Android 10

Display User Guide

1.0

2020.02.27

文档履历

版本号	日期	制/修订人	内容描述
1.0	2020.02.27	AW	Initialize

目录

1. 概述

1.1 编写目的

让显示应用开发人员了解显示驱动的接口及使用流程，快速上手，进行开发；让新人接手工作时能快速地了解驱动接口，进行调试排查问题。

1.2 适用范围

本模块设计适用于 A133/A100 Android 10 平台。

1.3 适用人员

与显示相关的应用开发人员，与显示相关的其他模块的开发人员，以及新人。

2. 模块介绍

2.1 模块功能介绍

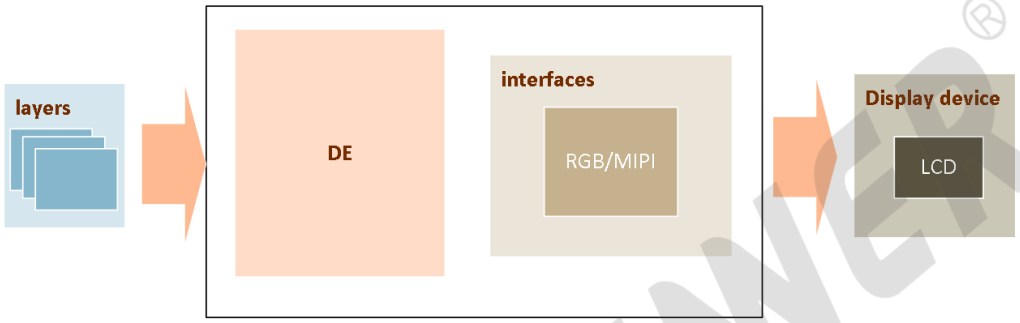

图 1: 模块框图

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。DE 一般有 1-2 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。DE 的每个独立的单元有 1-4 个通道（典型地，de0 有 4 上，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
- 支持多图层叠加混合处理
- 支持多种显示效果处理（alpha, colorkey, 图像增强，亮度/对比度/饱和度/色度调整）
- 支持智能背光调节
- 支持多种图像数据格式输入 (arg,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 硬件术语介绍

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 软件术语介绍

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。有时也指一块
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 模块配置说明

2.3.1 devicetree 通用配置说明

dts 路径为 kernel/linux-4.9/arch/arm64/boot/dts/sunxi/sun50iw10p1.dtsi。

- DE

```
disp: disp@01000000 {
        compatible = "allwinner,sunxi-disp";
        reg = <0x0 0x01000000 0x0 0x01400000>,/*de*/
        <0x0 0x06510000 0x0 0x200>,/*display_if_top*/
        <0x0 0x06511000 0x0 0x1000>,/*tcon_lcd0*/
        <0x0 0x06512000 0x0 0x1000>,/*tcon_lcd1*/
        <0x0 0x06515000 0x0 0x1000>,/*tcon_tv0*/
        <0x0 0x06516000 0x0 0x1000>;/*tcon_tv1*/
        interrupts = <GIC_SPI 88 IRQ_TYPE_LEVEL_HIGH>, /*DE*/
        <GIC_SPI 64 IRQ_TYPE_LEVEL_HIGH>, /*tcon_lcd0*/
        <GIC_SPI 65 IRQ_TYPE_LEVEL_HIGH>, /*tcon_lcd1*/
        <GIC_SPI 66 IRQ_TYPE_LEVEL_HIGH>, /*tcon_tv0*/
        <GIC_SPI 67 IRQ_TYPE_LEVEL_HIGH>; /*tcon_tv1*/
        clocks = <&clk_de>,
                 <&clk_display_top>,
                 <&clk_tcon_lcd>,
                 <&clk_tcon_lcd1>,
                 <&clk_tcon_tv>,
                 <&clk_tcon_tv1>,
                 <&clk_lvds>;
        boot_disp = <0>;
        fb_base = <0>;
        iommus = <&mmu_aw 0 0>;
        status = "okay";
};
```

图 2: devicetree DE 配置

- tcon

```
tv0: tv0@01c94000 {
        compatible = "allwinner,sunxi-tv";
        reg = <0x0 0x06520000 0x0 0x100>,
              <0x0 0x06524000 0x0 0x3fc>;
        clocks = <&clk_tve_top>,<&clk_tve>;
        device_type = "tv0";
        pinctrl-names = "active","sleep";
        status = "disabled";
};

lcd0: lcd0@01c0c000 {
        compatible = "allwinner,sunxi-lcd0";
        pinctrl-names = "active","sleep";

        status = "okay";
};

lcd1: lcd1@01c0c001 {
        compatible = "allwinner,sunxi-lcd1";
        pinctrl-names = "active","sleep";

        status = "okay";
};

boot_disp: boot_disp {
        compatible = "allwinner,boot_disp";
};

hdmi: hdmi@06000000 {
        compatible = "allwinner,sunxi-hdmi";
        reg = <0x0 0x06000000 0x0 0x100000>;
        interrupts = <GIC_SPI 63 IRQ_TYPE_NONE>;
        clocks = <&clk_hdmi>,<&clk_hdmi_slow>,<&clk_hdmi_hdcp>,<&clk_hdmi_cec>;

        status = "okay";
};
```

图 3: devicetree TCON 配置

2.3.2 board.dts 配置说明

路径是 device/config/chips/a100/configs/b1/board.dts

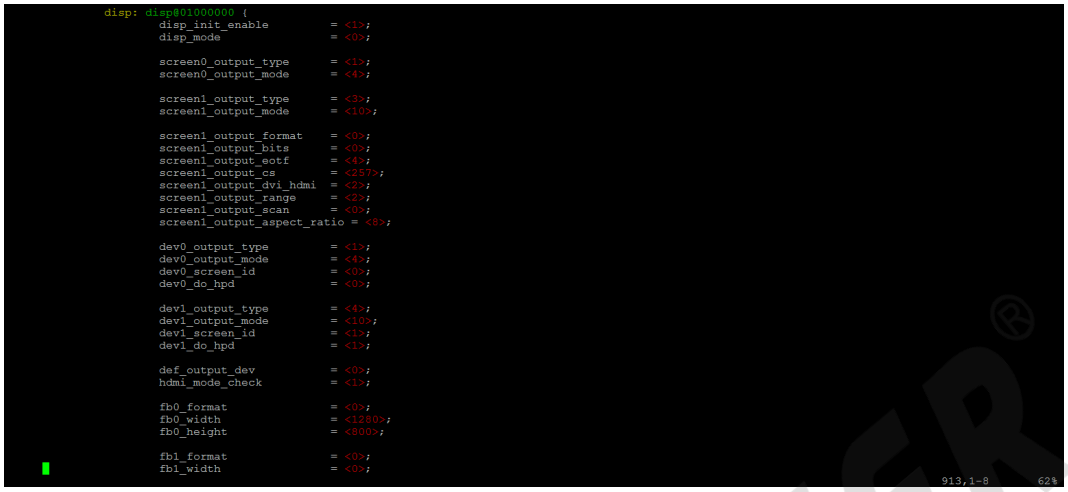

图 4: boarddts 配置

配置如上，参数的意义，请仔细阅读此节点上面的注释。

2.3.3 menuconfig 配置说明

在命令行中进入内核根目录，执行 make ARCH=arm64 menuconfig 进入配置主界面。注意: 对于 32 位 cpu 是 make ARCH=arm menuconfig。

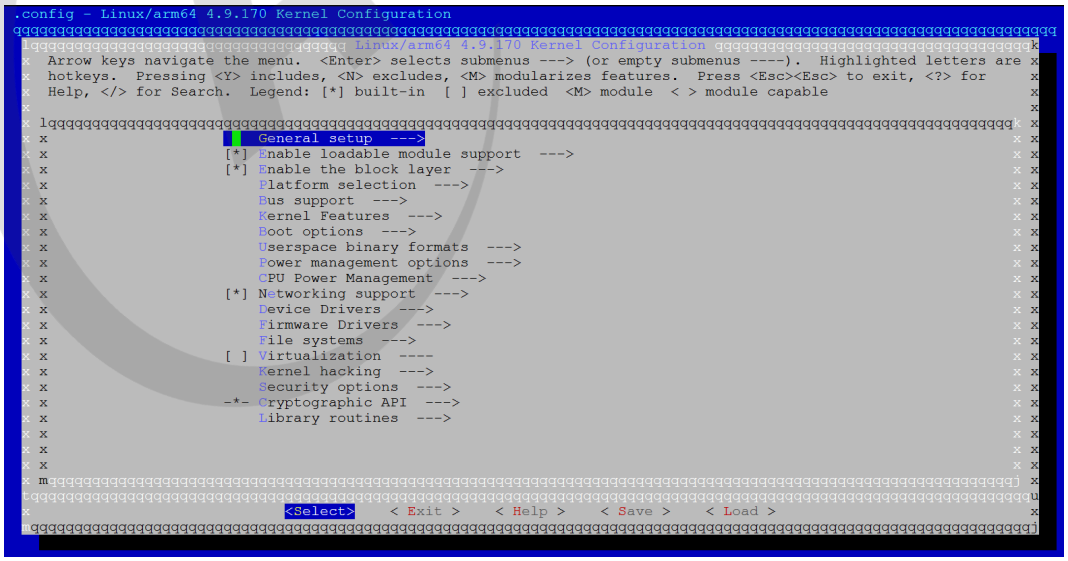

图 5: menuconfig 配置

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

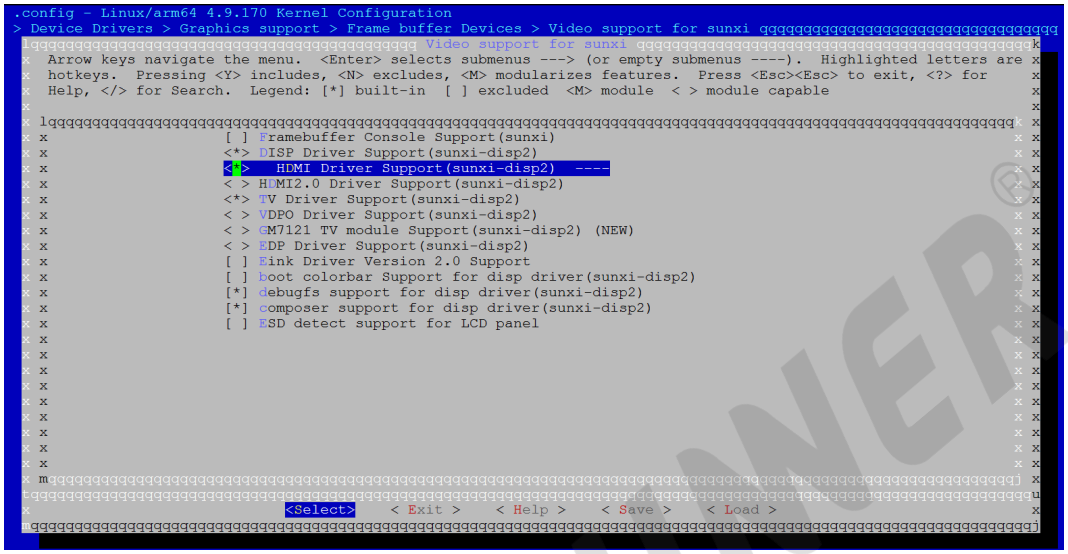

图 6: disp2 配置

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

linux 4.9 以后源码结构如下：

```
├─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
```

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发文件上面的注释：kernel/linux-4.9/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:

```
truct 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。

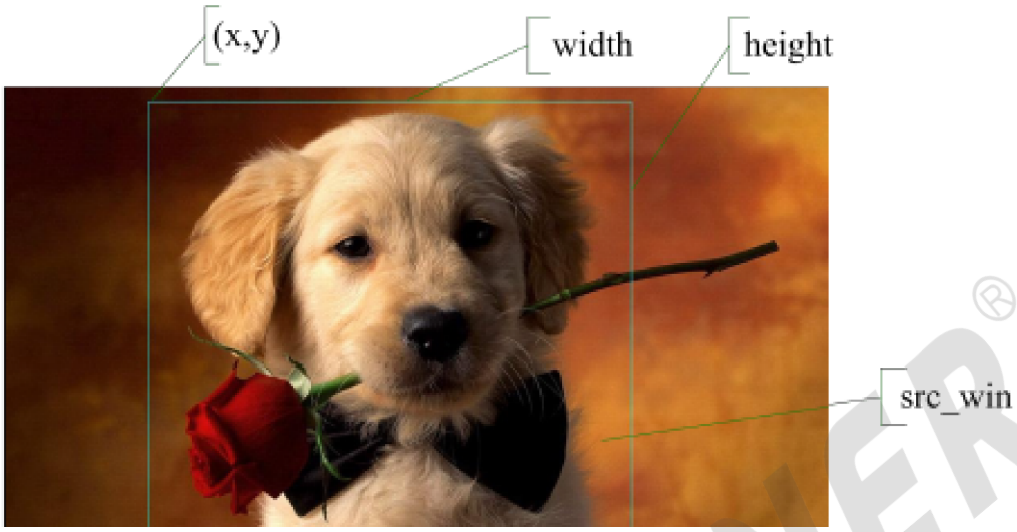

图 7: size 和 crop 示意图

- 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,height 是相等的，如果需要缩放，crop 和 screen_win 的 width,height 可以不相等。

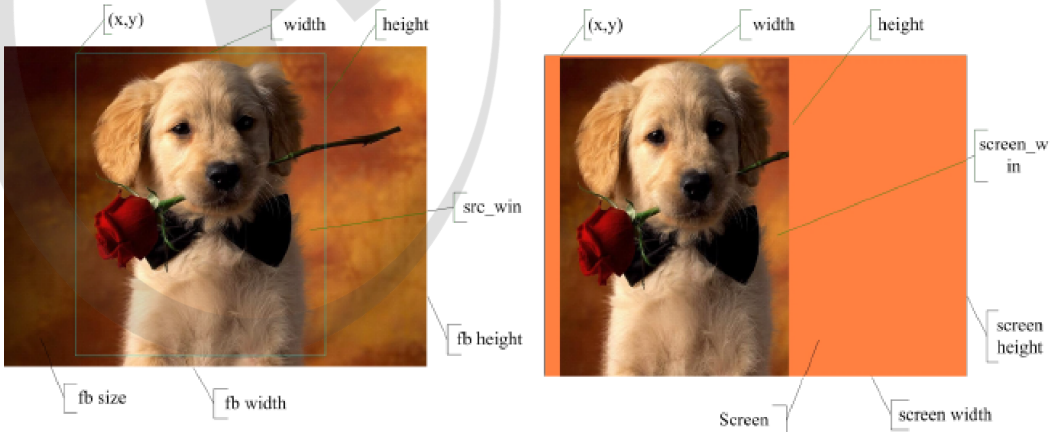

图 8: crop 和 screen win 示意图

- 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 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、显示输出接口等显示资源进行操作。

3.2.1 Global Interface

3.2.1.1 DISP_SHADOW_PROTECT

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
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
```

- RETURNS

```
如果成功，返回DIS_SUCCESS，否则，返回失败号；
```

- DESCRIPTION

```
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_SHADOV
```

- DEMO

```
//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);
```

3.2.1.2 DISP_SET_BKCOLOR

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
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；
- RETURNS
如果成功，返回DIS_SUCCESS，否则，返回失败号；
```

- DESCRIPTION

```
该函数用于设置显示背景色。
```

- DEMO

```
//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 int)&bk;
ioctl(dispfd, DISP_SET_BKCOLOR, (void*)arg);
```

3.2.1.3 DISP_GET_BKCOLOR

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
handle：显示驱动句柄；
cmd：DISP_GET_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；
```

- RETURNS

```
如果成功，返回DIS_SUCCESS，否则，返回失败号；
```

- DESCRIPTION

该函数用于获取显示背景色。

- DEMO

```
//获取显示背景色，dispfd 为显示驱动句柄，sel 为屏0/1
disp_color bk;
unsigned long arg[3];
arg[0] = 0;
arg[1] = (unsigned int)&bk;
ioctl(dispfd, DISP_GET_BKCOLOR, (void*)arg);
```

3.2.1.4 DISP_GET_SCN_WIDTH

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
handle：显示驱动句柄；
cmd：DISP_GET_SCN_WIDTH；
arg：显示通道0/1；
```

- RETURNS

如果成功，返回当前屏幕水平分辨率，否则，返回失败号；

- DESCRIPTION

该函数用于获取当前屏幕水平分辨率。

- DEMO

```
//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);
```

3.2.1.5 DISP_GET_SCN_HEIGHT

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
handle：显示驱动句柄；
cmd：DISP_GET_SCN_HEIGHT；
arg：arg[0]为显示通道0/1；
```

- RETURNS

```
如果成功，返回当前屏幕垂直分辨率，否则，返回失败号；
```

- DESCRIPTION 该函数用于获取当前屏幕垂直分辨率。
  - DEMO

```
//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);
```

3.2.1.6 DISP_GET_OUTPUT_TYPE

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

handle：显示驱动句柄；
cmd：DISP_GET_OUTPUT_TYPE；
arg：arg[0]为显示通道0/1；

- RETURNS

如果成功，返回当前显示输出类型，否则，返回失败号；

- DESCRIPTION

该函数用于获取当前显示输出类型(LCD,TV,HDMI,VGA,NONE)。

- DEMO

```
//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);
```

3.2.1.7 DISP_GET_OUTPUT

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
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
```

- RETURNS

```
如果成功，返回0，否则，返回失败号；
```

- DESCRIPTION 该函数用于获取当前显示输出类型及模式 (LCD,TV,HDMI,VGA,NONE)。
  - DEMO

```
//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;
```

3.2.1.8 DISP_VSYNC_EVENT_EN

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
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
```

- RETURNS

```
如果成功，返回DIS_SUCCESS，否则，返回失败号；
```

- DESCRIPTION

```
该函数开启/关闭vsync 消息发送功能。
```

- DEMO

```
//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);
```

3.2.1.9 DISP_DEVICE_SWITCH

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
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
```

- RETURNS

```
如果成功，返回DIS_SUCCESS，否则，返回失败号；
```

- DESCRIPTION

```
该函数用于切换输出类型
```

- DEMO

```
//切换
unsigned long arg[3];
arg[0] = 0;
arg[1] = (unsigned long)DISP_OUTPUT_TYPE_LCD;
arg[2] = 0;
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。
```

3.2.1.10 DISP_DEVICE_SET_CONFIG

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
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
```

- RETURNS

```
如果成功，返回DIS_SUCCESS，否则，返回失败号；
```

- DESCRIPTION

```
该函数用于切换输出类型并设置输出设备的属性参数
```

- DEMO

```
//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_LCD;
config.mode = 0;
config.format = DISP_CSC_TYPE_RGB;
config.bits = DISP_DATA_8BITS;
```

```
config.eotf = DISP_EOTF_GAMMA22;
config.cs = DISP_BT709;
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。
```

3.2.1.11 DISP_DEVICE_GET_CONFIG

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
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
```

- RETURNS

```
如果成功，返回DIS_SUCCESS，否则，返回失败号；
```

- DESCRIPTION

该函数用于获取当前输出类型及相关的属性参数

- DEMO

```
//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。
```

3.2.2 Layer Interface

3.2.2.1 DISP_LAYER_SET_CONFIG

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
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
```

- RETURNS

如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。

- DESCRIPTION

该函数用于设置多个图层信息。

- DEMO

```
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);
```

3.2.2.2 DISP_LAYER_GET_CONFIG

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
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；
```

- RETURNS

```
如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。
```

- DESCRIPTION

该函数用于获取图层参数。

- DEMO

```
//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);
```

3.2.2.3 DISP_LAYER_SET_CONFIG2

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

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

- RETURNS

如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。

- DESCRIPTION

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。

- DEMO

```
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);
```

3.2.2.4 DISP_LAYER_GET_CONFIG2

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
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；
```

- RETURNS

```
如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。
```

- DESCRIPTION

该函数用于获取图层参数。

- DEMO

```
//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);
```

3.2.3 capture interface

3.2.3.1 DISP_CAPTURE_START

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
handle：显示驱动句柄；
cmd：DISP_CAPTURE_START
arg：arg[0]为显示通道0/1；
```

- RETURNS

如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。

- DESCRIPTION

该函数启动截屏功能。

- DEMO

```
//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);
```

3.2.3.2 DISP_CAPTURE_COMMIT

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

handle：显示驱动句柄；
cmd：DISP_CAPTURE_COMMIT
arg：arg[0]为显示通道0/1；

- RETURNS

如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。

- DESCRIPTION

该函数提交截屏信息，提交后才走在启动截屏功能。

- DEMO

```
//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);
```

3.2.3.3 DISP_CAPTURE_STOP

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
handle：显示驱动句柄；
cmd：DISP_CAPTURE_STOP
arg：arg[0]为显示通道0/1；
```

- RETURNS

```
如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。
```

- DESCRIPTION

```
该函数停止截屏功能。
```

- DEMO

```
//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);
```

3.2.3.4 DISP_CAPTURE_QUERY

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS.

```
handle：显示驱动句柄；
cmd：DISP_CAPTURE_QUERY
arg：arg[0]为显示通道0/1；
```

- RETURNS

```
如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。
```

- DESCRIPTION

```
该函数查询刚结束的图像帧是否截屏成功。
```

- DEMO

```
//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);
```

3.2.4 LCD Interface

3.2.4.1 DISP_LCD_SET_BRIGHTNESS

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
handle：显示驱动句柄；
cmd：DISP_LCD_SET_BRIGHTNESS
arg：arg[0]为显示通道0/1；arg[1]为背光亮度值，（0~255）
```

- RETURNS

```
如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。
```

- DESCRIPTION

```
该函数用于设置LCD 亮度。
```

- DEMO

```
//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;//显示通道0
arg[1] = bl;
ioctl(dispfd, DISP_LCD_SET_BRIGHTNESS, (void*)arg);
```

3.2.4.2 DISP_LCD_GET_BRIGHTNESS

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
handle 显示驱动句柄；
cmd DISP_LCD_GET_BRIGHTNESS
arg arg[0]为显示通道0/1；
```

- RETURNS

```
如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。
```

- DESCRIPTION

```
该函数用于获取LCD 亮度。
```

- DEMO

```
//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);
```

3.2.4.3 DISP_LCD_SET_GAMMA_TABLE

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
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。
```

- RETURNS

```
如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。
```

- DESCRIPTION 该函数用于设置 lcd 的 gamma table。
  - DEMO

```
//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”);
```

3.2.4.4 DISP_LCD_GAMMA_CORRECTION_ENABLE

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
handle 显示驱动句柄；
cmd DISP_LCD_GAMMA_CORRECTION_ENABLE
arg arg[0]为显示通道0/1；
```

- RETURNS

```
如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。
```

- DESCRIPTION

```
该函数用于使能lcd 的gamma 校正功能。
```

- DEMO

```
//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”);
```

3.2.4.5 DISP_LCD_GAMMA_CORRECTION_DISABLE

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
handle 显示驱动句柄；
cmd DISP_LCD_GAMMA_CORRECTION_DISABLE
arg arg[0]为显示通道0/1；
```

- RETURNS

```
如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。
```

- DESCRIPTION

```
该函数用于关闭lcd 的gamma 校正功能。
```

- DEMO

```
//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”);
```

3.2.5 smart backlight

3.2.5.1 DISP_SMBL_ENABLE

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
handle 显示驱动句柄；
cmd DISP_SMBL_ENABLE
arg arg[0]为显示通道0/1；
```

- RETURNS

```
如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。
```

- DESCRIPTION

```
该函数用于使能智能背光功能。
```

- DEMO

```
//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);
```

3.2.5.2 DISP_SMBL_DISABLE

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
handle 显示驱动句柄；
cmd DISP_SMBL_DISABLE
arg arg[0]为显示通道0/1；
```

- RETURNS

```
如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。
```

- DESCRIPTION

```
该函数用于关闭智能背光功能。
```

- DEMO

```
//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);
```

3.2.5.3 DISP_SMBL_SET_WINDOW

- PROTOTYPE

```
int ioctl(int handle, unsigned int cmd,unsigned int *arg);
```

- ARGUMENTS

```
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
- RETURNS
如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。
```

- DESCRIPTION

```
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。
```

- DEMO

```
//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);
```

3.2.6 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;
}
```

3.2.7 enhance

3.2.7.1 enhance_mode

- 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

```
no
```

- 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:
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));
```

3.2.7.2 enhance_bright/contrast/saturation/edge/detail/denoise

- SYSFS NODE

```
/sys/class/disp/disp/attr/disp
/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：0~100，数据越大，调节幅度越大。
```

- RETURNS

no

- 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。

- DEMO

```
//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;
c/c++代码:
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));
```

3.2.8 Data Structure

3.2.8.1 disp_fb_info

- PROTOTYPE

```
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;
```

- MEMBERS

```
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],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
b_trd_src: 1:3D source; 0：2D source
trd_mode :source 3D mode, 详见disp_3d_src_mode
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
```

- DESCRIPTION

```
disp_fb_info 用于描述一个display frambuffer 的属性信息。
```

3.2.8.2 disp_layer_info

- 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;
```

- MEMBERS

```
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
```

- DESCRIPTION

```
disp_layer_info 用于描述一个图层的属性信息。
```

3.2.8.3 disp_layer_config

- PROTOTYPE

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

- MEMBERS

```
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。
```

- DESCRIPTION

```
disp_layer_config 用于描述一个图层配置的属性信息。
```

3.2.8.4 disp_layer_config2

- 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;
};
```

- MEMBERS

```
这里仅介绍相对disp_layer_config 新增的成员
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
```

- DESCRIPTION

disp_layer_config2 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。

3.2.8.5 disp_color_info

- PROTOTYPE

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

- MEMBERS

alpha :颜色的透明度
red :红
green :绿
blue :蓝

- DESCRIPTION

disp_color_info 用于描述一个颜色的信息。

3.2.8.6 disp_rect

- PROTOTYPE

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

- MEMBERS

```
x :起点x 值
y :起点y 值
width :宽
height :高
```

- DESCRIPTION

```
disp_rect 用于描述一个矩形窗口的信息。
```

3.2.8.7 disp_rect64

- PROTOTYPE

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

- MEMBERS

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

- DESCRIPTION

disp_rect64 用于描述一个矩形窗口的信息。

3.2.8.8 disp_position

- PROTOTYPE

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

- MEMBERS

x :x
y :y

- DESCRIPTION

disp_position 用于描述一个坐标的信息。

3.2.8.9 disp_rectsz

- PROTOTYPE

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

- MEMBERS

width : 宽
height : 高

- DESCRIPTION

disp_rectsz 用于描述一个矩形尺寸的信息。

3.2.8.10 disp_atw_info

- PROTOTYPE

```
/\* 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;
};
```

- MEMBERS

```
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
```

- DESCRIPTION

disp_atw_info 用于描述图层的asynchronous time wrap(异步时移)信息。

3.2.8.11 disp_pixel_format

- PROTOTYPE

```
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;;
```

- MEMBERS

```
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
```

- DESCRIPTION

```
disp_pixel_format 用于描述像素格式。
```

3.2.8.12 disp_data_bits

- PROTOTYPE

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

- DESCRIPTION disp_data_bits 用于描述图像的数据宽度。

3.2.8.13 disp_eotf

- PROTOTYPE

```
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 */
};
```

- DESCRIPTION

disp_eotf 用于描述图像的光电转换特性。

3.2.8.14 disp_buffer_flags

- PROTOTYPE

```
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;
```

- MEMBERS

```
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
```

- DESCRIPTION

disp_buffer_flags 用于描述3D 源模式。

3.2.8.15 disp_3d_out_mode

- PROTOTYPE

```
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
}disp_3d_out_mode;
```

- MEMBERS

```
//for lcd
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
```

- DESCRIPTION

```
disp_3d_out_mode 用于描述3D 输出模式。
```

3.2.8.16 disp_color_space

- PROTOTYPE

```
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,
};
```

- MEMBERS

```
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
```

- DESCRIPTION

```
disp_color_space 用于描述颜色空间类型。
```

3.2.8.17 disp_csc_type

- PROTOTYPE

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

- DESCRIPTION

disp_csc_type 用于描述图像颜色格式。

3.2.8.18 disp_output_type

- PROTOTYPE

```
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;
```

- MEMBERS

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

- DESCRIPTION

disp_output_type 用于描述显示输出类型。

3.2.8.19 disp_tv_mode

- PROTOTYPE

```
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;
```

- DESCRIPTION

disp_tv_mode 用于描述TV 输出模式。

3.2.8.20 disp_output

- PROTOTYPE

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

- MEMBERS

Type:输出类型
Mode:输出模式，480P/576P, etc.

- DESCRIPTION

disp_output 用于描述显示输出类型，模式

3.2.8.21 disp_layer_mode

- PROTOTYPE

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

- MEMBERS

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

- DESCRIPTION

disp_layer_mode 用于描述图层模式。

3.2.8.22 disp_scan_flags

- PROTOTYPE

```
enum disp_scan_flags
{
   DISP_SCAN_PROGRESSIVE = 0,//non interlace
   DISP_SCAN_INTERLACED_ODD_FLD_FIRST = 1 << 0,//interlace ,odd field first
   DISP_SCAN_INTERLACED_EVEN_FLD_FIRST = 1 << 1,//interlace,even field first
};
```

- MEMBERS

```
DISP_SCAN_PROGRESSIVE: 逐行模式
DISP_SCAN_INTERLACED_ODD_FLD_FIRST : 隔行模式，奇数行优先
DISP_SCAN_INTERLACED_EVEN_FLD_FIRST : 隔行模式，偶数行优先
```

- DESCRIPTION

```
disp_scan_flags 用于描述显示Buffer 的扫描方式。
```

3.2.8.23 disp_device_config

- PROTOTYPE

```
/\* 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;
};
```

- MEMBERS

```
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
```

- DESCRIPTION

```
disp_device_config 用于描述输出设备的属性信息。
```

3.2.8.24 disp_device_config

- PROTOTYPE

```
/\* 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;
};
```

- MEMBERS

```
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: 光电转换特性信息
cs: 输出的颜色空间类型
```

- DESCRIPTION

disp_device_config 用于描述输出设备的属性信息。

4. 调试

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:
de_rate 432000000 Hz /* de 的时钟频率*/, ref_fps=50 /* 输出设备的参考刷新率*/
hdmi output mode(4) fps:50.5 1280x 720
err:0 skip:54 irq:21494 vsync:0
BUF enable ch[0] lyr[0] z[0] prem[N] a[globl 255] fmt[ 1] fb[1920,1080;1920,1080;1920,1080] crop[ 0, 0,1920,1080] frame[ 32, 18,1216, 684]
addr[716da000, 0, 0] flags[0x 0] trd[0,0]
screen 1:
de_rate 432000000 Hz /* de 的时钟频率*/, ref_fps=50 /* 输出设备的参考刷新率*/
tv output mode(11) fps:50.5 720x 576 /* TV 输出| 模式为（11：PAL） | 刷新率为：50.5Hz | 分辨率为：720x576 */
err:0 skip:54 irq:8372 vsync:0
BUF enable ch[0] lyr[0] z[0] prem[Y] a[globl 255] fmt[ 0] fb[ 720, 576; 720, 576; 720, 576] crop[ 0, 0, 720, 576] frame[ 18, 15, 684, 546]
addr[739a8000, 0, 0] flags[0x 0] trd[0,0]
acquire: 225, 2.6 fps
release: 224, 2.6 fps
display: 201, 2.5 fps
```

图层各信息描述如下：

```
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：
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。

acquire/release/display 含义如下,只在android 方案中有效：
acquire： 是hw composer 传递给disp driver 的图像帧数以及帧率，帧率只要有在有图像更新时才有效，静止时的值是不准确的
release: 是disp driver 显示完成之后，返还给android 的图像帧数以及帧率，帧率只要有在有图像更新时才有效，静止时的值是不准确的
display： 是disp 显示到输出设备上的帧数以及帧率，帧率只要有在有图像更新时才有效，静止时的值是不准确的如果acquire 与release
不一致，说明disp 有部分图像帧仍在使用，未返还，差值在1~2
之间为正常值。二者不能相等，如果相等，说明图像帧全部返还，显示将会出
现撕裂现象。如果display 与release 不一致，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  > /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。1，DE 输出；2-7，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;
/* mount debugfs */
# mount -t debugfs none /sys/kernek/debug;
/* 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
 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
```

4.5 HWC 调试

系统内集成了调试工具 hwcdebug，通过该工具执行调试；注意：如果 hwcdebug 提示 can't get IDisplayConfig service，需要先执行 su

```
$ hwcdebug --help
usage: hwcdebug [options] [categories...]
options include:
  -o      enable hwc log output which indicate by the
          following categories
  -x      disbale all hwc log output
  --list  list the available logging categories
  --dump <type> <count> [z]
          request dump buffer raw data into file
          type: 0 - input  buffer
                1 - output buffer
                2 - hardware rotated buffer
          count: total request count
          z : which z order of input buffer will be dump
  --freeze <enable> -s <display>
          freezes the output content of the specified logic display for debugging
  --help  show this usage
```

4.5.1 查看 android 帧率

```
# hwcdebug -o fps && logcat -s sunxihwc

结果示例如下：
```

```
02-28 11:33:55.939  1820  1968 I sunxihwc: onFramePresent: freezed 0, refresh rate 61.00
02-28 11:33:55.955  1820  1968 I sunxihwc: onFramePresent: freezed 0, refresh rate 60.50
02-28 11:33:55.972  1820  1968 I sunxihwc: onFramePresent: freezed 0, refresh rate 60.61
02-28 11:33:55.988  1820  1968 I sunxihwc: onFramePresent: freezed 0, refresh rate 60.60
```

4.5.2 查看 android 图层信息

```
# hwcdebug -o layer && logcat -s sunxihwc
```

结果示例如下：

```
Composition|Channel|Slot|Z-hw|SF-Zorder|Alpha| BlendingMode|Transform| Dataspace|  Handle |     sourceCrop     |     displayFrame     | scale |
-----------+-------+----+----+---------+-----+-------------+---------+----------+----------+--------------------+--------------------+---------+
    Device|   0   |  0 |  0 |       0|1.000|         None|     None|0x00000000|0xea528860|  159,   0, 1159, 1600|   0,   0,  800, 1280|0.80 0.80| hardware layer
    Device|   1   |  0 |  1 |       1|1.000|Premultiplied|     None|0x00000000|0xea528fe0|    0,   0,  800, 1280|   0,   0,  800, 1280|1.00 1.00| hardware layer
    Device|   2   |  0 |  2 |       2|1.000|Premultiplied|     None|0x00000000|0xea527f00|    0,   0,   24, 1280|   0,   0,   24, 1280|1.00 1.00| hardware layer
    Device|   2   |  1 |  3 |       3|1.000|Premultiplied|     None|0x00000000|0xea5282c0|    0,   0,   48, 1280|  752,   0,  800, 1280|1.00 1.00| hardware layer
    Device|  -1   | -1 | -1 |     255|1.000|Premultiplied|     None|0x00000000|0xea5275a0|    0,   0,  800, 1280|   0,   0,  800, 1280|0.00 0.00| client target
```

各字段表示的意义如下：

字段	意义
Composition	合成类型，Device=DE 合成，Client=GPU 合成
Channel	使用的 blending channel 索引号
Slot	channel 内 overlay 索引号
Z-hw	图层在 DE 内的 zorder
SF-Zorder	SurfaceFlinger 设置的 zorder
Alpha	plane alpha，面 alpha
BlendingMode	blending 模式
Transform	图层旋转角度
Dataspace	图层 Dataspace
Handle	当前图层的 buffer handle
sourceCrop	图像源的裁减区，只有裁减区内的内容会显示在屏幕上，

字段	意义
displayFrame	图像源的裁减区显示在屏幕上的矩形窗口，x_left, y_top, x_right, y_bottom
scale	图层的缩放系数

不能使用 DE 合成的原因:

```
"hardware layer"          : DE 硬件合成
"force gpu composition"   : 强制 GPU 合成
"no free channel"         : 没有空闲的 DE pipe
"scale error"             : Scaler 能力不足
"bandwidth limit"         : 带宽不足
"cover by client target"  : 与 Framebuffer Target 相交
"bufferhandle is null"    : buffer handle is null
"skip by surfaceflinger"  : SurfaceFlinger 标识不走 DE 硬件合成
"client target"           : FrameBuffer Target
"format not supported"    : 不支持的图层格式
"AFBC not supported"      : 不支持 AFBC 压缩格式
"non physical continuous memory" : 非物理连续内存
"transform not supported" : 硬件不支持该图层旋转
```

4.6 常见问题

4.6.1 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，确定是 PWM 问题。否则，尝试换个屏再试。

- 步骤二

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。

- 步骤三

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。

4.6.2 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 4.1 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 4.2 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 4.3 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。

4.6.3 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 4.1 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。

4.6.4 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 4.1 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。

4.6.5 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 4.1 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。

4.6.6 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. Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner. The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This document neither states nor implies warranty of any kind, including fitness for any particular application.tates nor implies warranty of any kind, including fitness for any particular application.