Rockchip DRM Display Driver 开发指南

文件标识：RK-YH-YF-455

发布版本：V3.0.0

日期：2022-03-17

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

本文主要介绍 Rockchip 平台处理器基于 DRM 显示框架 VOP 以及相关显示接口的基本特性、工作流程和常见问题分析。目的是为了相关工程师能对 DRM 显示驱动框架和硬件接口有更好的理解，并通过常见问题的分析能快速定位问题、解决问题。

产品版本

芯片名称	内核版本
RK3036	Linux kernel 4.4 及以上内核
RK312X/PX3SE	Linux kernel 4.4 及以上内核
RK3288	Linux kernel 4.4 及以上内核
RK322X/RK312XH	Linux kernel 4.4 及以上内核
RK3308	Linux kernel 4.4 及以上内核
RK322XH/RK332X	Linux kernel 4.4 及以上内核
RK3326/PX30	Linux kernel 4.4 及以上内核
RK3368/PX5	Linux kernel 4.4 及以上内核
RK3399	Linux kernel 4.4 及以上内核
RK1808	Linux kernel 4.4 及以上内核
RV1109/RV1126	Linux kernel 4.19 及以上内核
RK356X	Linux kernel 4.19 及以上内核
RK3588	Linux kernel 5.10 及以上内核
RV1103/RV1106	Linux kernel 5.10 及以上内核

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

硬件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	黄家钗	2019-1-5	初始版本
V2.0.0	黄家钗	2019-11-15	针对 Linux 4.19 更新
V3.0.0	黄家钗	2022-3-17	针对 RK356X/RK3588/RV1103/RV1106 以及 Linux kernel 5.10 更新

目录

Rockchip DRM Display Driver 开发指南
1. DRM 概述
   1.1 基本概念
   1.2 显示通路
   1.3 DRM 驱动和 libdrm 的交互过程
2. 驱动文件和加载流程
   2.1 U-Boot 驱动
      2.1.1 驱动目录
      2.1.2 驱动文件
      2.1.3 接口说明
   2.2 kernel 驱动
      2.2.1 驱动目录
      2.2.2 驱动文件
   2.3 驱动加载流程
3. Display feature
   3.1 各平台 VOP 基础特性和支持的显示接口
   3.2 各平台显示接口最大输出分辨率和协议标准
4. 硬件相关
   4.1 RGB输出/TTL模式硬件连接
      4.1.1 VOP 1.0 RGB 接口硬件连接方式
      4.1.2 VOP 2.0 及之后的版本 RGB 接口硬件连接方式
      4.1.3 不同 display mode index 对应的软件配置
      4.1.4 BT.656 和 BT.1120 的硬件连接方式
   4.2 LVDS Data Mapping
5. 扫描时序说明
   5.1 常见的扫描时序图
   5.2 DRM 对扫描时序的定义
   5.3 软件配置的对应关系和转换
   5.4 查看当前配置的时序
6. 带宽的计算方法
   6.1 图像的带宽
   6.2 显示接口的带宽
7. 常用的 debug 手段
   7.1 dump 当前的显示状态
      7.1.1 使用命令
      7.1.2 参数说明
   7.2 dump当前显示的 buffer
      7.2.1 使用说明
   7.3 调整 DRM 打印 Log 等级抓 Log
   7.4 查看当前显示时钟
   7.5 强行开关显示设备
   7.6 查看 DRM buffer 使用情况
   7.7 查看 GPIO 状态
   7.8 modetest 的使用
   7.9 显示进程的暂停、启动
   7.10 获取 EDID 信息
   7.11 查看 HDMI 状态
      7.11.1 参考例子
8. FAQ
   8.1 如何开关 U-Boot logo 显示
   8.2 如何配置 U-Boot logo 全屏或者居中显示
   8.3 U-Boot logo 要求
   8.4 U-Boot logo 切换到内核 logo 出现闪屏/无法显示问题
   8.5 VOP POST_BUF_EMPTY
   8.6 显示效果调节
   8.7 屏无法点亮/休眠唤醒显示异常/不显示问题
   8.8 RK3308 如何打开显示功能
   8.9 关闭 iommu 的方法
   8.10 各种接口屏配置

8.11 RGB/MCU 屏帧率计算问题
8.12 如何编写第三方转换芯片驱动
8.13 VOP2 绑定每个 VP 所使用的图层
8.14 RK3588 DSC 支持几个 slice，slice width 最大支持多少
8.15 RK3568 VP 和各显示接口的连接关系
8.16 RK3588 VP 和各显示接口的连接关系
8.17 超过 4kP60 对 aclk 的要求
9. 参考文档

1. DRM 概述

DRM 全称是 Direct Rendering Manager，进行显示输出管理、buffer 分配、帧缓冲。对应 userspace 库为 libdrm，libdrm 库提供了一系列友好的控制封装，使用户可以方便的进行显示的控制和 buffer 申请。DRM 的设备节点为 "/dev/dri/cardX"， X 为 0-15 的数值，默认使用的是 /dev/dri/card0。

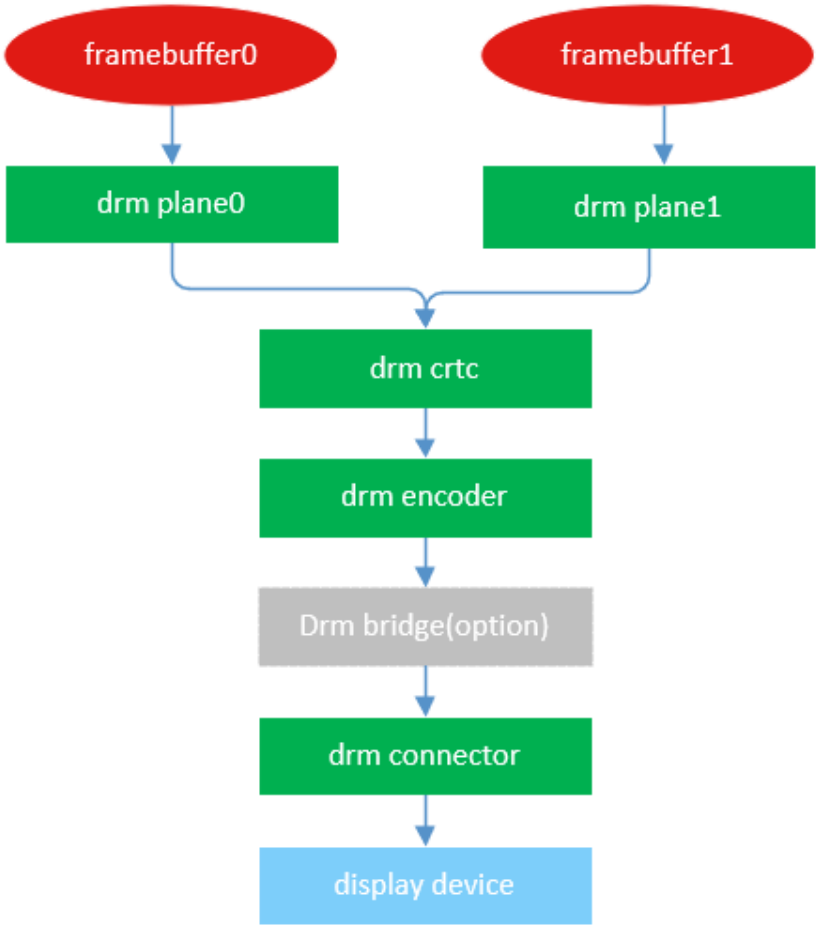
Rockchip 平台从 Linux 4.4 内核开始，显示驱动全部切到 DRM 显示框架。

1.1 基本概念

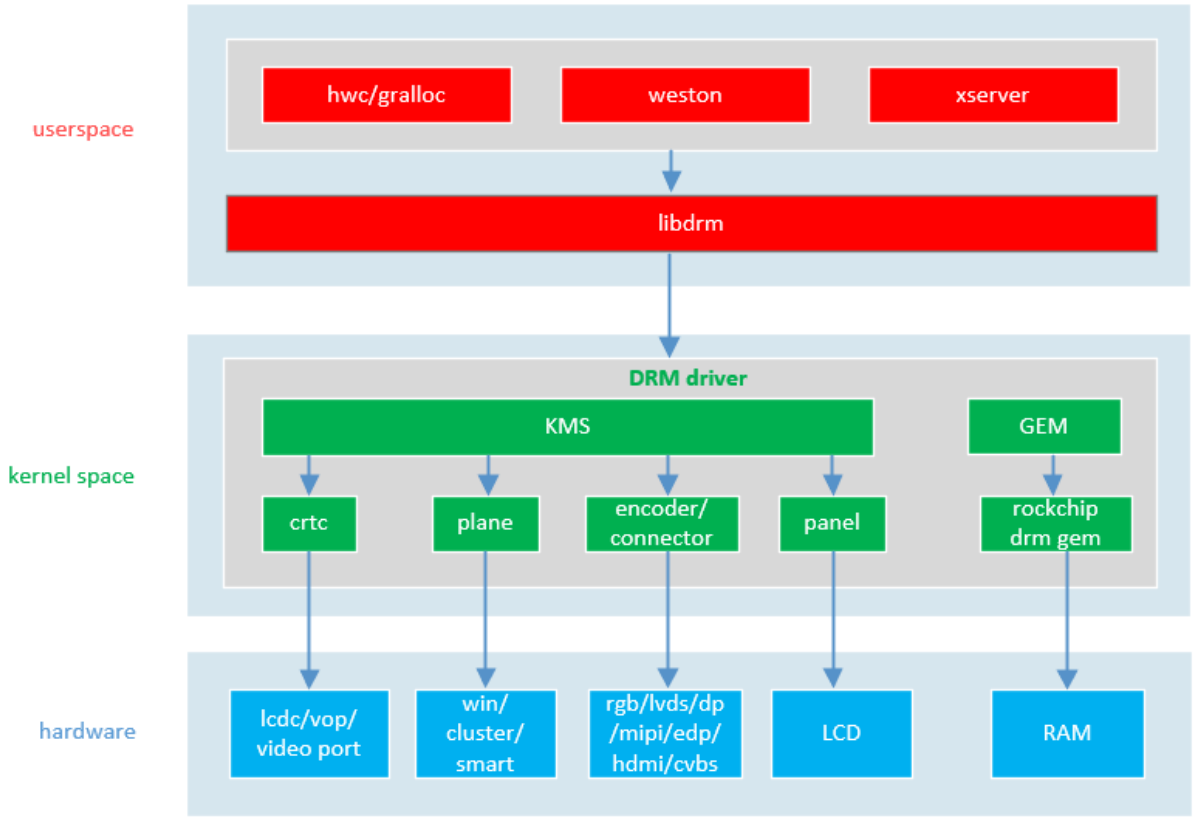
为了方便管理显示通路上的各种硬件模块，DRM 定义了以下几个概念：

基本概念	说明
CRTC	显示控制器，在 rockchip 平台是 SOC 内部 VOP（部分文档也称为 LCDC）模块或者 VOP2 中 Video Port 的抽象
Plane	图层，在 rockchip 平台是 SOC 内部 VOP（LCDC）模块 win 图层的抽象
Encoder	输出转换器，指RGB、LVDS、DSI、eDP、DP、HDMI、CVBS、VGA 等显示接口
Connector	连接器，指 encoder 和 panel 之间交互的接口部分
Bridge	桥接设备，一般用于注册 encoder 后面另外再接的转换芯片，如 DSI2HDMI 转换芯片
Panel	泛指屏，各种 LCD 显示设备的抽象
GEM	DRM 下 buffer 管理和分配，类似 ION、DMA BUFFER

1.2 显示通路



1.3 DRM 驱动和 libdrm 的交互过程



2. 驱动文件和加载流程

2.1 U-Boot 驱动

2.1.1 驱动目录

```
drivers/video/drm/
```

2.1.2 驱动文件

Driver	File
Core	rockchip_display.c rockchip_crtc.c rockchip_connector.c rockchip_phy.c rockchip_panel.c rockchip_bridge.c
VOP	rockchip_vop.c rockchip_vop_reg.c rockchip_vop2.c rockchip_vop2_reg.c
RGB	rockchip_rgb.c inno_video_combo_phy.c
LVDS	rockchip_lvds.c inno_video_combo_phy.c
MIPI-DSI	drm_mipi_dsi.c dw_mipi_dsi2.c inno_mipi_phy.c inno_video_combo_phy.c samsung_mipi_dcphy.c
eDP	rockchip_analogix_dp.c rockchip_analogix_dp_reg.c
HDMI	dw_hdmi.c rockchip_dw_hdmi.c rockchip-inno-hdmi-phy.c inno_hdmi.c dw_hdmi_qp.c rockchip_dw_hdmi_qp.c phy-rockchip-samsung-hdptx-hdmi.c
TVE /CVBS	rockchip_drm_tve.c
DP	dw-dp.c drm_dp_helper.c phy-rockchip-usbdp.c

以上驱动文件在不同的 U-Boot 版本可能会做些小的调整，但多数驱动文件和框架部分基本不会变化，查阅的时候可以根据实际情况调整。

2.1.3 接口说明

1. 显示 U-Boot logo

```
void rockchip_show_logo(void)
```

2. 显示指定的 bmp 图片，目前主要用于充电 logo 的显示

```
void rockchip_show_bmp(const char *bmp)
```

3. 将 U-Boot 中确定的一些变量通过修改 dtb 文件传递给内核，包括 kernel logo 的大小、地址、格式、输出扫描时序以及过扫描的配置等信息

```
void rockchip_display_fixup(void *blob)
```

2.2 kernel 驱动

2.2.1 驱动目录

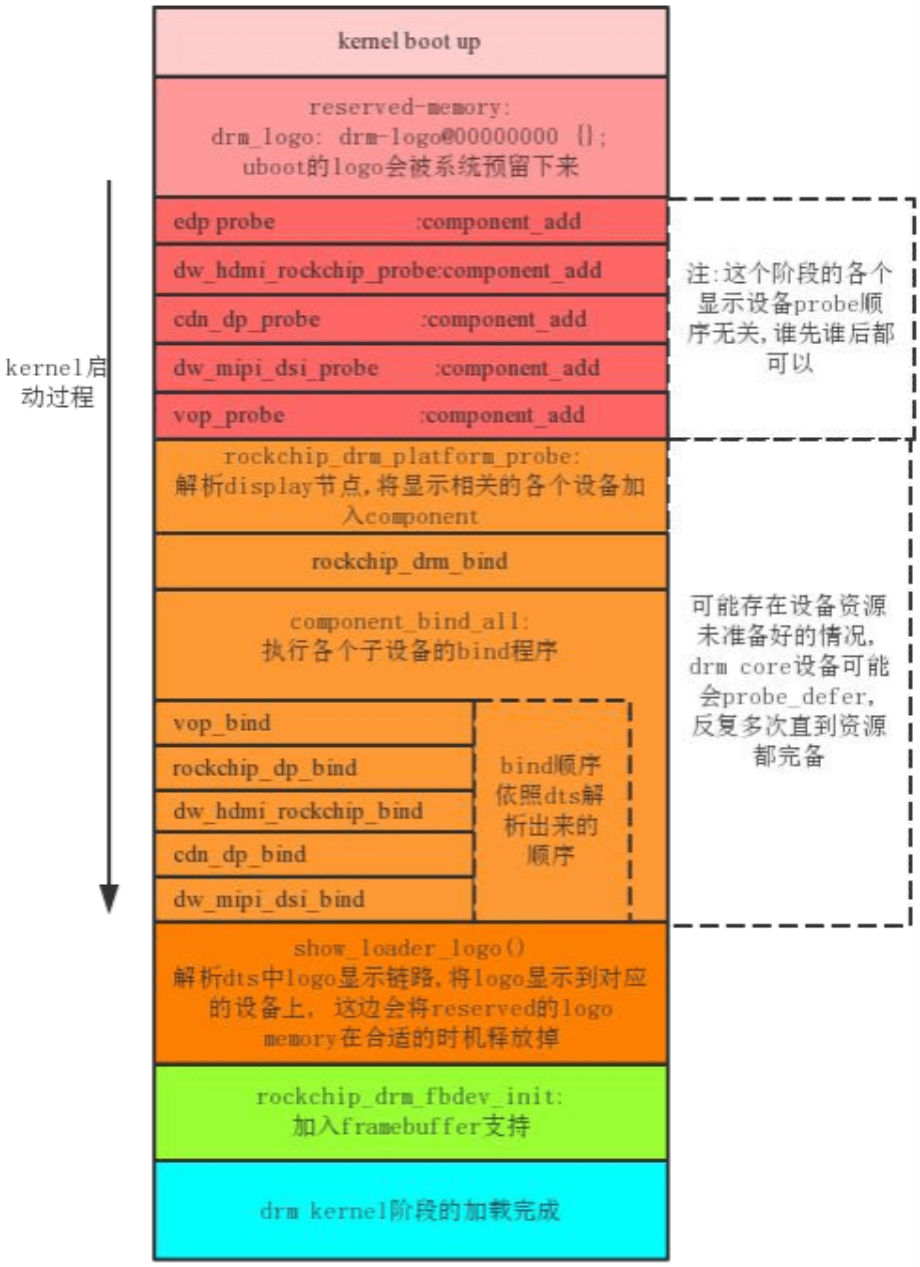
```
drivers/gpu/drm/rockchip/
drivers/gpu/drm/bridge/analogix/
drivers/gpu/drm/bridge/synopsys/
drivers/phy/rockchip/
```

2.2.2 驱动文件

Driver	File	Doc
Core	rockchip_drm_drv.c rockchip_drm_fb.c rockchip_drm_fbdev.c rockchip_drm_gem.c rockchip_drm_logo.c rockchip_drm_direct_show.c panel-simple.c	rockchip-drm.txt or rockchip-drm.yaml
VOP	rockchip_drm_vop.c rockchip_vop_reg.c rockchip_drm_vop2.c rockchip_vop2_reg.c	rockchip-vop.txt or rockchip-vop.yaml
RGB	rocckhip_rgb.c phy-rockchip-inno-video-combo-phy.c	rockchip-rgb.txt
LVDS	rockchip_lvds.c phy-rockchip-inno-video-combo-phy.c	rockchip-lvds.txt
MIPI-DSI	dw-mipi-dsi.c phy-rockchip-inno-dsidphy.c phy-rockchip-inno-video-combo-phy.c dw-mipi-dsi2-rockchip.c phy-rockchip-samsung-dcphy.c	dw_mipi_dsi_rockchip.txt phy-rockchip-inno-mipi-dphy.txt
eDP	analogix_dp_core.c analogix_dp-rockchip.c analogix_dp_reg.c phy-rockchip-dp.c	analogix_dp-rockchip.txt analogix_dp.txt rockchip-dp-phy.txt
DP	cdn-dp-core.c cdn-dp-reg.c dw-dp.c phy-rockchip-usbdp.c phy-rockchip-samsung-hdptx-hdmi.c	cdn-dp-rockchip.txt
HDMI	inno_hdmi.c dw-hdmi.c dw_hdmi-rockchip.c phy-rockchip-inno-hdmi-phy.c dw-hdmi-qp.c phy-rockchip-samsung-hdptx-hdmi.c	inno_hdmi-rockchip.txt dw_hdmi-rockchip.txt phy-rockchip-inno-hdmi-phy.txt
TVE/CVBS	rockchip_drm_tve.c	rockchip_drm_tve.txt

以上驱动文件在不同的 kernel 版本可能会做些小的调整，但多数驱动文件和框架部分基本不会变化，查阅的时候可以根据实际情况调整。

2.3 驱动加载流程



3. Display feature

3.1 各平台 VOP 基础特性和支持的显示接口

SOC	VOP version	VOP base feature										Display output interface											
		8K	4K	MMU	i-MODE	A/I FBDC	MULI AREA	BCSH	gamma	3D-LUT	POST SCAL	RGB	MCU	BT.656	BT.1120	LVDS	DUAL LVDS	MIPI	DUAL MIPI	EDP	DP	HDMI	CVBS
RK3066/PX2	V1.0	×	×	×	×	×	×	×	√	×	×	√	×	×	×	√	×	×	×	×	×	√	×
RK3188/PX3	V1.0	×	×	×	×	×	×	×	√	×	×	√	√	×	×	×	×	×	×	×	×	×	×
RK3126/RK3126C	V1.0	×	×	√	×	×	×	√	√	×	×	√	×	×	×	√	×	√	×	×	×	×	×
RK3128/PX3SE	V1.0	×	×	√	×	×	×	√	√	×	×	√	×	×	×	√	×	√	×	×	×	√	√
RK3036	V1.0	×	×	√	×	×	×	√	√	×	×	×	×	×	×	×	×	×	×	×	×	√	√
RK322X/RK312XH	V1.0	×	√	√	√	×	×	√	×	×	√	×	×	×	×	×	×	×	×	×	×	√	√
RK322XH/RK332X	V1.0	×	√	√	√	×	×	√	×	×	√	×	×	×	×	×	×	×	×	×	×	√	√
SOFIA 3GR	V1.0	×	×	√	×	×	×	√	√	×	×	√	×	×	×	√	×	√	×	×	×	×	×
RV1108	V1.0	×	×	×	√	×	×	√	√	×	×	√	×	×	×	×	×	√	×	×	×	√	√
RK3288	V1.0	×	√	√	×	×	√	√	√	×	√	√	×	×	×	√	√	√	√	√	×	√	×
RK3368/PX5	V1.0	×	√	√	×	√	√	√	√	×	√	√	×	×	×	√	×	√	√	√	×	√	×
RK3399	V1.0	×	√	√	√	√	√	√	√	×	√	×	×	×	×	×	×	√	√	√	√	√	×
RK3326/PX30	V1.0	×	×	√	√	√	√	√	√	×	×	√	√	×	×	√	×	√	×	×	×	×	×
RK3308	V1.0	×	×	×	×	×	×	√	√	×	×	√	√	×	×	×	×	×	×	×	×	×	×
RK1808	V1.0	×	×	√	×	×	×	√	√	×	×	√	√	×	×	×	×	√	×	×	×	×	×
RV1109/RV1126	V1.0	×	×	√	×	×	×	√	√	×	×	√	√	×	√	×	×	√	×	×	×	×	×
RK356X	V2.0	×	√	√	√	√	√	√	√	√	√	√	×	√	√	√	×	√	√	√	×	√	×
RK3588	V2.0	√	√	√	√	√	√	√	√	√	×	√	×	√	√	×	×	√	√	√	√	√	×
RV1103/RV1106	V1.0	×	×	×	×	×	×	√	√	×	×	√	√	√	√	×	×	×	×	×	×	×	×

3.2 各平台显示接口最大输出分辨率和协议标准

平台	显示接口	最大输出分辨率	协议标准
RK3036	HDMI	1920x1080@60hz	支持 HDMI 1.4a 协议标准
	CVBS	720x480i@60hz/720x576i@50hz	支持 NTSC/PAL 标准输出
RK312X/PX3SE	RGB	1280x800@60hz	支持 RGB666/sRGB888
	LVDS	1280x800@60hz	支持 VESA 和 JEIDA LVDS 数据格式
	MIPI	1280x800@60hz	支持 DSI v1.0，DCS v1.0，DPHY v1.0 协议标准
	HDMI	1920x1080@60hz	支持 HDMI 1.4a 协议标准
RK322X/RK312XH	HDMI	4096x2160@60hz	支持 HDMI 1.4a 和 2.0 协议标准
	CVBS	720x480i@60hz/720x576i@50hz	支持 NTSC/PAL 标准输出
RK3288	RGB	1920x1080@60hz	支持 GB888/RGB666/sRGB888
	LVDS	单通道：1280x800@60hz 双通道：1920x1080@60hz	支持 VESA 和 JEIDA LVDS 数据格式
	eDP	2560x1600@60hz	支持 DP1.2a 和 eDP1.3 协议标准
	MIPI	单通道：1920x1080@60hz 双通道：2560x1600@60hz	支持 DSI v1.1，DCS v1.1，DPHY v1.1 协议标准
	HDMI	VOP BIG:：3840x2160@60hz VOP LIT： 1920x1080@60hz	支持 HDMI 1.4a 和 2.0 协议标准
RK3308	RGB	1280x800@60hz	支持 RGB888/RGB666/sRGB888/MCU
RK1808	RGB	1280x800@60hz	支持 RGB888/RGB666/sRGB888/MCU
	MIPI	1280X800@60hz	支持 DSI v1.0，DCS v1.0，DPHY v1.0 协议标准
RK322XH/RK332X	HDMI	3840x2160@60hz	支持 HDMI 1.4a 和 2.0 协议标准
	CVBS	720x480i@60hz/720x576i@50hz	支持 NTSC/PAL 标准输出
RK3326/px30	RGB	1280x800@60hz	支持 RGB888/RGB666/sRGB888/MCU
	LVDS	1280x800@60hz	支持 VESA 和 JEIDA LVDS 数据格式
	MIPI	1920x1080@60hz	支持 DSI v1.0，DCS v1.0，DPHY v1.0 协议标准
RK3368/PX5	RGB	1280x800@60hz	支持 RGB888/RGB666/sRGB888/MCU
	LVDS	1280x800@60hz	支持 VESA 和 JEIDA LVDS 数据格式
	MIPI	1920x1080@60hz	支持 DSI v1.0，DCS v1.0， DPHY v1.0 协议标准
	eDP	2560x1600@60hz	支持 DP1.2a 和 eDP1.3 协议标准
	HDMI	4096x2160@60hz	支持 HDMI 1.4a 和 2.0 协议标准
RK3399	MIPI	单通道：1280x800@60hz 双通道：2560x1600@60hz	支持 DSI v1.1，DCS v1.1，DPHY v1.1 协议标准 准
	eDP	2560x1600@60hz	支持 DP1.2a 和 eDP1.3 协议标准
	HDMI	VOP BIG: 4096x2160@60hz VOP LIT: 2560x1600@60hz	支持 HDMI 1.4a 和 2.0a 协议标准

	DP	VOP BIG: 3840x2160@60hz VOP LIT: 2560x1600@60hz	支持 DP 1.3 协议标准
RV1109/RV1126	RGB	1920x1080@60hz	支持 RGB888/RGB666/sRGB888/MCU/BT.1120
	MIPI	1920x1080@60hz	支持 DSI v1.1，DCS v1.1， DPHY v1.2 协议标准
RK356X	RGB	1920x1080@60hz	支持 RGB888/RGB666/sRGB888/MCU/BT.656/BT.1120
	MIPI	单通道：1920x1080@60hz 双通道：2560x1600@60hz	支持 DSI v1.1，DCS v1.1，DPHY v1.1 协议标准准
	eDP	2560x1600@60hz	支持 DP 1.2a 和 eDP 1.3 协议标准
	HDMI	4096x2160@60hz	支持 HDMI 1.4a 和 2.0a 协议标准
RK3588	RGB	1920x1080@60hz	支持 BT.656/BT.1120
	MIPI	3840x2160@60hz	支持 DSI v1.1，DCS v1.1， DPHY v2.0，CPHY V1.1 协议标准
	eDP	3840x2160@60hz	支持 DP1.2a 和 eDP1.3 协议标准
	HDMI	7680x4320@60hz	支持 HDMI 2.1 协议标准
	DP	7680x4320@30hz	支持 DP1.4 协议标准
RV1103/RV1106	RGB	1280x720@60hz	支持 RGB666/sRGB888/MCU/BT.656/BT.1120

4. 硬件相关

4.1 RGB输出/TTL模式硬件连接

4.1.1 VOP 1.0 RGB 接口硬件连接方式

1. 判断是 VOP 1.0 还是 VOP 2.0 的设计，可以从 3.1 章节的 VOP version 中查询；
2. 对于 SOC 支持 24bit RGB 输出的硬件连接方式

interface	RGB parallel						
display mode index	mode0	mode1	mode2	mode3	mode4	mode5	
display mode	RGB parallel 24 bit	RGB parallel 18 bit	RGB parallel 18 bit	RGB parallel 16 bit	RGB parallel 16 bit	serial 3x8/4x8	serial 3x6/4x6
dclk	dclk						
vsync	vsync						
hsync	hsync						
den	den						
data	data[23:0]	data[23:18] data[15:10] data[7:2]	data[17:0]	data[23:19] data[15:10] data[7:3]	data[15:0]	data[7:0]	data[7:2]
D23	R7	R5	–	R4	–	–	–
D22	R6	R4	–	R3	–	–	–
D21	R5	R3	–	R2	–	–	–
D20	R4	R2	–	R1	–	–	–
D19	R3	R1	–	R0	–	–	–
D18	R2	R0	–	–	–	–	–
D17	R1	–	R5	–	–	–	–
D16	R0	–	R4	–	–	–	–
D15	G7	G5	R3	G5	R4	–	–
D14	G6	G4	R2	G4	R3	–	–
D13	G5	G3	R1	G3	R2	–	–
D12	G4	G2	R0	G2	R1	–	–
D11	G3	G1	G5	G1	R0	–	–
D10	G2	G0	G4	G0	G5	–	–
D9	G1	–	G3	–	G4	–	–
D8	G0	–	G2	–	G3	–	–
D7	B7	B5	G1	B4	G2	D7	D5
D6	B6	B4	G0	B3	G1	D6	D4
D5	B5	B3	B5	B2	G0	D5	D3
D4	B4	B2	B4	B1	B4	D4	D2
D3	B3	B1	B3	B0	B3	D3	D1
D2	B2	B0	B2	–	B2	D2	D0
D1	B1	–	B1	–	B1	D1	–
D0	B0	–	B0	–	B0	D0	–

3. 对于 SOC 支持 18bit RGB 输出的硬件连接方式

interface	RGB parallel			
display mode index	mode2	mode4	mode5	
display mode	RGB parallel 18 bit	RGB parallel 16 bit	serial 3x8/4x8	serial 3x6/4x6
dclk	dclk			
vsync	vsync			
hsync	hsync			
den	den			
data	data[17:0]	data[15:0]	data[7:0]	data[7:2]
D17	R5	–	–	–
D16	R4	–	–	–
D15	R3	R4	–	–
D14	R2	R3	–	–
D13	R1	R2	–	–
D12	R0	R1	–	–
D11	G5	R0	–	–
D10	G4	G5	–	–
D9	G3	G4	–	–
D8	G2	G3	–	–
D7	G1	G2	D7	D5
D6	G0	G1	D6	D4
D5	B5	G0	D5	D3
D4	B4	B4	D4	D2
D3	B3	B3	D3	D1
D2	B2	B2	D2	D0
D1	B1	B1	D1	–
D0	B0	B0	D0	–

4. 对于 VOP 1.0 中 MCU 接口 DATA 线连接方式和 RGB parallel 的连接方式一致，需要注意的是，RGB parallel 中的 4 个 时钟信号复用成 MCU 接口的控制信号，以下是具体的对应关系：

dclk	mcu_rs	1表示发送的是数据，0表示发送的是命令
vsync	mcu_csn	片选信号，低有效
hsync	mcu_wrn	写使能信号，上升沿有效
den	mcu_rdn	1：表示发数据到屏，0：表示从屏读数据

4.1.2 VOP 2.0 及之后的版本 RGB 接口硬件连接方式

interface	RGB parallel		
display mode index	mode0	mode1	mode3
display mode	RGB parallel 24 bit	RGB parallel 18 bit	RGB parallel 15 bit
dclk	dclk		
vsync	vsync		
hsync	hsync		
den	den		
data	data[23:0]	data[23:18] data[15:10] data[7:2]	data[23:19] data[15:10] data[7:3]
D23	R7	R5	R4
D22	R6	R4	R3
D21	R5	R3	R2
D20	R4	R2	R1
D19	R3	R1	R0
D18	R2	R0	—
D17	R1	—	—
D16	R0	—	—
D15	G7	G5	G5
D14	G6	G4	G4
D13	G5	G3	G3
D12	G4	G2	G2
D11	G3	G1	G1
D10	G2	G0	G0
D9	G1	—	—
D8	G0	—	—
D7	B7	B5	B4
D6	B6	B4	B3
D5	B5	B3	B2
D4	B4	B2	B1
D3	B3	B1	B0
D2	B2	B0	—
D1	B1	—	—
D0	B0	—	—

4.1.3 不同 **display mode index** 对应的软件配置

display mode index	bus format
mode0	MEDIA_BUS_FMT_RGB888_1X24
mode1	MEDIA_BUS_FMT_RGB666_1X24_CPADHI
mode2	MEDIA_BUS_FMT_RGB666_1X18
mode3	MEDIA_BUS_FMT_RGB565_1X24_CPADLO
mode4	MEDIA_BUS_FMT_RGB565_1X16
mode5	MEDIA_BUS_FMT_RGB888_3X8

4.1.4 **BT.656** 和 **BT.1120** 的硬件连接方式

SOC TX 引脚	Clock	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	软件 bus_format 配置
BT.656 接法	Clock	-	-	-	-	-	-	-	-	D7	D6	D5	D4	D3	D2	D1	D0	MEDIA_BUS_FMT_UYVY8_2X8
BT.1120 接法1	Clock	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	C7	C6	C5	C4	C3	C2	C1	C0	MEDIA_BUS_FMT_YUYV8_1X16
BT.1120 接法2	Clock	C7	C6	C5	C4	C3	C2	C1	C0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	MEDIA_BUS_FMT_UYVY8_1X16

根据接收端的 Y/U/V 处理顺序不同，如果出现显示颜色不对，还可以在 DTS 中调整不同的 bus_format 来适配，BT.656 和 BT.1120 分别有以下四种配置：

BT.656：

```
#define MEDIA_BUS_FMT_UYVY8_2X8         0x2006
#define MEDIA_BUS_FMT_VYUY8_2X8         0x2007
#define MEDIA_BUS_FMT_YUYV8_2X8         0x2008
#define MEDIA_BUS_FMT_YVYU8_2X8         0x2009
```

BT.1120：

```
#define MEDIA_BUS_FMT_UYVY8_1X16        0x200f
#define MEDIA_BUS_FMT_VYUY8_1X16        0x2010
#define MEDIA_BUS_FMT_YUYV8_1X16        0x2011
#define MEDIA_BUS_FMT_YVYU8_1X16        0x2012
```

MEDIA_BUS_FMT 的定义可以参考：

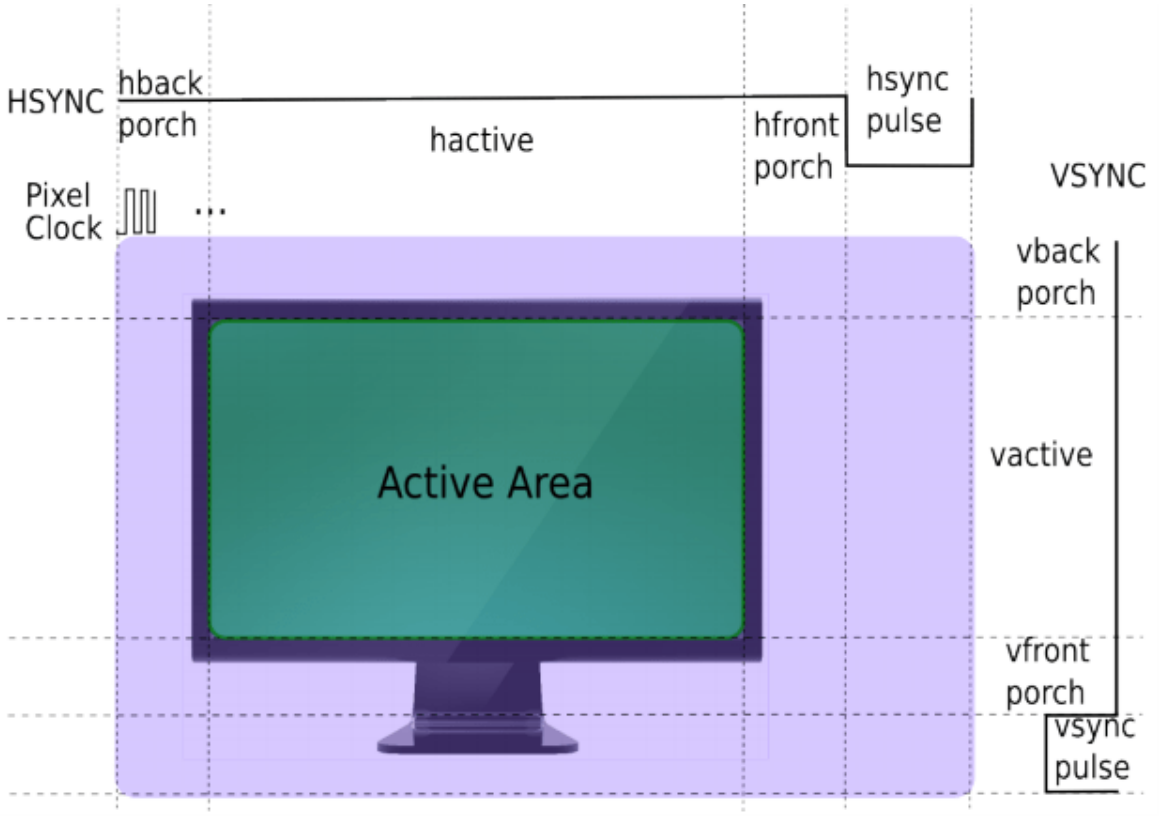
```
./include/uapi/linux/media-bus-format.h
```

4.2 LVDS Data Mapping

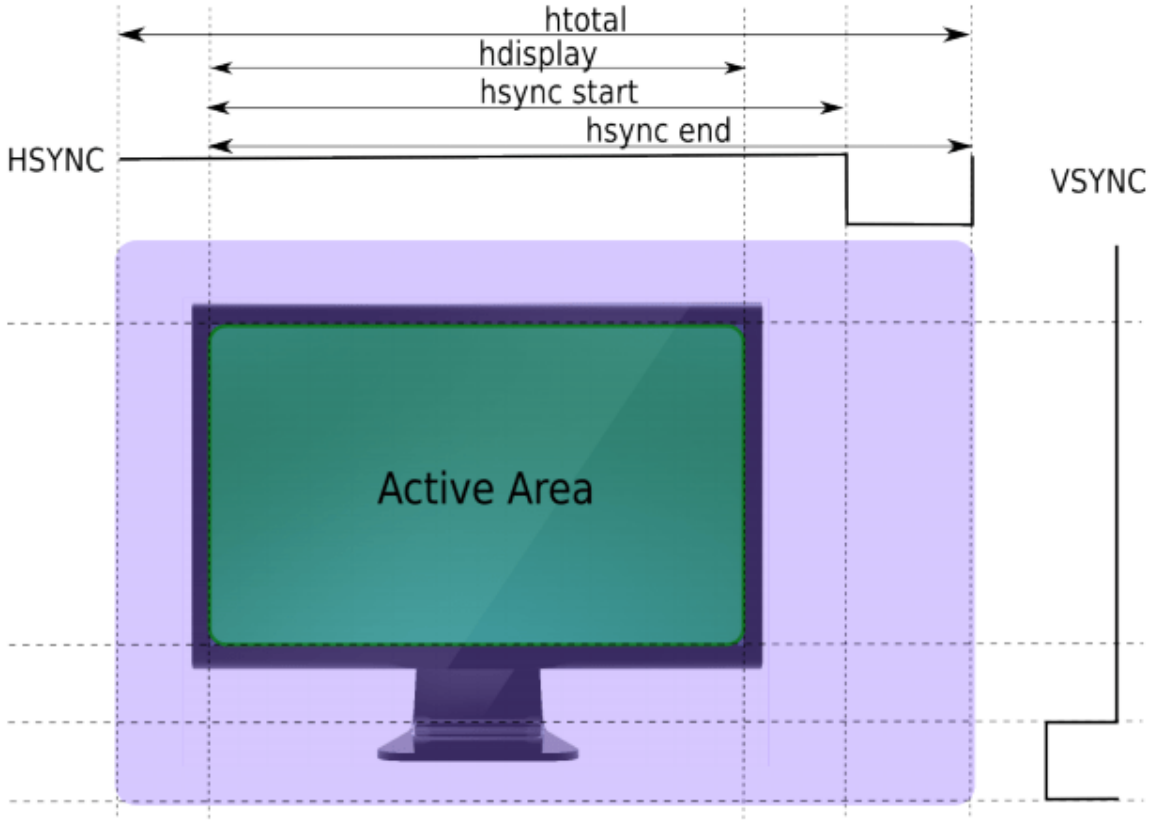
bus-format	Timeslot	Data organization			
		Lane3	Lane2	Lane1	Lane0
MEDIA_BUS_FMT_RGB666_1X7X3_SPWG	0	—	DEN	B1	G0
	1	—	VSYNC	B0	R5
	2	—	HSYNC	G5	R4
	3	—	B5	G4	R3
	4	—	B4	G3	R2
	5	—	B3	G2	R1
	6	—	B2	G1	R0
MEDIA_BUS_FMT_RGB888_1X7X4_SPWG	0	GND	DEN	B1	G0
	1	B7	VSYNC	B0	R5
	2	B6	HSYNC	G5	R4
	3	G7	B5	G4	R3
	4	G6	B4	G3	R2
	5	R7	B3	G2	R1
	6	R6	B2	G1	R0
MEDIA_BUS_FMT_RGB888_1X7X4_JEIDA	0	GND	DEN	B3	G2
	1	B1	VSYNC	B2	R7
	2	B0	HSYNC	G7	R6
	3	G1	B7	G6	R5
	4	G0	B6	G5	R4
	5	R1	B5	G4	R3
	6	R0	B4	G3	R2

5. 扫描时序说明

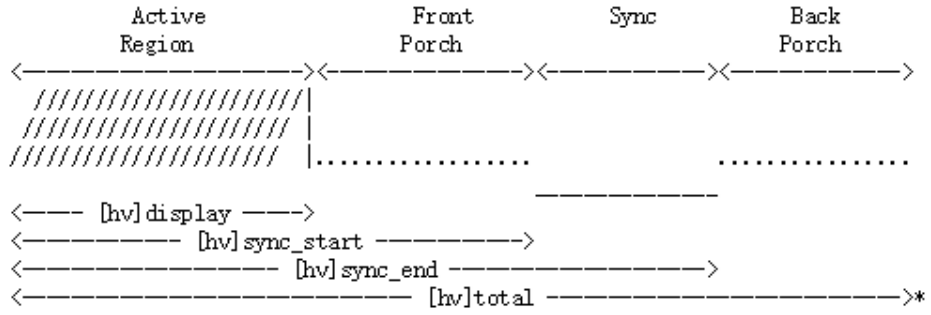
5.1 常见的扫描时序图



5.2 DRM 对扫描时序的定义



5.3 软件配置的对应关系和转换




```c
void drm_display_mode_from_videomode(const struct videomode *vm,
                                     struct drm_display_mode *dmode)
{
        dmode->hdisplay = vm->hactive;
        dmode->hsync_start = dmode->hdisplay + vm->hfront_porch;
        dmode->hsync_end = dmode->hsync_start + vm->hsync_len;
        dmode->htotal = dmode->hsync_end + vm->hback_porch;

        dmode->vdisplay = vm->vactive;
        dmode->vsync_start = dmode->vdisplay + vm->vfront_porch;
        dmode->vsync_end = dmode->vsync_start + vm->vsync_len;
        dmode->vtotal = dmode->vsync_end + vm->vback_porch;

        dmode->clock = vm->pixelclock / 1000;

        if (vm->flags & DISPLAY_FLAGS_HSYNC_HIGH)
                dmode->flags |= DRM_MODE_FLAG_PHSYNC;
        ...

        drm_mode_set_name(dmode);
}

void drm_display_mode_to_videomode(const struct drm_display_mode *dmode,
                                   struct videomode *vm)
{
        vm->hactive = dmode->hdisplay;
        vm->hfront_porch = dmode->hsync_start - dmode->hdisplay;
        vm->hsync_len = dmode->hsync_end - dmode->hsync_start;
        vm->hback_porch = dmode->htotal - dmode->hsync_end;

        vm->vactive = dmode->vdisplay;
        vm->vfront_porch = dmode->vsync_start - dmode->vdisplay;
        vm->vsync_len = dmode->vsync_end - dmode->vsync_start;
        vm->vback_porch = dmode->vtotal - dmode->vsync_end;

        vm->pixelclock = dmode->clock * 1000;

        if (dmode->flags & DRM_MODE_FLAG_PHSYNC)
                vm->flags |= DISPLAY_FLAGS_HSYNC_HIGH;
        ...
}
```

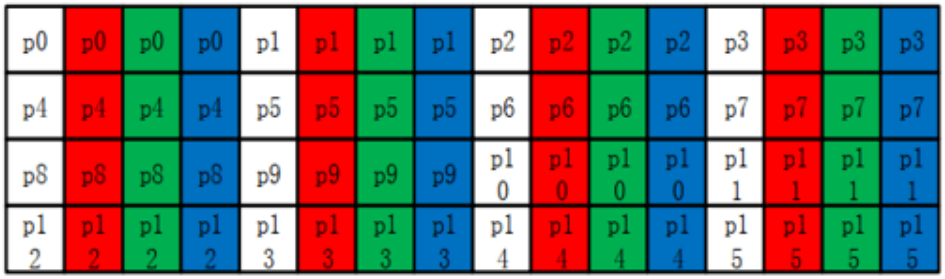
5.4 查看当前配置的时序

```
rk3568_r:/ # cat /d/dri/0/summary
Video Port0: DISABLED
Video Port1: ACTIVE
    Connector: DSI-1
        bus_format[100a]: RGB888_1X24
        overlay_mode[0] output_mode[0] color_space[0]
    Display mode: 1080x1920p60
        clk[132000] real_clk[132000] type[48] flag[a]
        H: 1080 1095 1097 1127
        V: 1920 1935 1937 1952
    Cluster0-win0: ACTIVE
        win_id: 4
        format: AB24 little-endian (0x34324241)[AFBC] SDR[0] color_space[0]
        rotate: xmirror: 0 ymirror: 0 rotate_90: 0 rotate_270: 0
        csc: y2r[0] r2y[0] csc mode[0]
        zpos: 0
        src: pos[0, 0] rect[1080 x 1920]
        dst: pos[0, 0] rect[1080 x 1920]
        buf[0]: addr: 0x0000000000edb000 pitch: 4352 offset: 0
Video Port2: DISABLED
```

6. 带宽的计算方法

6.1 图像的带宽

以 1080P ARGB 格式的图像数据为例：

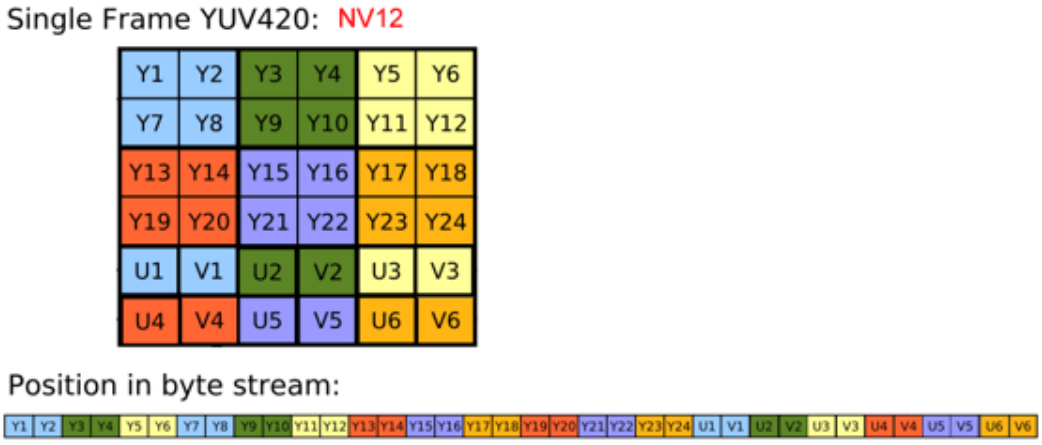


ARGB 格式一个像素占用的内存大小：4 Byte

1080P ARGB 格式的数据占用内存：1920 x 1080 x 4Byte/pixel = 8,100 Kbyte

如果按 60fps 刷新，占用的带宽是： 8,100 x 60fps = 474.6 Mbyte/s

如果数据格式改成 YUV420SP(NV12):



NV12 格式一个像素占用的内存大小：1.5 Byte

1080P NV12 格式的数据占用内存：1920 x 1080 x 1.5Byte/pixel = 3,037.5 Kbyte

如果按 60fps 新，占用的带宽是： 3,037.5 x 60fps = 178 Mbyte/s

6.2 显示接口的带宽

```
disp_timings0: display-timings {
        native-mode = <&dsi0_timing0>;
        dsi0_timing0: timing0 {
                clock-frequency = <132000000>;
                hactive = <1080>;
                vactive = <1920>;
                hfront-porch = <15>;
                hsync-len = <2>;
                hback-porch = <30>;
                vfront-porch = <15>;
                vsync-len = <2>;
                vback-porch = <15>;
                hsync-active = <0>;
                vsync-active = <0>;
                de-active = <0>;
                pixelclk-active = <1>;
        };
};
```

以上面这张图配置的时序为例，当前这个时序下，按 60帧刷新需要的 dclk 是：131994240 hz，dts 实际按取整 132000000 hz 配置：

```
htotal = hfp + hsync + hbp + hactive = 15 + 2 + 30 + 1080 = 1,127
vtotal = vfp + vsync + vbp + vactive = 15 + 2 + 15 + 1920 = 1,952
dclk = htotal x vtotal x fps = 1127 x 1952 x 60fps = 131,994,240
```

MIPI 接口上传输的频率是：

```
132M x 3(RGB) x 8(bpc) / 4(lane) / 0.9 = 880 Mbps
```

其中：

x3(RGB)： 是每一个 pixel 有 RGB 3个分量；

x8(bpc)：是每一个分量的位深是 8bit；

/4(lane): 是这么多数据量在 4 lane 上传输，/4 是计算每 lane 的数据量；

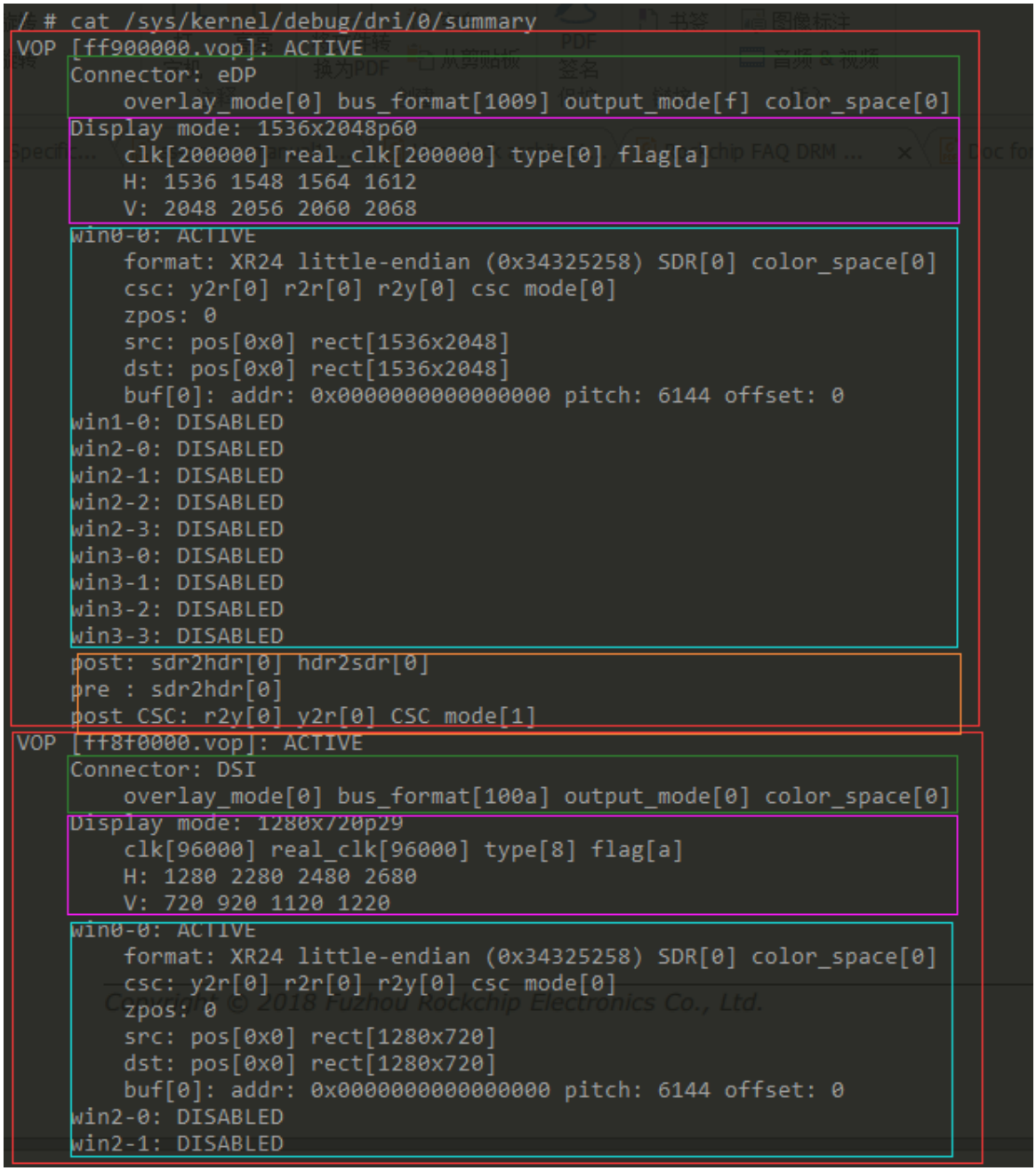
/0.9: 是考虑 mipi 时序的传输效率；

7. 常用的 debug 手段

7.1 dump 当前的显示状态

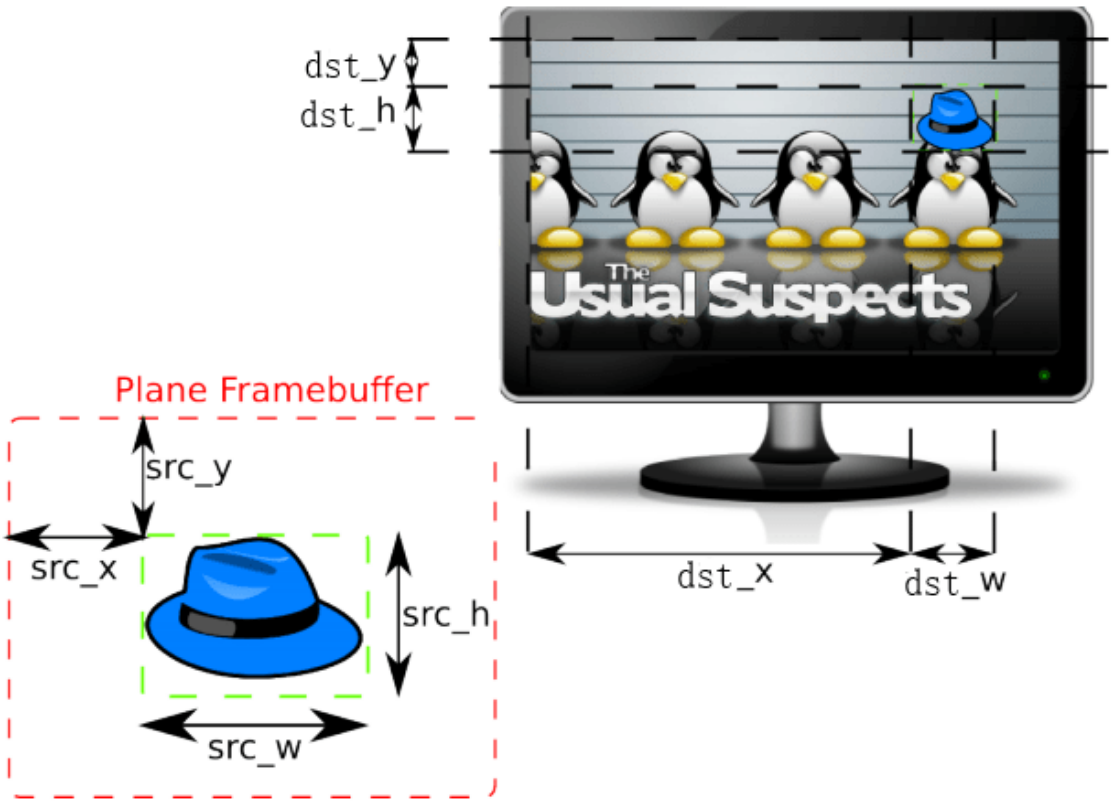
7.1.1 使用命令

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



7.1.2 参数说明

1. 两个红色方框表示两个显示设备使用的 vop 分别是 ff900000.vop 和 ff8f0000.vop；
2. 绿色部分表示 connector 信息，两个显示设备分别为 eDP 屏和 MIPI 屏；
3. 粉色部分为显示模式，可以知道具体的时序、DCLK 以及帧率，上图中两个设备分别为分辨率为 1536x2048p60 的 eDP 屏和分辨率 1280x720p29 的 MIPI 屏；
4. 蓝色部分是 VOP 图层信息，第一个显示设备打开 win0 图层，大小为 1536x2048 格式为 XRGB 第二个显示设备打开 win0 图层，大小 1280x720 格式为XRGB， src 和 dst 表示源数据和显示的大小和位置，如果 src 和 dst 的大小不一致，VOP 会进行缩放处理，如下图所示:



5. 橙色部分为 VOP HDR、CSC 的一些状态信息；
6. VOP2 平台的 summary 信息基本和之前平台的一致，只是把之前 VOP 改成了 Video Port，图层名字从原来的 winx， 变成了 Cluster-winx 或者 Esmart-winx:

```
[root@RK3588:/]#
[root@RK3588:/]#
[root@RK3588:/]#
[root@RK3588:/]# cat /sys/kernel/debug/dri/0/summary
Video Port0: DISABLED
Video Port1: DISABLED
Video Port2: DISABLED
Video Port3: ACTIVE
    Connector: DSI-1
        bus_format[100a]: RGB888_1X24
        overlay_mode[0] output_mode[0] color_space[0], eotf:0
    Display mode: 1080x1920p60
        clk[132000] real_clk[132000] type[48] flag[a]
        H: 1080 1095 1099 1129
        V: 1920 1935 1937 1952
    Esmart3-win0: ACTIVE
        win_id: 11
        format: XR24 little-endian (0x34325258) SDR[0] color_space[0] glb_alpha[0xff]
        rotate: xmirror: 0 ymirror: 0 rotate_90: 0 rotate_270: 0
        csc: y2r[0] r2y[0] csc mode[0]
        zpos: 3
        src: pos[0, 0] rect[1080 x 1920]
        dst: pos[0, 0] rect[1080 x 1920]
        buf[0]: addr: 0x0000000000000000 pitch: 4320 offset: 0
```

7.2 dump当前显示的 buffer

为了确认应用送到显示的 buffer 是否正常，内核提供了节点用于 dump 当前正在显示的 buffer，需要通过在make menuconfig 打开 CONFIG_ROCKCHIP_DRM_DEBUG 启用该功能。

7.2.1 使用说明

```
/sys/kernel/debug/dri/0/ff900000.vop/vop_dump # cat dump
  echo dump    > dump to dump one frame
  echo dumpon  > dump to start vop keep dumping
  echo dumpoff > dump to stop keep dumping
  echo dumpn   > dump n is the number of dump times
  dump path is /data/vop_buf
  if fd err = -3 try rm -r /data/vopbuf echo dump1 > dump can fix it
  if fd err = -28 save needed data try rm -r /data/vopbuf
```

7.3 调整 DRM 打印 Log 等级抓 Log

DRM 根据不同的接口设定以下几个打印等级，可以通过 DRM 的 debug 节点决定开关哪个接口的打印，比如我要查看 ATOMIC 的打印，就你可以输入命令：

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

如果想看到所有的调试信息， 可以输入命令：

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

其中 0x20 Bit5 对应的是 vsync相关的信息，这个打印的内容非常多，如果调试的时候不关注这个信息，一般不要设置这个 bit。

其他更多的打印等级可以查看下面的定义：

```
Enable debug output, where each bit enables a debug category.
Bit 0 (0x01)  will enable CORE messages (drm core code)
Bit 1 (0x02)  will enable DRIVER messages (drm controller code)
Bit 2 (0x04)  will enable KMS messages (modesetting code)
Bit 3 (0x08)  will enable PRIME messages (prime code)
Bit 4 (0x10)  will enable ATOMIC messages (atomic code)
Bit 5 (0x20)  will enable VBL messages (vblank code)
Bit 7 (0x80)  will enable LEASE messages (leasing code)
Bit 8 (0x100) will enable DP messages (displayport code)
```

7.4 查看当前显示时钟

时钟是影响显示模块的关键因素之一，所以我们经常要获取当前系统提供的时钟是多少，可以通过以下命令获取：

1. 获取整个时钟树

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

2. 如果只关注 VOP 的时钟

   ```
   cat /sys/kernel/debug/clk/clk_summary | grep vop
   ```

对于 HDMI/DP/VGA/CVBS 等显示接口，每个分辨率对应的 DCLK 是有严格的标准要求，如果 DCLK 时钟不对可能会出现无法显示或者出现显示兼容性问题，而对于 eDP/LVDS/MIPI/RGB 等显示接口一般有一定的余量。

7.5 强行开关显示设备

以LVDS为例：

关LVDS: echo off > /sys/class/drm/card0-LVDS-1/status

开LVDS: echo on > /sys/class/drm/card0-LVDS-1/status

7.6 查看 DRM buffer 使用情况

通过下面的命令可以知道通过 DRM GEM 申请的 buffer 使用情况：

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

```
/ # cat /sys/kernel/debug/dri/0/mm_dump
0x0000000000000000-0x0000000000c00000: 12582912: used
0x0000000000c00000-0x0000000100000000: 4282384384: free
total: 4294967296, used 12582912 free 4282384384
```

7.7 查看 GPIO 状态

在项目刚开始 bring up 点屏阶段，经常需要通过控制 GPIO 的状态来控制屏或者背光的电源，我们可以通过以下命令确认软件配置的 GPIO 状态是否正确：

```
cat /sys/kernel/debug/gpio
```

```
GPIOs 0-31, platform/pinctrl, gpio0:
 gpio-1   (                    |vcc_sd               ) out lo
 gpio-4   (                    |bt_default_wake_host) in  lo
 gpio-5   (                    |GPIO Key Power       ) in  hi
 gpio-9   (                    |bt_default_reset     ) out lo
 gpio-10  (                    |reset                ) out hi

GPIOs 32-63, platform/pinctrl, gpio1:
 gpio-34  (                    |int-n                ) in  hi
 gpio-45  (                    |enable               ) out hi
 gpio-46  (                    |vsel                 ) out lo
 gpio-49  (                    |vsel                 ) out lo
```

7.8 modetest 的使用

modetest 是集成在 libdrm 中用来测试基于 libdrm 接口的测试程序，可能在以下几个场景中会用到 modetest：

1. 芯片 bring up 阶段，显示驱动已经加载完成， 上层显示框架或者GPU 未 ready 的时候可以用 modetest 测试显示模块
2. 一些场景显示异常， 可以通过 modetest 确认是驱动问题还是应用绘制问题
3. 需要临时手动修改 drm 一些属性的值

可以参考以下使用说明：

```
[root@RK3588:/]# modetest -h
usage: modetest [-acDdefMPpsCvrw]

 Query options:

        -c      list connectors
        -e      list encoders
        -f      list framebuffers
        -p      list CRTCs and planes (pipes)

 Test options:

        -P <plane_id>@<crtc_id>:<w>x<h>[+<x>+<y>][*<scale>][@<format>]  set a plane
        -s <connector_id>[,<connector_id>][@<crtc_id>]:[#<mode index>]<mode>[-<vrefresh>][@<format>]    set a mode
        -C      test hw cursor
        -v      test vsynced page flipping
        -r      set the preferred mode for all connectors
        -w <obj_id>:<prop_name>:<value> set property
        -a      use atomic API
        -F pattern1,pattern2    specify fill patterns

 Generic options:

        -d      drop master after mode set
        -M module       use the given driver
        -D device       use the given device

        Default is to dump all info.
```

7.9 显示进程的暂停、启动

有时候为了调试方便，希望应用停在某一个固定的场景，我们可以使用下面的命令在暂停和恢复显示进程。

- Android Surfaceflinger 进程：

1. 暂停进程

```
kill -STOP `pgrep surfaceflinger`
```

2. 恢复进程

```
kill -CONT `pgrep surfaceflinger`
```

- Linux weston进程：

1. 暂停进程

```
Kill -STOP  `pgrep weston`
```

2. 恢复进程

```
kill -CONT `pgrep surfaceflinger`
```

7.10 获取 EDID 信息

以 HDMI 为例：

```
cat /sys/class/drm/card0-HDMI-A-1/edid > /data/edid.bin
```

7.11 查看 HDMI 状态

```
cat /sys/kernel/debug/dw-hdmi/status
```

```
/ # cat /sys/kernel/debug/dw-hdmi/status
PHY: enabled                    Mode: HDMI
Pixel Clk: 148500000Hz          TMDS Clk: 148500000Hz
Color Format: RGB               Color Depth: 8 bit
Colorimetry: ITU.BT709          EOTF: Off
```

7.11.1 参考例子

1. dump 一帧当前显示的 buffer

   echo dump > /sys/kernel/debug/dri/0/**ff900000.vop**/vop_dump/dump

2. 连续 dump n 帧显示的 buffer

   echo dumpn > /sys/kernel/debug/dri/0/**ff900000.vop**/vop_dump/dump

3. dump 出来的文件保存在 /data/vop_buf/，可以使用 7yuv 软件查看

4. 以上路径加粗部分 [**ff900000.vop**] 在不同平台不一样，具体路径可以和 cat /d/dri/0/summary 节点的 VOP/Video Port 对应，比如对于 RK3588 路径会变成：/sys/kernel/debug/dri/0/**video_port0**/dump

8. FAQ

8.1 如何开关 U-Boot logo 显示

以 MIPI DSI0 为例，可以在 dts 文件里找到 route_dsi0 节点，配置 status 状态：

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi b/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
index 543d78d3f182..b634192d77df 100644
--- a/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
@@ -655,7 +655,7 @@
 };

 &route_dsi0 {
-       status = "okay";
+       status = "disabled";
        connect = <&vp3_out_dsi0>;
 };
```

8.2 如何配置 U-Boot logo 全屏或者居中显示

可以通过 dts route 节点下 logo,mode 配置，默认是 “center” 居中显示，如果是想要全屏显示修改成 “fullscreen”。

```
--- a/arch/arm64/boot/dts/rockchip/rk3588s.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588s.dtsi
@@ -1065,7 +1065,7 @@
                                status = "disabled";
                                logo,uboot = "logo.bmp";
                                logo,kernel = "logo_kernel.bmp";
-                               logo,mode = "center";
+                               logo,mode = "fullscreen";
                                charge_logo,mode = "center";
                                connect = <&vp3_out_dsi0>;
                        };
```

8.3 U-Boot logo 要求

1. U-Boot logo 和内核 logo 的大小一致，偶数像素对齐；
2. 支持 8bit 和 24bit bmp 图片；

8.4 U-Boot logo 切换到内核 logo 出现闪屏/无法显示问题

1. 确认 DRM 驱动是否有正常加载

   DRM 驱动加载过程中可能会出现一些资源没有准备好，导致 DRM 驱动 bind 失败，可能会出现类似以下 log：

   ```
   [    1.792387] dw-mipi-dsi2 fde20000.dsi: [drm:dw_mipi_dsi2_bind] *ERROR* Failed to find panel or bridge: -517
   ```

   这个 log 说明此时 panel 或者 bridge 没准备好，正常框架会在一段时间后会重新开始 bind，但如果最后出现以下 log 说明此时 drm驱动已经加载成功了，这种情况我们就不需要太在意前面一两次 bind 失败的 log：

   ```
   [    2.566831] rockchip-drm display-subsystem: [drm] fb0: rockchipdrmfb frame buffer device
   ```

   如果开机 log 一直不停的刷 bind 失败的 log，那可能要检查的你 dts 配置，产品最经常遇到的是 GPIO 口被其他设备先注册、 panel 的 compatible 未正确配置导致 panel 注册失败、backlight 驱动注册失败等。

2. DDR 变频导致

   (1) 关闭 dts 文件中 DDR 变频节点，保证内核阶段不做 DDR 变频，修改方法如下：

   ```
   &dmc {
       status = “disabled”;
   };
   ```

   (2) 关闭 DDR 变频时对 DCLK 的调整，修改方法如下：

   ```
   &dmc {
       vop-dclk-mode = <1>;
   };
   ```

3. clk tree 变化导致

   部分平台 U-Boot 中的 clk tree 配置和内核的 clk tree 配置是独立的，如果两个驱动的 clk 策略不一致，就会出现在 clk 重新初始化的时候出现闪屏问题，以RK3399 为例，可以按如下方法确认：

   (1) 在rk3399_clk_init()@kernel/drivers/clk/rockchip/clk-rk3399.c 函数入口处加上 while(1)；确认是否会有闪屏问题；

   (2) 在rk3399_clk_init()@kernel/drivers/clk/rockchip/clk-rk3399.c 函数结束处加上 while(1)，确认是否会有闪屏问题；

   (3) 如果步骤 (1) 中无闪屏现象步骤 (2) 中有闪屏现象，那基本可以确认是 clk tree 变化导致闪屏问题，可以直接找对应平台 cru 负责人或者提交 redmine 并说明转给 pll 相关负责人。

4. 时钟被关闭导致

   有一些流程上的问题或者软件上的 bug 可能存在一些必要的时钟在驱动注册的时候没有被使能，导致这些时钟在内核跑完后被框架自动关闭从而导致显示异常，可以尝试按下面的修改默认不关闭时钟做测试：

   在 dts 文件中，找到 chosen 节点，在 bootargs 末尾加上 clk_ignore_unused：如bootargs = "xxxx clk_ignore_unused”；

```
--- a/arch/arm64/boot/dts/rockchip/rk3399-linux.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3399-linux.dtsi
@@ -47,7 +47,7 @@
        compatible = "rockchip,linux", "rockchip,rk3399";

        chosen {
-               bootargs = "earlycon=uart8250,mmio32,0xff1a0000";
+               bootargs = "earlycon=uart8250,mmio32,0xff1a0000 clk_ignore_unused";
        };
```

5. U-Boot logo 图片和 kernel logo 图片大小不一致

   rockchip 平台有要求 U-Boot logo 和 kernel logo 的图片分辨率大小一样，如果出现 U-Boot logo 显示正常，到内核阶段显示异常，可以确认下 kernel 目录下 logo.bmp 和 logo_kernel.bmp 分辨率是否一致；

```
$file logo.bmp  logo_kernel.bmp
logo.bmp:        PC bitmap, Windows 3.x format, 654 x 258 x 8
logo_kernel.bmp: PC bitmap, Windows 3.x format, 654 x 258 x 8
```

6. 内核初始化过程一些电源/GPIO 被重新初始化

   该问题涉及的可能性很多，总之在新项目 porting 过程中要及时确认显示相关的 GPIO 电源是否和其他模块有冲突；

   如果 DTS 确认无误，可以在内核代码搜索串口中的关键字，通过二分法在各个模块加载的位置 while 住，逐步确认导致闪屏问题的点；

7. VOP优先级配置问题

   如果 VOP 优先级没有被配置最高，有可能在内核加载阶段被其他IP抢占总线导致闪屏问题，该问题一般会在 SDK 发布前 Fix。

8. 测试相关电源和信号

   如果以上还未找到闪屏问题，请使用示波器抓取 CLK、DATA 和电源等相关信号从 U-Boot 到内核阶段的波形图并提交 redmine。

8.5 VOP POST_BUF_EMPTY

可能会出现类似以下 log：

```
rockchip-vop ff8f0000.vop: [drm:vop_isr] *ERROR* POST_BUF_EMPTY irq err
rockchip-vop ff8f0000.vop: [drm:vop_isr] *ERROR* POST_BUF_EMPTY irq err
rockchip-vop ff8f0000.vop: [drm:vop_isr] *ERROR* POST_BUF_EMPTY irq err
rockchip-vop ff8f0000.vop: [drm:vop_isr] *ERROR* POST_BUF_EMPTY irq err
rockchip-vop ff8f0000.vop: [drm:vop_isr] *ERROR* POST_BUF_EMPTY irq err
rockchip-vop ff8f0000.vop: [drm:vop_isr] *ERROR* POST_BUF_EMPTY irq err
rockchip-vop ff8f0000.vop: [drm:vop_isr] *ERROR* POST_BUF_EMPTY irq err
rockchip-vop ff8f0000.vop: [drm:vop_isr] *ERROR* POST_BUF_EMPTY irq err
rockchip-vop ff8f0000.vop: [drm:vop_isr] *ERROR* POST_BUF_EMPTY irq err
```

可能原因有：

1. 带宽不够

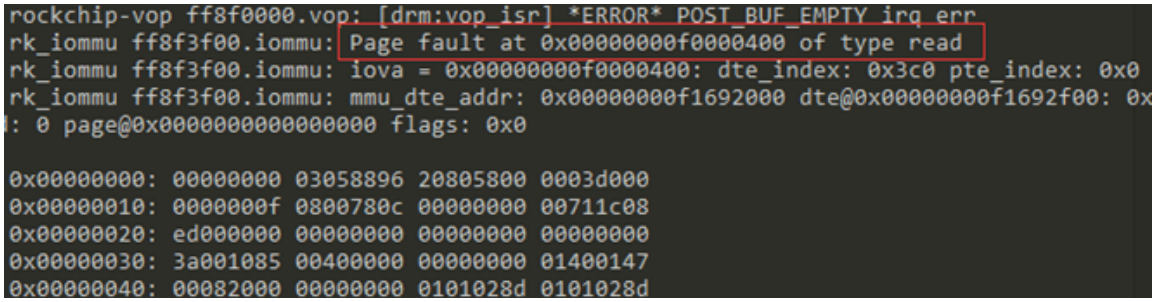
   如果系统带宽不够会导致 VOP 不能及时取到数据，从而报 post buf empty，可以做如下尝试：

   (1) 尝试将 ddr 固定最高频率;

   (2) 将屏的消隐期加长，提高一行的取数时间;

2. iommu 出错

   确认是否如下图所示的 iommu pagefault 错误，如果有，请先尝试更新到最新代码测试，如果最新代码还存在该问题，请提交remine系统，并附上相关 log；



3. Logic 电压太低

   Logic 电压太低会导致 VOP 异常，可以尝试提高 100mv 测试；

4. AFBDC/IFBDC 对齐要求

   对于 PX30/RK3326、RK3368、RK3399、RK356X、RK3588 平台如果屏的分辨率非 16pixel 对齐可以尝试关闭 afbdc/ifbdc 功能，修改方法参考文档《FAQ-DRM-HWC》 1.3.1 章节。

8.6 显示效果调节

VOP 内部的 BCSH 模块支持 亮度、对比度、饱和度、色度的调节，可以通过 modetest 配置 connector 下的对应属性调节，对于android 系统，有实现了基于安卓系统的属性配置默认值是50，N 每次+1。

android 9.0 之前使用命令：

```
setprop persist.sys.brightness.main val
setprop persist.sys.contrast.main val
setprop persist.sys.saturation.main val
setprop persist.sys.hue.main val
setprop sys.display.timeline N+1
```

android 9.0 及之后使用命令：

```
setprop persist.vendor.brightness.main val
setprop persist.vendor.contrast.main val
setprop persist.vendor.saturation.main val
setprop persist.vendor.hue.main val
setprop vendor.display.timeline N+1
```

从 Android 11 开始 SDK 有集成了整套显示效果调节工具和文档，可以参考 Android SDK 中的以下文档说明：

RKDocs/common/display/Rockchip_Introduction_DisplayAdjust_APK_CN.pdf

8.7 屏无法点亮/休眠唤醒显示异常/不显示问题

按以下几个方面做进一步确认：

1. 确认是否有背光；
2. 确认屏的相关电源及复位控制是否正常；
3. 确认上下电时序是否满足屏的spec要求；

8.8 RK3308 如何打开显示功能

RK3308 默认不支持显示功能，如果需要开显示，需要做以下配置：

1. U-Boot 修改

```
--- a/configs/evb-rk3308_defconfig
+++ b/configs/evb-rk3308_defconfig
@@ -4,7 +4,6 @@ CONFIG_SYS_MALLOC_F_LEN=0x2000
CONFIG_ROCKCHIP_RK3308=y
CONFIG_ROCKCHIP_SPL_RESERVE_IRAM=0x0
CONFIG_RKIMG_BOOTLOADER=y
-# CONFIG_USING_KERNEL_DTB is not set
CONFIG_TARGET_EVB_RK3308=y
CONFIG_DEFAULT_DEVICE_TREE="rk3308-evb"
CONFIG_DEBUG_UART=y
@@ -55,6 +54,11 @@ CONFIG_USB_GADGET_DOWNLOAD=y
CONFIG_G_DNL_MANUFACTURER="Rockchip"
CONFIG_G_DNL_VENDOR_NUM=0x2207
CONFIG_G_DNL_PRODUCT_NUM=0x330d
+CONFIG_DM_VIDEO=y
+CONFIG_DISPLAY=y
+CONFIG_DRM_ROCKCHIP=y
+CONFIG_DRM_ROCKCHIP_RGB=y
+CONFIG_LCD=y
CONFIG_USE_TINY_PRINTF=y
CONFIG_SPL_TINY_MEMSET=y
CONFIG_ERRNO_STR=y
```

2. kernel 修改

   RK3308 VOP 不支持 IOMMU，所以分配内存需要从预留的 cma buffer 分配，默认 CMA_SIZE 为 16M，如果出现分配内存失败，可以参考如下方法修改CMA_SIZE：

```
--- a/arch/arm64/configs/rk3308_linux_defconfig
+++ b/arch/arm64/configs/rk3308_linux_defconfig
@@ -44,6 +44,7 @@ CONFIG_HZ_1000=y
 # CONFIG_COMPACTION is not set
 # CONFIG_BOUNCE is not set
 CONFIG_DEFAULT_MMAP_MIN_ADDR=32768
+CONFIG_CMA=y
 # CONFIG_UNMAP_KERNEL_AT_EL0 is not set
 # CONFIG_ARM64_HW_AFDBM is not set
 # CONFIG_ARM64_PAN is not set
@@ -89,6 +90,8 @@ CONFIG_RFKILL=y
 CONFIG_DEVTMPFS=y
 CONFIG_DEVTMPFS_MOUNT=y
 # CONFIG_ALLOW_DEV_COREDUMP is not set
+CONFIG_DMA_CMA=y
+CONFIG_CMA_SIZE_MBYTES=32
 # CONFIG_BLK_DEV is not set
 CONFIG_NETDEVICES=y
 # CONFIG_NET_CORE is not set
```

8.9 关闭 **iommu** 的方法

关闭 vop iommu 后通过 DRM 申请的内存会从 CMA 内存分配，系统默认 CMA 内存大小16M，需要根据场景需求调整到对应的大小，否则会出现分配内存失败。

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3588s.dtsi b/ar
index 915034fc3d0a..26625ff51219 100644
--- a/arch/arm64/boot/dts/rockchip/rk3588s.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3588s.dtsi
@@ -3423,7 +3423,7 @@ vop: vop@fdd90000 {
                                     "dclk_vp1",
                                     "dclk_vp2",
                                     "dclk_vp3";
-                iommus = <&vop_mmu>;
+                //iommus = <&vop_mmu>;
                 power-domains = <&power RK3588_PD_VOP>;
                 rockchip,grf = <&sys_grf>;
                 rockchip,vop-grf = <&vop_grf>;
```

8.10 各种接口屏配置

请参考文档《Rockchip_DRM_Panel_Porting_Guide.pdf》、《RK3588_MIPI_DSI2_Developer_Guide_CN.pdf》。

8.11 RGB/MCU 屏帧率计算问题

已知：

```
htotal = hactive + hback-porch + hfront-porch + hsync-len
vtotal = vactive + vback-porch + vfront-porch + vsync-len
```

N 是每一个像素需要 N 个 cycly 发送完：

bus_format	一个 Pixel 要 N 个 Cycle 发送
MEDIA_BUS_FMT_RGB888_1X24 MEDIA_BUS_FMT_RGB666_1X18 MEDIA_BUS_FMT_RGB565_1X16	1
MEDIA_BUS_FMT_RGB888_3X8	3
MEDIA_BUS_FMT_RGB888_DUMMY_4X8	4

根据以上信息，可以计算帧率：

接口类型	帧率计算
RGB	fps = dclk / (htotal x vtotal x N)
MCU	fps = dclk / (htotal x vtotal x （mcu-pix-total + 1) x N)

8.12 如何编写第三方转换芯片驱动

有些第三方转换芯片需要软件驱动的，这个时候我们需要借助 DRM 框架 bridge 接口向框架注册 connector，比如 RGB2HDMI 的 SII902X 为例，此时 rockchip_rgb.c 充当 encoder 角色，sii902x.c 充当 connector 角色，具体可以参考 SII902X 驱动的实现：

```
drivers/gpu/drm/bridge/sii902x.c
```

8.13 VOP2 绑定每个 VP 所使用的图层

由于 VOP2 支持不同的 VP 共享图层，为了充分合理的使用所有图层资源，我们会根据当前产品 dts 配置的接口类型和数量在 U-Boot 中生成了一种默认的图层策略，如果有些产品没有开 U-Boot logo 显示或者对图层使用有特殊的需求，可以参考下面的写法在 dts 根据需求指定：

rockchip,plane-mask：指定当前 VP 使用的图层；

rockchip,primary-plane：指定 primary 图层，当前 VP 的 primary 图层一定是 rockchip,plane-mask 中的一个，我们一般选用 Smart 或者 Esmart 图层。

```
#include <dt-bindings/display/rockchip_vop.h>

&vp0 {
    rockchip,plane-mask = <(1 << ROCKCHIP_VOP2_CLUSTER1 | 1 << ROCKCHIP_VOP2_SMART1)>;
    rockchip,primary-plane = <ROCKCHIP_VOP2_SMART1>;
};

&vp1 {
    rockchip,plane-mask = <(1 << ROCKCHIP_VOP2_CLUSTER0 | 1 << ROCKCHIP_VOP2_ESMART0 | 1
<< ROCKCHIP_VOP2_SMART0)>;
    rockchip,primary-plane = <ROCKCHIP_VOP2_SMART0>;
};

&vp2 {
    rockchip,plane-mask = <(1 << ROCKCHIP_VOP2_ESMART1)>;
    rockchip,primary-plane = <ROCKCHIP_VOP2_ESMART1>;
};
```

对于一些 Linux 系统，可能还希望指定鼠标层或者希望 plane 和 crtc 是唯一绑定关系，可以在 VP 节点加入以下配置【如果你不了解这个需求，可以忽略这部分配置】：

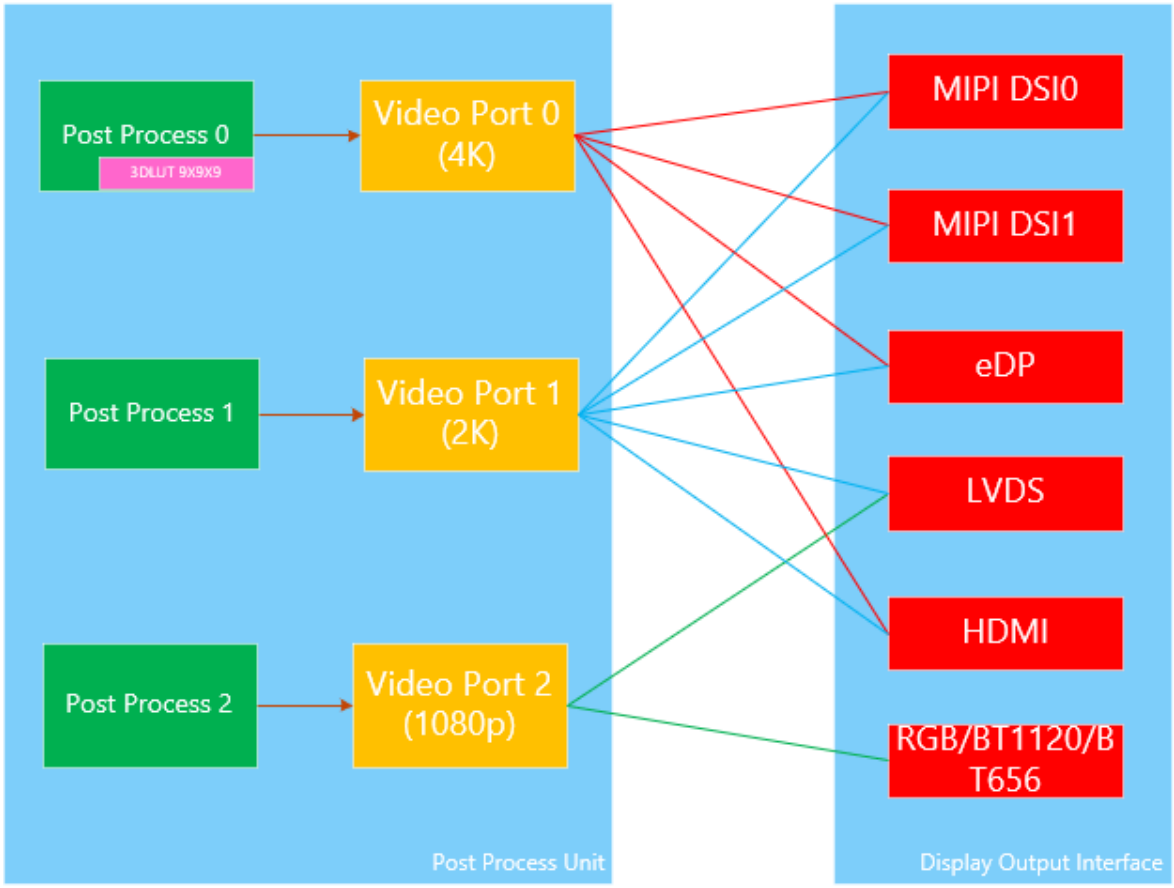
```
&vp0 {
    cursor-win-id = <ROCKCHIP_VOP2_CLUSTER0>;
};

&vop {
        disable-win-move;
}
```

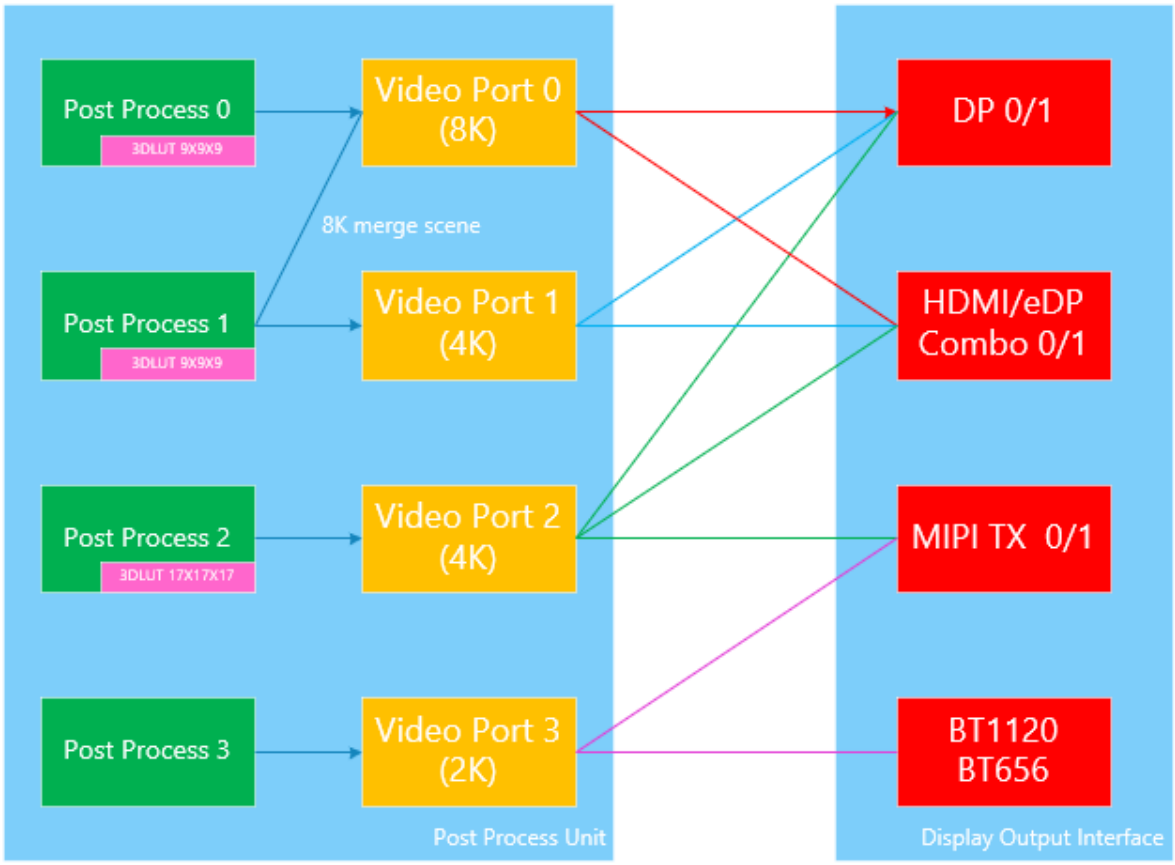
8.14 RK3588 DSC 支持几个 slice，slice width 最大支持多少

RK3588 有 2 个 DSC，HDMI0 和 DSI0 使用 DSC0(DSC_8K)，最大支持 8 slice， 最大的 slice width 是 960；HDMI0 和 DSI1 使用 DSC1(DSC_4K)，最大支持 2 slice，最大的 slice width 是 2048。

8.15 RK3568 VP 和各显示接口的连接关系



8.16 RK3588 VP 和各显示接口的连接关系



8.17 超过 4kP60 对 aclk 的要求

对于类似 RK3588 这种支持的分辨率超过 4KP60 的【如 4KP120, 8KP30, 8KP60 等分辨率】，默认的 500M aclk 无法满足这些分辨率的需求，所以需要在 DTS 中设置 aclk 频率为 800M，否则会出现 VOP 性能不够导致的显示横条纹问题，修改方法如下：

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

9. 参考文档

1. Rockchip_drm_integration_helper-zh.pdf
2. Rockchip_DRM_Panel_Porting_Guide.pdf
3. Rockchip rk fb development guide.pdf
4. Wikipedia for Direct_Rendering_Manager
5. Linux DRM Developer's Guide