Linux Display

开发指南

版本号: 2.6
发布日期: 2023.08.29

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.7.8	AWA0723	创建初始版本
2.0	2020.11.10	AWA01639	更新至 linux5.4 版本
2.1	2020.12.17	AWA1693 AWA1727	1. 完善了接口说明 2. 增加了模块的约束条件说明
2.2	2021.01.20	KPA0527	1. 增加硬件通道相关描述 2. 增加 T 系列平台特定场景描述
2.3	2022.07.11	AW1836	删去 5.10 版本以前相关信息，增加支持 Linux-5.10
2.4	2022.11.23	AWA2075	更新至 linux5.15 版本
2.5	2023.03.08	AWA0723	更新到 A523
2.6	2023.08.29	AWA2130	1. 更新双显启动 LOGO 配置方法 2. 更新硬件结构介绍

目　录

插　图

1 前言

1.1 文档简介

介绍 Sunxi 平台上 Display 驱动模块的一般使用方法及调试接口，为开发与调试提供参考。

1.2 目标读者

- Display 驱动开发人员/维护人员
- Display 模块的应用层使用者

1.3 适用范围

表 1-1: 适用产品

内核版本	驱动文件
Linux-4.9	lichee/{KERNEl_VER}/driver/video/fbdev/sunxi/disp2/*
Linux-5.4	lichee/{KERNEl_VER}/driver/video/fbdev/sunxi/disp2/*
Linux-5.10 及以上	bsp/drivers/video/sunxi/disp2/*

2 模块介绍

2.1 模块功能介绍

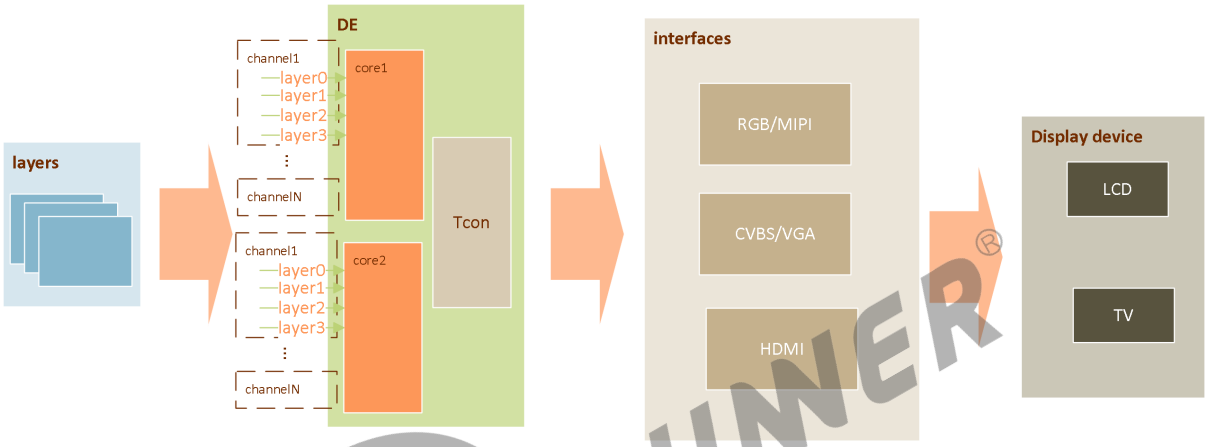

图 2-1: 模块框图

本模块框图如上，由显示引擎（DE）和各类型控制器（tcon）组成。输入图层（layers）在 DE 中进行显示相关处理后，通过一种或多种接口输出到显示设备上显示，以达到将众多应用渲染的图层合成后在显示器呈现给用户观看的作用。DE 有 2 个独立单元（可以简称 de0、de1），可以分别接受用户输入的图层进行合成，输出到不同的显示器，以实现双显。DE 的每个独立的单元有 1-4 个通道（典型地，de0 有 4 个，de1 有 2 个），每个通道可以同时处理接受 4 个格式相同的图层。sunxi 平台有视频通道和 UI 通道之分。视频通道功能强大，可以支持 YUV 格式和 RGB 图层。UI 通道只支持 RGB 图层。简单来说，显示模块的主要功能如下：

- 支持 lcd(hv/lvds/cpu/dsi) 输出
- 支持双显输出
- 支持多图层叠加混合处理
- 支持多种显示效果处理

VI通道支持的处理效果有：
1) 叠加;
2) 缩放;
3) 鲜艳度和对比度增强/黑白电平拉伸/瞬态亮度改善/亮度峰化/饱和度增强/丽色曲率改善;
4) alpha blending/color key;
5) 颜色空间转换。

UI通道支持的处理效果有：
1) 叠加;
2) 缩放;
3) alpha blending/color key;
4) 颜色空间转换。

- 支持智能背光调节
- 支持多种图像数据格式输入 (argb, yuv)
- 支持图像缩放处理
- 支持截屏
- 支持图像转换

2.2 相关术语介绍

2.2.1 硬件术语

表 2-1: 硬件术语

术语	解释
de	display engine，显示引擎，负责将输入的多图层进行叠加、混合、缩放等处理的硬件模块
channel	一个硬件通道，包含若干图层处理单元，可以同时处理若干（典型 4 个）格式相同的图层
layer	一个图层处理单元，可以处理一张输入图像，按支持的图像格式分 video 和 ui 类型
capture	截屏，将 de 的输出保存到本地文件
alpha	透明度，在混合时决定对应图像的透明度
transform	图像变换，如平移、旋转等
overlay	图像叠加，按顺序将图像叠加一起的效果。z 序大的靠近观察者，会把 z 序小的挡住
blending	图像混合，按 alpha 比例将图像合成一起的效果
enhance	图像增强，有目的地处理图像数据以达到改善图像效果的过程或方法

2.2.2 软件术语

表 2-2: 软件术语

术语	解释
fb	帧缓冲（framebuffer）,Linux 为显示设备提供的一个接口，把显存抽象成的一种设备
al	抽象层，驱动中将底层硬件抽象成固定业务逻辑的软件层
lowlevel	底层，直接操作硬件寄存器的软件层

2.3 模块配置介绍

2.3.1 Device Tree 配置说明

- Linux-4.9/Linux-5.4

路径：kernel/{KERNEl_VER}/arch/{ARCH}/boot/dts/{CHIP}.dtsi

(ARCH 为指令集架构，为 arm/arm64; CHIP 为研发代号，如 sun55iw3p1 等)

- Linux-5.10 及以上

路径：bsp/configs/{KERNEl_VER}/{CHIP}.dtsi

sun55iw3p1 配置示例：

dtsi 文件的 &disp 节点，用来配置显示资源（regs、interrupts、clk、reset、iommu 等）信息，一般无需修改。

```
disp: disp@5000000 {
    compatible = "allwinner,sunxi-disp";
    reg = <0x0 0x05000000 0x0 0x400000>, /*de*/
          <0x0 0x05500000 0x0 0x1000>, /* display_if_top */
          <0x0 0x05501000 0x0 0x1000>, /* tcon0 - tcon_lcd0 */
          <0x0 0x05502000 0x0 0x1000>, /* tcon1 - tcon_lcd1 */
          <0x0 0x05503000 0x0 0x1000>, /* tcon2 - tcon_tv0 */
          <0x0 0x05504000 0x0 0x1000>, /* tcon3 - tcon_tv1 */
          <0x0 0x05731000 0x0 0x1000>, /* tcon4 - tcon_lcd2 */
          <0x0 0x05506000 0x0 0x1fff>, /* dsi0 */
          <0x0 0x05508000 0x0 0x1fff>; /* dsi1 */
    interrupts = <GIC_SPI 87 IRQ_TYPE_LEVEL_HIGH>, /* DE */
          <GIC_SPI 90 IRQ_TYPE_LEVEL_HIGH>, /* tcon_lcd0 */
          <GIC_SPI 92 IRQ_TYPE_LEVEL_HIGH>, /* tcon_lcd1 */
          <GIC_SPI 91 IRQ_TYPE_LEVEL_HIGH>, /* tcon_tv0 */
          <GIC_SPI 96 IRQ_TYPE_LEVEL_HIGH>, /* tcon_tv1 */
          <GIC_SPI 97 IRQ_TYPE_LEVEL_HIGH>, /* tcon_lcd2 */
          <GIC_SPI 94 IRQ_TYPE_LEVEL_HIGH>, /* dsi0 */
          <GIC_SPI 95 IRQ_TYPE_LEVEL_HIGH>; /* dsi1 */
    clocks = <&ccu CLK_DE>,
        <&ccu CLK_DE>,
        <&ccu CLK_DE0>,
        <&ccu CLK_DE0>,
        <&ccu CLK_VO0_TCONLCD0>,
        <&ccu CLK_VO0_TCONLCD1>,
        <&ccu CLK_TCONTV>,
        <&ccu CLK_TCONTV1>,
        <&ccu CLK_VO1_TCONLCD0>,
        <&ccu CLK_BUS_VO0_TCONLCD0>,
        <&ccu CLK_BUS_VO0_TCONLCD1>,
        <&ccu CLK_BUS_TCONTV>,
        <&ccu CLK_BUS_TCONTV1>,
        <&ccu CLK_BUS_VO1_TCONLCD0>,
        <&ccu CLK_DPSS_TOP0>,
```

```
        <&ccu CLK_DPSS_TOP0>,
        <&ccu CLK_DPSS_TOP0>,
        <&ccu CLK_DPSS_TOP0>,
        <&ccu CLK_DPSS_TOP1>,
        <&ccu CLK_DSI0>,
        <&ccu CLK_DSI1>,
        <&ccu CLK_BUS_DSI0>,
        <&ccu CLK_BUS_DSI1>,
        <&ccu CLK_COMBPHY0>,
        <&ccu CLK_COMBPHY1>;
    clock-names =  "clk_de0",
          "clk_de1",
          "clk_bus_de0",
          "clk_bus_de1",
          "clk_tcon0",
          "clk_tcon1",
          "clk_tcon2",
          "clk_tcon3",
          "clk_tcon4",
          "clk_bus_tcon0",
          "clk_bus_tcon1",
          "clk_bus_tcon2",
          "clk_bus_tcon3",
          "clk_bus_tcon4",
          "clk_bus_dpss_top0",
          "clk_bus_dpss_top1",
          "clk_bus_dpss_top2",
          "clk_bus_dpss_top3",
          "clk_bus_dpss_top4",
          "clk_mipi_dsi0",
          "clk_mipi_dsi1",
          "clk_bus_mipi_dsi0",
          "clk_bus_mipi_dsi1",
          "clk_mipi_dsi_combphy0",
          "clk_mipi_dsi_combphy1";
    resets = <&ccu RST_BUS_DE0>,
        <&ccu RST_BUS_DE0>,
        <&ccu RST_BUS_VO0_TCONLCD0>,
        <&ccu RST_BUS_VO0_TCONLCD1>,
        <&ccu RST_BUS_TCONTV>,
        <&ccu RST_BUS_TCONTV1>,
        <&ccu RST_BUS_VO1_TCONLCD0>,
        <&ccu RST_BUS_LVDS0>,
        <&ccu RST_BUS_LVDS1>,
        <&ccu RST_BUS_DPSS_TOP0>,
        <&ccu RST_BUS_DPSS_TOP0>,
        <&ccu RST_BUS_DPSS_TOP0>,
        <&ccu RST_BUS_DPSS_TOP0>,
        <&ccu RST_BUS_DPSS_TOP1>,
        <&ccu RST_BUS_DSI0>,
        <&ccu RST_BUS_DSI1>;
    reset-names =  "rst_bus_de0",
          "rst_bus_de1",
          "rst_bus_tcon0",
          "rst_bus_tcon1",
          "rst_bus_tcon2",
          "rst_bus_tcon3",
          "rst_bus_tcon4",
          "rst_bus_lvds0",
          "rst_bus_lvds1",
```

```
        "rst_bus_dpss_top0",
        "rst_bus_dpss_top1",
        "rst_bus_dpss_top2",
        "rst_bus_dpss_top3",
        "rst_bus_dpss_top4",
        "rst_bus_mipi_dsi0",
        "rst_bus_mipi_dsi1";
    assigned-clocks = <&ccu CLK_DE>,
          <&ccu CLK_VO0_TCONLCD0>,
          <&ccu CLK_VO0_TCONLCD1>,
          <&ccu CLK_VO1_TCONLCD0>,
          <&ccu CLK_BUS_TCONTV>,
          <&ccu CLK_DSI0>,
          <&ccu CLK_DSI1>,
          <&ccu CLK_COMBPHY0>,
          <&ccu CLK_COMBPHY1>;
    assigned-clock-parents = <&ccu CLK_PLL_VIDEO3_4X>,
            <&ccu CLK_PLL_VIDEO0_4X>,
            <&ccu CLK_PLL_VIDEO0_4X>,
            <&ccu CLK_PLL_VIDEO0_4X>,
            <&ccu CLK_PLL_VIDEO0_4X>,
            <&ccu CLK_PLL_PERI0_150M>,
            <&ccu CLK_PLL_PERI0_150M>,
            <&ccu CLK_PLL_VIDEO0_4X>,
            <&ccu CLK_PLL_VIDEO0_4X>;
    assigned-clock-rates = <600000000>;
    iommus = <&mmu_aw 5 0>;

    boot_disp = <0>;
    fb_base = <0>;
};
```

tcon 节点定义，为 dts 提供引用节点。

```
lcd0: lcd0@1c0c000 {
    compatible = "allwinner,sunxi-lcd0";
    /* Fake registers to avoid dtc compiling warnings */
    reg = <0x0 0x1c0c000 0x0 0x0>;
    pinctrl-names = "active","sleep";
};

lcd1: lcd1@1c0c000 {
    compatible = "allwinner,sunxi-lcd1";
    /* Fake registers to avoid dtc compiling warnings */
    reg = <0x0 0x1c0c000 0x0 0x0>;
    pinctrl-names = "active","sleep";
};
```

2.3.2 uboot-board.dts 配置说明

uboot-board.dts 用于配置一些板级相关的显示参数 (用于 uboot 阶段)，路径：device/config/chips/{IC}/configs/{BOARD}/uboot-board.dts。节点参数配置说明：

uboot 阶段未使用节点：

```
disp_init_enable
disp_mode
screen0_output_type
screen0_output_mode
screen1_output_type
screen1_output_mode
screen1_output_format
screen1_output_bits
screen1_output_eotf
screen1_output_cs
screen1_output_dvi_hdmi
screen1_output_range
screen1_output_scan
screen1_output_aspect_ratio
disp_para_zone
hdmi_mode_check
```

uboot 阶段相关节点：

dev_num

显示设备数量

一般配置为 1，部分平台支持双 logo，需要双 logo 功能时需配置为 2

设置相应值的对应含义为：

```
1：开启dev0
2：开启dev0 + dev1
```

dev0_screen_id

显示引擎索引

设备 0 连接的到哪个 DE（display engine），一般 dev0 填 0，dev1 填 1

设置相应值的对应含义为：

```
0：连接到DE0（一般为主显）
1：连接到DE1（一般为副显）
```

dev0_output_type

设备 0 的输出类型

设置相应值的对应含义为：

```
0：None
1：LCD
2：TV(cvbs)
4：HDMI
8：VGA
32：EDP
```

screen0_to_lcd_index

dev0_output_type 配置为 1（LCD）时有效，lcd 设备的索引，对应使用 lcd（index）节点的屏参

信息。

设置相应值的对应含义为：

```
0：dev0使用&lcd0
1：dev0使用&lcd1
2：dev0使用&lcd2
```

dev0_output_mode

TV 设备接口（CVBS、HDMI、DP）输出模式配置

设置相应值的对应含义为：

```
5：720p@60hz
10：1080p60hz
配置参数与enum disp_tv_mode一一对应，可参考：longon/brandy/brandy-2.0/u-boot-2018/include/sunxi_display2.h：
    disp_tv_mode
```

dev0_do_hpd

设备热插拔检测

开启后，监测到设备插入才会开启设备，TV 设备一般配置为开启

设置相应值的对应含义为：

```
1：开启
0：关闭
```

dev1_output_type、screen1_to_lcd_index、dev1_output_mode、dev1_screen_id、dev1_do_hpd

与 dev0 类似，配置 dev1 的属性，参考 dev0 配置

def_output_dev

单显配置时生效（dev_num = 1），指定哪个作为唯一显示，一般是 0

设置相应值的对应含义为：

```
0：使用dev0
1：使用dev1
```

fb0_format

framebuffer 存储像素格式

一般用来存储 bootlogo，配置为 bootlogo 对应格式

设置相应值的对应含义为：

```
10：rgb565，bpp=16
8：rgb888，bpp=24
other：argb8888，bpp=32
```

fb0_width

framebuffer 宽度

设置相应值的对应含义为：

```
不能小于bootlogo大小，一般配置为logo大小，单位为像素
```

fb0_height

framebuffer 高度

设置相应值的对应含义为：

```
不能小于bootlogo大小，一般配置为logo大小，单位为像素
```

fb0_rot_used

framebuffer 旋转使用

设置相应值的对应含义为：

```
1：开启
0：关闭
```

fb0_rot_degree

framebuffer 旋转角度

设置相应值的对应含义为：

```
0：0°
1：90°
2：180°
3：270°
```

chn_cfg_mode

通道分配模式

配置显示引擎的通道分配，根据应用场景/产品配置，uboot 与 kernel dts 需要同步这个配置

设置相应值的对应含义为：

```
0：6chn绑定到de0，极致单显性能，会带来更高的功耗
1：4chn绑定到de0，单显，功耗与性能均衡
3：4+3绑定，双显，常用双显配置
4：4+3绑定，双显，当副显分辨率>=2.5K时配置
```

sun55iw3p1 配置示例：

```
&disp {
    disp_init_enable     = <1>;
    disp_mode            = <0>;

    screen0_output_type    = <1>;
```

```
        screen0_output_mode     = <4>;

        screen1_output_type     = <3>;
        screen1_output_mode     = <5>;
        screen1_output_format   = <0>;
        screen1_output_bits     = <0>;
        screen1_output_eotf     = <4>;
        screen1_output_cs       = <257>;
        screen1_output_dvi_hdmi = <2>;
        screen1_output_range    = <2>;
        screen1_output_scan     = <0>;
        screen1_output_aspect_ratio = <8>;

        dev_num             = <2>;
        dev0_output_type        = <1>;
        screen0_to_lcd_index    = <0>;
        dev0_output_mode        = <4>;
        dev0_screen_id          = <0>;
        dev0_do_hpd             = <0>;

        dev1_output_type        = <4>;
        screen1_to_lcd_index    = <2>;
        dev1_output_mode        = <10>;
        dev1_screen_id          = <1>;
        dev1_do_hpd             = <1>;

        def_output_dev          = <0>;
        hdmi_mode_check         = <1>;

        fb0_format              = <0>;
        fb0_width               = <1280>;
        fb0_height              = <800>;
        fb0_rot_used            = <0>;
        fb0_rot_degree          = <0>;

        fb1_format              = <0>;
        fb1_width               = <1920>;
        fb1_height              = <1080>;
        fb1_rot_used            = <0>;
        fb1_rot_degree          = <0>;

        chn_cfg_mode            = <3>;

        disp_para_zone          = <1>;
        /*VCC-LCD*/
        dcdc4-supply = <&reg_dcdc4>;
        /*VCC-DSI*/
        cldo1-supply = <&reg_cldo1>;
        /*VCC-PD*/
        cldo3-supply = <&reg_cldo3>;
};
```

2.3.3 board.dts 配置说明

board.dts 用于配置一些板级相关的显示参数，路径是 device/config/chips/{IC}/configs/{BOARD}/board.dts, 其中 {IC} 是 IC 型号，如 A523，{BOARD} 是板型名称，如 pro1。

kernel 阶段未使用节点：

```
dev0_output_type
dev0_output_mode
dev0_screen_id
dev0_do_hpd
dev1_output_type
dev1_output_mode
dev1_screen_id
dev1_do_hpd
def_output_dev
hdmi_mode_check
disp_para_zone
```

kernel 阶段相关节点：

disp_init_enable

初始化使能，一般配置为 1

设置相应值的对应含义为：

```
0：失能
1：使能
```

disp_mode

显示模式

当平滑启动功能失败时有效，配置 framebuffer 的分配，一般配置为 0

设置相应值的对应含义为：

```
0：fb0 for screen0
1：fb0 for screen1
```

screen0_output_type

显示引擎 0 的输出类型

当平滑启动功能失败时有效，显示引擎 0 备用输出类型

设置相应值的对应含义为：

```
0：None
1：LCD
2：TV(CVBS)
3：HDMI
4：VGA
5：VDPO
6：EDP
```

screen0_to_lcd_index

显示引擎 0 的 LCD 索引

对应使用哪个 lcd 节点

设置相应值的对应含义为：

```
0：screen0使用&lcd0
1：screen0使用&lcd1
2：screen0使用&lcd2
```

screen0_output_mode

TV 设备接口（CVBS、HDMI、DP）输出模式配置

设置相应值的对应含义为：

```
5：720p@60hz
10：1080p60hz
配置参数与enum disp_tv_mode一一对应，可参考：longon/bsp/include/video/sunxi_display2.h：disp_tv_mode
```

screen0_output_format

TV 设备接口（CVBS、HDMI、DP）输出像素格式配置

设置相应值的对应含义为：

```
0：RGB
1：YUV444
2：YUV422
3：YUV420
```

screen0_output_bits

TV 设备接口（CVBS、HDMI、DP）输出 bit 配置

设置相应值的对应含义为：

```
0：8Bits
1：10Bits
```

screen0_output_eotf

TV 设备接口（CVBS、HDMI、DP）输出 eotf 配置

设置相应值的对应含义为：

```
0x001：BT709
0x006：BT601
配置参数与enum disp_eotf 一一对应，可参考：longon/bsp/include/video/sunxi_display2.h：disp_eotf
```

screen0_output_cs

TV 设备接口（CVBS、HDMI、DP）输出 color space 配置

设置相应值的对应含义为：

```
0x101：BT709
0x104：BT601
0x108：BT2020
配置参数与enum disp_color_space 一一对应，可参考：longon/bsp/include/video/sunxi_display2.h：disp_color_space
```

screen0_output_dvi_hdmi

设置相应值的对应含义为：

```
0：undefined
1：DVI
2：HDMI
```

screen0_output_range

TV 设备接口（CVBS、HDMI、DP）输出 color range 配置

设置相应值的对应含义为：

```
0：Default
1：full 0~255
2：limited 16~235
```

screen0_output_scan

设置相应值的对应含义为：

```
0：undefined
1：overscan
2：underscan
```

screen1_output_type、screen1_to_lcd_index、screen1_output_mode、screen1_output_format、screen1_output_bits、screen1_output_eotf、screen1_output_cs、screen1_output_dvi_hdmi、screen1_output_range、screen1_output_scan、screen1_output_aspect_ratio

与 screen0 相同，参考 screen0

fb_format

framebuffer 存储像素格式

设置相应值的对应含义为：

```
一般配置为0（ARGB）
```

fb_num

framebuffer 数量

fb_debug

framebuffer 框架打印信息控制

fb0_map

framebuffer 映射

设置相应值的对应含义为：

```
<0 1 0 16>
从左到右表示
0：使用disp0
1：使用chn1
0：使用layer0
16：zorder顺序
```

fb0_width

framebuffer 宽度，大小为像素大小

设置相应值的对应含义为：

fb0_height

framebuffer 高度，大小为像素大小

fbn_map、fbn_width、fbn_height

与 fb0 类型，参考 fb0

chn_cfg_mode

通道分配模式

配置显示引擎的通道分配，根据应用场景/产品配置，uboot 与 kernel dts 需要同步这个配置

设置相应值的对应含义为：

```
0：6chn绑定到de0，极致单显性能，会带来更高的功耗
1：4chn绑定到de0，单显，功耗与性能均衡
3：4+3绑定，双显，常用双显配置
4：4+3绑定，双显，当副显分辨率>=2.5K时配置
```

sun55iw3p1 配置示例：

```
&disp {
  disp_init_enable      = <1>;
  disp_mode             = <0>;

  screen0_output_type   = <1>;
  screen0_to_lcd_index  = <1>;
  screen0_output_mode   = <4>;

  screen1_output_type   = <3>;
  screen1_to_lcd_index  = <2>;
  screen1_output_mode   = <10>;
  screen1_output_format = <0>;
  screen1_output_bits   = <0>;
  screen1_output_eotf   = <4>;
  screen1_output_cs     = <257>;
  screen1_output_dvi_hdmi = <2>;
  screen1_output_range  = <2>;
  screen1_output_scan   = <0>;
  screen1_output_aspect_ratio = <8>;

  dev0_output_type      = <1>;
```

```
    dev0_output_mode      = <4>;
    dev0_screen_id        = <0>;
    dev0_do_hpd           = <0>;

    dev1_output_type      = <4>;
    dev1_output_mode      = <10>;
    dev1_screen_id        = <1>;
    dev1_do_hpd           = <1>;

    def_output_dev        = <0>;
    hdmi_mode_check       = <1>;

    fb_format             = <0>;
    fb_num                = <1>;
    fb_debug              = <1>;
    /*<disp channel layer zorder>*/
    fb0_map               = <0 1 0 16>;
    fb0_width             = <1200>;
    fb0_height            = <1920>;
    /*<disp channel layer zorder>*/
    fb1_map               = <0 2 0 16>;
    fb1_width             = <300>;
    fb1_height            = <300>;
    /*<disp channel layer zorder>*/
    fb2_map               = <1 0 0 16>;
    fb2_width             = <1280>;
    fb2_height            = <720>;
    /*<disp channel layer zorder>*/
    fb3_map               = <1 1 0 16>;
    fb3_width             = <300>;
    fb3_height            = <300>;

    chn_cfg_mode          = <1>;
    disp_para_zone        = <1>;

    pwms = <&a_pwm 0 5000000 1>;
    /* VCC-LCD*/
    /* dc1sw-supply = <&reg_dc1sw>; */
    /* VCC-DSI */
    /* eldo3-supply = <&reg_eldo3>; */
    /* VCC-PD */
    /* dcdc1-supply = <&reg_dcdc1>; */

    cldo4-supply = <&reg_cldo4>;
    cldo1-supply = <&reg_cldo1>;
    pinctrl-names = "active", "sleep";
    pinctrl-0 = <&a_pwm0_pin_active>;
    pinctrl-1 = <&a_pwm0_pin_sleep>;
};
```

2.3.4 kernel menuconfig 配置说明

在命令行中进入 longan 根目录，执行./build.sh menuconfig 进入配置主界面。

- Linux-4.9/Linux-5.4

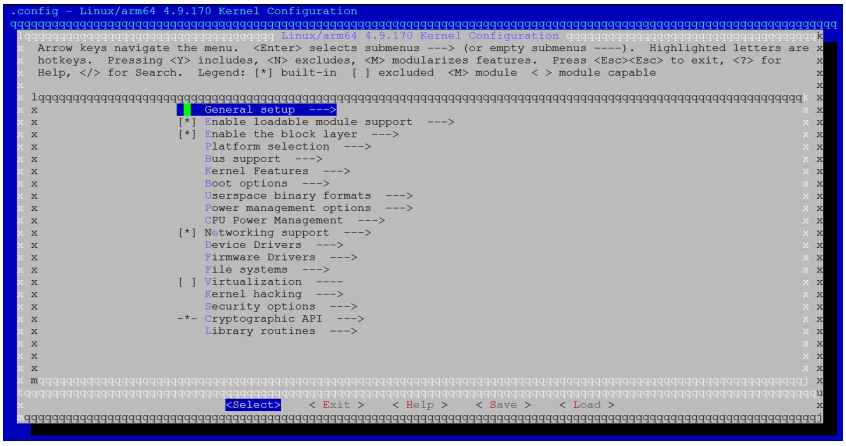

图 2-2: menuconfig 配置

- Linux-5.10 及以上

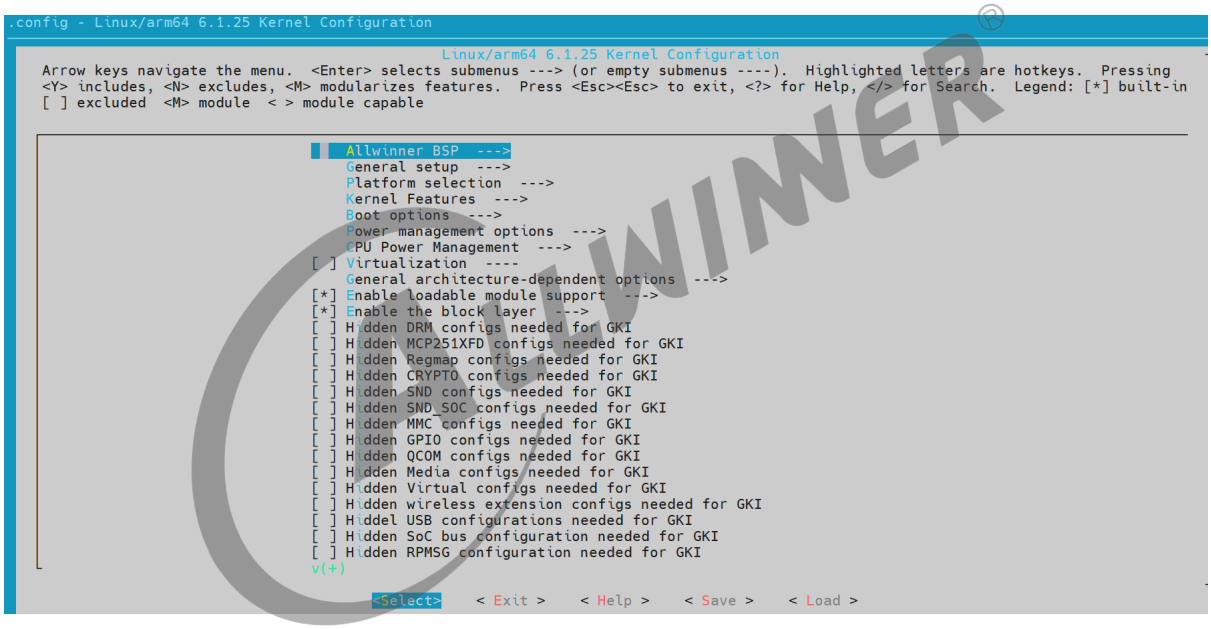

图 2-3: menuconfig 配置 2

进行操作界面后可以按上下切换菜单，按 enter 选中。界面下方也有操作提示。

- Linux-4.9/Linux-5.4

按如下步骤找到 disp2 的驱动菜单：Device Drivers > Graphics support > Frame buffer Devices > Video support for sunxi。

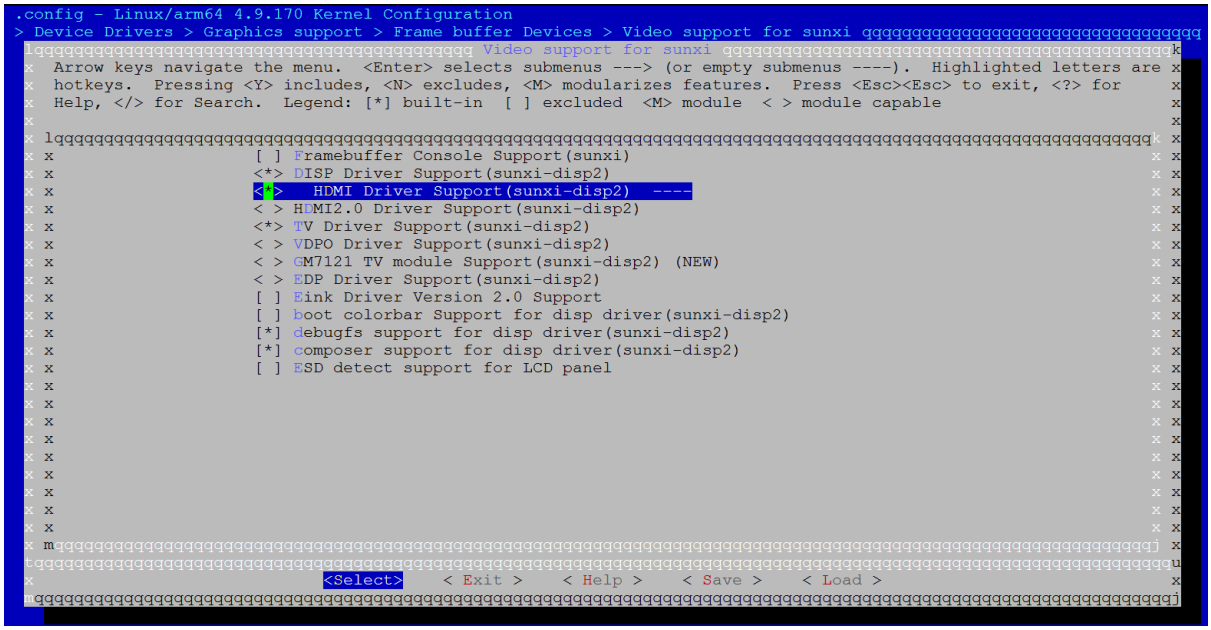

图 2-4: disp2 配置

- Linux-5.10 及以上

按如下步骤找到 disp2 的驱动菜单：Allwinner BSP > Device Drivers > Video Drivers。

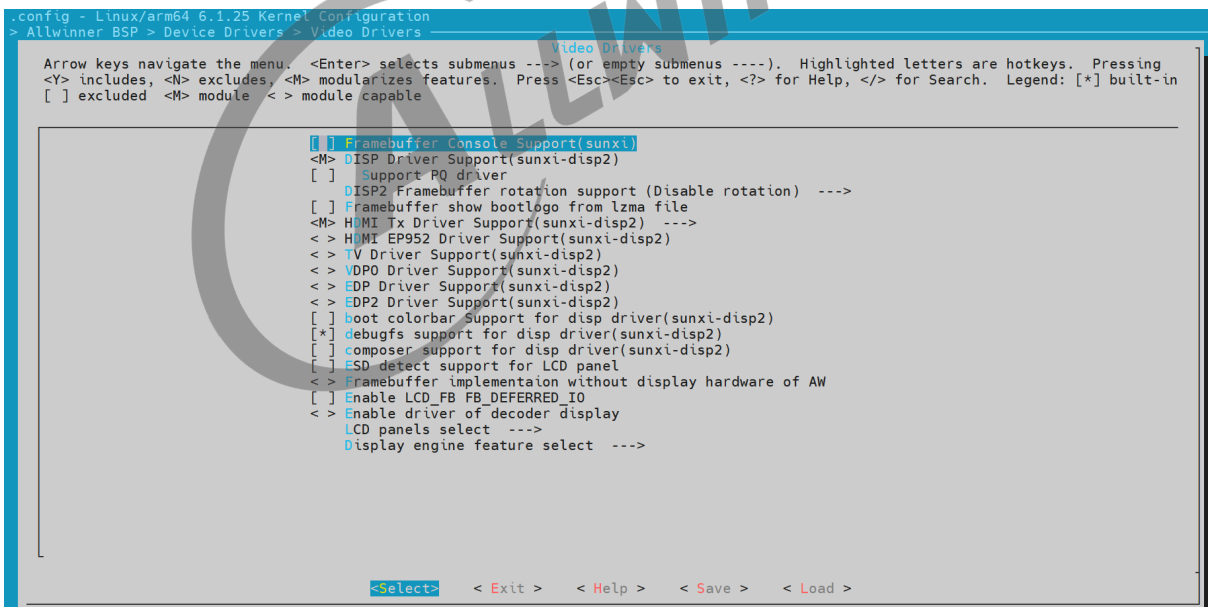

图 2-5: disp2 配置 2

其中：

- DISP Driver Support(sunxi-disp2)

DE 驱动请选上。

- debugfs support for disp driver(sunxi-disp2)

调试节点，建议选上，方便调试。

- composer support for disp driver(sunxi-disp2)

disp2 的 fence 处理。安卓必须选上。纯 linux 不需要也行。

2.4 源码结构介绍

源码结构如下：

```
├──drivers
│   ├──video
│   │   ├──fbdev
│   │   │   ├──sunxi                  --display driver for sunxi
│   │   │   │   ├──disp2/               --disp2 的目录
│   │   │   │   │   ├──disp
│   │   │   │   │   │   ├──dev_disp.c          --display driver 层
│   │   │   │   │   │   ├──dev_fb.c           --framebuffer driver 层
│   │   │   │   │   │   ├──de                --bsp层
│   │   │   │   │   │   │   ├──disp_lcd.c        --disp_manager.c ..
│   │   │   │   │   │   │   ├──disp_al.c         --al层
│   │   │   │   │   │   │   │   └──lowlevel_sun*i/    --lowlevel 层
│   │   │   │   │   │   │   │   ├──de_lcd.c ...
│   │   │   │   │   │   │   └──disp_sys_int.c      --OSAL 层,与操作系统相关层
│   │   │   │   │   │   ├──lcd/ lcd driver
│   │   │   │   │   │   │   │ ├─lcd_src_interface.c  --与display 驱动的接口
│   │   │   │   │   │   │   │ ├─default_panel.c...  --平台已经支持的屏驱动
include
├──video video header dir
│   ├──sunxi_display2.h display header file
```

2.5 驱动框架介绍

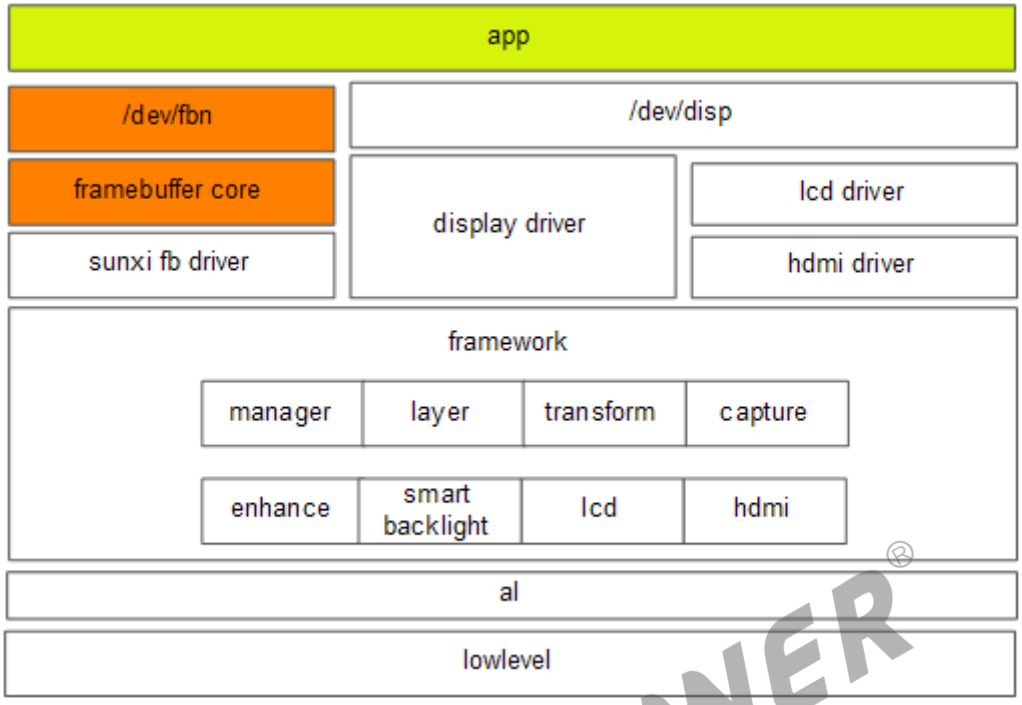

图 2-6: 驱动框图

显示驱动可划分为三个层面：驱动层，框架层及底层。底层与图形硬件相接，主要负责将上层配置的功能参数转换成硬件所需要的参数，并配置到相应寄存器中。显示框架层对底层进行抽象封装成一个个的功能模块。驱动层对外封装功能接口，通过内核向用户空间提供相应的设备结点及统一的接口。在驱动层，分为三 ramebuffer 驱动，disp 驱动，lcd 驱动。Framebuffer 驱动与 framebuffer core 对接，实现 linux 标准的 framebuffer 接口。Disp 驱动是是整个显示驱动中的核心驱动模块，所有的接口都由 disp 驱动来提供，包括 lcd 的接口。

2.6 硬件结构介绍

2.6.1 DE

DE 的功能：接收应用层传输下来的多个图层，将它们合成为一个最终图像，然后给到 TCON，TCON 再通过不同的显示接口，发送到不同的显示设备上。

DE 的内部有多个核心，可以抽象为不同的显示设备，连接不同的 TCON 来传输图像。

DE 有多个接收通道，通道分为：视频通道和 RGB 通道。视频通道支持 YUV 和 RGB 格式，而 RGB 通道则只支持 RGB 格式。不同的通道之间可以做 alpha 混合，而且 zorder 可自行配置。

2.6.2 Interface

显示接口，常见的有：RGB、MIPI、CVBS、VGA、HDMI 等。这些显示接口是与显示设备通信的桥梁，画面信息通过不同显示接口的传输方式，传输给显示设备，显示设备经过解析后，才能呈现到屏幕上。

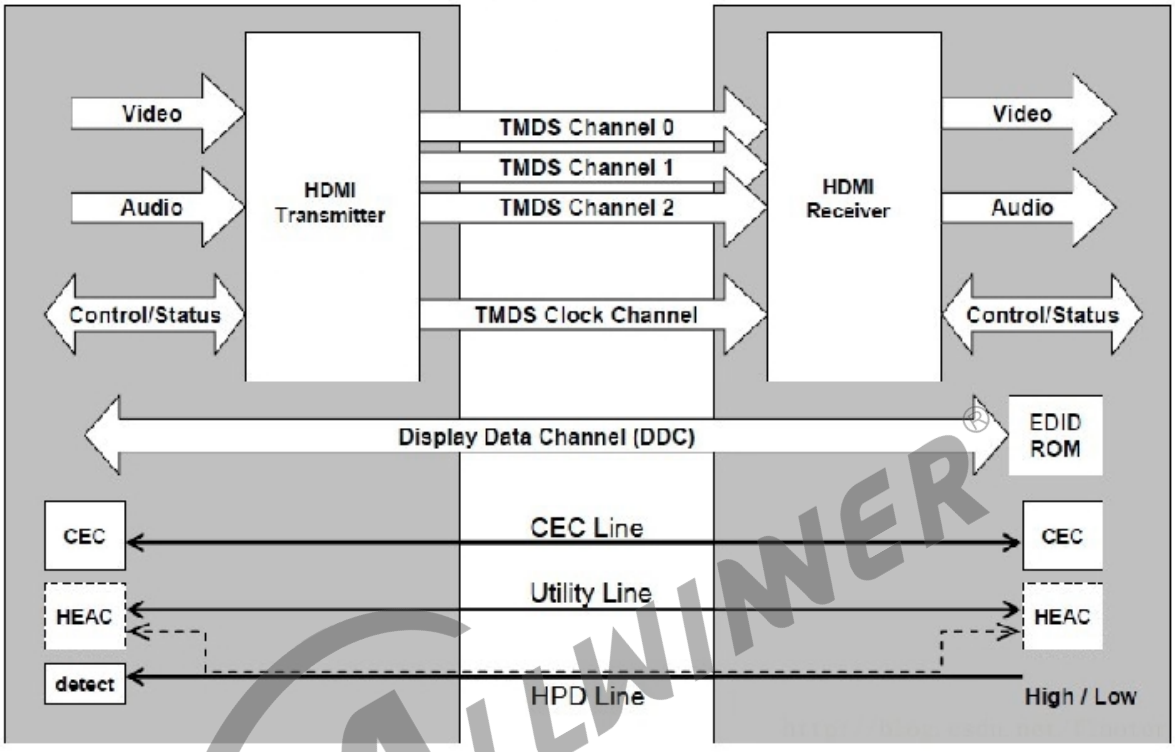

图 2-7: HDMI 接口

2.6.3 显示设备

常见的显示设备，例如：LCD、TV 等。这些显示设备就是最终把画面输出到用户眼睛的终端，也是显示效果最终呈现的设备。芯片输出的显示效果会受到显示设备的制约，显示能力强的显示设备才能更好地呈现出芯片输出的画面。

图 2-8: LCD 屏

图 2-9: TV

3 模块接口说明

3.1 模块接口概述

模块使用主要通过 ioctl 实现，对应的驱动节点是/dev/disp2。具体定义请仔细阅读头文件上面的注释：bsp/include/video/sunxi_display2.h。对于显示模块来说，把图层参数设置到驱动，让显示器显示为最重要。sunxi 平台的 DE 接受用户设置的图层以 disp,channel,layer_id 三个索引唯一确定（disp:0/1, channel: 0/1/2/3，layer_id:0/1/2/3），其中 disp 表示显示器索引，channel 表示通道索引，layer_id 表示通道内的图层索引。下面着重地把图层的参数从头文件中拿出来介绍:

```
struct disp_fb_info2 {
    int                 fd;
    struct disp_rectsz      size[3];
    unsigned int            align[3];
    enum disp_pixel_format  format;
    enum disp_color_space   color_space;
    int                 trd_right_fd;
    bool                pre_multiply;
    struct disp_rect64      crop;
    enum disp_buffer_flags  flags;
    enum disp_scan_flags    scan;
    enum disp_eotf          eotf;
    int                 depth;
    unsigned int            fbd_en;
    int                 metadata_fd;
    unsigned int            metadata_size;
    unsigned int            metadata_flag;
};
```

- fd

显存的文件句柄

- size 与 crop

Size 表示 buffer 的完整尺寸，crop 则表示 buffer 中需要显示裁减区。如下图所示，完整的图像以 size 标识，而矩形框住的部分为裁减区，以 crop 标识，在屏幕上只能看到 crop 标识的部分，其余部分是隐藏的，不能在屏幕上显示出来的。

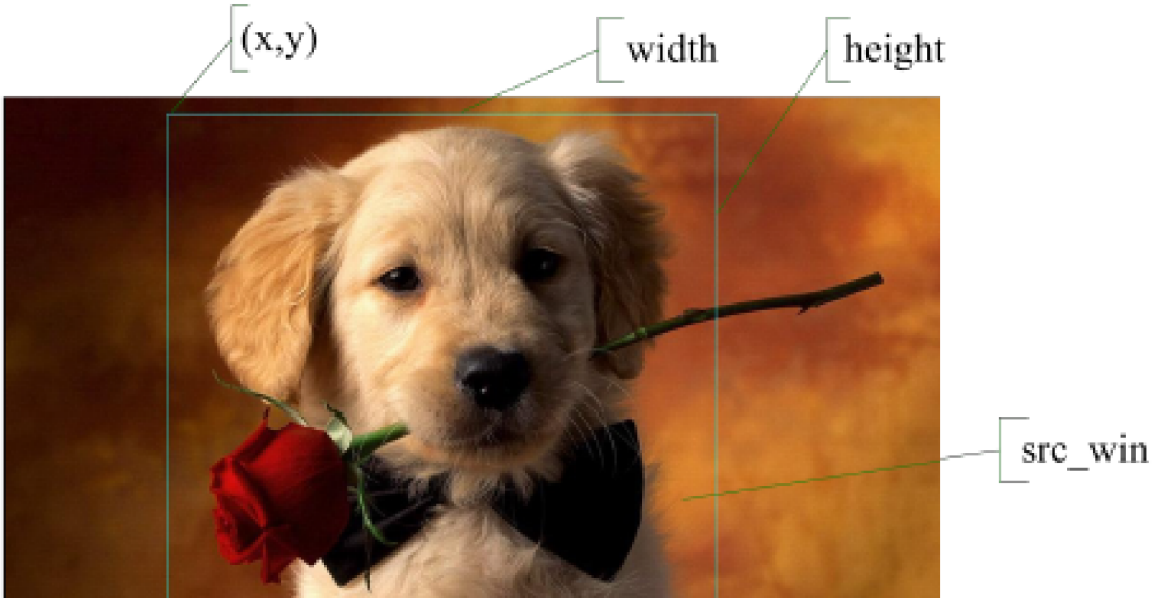

图 3-1: size 和 crop 示意图

- crop 和 screen_win

crop 上面已经介绍过，Screen_win 为 crop 部分 buffer 在屏幕上显示的位置。如果不需要进行缩放的话，crop 和 screen_win 的 width 和 height 是相等的。如果需要缩放，crop 和 screen_win 的 width 和 height 可以不相等。

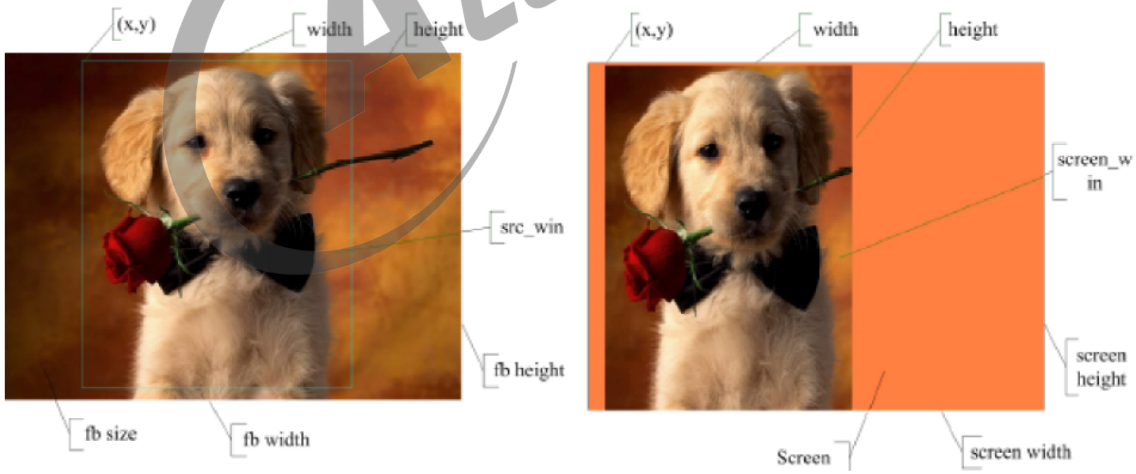

图 3-2: screenwin

- align

显存的对齐字节数

- format

输入图层的格式。Ui 通道支持的格式：

```
DISP_FORMAT_ARGB_8888
DISP_FORMAT_ABGR_8888
DISP_FORMAT_RGBA_8888
DISP_FORMAT_BGRA_8888
DISP_FORMAT_XRGB_8888
DISP_FORMAT_XBGR_8888
DISP_FORMAT_RGBX_8888
DISP_FORMAT_BGRX_8888
DISP_FORMAT_RGB_888
DISP_FORMAT_BGR_888
DISP_FORMAT_RGB_565
DISP_FORMAT_BGR_565
DISP_FORMAT_ARGB_4444
DISP_FORMAT_ABGR_4444
DISP_FORMAT_RGBA_4444
DISP_FORMAT_BGRA_4444
DISP_FORMAT_ARGB_1555
DISP_FORMAT_ABGR_1555
DISP_FORMAT_RGBA_5551
DISP_FORMAT_BGRA_5551
DISP_FORMAT_A2R10G10B10
DISP_FORMAT_A2B10G10R10
DISP_FORMAT_R10G10B10A2
DISP_FORMAT_B10G10R10A2
```

Video 通道支持的格式：

```
DISP_FORMAT_ARGB_8888
DISP_FORMAT_ABGR_8888
DISP_FORMAT_RGBA_8888
DISP_FORMAT_BGRA_8888
DISP_FORMAT_XRGB_8888
DISP_FORMAT_XBGR_8888
DISP_FORMAT_RGBX_8888
DISP_FORMAT_BGRX_8888
DISP_FORMAT_RGB_888
DISP_FORMAT_BGR_888
DISP_FORMAT_RGB_565
DISP_FORMAT_BGR_565
DISP_FORMAT_ARGB_4444
DISP_FORMAT_ABGR_4444
DISP_FORMAT_RGBA_4444
DISP_FORMAT_BGRA_4444
DISP_FORMAT_ARGB_1555
DISP_FORMAT_ABGR_1555
DISP_FORMAT_RGBA_5551
DISP_FORMAT_BGRA_5551
DISP_FORMAT_YUV444_I_AYUV
DISP_FORMAT_YUV444_I_VUYA
DISP_FORMAT_YUV422_I_YVYU
DISP_FORMAT_YUV422_I_YUYV
DISP_FORMAT_YUV422_I_UYVY
DISP_FORMAT_YUV422_I_VYUY
DISP_FORMAT_YUV444_P
DISP_FORMAT_YUV422_P
DISP_FORMAT_YUV420_P
DISP_FORMAT_YUV411_P
DISP_FORMAT_YUV422_SP_UVUV
```

```
DISP_FORMAT_YUV422_SP_VUVU
DISP_FORMAT_YUV420_SP_UVUV
DISP_FORMAT_YUV420_SP_VUVU
DISP_FORMAT_YUV411_SP_UVUV
DISP_FORMAT_YUV411_SP_VUVU
DISP_FORMAT_YUV444_I_AYUV_10BIT
DISP_FORMAT_YUV444_I_VUYA_10BIT
```

3.2 约束条件

3.2.1 叠加

1. RGB 格式没有透明度 A 分量的 layer 可以放在一个 channel，优先从 channel0 开始分配；
2. 不交叠的 layer 可以放在一个 channel；
3. 每个 layer 分配到同一 channel 必须是同一缩放比例的；
4. 分配 layer 优先从 channel0 开始，video 优先分配 video channel；
5. 缩放都需要满足 scaler 缩放性能要求；

```
缩放比例从1/16x 到32x；
UI channel的缩放要求输入/输出大小为 4x2 到 8192x8192；
VI channel的缩放要求输入/输出的大小为 8x4 到 8192x8192。
```

6. 3D layer 需占 2 个 layer；
7. 格式说明：

```
1. UI channel格式请对照Ui通道支持格式，同一channel可以有不同的RGB格式；
2. VI channel格式请对照Video通道支持格式，同一channel，图层输入格式要么是RGB，要么是YUV，当输入为RGB时，4个
   layer可以是不同的RGB分量格式；当输入为YUV时，4个layer只能是同一YUV分量格式。
```

3.2.2 视频通道叠加

1. 若为 video format，则 4 个 layer 必须同一 format；
2. 若为 ui format，则 4 个 layer 都必须为 ui format；
3. 后处理 smart color，只在 video channel0，有且是 display0 存在；

3.2.3 SNR

1. 只支持 yuv420/yuv422 格式；
2. 不支持 interleave 格式；

3.2.4 回写

1. 可选择从 display0 或 display1 输出；
2. 输出支持缩小，不支持放大；

3.2.5 对齐方式

1. yuv 数据要符合 2/4 pixel 对齐；
2. atw 宽高需要能被块 NxN 的 N 值整除，如 40x40，需要被 40 整除；

3.3 模块使用接口说明

sunxi 平台下显示驱动给用户提供了众多功能接口，可对图层、LCD 等显示资源进行操作。

3.4 Global Interface

3.4.1 DISP_SHADOW_PROTECT

- 作用：DISP_SHADOW_PROTECT（1）与 DISP_SHADOW_PROTECT（0）配对使用，在两个接口调用之间的接口调用将不马上执行, 而等到调用 DISP_SHADOW_PROTECT（0）后才一并执行。
- 参数：
  handle：显示驱动句柄；
  cmd：DISP_SHADOW_PROTECT；
  arg：arg[0] 为显示通道 0/1；arg[1] 为 protect 参数，1 表示 protect, 0: 表示 not protect。
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//启动cache，dispfd 为显示驱动句柄
unsigned long arg[3];
arg[0] = 0;//屏0
arg[1] = 1;//protect
ioctl(dispfd, DISP_SHADOW_PROTECT, (void*)arg);
//do somthing other
arg[1] = 0;
ioctl(dispfd, DISP_SHADOW_PROTECT, (void*)arg);
```

3.4.2 DISP_SET_BKCOLOR

- 作用：该函数用于设置显示背景色
- 参数：handle：显示驱动句柄
  cmd：DISP_SET_BKCOLOR
  arg：arg[0] 为显示通道 0/1；arg[1] 为 backcolor 信息，指向 disp_color 数据结构指针。
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//设置显示背景色，dispfd 为显示驱动句柄，sel 为屏0/1
disp_color bk;
unsigned long arg[3];
bk.red = 0xff;
bk.green = 0x00;
bk.blue = 0x00;
arg[0] = 0;
arg[1] = (unsigned long)&bk;
ioctl(dispfd, DISP_SET_BKCOLOR, (void*)arg);
```

3.4.3 DISP_GET_SCN_WIDTH

- 作用：该函数用于获取当前屏幕水平分辨率。
- 参数：handle：显示驱动句柄
  cmd：DISP_GET_SCN_WIDTH
  arg：arg[0] 为显示通道 0/1
- 返回：正值：成功并返回当前屏幕水平分辨率
  其他: 失败号
- 示例

```
//获取屏幕水平分辨率
unsigned int screen_width;
unsigned long arg[3];
arg[0] = 0;
screen_width = ioctl(dispfd, DISP_GET_SCN_WIDTH, (void*)arg);
```

3.4.4 DISP_GET_SCN_HEIGHT

- 作用：该函数用于获取当前屏幕垂直分辨率
- 参数：handle：显示驱动句柄
  cmd：DISP_GET_SCN_HEIGHT
  arg：arg[0] 为显示通道 0/1

- 返回：正值：成功并返回当前屏幕垂直分辨率
  其他: 失败号
- 示例

```
//获取屏幕垂直分辨率
unsigned int screen_height;
unsigned long arg[3];
arg[0] = 0;
screen_height = ioctl(dispfd, DISP_GET_SCN_HEIGHT, (void*)arg);
```

3.4.5 DISP_GET_OUTPUT_TYPE

- 作用：该函数用于获取当前显示输出类型 (LCD,TV,HDMI,VGA,NONE)
- 参数：handle：显示驱动句柄
  cmd：DISP_GET_OUTPUT_TYPE
  arg：arg[0] 为显示通道 0/1
- 返回：正值: 成功，返回当前显示输出类型
  其他: 失败号
- 示例

```
//获取当前显示输出类型
disp_output_type output_type;
unsigned long arg[3];
arg[0] = 0;
output_type = (disp_output_type)ioctl(dispfd, DISP_GET_OUTPUT_TYPE, (void*)arg);
```

3.4.6 DISP_GET_OUTPUT

- 作用：该函数用于获取当前显示输出类型及模式 (LCD,TV,HDMI,VGA,NONE)
- 参数：handle：显示驱动句柄
  cmd：DISP_GET_OUTPUT
  arg：arg[0] 为显示通道 0/1；arg[1] 为指向 disp_output 结构体的指针，用于保存返回值
- 返回：0: 成功
  其他: 失败号
- 示例

```
//获取当前显示输出类型
unsigned long arg[3];
struct disp_output output;
enum disp_output_type type;
enum disp_tv_mode mode;
arg[0] = 0;
arg[1] = (unsigned long)&output;
```

```
ioctl(dispfd, DISP_GET_OUTPUT, (void*)arg);
type = (enum disp_output_type)output.type;
mode = (enum disp_tv_mode)output.mode;
```

3.4.7 DISP_VSYNC_EVENT_EN

- 作用：该函数开启/关闭 vsync 消息发送功能
- 参数：handle：显示驱动句柄
  cmd：DISP_VSYNC_EVENT_EN
  arg：arg[0] 为显示通道 0/1；arg[1] 为 enable 参数，0：disable，1：enable
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//开启/关闭vsync 消息发送功能，dispfd 为显示驱动句柄，sel 为屏0/1
unsigned long arg[3];
arg[0] = 0;
arg[1] = 1;
ioctl(dispfd, DISP_VSYNC_EVENT_EN, (void*)arg);
```

3.4.8 DISP_DEVICE_SWITCH

- 作用：该函数用于切换输出类型
- 参数：handle：显示驱动句柄
  cmd：DISP_DEVICE_SWITCH
  arg：arg[0] 为显示通道 0/1；arg[1] 为输出类型；arg[2] 为输出模式，在输出类型不为 LCD 时有效
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//切换
unsigned long arg[3];
arg[0] = 0;
arg[1] = （unsigned long)DISP_OUTPUT_TYPE_HDMI;
arg[2] = (unsigned long)DISP_TV_MOD_1080P_60HZ;
ioctl(dispfd, DISP_DEVICE_SWITCH, (void*)arg);
说明：如果传递的type 是DISP_OUTPUT_TYPE_NONE，将会关闭当前显示通道的输出。
```

3.4.9 DISP_DEVICE_SET_CONFIG

- 作用：该函数用于切换输出类型并设置输出设备的属性参数

- 参数：handle：显示驱动句柄
  cmd：DISP_DEVICE_SET_CONFIG
  arg：arg[0] 为显示通道 0/1；arg[1] 为指向 disp_device_config 的指针
- 返回：DIS_SUCCESS: 成功
  其他: 失败
- 示例

```
//切换输出类型并设置输出设备的属性参数
unsigned long arg[3];
struct disp_device_config config;
config.type = DISP_OUTPUT_TYPE_HDMI;
config.mode = DISP_TV_MOD_1080P_60HZ;
config.format = DISP_CSC_TYPE_YUV420;
config.bits = DISP_DATA_10BITS;
config.eotf = DISP_EOTF_SMPTE2084;
config.cs = DISP_BT2020NC;
arg[0] = 0;
arg[1] = （unsigned long)&config;
ioctl(dispfd, DISP_DEVICE_SET_CONFIG, (void*)arg);
说明：如果传递的type 是DISP_OUTPUT_TYPE_NONE，将会关闭当前显示通道的输出。
```

3.4.10 DISP_DEVICE_GET_CONFIG

- 作用：该函数用于获取当前输出类型及相关的属性参数
- 参数：handle：显示驱动句柄
  cmd：DISP_DEVICE_GET_CONFIG
  arg：arg[0] 为显示通道 0/1；arg[1] 为指向 disp_device_config 的指针
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//获取当前输出类型及相关的属性参数
unsigned long arg[3];
struct disp_device_config config;
arg[0] = 0;
arg[1] = （unsigned long)&config;
ioctl(dispfd, DISP_DEVICE_GET_CONFIG, (void*)arg);
说明：如果返回的type 是DISP_OUTPUT_TYPE_NONE，表示当前输出显示通道为关闭状态。
```

3.5 Layer Interface

3.5.1 DISP_LAYER_SET_CONFIG

- 作用：该函数用于设置多个图层信息

- 参数：handle：显示驱动句柄
  cmd：DISP_SET_LAYER_CONFIG
  arg：arg[0] 为显示通道 0/1；arg[1] 为图层配置参数指针；arg[2] 为需要配置的图层数目
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
struct
{
   disp_layer_info info,
   bool enable;
   unsigned int channel,
   unsigned int layer_id,
}disp_layer_config;
//设置图层参数，dispfd 为显示驱动句柄
unsigned long arg[3];
struct disp_layer_config config;
unsigned int width = 1280;
unsigned int height = 800;
unsigned int ret = 0;
memset(&config, 0, sizeof(struct disp_layer_config));
config.channnel = 0;//blending channel
config.layer_id = 0;//layer index in the blending channel
config.info.enable = 1;
config.info.mode = LAYER_MODE_BUFFER;
config.info.fb.addr[0] = (unsigned long long)mem_in; //FB 地址
config.info.fb.size[0].width = width;
config.info.fb.align[0] = 4;//bytes
config.info.fb.format = DISP_FORMAT_ARGB_8888; //DISP_FORMAT_YUV420_P
config.info.fb.crop.x = 0;
config.info.fb.crop.y = 0;
config.info.fb.crop.width = ((unsigned long)width) << 32;//定点小数。高32bit 为整数，低32bit 为小数
config.info.fb.crop.height= ((uunsigned long)height)<<32;//定点小数。高32bit 为整数，低32bit 为小数
config.info.fb.flags = DISP_BF_NORMAL;
config.info.fb.scan = DISP_SCAN_PROGRESSIVE;
config.info.alpha_mode = 2; //global pixel alpha
config.info.alpha_value = 0xff;//global alpha value
config.info.screen_win.x = 0;
config.info.screen_win.y = 0;
config.info.screen_win.width = width;
config.info.screen_win.height= height;
config.info.id = 0;
arg[0] = 0;//screen 0
arg[1] = (unsigned long)&config;
arg[2 = 1; //one layer
ret = ioctl(dispfd, DISP_LAYER_SET_CONFIG, (void*)arg);
```

3.5.2 DISP_LAYER_GET_CONFIG

- 作用：该函数用于获取图层参数
- 参数：handle：显示驱动句柄
  cmd：DISP_LAYER_GET_CONFIG
  arg：arg[0] 为显示通道 0/1；arg[1] 为图层配置参数指针；arg[2] 为需要获取配置的图层数目

- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//设置图层参数，dispfd 为显示驱动句柄
unsigned long arg[3];
struct disp_layer_config config;
memset(&config, 0, sizeof(struct disp_layer_config));
arg[0] = 0; //disp
arg[1] = (unsigned long)&config;
arg[2] = 1; //layer number
ret = ioctl(dispfd, DISP_GET_LAYER_CONFIG, (void*)arg);
```

3.5.3 DISP_LAYER_SET_CONFIG2

- 作用：该函数用于设置多个图层信息，注意该接口只接受 disp_layer_config2 的信息
- 参数：handle：显示驱动句柄
  cmd：DISP_SET_LAYER_CONFIG2
  arg：arg[0] 为显示通道 0/1；arg[1] 为图层配置参数 (disp_layer_config2) 的指针；arg[2] 为需要配置的图层数目
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
struct
{
   disp_layer_info info,
   bool enable;
   unsigned int channel,
   unsigned int layer_id,
}disp_layer_config2;
//设置图层参数，dispfd 为显示驱动句柄
unsigned long arg[3];
struct disp_layer_config2 config;
unsigned int width = 1280;
unsigned int height = 800;
unsigned int ret = 0;
memset(&config, 0, sizeof(struct disp_layer_config2));
config.channnel = 0;//blending channel
config.layer_id = 0;//layer index in the blending channel
config.info.enable = 1;
config.info.mode = LAYER_MODE_BUFFER;
config.info.fb.addr[0] = (unsigned long long)mem_in; //FB 地址
config.info.fb.size[0].width = width;
config.info.fb.align[0] = 4;//bytes
config.info.fb.format = DISP_FORMAT_ARGB_8888; //DISP_FORMAT_YUV420_P
config.info.fb.crop.x = 0;
config.info.fb.crop.y = 0;
config.info.fb.crop.width = ((unsigned long)width) << 32;//定点小数。高32bit 为整数，低32bit 为小数
config.info.fb.crop.height= ((uunsigned long)height)<<32;//定点小数。高32bit 为整数，低32bit 为小数
config.info.fb.flags = DISP_BF_NORMAL;
```

```
config.info.fb.scan = DISP_SCAN_PROGRESSIVE;
config.info.fb.eotf = DISP_EOTF_SMPTE2084; //HDR
config.info.fb.metadata_buf = (unsigned long long)mem_in2;
config.info.alpha_mode = 2; //global pixel alpha
config.info.alpha_value = 0xff;//global alpha value
config.info.screen_win.x = 0;
config.info.screen_win.y = 0;
config.info.screen_win.width = width;
config.info.screen_win.height= height;
config.info.id = 0;
arg[0] = 0;//screen 0
arg[1] = (unsigned long)&config;
arg[2 = 1; //one layer
ret = ioctl(dispfd, DISP_LAYER_SET_CONFIG2, (void*)arg);
```

3.5.4 DISP_LAYER_GET_CONFIG2

- 作用：该函数用于获取图层参数
- 参数：handle：显示驱动句柄
  cmd：DISP_LAYER_GET_CONFIG2
  arg：arg[0] 为显示通道 0/1；arg[1] 为图层配置参数 (disp_layer_config2) 的指针；arg[2] 为需要获取配置的图层数目
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//设置图层参数，dispfd 为显示驱动句柄
unsigned long arg[3];
struct disp_layer_config2 config;
memset(&config, 0, sizeof(struct disp_layer_config2));
arg[0] = 0; //disp
arg[1] = (unsigned long)&config;
arg[2] = 1; //layer number
ret = ioctl(dispfd, DISP_GET_LAYER_CONFIG2, (void*)arg);
```

3.6 capture interface

3.6.1 DISP_CAPTURE_START

- 作用：该函数启动截屏功能
- 参数：handle：显示驱动句柄
  cmd：DISP_CAPTURE_START
  arg：arg[0] 为显示通道 0/1
- 返回：DIS_SUCCESS: 成功
  其他: 失败号

- 示例

```
//启动截屏功能，dispfd 为显示驱动句柄
arg[0] = 0;//显示通道0
ioctl(dispfd, DISP_CAPTURE_START, (void*)arg);
```

3.6.2 DISP_CAPTURE_COMMIT

- 作用：该函数提交截屏信息，提交后才走在启动截屏功能
- 参数：handle：显示驱动句柄
  cmd：DISP_CAPTURE_COMMIT
  arg：arg[0] 为显示通道 0/1；arg[1] 为 struct disp_capture_info 参数，用以设置截取的窗口信息和保存图片的信息。
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//提交截屏功能，dispfd 为显示驱动句柄
unsigned long arg[3];
struct disp_capture_info info;
arg[0] = 0;
screen_width = ioctl(dispfd, DISP_GET_SCN_WIDTH, (void*)arg);
screen_height = ioctl(dispfd, DISP_GET_SCN_HEIGHT, (void*)arg);
info.window.x = 0;
info.window.y = 0;
info.window.width = screen_width;
info.window.y = screen_height;
info.out_frame.format = DISP_FORMAT_ARGB_8888;
info.out_frame.size[0].width = screen_width;
info.out_frame.size[0].height = screen_height;
info.out_frame.crop.x = 0;
info.out_frame.crop.y = 0;
info.out_frame.crop.width = screen_width;
info.out_frame.crop.height = screen_height;
info.out_frame.addr[0] = fb_address; //buffer address
arg[0] = 0;//显示通道0
arg[1] = (unsigned long)&info;
ioctl(dispfd, DISP_CAPTURE_COMMIT, (void*)arg);
```

3.6.3 DISP_CAPTURE_STOP

- 作用：该函数停止截屏功能
- 参数：handle：显示驱动句柄
  cmd：DISP_CAPTURE_STOP
  arg：arg[0] 为显示通道 0/1

- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//停止截屏功能，dispfd 为显示驱动句柄
unsigned long arg[3];
arg[0] = 0;//显示通道0
ioctl(dispfd, DISP_CAPTURE_STOP, (void*)arg);
```

3.6.4 DISP_CAPTURE_QUERY

- 作用：该函数查询刚结束的图像帧是否截屏成功
- 参数：handle：显示驱动句柄
  cmd：DISP_CAPTURE_QUERY
  arg：arg[0] 为显示通道 0/1
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//查询截屏是否成功，dispfd 为显示驱动句柄
unsigned long arg[3];
arg[0] = 0;//显示通道0
ioctl(dispfd, DISP_CAPTURE_QUERY, (void*)arg);
```

3.7 LCD Interface

3.7.1 DISP_LCD_SET_BRIGHTNESS

- 作用：该函数用于设置 LCD 的亮度
- 参数：handle：显示驱动句柄
  cmd：DISP_LCD_SET_BRIGHTNESS
  arg：arg[0] 为 DE0/1 对应的显示通道；arg[1] 为背光亮度值（0~255）
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//设置LCD 的背光亮度，dispfd 为显示驱动句柄
unsigned long arg[3];
unsigned int bl = 197;
arg[0] = 0;//DE0输出的屏
arg[1] = bl;
```

```
ioctl(dispfd, DISP_LCD_SET_BRIGHTNESS, (void*)arg);
```

3.7.2 DISP_LCD_GET_BRIGHTNESS

- 作用：该函数用于获取 LCD 的亮度
- 参数：handle：显示驱动句柄
  cmd：DISP_LCD_GET_BRIGHTNESS
  arg：arg[0] 为 DE0/1 对应的显示通道
- 返回：DIS_SUCCESS: 成功
  其他: 失败
- 示例

```
//获取LCD 的背光亮度，dispfd 为显示驱动句柄
unsigned long arg[3];
unsigned int bl;
arg[0] = 0;//显示通道0
bl = ioctl(dispfd, DISP_LCD_GET_BRIGHTNESS, (void*)arg);
```

3.7.3 DISP_LCD_SET_GAMMA_TABLE

- 作用：该函数用于获取显示背景色。
- 参数：handle：显示驱动句柄
  cmd：DISP_LCD_SET_GAMMA_TABLE
  arg：arg[0] 为显示通道 0/1；arg[1] 为 gamma table 的首地址；arg[2] 为 gamma table 的 size，字节为单位，建议为 1024，不能超过这个值
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//设置lcd 的gamma table，dispfd 为显示驱动句柄
unsigned long arg[3];
unsigned int gamma_tbl[1024];
unsigned int size = 1024;
/* init gamma table */
/* gamma_tbl[nn] = xx; */
arg[0] = 0;//显示通道0
arg[1] = gamma_tbl;
arg[2] = size;
if (ioctl(dispfd, DISP_LCD_SET_GAMMA_TABLE, (void*)arg))
   printf(“set gamma table fail!\n”);
else
   printf(“set gamma table success\n”);
```

3.7.4 DISP_LCD_GAMMA_CORRECTION_ENABLE

- 作用：该函数用于使能 lcd 的 gamma 校正功能
- 参数：handle：显示驱动句柄
  cmd：DISP_LCD_GAMMA_CORRECTION_ENABLE
  arg：arg[0] 为显示通道 0/1
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//使能lcd 的gamma 校正功能，dispfd 为显示驱动句柄
unsigned long arg[3];
arg[0] = 0;//显示通道0
if (ioctl(dispfd, DISP_LCD_GAMMA_CORRECTION_ENABLE, (void*)arg))
  printf(“enable gamma correction fail!\n”);
else
  printf(“enable gamma correction success\n”);
```

3.7.5 DISP_LCD_GAMMA_CORRECTION_DISABLE

- 作用：该函数用于关闭 lcd 的 gamma 校正功能。
- 参数：handle：显示驱动句柄
  cmd：DISP_LCD_GAMMA_CORRECTION_DISABLE
  arg：arg[0] 为显示通道 0/1
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//关闭lcd 的gamma 校正功能，dispfd 为显示驱动句柄
unsigned long arg[3];
arg[0] = 0;//显示通道0
if (ioctl(dispfd, DISP_LCD_GAMMA_CORRECTION_DISABLE, (void*)arg))
  printf(“disable gamma correction fail!\n”);
else
  printf(“disable gamma correction success\n”);
```

3.8 smart backlight

3.8.1 DISP_SMBL_ENABLE

- 作用：该函数用于使能智能背光功能

- 参数：handle：显示驱动句柄
  cmd：DISP_SMBL_ENABLE
  arg：arg[0] 为显示通道 0/1
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//开启智能背光功能，dispfd 为显示驱动句柄
unsigned long arg[3];
arg[0] = 0;//显示通道0
ioctl(dispfd, DISP_SMBL_ENABLE, (void*)arg);
```

3.8.2 DISP_SMBL_DISABLE

- 作用：该函数用于关闭智能背光功能
- 参数：handle：显示驱动句柄
  cmd：DISP_SMBL_DISABLE
  arg：arg[0] 为显示通道 0/1
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//关闭智能背光功能，dispfd 为显示驱动句柄
unsigned long arg[3];
arg[0] = 0;//显示通道0
ioctl(dispfd, DISP_SMBL_DISABLE, (void*)arg);
```

3.8.3 DISP_SMBL_SET_WINDOW

- 作用：该函数用于设置智能背光开启效果的窗口，智能背光在设置的窗口中有效
- 参数：handle：显示驱动句柄
  cmd：DISP_SMBL_SET_WINDOW
  arg：arg[0] 为显示通道 0/1，arg[1] 为指向 struct disp_rect 的指针
- 返回：DIS_SUCCESS: 成功
  其他: 失败号
- 示例

```
//设置智能背光窗口，dispfd 为显示驱动句柄
unsigned long arg[3];
unsigned int screen_width, screen_height;
struct disp_rect window;
screen_width = ioctl(dispfd, DISP_GET_SCN_WIDTH, (void*)arg);
```

```
screen_height = ioctl(dispfd, DISP_GET_SCN_HEIGHT, (void*)arg);
window.x = 0;
window.y = 0;
window.width = screen_width / 2;
widnow.height = screen_height;
arg[0] = 0;//显示通道0
arg[1] = (unsigned long)&window;
ioctl(dispfd, DISP_SMBL_SET_WINDOW, (void*)arg);
```

3.9 sysfs 接口描述

以下两个函数在下面接口的 demo 中会使用到。

```
const int MAX_LENGTH = 128;
const int MAX_DATA = 128;
static ssize_t read_data(const char *sysfs_path, char *data)
{
    ssize_t err = 0;
    FILE *fp = NULL;
    fp = fopen(sysfs_path, "r");
    if (fp) {
        err = fread(data, sizeof(char), MAX_DATA ,fp);
        fclose(fp);
    }
    return err;
}

static ssize_t write_data(const char *sysfs_path, const char *data, size_t len)
{
    ssize_t err = 0;
    int fd = -1;
    fd = open(sysfs_path, O_WRONLY);
    if (fp) {
        errno = 0;
        err = write(fd, data, len);
        if (err < 0) {
            err = -errno;
        }
        close(fd);
    } else {
        ALOGE("%s: Failed to open file: %s error: %s", __FUNCTION__, sysfs_path, strerror(errno));
        err = -errno;
    }
    return err;
}
```

3.10 enhance

3.10.1 enhance_mode

在 video channel& 显示 YUV 数据的时候才有效，一般是 DE0 才支持。

```
- SYSFS NODE

/sys/class/disp/disp/attr/disp
/sys/class/disp/disp/attr/enhance_mode

- ARGUMENTS

disp display channel，比如0: disp0, 1：disp1
enhance_mode： enhance mode，0：standard, 1: enhance, 2: soft, 3: enahnce + demo

- RETURNS

none

- DESCRIPTION

该接口用于设置色彩增强的模式

- DEMO

//设置disp0 的色彩增强的模式为增强模式
echo 0 > /sys/class/disp/disp/attr/disp;
echo 1 > /sys/class/disp/disp/attr/enhance_mode;
//设置disp1 的色彩增强的模式为柔和模式
echo 1 > /sys/class/disp/disp/attr/disp;
echo 2 > /sys/class/disp/disp/attr/enhance_mode;
//设置disp0 的色彩增强的模式为增加模式，并且开启演示模式
echo 0 > /sys/class/disp/disp/attr/disp;
echo 3 > /sys/class/disp/disp/attr/enhance_mode;
```

c/c++ 代码示例：

```
char sysfs_path[MAX_LENGTH];
char sysfs_data[MAX_DATA];
unisgned int disp = 0
unsigned int enhance_mode = 1;
snprintf(sysfs_path,sizeof(sysfs_full_path),"sys/class/disp/disp/attr/disp");
snprintf(sysfs_data, sizeof(sysfs_data),"%d",disp);
write_data(sysfs_path, sys_data, strlen(sysfs_data));
snprintf(sysfs_path,sizeof(sysfs_full_path),
"/sys/class/disp/disp/attr/enhance_mode");
snprintf(sysfs_data, sizeof(sysfs_data), "%d",enhance_mode);
write_data(sysfs_path, sys_data, strlen(sysfs_data));
```

3.10.2 enhance_bright/contrast/saturation/edge/detail/denoise

```
- SYSFS NODE

/sys/class/disp/disp/attr/disp
/sys/class/disp/disp/attr/disp_color_temperature_store /* 色温*/
/sys/class/disp/disp/attr/enhance_bright /* 亮度*/
/sys/class/disp/disp/attr/enhance_contrast /* 对比度*/
/sys/class/disp/disp/attr/enhance_saturation /* 饱和*/
/sys/class/disp/disp/attr/enhance_edge /* 边缘锐度*/
/sys/class/disp/disp/attr/enhance_detail /* 细节增强*/
```

```
/sys/class/disp/disp/attr/enhance_denoise /* 降噪*/

- ARGUMENTS

disp display channel，比如0: disp0, 1：disp1
enhance_xxx： 范围：0~10，数据越大，调节幅度越大。

- RETURNS

none

- DESCRIPTION

该接口用于设置图像的亮度/对比度/饱和度/边缘锐度/细节增强/降噪的调节幅度。
在设置之前请确保enhance_mode 的值为1，否则设置不会生效。
- DEMO

//设置disp0 的图像亮度为8
echo 0 > /sys/class/disp/disp/attr/disp;
echo 8 > /sys/class/disp/disp/attr/enhance_bright;
//设置disp1 的饱和度为5
echo 1 > /sys/class/disp/disp/attr/disp;
echo 5 > /sys/class/disp/disp/attr/enhance_saturation;
```

c/c++ 代码示例：

```
char sysfs_path[MAX_LENGTH];
char sysfs_data[MAX_DATA];
unisgned int disp = 0
unsigned int enhance_bright = 80;
snprintf(sysfs_path,sizeof(sysfs_full_path),"sys/class/disp/disp/attr/disp");
snprintf(sysfs_data, sizeof(sysfs_data),"%d",disp);
write_data(sysfs_path, sys_data, strlen(sysfs_data));
snprintf(sysfs_path,sizeof(sysfs_full_path),
"/sys/class/disp/disp/attr/enhance_bright");
snprintf(sysfs_data, sizeof(sysfs_data), "%d",enhance_bright);
write_data(sysfs_path, sys_data, strlen(sysfs_data));
```

3.11 Data Structure

3.11.1 disp_fb_info

- 作用：用于描述一个 display frambuffer 的属性信息
- 成员：addr：frame buffer 的内容地址，对于 interleaved 类型，只有 addr[0] 有效；planar 类型，三个都有效；UV combined 的类型 addr[0] 和 addr[1] 有效 size：size of framebuffer, 单位为 pixel align：对齐位宽为 2 的指数 format: pixel format, 详见 disp_pixel_format color_space: color space mode, 详见 disp_cs_mode pre_multiply: 是否进行预乘操作标记 trd_right_addr: used when in frame packing 3d mode crop: 用于显示的 buffer 裁减区 flags: 标识 2D 或 3D 的 buffer scan: 标识描述类型，progress, interleaved
- 结构定义：

```
typedef struct
{
  unsigned long long addr[3]; /* address of frame buffer, single addr for interleaved fomart, double addr for semi-
    planar fomart triple addr for planar format */
  disp_rectsz size[3]; //size for 3 component,unit: pixels
  unsigned int align[3]; //align for 3 comonent,unit: bytes(align=2^n,i.e. 1/2/4/8/16/32..)
  disp_pixel_format format;
  disp_color_space color_space; //color space
  unsigned int trd_right_addr[3];/* right address of 3d fb, used when in frame packing 3d mode */
  bool pre_multiply; //true: pre-multiply fb
  disp_rect64 crop; //crop rectangle boundaries
  disp_buffer_flags flags; //indicate stereo or non-stereo buffer
  disp_scan_flags scan; //scan type & scan order
}disp_fb_info;
```

3.11.2 disp_layer_info

- 作用：用于描述一个图层的属性信息
- 成员：mode：图层的模式，详见 disp_layer_mode zorder：layer zorder, 优先级高的图层可能会覆盖优先级低的图层 alpha_mode：0:pixel alpha, 1:global alpha, 2:global pixel alpha alpha_value: layer global alpha value，valid while alpha_mode(1/2) screenn_win: screen window，图层在屏幕上显示的矩形窗口 fb: framebuffer 的属性，详见 disp_fb_info，valid when BUFFER_MODE color: display color, valid when COLOR_MODE b_trd_out: if output in 3d mode，used for scaler layer out_trd_mode: output 3d mode，详见 disp_3d_out_mode id: frame id, 设置给驱动的图像帧号，可以通过 DISP_LAYER_GET_FRAME_ID 获取当前显示的帧号以做一下特定的处理，比如释放掉已经显示完成的图像帧 buffer
- 结构定义：

```
- PROTOTYPE

typedef struct
{
  disp_layer_mode mode;
  unsigned char zorder; /*specifies the front-to-back ordering of the layers on the screen, the top layer having the
    highest Z value can't set zorder, but can get */
  unsigned char alpha_mode; //0: pixel alpha; 1: global alpha; 2: global pixel alpha
  unsigned char alpha_value; //global alpha value
  disp_rect screen_win; //display window on the screen
  bool b_trd_out; //3d display
  disp_3d_out_mode out_trd_mode;//3d display mode
  union {
    unsigned int color; //valid when LAYER_MODE_COLOR
    disp_fb_info fb; //framebuffer, valid when LAYER_MODE_BUFFER
  };
  unsigned int id; /* frame id, can get the id of frame
  display currently by DISP_LAYER_GET_FRAME_ID */
}disp_layer_info;
```

3.11.3 disp_layer_config

- 作用：用于描述一个图层配置的属性信息
- 成员：info：图像的信息属性 enable：使能标志 channel：图层所在的通道 id（0/1/2/3）layer_id：图层的 id。此 id 是在通道内的图层 id。即 (channel,layer_id)=(0,0) 表示通道 0 中的图层 0 之意。
- 结构定义：

```
typedef struct
{
  disp_layer_info info;
  bool enable;
  unsigned int channel;
  unsigned int layer_id;
}disp_layer_config;
```

3.11.4 disp_layer_config2

- 作用：用于描述一个图层配置的属性信息，与 disp_layer_config 的差别在于支持的功能更多，支持 ATW/FBD/HDR 功能。该结构体只能使用 DISP_LAYER_SET_CONFIG2 命令接口。
- 成员：format: 数据宽度会在 format 中体现出来 atw: 异步时移信息，详细见 struct disp_atw_info eotf: 光电转换特性信息，HDR 图像时需要，定义见 disp_eotf metadata_buf: 指向携带 metadata 的 buffer 的地址 metadata_size: metadata buffer 的大小 metadata_flag: 标识 metadata buffer 中携带的信息类型其他: 参考前述
- 结构定义：

```
- PROTOTYPE

/\* disp_fb_info2 - image buffer info v2
\*
\* @addr: buffer address for each plane
\* @size: size<width,height> for each buffer, unit:pixels
\* @align: align for each buffer, unit:bytes
\* @format: pixel format
\* @color_space: color space
\* @trd_right_addr: the right-eye buffer address for each plane,
\* valid when frame-packing 3d buffer input
\* @pre_multiply: indicate the pixel use premultiplied alpha
\* @crop: crop rectangle for buffer to be display
\* @flag: indicate stereo/non-stereo buffer
\* @scan: indicate interleave/progressive scan type, and the scan order
\* @metadata_buf: the phy_address to the buffer contained metadata for
fbc/hdr
\* @metadata_size: the size of metadata buffer, unit:bytes
\* @metadata_flag: the flag to indicate the type of metadata buffer
\* 0 : no metadata
\* 1 << 0: hdr static metadata
\* 1 << 1: hdr dynamic metadata
\* 1 << 4: frame buffer compress(fbc) metadata
```

```
\* x : all type could be "or" together
\*/
struct disp_fb_info2 {
   unsigned long long addr[3];
   struct disp_rectsz size[3];
   unsigned int align[3];
   enum disp_pixel_format format;
   enum disp_color_space color_space;
   unsigned int trd_right_addr[3];
   bool pre_multiply;
   struct disp_rect64 crop;
   enum disp_buffer_flags flags;
   enum disp_scan_flags scan;
   enum disp_eotf eotf;
   unsigned long long metadata_buf;
   unsigned int metadata_size;
   unsigned int metadata_flag;
};
/\* disp_layer_info2 - layer info v2
\*
\* @mode: buffer/clolor mode, when in color mode, the layer is widthout buffer
\* @zorder: the zorder of layer, 0~max-layer-number
\* @alpha_mode:
\* 0: pixel alpha;
\* 1: global alpha
\* 2: mixed alpha, compositing width pixel alpha before global alpha
\* @alpha_value: global alpha value, valid when alpha_mode is not pixel alpha
\* @screen_win: the rectangle on the screen for fb to be display
\* @b_trd_out: indicate if 3d display output
\* @out_trd_mode: 3d output mode, valid when b_trd_out is true
\* @color: the color value to be display, valid when layer is in color mode
\* @fb: the framebuffer info related width the layer, valid when in buffer mode
\* @id: frame id, the user could get the frame-id display currently by
\* DISP_LAYER_GET_FRAME_ID ioctl
\* @atw: asynchronous time wrap information
\*/
struct disp_layer_info2 {
   enum disp_layer_mode mode;
   unsigned char zorder;
   unsigned char alpha_mode;
   unsigned char alpha_value;
   struct disp_rect screen_win;
   bool b_trd_out;
   enum disp_3d_out_mode out_trd_mode;
   union {
      unsigned int color;
      struct disp_fb_info2 fb;
   };
   unsigned int id;
   struct disp_atw_info atw;
};
/\* disp_layer_config2 - layer config v2
\*
\* @info: layer info
\* @enable: indicate to enable/disable the layer
\* @channel: the channel index of the layer, 0~max-channel-number
\* @layer_id: the layer index of the layer widthin it's channel
\*/

struct disp_layer_config2 {
```

```
    struct disp_layer_info2 info;
    bool enable;
    unsigned int channel;
    unsigned int layer_id;
};
```

3.11.5 disp_color_info

- 作用：用于描述一个颜色的信息
- 成员：alpha：颜色的透明度 red：红 green：绿 blue：蓝
- 结构定义：

```
- PROTOTYPE

typedef struct
{
    u8 alpha;
    u8 red;
    u8 green;
    u8 blue;
}disp_color_info;
```

3.11.6 disp_rect

- 作用：用于描述一个矩形窗口的信息
- 成员：x：起点 x 值 y：起点 y 值 width：宽 height：高
- 结构定义：

```
typedef struct
{
    s32 x;
    s32 y;
    u32 width;
    u32 height;
}disp_rect;
```

3.11.7 disp_rect64

- 作用：用于描述一个矩形窗口的信息
- 成员：x：起点 x 值, 定点小数，高 32bit 为整数，低 32bit 为小数 y：起点 y 值, 定点小数，高 32bit 为整数，低 32bit 为小数 width：宽, 定点小数，高 32bit 为整数，低 32bit 为小数 height：高, 定点小数，高 32bit 为整数，低 32bit 为小数
- 结构定义：

```
typedef struct
{
  long long x;
  long long y;
  long long width;
  long long height;
}disp_rect64;
```

3.11.8 disp_position

- 作用：用于描述一个坐标的信息
- 结构定义：

```
typedef struct
{
  s32 x;
  s32 y;
}disp_posistion;
```

3.11.9 disp_rectsz

- 作用：用于描述一个矩形尺寸的信息
- 结构定义：

```
typedef struct
{
  u32 width;
  u32 height;
}disp_rectsz;
```

3.11.10 disp_atw_info

- 作用：用于描述图层的 asynchronous time wrap(异步时移) 信息
- 成员：used：是否开启 mode: ATW 的模式，左右或上下模式 b_row：宏块的行数 b_col：宏块的列数 cof_addr: ATW 系数 buffer 的地址
- 结构定义：

```
/\* disp_atw_mode - mode for asynchronous time warp
\*
\* @NORMAL_MODE: dual buffer, left eye and right eye buffer is individual
\* @LEFT_RIGHT_MODE: single buffer, the left half of each line buffer
\* is for left eye, the right half is for the right eye
\* @UP_DOWN_MODE: single buffer, the first half of the total buffer
\* is for the left eye, the second half is for the right eye
\*/
enum disp_atw_mode {
```

```
    NORMAL_MODE,
    LEFT_RIGHT_MODE,
    UP_DOWN_MODE,
};
/\* disp_atw_info - asynchronous time wrap infomation
\*
\* @used: indicate if the atw funtion is used
\* @mode: atw mode
\* @b_row: the row number of the micro block
\* @b_col: the column number of the micro block
\* @cof_addr: the address of buffer contaied coefficient for atw
\*/
struct disp_atw_info {
    bool used;
    enum disp_atw_mode mode;
    unsigned int b_row;
    unsigned int b_col;
    unsigned long cof_addr;
};
```

3.11.11 disp_pixel_format

- 作用：用于描述像素格式
- 成员：DISP_FORMAT_ARGB_8888: 32bpp，A 在最高位，B 在最低位 DISP_FORMAT_YUV420_P：planar yuv 格式，分三块存放，需三个地址，P3 在最高位 DISP_FORMAT_YUV422_SP_UVUV: semi-planar yuv 格式，分两块存放，需两个地址，UV 的顺序为 U 在低位；DISP_FORMAT_YUV420_SP_UVUV 类似 DISP_FORMAT_YUV422_SP_VUVU: semi-planar yuv 格式，分两块存放，需两个地址，UV 的顺序为 V 在低位；DISP_FORMAT_YUV420_SP_VUVU 类似
- 结构定义：

```
typedef enum
{
    DISP_FORMAT_ARGB_8888 = 0x00,//MSB A-R-G-B LSB
    DISP_FORMAT_ABGR_8888 = 0x01,
    DISP_FORMAT_RGBA_8888 = 0x02,
    DISP_FORMAT_BGRA_8888 = 0x03,
    DISP_FORMAT_XRGB_8888 = 0x04,
    DISP_FORMAT_XBGR_8888 = 0x05,
    DISP_FORMAT_RGBX_8888 = 0x06,
    DISP_FORMAT_BGRX_8888 = 0x07,
    DISP_FORMAT_RGB_888 = 0x08,
    DISP_FORMAT_BGR_888 = 0x09,
    DISP_FORMAT_RGB_565 = 0x0a,
    DISP_FORMAT_BGR_565 = 0x0b,
    DISP_FORMAT_ARGB_4444 = 0x0c,
    DISP_FORMAT_ABGR_4444 = 0x0d,
    DISP_FORMAT_RGBA_4444 = 0x0e,
    DISP_FORMAT_BGRA_4444 = 0x0f,
    DISP_FORMAT_ARGB_1555 = 0x10,
    DISP_FORMAT_ABGR_1555 = 0x11,
    DISP_FORMAT_RGBA_5551 = 0x12,
    DISP_FORMAT_BGRA_5551 = 0x13,
```

```
    /\* SP: semi-planar, P:planar, I:interleaved
    \* UVUV: U in the LSBs; VUVU: V in the LSBs \*/
    DISP_FORMAT_YUV444_I_AYUV = 0x40,//MSB A-Y-U-V LSB
    DISP_FORMAT_YUV444_I_VUYA = 0x41,//MSB V-U-Y-A LSB
    DISP_FORMAT_YUV422_I_YVYU = 0x42,//MSB Y-V-Y-U LSB
    DISP_FORMAT_YUV422_I_YUYV = 0x43,//MSB Y-U-Y-V LSB
    DISP_FORMAT_YUV422_I_UYVY = 0x44,//MSB U-Y-V-Y LSB
    DISP_FORMAT_YUV422_I_VYUY = 0x45,//MSB V-Y-U-Y LSB
    DISP_FORMAT_YUV444_P = 0x46,//MSB P3-2-1-0 LSB, YYYY UUUU VVVV
    DISP_FORMAT_YUV422_P = 0x47,//MSB P3-2-1-0 LSB, YYYY UU VV
    DISP_FORMAT_YUV420_P = 0x48,//MSB P3-2-1-0 LSB, YYYY U V
    DISP_FORMAT_YUV411_P = 0x49,//MSB P3-2-1-0 LSB, YYYY U V
    DISP_FORMAT_YUV422_SP_UVUV = 0x4a,//MSB V-U-V-U LSB
    DISP_FORMAT_YUV422_SP_VUVU = 0x4b,//MSB U-V-U-V LSB
    DISP_FORMAT_YUV420_SP_UVUV = 0x4c,
    DISP_FORMAT_YUV420_SP_VUVU = 0x4d,
    DISP_FORMAT_YUV411_SP_UVUV = 0x4e,
    DISP_FORMAT_YUV411_SP_VUVU = 0x4f,
    DISP_FORMAT_8BIT_GRAY = 0x50,
    DISP_FORMAT_YUV444_I_AYUV_10BIT = 0x51,
    DISP_FORMAT_YUV444_I_VUYA_10BIT = 0x52,
    DISP_FORMAT_YUV422_I_YVYU_10BIT = 0x53,
    DISP_FORMAT_YUV422_I_YUYV_10BIT = 0x54,
    DISP_FORMAT_YUV422_I_UYVY_10BIT = 0x55,
    DISP_FORMAT_YUV422_I_VYUY_10BIT = 0x56,
    DISP_FORMAT_YUV444_P_10BIT = 0x57,
    DISP_FORMAT_YUV422_P_10BIT = 0x58,
    DISP_FORMAT_YUV420_P_10BIT = 0x59,
    DISP_FORMAT_YUV411_P_10BIT = 0x5a,
    DISP_FORMAT_YUV422_SP_UVUV_10BIT = 0x5b,
    DISP_FORMAT_YUV422_SP_VUVU_10BIT = 0x5c,
    DISP_FORMAT_YUV420_SP_UVUV_10BIT = 0x5d,
    DISP_FORMAT_YUV420_SP_VUVU_10BIT = 0x5e,
    DISP_FORMAT_YUV411_SP_UVUV_10BIT = 0x5f,
    DISP_FORMAT_YUV411_SP_VUVU_10BIT = 0x60,
  }disp_pixel_format;;
```

3.11.12 disp_data_bits

- 作用：用于描述图像的数据宽度
- 结构定义：

```
  enum disp_data_bits {
    DISP_DATA_8BITS = 0,
    DISP_DATA_10BITS = 1,
    DISP_DATA_12BITS = 2,
    DISP_DATA_16BITS = 3,
  };
```

3.11.13 disp_eotf

- 作用：用于描述图像的光电转换特性

- 结构定义：

```
enum disp_eotf {
    DISP_EOTF_RESERVED = 0x000,
    DISP_EOTF_BT709 = 0x001,
    DISP_EOTF_UNDEF = 0x002,
    DISP_EOTF_GAMMA22 = 0x004, /* SDR */
    DISP_EOTF_GAMMA28 = 0x005,
    DISP_EOTF_BT601 = 0x006,
    DISP_EOTF_SMPTE240M = 0x007,
    DISP_EOTF_LINEAR = 0x008,
    DISP_EOTF_LOG100 = 0x009,
    DISP_EOTF_LOG100S10 = 0x00a,
    DISP_EOTF_IEC61966_2_4 = 0x00b,
    DISP_EOTF_BT1361 = 0x00c,
    DISP_EOTF_IEC61966_2_1 = 0X00d,
    DISP_EOTF_BT2020_0 = 0x00e,
    DISP_EOTF_BT2020_1 = 0x00f,
    DISP_EOTF_SMPTE2084 = 0x010, /* HDR10 */
    DISP_EOTF_SMPTE428_1 = 0x011,
    DISP_EOTF_ARIB_STD_B67 = 0x012, /* HLG */
};
```

3.11.14 disp_buffer_flags

- 作用：用于描述 3D 源模式
- 成员：DISP_BF_NORMAL: 2d DISP_BF_STEREO_TB: top bottom 模式 DISP_BF_STEREO_FP: framepacking DISP_BF_STEREO_SSF: side by side full, 左右全景 DISP_BF_STEREO_SSH: side by side half, 左右半景 DISP_BF_STEREO_LI: line interleaved, 行交错模式
- 结构定义：

```
typedef enum
{
    DISP_BF_NORMAL = 0,//non-stereo
    DISP_BF_STEREO_TB = 1 << 0,//stereo top-bottom
    DISP_BF_STEREO_FP = 1 << 1,//stereo frame packing
    DISP_BF_STEREO_SSH = 1 << 2,//stereo side by side half
    DISP_BF_STEREO_SSF = 1 << 3,//stereo side by side full
    DISP_BF_STEREO_LI = 1 << 4,//stereo line interlace
}disp_buffer_flags;
```

3.11.15 disp_3d_out_mode

- 作用：用于描述 3D 输出模式
- 成员：DISP_3D_OUT_MODE_CI_1：列交织 DISP_3D_OUT_MODE_CI_2：列交织 DISP_3D_OUT_MODE_CI_3：列交织 DISP_3D_OUT_MODE_CI_4：列交织 DISP_3D_OUT_MODE_LIRGB：行交织 DISP_3D_OUT_MODE_TB: top bottom 上下模式 DISP_3D_OUT_MODE_FP: framepacking DISP_3D_OUT_MODE_SSF: side by side full, 左右

全景 DISP_3D_OUT_MODE_SSH: side by side half, 左右半景 DISP_3D_OUT_MODE_LI: line interleaved, 行交织 DISP_3D_OUT_MODE_FA: field alternate 场交错

- 结构定义：

```
typedef enum
{
  //for lcd
  DISP_3D_OUT_MODE_CI_1 = 0x5,//column interlaved 1
  DISP_3D_OUT_MODE_CI_2 = 0x6,//column interlaved 2
  DISP_3D_OUT_MODE_CI_3 = 0x7,//column interlaved 3
  DISP_3D_OUT_MODE_CI_4 = 0x8,//column interlaved 4
  DISP_3D_OUT_MODE_LIRGB = 0x9,//line interleaved rgb
  //for hdmi
  DISP_3D_OUT_MODE_TB = 0x0,//top bottom
  DISP_3D_OUT_MODE_FP = 0x1,//frame packing
  DISP_3D_OUT_MODE_SSF = 0x2,//side by side full
  DISP_3D_OUT_MODE_SSH = 0x3,//side by side half
  DISP_3D_OUT_MODE_LI = 0x4,//line interleaved
  DISP_3D_OUT_MODE_FA = 0xa,//field alternative
}disp_3d_out_mode;
```

3.11.16 disp_color_space

- 作用：用于描述颜色空间类型
- 成员：DISP_BT601：用于标清视频，SDR 模式 DISP_BT709：用于高清视频，SDR 模式 DISP_BT2020NC：用于 HDR 模式
- 结构定义：

```
enum disp_color_space
{
  DISP_UNDEF = 0x00,
  DISP_UNDEF_F = 0x01,
  DISP_GBR = 0x100,
  DISP_BT709 = 0x101,
  DISP_FCC = 0x102,
  DISP_BT470BG = 0x103,
  DISP_BT601 = 0x104,
  DISP_SMPTE240M = 0x105,
  DISP_YCGCO = 0x106,
  DISP_BT2020NC = 0x107,
  DISP_BT2020C = 0x108,
  DISP_GBR_F = 0x200,
  DISP_BT709_F = 0x201,
  DISP_FCC_F = 0x202,
  DISP_BT470BG_F = 0x203,
  DISP_BT601_F = 0x204,
  DISP_SMPTE240M_F = 0x205,
  DISP_YCGCO_F = 0x206,
  DISP_BT2020NC_F = 0x207,
  DISP_BT2020C_F = 0x208,
  DISP_RESERVED = 0x300,
  DISP_RESERVED_F = 0x301,
};
```

3.11.17 disp_csc_type

- 作用：用于描述图像颜色格式
- 结构定义：

```
enum disp_csc_type
{
  DISP_CSC_TYPE_RGB = 0,
  DISP_CSC_TYPE_YUV444 = 1,
  DISP_CSC_TYPE_YUV422 = 2,
  DISP_CSC_TYPE_YUV420 = 3,
};
```

3.11.18 disp_output_type

- 作用：用于描述显示输出类型
- 成员：DISP_OUTPUT_TYPE_NONE：无显示输出 DISP_OUTPUT_TYPE_LCD: LCD 输出 DISP_OUTPUT_TYPE_TV: TV 输出 DISP_OUTPUT_TYPE_HDMI: HDMI 输出 DISP_OUTPUT_TYPE_VGA: VGA 输出
- 结构定义：

```
typedef enum
{
  DISP_OUTPUT_TYPE_NONE = 0,
  DISP_OUTPUT_TYPE_LCD = 1,
  DISP_OUTPUT_TYPE_TV = 2,
  DISP_OUTPUT_TYPE_HDMI = 4,
  DISP_OUTPUT_TYPE_VGA = 8,
}disp_output_type;
```

3.11.19 disp_tv_mode

- 作用：用于描述 TV 输出模式
- 结构定义：

```
typedef enum
{
  DISP_TV_MOD_480I = 0,
  DISP_TV_MOD_576I = 1,
  DISP_TV_MOD_480P = 2,
  DISP_TV_MOD_576P = 3,
  DISP_TV_MOD_720P_50HZ = 4,
  DISP_TV_MOD_720P_60HZ = 5,
  DISP_TV_MOD_1080I_50HZ = 6,
  DISP_TV_MOD_1080I_60HZ = 7,
  DISP_TV_MOD_1080P_24HZ = 8,
  DISP_TV_MOD_1080P_50HZ = 9,
```

```
    DISP_TV_MOD_1080P_60HZ = 0xa,
    DISP_TV_MOD_1080P_24HZ_3D_FP = 0x17,
    DISP_TV_MOD_720P_50HZ_3D_FP = 0x18,
    DISP_TV_MOD_720P_60HZ_3D_FP = 0x19,
    DISP_TV_MOD_1080P_25HZ = 0x1a,
    DISP_TV_MOD_1080P_30HZ = 0x1b,
    DISP_TV_MOD_PAL = 0xb,
    DISP_TV_MOD_PAL_SVIDEO = 0xc,
    DISP_TV_MOD_NTSC = 0xe,
    DISP_TV_MOD_NTSC_SVIDEO = 0xf,
    DISP_TV_MOD_PAL_M = 0x11,
    DISP_TV_MOD_PAL_M_SVIDEO = 0x12,
    DISP_TV_MOD_PAL_NC = 0x14,
    DISP_TV_MOD_PAL_NC_SVIDEO = 0x15,
    DISP_TV_MOD_3840_2160P_30HZ = 0x1c,
    DISP_TV_MOD_3840_2160P_25HZ = 0x1d,
    DISP_TV_MOD_3840_2160P_24HZ = 0x1e,
    DISP_TV_MODE_NUM = 0x1f,
}disp_tv_mode;
```

3.11.20 disp_output

- 作用：用于描述显示输出类型，模式
- 成员：Type：输出类型 Mode：输出模式，480P/576P, etc.
- 结构定义：

```
struct disp_output
{
    unsigned int type;
    unsigned int mode;
};
```

3.11.21 disp_layer_mode

- 作用：用于描述图层模式
- 枚举值：LAYER_MODE_BUFFER: buffer 模式，带 buffer 的图层 LAYER_MODE_COLOR: 单色模式，无 buffer 的图层，只需要一个颜色值表示图像内容
- 结构定义：

```
enum disp_layer_mode
{
    LAYER_MODE_BUFFER = 0,
    LAYER_MODE_COLOR = 1,
};
```

3.11.22 disp_device_config

- 作用：用于描述输出设备的属性信息
- 成员：type：设备类型，如 HDMI/TV/LCD 等 mode：分辨率 format：输出的数据格式，比如 RGB/YUV444/422/420 bits：输出的数据位宽，8/10/12/16bits eotf：光电特性信息 cs：输出的颜色空间类型
- 结构定义：

```
/\* disp_device_config - display deivce config
\*
\* @type: output type
\* @mode: output mode
\* @format: data format
\* @bits: data bits
\* @eotf: electro-optical transfer function
\* SDR : DISP_EOTF_GAMMA22
\* HDR10: DISP_EOTF_SMPTE2084
\* HLG : DISP_EOTF_ARIB_STD_B67
\* @cs: color space type
\* DISP_BT601: SDR for SD resolution(< 720P)
\* DISP_BT709: SDR for HD resolution(>= 720P)
\* DISP_BT2020NC: HDR10 or HLG or wide-color-gamut
\*/
struct disp_device_config {
    enum disp_output_type type;
    enum disp_tv_mode mode;
    enum disp_csc_type format;
    enum disp_data_bits bits;
    enum disp_eotf eotf;
    enum disp_color_space cs;
    unsigned int reserve1;
    unsigned int reserve2;
    unsigned int reserve3;
    unsigned int reserve4;
    unsigned int reserve5;
    unsigned int reserve6;
};
```

4 调试方法

4.1 查看显示模块的状态

```
cat /sys/class/disp/disp/attr/sys
```

示例如下：

```
# cat /sys/class/disp/disp/attr/sys
screen[0] -> dev[1]
de_rate 450000000 hz, ref_fps:60
mgr0: 1200x1920 fmt[rgb] cs[0x204] range[full] eotf[0x4] bits[8bits] err[0] force_sync[615] unblank direct_show[false]
        iommu[1] rcq_en[1]
rcq info: rcq_irq[2517] rcq_update_rquest[2431] rcq_update_req_irq[3933] rcq_finish_irq[3934]
dmabuf: cache[16] cache max[24] umap skip[0] umap skip max[0]
      lcd output     backlight( 84)  fps:59.5       1200x1920
      err:0   skip:0  irq:3934       vsync:3934     vsync_skip:0
  BUF    enable ch[0] lyr[0] z[2] prem[Y] a[globl 255] fmt[  1] fbd_type[none] fb[1200,  36;  0,  0;  0,  0] crop[  0,  0,1200,
         36] frame[  0,  0,1200,  36] addr[fed80000,fed8a8c0,fed95180] flags[0x      0] trd[0,0] depth[ 0] transf[0] sampling
         [1200:1200,1920:1920]
  BUF    enable ch[0] lyr[1] z[3] prem[Y] a[globl 255] fmt[  1] fbd_type[none] fb[1200,  72;  0,  0;  0,  0] crop[  0,  0,1200,
         72] frame[  0,1848,1200,  72] addr[fed00000,fed15180,fed2a300] flags[0x      0] trd[0,0] depth[ 0] transf[0] sampling
         [1200:1200,1920:1920]
  BUF    enable ch[1] lyr[0] z[0] prem[Y] a[globl 255] fmt[  1] fbd_type[afbd] fb[1792,1360;  0,  0;  0,  0] crop[  38,  61,
         767,1228] frame[  0,  0,1200,1920] addr[f9000000,f9253000,f94a6000] flags[0x      0] trd[0,0] depth[ 0] transf[0]
         sampling[ 767: 767,1228:1228]
  BUF    enable ch[2] lyr[0] z[1] prem[Y] a[globl 255] fmt[  1] fbd_type[afbd] fb[1216,1920;  0,  0;  0,  0] crop[  0,
         0,1200,1920] frame[  0,  0,1200,1920] addr[f8600000,f883a000,f8a74000] flags[0x      0] trd[0,0] depth[ 0] transf[0]
         sampling[1200:1200,1920:1920]
disp[0]all:2463, sub:2463, cur:2463, free:2436, skip:0
```

Display 系统信息如下：

```
screen[0] -> dev[1]：硬件底层连接关系，DE1 -> Tcon1；
de_rate：DE的工作频率；
ref_fps：参考fps，单位hz；
mgr0：逻辑显示实例id index；
1200x1920：设备输出分辨率；
fmt[rgb] cs[0x204] range[full] eotf[0x4] bits[8bits]：设备属性，依次表示：输出格式，颜色范围，eotf，输出bit；
unblank：设备是否blank，unblank 开启显示，blank 关闭显示；
direct_show：cvbs的直显模式；
iommu：iommu是否使能；
rcq_en：rcq更新regs是否使能

rcq info：
  rcq_irq[2517]：rcq 完成中断次数；
  rcq_update_rquest[2431]：rcq 更新请求次数；
  rcq_update_req_irq[3933]：rcq 更新请求时的vsync计数；
  rcq_finish_irq[3934]: rcq 完成中断时的vsync计数；

dmabuf:
```

cache[16]：驱动内部缓存的dma_buf数量，若这个数超过50，则可能导致系统dmabuf内存被耗尽，从而无法map；
cache max[24]：驱动内部缓存的dma_buf某个时刻出现的最大值，也就是cache的峰值；
umap skip[0]：跳过umap dma_buf的次数，由于某些特殊情况，驱动或者无法判断是否有新帧被显示，送新帧时无法释放前面的显示帧情况发生，表示一次umap skip；
umap skip max[0]：跳过umap dma_buf的次数的最大值，也就是umap skip的峰值；

err：de 缺数的次数，de 缺数可能会出现屏幕抖动，花屏的问题。de 缺数一般为带宽不足引起；
skip：表示de 跳帧的次数，跳帧会出现卡顿问题。跳帧是指本次中断响应较慢，de 模块判断在本次中断已经接近或者超过了消隐区，将放弃本次更新图像的机会，选择继续显示原有的图像；
irq：表示该通路上垂直消隐区中断执行的次数，一直增长表示该通道上的timing controller 正在运行当中；
vsync：表示显示模块往用户空间中发送的vsync 消息的数目，一直增长表示正在不断地发送中；

图层各信息描述如下：

```
BUF   enable ch[2] lyr[0] z[1] prem[Y] a[globl 255] fmt[  1] fbd_type[afbd] fb[1216,1920;  0,  0;  0,  0] crop[  0,
      0,1200,1920] frame[  0,  0,1200,1920] addr[f8600000,f883a000,f8a74000] flags[0x     0] trd[0,0] depth[ 0] transf[0]
      sampling[1200:1200,1920:1920]

BUF: 图层类型，BUF/COLOR，一般为BUF，即图层是带BUFFER 的。COLOR 意思是显示一个纯色的画面，不带BUFFER。
enable: 显示处于enable 状态
ch[0]: 该图层处于blending 通道0
lyr[0]: 该图层处于当前blending 通道中的图层0
z[0]: 图层z 序，越小越在底部，可能会被z 序大的图层覆盖住
prem[Y]: 是否预乘格式，Y 是，N 否
a: alpha 参数， globl/pixel/alpha 值
fmt: 图层格式，值64 以下为RGB 格式；以上为YUV 格式，常见的72 为YV12,76 为NV12
fb: 图层buffer 的size，width,height，三个分量
crop: 图像buffer 中的裁减区域， [x,y,w,h]
frame: 图层在屏幕上的显示区域，[x,y,w,h]
addr: 三个分量的地址
flags: 一般为0， 3D SS 时0x4, 3D TB 时为0x1, 3D FP 时为0x2;
trd: 是否3D 输出，3D 输出的类型（HDMI FP 输出时为1）各counter 描述如下：
sampling：对输入图层采样过滤信息[in_w:out_w,in_h:out_h]
```

hw composer 相关信息如下（只在 android 方案中有效）：

```
disp[0]all:2463, sub:2463, cur:2463, free:2436, skip:0
all：fence timeline总数
submmit_count：最后一次送显的fence 序号
free_count：已经释放的fence的序号
current_count：当前正在显示的fence 序号
skip_count：跳帧总数，说明在disp 中存在丢帧情况，原因为在一个active 区内hwcomposer 传递多于一帧的图像帧下来
```

调试说明：

1. 对于android 系统，可以dumpsys SurfaceFlinger 打印surface 的信息，如果信息与disp 中sys 中的信息不一致，很大可能是hwc 的转换存在问题。
2. 如果发现图像刷新比较慢，存在卡顿问题，可以看一下输出设备的刷新率，对比一下ref_fps 与fps 是否一致，如果不一致，说明tcon 的时钟频率或timing 没配置正确。如果ref_fps 与屏的spec 不一致，则需要检查sys_config 中的时钟频率和timing配置是否正确。屏一般为60Hz，而如果是TV 或HDMI，则跟模式有关，比较常见的为60/50/30/24Hz。
   如果是android 方案，还可以看一下display 与release 的counter 是否一致，如果相差太大，说明android 送帧不均匀，造成丢帧。
3. 如果发现图像刷新比较慢，存在卡顿问题，也需要看一下skip counter，如果skip
   counter 有增长，说明现在的系统负荷较重，对vblank 中断的响应较慢，出现跳帧，导致了图像卡顿问题。
4. 如果屏不亮，怀疑背光时，可以看一下屏的背光值是否为0。
   如果为0，说明上层传递下来的背光值不合理；如果不为0，背光还是不亮，则为驱动或硬件问题了。硬件上可以通过测量bl_en 以及pwm 的电压值来排查问题。
5. 如果花屏或图像抖动，可以查看err counter，如果err counter 有增长，则说明de缺数，有可能是带宽不足，或者瞬时带宽不足问题。

4.2 截屏

```
echo 0 > /sys/class/disp/disp/attr/disp
echo /data/filename.bmp > /sys/class/disp/disp/attr/capture_dump
```

该调试方法用于截取 DE 输出到 TCON 前的图像，用于显示通路上分段排查。如果截屏没有问题而界面异常，可以确定 TCON 到显示器间出错。第一个路径接受显示器索引 0 或 1；第二个路径接受文件路径。

4.3 colorbar

```
echo 0 > /sys/class/disp/disp/attr/disp
echo num > /sys/class/disp/disp/attr/colorbar
```

第一个路径接受显示器索引 0 或 1。第二个路径表示 TCON 选择的输入源。0，DE 输出；1-7，TCON 自检用的 colorbar；8,DE 自检用的 colorbar。

4.4 显示模块 debugfs 接口

4.4.1 总述

```
/* 目录：*/
# /sys/kernel/debug/dispdbg;

/* 挂载 */
# mount -t debugfs none /sys/kernel/debug;

/* 文件结点*/
# ls
# name command param start info
name: 表示操作的对象名字
command: 表示执行的命令
param: 表示该命令接收的参数
start: 输入1 开始执行命令
info: 保存命令执行的结果

只读，大小是1024 bytes。
```

4.4.2 切换显示输出设备

```
name: disp0/1/2 //表示显示通道0/1/2
command: switch
param: type mode
参数说明：type:0(none),1(lcd),2(tv),4(hdmi),8(vga)
mode 详见disp_tv_mode 定义
例子：
/* 显示通道0 输出LCD */
echo disp0 > name;echo switch > command;echo 1 0 > param;echo 1 > start;
/* 关闭显示通道0 的输出*/
echo disp0 > name;echo switch > command;echo 0 0 > param;echo 1 > start;
```

4.4.3 开关显示输出设备

```
name: disp0/1/2 //表示显示通道0/1/2
command: blank
param: 0/1
参数说明：1 表示blank，即关闭显示输出；0 表示unblank，即开启显示输出
例子：
/* 关闭显示通道0 的显示输出*/
echo disp0 > name;echo blank > command;echo 1 > param;echo 1 > start;
/* 开启显示通道1 的显示输出*/
echo disp1 > name;echo blank > command;echo 0 > param;echo 1 > start;
```

4.4.4 电源管理 (suspend/resume) 接口

```
name: disp0/1/2 //表示显示通道0/1/2
command: suspend/resume //休眠，唤醒命令
param: 无
sunxi 平台显示模块（disp2）使用文档密级：1
Tyle sunxi display2 模块使用文档(第60 页) 2013-9-12
CopyRight©2013 All Winner Technology, Right Reserved
例子：
/* 让显示模块进入休眠状态*/
echo disp0 > name;echo suspend > command;echo 1 > start;
/* 让显示模块退出休眠状态*/
echo disp1 > name;echo resume > command;echo 1 > start;
```

4.4.5 调节 lcd 屏幕背光

```
name: lcd0/1/2 //表示lcd0/1/2
command: setbl //设置背光亮度
param: xx
参数说明：背光亮度值，范围是0~255。
例子：
/* 设置背光亮度为100 */
echo lcd0 > name;echo setbl > command;echo 100 > param;echo 1 > start;
```

```
/* 设置背光亮度为0 */
echo lcd0 > name;echo setbl > command;echo 0 > param;echo 1 > start;
```

4.4.6 vsync 消息开关

```
name: disp0/1/2 //表示显示通道0/1/2
command: vsync_enable //开启/关闭vsync 消息
param: 0/1
参数说明：0：表示关闭； 1：表示开启
例子：
/* 关闭显示通道0 的vsync 消息*/
echo disp0 > name;echo vsync_enable > command;echo 0 > param;echo 1 > start;
/* 开启显示通道1 的vsync 消息*/
echo disp1 > name;echo vsync_enable > command;echo 1 > param;echo 1 > start;
```

4.4.7 查看 enhance 的状态

```
name: enhance0/1/2 //表示enhance0/1/2
command: getinfo //获取enhance 的状态
param: 无
例子：
/* 获取显示通道0 的enhance 状态信息*/
# echo enhance0 > name;echo getinfo > command;echo 1 > start;cat info;
# enhance 0: enable, normal
```

4.4.8 查看智能背光的状态

```
name: smbl0/1/2 //表示显示通道0/1/2
command: getinfo //获取smart backlight 的状态
param: 无
例子：
/* 获取显示通道0 的smbl 状态信息*/
# echo smbl0 > name;echo getinfo > command;echo 1 > start;cat info;
# smbl 0: disable, window<0,0,0,0>, backlight=0, save_power=0 percent
显示的是智能背光是否开启，有效窗口大小，当前背光值，省电比例
```

4.4.9 T5 平台如何调节色温, 亮度, 对比度, 饱和度的值

⚠ 注意

T5 平台仅在 ch0 通道上支持调节，请确保需要调节的显示内容在该通道上显示。

```
//设置disp0 的图像亮度为80
echo 0 > /sys/class/disp/disp/attr/disp;
echo 80 > /sys/class/disp/disp/attr/enhance_bright;
```

```
//设置disp1 的饱和度为50
echo 1 > /sys/class/disp/disp/attr/disp;
echo 50 > /sys/class/disp/disp/attr/enhance_saturation;
```

5 常见问题

5.1 黑屏（无背光）

问题现象：机器接 LCD 输出，发现 LCD 没有任何显示，仔细查看背光也不亮。

问题分析：此现象说明 LCD 背光供电不正常，不排除还有其他问题，但没背光的问题必须先解决。

问题排查步骤：

- 步骤一

使用电压表量 LCD 屏的各路电压，如果背光管脚电压不正常，请对照原理图确定背光电压对应 GPIO、电源或者 PWM 有没使能。否则，尝试换个屏再试。

- 步骤二

如果怀疑是 GPIO、电源没有使能，那需要对照原理图，在 board.dts 配置上

- 步骤三

如果怀疑是 PWM 没有使能，先看看随 sdk 有没发布 PWM 模块使用指南，如果有按照里面步骤进行排查。如果 sdk 没有发布 PWM 模块使用指南。可以 cat /sys/kernel/debug/pwm 看看有没输出。如果没有就是 PWM 驱动没有加载，请检查一下 menuconfig 有没打开。

- 步骤四

如果步骤三未解决问题，请排查 dts 或 board.dts 配置。如果还没有解决，可以寻求技术支持。

5.2 黑屏（有背光）

问题现象：机器接 LCD，发现有背光，界面输出黑屏。

问题分析：此现象说明没有内容输出，可能是 DE、TCON 出错或应用没有送帧。

问题排查步骤：

- 步骤一

根据 “查看显示模块的状态” 章节排查应用输入的图层信息是否正确。其中，宽高、显存的文件句柄出错问题最多。

- 步骤二

根据 “截屏” 章节截屏，看看 DE 输出是否正常。如果不正常，排查 DE 驱动配置是否正确；如果正常，接着下面步骤。

- 步骤三

根据 “colorbar” 章节输出 colorbar，如果 TCON 自身的 colorbar 也没有显示，排查硬件通路；如果有显示，排查 TCON 输入源选择的寄存器。后者概率很低，此时可寻求技术支持。

5.3 绿屏

问题现象：显示器出现绿屏，切换界面可能有其他变化。

问题分析：此现象说明处理图层时 DE 出错。可能是应用送显的 buffer 内容或者格式有问题；也可能 DE 配置出错。

问题排查步骤：

- 步骤一

根据 “查看显示模块的状态” 章节排查应用输入的图层信息是否正确。其中，图层格式填错的问题最多。

- 步骤二

导出 DE 寄存器，排查异常。此步骤比较复杂，需要寻求技术支持。

5.4 界面卡住

问题现象：界面定在一个画面，不再改变。

问题分析：此现象说明显示通路一般是正常的，只是应用没有继续送帧。

问题排查步骤：

- 步骤一

根据 “查看显示模块的状态” 章节排查应用输入的图层信息有没改变，特别关注图层的地址。

- 步骤二

排查应用送帧逻辑，特别关注死锁，线程、进行异常退出，fence 处理异常。

5.5 局部界面花屏

问题现象：画面切到特定场景时候，出现局部花屏，并不断抖动。

问题分析：此现象是典型的 DE scaler 出错现象。

问题排查步骤：

根据 “查看显示模块的状态” 章节查看问题出现时带缩放图层的参数。如果屏幕输出的宽 x 高除以 crop 的宽 x 高小于 1/16 或者大于 32，那么该图层不能走 DE 缩放，改用 GPU，或应用修改图层宽高。

5.6 快速切换界面花屏

问题现象：快速切换界面花屏，变化不大的界面显示正常。

问题分析：此现象是典型的性能问题，与显示驱动关系不大。

问题排查步骤：

- 步骤一

排查 DRAM 带宽是否满足场景需求。

- 步骤二

若是安卓系统，排查 fence 处理流程；若是纯 linux 系统，排查送帧流程、swap buffer、pandisplay 流程。

5.7 显示偏色

问题现象：图像显示颜色异常偏色，其他显示现象正常。问题分析：此现象是显示设备偏色问题，可能与颜色空间的配置不对相关。

问题排查步骤：

- 步骤一

查看显示节点颜色空间值是否异常

```
cat /sys/class/disp/disp/attr/sys
```

```
# cat /sys/class/disp/disp/attr/sys
screen 0:
de_rate 696000000 hz, ref fps:61
mgr0: 1280x800 fmt[rgb] cs[0x204] range[full] eotf[0x4] bits[8bits] err[0] force_sync[0] unblank direct_show[false]
dmabuf: cache[0] cache max[0] umap skip[0] overflow[0]
        lcd output      backlight( 50)  fps:61.4        1280x 800
        err:0   skip:18 irq:127547      vsync:0 vsync_skip:0
   BUF    enable ch[1] lyr[0] z[0] prem[N] a[globl 255] fmt[  0] fb[1280, 800;1280, 800;1280, 800] crop[   0,   0,1280, 800] frame[
 0,   0,1280, 800] addr[ff800000,       0,       0] flags[0x       0] trd[0,0]
depth[ 0] transf[0]
```

图 5-1: 显示调试节点

- 步骤二

一般的颜色空间配置原则是

```
DISP_BT601 = 0x104 (<720P)
DISP_BT709 = 0x101 (>=720p)
DISP_BT2020NC_F = 0x207 (4k )
```

如果颜色空间设置不对，排查应用层对颜色空间的改动。

5.8 调节亮度对比度饱和度

问题描述：T5 平台如何调节色温, 亮度, 对比度, 饱和度的值?

问题解决：可通过/sys/class/disp/disp/attr/内节点调节相关参数，参考 enhance 小节。

5.9 如何实现双显 LOGO 启动

问题描述：目前已支持双显的驱动版本，如何实现双显 LOGO 启动?

问题解决：

以 LVDS 1280x800 + HDMI 1920x1080 为例：

- 步骤一、修改 uboot-board.dts

```
&disp {
    ...
    dev_num            = <2>;  # 关键配置：指示有 2 个启动显示设备，不定义则默认 1 个
    dev0_output_type   = <1>;
    dev0_output_mode   = <4>;
    dev0_screen_id     = <0>;
    dev0_do_hpd        = <0>;

    dev1_output_type   = <4>;  # 配置副屏设备的信息
    dev1_output_mode   = <10>;
    dev1_screen_id     = <1>;
    dev1_do_hpd        = <1>;

    ...

    fb0_format         = <0>;
    fb0_width          = <1280>;
    fb0_height         = <800>;
    fb0_rot_used       = <0>;
    fb0_rot_degree     = <0>;

    fb1_format         = <0>;  # 配置副屏 fb 的信息
    fb1_width          = <1920>;
    fb1_height         = <1080>;
    fb1_rot_used       = <0>;
    fb1_rot_degree     = <0>;
    ...
};
```

- 步骤二、修改 board.dts

```
&disp {
  ...
  fb_format          = <0>;
  fb_num             = <2>;     # 关键配置：定义 2 个 framebuffer 承接 uboot logo
  fb_debug           = <1>;
  /*<disp channel layer zorder>*/
  fb0_map            = <0 1 0 16>; # 主显 framebuffer配置
  fb0_width          = <1280>;    # 宽高与 uboot 阶段保持一致
  fb0_height         = <800>;
  /*<disp channel layer zorder>*/
  fb1_map            = <1 1 0 16>; # 副显 framebuffer配置
  fb1_width          = <1920>;    # 宽高与 uboot 阶段保持一致
  fb1_height         = <1080>;
  ...
};
```

- 步骤三、接线上电启动

例如：LCD + HDMI：连接好 LCD 排线和 HDMI 线缆，上电启动，即可看到 2 个屏幕都显示出 LOGO；

- 步骤四、单独自定义 LOGO 图片

目前驱动的逻辑仅验证双显 LOGO 图片的可行性，所以 2 个屏幕都显示同一张 LOGO 图片（即：bootlogo.bmp）。如果想每个屏幕独立显示不同的启动 LOGO，只需改动以下驱动代码。

⚠ 注意

图片大小不能大于屏幕分辨率！

路径：<sdk 路径>/brandy/brandy-2.0/u-boot-2018/drivers/video/sunxi/logo_display/cmd_sunxi_bmp.c

```
int sunxi_bmp_display(char *name)
{
  ...
  /* 默认将 boot_logo.bmp 显示到每个屏幕的 framebuffer 中，每个屏幕 1 个 framebuffer。
     可自定义为：按不同名字显示到不同的 framebuffer 中。*/
  for (i = 0; i < get_framebuffer_num(); i++) {
    ret = show_bmp_on_fb(bmp_head_addr, i);
    if (ret != 0)
      pr_error("show bmp on fb failed !%d\n", ret);
  }
  ...
}
```

著作权声明

商标声明

免责声明