

密级状态：绝密(     )     秘密(     )     内部(     )     公开(✓)

## 基于 **DRM** 的 **Android** 显示使用指南

(第一系统产品部)

文件状态： [✓] 正在修改 [ ] 正式发布	当前版本：	V1.0
	作     者：	郑阳、许惠聪、许碧绿、操瑞杰
	完成日期：	2018-01-29
	审     核：	XXX
	完成日期：	201X-XX-XX

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Semiconductor Co., Ltd

(版本所有,翻版必究)

## 版 本 历 史

版本号	作者	修改日期	修改说明	备注
V1.0	ZY/XHC/XBL/CRJ	2018-01-29	发布初版	

# 目 录

1. 简介 .....	1
2. 屏幕配置 .....	1
2.1 配置 DTS .....	1
2.1.1 使能对应显示设备节点 .....	1
2.1.2 使能显示接口组件 .....	1
2.1.3 绑定 PLL（只有 rk3399 需要） .....	2
2.2 LCD PANEL TIMING 配置 .....	3
2.2.1 dts 中配置 timing .....	3
2.2.2 panel-simple 中配置 timing .....	4
2.3 MIPI .....	7
2.3.1 mipi 基本配置 .....	7
2.3.2 绑定 VOP .....	10
2.3.3 初始化命令详解 .....	11
2.4 HDMI 相关配置说明 .....	14
2.4.1 信号强度配置(3288/3368/3399): .....	14
2.4.2 ddc 的 i2c 速率配置: .....	15
2.4.3 使能 hdcp1.4 .....	16
2.4.4 打开音频 .....	16
2.5 DP 配置说明（RK3399） .....	16
2.5.1 DP 检测 .....	16
2.5.2 绑定 typec 口 .....	17
2.5.3 注册 dp 驱动 .....	17
2.5.4 不使用 fusb302 将 typec 口固定作 dp 输出 .....	18
2.6 参考配置 .....	18

2.6.1 单 MIPI 屏.....	18
2.6.2 双 MIPI 单屏.....	23
2.6.3 单 EDP 屏.....	26
2.6.4 两个单 MIPI 屏.....	31
2.6.5 单 LVDS 屏.....	31
2.6.6 单 HDMI(vopb).....	37
2.6.7 单 VGA (ADV7125, 基于 rk3288 PopMatal 板) .....	39
2.6.8 EDP(vopb) + HDMI(vopl).....	45
2.6.9 LVDS(vopl) + HDMI(vopb).....	46
2.6.10 MIPI(vopb) + HDMI(vopl).....	46
2.6.11 HDMI(vopb) + DP(vopl).....	47
2.6.12 HDMI + CVBS (单 VOP) .....	48
2.6.13 HDMI (vopb) + CVBS (vopl) .....	50
2.6.14 MIPI + EDP (3288 需要先配置 build.prop) .....	51
2.6.15 MIPI + LVDS/RGB (3399 不支持 LVDS/RGB, 需要外接转换 IC) .....	60
2.6.16 EDP + LVDS/RGB (3399 不支持 LVDS/RGB, 需要外接转换 IC) .....	66
<b>3. 显示框架配置.....</b>	<b>72</b>
3.1 主副显示器配置 .....	72
3.2 主副显示器接口查询 .....	73
3.3 FRAMEBUFFER 分辨率配置 .....	73
3.4 分辨率过滤配置 .....	73
<b>4. 常用调试方法.....</b>	<b>76</b>
4.1 查看 VOP 状态.....	76
4.2 查看 CONNECTOR 状态 .....	77
4.3 查看 HDMI 状态 .....	78
4.4 命令行设置分辨率 .....	79

<b>5. 参考文档</b> .....	<b>80</b>
----------------------	-----------

## 1. 简介

DRM 全称是 Direct Rendering Manager 是 DRI(Direct Rendering Infrastructure)框架的一个组件。Android 新版本逐渐从 Framebuffer 框架迁移到 DRM 上。从内核 4.4 开始, RK 的显示框架逐渐迁移到 DRM 上。本文档介绍如何使用新的显示框架。

## 2. 屏幕配置

### 2.1 配置 dts

#### 2.1.1 使能对应显示设备节点

打开显示设备执行相关 hdmi 的 probe 函数注册显示设备驱动, 如打开 HDMI 需要添加:

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

#### 2.1.2 使能显示接口组件

display-subsystem 注册会把所有打开的设备以组件的形式加在一起, 等所有的组件加载完毕后, 统一进行 bind/unbind。

##### 2.1.2.1 绑定 VOP

如果平台存在两个 VOP (RK3288、RK3399): vopb (支持 4K)、vopl (只支持 2K), 当显示设备节点打开时, 显示接口对应 vopb 和 vopl 的 ports 都会打开, 需要关闭用不到的那个 VOP。

比如 hdmi 绑定到 vopb 需要添加:

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

反之若绑定到 vopl 则添加:

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

如果平台只有一个 VOP，可以跳过。

### 2.1.2.2 开机 logo

如果 uboot logo 未开启，那 kernel 阶段也无法显示开机 logo，只能等到 android 启动后才能看到显示。在 dts 里面将对应的 route 使能即可打开 uboot logo 支持，比如打开 hdmi 的 uboot logo 显示:

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

### 2.1.3 绑定 PLL（只有 rk3399 需要）

rk3399 的 hdmi 所绑定的 vop 时钟需要挂载到 vppll 上，若是双显需将另一个 vop 时钟挂到 cppll 这样可以分出任意 dclk 的频率。如当 hdmi 绑定到 vopb 时配置:

```
&vopb {  
    assigned-clocks = <&cru DCLK_VOP0_DIV>;  
    assigned-clock-parents = <&cru PLL_VPLL>;  
};  
&vopl {  
    assigned-clocks = <&cru DCLK_VOP1_DIV>;  
    assigned-clock-parents = <&cru PLL_CPLL>;  
};
```

当 hdmi 绑定到 vopl 时配置:

```
&vopb {  
    assigned-clocks = <&cru DCLK_VOP0_DIV>;  
    assigned-clock-parents = <&cru PLL_CPLL>;  
};  
&vopl {  
    assigned-clocks = <&cru DCLK_VOP1_DIV>;  
    assigned-clock-parents = <&cru PLL_VPLL>;
```

```
};
```

## 2.2 LCD Panel timing 配置

目前我们平台 LCD panel（MIPI/EDP/LVDS）的 timing 配置有以下两种：

- ① dts 文件中配置 disp\_timings 结构。
- ② 在 drivers/gpu/drm/panel/panel-simple.c 中添加 timing 结构。

下面分别对这两种配置进行详细说明，以 RK3399 SDK 上的 EDP panel（lg\_lp079qx1\_sp0v）为例：

### 2.2.1 dts 中配置 timing

这种方式直接在 dts 中配置 disp\_timings 结构，简单明了，推荐客户使用该方式，dts 中的配置如下：

```
edp_panel: edp-panel {
    compatible = "panel-simple";
    backlight = <&backlight>;
    power-supply = <&vcc3v3_s0>;
    enable-gpios = <&gpio1 13 GPIO_ACTIVE_HIGH>;

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

    disp_timings: display-timings {
        native-mode = <&timing0>;

        timing0: timing0 {
            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>;
    };
};
};

```

Property	Value	Comment
clock-frequency	200000000	Dclk 频率，单位 Hz
hactive	1536	行有效像素
vactive	2048	列有效像素
hfront-porch	12	行前消隐
hsync-len	16	行同步信号
hback-porch	48	行后消隐
vfront-porch	8	列前消隐
vsync-len	4	列同步信号
vback-porch	8	列后消隐
hsync-active	0	Hsync 信号极性配置
vsync-active	0	Vsync 信号极性配置
de-active	0	Den 信号极性配置
pixelclk-active	0	Dclk 信号极性配置

### 2.2.2 panel-simple 中配置 timing

把 Timing 写在 panel-simple.c 中，直接以短字符串匹配，该方式为 upstream 推荐的使

用方式。

这种方式 dts 中的配置如下：

```
edp_panel: edp-panel {
    compatible = "lg,lp079qx1-sp0v";
    backlight = <&backlight>;
    power-supply = <&vcc3v3_s0>;
    enable-gpios = <&gpio1 13 GPIO_ACTIVE_HIGH>;

    ports {
        panel_in_edp: endpoint {
            remote-endpoint = <&edp_out_panel>;
        };
    };
};
```

drivers/gpu/drm/panel/panel-simple.c 中添加对应的结构。分下面两块：

① 对 EDP/LVDS 屏来说，首先需要在 platform\_of\_match[] 结构中添加 of\_device\_id 结构：

```
{
    .compatible = "lg,lp079qx1-sp0v",
    .data = &lg_lp079qx1_sp0v,
},
```

MIPI 屏添加的位置为 dsi\_of\_match[] 结构中，以 SDK 的 MIPI panel 为例：

```
{
    .compatible = "boe,tv080wum-nl0",
    .data = &boe_tv080wum_nl0
},
```

**Note:** compatible 中的名字需要跟 dts 中的 compatible 名字一致。

② 添加上面 data 所指定的 &lg\_lp079qx1\_sp0v 结构：

```
static const struct panel_desc lg_lp079qx1_sp0v = {
    .modes = &lg_lp079qx1_sp0v_mode,
    .num_modes = 1,
    .size = {
        .width = 129,
        .height = 171,
    },
    .bus_format = MEDIA_BUS_FMT_RGB666_1X18,
```

```
};
```

Property	Value	Comment
num_modes	1	设置为 1 即可
size	129、171	屏的物理尺寸
bus_format	MEDIA_BUS_FMT_RGB666_1X18	主控输出给屏的数据格式

③ 添加第②步中 modes 指定的&lg\_lp079qx1\_sp0v\_mode 结构:

```
static const struct drm_display_mode lg_lp079qx1_sp0v_mode = {
    .clock = 200000,
    .hdisplay = 1536,
    .hsync_start = 1536 + 12,
    .hsync_end = 1536 + 12 + 16,
    .htotal = 1536 + 12 + 16 + 48,
    .vdisplay = 2048,
    .vsync_start = 2048 + 8,
    .vsync_end = 2048 + 8 + 4,
    .vtotal = 2048 + 8 + 4 + 8,
    .vrefresh = 60,
    .flags = DRM_MODE_FLAG_NVSYNC | DRM_MODE_FLAG_NHSYNC,
};
```

Property	Value	Comment
clock	200000	Dclk, 单位 kHz
hdisplay	1536	行有效分辨率
hsync_start	1536 + 12	Hdisplay + HFP
hsync_end	1536 + 12 + 16	Hdisplay + HFP + HSYNC
htotal	1536 + 12 + 16 + 48	行总像素 Hdisplay + HFP + HSYNC + HBP
vdisplay	2048	列有效分辨率
vsync_start	2048 + 8	Vdisplay + VFP

vsync_end	2048 + 8 + 4	Vdisplay + VFP + VSYNC
vtotal	2048 + 8 + 4 + 8	列总像素 Vdisplay + VFP + VSYNC + VBP
vrefresh	60	刷新率
flags	DRM_MODE_FLAG_NVSYN C   DRM_MODE_FLAG_NHSYN C	Video mode flags

## 2.3 MIPI

### 2.3.1 mipi 基本配置

#### 2.3.1.1 MIPI-DSI Host 以及 Panel 的配置

关于屏的 timing 信息我们有两种配置方式：

- ① dts 文件中配置 disp\_timings 结构。见下面截图中的结构。
- ② 在 drivers/gpu/drm/panel/panel-simple.c 中添加 timing 结构。见 2.2 章节。

```
&dsi0 {
    status = "okay";
    rockchip, lane-rate = <1000>;

    panel: panel {
        compatible = "simple-panel-dsi";
        reg = <0>;
        backlight = <&backlight>;
        enable-gpios = <&gpio7 3 GPIO_ACTIVE_HIGH>;
        reset-gpios = <&gpio7 4 GPIO_ACTIVE_HIGH>;
        dsi, flags = <(MIPI_DSI_MODE_VIDEO | MIPI_DSI_MODE_VIDEO_BURST |
                     MIPI_DSI_MODE_LPM)>;
        dsi, format = <MIPI_DSI_FMT_RGB888>;
        dsi, lanes = <4>;
        reset-delay-ms = <20>;
        init-delay-ms = <20>;
        enable-delay-ms = <20>;
        prepare-delay-ms = <20>;
        unprepare-delay-ms = <20>;

        panel-init-sequence = [
            39 00 04 b9 ff 83 94
            15 00 02 df 8e
            05 64 01 11
            05 14 01 29
        ];
        panel-exit-sequence = [
            05 64 01 28
            05 96 01 10
        ];

        disp_timings: display-timings {
            native-mode = <&timing0>;
            timing0: timing0 {
                clock-frequency = <150000000>;
                hactive = <1200>;
                vactive = <1920>;
                hback-porch = <80>;
                hfront-porch = <81>;
                vback-porch = <21>;
                vfront-porch = <21>;
                hsync-len = <10>;
                vsync-len = <3>;
                hsync-active = <0>;
                vsync-active = <0>;
                de-active = <0>;
                pixelclk-active = <0>;
            };
        };
    };
};
```

### 2.3.1.2 属性说明

Property	Value	Comment
rockchip, lane-rate	1000	Mipi clk 速率，单位 Mbps/lane。如果没有配置该属性，驱动会根据屏的 timing 计算 lane-rate
compatible	simple-panel-dsi	与 mipi panel 驱动进行匹配
reg	0	Mipi DSI virtual channel

backlight	&backlight	引用背光节点，panel 驱动会解析并对背光进行控制
enable-gpios	&gpio7 3 GPIO_ACTIVE_HIGH	屏的 enable 脚的 gpio 配置，参考原理图。
reset-gpios	&gpio7 4 GPIO_ACTIVE_HIGH	屏的 reset 脚的 gpio 配置，参考原理图。
dsi,flags	MIPI_DSI_MODE_VIDEO   MIPI_DSI_MODE_VIDEO_BURST   MIPI_DSI_MODE_LPM	Dsi host 的一些 flag 选项。  MIPI_DSI_MODE_VIDEO_BURST: 数据发送模式，绝大部分 mipi 屏都只支持或者兼容支持该模式。  MIPI_DSI_MODE_LPM: 在 low power 模式下发送命令
dsi,format	MIPI_DSI_FMT_RGB888	Pixel format 具体看屏的要求，mipi 一般是 888
dsi,lanes	4	Lane 的条数，dual-channel 屏为 8。参考屏规格书
reset-delay-ms	20	Reset 信号有效脉冲宽度，可选，参考屏规格书
init-delay-ms	20	发送初始化命令之前的延时，可选，参考屏规格书
enable-delay-ms	20	开启背光之前的延时，可选，参考屏规格书
prepare-delay-ms	20	Enable 信号控制后的延时，可选，参考屏规格书
unprepare-delay-ms	20	屏幕掉电 gpio 操作后的延时，可选，参考屏规格书
panel-init-sequence	05 64 01 11 等	屏上电的初始化序列，一般由屏的 fae 提供。
panel-exit-sequence	05 96 01 10 等	屏下电序列，一般由屏的 fae 提供
disp_timings		屏的 timing 信息，参考屏规格书

### 2.3.1.3 MIPI-DPHY

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

**Note:** 只适用于使用 Non-SNPS PHY 作为 DPHY 的 Soc 系列，比如 RK3366/RK3368。

对于使用 SNPS PHY 作为 DPHY 的 Soc 系统，比如 RK3288/RK3399，不需要配置该 node。

### 2.3.1.4 uboot logo

```
&route_dsi0 {  
    status = "okay";  
};
```

Route\_dsi0 是 uboot 显示的开关。okay 表示开启 uboot 显示。

## 2.3.2 绑定 VOP

关于 mipi dsi 连接那个 vop 的配置，配置分 uboot 和 kernel 两部分。针对 rk3288 平台，这部分配置放在 rk3288-android.dtsi 中。

### 2.3.2.1 uboot 配置

```
route_dsi0: route-dsi0 {  
    status = "disabled";  
    logo,uboot = "logo.bmp";  
    logo,kernel = "logo_kernel.bmp";  
    logo,mode = "center";  
    charge_logo,mode = "center";  
    connect = <&vopb_out_dsi0>;  
};
```

上面 connect = <&vopb\_out\_dsi0>表示 dsi0 连接 vopb，如果需要连接 vopl，可以设置为<&vopl\_out\_dsi0>。

### 2.3.2.2 kernel 中的配置

在没有做任何配置的前提下，我们 kernel 中默认接 vopl，如需要接跟 uboot 保持一致接

vopb, 那么需要把 dsi0\_in\_vop1 给关闭, 如下所示:

```
&dsi0_in_vop1 {  
    status = "disabled";  
};
```

### 2.3.3 初始化命令详解

#### 2.3.3.1 命令格式说明

```
panel-init-sequence = [  
    39 00 04 b9 ff 83 94  
    15 00 02 df 8e  
    05 64 01 11  
    05 14 01 29  
];  
  
panel-exit-sequence = [  
    05 64 01 28  
    05 96 01 10  
];
```

命令的前面三个字节分别表示命令类型、延时和命令净荷长度。从第四个字节开始表示命令的有效 payload。这个字节数需要与第三个字节一致。

第一个字节代表的命令类型分两大类: DCS 命令和 Generic 命令。DCS 命令是有 mipi 标准协议里面指定的命令, 具体可以参考《MIPI Alliance Specification for Display Command Set.pdf》。Generic 命令一般应用于厂商自定义的命令。具体使用哪种类型需要参考屏规格书或者咨询屏厂 FAE。如果没有特别指定, 建议按 DCS 命令类型配置。

DCS 命令的类型有三种: 0x05/0x15/0x39。Generic 命令的类型分为: 0x03/0x13/0x23/0x29。详细的含义见下面表格:



Data Type, hex	Data Type, binary	Description	Packet Size
0x01	00 0001	Sync Event, V Sync Start	Short
0x11	01 0001	Sync Event, V Sync End	Short
0x21	10 0001	Sync Event, H Sync Start	Short
0x31	11 0001	Sync Event, H Sync End	Short
0x08	00 1000	End of Transmission packet (EoTp)	Short
0x02	00 0010	Color Mode (CM) Off Command	Short
0x12	01 0010	Color Mode (CM) On Command	Short
0x22	10 0010	Shut Down Peripheral Command	Short
0x32	11 0010	Turn On Peripheral Command	Short
0x03	00 0011	Generic Short WRITE, no parameters	Short
0x13	01 0011	Generic Short WRITE, 1 parameter	Short
0x23	10 0011	Generic Short WRITE, 2 parameters	Short
0x04	00 0100	Generic READ, no parameters	Short
0x14	01 0100	Generic READ, 1 parameter	Short
0x24	10 0100	Generic READ, 2 parameters	Short
0x05	00 0101	DCS Short WRITE, no parameters	Short
0x15	01 0101	DCS Short WRITE, 1 parameter	Short
0x06	00 0110	DCS READ, no parameters	Short
0x37	11 0111	Set Maximum Return Packet Size	Short
0x09	00 1001	Null Packet, no data	Long
0x19	01 1001	Blanking Packet, no data	Long
0x29	10 1001	Generic Long Write	Long
0x39	11 1001	DCS Long Write/write_LUT Command Packet	Long
0x0C	00 1100	Loosely Packed Pixel Stream, 20-bit YCbCr, 4:2:2 Format	Long
0x1C	01 1100	Packed Pixel Stream, 24-bit YCbCr, 4:2:2 Format	Long
0x2C	10 1100	Packed Pixel Stream, 16-bit YCbCr, 4:2:2 Format	Long
0x0D	00 1101	Packed Pixel Stream, 30-bit RGB, 10-10-10 Format	Long
0x1D	01 1101	Packed Pixel Stream, 36-bit RGB, 12-12-12 Format	Long

Data Type, hex	Data Type, binary	Description	Packet Size
0x3D	11 1101	Packed Pixel Stream, 12-bit YCbCr, 4:2:0 Format	Long
0x0E	00 1110	Packed Pixel Stream, 16-bit RGB, 5-6-5 Format	Long
0x1E	01 1110	Packed Pixel Stream, 18-bit RGB, 6-6-6 Format	Long
0x2E	10 1110	Loosely Packed Pixel Stream, 18-bit RGB, 6-6-6 Format	Long
0x3E	11 1110	Packed Pixel Stream, 24-bit RGB, 8-8-8 Format	Long
0xX0 and 0xFF, unspecified	XX 0000 XX 1111	DO NOT USE All unspecified codes are reserved	

第二个字节表示的是发送完该命令以后需要延时的时间，单位为 ms。这个时间需要参考屏规

格书或者由屏厂 FAE 提供。

第三个字节表示的是命令 payload 的长度。

以 panel-init-sequence 中的第一条命令[39 00 04 b9 ff 83 94]为例，详细解释一下各个字节表示的含义：

0x39 表示这是一个 DCS 长包命令，payload 长度大于 2。

0x00 表示该命令发送后无需延时。

0x04 表示命令长度为 4。

0xb9 0xff 0x83 0x94 表示实际发送的命令。

### 2.3.3.2 示例

以 Japan Display 的 H070ME05000 进行说明，该屏的规格书有详细的初始化序列说明。

#### (B) On sequence

sequence	Data Type (hex)	index (hex)	parameters # (hex)	description	comment
SLEEP MODE					
↓					
DCDC_EN L->H				DCDC_EN L->H (VSP,VSN on)	
wait 20ms					
command	05	01	- -	soft reset	
wait 5ms					
command	23	B0	1 00	MCAP	
command	29	B3	1 04 2 08 3 00 4 22 5 00	Interface setting	
command	29	B4	1 0C	Interface ID setting	
command	29	B6	1 3A 2 D3	DSI control	
command	15	51	1 E6	write display brightness	
command	15	53	1 2C	write control display	
command	15	3A	1 77	set pixel format	
command	39	2A	1 00 2 00 3 04 4 AF	set column address	
command	39	2B	1 00 2 00 3 07 4 7F	set page address	
send image	39	2C/3C		write memory / write memory continue	
command	05	11	- -	exit sleep mode	
wait 120ms					
command	05	29	- -	set display on	
wait min 0ms					
LED_EN L->H				LED_EN L->H	
↓					
NORMAL MODE					

```
panel-init-sequence = [
    05 05 00 01 01
    23 00 00 02 b0 00
    23 00 00 02 d6 01
    29 00 00 06 b3 14 08 00 22 00
    29 00 00 02 b4 0c
    29 00 00 03 b6 3a c3
    15 00 00 02 51 e6
    15 00 00 02 53 2c
    15 00 00 02 3a 77
    39 00 00 05 2a 00 00 04 af
    39 00 00 05 2b 00 00 07 7f
    05 78 00 01 29
    05 00 00 01 11
];
```

### (C) Off sequence

sequence	DataTyp (hex)	index (hex)	parameters # (hex)	description	comment
NORMAL MODE					
↓					
command	05	28	-	set display off	
wait 20ms					
command	05	10	-	enter sleep mode	
wait 80ms					
DCDC_EN H->L				DCDC_EN H->L (VSP,VSN off)	
wait 20ms					
↓					
SLEEP MODE					

```
panel-exit-sequence = [
    05 14 00 01 28
    05 50 00 01 10
];
```

## 2.4 HDMI 相关配置说明

### 2.4.1 信号强度配置(3288/3368/3399):

由于硬件走线差异,不同板子有可能需要不同的驱动强度配置,当遇到电视兼容性问题时可以尝试修改这个看是否有改善。

hdmi 信号强度可通过 dts 的 rockchip.phy-table 属性配置,格式定义:<PIXELCLOCK PHY\_CKSYMTXCTRL PHY\_TXTERM PHY\_VLEVCTRL>.

PIXELCLOCK 表示所在行参数所对应的最大 pixelclock 频率。

PHY\_CKSYMTXCTRL 寄存器(0x09)值用于调整 HDMI 信号的预加重和上升斜率，加大预加重或 sloop boost，可以提升 Data 信号的上升斜率，但会降低信号的上升/下降时间。

Bit[0]: Clock 信号使能

Bit[3:1]: Data 信号预加重，定义如下

Bit[4:5]: Data 信号 sloop boost

PHY\_TXTERM 寄存器(0x19)值用于调整端接电阻值

Bit[0:2]: 值越大，端接电阻值越大。

PHY\_VLEVCTRL 寄存器(0x0e)值用于调整 HDMI 的信号幅度，具体定义如下：

Bit[0:4] : tmds\_clk +/- 信号幅度，值越低，驱动能力越强；

Bit[5:9]: tmds\_data +/- 信号幅度，值越低，驱动能力越强。

如：

```
&hdmi {
    rockchip,phy-table =
        <74250000 0x8009 0x0004 0x0272>,
        <165000000 0x802b 0x0004 0x0209>,
        <297000000 0x8039 0x0005 0x028d>,
        <594000000 0x8039 0x0000 0x019d>,
        <000000000 0x0000 0x0000 0x0000>;
};
```

其中<74250000 0x8009 0x0004 0x0272>，表示 pixeclock 为 74.25M(720p 分辨率)

以下是 PHY\_CKSYMTXCTRL 寄存器值为 0x8009；PHY\_TXTERM 值为 0x0004；PHY\_VLEVCTRL 值为 0x0272。修改后也可用 cat /d/dw-hdmi/phy 命令查看对应的寄存器值确认是否有修改成功。

### 2.4.2 ddc 的 i2c 速率配置：

目前 i2c 速率通过 clk 高电平和低电平的时间来定义，如下为实测 i2c 速率为 50k 时候的配置，调整 i2c 速率只需将这 2 个值按对应的比例修改即可。

```
&hdmi {
    ddc-i2c-scl-high-time-ns = <9625>;
```

```
        ddc-i2c-scl-low-time-ns = <10000>;  
    }
```

### 2.4.3 使能 hdcp1.4

```
&hdmi {  
    hdcp1x-enable = <1>;  
}
```

使能 hdcp1.4 后还需要通过工具烧录对应 key，当前工具版本为“ProvisioningTool20170526.zip”具体操作详见工具内的使用文档。

### 2.4.4 打开音频

3368 和 3288 默认 hdmi 声卡和 codec 公用，需确认配置如下：

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

3399 目前 hdmi 声卡和 dp 公用：

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

## 2.5 DP 配置说明（RK3399）

### 2.5.1 DP 检测

3399 的 typec 支持 dp/usb3/usb2，sdk 默认使用 fusb302 来检测接入的设备类型。当设备接入时，fusb302 通过 extcon 传递给 usb 驱动。fusb302 是通过 i2c 外挂的芯片，下面配置是挂到 i2c4 上时打开的配置，若挂在其他 i2c 上则需要对应修改。

```
&i2c4 {  
    status = "okay";  
    fusb0: fusb30x@22 {  
        status = "okay";  
    };  
};
```

## 2.5.2 绑定 typec 口

3399 有两个功能相同的 typec 口，都支持 dp 输出，不过由于 dp 控制器只有一个，所以同一时刻最多只能有一个 typec 口输出 dp 信号。

typec0 口包括 usb 控制器(&usbdrd3\_0); usb3phy(&tcphy0)和 usb2phy (&u2phy0);

typec1 口包括 usb 控制器(&usbdrd3\_1);usb3phy(&tcphy1)和 usb2phy (&u2phy1);

若 fusb302 接到 typec0 口时需配置如下：

```
&tcphy0 {
    extcon = <&fusb0>;
    status = "okay";
};
&u2phy0 {
    status = "okay";
    extcon = <&fusb0>;
};
&usbdrd3_0 {
    extcon = <&fusb0>;
    status = "okay";
};
```

若 fusb302 接到 typec1 口时需配置如下：

```
&tcphy1{
    extcon = <&fusb0>;
    status = "okay";
};
&u2phy1 {
    status = "okay";
    extcon = <&fusb0>;
};
&usbdrd3_1 {
    extcon = <&fusb0>;
    status = "okay";
};
```

## 2.5.3 注册 dp 驱动

打开 dp 的 dts 同时绑定 extcon 配置：

```
&cdn_dp {  
    status = "okay";  
    extcon = <&fusb0>;  
}
```

### 2.5.4 不使用 fusb302 将 typec 口固定作 dp 输出

有些产品不接 fusb302，而将其中一个 typec 口固定做 dp 输出，这时需要自己添加一个 extcon 驱动如 vpd0（目前补丁 vpd0.patch 还未提交）。然后将 dp 和 usb 的 extcon 设置成 vpd0 如下：

```
vpd0: virtual-pd0 {  
    status = "okay";  
}  
&cdn_dp {  
    status = "okay";  
    extcon = <&vpd0>;  
}  
&tcphy0 {  
    extcon = <&vpd0>;  
    status = "okay";  
};
```

//以上为接到 typec0 口的配置，若是接到 typec1 口则需将 tcphy0 改为 tcphy1 如下：

```
/*&tcphy1 {  
    extcon = <&vpd0 >;  
    status = "okay";  
};*/
```

## 2.6 参考配置

### 2.6.1 单 MIPI 屏

参考 dts:

RK3288: arch/arm/boot/dts/rk3288-evb-android-rk818-mipi.dts

RK3399: arch/arm64/boot/dts/rockchip/rk3399-evb-rev3-android.dts

### 2.6.1.1 使能 MIPI-DSI Host 以及 Panel

- RK3288:

```
&dsi0 {
    status = "okay";
    rockchip, lane-rate = <1000>;

    panel: panel {
        compatible = "simple-panel-dsi";
        reg = <0>;
        enable-gpios = <&gpio7 4 GPIO_ACTIVE_HIGH>;
        power-supply = <&vcc_lcd>;
        dsi, flags = <(MIPI_DSI_MODE_VIDEO |
MIPI_DSI_MODE_VIDEO_BURST)>;
        dsi, format = <MIPI_DSI_FMT_RGB888>;
        dsi, lanes = <4>;
        reset-delay-ms = <20>;
        init-delay-ms = <20>;
        enable-delay-ms = <120>;
        prepare-delay-ms = <120>;
        status = "okay";

        display_timings: display-timings {
            native-mode = <&timing0>;

            timing0: timing0 {
                clock-frequency = <150000000>;
                hactive = <1200>;
                vactive = <1920>;
                hback-porch = <80>;
                hfront-porch = <81>;
                vback-porch = <21>;
                vfront-porch = <21>;
                hsync-len = <10>;
                vsync-len = <3>;
                hsync-active = <0>;
                vsync-active = <0>;
                de-active = <0>;
                pixelclk-active = <0>;
            }
        }
    }
}
```



```
};
};
};
};
```

- RK3399:

```
&dsi {
    status = "okay";
    panel@0 {
        compatible = "simple-panel-dsi";
        reg = <0>;
        backlight = <&backlight>;
        enable-gpios = <&gpio1 13 GPIO_ACTIVE_HIGH>;

        dsi,flags = <(MIPI_DSI_MODE_VIDEO |
MIPI_DSI_MODE_VIDEO_BURST)>;
        dsi,format = <MIPI_DSI_FMT_RGB888>;
        dsi,lanes = <4>;

        panel-init-sequence = [
            05 64 01 11
            05 14 01 29
        ];

        panel-exit-sequence = [
            05 64 01 28
            05 96 01 10
        ];

        display_timings: display-timings {
            native-mode = <&timing0>;

            timing0: timing0 {
                clock-frequency = <150000000>;
                hactive = <1200>;
                vactive = <1920>;
                hback-porch = <21>;
                hfront-porch = <120>;
                vback-porch = <18>;
                vfront-porch = <21>;
```

```

        hsync-len = <20>;
        vsync-len = <3>;
        hsync-active = <0>;
        vsync-active = <0>;
        de-active = <0>;
        pixelclk-active = <0>;
    };
};
};
};

```

**Note:** 如需添加初始化命令，参考 2.3 章节。

### 2.6.1.2 使能 MIPI-DPHY

```

&mipi_dphy {
    status = "okay"
};

```

**Note:** 只适用于使用 Non-SNPS PHY 作为 DPHY 的 Soc 系列，比如 RK3366/RK3368。对于使用 SNPS PHY 作为 DPHY 的 Soc 系统，比如 RK3288/RK3399，不需要配置该 node。

### 2.6.1.3 uboot-logo 开启配置

- RK3288:

```

&route_dsi0 {
    status = "okay"
};

```

- RK3399:

```

&route_dsi {
    status = "okay"
};

```

### 2.6.1.4 uboot 指定 DSI 对应 VOP

- RK3288:

DSI 连接 vop1

```

&route_dsi0 {

```

```
connect = <&vopl_out_dsi0>;
};
```

DSI 连接 vopb

```
&route_dsi0 {
    connect = <&vopb_out_dsi0>;
};
```

- RK3399:

DSI 连接 vopl

```
&route_dsi {
    connect = <&vopl_out_dsi>;
};
```

DSI 连接 vopb

```
&route_dsi {
    connect = <&vopb_out_dsi>;
};
```

### 2.6.1.5 kernel 指定 DSI 对应 VOP

- RK3288:

DSI 连接 VOPL

```
&dsi0_in_vopl {
    status = "okay";
};
&dsi0_in_vopb {
    status = "disabled";
};
```

DSI 连接 VOPB

```
&dsi0_in_vopb {
    status = "okay";
};
&dsi0_in_vopl {
    status = "disabled";
};
```

- RK3399

DSI 连接 VOPL

```
&dsi_in_vopl {
    status = "okay";
};
&dsi_in_vopb {
    status = "disabled";
};
```

DSI 连接 VOPB

```
&dsi_in_vopb {
    status = "okay";
};
&dsi_in_vopl {
    status = "disabled";
};
```

## 2.6.2 双 MIPI 单屏

参考 dts:

RK3288: arch/arm/boot/dts/rk3288-x7811-rk818-dual-dsi.dts

### 2.6.2.1 MIPI-DSI DSI0 以及 Panel

```
&dsi0 {
    status = "okay";
    rockchip,dual-channel = <&dsi1>;
    rockchip,lane-rate = <900>;

    panel: panel@0 {
        compatible = "simple-panel-dsi";
        reg = <0>;
        backlight = <&backlight>;
        reset-gpios = <&gpio1 24 GPIO_ACTIVE_LOW>;
        power-supply = <&vcc_lcd>;
        dsi,flags = <(MIPI_DSI_MODE_VIDEO | MIPI_DSI_MODE_VIDEO_BURST |
                     MIPI_DSI_MODE_LPM | MIPI_DSI_MODE_EOT_PACKET)>;
        dsi,format = <MIPI_DSI_FMT_RGB888>;
        dsi,lanes = <8>;

        reset-delay-ms = <120>;
```

```

init-delay-ms = <120>;
enable-delay-ms = <120>;
prepare-delay-ms = <120>;

panel-init-sequence = [
    05 64 01 11
    05 14 01 29
];

panel-exit-sequence = [
    05 64 01 28
    05 96 01 10
];

disp_timings: display-timings {
    native-mode = <&timing0>;

    timing0: timing0 {
        clock-frequency = <230000000>;
        hactive = <1536>;
        vactive = <2048>;
        hback-porch = <180>;
        hfront-porch = <180>;
        vback-porch = <10>;
        vfront-porch = <14>;
        hsync-len = <48>;
        vsync-len = <2>;
        hsync-active = <0>;
        vsync-active = <0>;
        de-active = <0>;
        pixelclk-active = <0>;
    };
};
};
};

```

**Note:** 相比单 MIPI 屏，改动主要是下面两处：

- ① 添加了 `rockchip,dual-channel = <&dsi1>;`
- ② 修改 lane 的条数 `dsi,lanes = <8>;`

### 2.6.2.2 MIPI-DSI DSI1

```
&dsi1 {
    status = "okay"
};
```

### 2.6.2.3 Uboot LOGO

```
&route_dsi0 {
    status = "okay"
};
```

### 2.6.2.4 uboot 指定 DSI 对应的 VOP

- RK3288:

DSI 连接 vopl

```
&route_dsi0 {
    connect = <&vopl_out_dsi0>;
};
```

DSI 连接 vopb

```
&route_dsi0 {
    connect = <&vopb_out_dsi0>;
};
```

- RK3399:

DSI 连接 vopl

```
&route_dsi {
    connect = <&vopl_out_dsi>;
};
```

DSI 连接 vopb

```
&route_dsi {
    connect = <&vopb_out_dsi>;
};
```

### 2.6.2.5 kernel 指定 DSI 对应的 VOP

- RK3288:

DSI 连接 VOPL

```
&dsi0_in_vopl {
    status = "okay";
};
&dsi0_in_vopb {
    status = "disabled";
};
```

DSI 连接 VOPB

```
&dsi0_in_vopb {
    status = "okay";
};
&dsi0_in_vopl {
    status = "disabled";
};
```

- RK3399

DSI 连接 VOPL

```
&dsi_in_vopl {
    status = "okay";
};
&dsi_in_vopb {
    status = "disabled";
};
```

DSI 连接 VOPB

```
&dsi_in_vopb {
    status = "okay";
};
&dsi_in_vopl {
    status = "disabled";
};
```

### 2.6.3 单 EDP 屏

参考 dts 配置文件:

Rk3288: arch/arm/boot/dts/rk3288-evb-android-rk818-edp.dts

Rk3399: arch/arm64/boot/dts/rockchip/rk3399-evb-rev3-android-edp.dts

### 2.6.3.1 使能 EDP host

- RK3288:

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

- RK3399 & RK3368:

```
&edp {
    status = "okay";

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

            edp_out_panel: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&panel_in_edp>;
            };
        };
    };
};
```

**Note:** 3288 中的 ports 结构在 rk3288-evb.dtsi 中已经有定义，所以无需重复添加。

### 2.6.3.2 使能 EDP phy

- RK3288 & RK3368:

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

**Note:** rk3399 无需配置该项。



### 2.6.3.3 配置 EDP panel

- RK3288:

```
&edp_panel {
    compatible = "panel-simple";
    enable-gpios = <&gpio7 4 GPIO_ACTIVE_HIGH>;
    enable-delay-ms = <120>;
    pinctrl-0 = <&lcd_cs>;
    power-supply = <&vcc_lcd>;
    status = "status";

    disp_timings: display-timings {
        native-mode = <&timing0>;

        timing0: timing0 {
            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>;
        };
    };
};
```

- RK3399:

```
edp_panel: edp-panel {
    compatible = "lg,lp079qx1-sp0v";
    backlight = <&backlight>;
    power-supply = <&vcc3v3_s0>;
    enable-gpios = <&gpio1 13 GPIO_ACTIVE_HIGH>;
```

```
ports {
    panel_in_edp: endpoint {
        remote-endpoint = <&edp_out_panel>;
    };
};
};
```

属性说明:

Property	Value	Comment
backlight	&backlight	引用背光节点，panel 驱动会对背光进行处理。
enable-gpios	&gpio7 4 GPIO_ACTIVE_HIGH	屏的 enable 脚的 gpio 配置，参考原理图。
enable-delay-ms	120	Enable 信号有效以后的延时，单位 ms
power-supply	&vcc_lcd 或 &vcc3v3_s0	Sdk 屏的 3.0 供电，这个由 lcd_en 来控制。这个是 SDK 特有的控制手段，客户一般无需配置。参考原理图。
pinctrl-0	&lcd_cs	这个就是 enable-gpios，我们 iomux 存在复用，加这个是为了确认为 gpio。不加也可以，默认使用为 gpio

**Note:**

① 3288 edp panel 的 backlight 在 rk3288-android.dtsi 里面已有包含，所以这里无需重复添加。

② 3399 的配置中没有 disp\_timings 结构。这个在 drivers/gpu/drm/panel/panel-simple.c 中定义相应的结构。详细介绍见 2.2 章节。

#### 2.6.3.4 开启 Uboot LOGO

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

### 2.6.3.5 uboot 中 DSI 挂在 VOP

RK3288 默认指定 uboot 中 EDP 对应的 VOP 为 vopl。而 RK3399 默认指定 uboot 中 EDP 对应的 VOP 为 vopb。

连接 vopb 时：

```
&route_edp {  
    connect = <&vopb_out_edp>;  
};
```

连接 vopl 时：

```
&route_edp {  
    connect = <&vopl_out_edp>;  
};
```

### 2.6.3.6 kernel 中 DSI 挂载 VOP

- RK3288:

RK3288 SDK kernel 中默认接 vopl，在 rk3288-android.dtsi 中有把 edp\_in\_vopb 给关掉。若改成接 vopb，需要把 edp\_in\_vopl 给关闭，把 edp\_in\_vopb 开启，如下所示：

```
&edp_in_vopl {  
    status = "disabled";  
};  
  
&edp_in_vopb {  
    status = "okay";  
};
```

- RK3399:

RK3399 接 vopb 只需如下配置：

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

## 2.6.4 两个单 MIPI 屏

### 2.6.4.1 两个相同 MIPI 屏接一个 VOP

配置与 2.3.2 双 MIPI 单屏一致。请参考该章节。

### 2.6.4.2 两个相同 MIPI 屏接两个 VOP

3288 不支持，3399 理论上支持，但是目前只做了简单的验证，没有客户量产。

### 2.6.4.3 两个不同 MIPI 屏接两个 VOP

Rk3399 理论上支持，暂未验证。

## 2.6.5 单 LVDS 屏

参考 dts:

RK3288: arch/arm/boot/dts/rk3288-evb-android-rk818-lvds.dts

**RK3399: RK3399 无 LVDS 接口，需外接转换 IC**

### 2.6.5.1 配置 LVDS host

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

### 2.6.5.2 配置 LVDS Panel

```
&lvds_panel {
    status = "okay";
    compatible = "simple-panel";
    backlight = <&backlight>;
    bus-format = <MEDIA_BUS_FMT_RGB666_1X18>;
    enable-gpios = <&gpio7 4 GPIO_ACTIVE_HIGH>;
    enable-delay-ms = <10>;
```

```
power-supply = <&vcc_lcd>;
rockchip,data-mapping = "jeida";
rockchip,data-width = <24>;
rockchip,output = "lvds";

display-timings {
    native-mode = <&timing0>;
    timing0: timing0 {
        clock-frequency = <71000000>;
        hactive = <1280>;
        vactive = <800>;
        hback-porch = <100>;
        hfront-porch = <18>;
        vback-porch = <8>;
        vfront-porch = <6>;
        hsync-len = <10>;
        vsync-len = <2>;
        hsync-active = <0>;
        vsync-active = <0>;
        de-active = <0>;
        pixelclk-active = <0>;
    };
};
};
```

属性说明:

Property	Value	Comment
backlight	&backlight	引用背光节点，panel 驱动会对背光进行处理。
bus-format	MEDIA_BUS_FMT_RGB666_1X18	主控输出给屏的数据格式
enable-gpios	&gpio7 4 GPIO_ACTIVE_HIGH	屏的 enable 脚的 gpio 配置，参考原理图。
enable-delay-ms	10	Enable 信号有效以后的延时，单位是 ms
rockchip,data-	vesa or jeida	LVDS 信号的两种编码方式，具体对应关系参考 4.3 的

mapping		data mapping 说明。
rockchip,data-width	18 or 24 or 30	LVDS 的数据位，RGB 三个分量都是 6bit 的填 18， RGB 三个分量都是 8bit 的填 24，RGB 三个分量都是 10bit 的填 30。
rockchip,output	lvds or rgb	LVDS 输出的两种模式：  lvds: LVDS 屏的配置；  rgb: RGB 屏的配置。

### 2.6.5.3 开启 uboot logo

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

### 2.6.5.4 uboot 中 lvds 挂载 vop

接 vopb:

```
&route_lvds {
    connect = <&vopb_out_lvds>;
};
```

接 vopl:

```
&route_lvds {
    connect = <&vopl_out_lvds>;
};
```

### 2.6.5.5 kernel 中 lvds 挂在 vop

接 vopb:

```
&lvds_in_vopb {
    status = "okay";
};
&lvds_in_vopl {
```

```
status = "disabled";  
};
```

接 vopl:

```
&lvds_in_vopl {  
    status = "okay";  
};  
&lvds_in_vopb {  
    status = "disabled";  
};
```

### 2.6.5.6 Data\_mapping

#### 1) 6 bit output mode

采用 4+1 的传输模式，即 4 组数据信号加一组时钟信号，最后一组数据信号传输无效数据。

		VESA_6BIT	JEIDA_6BIT
Y 0	TX0	R0	R2
	TX1	R1	R3
	TX2	R2	R4
	TX3	R3	R5
	TX4	R4	R6
	TX6	R5	R7
	TX7	G0	G2
Y 1	TX8	G1	G3
	TX9	G2	G4
	TX12	G3	G5
	TX13	G4	G6
	TX14	G5	G7
	TX15	B0	B2
	TX18	B1	B3
Y 2	TX19	B2	B4
	TX20	B3	B5
	TX21	B4	B6
	TX22	B5	B7
	TX24	HSYNC	HSYNC
	TX25	VSYNC	VSYNC
	TX26	ENABLE	ENABLE
Y 3	TX27	GND	GND
	TX5	GND	GND
	TX10	GND	GND
	TX11	GND	GND
	TX16	GND	GND
	TX17	GND	GND
	TX23	RSVD	RSVD

## 2) 8 bit output mode

采用 4+1 的传输模式，即 4 组数据信号加一组时钟信号。



		VESA_8BIT	JEIDA_8BIT
Y 0	TX0	R0	R2
	TX1	R1	R3
	TX2	R2	R4
	TX3	R3	R5
	TX4	R4	R6
	TX6	R5	R7
	TX7	G0	G2
Y 1	TX8	G1	G3
	TX9	G2	G4
	TX12	G3	G5
	TX13	G4	G6
	TX14	G5	G7
	TX15	B0	B2
	TX18	B1	B3
Y 2	TX19	B2	B4
	TX20	B3	B5
	TX21	B4	B6
	TX22	B5	B7
	TX24	HSYNC	HSYNC
	TX25	VSYNC	VSYNC
	TX26	ENABLE	ENABLE
Y 3	TX27	R6	R0
	TX5	R7	R1
	TX10	G6	G0
	TX11	G7	G1
	TX16	B6	B0
	TX17	B7	B1
	TX23	RSVD	RSVD

### 3) 10 bit output mode

采用 5+1 的传输模式，即 5 组数据信号加一组时钟信号。

		VESA_10BIT	JEIDA_10BIT
Y 0	TX0	R0	R4
	TX1	R1	R5
	TX2	R2	R6
	TX3	R3	R7
	TX4	R4	R8
	TX6	R5	R9
	TX7	G0	G4
Y 1	TX8	G1	G5
	TX9	G2	G6
	TX12	G3	G7
	TX13	G4	G8
	TX14	G5	G9
	TX15	B0	B4
	TX18	B1	B5
Y 2	TX19	B2	B6
	TX20	B3	B7
	TX21	B4	B8
	TX22	B5	B9
	TX24	HSYNC	HSYNC
	TX25	VSYNC	VSYNC
	TX26	ENABLE	ENABLE
Y 3	TX27	R6	R2
	TX5	R7	R3
	TX10	G6	G2
	TX11	G7	G3
	TX16	B6	B2
	TX17	B7	B3
	TX23	GND	GND
Y 4	TX27	R8	R0
	TX5	R9	R1
	TX10	G8	G0
	TX11	G9	G1
	TX16	B8	B0
	TX17	B9	B1
	TX23	GND	GND

## 2.6.6 单 HDMI(vopb)

- RK3288:

3288 hwc 需要做修改，我们之前的代码默认只支持 MIPI/EDP/LVDS 等独显。HDMI 独显需要在 hardware/rockchip/hwcomposer/drmresources.cpp 做如下修改：

```
diff --git a/drmresources.cpp b/drmresources.cpp
index 9f89687..eb5f23d 100755
--- a/drmresources.cpp
+++ b/drmresources.cpp
@@ -385,12 +385,23 @@ int DrmResources::Init() {
    }
}

+   if (!found_primary) {
+   for (auto &conn : connectors_) {
+       found_primary = true;
+       conn->set_display_possible(conn->possible_displays() |
HWC_DISPLAY_PRIMARY_BIT);
+       SetPrimaryDisplay(conn.get());
+       break;
+   }
+ }
+
    if (!found_primary) {
        ALOGE("failed to find primary display\n");
        return -ENODEV;
    }

    for (auto &conn : connectors_) {
+   if (GetConnectorFromType(HWC_DISPLAY_PRIMARY) == conn.get())
+       continue;
        if (!(conn->possible_displays() & HWC_DISPLAY_EXTERNAL_BIT))
            continue;
        if (conn->state() != DRM_MODE_CONNECTED)
```

在上面修改的前提下，再在 **kernel** 中做如下配置：

```
/* 打开 hdmi 设备节点 */
&hdmi {
    status = "okay";
};
/* 绑定 hdmi 到 vopb */
&hdmi_in_vopl {
    status = "disabled";
};
&hdmi_in_vopb {
```

```

    status = "okay";
};
/* 开启 hdmi 的 uboot logo 显示 */
&route_hdmi {
    status = "okay";
};

```

注：目前 3288 暂不支持 hdmi uboot logo 显示，还没做验证。

- RK3399:

```

/* 打开 hdmi 设备节点 */
&hdmi {
    status = "okay";
};
/* 绑定 hdmi 到 vopb */
&hdmi_in_vopl {
    status = "disabled";
};
/* 开启 hdmi 的 uboot logo 显示 */
&route_hdmi {
    status = "okay";
};
/* 关闭 vopl */
&vopl {
    status = disabled;
};

```

RK3399 还需将 vopb 时钟挂到 vppll 上如下：（其他芯片不需要配置）

```

&vopb {
    assigned-clocks = <&cru DCLK_VOP0_DIV>;
    assigned-clock-parents = <&cru PLL_VPLL>;
};

```

## 2.6.7 单 VGA（ADV7125, 基于 rk3288 PopMatal 板）

因为这个 VGA 是通过 RGB 转 VGA 输出，所以有两种不同的配置方式：

1、把 VGA 当做一个独立的 encoder, 有独立的 vga(bridge) 驱动。HWC 那边得到 encoder 类型是 VGA。

2、把 bridge 当做一个可以读取 edid 的特殊 RGB 屏，无需添加专门的 VGA 驱动。上报给

HWC 的 encoder 类型是 LVDS。

关于 VGA 热插拔时的分辨率切换，发现是在 R, G, B 三基色对应的 pin 插入的时候，会触发 I2C 的读写操作。发现 R、G、B 三基色对应的 pin 的电压值在插入的情况下比空载的情况下降低一倍。由此可以了解 VGA 热插拔实现分辨率切换的检测机制。

### 2.6.7.1 独立 VGA Encoder

参考 dts: arch/arm/boot/dts/rk3288-popmetal-android-vga.dts

VGA 驱动: drivers/gpu/drm/bridge/dumb-vga-dac.c

虽然有单独的 VGA 驱动，但是对于主控而言输出的还是 RGB 信号，所以还是需要开启 LVDS 的相关选项。

#### ① 打开 LVDS 设备节点

```
&lvds {
    status = "okay";

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

            rgb_out: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&adv7125_in>;
            };
        };
    };
};
```

#### ② 配置 bridge 节点

```
vga_bridge: vga-bridge {
    compatible = "adi,adv7125";
    psave-gpios = <&gpio7 21 GPIO_ACTIVE_HIGH>;
    vdd-supply = <&vcc_lcd>;
    rockchip,output = "rgb";
```

```
#address-cells = <1>;
#size-cells = <0>;

ports {

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

    port@0 {
        reg = <0>;
        adv7125_in: endpoint {
            remote-endpoint = <&rgb_out>;
        };
    };

    port@1 {
        reg = <1>;
        adv7125_out: endpoint {
            remote-endpoint = <&vga_in>;
        };
    };
};
};
```

### ③ 创建 vga 虚拟节点

```
vga {
    compatible = "vga-connector";
    ddc-i2c-bus = <&i2c2>;

    port {
        vga_in: endpoint {
            remote-endpoint = <&adv7125_out>;
        };
    };
};
```

### ④ 开启 uboot 显示

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

注：目前 uboot 中还没有添加改驱动。所以暂时不支持 uboot 显示

#### ⑤ uboot 中连接 vop

接 vopb:

```
&route_lvds {  
    connect = <&vopb_out_lvds>;  
};
```

接 vopl:

```
&route_lvds {  
    connect = <&vopl_out_lvds>;  
};
```

#### ⑥ kernel 中连接 vop

接 vopb:

```
&lvds_in_vopb {  
    status = "okay";  
};  
&lvds_in_vopl {  
    status = "disabled";  
};
```

接 vopl:

```
&lvds_in_vopl {  
    status = "okay";  
};  
&lvds_in_vopb {  
    status = "disabled";  
};
```

### 2.6.7.2 特殊 RGB 屏

参考 dts: arch/arm/boot/dts/rk3288-popmetal-android.dts

因为是由 RGB 经过数模转换成 VGA 输出，所以对于我们主控而言输出的是 RGB 数据，目前的做法是当作一个可以读取 EDID 信息的 RGB 屏来调试。RGB 目前在平台上跟 LVDS 统一，所以配置都是基于 LVDS。

#### ① 打开设备节点

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

## ② 配置 panel

```
&lvds_panel {
    status = "okay";
    compatible = "simple-panel";
    enable-gpios = <&gpio7 21 GPIO_ACTIVE_HIGH>;
    enable-delay-ms = <10>;
    power-supply = <&vcc_lcd>;
    ddc-i2c-bus = <&i2c2>;
    rockchip,output = "rgb";

    display-timings {
        native-mode = <&timing0>;

        timing0: timing0 {
            clock-frequency = <74250000>;
            hactive = <1280>;
            vactive = <720>;
            hback-porch = <220>;
            hfront-porch = <120>;
            vback-porch = <20>;
            vfront-porch = <5>;
            hsync-len = <40>;
            vsync-len = <5>;
            hsync-active = <0>;
            vsync-active = <0>;
            de-active = <0>;
            pixelclk-active = <0>;
        };
    };
};
```

属性说明:

Property	Value	Comment
----------	-------	---------



enable-gpios	&gpio7 GPIO_ACTIVE_HIGH	4 屏的 enable 脚的 gpio 配置，参考原理图。
enable-delay-ms	10	Enable 信号有效以后的延时，单位是 ms
power-supply	&vcc_lcd	Sdk 板子独有的供电方式，客户一般都 gpio 操作
ddc-i2c-bus	&i2c2	读取 edid 的 i2c 通道
rockchip,output	lvds or rgb	LVDS 输出的两种模式：  lvds: LVDS 屏的配置；  rgb: RGB 屏的配置。

### ③ 开启 uboot 显示

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

### ④ uboot 中连接 vop

接 vopb:

```
&route_lvds {
    connect = <&vopb_out_lvds>;
};
```

接 vopl:

```
&route_lvds {
    connect = <&vopl_out_lvds>;
};
```

### ⑤ kernel 中选择连接 vop

接 vopb:

```
&lvds_in_vopb {
    status = "okay";
};
&lvds_in_vopl {
    status = "disabled";
};
```

接 vopl:

```
&lvds_in_vopl {
```

```
status = "okay";
};
&lvds_in_vopb {
    status = "disabled";
};
```

## 2.6.8 EDP(vopb) + HDMI(vopl)

```
/* 打开 edp 设备节点 */
&edp {
    status = "okay";
};
/* 绑定 edp 到 vopb */
&edp_in_vopl {
    status = "disabled";
};
/* 开启 edp 的 uboot logo 显示 */
&route_edp {
    status = "okay"
};
/* 打开 hdmi 设备节点 */
&hdmi {
    status = "okay";
};
/* 绑定 hdmi 到 vopl */
&hdmi_in_vopb {
    status = "disabled";
};
```

RK3399 还需将 hdmi 绑定的 vopl 时钟挂到 vp11 上如下：（其他芯片不需要配置）

```
&vopb {
    assigned-clocks = <&cru DCLK_VOP0_DIV>;
    assigned-clock-parents = <&cru PLL_CPLL>;
};
&vopl {
    assigned-clocks = <&cru DCLK_VOP1_DIV>;
    assigned-clock-parents = <&cru PLL_VPLL>;
};
```

## 2.6.9 LVDS(vopl) + HDMI(vopb)

```
//使能 LVDS
&lvds {
    status = "okay";
};
//绑定 LVDS 到 vopl
&lvds_in_vopb {
    status = "disabled";
};
//LVDS 屏显示 uboot logo
&route_lvds{
    connect = <&vopl_out_lvds>;
    status = "okay"
};
//使能 HDMI
&hdmi {
    status = "okay";
};
//绑定 HDMI 到 vopb
&hdmi_in_vopl {
    status = "disabled";
};
```

RK3399 还需将 hdmi 绑定的 vopb 时钟挂到 vpll 上如下：（其他芯片不需要配置）

```
&vopb {
    assigned-clocks = <&cru DCLK_VOP0_DIV>;
    assigned-clock-parents = <&cru PLL_VPLL>;
};
&vopl {
    assigned-clocks = <&cru DCLK_VOP1_DIV>;
    assigned-clock-parents = <&cru PLL_CPLL>;
};
```

## 2.6.10 MIPI(vopb) + HDMI(vopl)

```
&mipi {
    status = "okay";
};
&mipi_in_vopl {
```

```

    status = "disabled";
};
&hdmi {
    status = "okay";
};
&hdmi_in_vopb {
    status = "disabled";
};
&route_mipi{
    status = "okay"
};

```

RK3399 还需将 hdmi 绑定的 vopl 时钟挂到 vp11 上如下：（其他芯片不需要配置）

```

&vopb {
    assigned-clocks = <&cru DCLK_VOP0_DIV>;
    assigned-clock-parents = <&cru PLL_CPLL>;};
&vopl {
    assigned-clocks = <&cru DCLK_VOP1_DIV>;
    assigned-clock-parents = <&cru PLL_VPLL>;
};

```

### 2.6.11 HDMI(vopb) + DP(vopl)

```

&hdmi {
    status = "okay";
};
&hdmi_in_vopl {
    status = "disabled";
};
&cdn_dp {
    status = "okay";
};
&dp_in_vopl {
    status = "disabled";
};
&route_hdmi {
    status = "okay"
};

```

RK3399 还需将 hdmi 绑定的 vopb 时钟挂到 vp11 上如下：（其他芯片不需要配置）

```

&vopb {

```

```
assigned-clocks = <&cru DCLK_VOP0_DIV>;
assigned-clock-parents = <&cru PLL_VPLL>;
};
&vopl {
    assigned-clocks = <&cru DCLK_VOP1_DIV>;
    assigned-clock-parents = <&cru PLL_CPLL>;
};
```

## 2.6.12 HDMI + CVBS (单 VOP)

### 2.6.12.1 非外接 CVBS 的平台

RK3228、RK3328 内部集成了 HDMI、CVBS，参考如下配置：

```
/* 打开 hdmi 设备节点 */
&hdmi {
    status = "okay";
};
/* 开启 hdmi 的 uboot logo 显示 */
&route_hdmi {
    status = "okay"
};
/* 打开 TVE 设备节点，输出 CVBS */
&tve {
    status = "okay"
};
```

### 2.6.12.2 使用 RK1000 输出 CVBS 的平台

```
/* 打开 hdmi 设备节点 */
&hdmi {
    status = "okay";
};
/* 开启 hdmi 的 uboot logo 显示 */
&route_hdmi {
    status = "okay"
};
```

RK3368 使用通过 I2C 通信的外接芯片 RK1000 输出 CVBS，由于 RK1000 的 CVBS 挂载在 LVDS 下，所以需要使能 LVDS，并创建 port 供 TVE 挂载，相关详细说明参考：

## Documentation/devicetree/bindings/media/video-interfaces.txt

```
&lvds {
    status = "okay";

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

            lvds_out_tve: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&tve_in_lvds>;
            };
        };
    };
};
```

RK1000 通过 I2C 进行通信，所以对应的 I2C 必须开启（下面例子为 I2C1）。RK1000 的 cvbs 显示需要开启两个模块：rk1000\_ctl 和 rk1000-tve 模块。CTL 也即 CORE CTRL，为 I2C 的总控制模块，TVE 也即 TV ENCODER，为 CVBS 控制模块。rk1000-tve 必须挂载在 lvds 下（红字部分）。

```
&i2c1 {
    status = "okay";

    rk1000_ctl: rk1000-ctl@40 {
        compatible = "rockchip,rk1000-ctl";
        status = "okay";
        reg = <0x40>;
        reset-gpios = <&gpio0 1 GPIO_ACTIVE_LOW>;
        clocks = <&cru SCLK_I2S_8CH_OUT>;
        clock-names = "mclk";
        pinctrl-names = "default";
        pinctrl-0 = <&i2s_8ch_mclk>;
    };

    rk1000-tve@42 {
        status = "okay";
```

```
compatible = "rockchip,rk1000-tve";
reg = <0x42>;
rockchip,data-width = <24>;
rockchip,output = "rgb";
rockchip,ctl = <&rk1000_ctl>;
ports {
    #address-cells = <1>;
    #size-cells = <0>;
    tve_in: port@0 {
        reg = <0>;
        tve_in_lvds: endpoint {
            remote-endpoint = <&lvds_out_tve>;
        };
    };
};
};
```

参数说明如下:

Property	Value	Comment
reset-gpios	&gpio0 1 GPIO_ACTIVE_LOW	配置 RK1000 的 reset 引脚, 参考原理图
clocks	&cru SCLK_I2S_8CH_OUT	RK1000 CORE CTRL 的 I2C 通信需要的 mclk。
clock-names	mclk	时钟名称
pinctrl-0	&i2s_8ch_mclk	mclk 的 pinctrl
pinctrl-names	default	mclk 的 pinctrl 名称
rockchip,data-width	24	输出色彩格式, 必须为 24 或 18
rockchip,output	rgb	输出格式, 必须为 rgb, lvd, duallvds
rockchip,ctl	&rk1000_ctl	引用 RK1000 CORE CTRL

### 2.6.13 HDMI (vopb) + CVBS (vopl)

暂缺。

## 2.6.14 MIPI + EDP（3288 需要先配置 build.prop）

对于双 LCD 显示，我们建议先单独调试好 MIPI 屏和 EDP 屏，再去做双 LCD 显示。这样步骤流程更清晰，遇到问题也便于定位。

参考 dts:

RK3288:arch/arm/boot/dts/rk3288-evb-android-rk818-mipi-edp.dts

RK3399:arch/arm64/boot/dts/rockchip/rk3399-evb-rev3-android-mipi-edp.dts

### 2.6.14.1 开启 mipi 以及配置 mipi panel

保持事先调试好的单 MIPI 屏的配置即可。

- RK3288:

```
&dsi0 {
    status = "okay";
    rockchip,lane-rate = <1000>;

    panel: panel {
        compatible = "simple-panel-dsi";
        reg = <0>;
        power-supply = <&vcc_lcd>;
        dsi,flags = <(MIPI_DSI_MODE_VIDEO |
MIPI_DSI_MODE_VIDEO_BURST)>;
        dsi,format = <MIPI_DSI_FMT_RGB888>;
        dsi,lanes = <4>;
        reset-delay-ms = <20>;
        init-delay-ms = <20>;
        enable-delay-ms = <120>;
        prepare-delay-ms = <120>;
        status = "okay";

        display_timings: display-timings {
            native-mode = <&timing0>;

            timing0: timing0 {
                clock-frequency = <150000000>;
                hactive = <1200>;
```



```

        vactive = <1920>;
        hback-porch = <80>;
        hfront-porch = <81>;
        vback-porch = <21>;
        vfront-porch = <21>;
        hsync-len = <10>;
        vsync-len = <3>;
        hsync-active = <0>;
        vsync-active = <0>;
        de-active = <0>;
        pixelclk-active = <0>;
    };
};
};
};

```

- RK3399:

```

&dsi {
    status = "okay";
    panel@0 {
        compatible = "simple-panel-dsi";
        reg = <0>;
        backlight = <&backlight>;
        enable-gpios = <&gpio1 13 GPIO_ACTIVE_HIGH>;

        dsi,flags = <(MIPI_DSI_MODE_VIDEO |
MIPI_DSI_MODE_VIDEO_BURST)>;
        dsi,format = <MIPI_DSI_FMT_RGB888>;
        dsi,lanes = <4>;

        panel-init-sequence = [
            05 64 01 11
            05 14 01 29
        ];

        panel-exit-sequence = [
            05 64 01 28
            05 96 01 10
        ];
    };
};

```

```
display_timings: display-timings {
    native-mode = <&timing0>;

    timing0: timing0 {
        clock-frequency = <150000000>;
        hactive = <1200>;
        vactive = <1920>;
        hback-porch = <21>;
        hfront-porch = <120>;
        vback-porch = <18>;
        vfront-porch = <21>;
        hsync-len = <20>;
        vsync-len = <3>;
        hsync-active = <0>;
        vsync-active = <0>;
        de-active = <0>;
        pixelclk-active = <0>;
    };
};
};
};
```

#### 2.6.14.2 开启 edp

- RK3288:

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

- RK3399:

```
&edp {
    status = "okay";

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

```
edp_out_panel: endpoint@0 {
    reg = <0>;
    remote-endpoint = <&panel_in_edp>;
};

};

};

};
```

**Note:** 3288 中的 **ports** 结构在 **rk3288-evb.dtsi** 中已经有定义，所以无需重复添加。

### 2.6.14.3 配置 edp panel

- RK3288:

```
&edp_panel {
    compatible = "simple-panel";
    enable-gpios = <&gpio7 4 GPIO_ACTIVE_HIGH>;
    enable-delay-ms = <120>;
    pinctrl-0 = <&lcd_cs>;
    power-supply = <&vcc_lcd>;
    status = "okay";

    display-timings {
        native-mode = <&F402>;

        F402: timing0 {
            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>;
        };
    };
};
```

```
};  
};
```

- RK3399:

```
edp_panel: edp-panel {  
    compatible = "lg,lp079qx1-sp0v", "panel-simple";  
    backlight = <&backlight1>;  
    power-supply = <&vcc3v3_s0>;  
  
    ports {  
        panel_in_edp: endpoint {  
            remote-endpoint = <&edp_out_panel>;  
        };  
    };  
  
    disp_timings: display-timings {  
        native-mode = <&F402>;  
  
        F402: timing0 {  
            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>;  
        };  
    };  
};
```

**Note:** **backlight** 如果两个 **panel** 共用同样的 **gpio** 和 **pwm**，那么就无需添加，不然会资源冲突。如果不同，那么 **backlight** 后面的**&backlight** 需要和 **mipi panel** 中的背光命名区分。

**Backlight** 的详细配置见 2.3.13.4。

**enable-gpios** 如果两个 **panel** 共用同样的 **gpio**，则无需添加该行。

#### 2.6.14.4 配置 **edp** 背光 **backlight**

我们 **sdk** 上 **edp** 和 **mipi** 共用背光设置，所以无需进行下面的配置，当然进行下面的配置也是可以正常显示的。下面的配置是给各自有独立背光的 **panel** 设定的。

- **RK3288:**

```
backlight1: backlight1 {
    compatible = "pwm-backlight";
    pwms = <&pwm1 0 25000 0>;
    brightness-levels = <
        0   1   2   3   4   5   6   7 8   9  10  11  12  13  14  15
       16  17  18  19  20  21  22  23 24  25  26  27  28  29  30  31
       32  33  34  35  36  37  38  39 40  41  42  43  44  45  46  47
       48  49  50  51  52  53  54  55 56  57  58  59  60  61  62  63
       64  65  66  67  68  69  70  71 72  73  74  75  76  77  78  79
       80  81  82  83  84  85  86  87 88  89  90  91  92  93  94  95
       96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111
      112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127
      128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
      144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159
      160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175
      176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191
      192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207
      208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223
      224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239
      240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
    >;
    default-brightness-level = <128>;
    enable-gpios = <&gpio7 2 GPIO_ACTIVE_HIGH>;
    pinctrl-names = "default";
    pinctrl-0 = <&bl_en>;
    pwms = <&pwm0 0 1000000 0>;
};

&pwm1 {
    status = "okay";
```

```
};
```

```
&backlight1 {
    status = "okay";
};
```

- RK3399:

```
backlight1: backlight1 {
    compatible = "pwm-backlight";
    pwms = <&pwm1 0 25000 0>;
    brightness-levels = <
        0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
        16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
        32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47
        48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63
        64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79
        80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95
        96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111
        112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127
        128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
        144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159
        160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175
        176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191
        192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207
        208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223
        224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239
        240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
    >;
    default-brightness-level = <200>;
};
```

```
&pwm1 {
    status = "okay";
};
```

```
&backlight1 {
    status = "okay";
};
```

**Note:** 3399 和 3288 相比，少了 gpio 的配置。因为 3399 sdk 上背光的 gpio 和 lcd\_en 共用，所以这里无需配置。

#### 2.6.14.5 配置 VOP 连接

- RK3288:

Mipi 接 vopb, edp 接 vopl:

```
&route_dsi0 {
    connect = <&vopb_out_dsi0>;
};
&route_edp {
    connect = <&vopl_out_edp>;
};
&dsi0_in_vopl {
    status = "disabled";
};
&dsi0_in_vopb {
    status = "okay";
};
&edp_in_vopl {
    status = "okay";
};
&edp_in_vopb {
    status = "disabled";
};
```

Mipi 接 vopl, edp 接 vopb:

```
&route_dsi0 {
    connect = <&vopl_out_dsi0>;
};
&route_edp {
    connect = <&vopb_out_edp>;
};
&dsi0_in_vopl {
    status = "okay";
};
&dsi0_in_vopb {
```

```

    status = "disabled";
};
&edp_in_vopl {
    status = "disabled";
};
&edp_in_vopb {
    status = "okay";
};

```

● RK3399:

Mipi 接 vopb, edp 接 vopl:

```

&route_dsi {
    connect = <&vopb_out_dsi>;
};
&route_edp {
    connect = <&vopl_out_edp>;
};
&dsi_in_vopl {
    status = "disabled";
};
&dsi_in_vopb {
    status = "okay";
};
&edp_in_vopl {
    status = "okay";
};
&edp_in_vopb {
    status = "disabled";
};

```

Mipi 接 vopl, edp 接 vopb:

```

&route_dsi {
    connect = <&vopl_out_dsi>;
};
&route_edp {
    connect = <&vopb_out_edp>;
};
&dsi_in_vopb {
    status = "disabled";
};

```



```
};  
&dsi_in_vopl {  
    status = "okay";  
};  
&edp_in_vopb {  
    status = "okay";  
};  
&edp_in_vopl {  
    status = "disabled";  
};
```

**Note: 3288** 为了 HDMI 4K 显示，默认 HDMI 接 VOPB，MIPI 和 EDP 默认接 VOPL。

#### 2.6.14.6 开启 uboot 显示

- RK3288:

```
&route_dsi0 {  
    status = "okay";  
};  
  
&route_edp {  
    status = "okay";  
};
```

- RK3399:

```
&route_dsi {  
    status = "okay";  
};  
  
&route_edp {  
    status = "okay";  
};
```

#### 2.6.15 MIPI + LVDS/RGB (3399 不支持 LVDS/RGB，需要外接转换 IC)

对于双 LCD 显示，我们建议先单独调试好 MIPI 屏和 LVDS/RGB 屏，再去做双 LCD 显示。这样步骤流程更清晰，遇到问题也便于定位。

参考 dts:

暂缺（客户那边按下面配置已经点亮）。

注：3288 需要先配置 build.prop，关于 build.prop 配置，具体见 3.1 章节，调试阶段可以按如下所示做配置：

```
adb root
adb remount
adb pull system/build.prop
修改这个文件，添加
sys.hwc.device.primary=DSI
sys.hwc.device.extend=LVDS
然后
adb push build.prop system
adb shell "chmod 644 system/build.prop"
adb reboot
```

#### 2.6.15.1 开启 mipi 以及配置 mipi panel

保持事先调试好的单 MIPI 屏的配置即可。

- RK3288:

```
&dsi0 {
    status = "okay";
    rockchip,lane-rate = <1000>;

    panel: panel {
        compatible = "simple-panel-dsi";
        reg = <0>;
        power-supply = <&vcc_lcd>;
        dsi,flags = <(MIPI_DSI_MODE_VIDEO |
MIPI_DSI_MODE_VIDEO_BURST)>;
        dsi,format = <MIPI_DSI_FMT_RGB888>;
        dsi,lanes = <4>;
        reset-delay-ms = <20>;
        init-delay-ms = <20>;
        enable-delay-ms = <120>;
        prepare-delay-ms = <120>;
        status = "okay";

        display_timings: display-timings {
```

```

native-mode = <&timing0>;

timing0: timing0 {
    clock-frequency = <150000000>;
    hactive = <1200>;
    vactive = <1920>;
    hback-porch = <80>;
    hfront-porch = <81>;
    vback-porch = <21>;
    vfront-porch = <21>;
    hsync-len = <10>;
    vsync-len = <3>;
    hsync-active = <0>;
    vsync-active = <0>;
    de-active = <0>;
    pixelclk-active = <0>;
};
};
};
};

```

### 2.6.15.2 开启 LVDS

- RK3288:

```

&lvds {
    status = "okay";
};

```

### 2.6.15.3 配置 LVDS panel

- RK3288:

```

&lvds_panel {
    compatible = "simple-panel";
    bus-format = <MEDIA_BUS_FMT_RGB666_1X18>;
    enable-gpios = <&gpio7 4 GPIO_ACTIVE_HIGH>;
    enable-delay-ms = <10>;
    power-supply = <&vcc_lcd>;
    rockchip,data-mapping = "jeida";

```

```
rockchip,data-width = <24>;
rockchip,output = "lvds";
status = "okay";

display-timings {
    native-mode = <&lvds_panel_name>;

    lvds_panel_name: timing0 {
        clock-frequency = <71000000>;
        hactive = <1280>;
        vactive = <800>;
        hfront-porch = <18>;
        hsync-len = <10>;
        hback-porch = <100>;
        vfront-porch = <6>;
        vsync-len = <2>;
        vback-porch = <8>;
        hsync-active = <0>;
        vsync-active = <0>;
        de-active = <0>;
        pixelclk-active = <0>;
    };
};
};
```

**Note: backlight** 如果两个 panel 共用同样的 gpio 和 pwm，那么就无需添加，不然会资源冲突。如果不同，那么 **backlight** 后面的&backlight 需要和 mipi panel 中的背光命名区分。

**Backlight** 的详细配置见 2.3.14.4。

**enable-gpios** 如果两个 panel 共用同样的 gpio，则无需添加该行。

#### 2.6.15.4 配置 LVDS 背光 **backlight**

我们 sdk 上 LVDS 和 MIPI 共用背光设置，所以无需进行下面的配置，当然进行下面的配置也是可以正常显示的。下面的配置是给各自有独立背光的 panel 设定的。

- RK3288:

```
backlight1: backlight1 {
    compatible = "pwm-backlight";
```

```

pwms = <&pwm1 0 25000 0>;
brightness-levels = <
    0   1   2   3   4   5   6   7 8   9  10  11  12  13  14  15
    16  17  18  19  20  21  22  23 24  25  26  27  28  29  30  31
    32  33  34  35  36  37  38  39 40  41  42  43  44  45  46  47
    48  49  50  51  52  53  54  55 56  57  58  59  60  61  62  63
    64  65  66  67  68  69  70  71 72  73  74  75  76  77  78  79
    80  81  82  83  84  85  86  87 88  89  90  91  92  93  94  95
    96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111
    112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127
    128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
    144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159
    160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175
    176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191
    192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207
    208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223
    224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239
    240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
>;
default-brightness-level = <128>;
enable-gpios = <&gpio7 2 GPIO_ACTIVE_HIGH>;
pinctrl-names = "default";
pinctrl-0 = <&bl_en>;
pwms = <&pwm0 0 1000000 0>;
};

```

```

&pwm1 {
    status = "okay";
};

```

```

&backlight1 {
    status = "okay";
};

```

### 2.6.15.5 配置 VOP 连接

- RK3288:

Mipi 接 vopb, lvds 接 vopl:

```
&route_dsi0 {
    connect = <&vopb_out_dsi0>;
};
&route_lvds {
    connect = <&vopl_out_lvds>;
};
&dsi0_in_vopl {
    status = "disabled";
};
&dsi0_in_vopb {
    status = "okay";
};
&lvds_in_vopl {
    status = "okay";
};
&lvds_in_vopb {
    status = "disabled";
};
```

Mipi 接 vopl, lvds 接 vopb:

```
&route_dsi0 {
    connect = <&vopl_out_dsi0>;
};
&route_lvds {
    connect = <&vopb_out_lvds>;
};
&dsi0_in_vopl {
    status = "okay";
};
&dsi0_in_vopb {
    status = "disabled";
};
&lvds_in_vopl {
    status = "disabled";
};
&lvds_in_vopb {
    status = "okay";
};
```

### 2.6.15.6 开启 uboot 显示

- RK3288:

```
&route_dsi0 {  
    status = "okay";  
};  
  
&route_lvds {  
    status = "okay";  
};
```

### 2.6.16 EDP + LVDS/RGB (3399 不支持 LVDS/RGB, 需要外接转换 IC)

对于双 LCD 显示, 我们建议先单独调试好 EDP 屏和 LVDS/RGB 屏, 再去做双 LCD 显示。这样步骤流程更清晰, 遇到问题也便于定位。

参考 dts:

暂缺。

注: 3288 需要先配置 build.prop, 关于 build.prop 配置, 具体见 3.1 章节, 调试阶段可以按如下所示做配置:

```
adb root  
adb remount  
adb pull system/build.prop  
修改这个文件, 添加  
sys.hwc.device.primary=eDP /*注意 e 要小写*/  
sys.hwc.device.extend=LVDS  
然后  
adb push build.prop system  
adb shell "chmod 644 system/build.prop"  
adb reboot
```

#### 2.6.16.1 开启 EDP 以及配置 EDP panel

保持事先调试好的单 EDP 屏的配置即可。

- RK3288:

```
&edp {
```

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

```
&edp_panel {
    compatible = "simple-panel";
    backlight = <&backlight>;
    enable-gpios = <&gpio7 4 GPIO_ACTIVE_HIGH>;
    enable-delay-ms = <120>;
    pinctrl-0 = <&lcd_cs>;
    power-supply = <&vcc_lcd>;
    status = "okay";

    display-timings {
        native-mode = <&timing0>;

        timing0: timing0 {
            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>;
        };
    };
};
```

### 2.6.16.2 开启 LVDS

- RK3288:

```
&lvds {
    status = "okay";
```



```
};
```

### 2.6.16.3 配置 LVDS panel

- RK3288:

```
&lvds_panel {
    compatible = "simple-panel";
    bus-format = <MEDIA_BUS_FMT_RGB666_1X18>;
    enable-gpios = <&gpio7 4 GPIO_ACTIVE_HIGH>;
    enable-delay-ms = <10>;
    power-supply = <&vcc_lcd>;
    rockchip,data-mapping = "jeida";
    rockchip,data-width = <24>;
    rockchip,output = "lvds";
    status = "okay";

    display-timings {
        native-mode = <&lvds_panel_name>;

        lvds_panel_name: timing0 {
            clock-frequency = <71000000>;
            hactive = <1280>;
            vactive = <800>;
            hfront-porch = <18>;
            hsync-len = <10>;
            hback-porch = <100>;
            vfront-porch = <6>;
            vsync-len = <2>;
            vback-porch = <8>;
            hsync-active = <0>;
            vsync-active = <0>;
            de-active = <0>;
            pixelclk-active = <0>;
        };
    };
};
```

**Note: backlight** 如果两个 panel 共用同样的 gpio 和 pwm，那么就无需添加，不然会资源冲突。如果不同，那么 **backlight** 后面的**&backlight** 需要和 **edp panel** 中的背光命名区分。

Backlight 的详细配置见 2.3.16.4。

**enable-gpios** 如果两个 panel 共用同样的 gpio，则无需添加该行。

#### 2.6.16.4 配置 LVDS 背光 backlight

我们 sdk 上 LVDS 和 EDP 共用背光设置，所以无需进行下面的配置，当然进行下面的配置也是可以正常显示的。下面的配置是给各自有独立背光的 panel 设定的。

- RK3288:

```
backlight1: backlight1 {
    compatible = "pwm-backlight";
    pwms = <&pwm1 0 25000 0>;
    brightness-levels = <
        0  1  2  3  4  5  6  7 8  9 10 11 12 13 14 15
        16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
        32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
        48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
        64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
        80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
        96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111
        112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127
        128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
        144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159
        160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175
        176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191
        192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207
        208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223
        224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239
        240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
    >;
    default-brightness-level = <128>;
    enable-gpios = <&gpio7 2 GPIO_ACTIVE_HIGH>;
    pinctrl-names = "default";
    pinctrl-0 = <&bl_en>;
    pwms = <&pwm0 0 1000000 0>;
};
```

```
&pwm1 {
    status = "okay";
};
```

```
&backlight1 {
    status = "okay";
};
```

#### 2.6.16.5 配置 VOP 连接

- RK3288:

edp 接 vopb, lvds 接 vopl:

```
&route_edp {
    connect = <&vopb_out_edp>;
};
&route_lvds {
    connect = <&vopl_out_lvds>;
};
&edp_in_vopl {
    status = "disabled";
};
&edp_in_vopb {
    status = "okay";
};
&lvds_in_vopl {
    status = "okay";
};
&lvds_in_vopb {
    status = "disabled";
};
```

edp 接 vopl, lvds 接 vopb:

```
&route_edp {
    connect = <&vopl_out_edp>;
};
&route_lvds {
    connect = <&vopb_out_lvds>;
};
```

```
&edp_in_vopl {
    status = "okay";
};
&edp_in_vopb {
    status = "disabled";
};
&lvds_in_vopl {
    status = "disabled";
};
&lvds_in_vopb {
    status = "okay";
};
```

#### 2.6.16.6 开启 **uboot** 显示

- RK3288:

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

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

### 3. 显示框架配置

当前 SDK 的显示框架增加了一些系统属性，用于帮助客户能够根据需求配置显示。

#### 3.1 主副显示器配置

属性	功能说明
<code>sys.hwc.device.primary</code>	设置显示接口做为主显
<code>sys.hwc.device.extend</code>	设置显示接口做为副显

以上两个属性的配置可加在产品配置目录下的 `system.prop` 里（如 `device/rockchip/rk3368/rk3368_box/system.prop`）。

默认情况下(即以上属性未配置时)，不支持热拔插设备（如 CVBS/MIPI/LVDS 等）会作为主显，支持热插拔设备（如 HDMI/DP 等）会作为次显。

通常主、副显只配置一个显示接口，例如 RK3399 BOX SDK 默认采用的配置，HDMI 作为主显示，DP 作为副显示。

```
sys.hwc.device.primary=HDMI-A
sys.hwc.device.extend=DP
```

当主/副显配置多个显示接口时，优先使用支持热拔插的设备。例如 RK3368 BOX SDK 默认采用的配置：

```
sys.hwc.device.primary=HDMI-A,TV
```

当 HDMI 插入时，主显使用 HDMI 作为显示，HDMI 拔出时，主显使用 CVBS 作为显示。

**注意：**由于主显的 `framebuffer` 分辨率无法动态更改，所以有两个或以上设备作为主显时，最好设定一个主显的 `framebuffer` 分辨率。设置方法见章节 3.3。

关于接口名称可以参见 `hardware/rockchip/hwcomposer/drmresources.cpp` 里的定义：

```
struct type_name connector_type_names[] = {
    { DRM_MODE_CONNECTOR_Unknown, "unknown" },//未知接口
    { DRM_MODE_CONNECTOR_VGA, "VGA" },//VGA
    { DRM_MODE_CONNECTOR_DVII, "DVI-I" },//DVI，暂不支持
```

```
{ DRM_MODE_CONNECTOR_DVID, "DVI-D" },//DVI, 暂不支持
{ DRM_MODE_CONNECTOR_DVIA, "DVI-A" },//DVI, 暂不支持
{ DRM_MODE_CONNECTOR_Composite, "composite" },//不支持
{ DRM_MODE_CONNECTOR_SVIDEO, "s-video" },//S 端子
{ DRM_MODE_CONNECTOR_LVDS, "LVDS" },//LVDS
{ DRM_MODE_CONNECTOR_Component, "component" },//分量信号 YPbPr
{ DRM_MODE_CONNECTOR_9PinDIN, "9-pin DIN" },//不支持
{ DRM_MODE_CONNECTOR_DisplayPort, "DP" },//DP
{ DRM_MODE_CONNECTOR_HDMIA, "HDMI-A" },//HDMI A 型口
{ DRM_MODE_CONNECTOR_HDMIB, "HDMI-B" },//HDMI B 型口, 不支持
{ DRM_MODE_CONNECTOR_TV, "TV" },// CVBS
{ DRM_MODE_CONNECTOR_eDP, "eDP" },//EDP
{ DRM_MODE_CONNECTOR_VIRTUAL, "Virtual" },//不支持
{ DRM_MODE_CONNECTOR_DSI, "DSI" },//MIPI
};
```

### 3.2 主副显示器接口查询

可以通过以下两个只读属性来分别查询主副显示器的输出接口的名称。

表 3-1 主副显示器查询

属性	功能说明
sys.hwc.device.main	查询当前主显的输出接口
sys.hwc.device.aux	查询当前副显的输出接口

### 3.3 FrameBuffer 分辨率配置

可以通过配置以下属性来设置 FrameBuffer 的分辨率：

```
persist.sys.framebuffer.main=1920x1080
```

### 3.4 分辨率过滤配置

因为初始获取到的全部分辨率过多，有些分辨率对用户来说并不需要，因此在 SDK 的 HWC 模块中对分辨率进行了过滤。

位于 device/rockchip/common/resolution\_white.xml 路径的配置文件定义了能够通过过滤的白名单，HWC 中会根据该配置文件对初始的分辨率进行过滤筛选后再传递给上层，该 XML 文件的每一个<resolution>块定义了一个能够通过过滤的分辨率，其中详细项的定义如下：

表 3-2 分辨率过滤项定义说明

项定义	说明
clock	时钟
hdisplay	行有效像素，见图 6-1 的标示
hsync_start	行同步起始像素，见图 6-1 的标示
hsync_end	行同步结束像素，见图 6-1 的标示
htotal	一行总像素，见图 6-1 的标示
hskew	见图 6-1 的标示
vdisplay	帧有效行，见图 6-1 的标示
vsync_start	帧同步开始行，见图 6-1 的标示
vsync_end	帧同步结束行，见图 6-1 的标示
vtotal	一帧总行数，见图 6-1 的标示
vscan	见图 6-1 的标示
vrefresh	显示设备帧率
flags	flags 的定义如下： DRM_MODE_FLAG_PHSYNC (1<<0) DRM_MODE_FLAG_NHSYNC (1<<1) DRM_MODE_FLAG_PVSYNC (1<<2) DRM_MODE_FLAG_NVSYNC (1<<3) DRM_MODE_FLAG_INTERLACE (1<<4)
vic	HDMI 标准对应定义的 VIC 值，如 HDMI 标准中未定义置 0

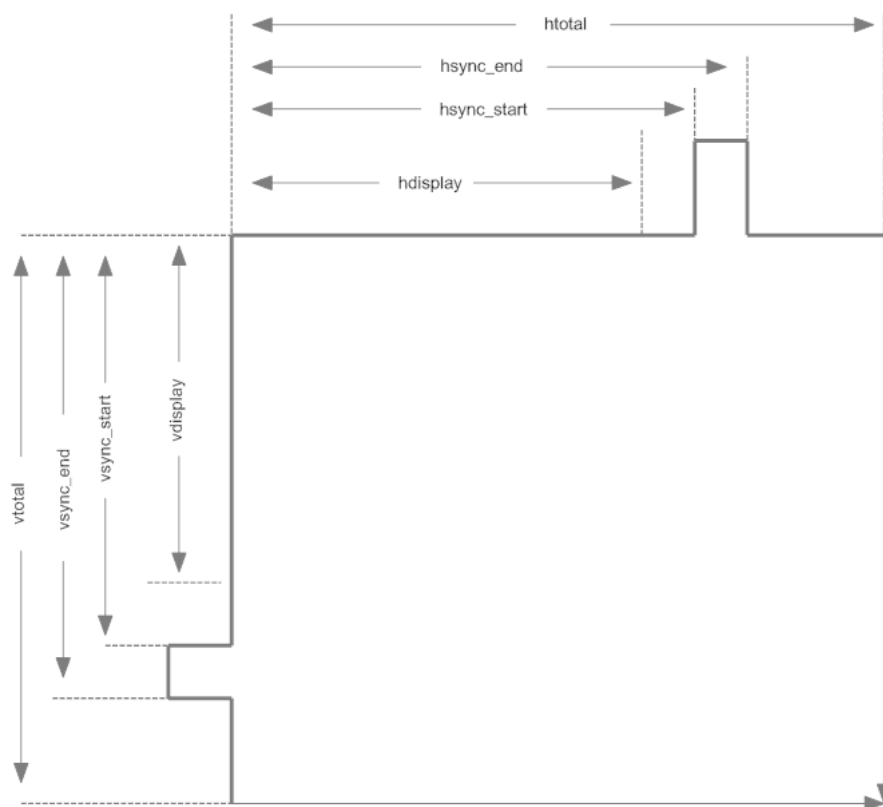


图 3-1 分辨率项定义示意图



## 4. 常用调试方法

### 4.1 查看 VOP 状态

```
cat /d/dri/0/summary
rk3399 box:/ # cat /d/dri/0/summary
VOP [ff900000.vop]: ACTIVE          VOP状态
Connector: HDMI-A
  bus_format[0x2025] output_mode[0xf]
Display mode: 1920x1080p60
  clk[148500] real_clk[148500] type[48] flag[5]
  H: 1920 2008 2052 2200
  V: 1080 1084 1089 1125      Connector状态
win0-0: ACTIVE
  format: NV12 little-endian (0x3231564e)
  zpos: 0
  src: pos[0x0] rect[3840x1080]
  dst: pos[0x0] rect[1920x1080]
  buf[0]: addr: 0x000000000ebc7000 pitch: 7680 offset: 0
  buf[1]: addr: 0x000000000ebc7000 pitch: 7680 offset: 8294400
win1-0: DISABLED
win2-0: ACTIVE
  format: AB24 little-endian (0x34324241)
  zpos: 1
  src: pos[0x0] rect[29x37]
  dst: pos[385x543] rect[29x37]
  buf[0]: addr: 0x0000000001abb000 pitch: 128 offset: 0
win2-0: DISABLED
win2-1: DISABLED
win2-2: DISABLED
win3-0: DISABLED
win3-0: DISABLED
win3-1: DISABLED
win3-2: DISABLED
VOP [ff8f0000.vop]: DISABLED      VOP状态
```

图 4-1

图 4-1 是 RK3399 连接 HDMI 时上述命令输出的 Log，可以提供三种信息：

- VOP 状态： VOPB 处于使能状态，VOPL 处于禁用状态。
- VOP 对应的 Connector 状态： VOPB 输出信号给 HDMI，bus\_format = 0x2025 表示 YUV444 8bit， output\_mode = 0x0f 表示 VOP 输出总线为 ROCKCHIP\_OUT\_MODE\_AAAA，输出 1920x1080P60。

常用的 bus\_format 由内核 uapi/linux/media-bus-format.h 定义：

```
#define MEDIA_BUS_FMT_RGB888_1X24      0x100a //RGB888
```

```
#define MEDIA_BUS_FMT_RGB101010_1X30      0x1018
//RGB101010
#define MEDIA_BUS_FMT_YUV8_1X24            0x2025 //YUV444
8bit
#define MEDIA_BUS_FMT_YUV10_1X30           0x2016 //YUV444
10bit
#define MEDIA_BUS_FMT_UYYVYY8_0_5X24       0x2026 //YUV420
8bit
#define MEDIA_BUS_FMT_UYYVYY10_0_5X30      0x2027
//YUV420 10bit
```

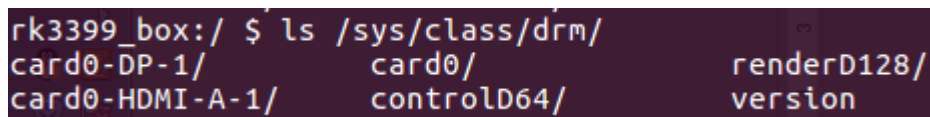
常用的 output\_mode 由内核 drivers/gpu/drm/rockchip/rockchip\_drm\_vop.h 定义:

```
#define ROCKCHIP_OUT_MODE_P888      0
#define ROCKCHIP_OUT_MODE_P666      1
#define ROCKCHIP_OUT_MODE_P565      2
#define ROCKCHIP_OUT_MODE_S888      8
#define ROCKCHIP_OUT_MODE_S888_DUMMY 12
#define ROCKCHIP_OUT_MODE_YUV420    14
/* for use special outface */
#define ROCKCHIP_OUT_MODE_AAAA      15
```

- 图层配置信息: win0 和 win2 使能, win2 buffer 格式为 ARGB, buffer 大小为 29x37; 目标窗口为 29x37, 窗口左上角坐标 (385, 543)。Win0 buffer 格式为 NV12, 大小为 3840x2160; 目标窗口大小为 1920x1080, 窗口左上角坐标 (0, 0)。

## 4.2 查看 Connector 状态

/sys/class/drm 目录下可以看到驱动注册的各个显卡。图 4-2 是 RK3399 BOX 平台 drm 目录结构, 可以看到注册了 card0-HDMI-A-1 和 card0-DP-1 两种输出, 分别表示 HDMI 和 DP。



```
rk3399_box:/ $ ls /sys/class/drm/
card0-DP-1/      card0/          renderD128/
card0-HDMI-A-1/  controlD64/     version
```

图 4-2

以 card0-HDMI-A-1 为例, 其目录下有以下文件:

- enabled 使能状态
- status 连接状态

- mode 当前输出分辨率
- modes 连接设备支持的分辨率列表
- audioformat 连接设备支持的音频格式
- edid 连接设备的 EDID，可以通过命令 `cat edid > /data/edid.bin` 保存下来。

## 4.3 查看 HDMI 状态

- 查看当前输出状态

```
cat /d/dw-hdmi/status
```

HDMI 状态打印如图 4-3 所示。

```
HDMI Output Status: PHY disabled

HDMI Output Status: PHY enabled
Pixel Clk: 148500000Hz      TMDs Clk: 148500000Hz
Color Format: YUV444        Color Depth: 8 bit
Colorimetry: ITU.BT709     EOTF: SDR
x0: 0                      y0: 0
x1: 0                      y1: 0
x2: 0                      y2: 0
white x: 0                 white y: 0
max lum: 0                 min lum: 0
max cll: 0                 max fall: 0
```

图 4-3

- HDMI Output Status 表示当前 PHY 状态，只有当 PHY 使能的时候才会有后续打印。
- Pixel Clk 表示当前输出的像素时钟
- TMDs Clk 表示当前输出 HDMI 符号率
- Color Format 表示输出的颜色格式，取值 RGB、YUV444、YUV422、YUV420。
- Color Depth 表示输出的颜色深度，取值 8bit、10bit、12bit、16bit。
- Colorimery 表示输出的颜色标准，取值 ITU.BT601、ITU.BIT709、ITU.BT2020。
- EOTF 表示输出的 HDR 电光转换曲线方式，有如下取值：

EOTF	含义
Unsupported	HDMI 不支持发送 HDR 信息
Not Defined	未定义

Off	不发送 HDR 信息
SDR	采用 SDR 曲线
ST2084	采用 ST2084 EOTF 曲线
HLG	采用 HLGEOTF 曲线

- (x0, y0)、(x1, y1)、(x2, y2)、(white x, white y)、max lum、min lum、max cll、maxfall

为静态 HDR 描述子信息，只有 EOTF 值为 SDR、ST2084、HLG 值时才会存在。

注意：内核代码需包含下面提交才可以支持查看状态。

```
commit eaca91814449199b1e6ad0b9fe0bba2215497c97
Author: Zheng Yang <zhengyang@rock-chips.com>
Date: Mon Nov 27 16:56:21 2017 +0800
```

```
drm: bridge: dw-hdmi: add hdmi status debugfs node
```

- 查看控制器寄存器

```
cat /d/dw-hdmi/ctrl
```

可以使用命令来修改寄存器，例如要修改 0x1000 寄存器为 0xF8，输入命令：

```
echo 1000 f8 > /d/dw-hdmi/ctrl
```

- 查看 PHY 寄存器

```
cat /d/dw-hdmi/phy
```

修改 PHY 寄存器与控制器类似，例如修改 0x06 寄存器为 0x8002，输入命令：

```
echo 06 8002 > /d/dw-hdmi/phy
```

注意：本命令只适用于 RK3288、RK3368、RK3399。

## 4.4 命令行设置分辨率

通过 persist.sys.resolution.main 以及 persist.sys.resolution.aux 设置主副屏分辨率，每次设置完更新 sys.display.timeline(每次加 1)使分辨率生效，例子如下：

- 设置 4k60:

```
setprop persist.sys.resolution.main 3840x2160@60-3840-4016-4104-4400-2160-2168-2178-2250-5
```

```
setprop sys.display.timeline 1
```

- 设置 1080p60:

```
setprop persist.sys.resolution.main 1920x1080@60-1920-2008-2052-2200-1080-1084-1089-1125-5
```

```
setprop sys.display.timeline 2
```

- 设置 720P60:

```
setprop persist.sys.resolution.main 1280x720@60.00-1390-1430-1650-725-730-750-5
```

```
setprop sys.display.timeline 3
```

- 设置 480P60:

```
setprop persist.sys.resolution.main 720x480@59.94-736-798-858-489-495-525-a
```

```
setprop sys.display.timeline 4
```

## 5. 参考文档

- 1 [https://markyzq.gitbooks.io/rockchip\\_drm\\_integration\\_helper/content/zh/](https://markyzq.gitbooks.io/rockchip_drm_integration_helper/content/zh/)
- 2 Android drm 显示框架介绍.ppt
- 3 kernel 的 Documentation/devicetree 文档