密级状态：绝密（　　）　秘密（　　）　内部（　）　公开（ √ ）

Rockchip 基于 DRM 框架的 HDMI 开发指南

（技术部，第二系统产品部）

文件状态： [　] 正在修改 [√] 正式发布	当前版本：	V1.0
	作　　者：	陈顺庆
	完成日期：	2017-12-14
	审　　核：	
	完成日期：	

福州瑞芯微电子股份有限公司

Fuzhou　Rockchips　Semiconductor　Co . , Ltd

版 本 历 史

版本号	作者	修改日期	修改说明	备注
V1.0	陈顺庆	2017-12-14	发布初始版本	
V1.1	陈顺庆	2018-3-21	补丁 5.2 设置 HDMI 旋转说明	

目 录

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，适用于以下 SDK：

- RK3399 Android7.1 SDK
- RK3368H Android7.1 SDK/ RK3368H Android8.1 SDK
- RK3126C Android8.1 SDK
- RK3288 Android7.1 SDK
- RK3399 Linux SDK

2. HDMI/DP 相关配置

2.1 HDMI 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。

2.1.3 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。

2.1.4 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"
};
```

2.1.5 绑定 PLL（只有 RK3399 需要）

rk3399 的 hdmi 所绑定的 vop 时钟需要挂载到 vpll 上，若是双显，需将另一个 vop 时钟挂到 cpll，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 HDMI 相关配置说明

2.2.1 信号强度配置(RK3288/RK3368/RK3399)

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 rockchip,phy-table。

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 rockchip,phy-table：

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)以下时，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。

2.2.2 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>;
}
```

2.2.3 打开音频

RK3399 目前 HDMI 声卡和 DP 公用：

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

2.3 DP 配置说明（RK3399）

2.3.1 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";
        };
};
```

2.3.2 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";
};
```

2.3.3 注册 DP 驱动

打开 dp 的 dts 同时绑定 extcon 配置：

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

2.3.4 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 口是的配置，若是接到 typec1 口则需将 tcphy0 改为 tcphy1 如下：
```

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

2.4 参考配置

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

2.4.2 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>;
};
```

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

2.4.4 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 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>;
};
```

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

表 3-1 主副显示器分辨率设置

属性	功能说明
sys.hwc.device.primary	设置显示接口做为主显
sys.hwc.device.extend	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_mid/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 MID SDK 默认采用的配置，HDMI 作为主显示，EDP 作为副显示。

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

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：

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

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 3-2 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 3-1 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 3-1 的标示
hsync_start	行同步起始像素，见图 3-1 的标示
hsync_end	行同步结束像素，见图 3-1 的标示
htotal	一行总像素，见图 3-1 的标示
hskew	见图 3-1 的标示
vdisplay	帧有效行，见图 3-1 的标示
vsync_start	帧同步开始行，见图 3-1 的标示
vsync_end	帧同步结束行，见图 3-1 的标示
vtotal	一帧总行数，见图 3-1 的标示
vscan	见图 3-1 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

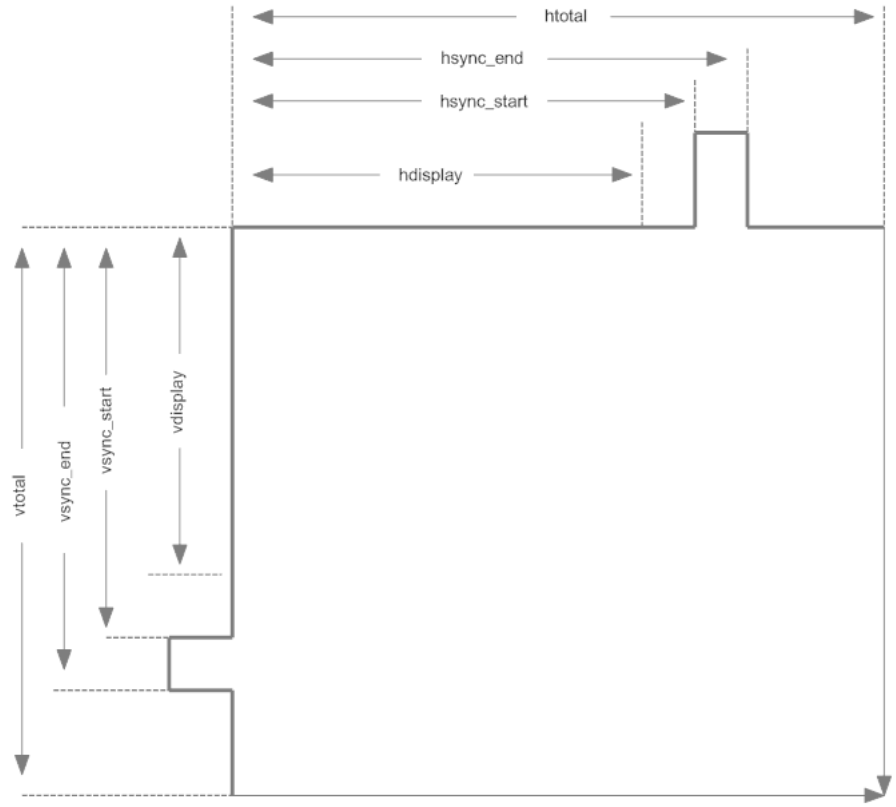


图 3-1 分辨率项定义示意图

4. 常用调试方法

4.1 查看 VOP 状态

```
cat /d/dri/0/summary
```

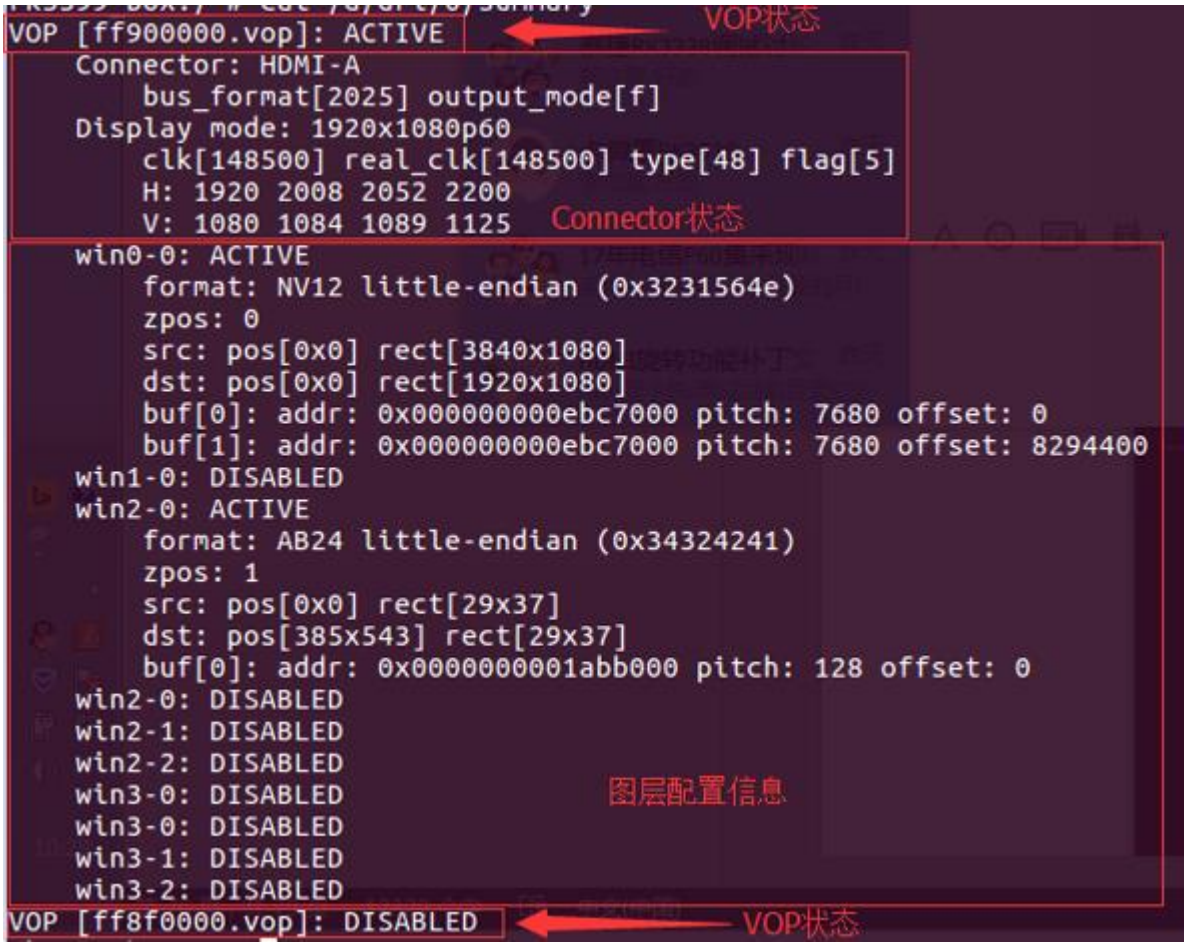


图 4-1

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输出接口，表 4-1 列出了 RK 平台上常见的

输出名称。

表 4-1

名称	类型
card0-DP-1	DP
card0-eDP-1	EDP
card0-HDMI-A-1	HDMI
card0-TV-1	CVBS
card0-LVDS-1	LVDS
card0-DSI-1	MIPI DSI

/sys/class/drm 是 RK3399 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。

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
```

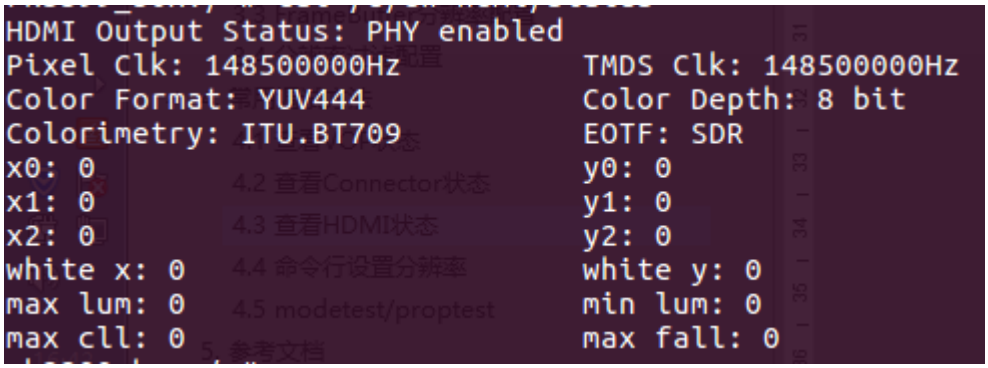


图 4-3

- 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。

4.4 命令行设置分辨率

- 当前可用显示分辨率列表：

  cat /sys/devices/platform/display-subsystem/drm/card0/card0-HDMI-A-1/mode

s

- 查看当前显示分辨率：

  cat /sys/devices/platform/display-subsystem/drm/card0/card0-HDMI-A-1/mode

- 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

    setprop sys.display.timeline 1

  - 设置 1080p60：

    setprop persist.sys.resolution.main 1920x1080@60

    setprop sys.display.timeline 2

  - 设置 720P60：

    setprop persist.sys.resolution.main 1280x720@60

    setprop sys.display.timeline 3

  - 设置 480P60：

    setprop persist.sys.resolution.main 720x480@60

    setprop sys.display.timeline 4

5. Q&A

5.1 EDID 没有读到的情况下怎么设置默认分辨率

```
Commit 727e0fe68d8f422698f4e257cb7c04f90b8692c0
Author: xuhuicong xhc@rock-chips.com
Date:   Tue Sep 26 17:32:56 2017 +0800

drm/edid: output common tv resolution and hdmi mode if no read the correct edid
```

```
Change-Id: Ib7379340e8c1d59382553d21b60165fe5fb371e8
Signed-off-by: xuhuicong xhc@rock-chips.com
```

在有上面的提交基础上，修改 def_modes 的值，对应的是 vic 值，如果 4 对应的是 edid_cea_modes 中的：

```
/* 4 - 1280x720@60Hz */
{ DRM_MODE("1280x720", DRM_MODE_TYPE_DRIVER, 74250, 1280, 1390,
            1430, 1650, 0, 720, 725, 730, 750, 0,
            DRM_MODE_FLAG_PHSYNC | DRM_MODE_FLAG_PVSYNC),
  .vrefresh = 60, .picture_aspect_ratio = HDMI_PICTURE_ASPECT_16_9, },
```

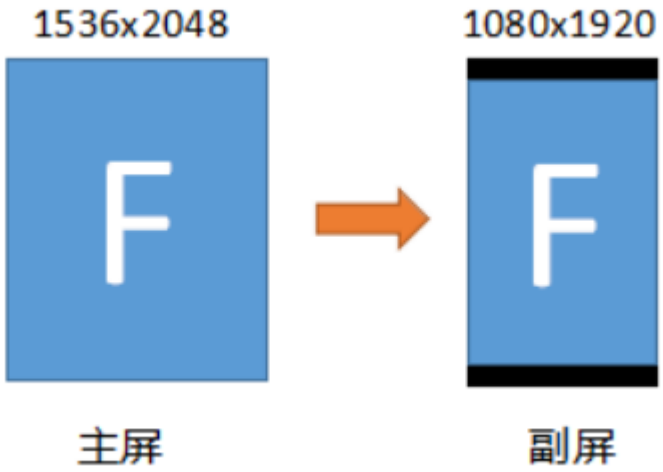
5.2 如何设置 HDMI 旋转

5.2.1 属性说明

在 system/build.prop 文件里面加上如下属性：

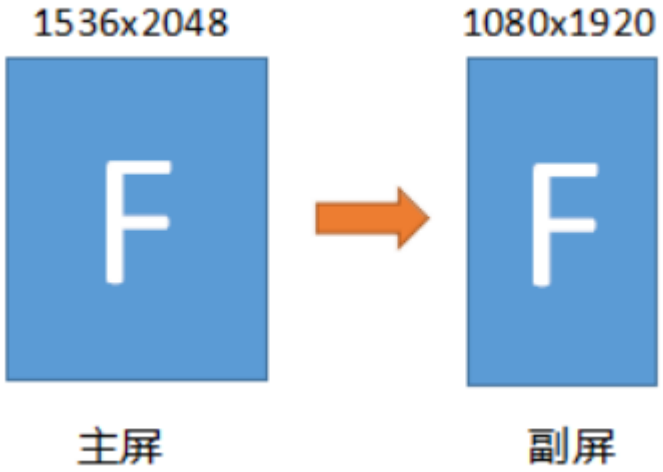
- 主屏的旋转方向，ro.sf.hwrotation=x【x=[0,90,180,270]】
- 如果主屏如(LVDS 或者 MIPI)和 hdmi 的物理属性（横竖屏）一致的话，加上 ro.same.orientation=true

  true：副屏图像完全参考主屏宽高比进行缩放， 若宽高比不一致， 则出现黑边，如下图：

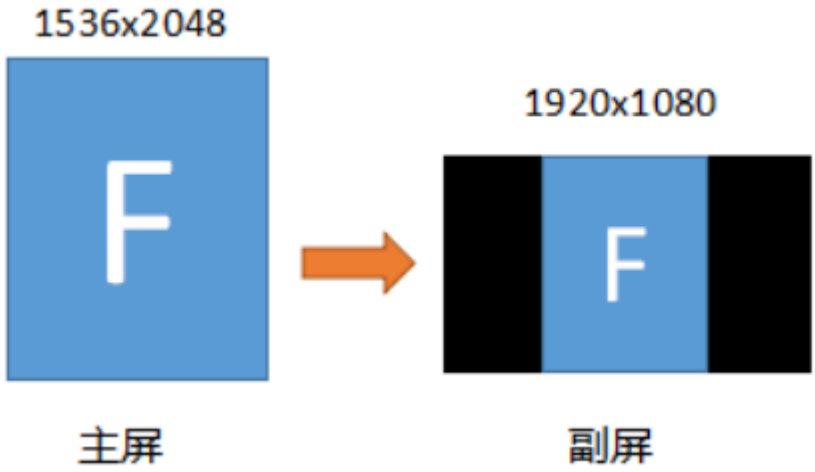


  false：副屏图像参考主屏横竖屏状态， **若副屏与主屏横竖屏状态一致**（ 即主副屏

frame.width / frame.hight > 1 或 均<1） ，在主副屏宽高比不一致的情况下拉伸图像， 铺满全屏,如下图：

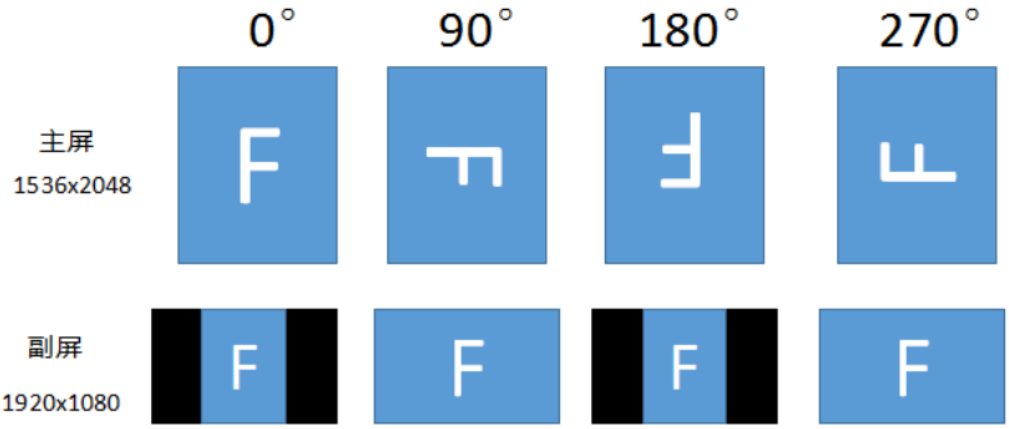


false：**若副屏与主屏横竖屏状态不一致，**例如主屏为竖屏图像， 副屏为横屏， 则按主屏宽高进行缩放， 出现黑边情况，如下图：

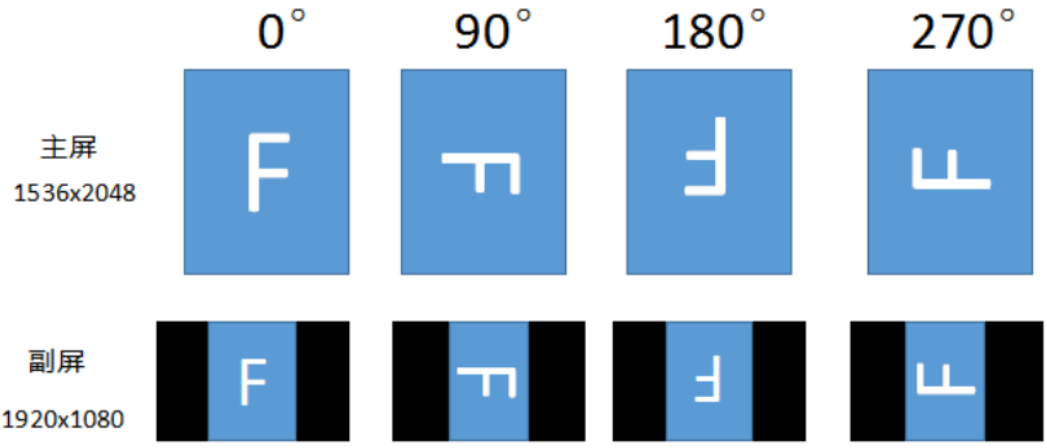


- 如果需要初始化次屏的方向，需要设置 ro.orientation.einit=x【x=[0,90,180,270]】
- 如果需要次屏也随着 g-sensor 旋转，可以设置 ro.rotation.external=true

false：副屏方向不随主屏 g-sensor 旋转， 即主屏旋转到任意方向， 副屏始终保持一个方向：



true：副屏方向随着主屏 g-sensor 旋转， 并与主屏旋转方向一致：

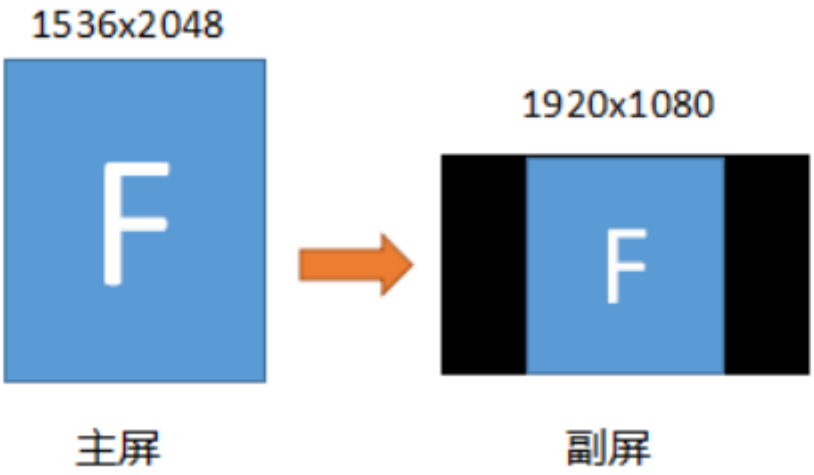


5.2.2 应用场景

①

```
ro.sf.hwrotation=0
ro.same.orientation=false
ro.orientation.einit=0
ro.rotation.external=false
```

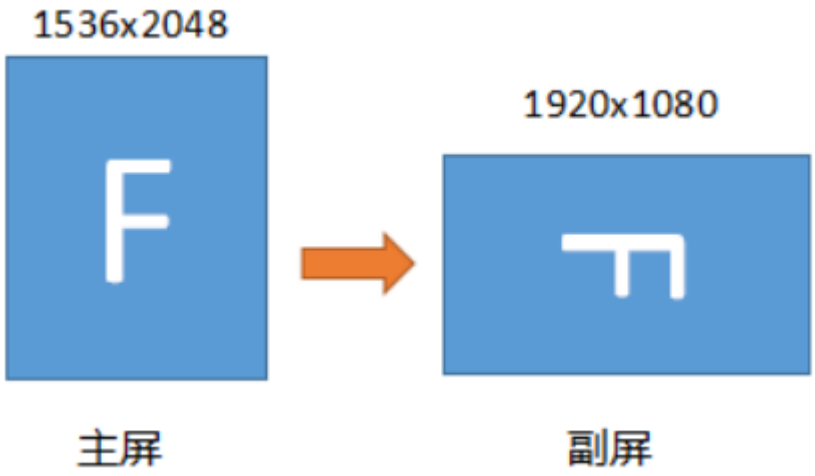
显示情况如下图所示：



②

```
ro.sf.hwrotation=0
ro.same.orientation=false
ro.orientation.einit=90
ro.rotation.external=false
```

显示情况如下图所示：

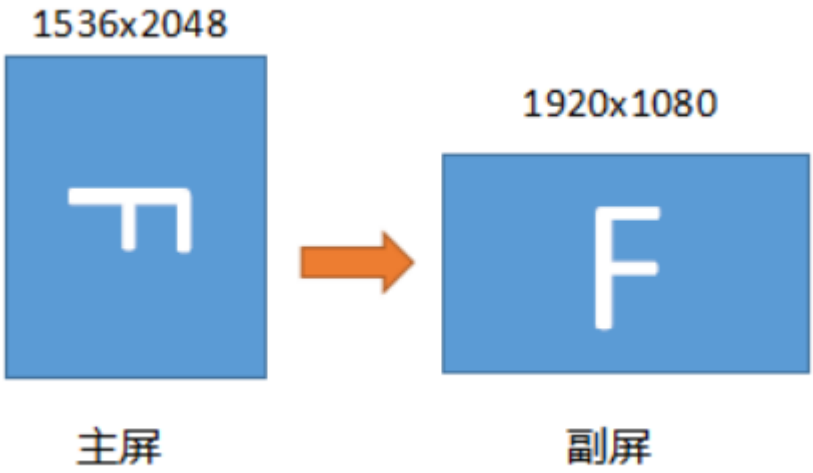


③

```
ro.sf.hwrotation=90
ro.same.orientation=false
ro.orientation.einit=0
```

```
ro.rotation.external=false
```

显示情况如下图所示：



④

```
ro.sf.hwrotation=90
ro.same.orientation=false
ro.orientation.einit=90
ro.rotation.external=false
```

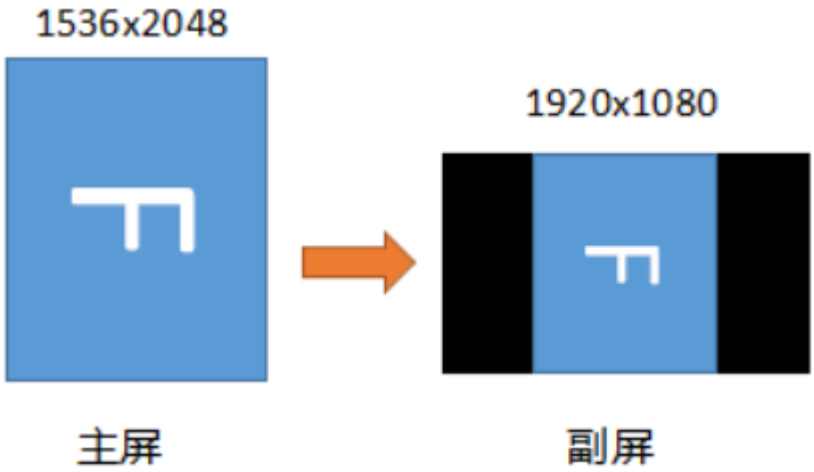
这个场景需要修改 patch 代码

native: isSfHwrotated = atoi(value) != 90;

base:boolean noRotate = "90".equals(SystemProperties.get("ro.sf.hwrotation"));

备注： 主屏旋转的本质， 是更改副屏接收到的图像数据类型， 横屏比例数据还是竖屏比例的数据。

修改后显示如下图：



5.2.3 patch

```
diff --git a/services/core/java/com/android/server/display/LocalDisplayAdapter.java
b/services/core/java/com/android/server/display/LocalDisplayAdapter.java
index 5ebe64d..8a52738 100644
--- a/services/core/java/com/android/server/display/LocalDisplayAdapter.java
+++ b/services/core/java/com/android/server/display/LocalDisplayAdapter.java
@@ -203,6 +203,22 @@ final class LocalDisplayAdapter extends DisplayAdapter {
                 } else {
                     mInfo.type = Display.TYPE_HDMI;
                     mInfo.flags |= DisplayDeviceInfo.FLAG_PRESENTATION;
+                    boolean noRotate =
"0".equals(SystemProperties.get("ro.sf.hwrotation"));// 1
+                    if(noRotate && mBuiltInDisplayId ==
SurfaceControl.BUILT_IN_DISPLAY_ID_HDMI){
+                        if (SystemProperties.getBoolean("ro.rotation.external",
false)) {
+                            mInfo.flags |=
DisplayDeviceInfo.FLAG_ROTATES_WITH_CONTENT;
+                        }
+                        String value = SystemProperties.get("ro.orientation.einit");
+                        if ("0".equals(value)) {
+                            mInfo.rotation = Surface.ROTATION_0;
+                        } else if ("90".equals(value)) {
+                            mInfo.rotation = Surface.ROTATION_90;
+                        } else if ("180".equals(value)) {
+                            mInfo.rotation = Surface.ROTATION_180;
+                        } else if ("270".equals(value)) {
+                            mInfo.rotation = Surface.ROTATION_270;
+                        }
+                    }
                     mInfo.name = getContext().getResources().getString(

com.android.internal.R.string.display_manager_hdmi_display_name);
```

```
                    mInfo.touch = DisplayDeviceInfo.TOUCH_EXTERNAL;

--- a/services/surfaceflinger/DisplayDevice.cpp
+++ b/services/surfaceflinger/DisplayDevice.cpp
@@ -449,12 +449,73 @@ void DisplayDevice::setProjection(int orientation,
     ALOGV(" viewport [%d %d]",mViewport.getWidth(),mViewport.getHeight());
     ALOGV(" frame [%d %d]", frame.getWidth(),frame.getHeight());
     ALOGV(" hw [%d %d]", getWidth(),getHeight());
-    if(strcmp(getDisplayName().string(),"Built-in Screen")
-        && frame.getWidth() > frame.getHeight())
-    {
+
+    bool isVirtualScreen = mType == DisplayDevice::DISPLAY_VIRTUAL;
+    if (isVirtualScreen && frame.getWidth() > frame.getHeight()) {
         frame = Rect(0,0,getWidth(),getHeight());
         ALOGV("update frame [%d,%d]",frame.getWidth(),frame.getHeight());
     }
+
+    bool isHdmiScreen = mType == DisplayDevice::DISPLAY_EXTERNAL;
+    if (isHdmiScreen) {
+        int eInitOrientation = 0;
+        bool isSfHwrotated = false;
+        bool isSupportRotation = false;
+        bool isPrimaryExternalSameOrientation = false;
+        Rect newFrame = Rect(0,0,getWidth(),getHeight());
+        Rect newFrameRotated = Rect(0,0,getHeight(),getWidth());
+        float frameRatio = (float)frame.getWidth() / frame.getHeight();
+        char value[PROPERTY_VALUE_MAX];
+        property_get("ro.sf.hwrotation", value, "0");
+        isSfHwrotated = atoi(value) != 0;
+        property_get("ro.same.orientation", value, "false");
+        isPrimaryExternalSameOrientation = !strcmp(value,"true");  //1
+        if(!isSfHwrotated) {
+            property_get("ro.orientation.einit", value, "0");
+            eInitOrientation = atoi(value) / 90;
+            property_get("ro.rotation.external", value, "false");
+            isSupportRotation = !strcmp(value,"true"); //1
+        }
+        if (isSupportRotation && !isPrimaryExternalSameOrientation) {
+            mClientOrientation = orientation;
+            if (eInitOrientation % 2 == 1) {
+                frame = frameRatio > 1.0 ? frame : newFrameRotated;
+                ALOGI("%d,[%d,%d]",__LINE__,frame.getWidth(),frame.getHeight());
+            } else {
+                frame = frameRatio > 1.0 ? newFrame : frame;
+                ALOGI("%d,[%d,%d]",__LINE__,frame.getWidth(),frame.getHeight());
+            }
+        } else if (isSupportRotation) {
+            mClientOrientation = orientation;
+            if (eInitOrientation % 2 == 1) {
+                //frame = frameRatio > 1.0 ? frame : frame;
```

```
+                ALOGI("%d,[%d,%d]",__LINE__,frame.getWidth(),frame.getHeight());
+            } else {
+                frame = frameRatio > 1.0 ? newFrame : newFrameRotated;
+                ALOGI("%d,[%d,%d]",__LINE__,frame.getWidth(),frame.getHeight());
+            }
+        } else if (eInitOrientation % 2 != 0) {
+            if (isPrimaryExternalSameOrientation) {
+                //frame = frameRatio > 1.0 ? frame : frame;
+                ALOGI("%d,[%d,%d]",__LINE__,frame.getWidth(),frame.getHeight());
+            } else {
+                frame = frameRatio > 1.0 ? frame : newFrameRotated;
+                ALOGI("%d,[%d,%d]",__LINE__,frame.getWidth(),frame.getHeight());
+            }
+        } else if (eInitOrientation % 2 == 0) {
+            if (isPrimaryExternalSameOrientation) {
+                frame = frameRatio > 1.0 ? newFrame : frame;
+                ALOGI("%d,[%d,%d]",__LINE__,frame.getWidth(),frame.getHeight());
+            } else {
+                frame = frameRatio > 1.0 ? newFrame : frame;
+                ALOGI("%d,[%d,%d]",__LINE__,frame.getWidth(),frame.getHeight());
+            }
+        } else {
+            frame = frameRatio > 1.0 ? newFrame : frame;
+            ALOGI("%d,[%d,%d]",__LINE__,frame.getWidth(),frame.getHeight());
+        }
+        ALOGV("update frame [%d,%d]",frame.getWidth(),frame.getHeight());
+    }
 #endif
     if (mType == DisplayDevice::DISPLAY_PRIMARY) {
         mClientOrientation = orientation;
```

5.3 如何设置 HDMI 缩放

如果 HDMI 是主屏：setprop persist.sys.overscan.main "overscan 100,100,100,100"

如果 HDMI 是副屏：setprop persist.sys.overscan.aux "overscan 100,100,100,100"

overscan 的 4 个参数分别指：left_margin，top_margin，right_margin，bottom_margin。