX 进行配置，还可以使用普通的 GPIO 进行检测。

对于不支持 MST 的平台，如 RK3588, 使用 DP_HPDP Pin 的时候配置如下：

```
&dp1 {
    pinctrl-0 = <&dp1m2_pins>;
    pinctrl-names = "default";
    status = "okay";
};

&dp1_in_vp2 {
    status = "okay";
};
```

使用普通 GPIO 作 HPD 检测的时候配置如下：

```
&dp1 {
    pinctrl-names = "default";
    pinctrl-0 = <&dp1_hpd>;
    hpd-gpios = <&gpio1 RK_PB5 GPIO_ACTIVE_HIGH>;
    status = "okay";
};

&dp1_in_vp2 {
    status = "okay";
};

&pinctrl {
    dp {
        dp1_hpd: dp1-hpd {
            rockchip,pins = <1 RK_PB5 RK_FUNC_GPIO &pcfg_pull_down>;
        };
    };
};
```

对于支持 MST 的平台，比如 RK3576, 使用 DP_HPDP Pin 的时候配置如下：

```

&dp {
    pinctrl-0 = <&dp1m2_pins>;
    pinctrl-names = "default";
    status = "okay";
};

&dp0 {
    status = "okay";
};

&dp0_in_vp2 {
    status = "okay";
};

```

使用普通 GPIO 作 HPD 检测的时候配置如下：

```

&dp {
    pinctrl-names = "default";
    pinctrl-0 = <&dp_hpd>;
    hpd-gpios = <&gpio1 RK_PB5 GPIO_ACTIVE_HIGH>;
    status = "okay";
};

&dp0 {
    status = "okay";
};

&dp0_in_vp2 {
    status = "okay";
};

&pinctrl {
    dp {
        dp_hpd: dp-hpd {
            rockchip,pins = <1 RK_PB5 RK_FUNC_GPIO &pcfg_pull_down>;
        };
    };
};

```

上述支持 MST 平台和 不支持 MST 平台的配置中，HPD 的配置是属于整个 DP 接口的配置，均配置在设备节点下。

DP 和 USB 3.0 共用 PHY，当 DP 为 非 Type-C 接口输出时，就需要指定 lane 配置给 DP 使用以及对应的 lane 序号，这部分内容在 DTS 中指定。对于 DP PHY lane 的配置，可以配置成 2 lane 模式或 4 lane 模式。

PHY lane 接口的物理编号和 Pin 脚的关系如下：

Pin Name	SSRX1	SSTX1	SSRX2	SSTX2
Phy Lane	0	1	2	3

对于 DP 配置 4 lane, dtsti 配置属性如下：

```

rockchip,dp-lane-mux = <x x x x>;

```

对于 DP 配置 2 lane, dtsi 配置属性如下：

```
rockchip,dp-lane-mux = <x x>;
```

其中，索引为 DP 的 lane, 值为 PHY 的 lane。

无论 2 lane 还是 4 lane 配置，硬件设计时一般使用如下的 OPTION1 或 OPTION2 两种中的一种。

Pin Name	Type-C Function	DPx4Lane Function		USB30 HOST+DPx2Lane Function	
		OPTION1	OPTION2	OPTION1	OPTION2
TYPEC0_SBU1/DP0_AUXP TYPEC0_SBU2/DP0_AUXN	TYPEC0_SBU1 TYPEC0_SBU2	DP0_AUXP DP0_AUXN	DP0_AUXP DP0_AUXN	DP0_AUXP DP0_AUXN	DP0_AUXP DP0_AUXN
TYPEC0_SSRX1P/DP0_TX0P TYPEC0_SSRX1N/DP0_TX0N	TYPEC0_SSRX1P TYPEC0_SSRX1N	DP0_TX0P DP0_TX0N	DP0_TX2P DP0_TX2N	TYPEC0_SSRX1P TYPEC0_SSRX1N	DP0_TX0P DP0_TX0N
TYPEC0_SSTX1P/DP0_TX1P TYPEC0_SSTX1N/DP0_TX1N	TYPEC0_SSTX1P TYPEC0_SSTX1N	DP0_TX1P DP0_TX1N	DP0_TX3P DP0_TX3N	TYPEC0_SSTX1P TYPEC0_SSTX1N	DP0_TX1P DP0_TX1N
TYPEC0_SSRX2P/DP0_TX2P TYPEC0_SSRX2N/DP0_TX2N	TYPEC0_SSRX2P TYPEC0_SSRX2N	DP0_TX2P DP0_TX2N	DP0_TX0P DP0_TX0N	DP0_TX2P DP0_TX2N	TYPEC0_SSRX2P TYPEC0_SSRX2N
TYPEC0_SSTX2P/DP0_TX3P TYPEC0_SSTX2N/DP0_TX3N	TYPEC0_SSTX2P TYPEC0_SSTX2N	DP0_TX3P DP0_TX3N	DP0_TX1P DP0_TX1N	DP0_TX3P DP0_TX3N	TYPEC0_SSTX2P TYPEC0_SSTX2N
TYPEC0_USB20_OTG_DP TYPEC0_USB20_OTG_DM	TYPEC0_USB20_OTG_DP TYPEC0_USB20_OTG_DM			TYPEC0_USB20_OTG_DP TYPEC0_USB20_OTG_DM	TYPEC0_USB20_OTG_DP TYPEC0_USB20_OTG_DM

对于 DP 4 lane 的 OPTION1 映射关系如下：

DP lane	Phy lane	Pin Name
0	0	SSRX1
1	1	SSTX1
2	2	SSRX2
3	3	SSTX2

其中 DP lane 为 DP 的 lane 的序号。

dts 的配置如下：

```
&usbdp_phy1 {
    rockchip,dp-lane-mux = <0 1 2 3>;
    status = "okay";
};
```

对于 DP 4 lane 的 OPTION2 映射关系如下：

DP lane	Phy lane	Pin Name
0	2	SSRX2
1	3	SSTX2
2	0	SSRX1
3	1	SSTX1

其中 DP lane 为 DP 的 lane 的序号。

dts 的配置如下：

```
&usbdp_phy1 {
    rockchip,dp-lane-mux = <2 3 0 1>;
    status = "okay";
};
```

对于 DP 2 lane 的 OPTION1 映射关系如下：

DP lane	Phy lane	Pin Name
0	2	SSRX2
1	3	SSTX2

DP 2 lane 的配置如下：

```
&usbdp_phy1 {
    rockchip,dp-lane-mux = <2 3>;
    status = "okay";
};
```

对于 DP 2 lane 的 OPTION2 映射关系如下：

DP lane	Phy lane	Pin Name
0	0	SSRX1
1	1	SSTX1

DP 2 lane 的配置如下：

```
&usbdp_phy1 {
    rockchip,dp-lane-mux = <0 1>;
    status = "okay";
};
```

DP 接 Panel 外设

使用 DP 接口接 eDP Panel 时，eDP 独有的特性无法支持，比如 PSR, Multi-SST, ALPM。Panel 的配置可以参考如下，并根据实际的硬件设计进行调整：

对于不支持 MST 的平台，如 RK3588 配置：

```
/ {
    ...
    panel-edp {
        compatible = "simple-panel";
        backlight = <&backlight>;
        power-supply = <&vcc3v3_lcd_edp>;
        prepare-delay-ms = <120>;
        enable-delay-ms = <120>;
        unprepare-delay-ms = <120>;
        disable-delay-ms = <120>;
        width-mm = <120>;
        height-mm = <160>;
    };
};
```

```

        panel-timing {
            clock-frequency = <200000000>;
            hactive = <1536>;
            vactive = <2048>;
            hfront-porch = <12>;
            hsync-len = <16>;
            hback-porch = <48>;
            vfront-porch = <8>;
            vsync-len = <4>;
            vback-porch = <8>;
            hsync-active = <0>;
            vsync-active = <0>;
            de-active = <0>;
            pixelclk-active = <0>;
        };

        port {
            panel_in_edp: endpoint {
                remote-endpoint = <&dp0_out>;
            };
        };
    };
    ...
};

&dp0 {
    force-hpd;
    status = "okay";
};

&dp0_in_vp2 {
    status = "okay";
};

&dp0_out {
    remote-endpoint = <&panel_in_edp>;
};

&usbdp_phy0 {
    rockchip,dp-lane-mux = <0 1 2 3>;
    status = "okay";
};

```

对于支持 MST 的平台，如 RK3576 对应的配置参考如下：

```

/ {
    ...
    panel-edp {
        compatible = "simple-panel";
        backlight = <&backlight>;
        power-supply = <&vcc3v3_lcd_edp>;
        prepare-delay-ms = <120>;
        enable-delay-ms = <120>;
        unprepare-delay-ms = <120>;
        disable-delay-ms = <120>;
        width-mm = <120>;
    };
};

```

```

height-mm = <160>;

panel-timing {
    clock-frequency = <200000000>;
    hactive = <1536>;
    vactive = <2048>;
    hfront-porch = <12>;
    hsync-len = <16>;
    hback-porch = <48>;
    vfront-porch = <8>;
    vsync-len = <4>;
    vback-porch = <8>;
    hsync-active = <0>;
    vsync-active = <0>;
    de-active = <0>;
    pixelclk-active = <0>;
};

port {
    panel_in_edp: endpoint {
        remote-endpoint = <&dp0_out_panel>;
    };
};
...
};

&dp {
    force-hpd;
    status = "okay";
};

&dp0 {
    status = "okay";

    ports {
        port@1 {
            reg = <1>;

            dp0_out_panel: endpoint {
                remote-endpoint = <&panel_in_edp>;
            };
        };
    };
};

&dp0_in_vp2 {
    status = "okay";
};

&usbdp_phy {
    rockchip,dp-lane-mux = <0 1 2 3>;
    status = "okay";
};

```

上述配置中，`force-hpd` 是描述整个 DP 接口 HPD 的属性，要放在设备节点下。显示通路的配置和具体的 DP 显示通路有关，所以 MST 的平台需要修改具体的显示通路子节点。

对于支持 MST 的平台，目前接 panel 时只能工作在 SST 模式下，所以显示通路只能使用 Steam-0, 对应 dp0 节点。

上述的配置中，dp0 节点中的 force-hpd 的属性配置后，驱动默认 eDP panel 都是处于连接的状态，这个属性不是必须的，要根据具体的屏是否有 HPD 引出，HPD 拉高和 AUX 访问是否有时序要求等确认是否要配置 force-hpd 属性。如果 panel 要求 AUX 的访问必须在 HPD 拉高之后，就不能配置 force-hpd 属性，否则有可能出现 HPD 未拉高之前就访问 AUX，导致 AUX 访问失败。如果还是需要配置 HPD 引脚，参考 1.2.1 DP Legacy Mode 的 HPD 的配置。

panel-timing 配置当前支持的 timing，如果 eDP panel 没有 EDID，或者 EDID 读到的 timing 不准，就需要配置 panel-timing 节点，否则可以不用配置，直接通过读 EDID 获取。

上下电时序和背光根据具体的屏幕和硬件设计进行配置。

DP 开机 logo

配置开机 logo 后，如果在开机前就插入 DP 显示器，即可在 U-Boot 阶段就开始显示 logo，否则，只能等到系统启动后才能看到应用显示的图像。添加 DP 开机 logo 支持的配置如下：

```
&route_dp0 {
    status = "okay";
    connect = <&vp2_out_dp0>;
};
```

需要注意的是，这里的 connect 属性配置 DP 在 U-Boot 阶段绑定 VOP Port2, 所以 dtsi 中的配置要允许 DP 绑定 VOP Port2：

```
&dp0_in_vp2 {
    status = "okay";
};
```

Note:

- 1 目前不支持 Type-C 接口的 DP 开机 logo;
- 2 对于支持 MST 的平台，开机 LOGO 只支持在 SST 模式下显示。

DP connector-split mode

DP connector-split mode 如图所示，一幅图像被平分成左右两部分，并分别通过 DP0/DP1 接口传输给显示器，下图中 DP0 作为左半屏，DP1 作为右半屏。

配置如下：

```
&dp0 {
    split-mode;
    status = "okay";
};

&dp0_in_vp2 {
    status = "okay";
};

&dp1 {
    status = "okay";
};
```

在作为左半屏的 DP 节点加入 `split-mode` 属性，并绑定要输出的 Video Port，在如上的配置中，即 DP0 作为左边屏，DP1 作为右半屏。在 Split Mode 模式下，两个 DP 当作一个 connector，只有 DP0 和 DP1 同时连接时，这个 connector 才处于连接状态，才会开始显示，只要有一个 DP 接口处于断开状态，connector 即处于断开状态，不会输出显示。在该模式下，两个 DP 接口输出的时序是一样的，建议使用两个一样的显示器。

在用户空间下，通过 `modetest` 或者 `cat dri` 的 `state` 节点(`cat /sys/kernel/debug/dri/0/state`)，只会看到一个 DP connector。

如果要在 split mode 下显示 U-boot logo，比如 DP0 作左半屏，需要添加的参考配置如下：

```
&route_dp0 {
    split-mode;
    status = "okay";
    connect = <&vp2_out_dp0>;
};

&route_dp1 {
    status = "disabled";
}
```

Note: RK3576 只有一个接口不支持这种方式的 connector-split mode，后续补充 RK3576 支持的 split-mode 功能，如果有 RK3576 上的 split-mode 功能需求，请联系 Rockchip。

HDR

HDR 功能默认在 SST 模式下支持，驱动不需要配置，MST下的 HDR 功能暂不支持。

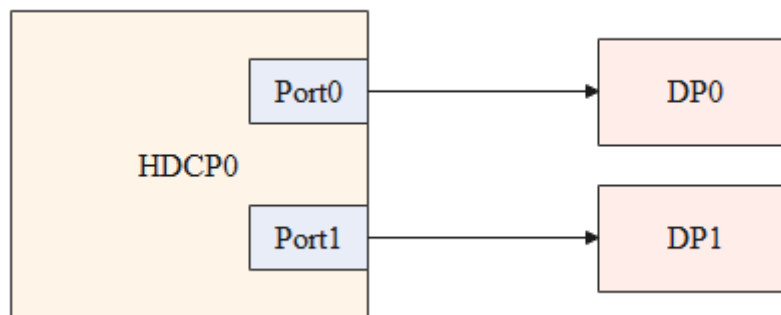
HDCP

DP 驱动基于 DRM 框架实现 HDCP 功能，用户使用 HDCP 功能需要在 userspace 调用 DRM 的接口实现。

HDCP1.3 DP 驱动默认支持，无需配置，HDCP Key 的烧录参考《Rockchip_RK3588_Developer_Guide_HDCP_CN》。

HDCP2.2 有单独的 HDCP2 控制器来控制 HDCP 的认证，使能 HDCP2 控制器需要配置 dts。

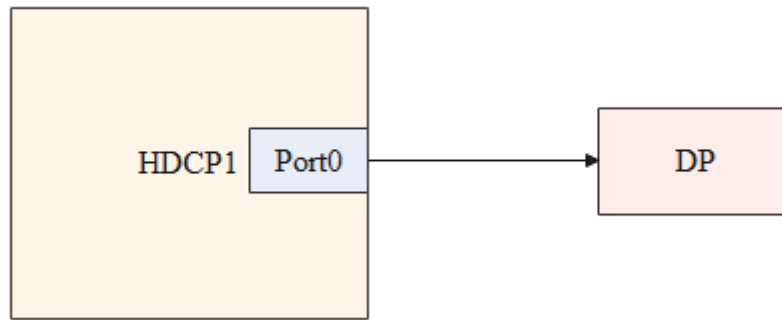
在 RK3588 上，DP0/DP1 和 HDCP0 相连，如下图：



使能 DP HDCP2.2 功能，需要使能如下节点：

```
&hdc0 {
    status = "okay";
};
```

在 RK3576 上，DP 和 HDCP1 相连，如下下图：



使能 DP HDCP2.2 功能，需要使能如下节点：

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

使用 HDCP2.2 除了驱动配置外，还需使能 userspace 的 HDCP2.2 的应用程序及生成 HDCP 控制器的固件，参考《Rockchip_RK3588_Developer_Guide_HDCP_CN》。

常用 DEBUG 方法

查看 connector 状态

在 /sys/class/drm 目录下可以看到驱动注册的各个 card，在如下显示的内容汇总，card0-DP-1 和 card0-DP-2 是 DP 显示设备

```
rk3588_s:/ # ls /sys/class/drm/
card0          card0-DP-2    card0-HDMI-A-1  card0-writeback-1  renderD128  version
card0-DP-1     card0-DSI-1  card0-HDMI-A-2  card1               renderD129
```

以 card0-DP-1 为例，其目录下有如下内容：

```
rk3588_s:/ # ls /sys/class/drm/card0-DP-1/
device dpms edid enabled modes power status subsystem uevent
```

enable 查看使能状态：

```
rk3588_s:/ # cat /sys/class/drm/card0-DP-1/enabled
disabled
```

status 查看连接状态：

```
rk3588_s:/ # cat /sys/class/drm/card0-DP-1/status
disconnected
```

modes 设备支持的分辨率列表：

```
rk3588_s:/ # cat /sys/class/drm/card0-DP-1/modes
1440x900
1280x1024
1280x1024
```

```
1280x960
1152x864
1024x768
1024x768
832x624
800x600
800x600
640x480
640x480
720x400
```

edid 设备的 EDID, 通过如下命令保存:

```
rk3588_s:/ # cat /sys/class/drm/card0-DP-1/edid > /data/edid.bin
```

强制使能/禁用 DP

```
#强制禁用 DP
rk3588_s:/ # echo off > /sys/class/drm/card0-DP-1/status
#强制使能 DP
rk3588_s:/ # echo on > /sys/class/drm/card0-DP-1/status
#恢复热插拔检测
rk3588_s:/ # echo detect > /sys/class/drm/card0-DP-1/status
```

DPCP 读写

DPCP 通过 AUX_CH 读写, 读写节点的实现在

```
/drivers/gpu/drm/drm_dp_aux_dev.c
```

使用此功能前, 先确认相关的编译选项是否已经配置:

```
CONFIG_DRM_DP_AUX_CHARDEV=y
```

读取 DPCD 如下:

```
#if 后面为 aux 节点, 当注册两个 DP 接口时, 会有 /dev/drm_dp_aux0 和 /dev/drm_dp_aux1
#skip 值为起始的 DPCD 寄存器地址
#count 值为要读取的 DPCD 寄存器的数量
dd if=/dev/drm_dp_aux0 bs=1 skip=$((0x00200)) count=2 status=none | od -tx1
#如下为读取地址为 0x00200 开始的 2 个 DPCD 寄存器的内容
rk3588_s:/ # dd if=/dev/drm_dp_aux1 bs=1 skip=$((0x00200)) count=2 status=none |
od -tx1
0000000 01 00
0000002
```

写入 DPCD 寄存器:

```
#echo 后为要写入的值，如下为需要写入两个 16 进制的值，分别为 0x0a, 0x80
#of 后面为 aux 节点，当注册两个 DP 接口时，会有 /dev/drm_dp_aux0 和 /dev/drm_dp_aux1
#seek 后为起始的 DPCD 寄存器地址
#count 值为要写入的 DPCD 寄存器的数量
#如下指令为把 0x0a 和 0x80 两个值写入 0x100 起始的两个 DPCD 寄存器处
echo -e -n "\x0a\x80" | dd of=/dev/drm_dp_aux0 bs=1 seek=$((0x100)) count=2
status=none
```

Type-C 接口 Debug

Type-C 接口的 HPD 检测部分由 PD 芯片完成，这部分的软件流程主要由 TCPM 的框架完成，TCPM 检测这部分 log 可以由以下方式获取：

```
rk3588_s:/ # ls -l /sys/kernel/debug/usb/
total 0
-r--r--r-- 1 root root 0 1970-01-01 00:00 devices
drwxr-xr-x 18 root root 0 1970-01-01 00:00 fc000000.usb
drwxr-xr-x 2 root root 0 1970-01-01 00:00 fc400000.usb
-r--r--r-- 1 root root 0 1970-01-01 00:00 fusb302-2-0022
drwxr-xr-x 4 root root 0 1970-01-01 00:00 ohci
-r--r--r-- 1 root root 0 1970-01-01 00:00 tcpm-2-0022
drwxr-xr-x 2 root root 0 1970-01-01 00:00 usbmon
drwxr-xr-x 2 root root 0 2021-01-01 12:00 uvcvideo
drwxr-xr-x 3 root root 0 1970-01-01 00:00 xhci
```

在 /sys/kernel/debug/usb/ 目录中，可以看到 fusb302-2-0022 和 tcpm-2-0022，其中 fusb302-2-0022 为 PD 芯片的节点，tcpm-2-0022 为 TCPM 框架的节点，获取 TCPM 框架的 log 命令如下：

```
cat /sys/kernel/debug/usb/tcpm-2-0022
```

Note: tcpm-2-0022, 中间的 2 为对应的 i2c 总线，最后的 0022 为 PD 芯片对应的 i2c 地址

获取 PD 芯片的 log 如下：

```
cat /sys/kernel/debug/usb/fusb302-2-0022
```

Note: fusb302-2-0022, 中间的 2 为对应的 i2c 总线，最后的 0022 为 PD 芯片对应的 i2c 地址，上述节点对应 fusb302 芯片，不同芯片节点名称不一样。

除了 log 外，在 Type-C 节点下还可以获取其他的一些信息，Type-C 节点路径如下：

```
console:/ # ls /sys/class/typec
port0 port0-partner
```

port0 表示 SoC 这端的 Type-C 接口，port0-partner 表示通过 Type-C 连接设备后设备端的节点目录。

Type-C 连接的正反面信息：

```
cat /sys/class/typec/port0/orientation
reverse
```

port0-partner 下可能有多个目录，对于 DP Alt Mode 对应的目录，其对应的目录先会有 displayport 子目录，并且 svid 的值为 0xff01。

```
ls -l /sys/class/typec/port0-partner/port0-partner.0/
total 0
-r--r--r-- 1 root root 4096 2022-04-14 14:50 active
-r--r--r-- 1 root root 4096 2022-04-14 14:50 description
drwxr-xr-x 2 root root 0 2022-04-14 14:50 displayport
lrwxrwxrwx 1 root root 0 2022-04-14 14:50 driver ->
../../../../../../../../../../../../bus/typec/drivers/typec_displayport
-r--r--r-- 1 root root 4096 2022-04-14 14:50 mode
drwxr-xr-x 2 root root 0 2022-04-14 14:50 mode1
lrwxrwxrwx 1 root root 0 2022-04-14 14:50 port -> ../../port0.0
drwxr-xr-x 2 root root 0 2022-04-14 14:50 power
lrwxrwxrwx 1 root root 0 2022-04-14 14:50 subsystem ->
../../../../../../../../../../../../bus/typec
-r--r--r-- 1 root root 4096 2022-04-14 14:50 svid
-rw-r--r-- 1 root root 4096 2022-04-14 14:50 uevent
-r--r--r-- 1 root root 4096 2022-04-14 14:50 vdo
```

```
cat /sys/class/typec/port0-partner/port0-partner.0/svid
ff01
```

获取当前的 pin assignment 信息：

```
cat /sys/class/typec/port0-partner/port0-partner.0/displayport/pin_assignment
C [D]
#当前连接的设备支出 C assignment 和 D assignment，目前配置的是 D assignment
```

Note: 以上描述的是使用TCPM框架的 PD 芯片的相关信息获取，若搭配使用的 PD 芯片不是基于 TCPM 框架，请同 PD 芯片 vendor 确认相关信息。

查看 DP 寄存器

RK3588 DP 相关寄存器：

```
#dp0 控制器
cat /sys/kernel/debug/regmap/fde50000.dp/registers
#usbdp phy0
cmn_reg0000 - cmn_reg015D:
io -4 -r -l 1400 0xfed88000
trsv_reg0200 - trsv_reg03C3:
io -4 -r -l 1808 0xfed88800
trsv_reg0400 - trsv_reg0435:
io -4 -r -l 212 0xfed89000
trsv_reg0600 - trsv_reg07C3:
io -4 -r -l 1808 0xfed89800
trsv_reg0800 - trsv_reg0835:
io -4 -r -l 212 0xfed8A000
#dp1 控制器
cat /sys/kernel/debug/regmap/fde60000.dp/registers
#usbdp phy1
cmn_reg0000 - cmn_reg015D:
io -4 -r -l 1400 0xfed98000
trsv_reg0200 - trsv_reg03C3:
io -4 -r -l 1808 0xfed98800
trsv_reg0400 - trsv_reg0435:
io -4 -r -l 212 0xfed99000
```

```
trsv_reg0600 - trsv_reg07C3:
io -4 -r -l 1808 0xfed99800
trsv_reg0800 - trsv_reg0835:
io -4 -r -l 212 0xfed9A000
# vo0_grf
cat /sys/kernel/debug/regmap/dummy-syscon@fd5a6000/registers
```

RK3576 DP 相关寄存器:

```
#dp 控制器
cat /sys/kernel/debug/regmap/27e40000.dp/registers
#usbdp phy
cmn_reg0000 - cmn_reg015D:
io -4 -r -l 1400 0x2b018000
trsv_reg0200 - trsv_reg03C3:
io -4 -r -l 1808 0x2b018800
trsv_reg0400 - trsv_reg0435:
io -4 -r -l 212 0x2b019000
trsv_reg0600 - trsv_reg07C3:
io -4 -r -l 1808 0x2b019800
trsv_reg0800 - trsv_reg0835:
io -4 -r -l 212 0x2b01a000
# vo1_grf
cat /sys/kernel/debug/regmap/dummy-syscon@0x0000000026036000/registers
```

Note: 需要在连接 DP 显示器并正常显示时，才能 dump phy 寄存器。

查看 VOP 状态

通过如下指令即可查询 VOP 的状态:

```
cat /sys/kernel/debug/dri/0/summary
```

获取的 VOP 状态如下图:

```
1|rk3588 s:/ # cat /sys/kernel/debug/dri/0/summary
Video Port0: DISABLED
Video Port1: DISABLED
Video Port2: DISABLED
Video Port3: ACTIVE
Connector: DSI-1
  bus_format[100a]: RGB888_1X24
  overlay_mode[0] output_mode[0] color_space[0], eotf:0
Display mode: 1080x1920p60
  clk[132000] real_clk[132000] type[48] flag[a]
  H: 1080 1095 1099 1129
  V: 1920 1935 1937 1952
Cluster3-win0: ACTIVE
  win_id: 6
  format: AB24 little-endian (0x34324241)[AFBC] SDR[0] color_space[0] glb_alpha[0xff]
  rotate: xmirror: 0 ymirror: 0 rotate_90: 0 rotate_270: 0
  csc: y2r[0] r2y[0] csc mode[0]
  zpos: 0
  src: pos[0, 0] rect[1080 x 1920]
  dst: pos[0, 0] rect[1080 x 1920]
  buf[0]: addr: 0x00000000f1777000 pitch: 4352 offset: 0
```

Annotations in the image:

- vop状态**: Points to the Video Port status lines.
- connector 状态**: Points to the Connector and bus_format lines.
- timing 信息**: Points to the Display mode and timing parameters.
- 图层信息**: Points to the Cluster3-win0 and its associated parameters.

Video Portx: 表示当前的 Video Port 的状态

Connector: Video Port 当前连接的输出接口

Display mode: Video Port 当前输出时序

Clusterx-winx(Esmartx-winx): 图层信息

在 Kernel 6.1 及以上版本，获取的信息如下：

```
root@linaro-alip:/# cat /sys/kernel/debug/dri/0/summary
Video Port0: ACTIVE
    Connector:DP-2          Encoder: DP-MST 0
        bus_format[100a]: RGB888_1X24
        overlay_mode[0] output_mode[f] SDR[0] color-encoding[BT.709] color-
range[Limited]
    Display mode: 1280x720p60
        clk[74250] real_clk[74250] type[0] flag[5]
        H: 1280 1390 1430 1650
        V: 720 725 730 750
    Esmart0-win0: ACTIVE
        win_id: 0
        format: XR24 little-endian (0x34325258) pixel_blend_mode[0]
glb_alpha[0xff]
    color: SDR[0] color-encoding[BT.601] color-range[Limited]
    rotate: xmirror: 0 ymirror: 0 rotate_90: 0 rotate_270: 0
    csc: y2r[0] r2y[0] csc mode[0]
    zpos: 0
    src: pos[0, 0] rect[1280 x 720]
    dst: pos[0, 0] rect[1280 x 720]
    buf[0]: addr: 0x0000000001017000 pitch: 5120 offset: 0
Video Port1: ACTIVE
    Connector:DP-5          Encoder: DP-MST 1
        bus_format[100a]: RGB888_1X24
        overlay_mode[0] output_mode[f] SDR[0] color-encoding[BT.709] color-
range[Limited]
    Display mode: 1280x720p60
        clk[74250] real_clk[74250] type[0] flag[5]
        H: 1280 1390 1430 1650
        V: 720 725 730 750
    Esmart1-win0: ACTIVE
        win_id: 1
        format: XR24 little-endian (0x34325258) pixel_blend_mode[0]
glb_alpha[0xff]
    color: SDR[0] color-encoding[BT.601] color-range[Limited]
    rotate: xmirror: 0 ymirror: 0 rotate_90: 0 rotate_270: 0
    csc: y2r[0] r2y[0] csc mode[0]
    zpos: 1
    src: pos[0, 0] rect[1280 x 720]
    dst: pos[0, 0] rect[1280 x 720]
    buf[0]: addr: 0x00000000001e1000 pitch: 5120 offset: 0
Video Port2: ACTIVE
    Connector:DP-6          Encoder: DP-MST 2
        bus_format[100a]: RGB888_1X24
        overlay_mode[0] output_mode[f] SDR[0] color-encoding[BT.709] color-
range[Limited]
    Display mode: 1280x720p60
        clk[74250] real_clk[74250] type[0] flag[5]
        H: 1280 1390 1430 1650
        V: 720 725 730 750
    Esmart2-win0: ACTIVE
        win_id: 2
```

```

        format: XR24 little-endian (0x34325258) pixel_blend_mode[0]
glb_alpha[0xff]
        color: SDR[0] color-encoding[BT.601] color-range[Limited]
        rotate: xmirror: 0 ymirror: 0 rotate_90: 0 rotate_270: 0
        csc: y2r[0] r2y[0] csc mode[0]
        zpos: 2
        src: pos[0, 0] rect[1280 x 720]
        dst: pos[0, 0] rect[1280 x 720]
        buf[0]: addr: 0x00000000015e2000 pitch: 5120 offset: 0

```

可以看到，Summary 多了 Encoder 信息。

在 RK3576 下，注册了 1 个 SST 模式下的 Encoder 和 3 个 MST 模式下的 Encoder，其中 3 个 MST Encoder 和 DP 3 路的显示数据流对应关系如下：

```

DP-MST 0 --> Stream-0
DP-MST 1 --> Stream-1
DP-MST 2 --> Stream-2

```

当 Encoder 为 MST Encoder 时，表示 DP 工作在 MST 模式下，如果 DP Connector 对应的 Encoder 为 TMDS-xxx，表示 DP 工作在 SST 模式下，举例如下：

```

root@linaro-alip:/# cat /sys/kernel/debug/dri/0/summary
Video Port0: ACTIVE
    Connector:DP-1      Encoder: TMDS-184
        bus_format[1018]: RGB101010_1X30
        overlay_mode[0] output_mode[f] SDR[0] color-encoding[BT.709] color-
range[Limited]
    Display mode: 1280x720p60
        clk[74250] real_clk[74250] type[0] flag[5]
        H: 1280 1390 1430 1650
        V: 720 725 730 750
    Esmart0-win0: ACTIVE
        win_id: 0
        format: XR24 little-endian (0x34325258) pixel_blend_mode[0]
glb_alpha[0xff]
        color: SDR[0] color-encoding[BT.601] color-range[Limited]
        rotate: xmirror: 0 ymirror: 0 rotate_90: 0 rotate_270: 0
        csc: y2r[0] r2y[0] csc mode[0]
        zpos: 0
        src: pos[0, 0] rect[1280 x 720]
        dst: pos[0, 0] rect[1280 x 720]
        buf[0]: addr: 0x000000000090f000 pitch: 5120 offset: 0
Video Port1: DISABLED
Video Port2: DISABLED

```

查看当前显示时钟

获取整个时钟树：

```
cat /sys/kernel/debug/clk/clk_summary
```

获取 dp aux 16M clk:

```
cat /sys/kernel/debug/clock/clock_summary | grep -e "clock_aux16m"
```

获取 vop dclk:

```
cat /sys/kernel/debug/clock/clock_summary | grep -e "dclk"
```

调整 DRM log 等级

DRM 有如下的打印等级定义，可以根据需要，动态的打开对应的 log 打印：

```
enum drm_debug_category {
    /**
     * @DRM_UT_CORE: Used in the generic drm code: drm_ioctl.c, drm_mm.c,
     * drm_memory.c, ...
     */
    DRM_UT_CORE = 0x01,
    /**
     * @DRM_UT_DRIVER: Used in the vendor specific part of the driver: i915,
     * radeon, ... macro.
     */
    DRM_UT_DRIVER = 0x02,
    /**
     * @DRM_UT_KMS: Used in the modesetting code.
     */
    DRM_UT_KMS = 0x04,
    /**
     * @DRM_UT_PRIME: Used in the prime code.
     */
    DRM_UT_PRIME = 0x08,
    /**
     * @DRM_UT_ATOMIC: Used in the atomic code.
     */
    DRM_UT_ATOMIC = 0x10,
    /**
     * @DRM_UT_VBL: Used for verbose debug message in the vblank code.
     */
    DRM_UT_VBL = 0x20,
    /**
     * @DRM_UT_STATE: Used for verbose atomic state debugging.
     */
    DRM_UT_STATE = 0x40,
    /**
     * @DRM_UT_LEASE: Used in the lease code.
     */
    DRM_UT_LEASE = 0x80,
    /**
     * @DRM_UT_DP: Used in the DP code.
     */
    DRM_UT_DP = 0x100,
    /**
     * @DRM_UT_DRMRES: Used in the drm managed resources code.
     */
    DRM_UT_DRMRES = 0x200,
};
```

DP 接口排查问题时, commit 异常的问题, 目前比较多的是打开 ATOMIC, 如下:

```
echo 0x10 > /sys/module/drm/parameters/debug
```

如果要打印 DPCD 的读写 log, 输入如下命令:

```
echo 0x100 > /sys/module/drm/parameters/debug
```

查看 DP MST 信息

支持 MST 功能的 DP 接口, 默认都会注册一个 SST Connector, 对应的 debugfs 路径是固定的。MST Connector 则是在插拔设备时动态的注册和注销。因此把查看 DP MST 信息的节点放在 DP 接口注册的 SST Connector 的 debugfs 路径下。如 RK3576 命令如下:

```
root@linaro-alip:/# cat /sys/kernel/debug/dri/0/DP-1/dp_mst_info
    mstb - [000000003a17fc25]: num_ports: 4
    port 3 - [00000000099bbb63] (output - SST SINK): ddps: 1, ldps: 0, sdp:
1/1, fec: false, conn: 00000000c74ff83b
    port 2 - [00000000505d66cf] (output - MST BRANCHING): ddps: 1, ldps: 0,
sdp: 0/0, fec: true, conn: 00000000b57a2623
        mstb - [0000000090afd0f3]: num_ports: 3
        port 1 - [00000000072aa867] (output - NONE): ddps: 0, ldps: 0,
sdp: 0/0, fec: false, conn: 000000005ba2b268
        port 8 - [0000000046d78f33] (output - SST SINK): ddps: 1, ldps:
0, sdp: 1/2, fec: true, conn: 000000002668a7af
        port 0 - [00000000a70f650f] (input - NONE): ddps: 1, ldps: 0,
sdp: 0/0, fec: false, conn: 0000000000000000
        port 1 - [000000003fd2f64a] (output - SST SINK): ddps: 1, ldps: 0, sdp:
1/1, fec: false, conn: 000000005ff7f122
        port 0 - [00000000c8a5769d] (input - NONE): ddps: 1, ldps: 0, sdp: 0/0,
fec: false, conn: 0000000000000000

*** Atomic state info ***
payload_mask: 7, max_payloads: 3, start_slot: 1, pbn_div: 60

| idx | port | vcpi | slots | pbn | dsc |      sink name      |
|----|-----|-----|-----|-----|-----|-----|
|   1 |    1 |    1 | 06 - 10 | 266 |   N |          U27U2D     |
|   2 |    8 |    2 | 11 - 15 | 266 |   N |        DELL U2723QE |
|   3 |    3 |    3 | 01 - 05 | 266 |   N |          U28E590     |

*** DPCD Info ***
dpcd: 14 1e c4 81 01 11 01 83 2a 3f 04 00 00 00 84
faux/mst: 00 01
mst ctrl: 07
branch oui: 90cc24 devid: SYNAS revision: hw: 1.0 sw: 5.5
payload table: 03 03 03 03 03 03 01 01 01 01 01 02 02 02 02 02 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

*** Connector path info ***
connector name | connector path
DP-2           mst:185-1
DP-3           mst:185-2
DP-6           mst:185-3
DP-5           mst:185-2-8
```

MST Port Info

第一部分为设备连接拓扑结构。

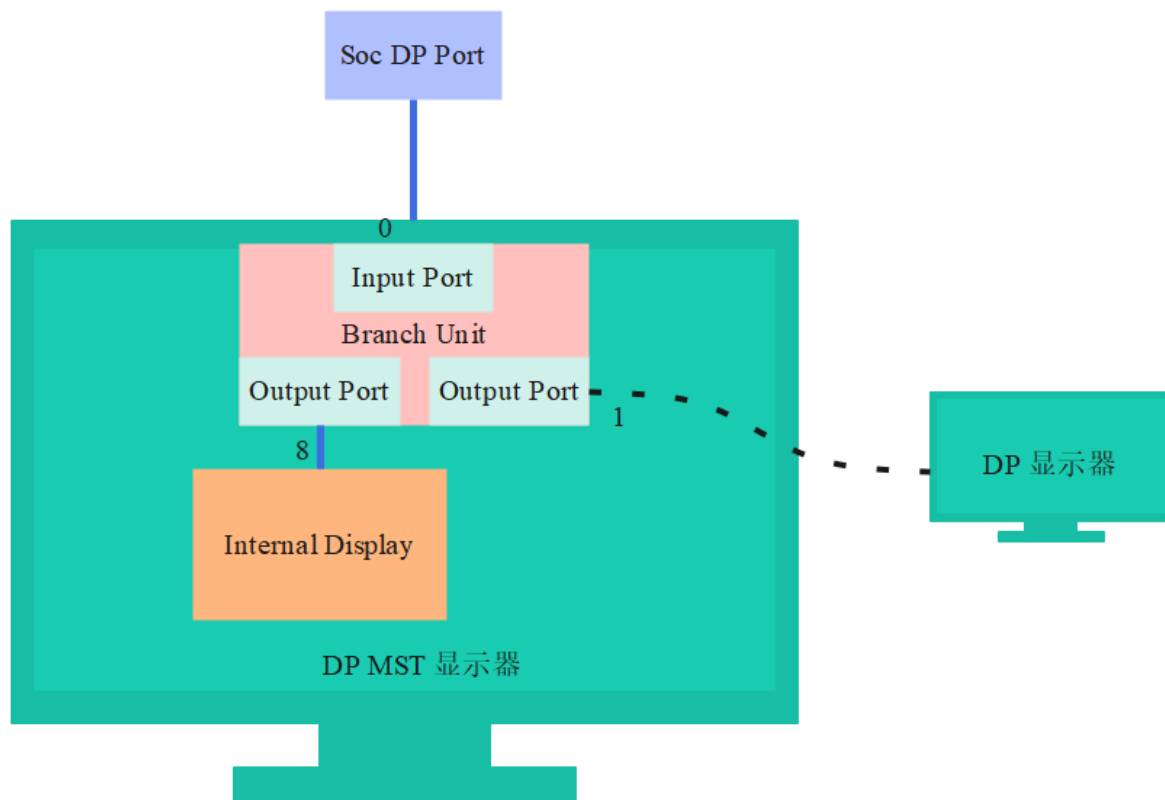
```
mstb - [000000003a17fc25]: num_ports: 4
  port 3 - [00000000099bbb63] (output - SST SINK): ddps: 1, ldps: 0, sdp:
1/1, fec: false, conn: 00000000c74ff83b
  port 2 - [00000000505d66cf] (output - MST BRANCHING): ddps: 1, ldps: 0,
sdp: 0/0, fec: true, conn: 00000000b57a2623
    mstb - [0000000090afd0f3]: num_ports: 3
      port 1 - [00000000072aa867] (output - NONE): ddps: 0, ldps: 0,
sdp: 0/0, fec: false, conn: 000000005ba2b268
      port 8 - [0000000046d78f33] (output - SST SINK): ddps: 1, ldps:
0, sdp: 1/2, fec: true, conn: 000000002668a7af
      port 0 - [00000000a70f650f] (input - NONE): ddps: 1, ldps: 0,
sdp: 0/0, fec: false, conn: 0000000000000000
      port 1 - [000000003fd2f64a] (output - SST SINK): ddps: 1, ldps: 0, sdp:
1/1, fec: false, conn: 000000005ff7f122
      port 0 - [00000000c8a5769d] (input - NONE): ddps: 1, ldps: 0, sdp: 0/0,
fec: false, conn: 0000000000000000
```

回顾 1.1.3 小节，RK3576 要工作在 MST 模式下，需要连接 MST HUB 或 MST 显示。HUB 或显示器的每个输入输出端口都有独立的编号，如下为 1 个输出口，3 个输出端口的 MST HUB Port 编号：

上图的输入端口和输出端口均为可以和外部其他设备连接的物理接口，按 DP 协议，物理端口可以使用的编号为 0~7，并且输入端口的编号要比输出口小。一般 MST HUB 都是从 0 开始编号。如上的 MST HUB，总共有 4 个 Port，其中 Port 0 为 Input Port，Port 1/2/3 为 Output Port，并且 Port 2 连接一个 SST 显示，获取的拓扑结构信息如下：

```
mstb - [00000000c8269fca]: num_ports: 4
  port 3 - [0000000007690f7c] (output - NONE): ddps: 0, ldps: 0, sdp: 0/0,
fec: false, conn: 00000000d0584c6f
  port 2 - [000000000812cc57a] (output - SST SINK): ddps: 1, ldps: 0, sdp:
1/1, fec: false, conn: 000000001ef4a070
  port 1 - [00000000efb170aa] (output - NONE): ddps: 0, ldps: 0, sdp: 0/0,
fec: false, conn: 000000001455081f
  port 0 - [0000000005aa73543] (input - NONE): ddps: 1, ldps: 0, sdp: 0/0,
fec: false, conn: 0000000000000000
```

如果接的是 MST 显示器，一般 MST 显示器有一个 DP 输入和一个 DP 输出，如下图所示：

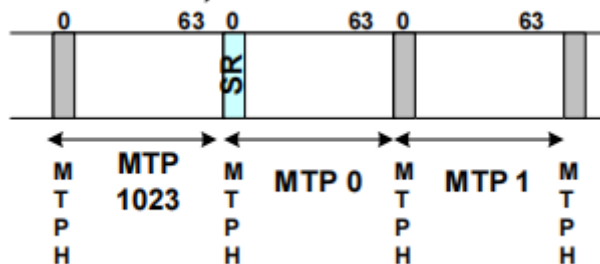


在上图中 Port0 为 Input Port, Port 1 为 Output port。Port8 为显示器内部的 Output port。MST 显示器这种内部的 Port 是 Logical Port, Port 编号从 8 到 15。当上述的 MST 显示器不串接其他显示器时, 获取的拓扑结构信息如下:

```
mstb - [0000000019f71241]: num_ports: 3
port 1 - [0000000095fb1fc0] (output - NONE): ddps: 0, ldps: 0, sdp: 0/0,
fec: false, conn: 000000004d03ffa2
port 8 - [00000000a66da753] (output - SST SINK): ddps: 1, ldps: 0, sdp:
1/2, fec: true, conn: 000000009a417589
port 0 - [000000009a0122a8] (input - NONE): ddps: 1, ldps: 0, sdp: 0/0,
fec: false, conn: 0000000000000000
```

Atomic state info

DP 在 MST 模式下, 基础的数据包格式称为 Multi-Stream Transport Packet (MTP), 64 个 link symbol 组成一个 MTP, 这 64 个 link symbol, 也称为 64 个 time slots, 编号从 0 到 63, 其中 time slot 0 为 MTPH, 其他的 time slot 可以用于传输各个显示通路数据。MTP 格式如下所示。



如下表的 payload info, port 为 MST 设备的 port 编号, vcpi 为显示通路的编号, slots 为对应显示通路占用的 time slot, pbn 为对应显示通路占用的带宽, sink name 为显示器名称。

idx	port	vcpi	slots	pbn	dsc	sink name
1	1	1 06 - 10	266	N		U27U2D
2	8	2 11 - 15	266	N		DELL U2723QE
3	3	3 01 - 05	266	N		U28E590

DPCD Info

这部分为获取的 DPCD 信息，主要关注 payload table, 记录了 time slots 的详细分配情况。

```
*** DPCD Info ***
dpcd: 14 1e c4 81 01 11 01 83 2a 3f 04 00 00 00 84
faux/mst: 00 01
mst ctrl: 07
branch oui: 90cc24 devid: SYNAS revision: hw: 1.0 sw: 5.5
payload table: 03 03 03 03 03 03 01 01 01 01 01 02 02 02 02 02 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Connector Path Info

connector name 为动态注册的 DP connector，connector path 为对应的路径。

```
*** Connector path info ***
connector name | connector path
DP-2          | mst:185-1
DP-3          | mst:185-2
DP-6          | mst:185-3
DP-5          | mst:185-2-8
DP-7          | mst:185-2-1
```

FAQ

插入 DP 无显示或显示异常

首先查看是否有如下 log:

```
[ 14.857002] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_enable] Update
mode to 1920x1080p60, type: 10(if:200) for vp2 dclk: 148500000
[ 14.857149] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_enable]
dclk_out2 div: 2 dclk_core2 div: 2
[ 14.857868] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_enable] set
dclk_vop2 to 148500000, get 148500000
[ 14.872406] dw-dp fde50000.dp: full-training link: 2 lanes at 5400 MHz
[ 14.893269] dw-dp fde50000.dp: clock recovery succeeded
[ 14.899797] dw-dp fde50000.dp: channel equalization succeeded
```

DP Link Training 成功

出现如上 log 时，说明已经检测到 DP 连接，并且 DP 已经成功 link training 并输出图像，出现无显示或显示异常的原因可能如下：

1. dclk 分的不准

可以看如下的 log 如下，请求的 dclk 为 25.175MHz，实际分到的为 20MHz，出现这种 clk 分配问题，抓取完整的 log 并提供时钟树 log 供进一步分析。

```
[ 268.733803] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_disable] Crtc atomic disable vp2
[ 268.759178] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_enable] Update mode to 640x480p60, type: 10(if:200) for vp2 dclk: 25175000
[ 268.759447] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_enable] dclk_out2 div: 2 dclk_core2 div: 2
[ 268.759665] rockchip_rk3588_pll_set_rate: Invalid rate : 25175000 for pll clk pll_v0pll
[ 268.759715] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_enable] set dclk_vop2 to 25175000, get 20000000
[ 268.775591] dw-dp fde50000.dp: full-training link: 4 lanes at 2700 MHz
[ 268.790059] dw-dp fde50000.dp: clock recovery succeeded
[ 268.795376] dw-dp fde50000.dp: channel equalization succeeded
```

2. 未分配图层

userspace 未分配图层，执行 `cat /sys/kernel/debug/dri/0/summary`，如果获取的信息如下所示，即没有图层信息，需要从 userspace 部分进一步分析。

```
rk3588_s:/ # cat /sys/kernel/debug/dri/0/summary
Video Port0: DISABLED
Video Port1: DISABLED
Video Port2: DISABLED
Video Port3: ACTIVE
  Connector: DSI-1
    bus_format[100a]: RGB888_1x24
    overlay_mode[0] output_mode[0] color_space[0], eotf:0
  Display mode: 1080x1920p60
    clk[132000] real_clk[132000] type[48] flag[a]
    H: 1080 1095 1099 1129
    V: 1920 1935 1937 1952
```

DP connected

如果未出现本小节开头出现的 log，先获取 DP 的连接状态如下：

```
cat /sys/class/drm/card0-DP-1/status
```

如果 DP 是 connected 状态，先分析 log 是否有异常报错，有异常报错从异常处分析，如果 log 无异常，打开 DRM 的 ATOMIC log 等级复现，确认是否在 drm atomic commit 中途异常返回，可能 atomic check 的某个环节 failed。

DP disconnected

对于 DP 标准口输出，确认 HPD 配置是否正确以及硬件连接是否正常，对于 Type-C 接口，参考后文的 Type-C 接口连接异常分析。

Type-C 接口连接异常

这里的 Type-C 接口连接异常指的是 CC 阶段和 PD 阶段即出现异常，首先获取 DP 的连接状态：

```
cat /sys/class/drm/card0-DP-1/status
```

连接异常时这里获取到状态都是 disconnected。

通过 tcpm 的调试节点获取 tcpm 的 log:

```
cat /sys/kernel/debug/usb/tcpm-2-0022
```

正常连接的 log 如下:

```
[ 25.026952] AMS DISCOVER_IDENTITY start
[ 25.026967] PD TX, header: 0x176f
[ 25.035314] PD TX complete, status: 0
[ 25.042866] PD RX, header: 0x524f [1]
[ 25.042880] Rx VDM cmd 0xff00a041 type 1 cmd 1 len 5
[ 25.042894] AMS DISCOVER_IDENTITY finished
[ 25.042898] cc:=4
[ 25.052343] Identity: 04e8:a020.0212
[ 25.052364] AMS DISCOVER_SVIDS start
[ 25.052372] PD TX, header: 0x196f
[ 25.061314] PD TX complete, status: 0
[ 25.067667] PD RX, header: 0x344f [1]
[ 25.067680] Rx VDM cmd 0xff00a042 type 1 cmd 2 len 3
[ 25.067695] AMS DISCOVER_SVIDS finished
[ 25.067705] cc:=4
[ 25.077097] SVID 1: 0xff01
[ 25.077114] SVID 2: 0x4e8
[ 25.077129] AMS DISCOVER_MODES start
[ 25.077135] PD TX, header: 0x1b6f
[ 25.086092] PD TX complete, status: 0
[ 25.092224] PD RX, header: 0x264f [1]
[ 25.092237] Rx VDM cmd 0xff01a043 type 1 cmd 3 len 2
[ 25.092252] AMS DISCOVER_MODES finished
[ 25.092256] cc:=4
[ 25.101432] Alternate mode 0: SVID 0xff01, VDO 1: 0x00000c05
[ 25.101517] AMS DISCOVER_MODES start
[ 25.101526] PD TX, header: 0x1d6f
[ 25.109717] PD TX complete, status: 0
[ 25.114919] PD RX, header: 0x284f [1]
[ 25.114937] Rx VDM cmd 0x4e8a043 type 1 cmd 3 len 2
[ 25.114951] AMS DISCOVER_MODES finished
[ 25.114956] cc:=4
[ 25.124604] Alternate mode 1: SVID 0x04e8, VDO 1: 0x00000001
[ 25.125676] AMS DFP_TO_UFP_ENTER_MODE start
[ 25.125686] PD TX, header: 0x1f6f
[ 25.134560] PD TX complete, status: 0
[ 25.137903] PD RX, header: 0x1a4f [1]
[ 25.137917] Rx VDM cmd 0xff01a144 type 1 cmd 4 len 1
[ 25.137930] AMS DFP_TO_UFP_ENTER_MODE finished
[ 25.137936] cc:=4
[ 25.145828] AMS STRUCTURED_VDMS start
[ 25.145836] PD TX, header: 0x216f
[ 25.154942] PD TX complete, status: 0
[ 25.161111] PD RX, header: 0x2c4f [1]
[ 25.161125] Rx VDM cmd 0xff01a150 type 1 cmd 16 len 2 //STATUS UPDATE
[ 25.161138] AMS STRUCTURED_VDMS finished
[ 25.161142] cc:=4
[ 25.171888] AMS STRUCTURED_VDMS start
[ 25.171911] PD TX, header: 0x236f
```

```
[ 25.182016] PD TX complete, status: 0
[ 25.185550] PD RX, header: 0x1e4f [1]
[ 25.185563] Rx VDM cmd 0xff01a151 type 1 cmd 17 len 1 //CONFIGURATION
[ 25.185577] AMS STRUCTURED_VDMS finished
[ 25.185581] cc:=4
[ 26.392673] PD RX, header: 0x204f [1]
[ 26.392687] Rx VDM cmd 0xff018106 type 0 cmd 6 len 2 //ATTENTION
```

从 log 看，正常的完整流程会有 DISCOVER_IDENTITY、DISCOVER_MODES、DFP_TO_UFP_ENTER_MODE、STATUS UPDATE、CONFIGURATION、ATTENTION等命令的交互，如果没有以上的交互流程，即说明 PD 的交互出现了异常。

如果上述的流程出现异常，可以提高 PD 芯片的 I2C 速率进一步测试，如果仍无法解决问题，需要提供完整的 tcpm 的 log 和 PD 芯片的 log 进一步分析。

AUX_CH 异常

AUX_CH 异常时，会导致读写 DPCD 和读 EDID 出现异常，log 中可能会出现如下报错：

```
[ 1368.952182] dw-dp fde50000.dp: failed to probe DP link: -110
```

如无法确认，可以打开 DRM debug log 的如下开关

```
echo 0x100 > /sys/module/drm/parameters/debug
```

通过 dmesg 获取 DPCP 读写的 log, 正常的 DPCP 读写的 log 如下，ret 为 0：

```
[ 6329.554538] rockchip-drm display-subsystem: [drm:drm_dp_dpcd_probe]
fde50000.dp: 0x00000 AUX -> (ret= 1) 12
[ 6329.554939] rockchip-drm display-subsystem: [drm:drm_dp_dpcd_read]
fde50000.dp: 0x00000 AUX -> (ret= 15) 12 14 c2 81 01 01 01 81 02 02 06 00 00 00
81
[ 6329.555383] rockchip-drm display-subsystem: [drm:drm_dp_dpcd_probe]
fde50000.dp: 0x00000 AUX -> (ret= 1) 12
```

AUX_CH 异常时，ret 值为异常类型值，如下：

```
[ 31.116976] rockchip-drm display-subsystem: [drm:drm_dp_dpcd_probe]
fde50000.dp: 0x00000 AUX -> (ret=-110)
```

AUX_CH 可能的异常原因有如下几点。

aux16m clk 值异常

aux16m clk rate 异常，aux16m clk rate 的 parent clk 是 GPLL，默认设置的 GPLL 为 1188MHz，aux16m clk 的默认值如下：

```
root@RK3588:/# cat /sys/kernel/debug/clock/clock_summary | grep "aux16m"
      clk_aux16m_1          1          2          0    15840000          0
0    50000
      clk_aux16m_0          1          2          0    15840000          0
0    50000
```

如果获取到 GPLL 不为 1188MHz, 并且 aux16m clk 也非默认值, 请先检查是否有在 SDK 的基础上对 /drivers/clk/rockchip/ 中的文件做了改动, 或对 VOP 的 DCLK parent 进行了重新配置。

phy power on/off 流程异常

这种异常一般出现在 Type-C 接口, USB 和 Type-C 共用 phy 的场景, 如果出现 Type-C 一面可以正常工作, 换另一面插入报错, 有可能拔出的时候 usb phy 没有 eixt, 导致重新插入时 PHY 未重新初始化, 可以添加 log 先确认 USB 插拔时是否有执行 phy power on/off, 以及 PHY 重新插入另一面是 PHY 是否有重新初始化。log 添加可以参考如下 patch。

```
diff --git a/drivers/phy/rockchip/phy-rockchip-usbdp.c
b/drivers/phy/rockchip/phy-rockchip-usbdp.c
index c6fc4a2aa558..6b35e12f40aa 100644
--- a/drivers/phy/rockchip/phy-rockchip-usbdp.c
+++ b/drivers/phy/rockchip/phy-rockchip-usbdp.c
@@ -823,6 +823,7 @@ static int udphy_power_on(struct rockchip_udphy *udphy, u8
mode)
{
    int ret;

+    dev_info(udphy->dev, "%s status:%x, mode:%x\n", __func__, udphy->status,
mode);
    if (!(udphy->mode & mode)) {
        dev_info(udphy->dev, "mode 0x%02x is not support\n", mode);
        return 0;
@@ -859,6 +860,7 @@ static int udphy_power_off(struct rockchip_udphy *udphy, u8
mode)
{
    int ret;

+    dev_info(udphy->dev, "%s status:%x, mode:%x\n", __func__, udphy->status,
mode);
    if (!(udphy->mode & mode)) {
        dev_info(udphy->dev, "mode 0x%02x is not support\n", mode);
        return 0;
@@ -883,6 +885,7 @@ static int rockchip_dp_phy_power_on(struct phy *phy)
struct rockchip_udphy *udphy = phy_get_drvdata(phy);
int ret, dp_lanes;

+    dev_info(udphy->dev, "%s\n", __func__);
    mutex_lock(&udphy->mutex);

    dp_lanes = udphy_dplane_get(udphy);
@@ -914,6 +917,7 @@ static int rockchip_dp_phy_power_off(struct phy *phy)
struct rockchip_udphy *udphy = phy_get_drvdata(phy);
int ret;

+    dev_info(udphy->dev, "%s\n", __func__);
    mutex_lock(&udphy->mutex);
    ret = udphy_dplane_enable(udphy, 0);
    if (ret)
@@ -1028,6 +1032,7 @@ static int rockchip_u3phy_init(struct phy *phy)
struct rockchip_udphy *udphy = phy_get_drvdata(phy);
int ret = 0;

+    dev_info(udphy->dev, "%s\n", __func__);
    mutex_lock(&udphy->mutex);
```

```

/* DP only or high-speed, disable U3 port */
if (!(udphy->mode & UDPHY_MODE_USB) || udphy->hs) {
@@ -1047,6 +1052,7 @@ static int rockchip_u3phy_exit(struct phy *phy)
    struct rockchip_udphy *udphy = phy_get_drvdata(phy);
    int ret = 0;

+    dev_info(udphy->dev, "%s\n", __func__);
    mutex_lock(&udphy->mutex);
    /* DP only or high-speed */
    if (!(udphy->mode & UDPHY_MODE_USB) || udphy->hs)
@@ -1363,6 +1369,7 @@ static int rk3588_udphy_init(struct rockchip_udphy *udphy)
    const struct rockchip_udphy_cfg *cfg = udphy->cfgs;
    int ret;

+    dev_info(udphy->dev, "%s\n", __func__);
    /* enable rx lfps for usb */
    if (udphy->mode & UDPHY_MODE_USB)
        grfreg_write(udphy->udphygrf, &cfg->grfcfg.rx_lfps, true);

```

DP dual mode 转接线导致异常

DP dual mode 要求 DP 口既支持 DP 信号输出，也要支持 HDMI 的 TMDS 信号传输，AUX 通道要支持 DP AUX 和 DDC(I2C)。

RK3588 DP 不支持 DP dual mode, 如果接入支持 DP dual mode 的线缆，会导致 AUX_CH 异常，这种一般出现在 DP 标准口转 HDMI 的转接线，或 DP 标准口转 HDMI 的转换器。如果用 DP 标准口转 HDMI 的转接线接 HDMI 显示器出现 AUX_CH 异常，并且转接线是支持 HDMI2.0 以下的协议版本，可能使用的转接线为支持 DP dual mode 的转接线，建议更换支持 HDMI 2.0 及以上版本的转接线。

信号干扰导致异常

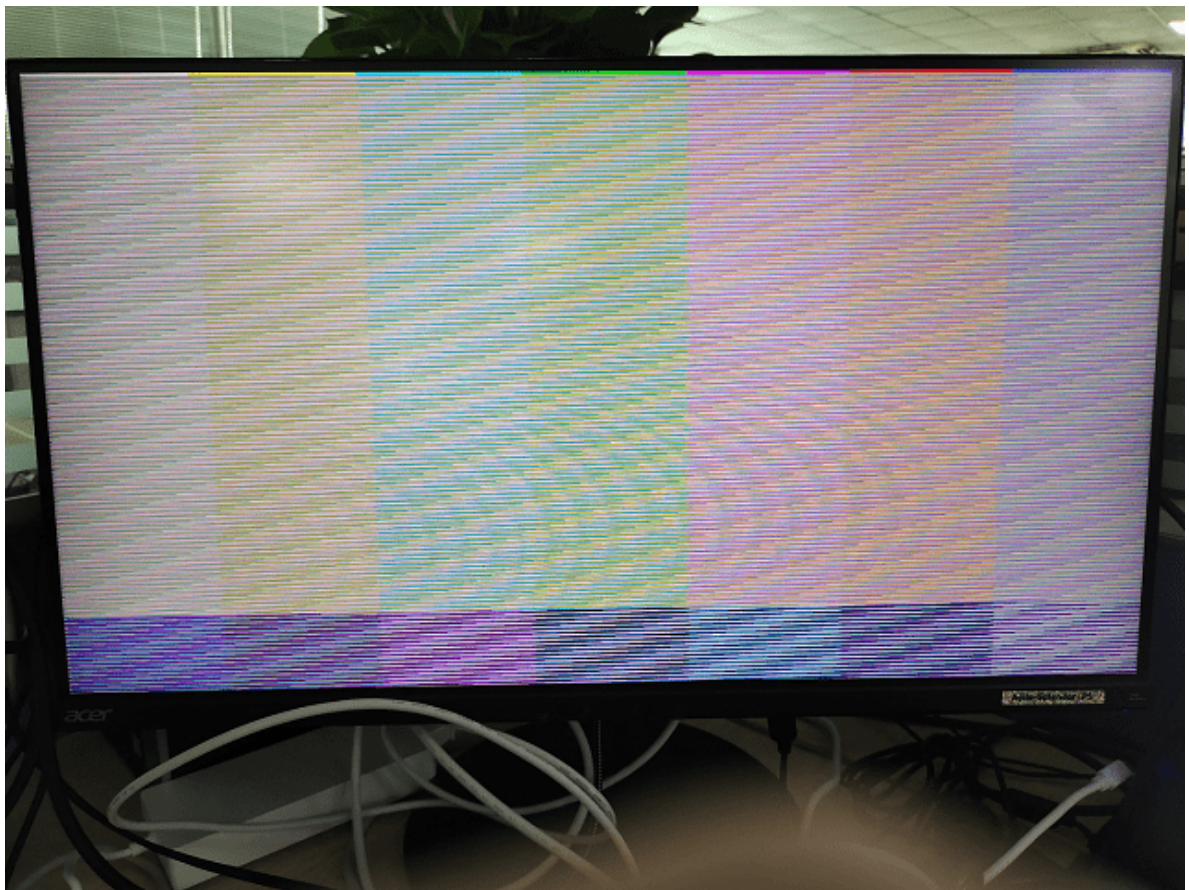
这种问题一般出现在 Type-C 直连的场景中，DP AUX 受到 USB DP/DM 上信号的干扰。可以通过硬件上把 USB DP/DM 断开，观察问题是否复现进行确认。解决这种问题，一种方式是选用质量更好的 Type-C，各信号间有做好屏蔽。另一种是不使用 USB DP/DM 传输数据。

硬件异常

首先需要确认硬件连接通路是否正常，AUX 差分信号是否正常传输，是否存在焊点虚焊；其次，AUX_CH 通路的外围电路是否参照 DP 协议进行设计；如果接的是转接芯片，先确认转接芯片的外围电路是否正常。

4K 120Hz 输出配置

RK3588 默认的 VOP ACLK 是 500M，对于输出的 4K 120Hz 这种高 pixel clk 的配置，会由于性能问题导致出现如下的显示异常：



对于这种问题，需要把 VOP ACLK 提高到 800M：

```
&vop {  
    assigned-clocks = <&cru ACLK_VOP>;  
    assigned-clock-rates = <800000000>;  
};
```

获取 VOP ACLK 如下：

```
cat /sys/kernel/debug/clk/clk_summary | grep "aclk_vop"
```

DP 带宽计算

SST 模式带宽计算

获取 DP 每条 lane 支持的带宽，公式如下：

$$bandwidth_per_lane = pixel_clk * bit_per_pixel * 1.25 / lane_count$$

其中，`bit_per_pixel` 是每个 pixel 的 bit 数，`1.25` 是 phy lane 的编码转换效率，`lane_count` 是可用的 lane 的数量，最终的计算结果 `bandwidth_per_lane` 即每条 lane 需要提供的最小带宽，如果当前的 lane rate 比需要的最小带宽小，对应的 pixel clk 的 display mode 就会被 DP 的驱动程序过滤掉。

对于使用转接线或拓展坞时，需要确定转接线和拓展坞支持的 lane rate 和 lane count 是否满足当前的带宽要求，如果无法满足，需要更换支持更高 lane rate 和更到 lane count 的转接线和拓展坞。

例如，对于一个 lane 数量为 2，最大的 lane rate 为 5.4 Gbps/lane 的拓展坞，如果要输出的 4K@60Hz，pixel clock 为 594MHz，RGB888 格式的图像数据时，需要的每条 lane 的带宽为：

$$bandwidth_per_lane = 594 * 24 * 1.25 / 2 = 8.91 Gbps / lane > 5.4 Gbps / lane$$

可以看到，当前的拓展坞不支持输出 4K@60Hz, pixel clock 为 594MHz, RGB888 格式的数据，需要使用 4 lane 输出的拓展坞，增加 PHY lane 的带宽，或输出 YUV420 格式的数据，减少需要使用 PHY lane 的带宽。

MST 模式带宽计算

MST 模式下的带宽计算与 STT 模式下类似，计算每一路流在每条 lanes 上所需要的带宽公式与 SST 模式一样，不过需要考虑多路流输出时是否会超出带宽限制。同时 MST 模式下，MTPH 的包头也占用了一定的带宽，需要考虑对应的带宽损耗。

比如按 RK3576 支持的最大带宽 4 lane、8.1Gbps 来计算：

一个 MTP 有 64 个 time slots, 其中有一个 time slot 为 MTPH, 所以每条 lane 支持的最大带宽为：

$$bandwidth_per_lane_max = 8.1 * 63 / 64 = 7.97Gbps$$

对于 4096x2160@60Hz, pixel clock 为 594MHz，RGB888 格式的显示输出，每条 lane 占用的带宽为

$$bandwidth_per_lane = 594 * 24 * 1.25 / 4 = 4.46Gbps$$

对于 2560x1440@60Hz, pixel clock 为 297MHz，RGB888 格式的显示输出，每条 lane 占用的带宽为

$$bandwidth_per_lane = 297 * 24 * 1.25 / 4 = 2.23Gbps$$

对于 1920x1080@60Hz, pixel clock 为 148.5MHz，RGB888 格式的显示输出，每条 lane 占用的带宽为

$$bandwidth_per_lane = 148.5 * 24 * 1.25 / 4 = 1.12Gbps$$

对于 RK3576，4 lane 8.1Gbps 最大的带宽下，最大三路同时输出时，三路分辨率分边设定为如上的 4096x2160@60Hz，2560x1440@60Hz，1920x1080@60Hz，每 lane 消耗的总带宽小于每 lane 最大支持的总带宽：

$$bandwidth_per_lane_total = 4.46 + 2.23 + 1.12 = 7.81Gbps < 7.97Gbps$$

这也是 MST 模式下，RK3576 3路输出支持的最大能力。

DP timing 限制

对于非标准分辨率，如果存在 porch 太小的场景，可能会导致 DP 无法输出显示，目前 DP 驱动会限制 HBP 最小值为 16，HSYNC 的最小值为 9，如果低于最小值，驱动会把对应的 HBP 或 HSYNC 调整到支持的最小值。

MST 模式使用限制

能力限制

对于 RK3576 的 DP 接口，每一路输出的最大能力如下：

DP Stream Channel	max width	max height	max pixel clock
Stream-0	4096	2160	1188MHz
Stream-1	2560	1440	300MHz
Stream-2	1920	1080	150MHz

对于 RK3576 的 VOP，每个 Video Port 的输出最大能力如下：

Vop Video Port	max width	max height	max pixel clock
Video Port 0	4096	2160	1200MHz
Video Port 1	2560	1600	300MHz
Video Port 2	1920	1080	150MHz

从 VOP 和 DP 的输出能力来看，如果在 RK3576 上 要用 DP MST 做三屏异显，能够输出支持的最大分辨率， 建议 Video Port0-> DP Stream-0, Video Port1->DP Steam1, Video Port2->DP Stream 2。
DTS 的配置如下：

```
&dp0 {
    status = "okay";
};

&dp0_in_vp0 {
    status = "okay";
};

&dp0_in_vp1 {
    status = "disabled";
};

&dp0_in_vp2 {
    status = "disabled";
};

&dp1 {
    status = "okay";
};

&dp1_in_vp0 {
    status = "disabled";
};

&dp1_in_vp1 {
    status = "okay";
};

&dp1_in_vp2 {
    status = "disabled";
};

&dp2 {
    status = "okay";
};

&dp2_in_vp0 {
    status = "disabled";
};

&dp2_in_vp1 {
    status = "disabled";
};

&dp2_in_vp2 {
```

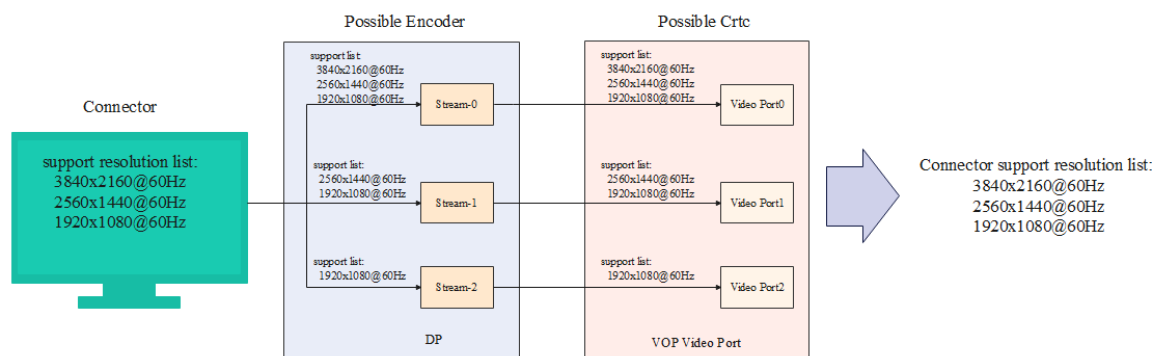
```
    status = "okay";  
};
```

如果在 RK3576 上 要用 DP MST 做三屏同显, 只能输出最大 1920x1080@60Hz, 建议 Video Port2-> DP Stream-0, Video Port2->DP Steam1, Video Port2->DP Stream 2。DTS 的配置如下:

```
&dp0 {  
    status = "okay";  
};  
  
&dp0_in_vp0 {  
    status = "disabled";  
};  
  
&dp0_in_vp1 {  
    status = "disabled";  
};  
  
&dp0_in_vp2 {  
    status = "okay";  
};  
  
&dp1 {  
    status = "okay";  
};  
  
&dp1_in_vp0 {  
    status = "disabled";  
};  
  
&dp1_in_vp1 {  
    status = "disabled";  
};  
  
&dp1_in_vp2 {  
    status = "okay";  
};  
  
&dp2 {  
    status = "okay";  
};  
  
&dp2_in_vp0 {  
    status = "disabled";  
};  
  
&dp2_in_vp1 {  
    status = "disabled";  
};  
  
&dp2_in_vp2 {  
    status = "okay";  
};  
};
```

分辨率过滤

DP MST 在 Linux DRM 框架下，是动态注册 Connector 的，并且 DP 的 Stream-0/1/2 注册成了 3 个 Encoder，但接入一个 Connector 时，不能确定这一个 Connector 最终输出时可能会使用到哪个 Encoder。如上一小节的三屏异显的配置，只有 3 个 DP Stream 都不支持的分辨率才会被过滤，如下图：



但由于每个 DP Stream 和每个 Vop Video Port 支持输出能力差异，可能导致有些分辨率在一些现实通路上无法输出，比如如果要在 Video Port2-> DP Stream-2 显示通路上输出 3840x2160@60Hz, 即无法正常输出显示。

对于上述这种情况，需要用户空间的应用程序对特定显示通路输出的分辨率进行限制。或者接入的显示设备最大的分辨率不超过 DP Stream-2 的最大支持的分辨率。