



# **RTOS Display**

## **开发指南**

**版本号: 1.1**  
**发布日期: 2021.4.13**

## 版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.7.20	AWA1639	1. 添加初版
1.1	2021.4.13	AWA1639	1. 更新配置说明 2. 支持列表增加 f133



## 目 录

<b>1 前言</b>	<b>1</b>
1.1 文档简介	1
1.2 目标读者	1
1.3 适用范围	1
<b>2 模块介绍</b>	<b>2</b>
2.1 模块功能介绍	2
2.2 相关术语介绍	2
2.2.1 硬件术语	2
2.2.2 软件术语	3
2.3 模块配置介绍	3
2.3.1 板级显示配置说明	3
2.3.1.1 使用显示私有方式进行配置	3
2.3.1.2 使用 sys_config.fex 的方式进行配置	5
2.3.2 SOC 平台配置说明	8
2.4 源码结构介绍	10
2.5 驱动框架介绍	11
<b>3 模块接口说明</b>	<b>12</b>
3.1 模块接口概述	12
3.2 模块使用接口说明	13
3.3 Global Interface	13
3.3.1 DISP_SHADOW_PROTECT	13
3.3.2 DISP_SET_BKCOLOR	14
3.3.3 DISP_GET_BKCOLOR	14
3.3.4 DISP_GET_SCN_WIDTH	15
3.3.5 DISP_GET_SCN_HEIGHT	16
3.3.6 DISP_GET_OUTPUT_TYPE	16
3.3.7 DISP_GET_OUTPUT	17
3.3.8 DISP_VSYNC_EVENT_EN	17
3.3.9 DISP_DEVICE_SWITCH	18
3.3.10 DISP_DEVICE_SET_CONFIG	18
3.3.11 DISP_DEVICE_GET_CONFIG	19
3.4 Layer Interface	19
3.4.1 DISP_LAYER_SET_CONFIG	19
3.4.2 DISP_LAYER_GET_CONFIG	21
3.4.3 DISP_LAYER_SET_CONFIG2	21
3.4.4 DISP_LAYER_GET_CONFIG2	22
3.5 capture interface	23
3.5.1 DISP_CAPTURE_START	23
3.5.2 DISP_CAPTURE_COMMIT	23
3.5.3 DISP_CAPTURE_STOP	24

3.5.4	DISP_CAPTURE_QUERY	25
3.6	LCD Interface	25
3.6.1	DISP_LCD_SET_BRIGHTNESS	25
3.6.2	DISP_LCD_GET_BRIGHTNESS	26
3.6.3	DISP_LCD_SET_GAMMA_TABLE	26
3.6.4	DISP_LCD_GAMMA_CORRECTION_ENABLE	27
3.6.5	DISP_LCD_GAMMA_CORRECTION_DISABLE	27
3.7	smart backlight	28
3.7.1	DISP_SMBL_ENABLE	28
3.7.2	DISP_SMBL_DISABLE	28
3.7.3	DISP_SMBL_SET_WINDOW	29
3.8	Data Structure	30
3.8.1	disp_fb_info	30
3.8.2	disp_layer_info	30
3.8.3	disp_layer_config	31
3.8.4	disp_layer_config2	32
3.8.5	disp_color_info	34
3.8.6	disp_rect	34
3.8.7	disp_rect64	35
3.8.8	disp_position	35
3.8.9	disp_rectsz	35
3.8.10	disp_atw_info	36
3.8.11	disp_pixel_format	37
3.8.12	disp_data_bits	38
3.8.13	disp_eotf	38
3.8.14	disp_buffer_flags	39
3.8.15	disp_3d_out_mode	39
3.8.16	disp_color_space	40
3.8.17	disp_csc_type	41
3.8.18	disp_output_type	41
3.8.19	disp_tv_mode	42
3.8.20	disp_output	42
3.8.21	disp_layer_mode	43
3.8.22	disp_device_config	43
<b>4</b>	<b>FAQ</b>	<b>45</b>
4.1	调试命令	45
4.2	模块使用范例	46
4.3	常见问题	46
4.3.1	黑屏（无背光）	46
4.3.2	黑屏（有背光）	47
4.3.3	绿屏	48
4.3.4	界面卡住	48

4.3.5 局部界面花屏 . . . . .	48
------------------------	----



## 插 图

2-1 模块框图 . . . . .	2
2-2 配置文件编译举例 . . . . .	4
2-3 板级配置文件说明 . . . . .	4
2-4 sys_config.fex display 部分举例 . . . . .	7
2-5 SOC 级配置文件说明 . . . . .	9
2-6 驱动框图 . . . . .	11



# 1 前言

## 1.1 文档简介

介绍 Sunxi 平台 RTOS 上 Display 驱动 hal 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: 适用产品列表

产品名称	内核版本	驱动文件
V833	Melis	rtos-hal/hal/source/disp2/
F133	Melis	rtos-hal/hal/source/disp2/

## 2 模块介绍

### 2.1 模块功能介绍

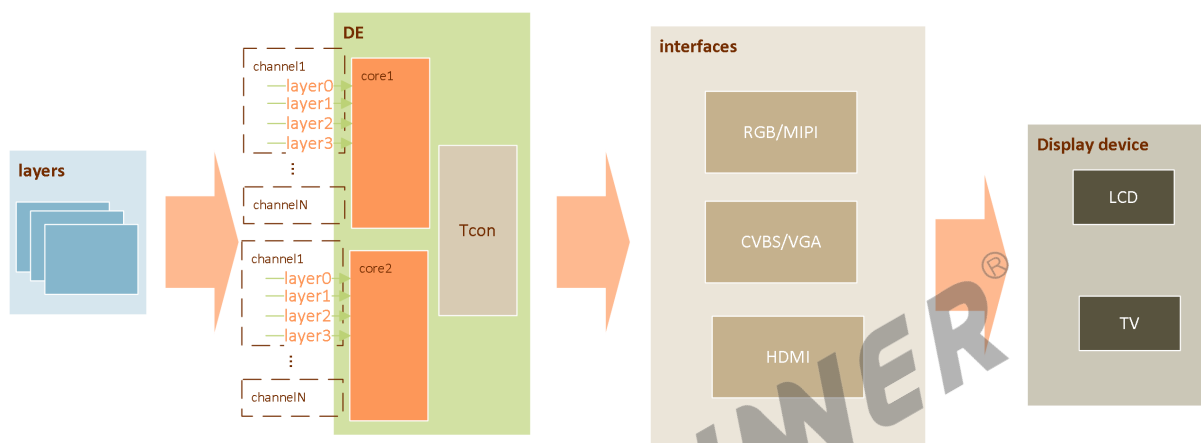


图 2-1: 模块框图

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。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 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）
- 支持多种图像数据格式输入（argb,yuv）
- 支持图像缩放处理

### 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
alpha	透明度, 在混合时决定对应图像的透明度
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

## 2.2.2 软件术语

表 2-2: 软件术语

术语	解释说明
HAL	Hardware Abstraction Layer, 硬件抽象层
RTOS	Real Time Operating System, 实时操作系统
GPIO	General Purpose Input/Output, 通用输入输出

## 2.3 模块配置介绍

### 2.3.1 板级显示配置说明

当前板级显示支持两种配置方法, 一是使用 sys\_config.fex 的方式进行配置, 二是通过显示驱动私有的方式进行配置, 目前 f133 使用的是方式 1, v459 使用的是方式 2。下面分别对两种方式进行说明。

#### 2.3.1.1 使用显示私有方式进行配置

路径: rtos-hal/hal/source/disp2/soc/ 编译配置选项可参考下图中的 v459\_perfl.c 所在行。

```

1 obj-y += platform_resource.o disp_board_config.o
1 obj-$(CONFIG_ARCH_SUN8IW19) += sun8iw19.o
2 obj-$(CONFIG_V459_PERF1) += v459_perf1.o
3

```

图 2-2: 配置文件编译举例

该文件用于模仿 linux 中的 board.dts 的配置，用于配置一些板级相关的资源，具体可参考下图，源文件中需要定义 6 个全局变量：g\_lcd0\_config、g\_lcd1\_config、g\_disp\_config、g\_lcd0\_config\_len、g\_lcd1\_config\_len 和 g\_disp\_config\_len，且变量名字固定，不能修改。说明如下：

- 1.g\_lcd0\_config：第一个屏的配置变量。
- 2.g\_lcd1\_config：第二个屏的配置变量。
- 3.g\_disp\_config：显示配置变量。决定加载驱动的时候显示类型和显示分辨率等。
- 4.g\_lcd0\_config\_len，g\_lcd1\_config\_len 和 g\_disp\_config\_len 是对应上面三个数组变量的长度，照搬即可。

```

+ struct property_t g_lcd0_config[] = {
+ struct property_t g_lcd1_config[] = {
+ struct property_t g_disp_config[] = {

u32 g_lcd0_config_len = sizeof(g_lcd0_config) / sizeof(struct property_t);
u32 g_lcd1_config_len = sizeof(g_lcd1_config) / sizeof(struct property_t);
u32 g_disp_config_len = sizeof(g_disp_config) / sizeof(struct property_t);

```

图 2-3: 板级配置文件说明

struct property\_t 数据类型，用于定义一个属性的信息：

1. 属性名字。对应其成员 name，字符串
2. 属性类型。对应其成员 type，看 enum proerty\_type 的定义，共有整型，字符串，GPIO，专用 pin 和电源。
3. 属性值。根据上面的属性类型来选择 union 中成员来赋值。

对于上述 5 种属性类型举例如下：整型：

```

{
    .name = "lcd_used",
    .type = PROPERTY_INTEGER,
    .v.value = 1,
},

```

字符串：

```
{
    .name = "lcd_driver_name",
    .type = PROPERTY_STRING,
    .v.str = "st7701s",
},
```

电源：

```
{
    .name = "lcd_power",
    .type = PROPERTY_POWER,
    .v.power = {
        .power_name = "dldo1",
        .power_type = AXP2101_REGULATOR,
        .power_id = AXP2101_ID_DLD01,
        .power_vol = 3300000,
    },
},
```

GPIO：

```
{
    .name = "lcd_gpio_0",
    .type = PROPERTY_GPIO,
    .v.gpio_list = {
        .gpio = GPIO0(9),
        .mul_sel = GPIO_DIRECTION_OUTPUT,
        .pull = 0,
        .drv_level = 3,
        .data = 1,
    },
},
```

专用 pin：GPIO 之外的功能（比如 RGB，LVDS）的管脚设置与上面的区别只有 type 要设置成 PROPERTY\_PIN。

而对于 g\_lcd0\_config 可以配置的属性以及各属性的可取值请参考《Sunxi\_display2 模块 LCD 调试文档》。若需要支持新的 LCD 屏，请参考《Sunxi\_display2 模块 LCD 调试文档.doc》，RTOS 驱动目录结构与 Linux 下的一致。

### 2.3.1.2 使用 sys\_config.fex 的方式进行配置

路 径：melis-v3.0/source/source/projects/{BOARD}/configs/sys\_config\_nor.fex(或 其他 fex，根据当前方案确定)，{BOARD} 为相应的板型名称，如 f133-prototype-machine。

```
[lcd0]
lcd_used                = 1

lcd_driver_name         = "V VX07H005A10"
lcd_backlight           = 150

lcd_if                  = 4

lcd_x                   = 800
lcd_y                   = 1280
lcd_width               = 800
lcd_height              = 1280

lcd_dclk_freq           = 80

lcd_pwm_used            = 1
lcd_pwm_ch              = 0
lcd_pwm_freq            = 50000
lcd_pwm_max_limit       = 255
lcd_pwm_pol             = 1

lcd_hbp                 = 48
lcd_ht                  = 864
lcd_hspw                = 8
lcd_vbp                 = 3
lcd_vt                  = 1388
lcd_vspw                = 2

lcd_frm                 = 0
lcd_io_phase            = 0
lcd_gamma_en            = 0
lcd_bright_curve_en     = 0
lcd_cmap_en             = 0

deu_mode                = 0
lcdgamma4iep            = 22
smart_color             = 90
lcd_dsi_if              = 0
lcd_dsi_lane            = 4
lcd_dsi_format          = 0
lcd_dsi_te              = 0

lcd_gpio_1              = port:PE11<1><0><3><1>
```

如图中所示，节点 lcd0 内为 lcd 相应部分的配置，其含义与前述的显示私有方式中的 g\_lcd0\_config 一致。gpio 引脚的配置的四个尖括号的含义分别对应前述的 mul\_sel、pull、drv\_level、data。

```
[disp]
disp_init_enable      = 1
disp_mode             = 0

screen0_output_type   = 1
screen0_output_mode   = 4

screen0_output_format = 1
screen0_output_bits   = 0
screen0_output_eotf   = 4
screen0_output_cs     = 257
screen0_output_dvi_hdmi = 2
screen0_output_range  = 2
screen0_output_scan   = 0
screen0_output_aspect_ratio = 8

screen1_output_type   = 1
screen1_output_mode   = 4
screen1_output_format = 1
screen1_output_bits   = 0
screen1_output_eotf   = 4
screen1_output_cs     = 260
screen1_output_dvi_hdmi = 2
screen1_output_range  = 2
screen1_output_scan   = 0
screen1_output_aspect_ratio = 8

fb0_format            = 0
fb0_width              = 0
fb0_height             = 0

fb1_format            = 0
fb1_width              = 0
fb1_height             = 0

lcd0_backlight        = 50
lcd1_backlight        = 50

lcd0_bright           = 50
lcd0_contrast          = 50
lcd0_saturation        = 57
lcd0_hue              = 50

lcd1_bright           = 50
lcd1_contrast          = 50
lcd1_saturation        = 57
lcd1_hue              = 50
```

图 2-4: sys\_config.fex display 部分举例

如图中所示，节点 disp 内为 display 相应部分的配置，其含义与前述的显示私有方式中的 g\_disp\_config 一致。

### 2.3.2 SOC 平台配置说明

路径：rtos-hal/hal/source/disp2/soc/ 编译配置选项可参考上一小节板级显示配置说明中的图“配置文件编译举例”中的 sun8iw19.c 所在行。该文件用于配置 SOC 平台相关的资源，具体可参考下图



```

u32 g_irq_no[] = {
    106, /*tcon-lcd0*/
    107, /*tcon-tv*/
    108 /*dsi*/
};

u32 g_reg_base[] = {
    0x05000000, /*de0*/
    0x05460000, /*disp_if_top*/
    0x05461000, /*tcon_lcd0*/
    0x05470000, /*tcon_tv*/
    0x05450000, /*dsi0*/
};

struct clk_info_t g_clk_no[] = {
    {
        "clk_de0",
        CLK_DE0,
        CLK_PLL_PERIPH0_2X,
        RST_BUS_DE0,
        NULL,
        NULL,
        NULL,
    },
    {
    },
    {
    },
    {
    },
    {
    },
    {
    },
    {
    },
    {
    },
    {
        "clk_bus_mipi_dsi0",
        CLK_BUS_MIPI_DSI,
        (hal_clk_id_t)-1,
        (hal_reset_id_t)-1,
        NULL,
        NULL,
        NULL,
    },
};

u32 g_irq_no_len = sizeof(g_irq_no) / sizeof(u32);
u32 g_reg_base_len = sizeof(g_reg_base) / sizeof(u32);
u32 g_clk_no_len = sizeof(g_clk_no) / sizeof(struct clk_info_t);

```

图 2-5: SOC 级配置文件说明

在该源文件中需要定义 6 个全局变量：g\_irq\_no, g\_reg\_base, g\_clk\_no、g\_irq\_no\_len, g\_reg\_base\_len 和 g\_clk\_no\_len, 且变量名字固定, 不能修改。各变量含义如下：

- g\_irq\_no. 显示模块的 irq 号, 顺序参考 linux 平台中 dts 配置的 disp 里面的顺序

- g\_reg\_base。显示模块的基地址。顺序参考 linux 平台中 dts 配置的 disp 里面的顺序
- g\_clk\_no。显示模块时钟信息。结构体的成员依次为时钟名，时钟 id，父时钟 id，配套的 reset id, 当不需要 reset 或者父时钟时，相应位置填入-1，最后三个保持为 NULL 即可。
- g\_irq\_no\_len, g\_reg\_base\_len 和 g\_clk\_no\_len 是上面对应变量数组的长度，照搬即可。

## 2.4 源码结构介绍

源码结构及主要驱动文件如下：

```
├─ disp/
│   ├── de/                --底层bsp代码
│   ├── dev_disp.c         --display driver 层
│   ├── disp_debug.c       --调试相关代码
│   ├── disp_sys_intf.c    --操作系统相关
│   └─ lcd/                --lcd驱动相关
├─ soc/
│   ├── disp_board_config.c --板级配置解析
│   ├── platform_resource.c --SOC平台配置解析
│   ├── sun8iw19.c          --板级配置文件
│   └─ v459_perfl.c         --SOC平台配置文件
```



## 2.5 驱动框架介绍

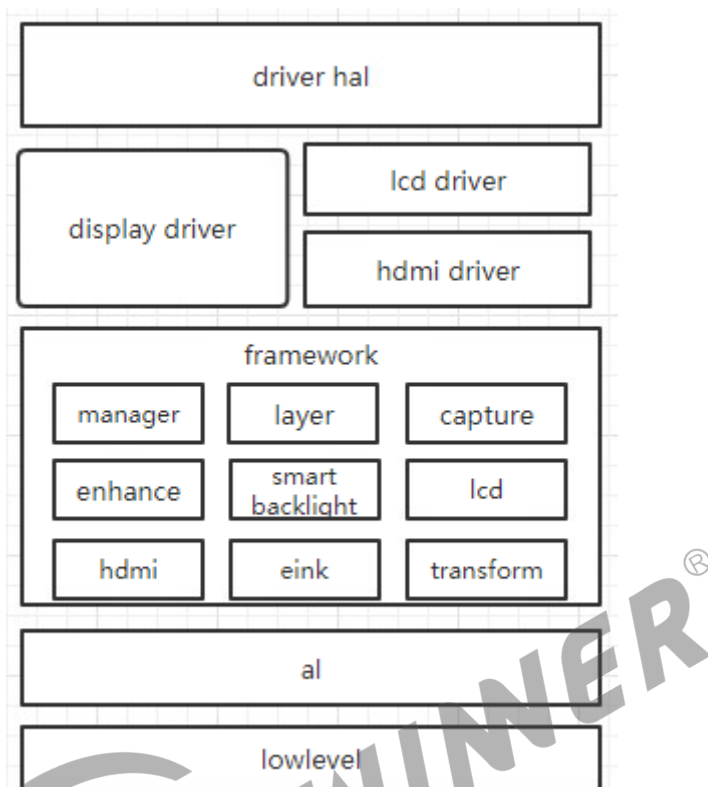


图 2-6: 驱动框图

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，通过固定的 rtos-hal 层接口向 OS 或其他 hal 驱动提供显示操作接口。当前驱动只将 lcd 相关的驱动移植完毕，eink、hdmi、tv 等代码虽存在于驱动中，但未移植验证，暂不支持使用。

## 3 模块接口说明

### 3.1 模块接口概述

disp 驱动向 OS 或其他 driver hal 暴露的接口如下表，模块使用主要通过 ioctl 实现，即 disp\_ioctl。

表 3-1: API 说明

API	解释说明
disp_probe	初始化 display 驱动
disp_ioctl	display 驱动 ioctl 接口
disp_release	关闭 display 驱动
disp_open	打开 display 驱动

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_layer_config {
    struct disp_layer_info info;           //图像的详细信息
    bool enable;                          //使能图层，如需要显示该图层，必须置为1
    unsigned int channel;                 //指定该图像在该channel显示
    unsigned int layer_id;               //指定该图像在该layer显示
};

struct disp_layer_info {
    enum disp_layer_mode mode;           //图像的类型，取值可以是LAYER_MODE_BUFFER或
    LAYER_MODE_COLOR, 前者表示使用颜色填充，后者表示使用相应地址作为图像输入
    unsigned char zorder;                //zorder，值越大，越靠近上层
    unsigned char alpha_mode;            //alpha blend的模式
    unsigned char alpha_value;           //全局alpha的值
    struct disp_rect screen_win;         //需要显示的大小，用于缩放，详见下
    bool b_trd_out;
    enum disp_3d_out_mode out_trd_mode;
    union {
        unsigned int color;             //当前述mode是LAYER_MODE_COLOR时有效，指定填充的颜色
        struct disp_fb_info fb;         //当前述mode是LAYER_MODE_COLOR时有效，描述buffer的
        详细信息
    };
    unsigned int id;
};

struct disp_fb_info {
```

```
unsigned long long addr[3];           //buffer的地址
struct disp_rectsz size[3];          //buffer的三个分量的大小
unsigned int align[3];               //buffer的三个分量的对齐值，单位byte
enum disp_pixel_format format;       //buffer的格式
enum disp_color_space color_space;
unsigned int trd_right_addr[3];
bool pre_multiply;
struct disp_rect64 crop;             //buffer的裁剪区域
enum disp_buffer_flags flags;
enum disp_scan_flags scan;
};
```

关于 addr、crop、size 与 screen\_win 的说明:

- addr 表示输入图像数据的物理地址，对于 interleaved 格式，只需要配置 addr[0]，对于 semi-planar 格式则需要使用 addr[0]、addr[1]，对于 planar format 格式则需要使用 addr[0]、addr[1]、addr[2]。
- size 表示输入图像数据的大小，数组大小为 3 的含义与 addr 类似。单位是 pixel。
- crop 表示输入图像需要显示的裁剪区域，x,y 为裁剪后区域的对应原始图像的左上角坐标，width 和 height 表示裁剪后的宽高，需要注意的是 crop 参数是个 64bit 的参数，前 32bit 表示小数，后 32bit 的单位是 pixel，即 crop 参数的每一个值都是 32 位的定点小数。
- screen\_win 表示的是需要显示的大小，用于缩放。当不需要缩放时，将 screen\_win 的大小设为与 crop 的宽高一致即可，当需要缩放时，将会将图像裁剪后的区域 crop.width×crop.height 缩放为 screen\_win.width×screen\_win.height 进行显示。

## 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、LCD、hdmi 等显示资源进行操作。

## 3.3 Global Interface

### 3.3.1 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

```
1 //启动cache, dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 arg[0] = 0;//屏0
4 arg[1] = 1;//protect
5 disp_ioctl( DISP_SHADOW_PROTECT, (void*)arg);
6 //do something other
7 arg[1] = 0;
8 disp_ioctl( DISP_SHADOW_PROTECT, (void*)arg);
```

### 3.3.2 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

```
1 //设置显示背景色, dispfd 为显示驱动句柄, sel 为屏0/1
2 disp_color bk;
3 unsigned long arg[3];
4 bk.red = 0xff;
5 bk.green = 0x00;
6 bk.blue = 0x00;
7 arg[0] = 0;
8 arg[1] = (unsigned int)&bk;
9 disp_ioctl( DISP_SET_BKCOLOR, (void*)arg);
10
```

### 3.3.3 DISP\_GET\_BKCOLOR

- 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\_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;
- 返回：
  - DIS\_SUCCESS: 成功
  - 其他: 失败号
- 示例

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

### 3.3.4 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

```
1 //获取屏幕水平分辨率
2 unsigned int screen_width;
3 unsigned long arg[3];
4 arg[0] = 0;
5 screen_width = disp_ioctl( DISP_GET_SCN_WIDTH, (void*)arg);
```

### 3.3.5 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

```
1 //获取屏幕垂直分辨率
2 unsigned int screen_height;
3 unsigned long arg[3];
4 arg[0] = 0;
5 screen_height = disp_ioctl( DISP_GET_SCN_HEIGHT, (void*)arg);
```

### 3.3.6 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

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

### 3.3.7 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

```
1 //获取当前显示输出类型
2 unsigned long arg[3];
3 struct disp_output output;
4 enum disp_output_type type;
5 enum disp_tv_mode mode;
6 arg[0] = 0;
7 arg[1] = (unsigned long)&output;
8 disp_ioctl( DISP_GET_OUTPUT, (void*)arg);
9 type = (enum disp_output_type)output.type;
10 mode = (enum disp_tv_mode)output.mode;
```

### 3.3.8 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

```
1 //开启/关闭vsync 消息发送功能，dispfd 为显示驱动句柄，sel 为屏0/1
2 unsigned long arg[3];
```

```
4   arg[0] = 0;
5   arg[1] = 1;
6   disp_ioctl( DISP_VSYNC_EVENT_EN, (void*)arg);
```

### 3.3.9 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

```
1
2   //切换
3   unsigned long arg[3];
4   arg[0] = 0;
5   arg[1] = (unsigned long)DISP_OUTPUT_TYPE_HDMI;
6   arg[2] = (unsigned long)DISP_TV_MOD_1080P_60HZ;
7   disp_ioctl( DISP_DEVICE_SWITCH, (void*)arg);
8   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。
```

### 3.3.10 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



```
1 //切换输出类型并设置输出设备的属性参数
2 unsigned long arg[3];
3 struct disp_device_config config;
4 config.type = DISP_OUTPUT_TYPE_HDMI;
5 config.mode = DISP_TV_MOD_1080P_60HZ;
6 config.format = DISP_CSC_TYPE_YUV420;
7 config.bits = DISP_DATA_10BITS;
8 config.eotf = DISP_EOTF_SMPTE2084;
9 config.cs = DISP_BT2020NC;
10 arg[0] = 0;
11 arg[1] = (unsigned long)&config;
12 disp_ioctl( DISP_DEVICE_SET_CONFIG, (void*)arg);
13 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。
```

### 3.3.11 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

```
1 //获取当前输出类型及相关的属性参数
2 unsigned long arg[3];
3 struct disp_device_config config;
4 arg[0] = 0;
5 arg[1] = (unsigned long)&config;
6 disp_ioctl( DISP_DEVICE_GET_CONFIG, (void*)arg);
7 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。
```

## 3.4 Layer Interface

### 3.4.1 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

```
1 struct
2 {
3     disp_layer_info info,
4     bool enable;
5     unsigned int channel,
6     unsigned int layer_id,
7 }disp_layer_config;
8 //设置图层参数, dispfd 为显示驱动句柄
9 unsigned long arg[3];
10 struct disp_layer_config config;
11 unsigned int width = 1280;
12 unsigned int height = 800;
13 unsigned int ret = 0;
14 memset(&config, 0, sizeof(struct disp_layer_config));
15 config.channnel = 0; //blending channel
16 config.layer_id = 0; //layer index in the blending channel
17 config.info.enable = 1;
18 config.info.mode = LAYER_MODE_BUFFER;
19 config.info.fb.addr[0] = (unsigned long long)mem_in; //FB 地址
20 config.info.fb.size[0].width = width;
21 config.info.fb.align[0] = 4; //bytes
22 config.info.fb.format = DISP_FORMAT_ARGB_8888; //DISP_FORMAT_YUV420_P
23 config.info.fb.crop.x = 0;
24 config.info.fb.crop.y = 0;
25 config.info.fb.crop.width = ((unsigned long)width) << 32; //定点小数。高32bit 为整数，低
26 32bit 为小数
27 config.info.fb.crop.height= ((unsigned long)height)<<32; //定点小数。高32bit 为整数，低
28 32bit 为小数
29 config.info.fb.flags = DISP_BF_NORMAL;
30 config.info.fb.scan = DISP_SCAN_PROGRESSIVE;
31 config.info.alpha_mode = 2; //global pixel alpha
32 config.info.alpha_value = 0xff; //global alpha value
33 config.info.screen_win.x = 0;
34 config.info.screen_win.y = 0;
35 config.info.screen_win.width = width;
36 config.info.screen_win.height= height;
37 config.info.id = 0;
38 arg[0] = 0; //screen 0
39 arg[1] = (unsigned long)&config;
40 arg[2] = 1; //one layer
41 ret = disp_ioctl( DISP_LAYER_SET_CONFIG, (void*)arg);
```

### 3.4.2 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

```
1 //设置图层参数, dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 struct disp_layer_config config;
4 memset(&config, 0, sizeof(struct disp_layer_config));
5 arg[0] = 0; //disp
6 arg[1] = (unsigned long)&config;
7 arg[2] = 1; //layer number
8 ret = disp_ioctl(DISP_GET_LAYER_CONFIG, (void*)arg);
9
```

### 3.4.3 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

```
1 struct
2 {
3     disp_layer_info info,
4
```

```

5     bool enable;
6     unsigned int channel,
7     unsigned int layer_id,
8 }disp_layer_config2;
9 //设置图层参数, dispfd 为显示驱动句柄
10 unsigned long arg[3];
11 struct disp_layer_config2 config;
12 unsigned int width = 1280;
13 unsigned int height = 800;
14 unsigned int ret = 0;
15 memset(&config, 0, sizeof(struct disp_layer_config2));
16 config.channnel = 0; //blending channel
17 config.layer_id = 0; //layer index in the blending channel
18 config.info.enable = 1;
19 config.info.mode = LAYER_MODE_BUFFER;
20 config.info.fb.addr[0] = (unsigned long long)mem_in; //FB 地址
21 config.info.fb.size[0].width = width;
22 config.info.fb.align[0] = 4; //bytes
23 config.info.fb.format = DISP_FORMAT_ARGB_8888; //DISP_FORMAT_YUV420_P
24 config.info.fb.crop.x = 0;
25 config.info.fb.crop.y = 0;
26 config.info.fb.crop.width = ((unsigned long)width) << 32; //定点小数。高32bit 为整数, 低
32bit 为小数
27 config.info.fb.crop.height = ((unsigned long)height) << 32; //定点小数。高32bit 为整数, 低
32bit 为小数
28 config.info.fb.flags = DISP_BF_NORMAL;
29 config.info.fb.scan = DISP_SCAN_PROGRESSIVE;
30 config.info.fb.eotf = DISP_EOTF_SMPTE2084; //HDR
31 config.info.fb.metadata_buf = (unsigned long long)mem_in2;
32 config.info.alpha_mode = 2; //global pixel alpha
33 config.info.alpha_value = 0xff; //global alpha value
34 config.info.screen_win.x = 0;
35 config.info.screen_win.y = 0;
36 config.info.screen_win.width = width;
37 config.info.screen_win.height = height;
38 config.info.id = 0;
39 arg[0] = 0; //screen 0
40 arg[1] = (unsigned long)&config;
41 arg[2] = 1; //one layer
42 ret = disp_ioctl( DISP_LAYER_SET_CONFIG2, (void*)arg);

```

### 3.4.4 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

```
1 //设置图层参数, dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 struct disp_layer_config2 config;
4 memset(&config, 0, sizeof(struct disp_layer_config2));
5 arg[0] = 0; //disp
6 arg[1] = (unsigned long)&config;
7 arg[2] = 1; //layer number
8 ret = disp_ioctl( DISP_GET_LAYER_CONFIG2, (void*)arg);
9
```

## 3.5 capture interface

### 3.5.1 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

```
1 //启动截屏功能, dispfd 为显示驱动句柄
2 arg[0] = 0; //显示通道0
3 disp_ioctl( DISP_CAPTURE_START, (void*)arg);
4
```

### 3.5.2 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, 用以设置截图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

```
1 //提交截屏功能, dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 struct disp_capture_info info;
4 arg[0] = 0;
5 screen_width = disp_ioctl( DISP_GET_SCN_WIDTH, (void*)arg);
6 screen_height = disp_ioctl( DISP_GET_SCN_HEIGHT, (void*)arg);
7 info.window.x = 0;
8 info.window.y = 0;
9 info.window.width = screen_width;
10 info.window.y = screen_height;
11 info.out_frame.format = DISP_FORMAT_ARGB_8888;
12 info.out_frame.size[0].width = screen_width;
13 info.out_frame.size[0].height = screen_height;
14 info.out_frame.crop.x = 0;
15 info.out_frame.crop.y = 0;
16 info.out_frame.crop.width = screen_width;
17 info.out_frame.crop.height = screen_height;
18 info.out_frame.addr[0] = fb_address; //buffer address
19 arg[0] = 0; //显示通道0
20 arg[1] = (unsigned long)&info;
21 disp_ioctl( DISP_CAPTURE_COMMIT, (void*)arg);
```

### 3.5.3 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

```
1 //停止截屏功能，dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 arg[0] = 0; //显示通道0
4 disp_ioctl( DISP_CAPTURE_STOP, (void*)arg);
5
```

### 3.5.4 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

```
1 //查询截屏是否成功，dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 arg[0] = 0; //显示通道0
4 disp_ioctl( DISP_CAPTURE_QUERY, (void*)arg);
5
```

## 3.6 LCD Interface

### 3.6.1 DISP\_LCD\_SET\_BRIGHTNESS

- 作用：该函数用于设置 LCD 亮度
- 参数：
  - 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）
- 返回：
  - DIS\_SUCCESS: 成功
  - 其他: 失败号
- 示例

```
1 //设置LCD 的背光亮度, dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 unsigned int bl = 197;
4 arg[0] = 0; //显示通道0
5 arg[1] = bl;
6 disp_ioctl( DISP_LCD_SET_BRIGHTNESS, (void*)arg);
7
```

### 3.6.2 DISP\_LCD\_GET\_BRIGHTNESS

- 作用：该函数用于获取 LCD 亮度
- 参数：
  - handle：显示驱动句柄
  - cmd：DISP\_LCD\_GET\_BRIGHTNESS
  - arg：arg[0] 为显示通道 0/1
- 返回：
  - DIS\_SUCCESS: 成功
  - 其他: 失败
- 示例

```
1 //获取LCD 的背光亮度, dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 unsigned int bl;
4 arg[0] = 0; //显示通道0
5 bl = disp_ioctl( DISP_LCD_GET_BRIGHTNESS, (void*)arg);
6
```

### 3.6.3 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



```
1 //设置lcd 的gamma table, dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 unsigned int gamma_tbl[1024];
4 unsigned int size = 1024;
5 /* init gamma table */
6 /* gamma_tbl[nn] = xx; */
7 arg[0] = 0; //显示通道0
8 arg[1] = gamma_tbl;
9 arg[2] = size;
10 if (disp_ioctl( DISP_LCD_SET_GAMMA_TABLE, (void*)arg))
11     printf( "set gamma table fail!\n" );
12 else
13     printf( "set gamma table success\n" );
14
```

### 3.6.4 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

```
1 //使能lcd 的gamma 校正功能, dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 arg[0] = 0; //显示通道0
4 if (disp_ioctl( DISP_LCD_GAMMA_CORRECTION_ENABLE, (void*)arg))
5     printf( "enable gamma correction fail!\n" );
6 else
7     printf( "enable gamma correction success\n" );
8
```

### 3.6.5 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

```
1 //关闭lcd 的gamma 校正功能, dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 arg[0] = 0; //显示通道0
4 if (disp_ioctl( DISP_LCD_GAMMA_CORRECTION_DISABLE, (void*)arg))
5     printf( "disable gamma correction fail!\n" );
6 else
7     printf( "disable gamma correction success\n" );
8
```

## 3.7 smart backlight

### 3.7.1 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

```
1 //开启智能背光功能, dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 arg[0] = 0; //显示通道0
4 disp_ioctl( DISP_SMBL_ENABLE, (void*)arg);
5
```

### 3.7.2 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

```
1 //关闭智能背光功能, dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 arg[0] = 0; //显示通道0
4 disp_ioctl( DISP_SMBL_DISABLE, (void*)arg);
5
```

### 3.7.3 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

```
1 //设置智能背光窗口, dispfd 为显示驱动句柄
2 unsigned long arg[3];
3 unsigned int screen_width, screen_height;
4 struct disp_rect window;
5 screen_width = disp_ioctl( DISP_GET_SCN_WIDTH, (void*)arg);
6 screen_height = disp_ioctl( DISP_GET_SCN_HEIGHT, (void*)arg);
7 window.x = 0;
8 window.y = 0;
9 window.width = screen_width / 2;
10 window.height = screen_height;
11 arg[0] = 0; //显示通道0
12 arg[1] = (unsigned long)&window;
13 disp_ioctl( DISP_SMBL_SET_WINDOW, (void*)arg);
14
```

## 3.8 Data Structure

### 3.8.1 disp\_fb\_info

- 作用：用于描述一个 display framebuffer 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],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
- 结构定义:

```
1  typedef struct
2  {
3      unsigned long long addr[3]; /* address of frame buffer, single addr for interleaved
4      fomart, double addr for semi-planar fomart triple addr for planar format */
5      disp_rectsz size[3]; //size for 3 component,unit: pixels
6      unsigned int align[3]; //align for 3 comonent,unit: bytes(align=2^n,i.e.
7      1/2/4/8/16/32..)
8      disp_pixel_format format;
9      disp_color_space color_space; //color space
10     unsigned int trd_right_addr[3];/* right address of 3d fb, used when in frame
11     packing 3d mode */
12     bool pre_multiply; //true: pre-multiply fb
13     disp_rect64 crop; //crop rectangle boundaries
14     disp_buffer_flags flags; //indicate stereo or non-stereo buffer
15     disp_scan_flags scan; //scan type & scan order
16 }disp_fb_info;
```

### 3.8.2 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:

```

1
2   - PROTOTYPE
3
4   typedef struct
5   {
6       disp_layer_mode mode;
7       unsigned char zorder; /*specifies the front-to-back ordering of the layers on the
8       screen, the top layer having the highest Z value can't set zorder, but can get */
9       unsigned char alpha_mode; //0: pixel alpha; 1: global alpha; 2: global pixel alpha
10      unsigned char alpha_value; //global alpha value
11      disp_rect screen_win; //display window on the screen
12      bool b_trd_out; //3d display
13      disp_3d_out_mode out_trd_mode; //3d display mode
14      union {
15          unsigned int color; //valid when LAYER_MODE_COLOR
16          disp_fb_info fb; //framebuffer, valid when LAYER_MODE_BUFFER
17      };
18      unsigned int id; /* frame id, can get the id of frame
19      display currently by DISP_LAYER_GET_FRAME_ID */
20  }disp_layer_info;

```

### 3.8.3 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:

```

1
2     typedef struct
3     {
4         disp_layer_info info;
5         bool enable;
6         unsigned int channel;
7         unsigned int layer_id;
8     }disp_layer_config;

```

### 3.8.4 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：

```

1
2     - PROTOTYPE
3
4     /* disp_fb_info2 - image buffer info v2
5     \*
6     \* @addr: buffer address for each plane
7     \* @size: size<width,height> for each buffer, unit:pixels
8     \* @align: align for each buffer, unit:bytes
9     \* @format: pixel format
10    \* @color_space: color space
11    \* @trd_right_addr: the right-eye buffer address for each plane,
12    \* valid when frame-packing 3d buffer input
13    \* @pre_multiply: indicate the pixel use premultiplied alpha
14    \* @crop: crop rectangle for buffer to be display
15    \* @flag: indicate stereo/non-stereo buffer
16    \* @scan: indicate interleave/progressive scan type, and the scan order
17    \* @metadata_buf: the phy_address to the buffer contained metadata for
18    fbc/hdr
19    \* @metadata_size: the size of metadata buffer, unit:bytes
20    \* @metadata_flag: the flag to indicate the type of metadata buffer
21    \* 0 : no metadata
22    \* 1 << 0: hdr static metadata
23    \* 1 << 1: hdr dynamic metadata
24    \* 1 << 4: frame buffer compress(fbc) metadata
25    \* x : all type could be "or" together
26    \*/

```

```

27     struct disp_fb_info2 {
28         unsigned long long addr[3];
29         struct disp_rectsz size[3];
30         unsigned int align[3];
31         enum disp_pixel_format format;
32         enum disp_color_space color_space;
33         unsigned int trd_right_addr[3];
34         bool pre_multiply;
35         struct disp_rect64 crop;
36         enum disp_buffer_flags flags;
37         enum disp_scan_flags scan;
38         enum disp_eotf eotf;
39         unsigned long long metadata_buf;
40         unsigned int metadata_size;
41         unsigned int metadata_flag;
42     };
43     /* disp_layer_info2 - layer info v2
44     */
45     /* @mode: buffer/color mode, when in color mode, the layer is without buffer
46     /* @zorder: the zorder of layer, 0~max-layer-number
47     /* @alpha_mode:
48     /* 0: pixel alpha;
49     /* 1: global alpha
50     /* 2: mixed alpha, compositing with pixel alpha before global alpha
51     /* @alpha_value: global alpha value, valid when alpha_mode is not pixel alpha
52     /* @screen_win: the rectangle on the screen for fb to be display
53     /* @b_trd_out: indicate if 3d display output
54     /* @out_trd_mode: 3d output mode, valid when b_trd_out is true
55     /* @color: the color value to be display, valid when layer is in color mode
56     /* @fb: the framebuffer info related with the layer, valid when in buffer mode
57     /* @id: frame id, the user could get the frame-id display currently by
58     /* DISP_LAYER_GET_FRAME_ID ioctl
59     /* @atw: asynchronous time wrap information
60     */
61     struct disp_layer_info2 {
62         enum disp_layer_mode mode;
63         unsigned char zorder;
64         unsigned char alpha_mode;
65         unsigned char alpha_value;
66         struct disp_rect screen_win;
67         bool b_trd_out;
68         enum disp_3d_out_mode out_trd_mode;
69         union {
70             unsigned int color;
71             struct disp_fb_info2 fb;
72         };
73         unsigned int id;
74         struct disp_atw_info atw;
75     };
76     /* disp_layer_config2 - layer config v2
77     */
78     /* @info: layer info
79     /* @enable: indicate to enable/disable the layer
80     /* @channel: the channel index of the layer, 0~max-channel-number
81     /* @layer_id: the layer index of the layer within it's channel
82     */
83
84     struct disp_layer_config2 {
85         struct disp_layer_info2 info;
86         bool enable;

```

```
87     unsigned int channel;  
88     unsigned int layer_id;  
89 };
```

### 3.8.5 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：

```
1  
2     - PROTOTYPE  
3  
4     typedef struct  
5     {  
6         u8 alpha;  
7         u8 red;  
8         u8 green;  
9         u8 blue;  
10    }disp_color_info;
```

### 3.8.6 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：

```
1  
2     typedef struct  
3     {  
4         s32 x;  
5         s32 y;  
6         u32 width;  
7         u32 height;
```



```
8 }disp_rect;
```

### 3.8.7 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：

```
1
2 typedef struct
3 {
4     long long x;
5     long long y;
6     long long width;
7     long long height;
8 }disp_rect64;
```

### 3.8.8 disp\_position

- 作用：用于描述一个坐标的信息
- 结构定义：

```
1
2 typedef struct
3 {
4     s32 x;
5     s32 y;
6 }disp_posistion;
```

### 3.8.9 disp\_rectsz

- 作用：用于描述一个矩形尺寸的信息
- 结构定义：

```
1
2     typedef struct
3     {
4         u32 width;
5         u32 height;
6     }disp_rectsz;
```

### 3.8.10 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：

```
1
2     /* disp_atw_mode - mode for asynchronous time warp
3     \*
4     \* @NORMAL_MODE: dual buffer, left eye and right eye buffer is individual
5     \* @LEFT_RIGHT_MODE: single buffer, the left half of each line buffer
6     \* is for left eye, the right half is for the right eye
7     \* @UP_DOWN_MODE: single buffer, the first half of the total buffer
8     \* is for the left eye, the second half is for the right eye
9     \*/
10    enum disp_atw_mode {
11        NORMAL_MODE,
12        LEFT_RIGHT_MODE,
13        UP_DOWN_MODE,
14    };
15    /* disp_atw_info - asynchronous time wrap infomation
16    \*
17    \* @used: indicate if the atw funtion is used
18    \* @mode: atw mode
19    \* @b_row: the row number of the micro block
20    \* @b_col: the column number of the micro block
21    \* @cof_addr: the address of buffer contaied coefficient for atw
22    \*/
23    struct disp_atw_info {
24        bool used;
25        enum disp_atw_mode mode;
26        unsigned int b_row;
27        unsigned int b_col;
28        unsigned long cof_addr;
29    };
```

### 3.8.11 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：

```

1  typedef enum
2  {
3
4      DISP_FORMAT_ARGB_8888 = 0x00, //MSB A-R-G-B LSB
5      DISP_FORMAT_ABGR_8888 = 0x01,
6      DISP_FORMAT_RGBA_8888 = 0x02,
7      DISP_FORMAT_BGRA_8888 = 0x03,
8      DISP_FORMAT_XRGB_8888 = 0x04,
9      DISP_FORMAT_XBGR_8888 = 0x05,
10     DISP_FORMAT_RGBX_8888 = 0x06,
11     DISP_FORMAT_BGRX_8888 = 0x07,
12     DISP_FORMAT_RGB_888 = 0x08,
13     DISP_FORMAT_BGR_888 = 0x09,
14     DISP_FORMAT_RGB_565 = 0x0a,
15     DISP_FORMAT_BGR_565 = 0x0b,
16     DISP_FORMAT_ARGB_4444 = 0x0c,
17     DISP_FORMAT_ABGR_4444 = 0x0d,
18     DISP_FORMAT_RGBA_4444 = 0x0e,
19     DISP_FORMAT_BGRA_4444 = 0x0f,
20     DISP_FORMAT_ARGB_1555 = 0x10,
21     DISP_FORMAT_ABGR_1555 = 0x11,
22     DISP_FORMAT_RGBA_5551 = 0x12,
23     DISP_FORMAT_BGRA_5551 = 0x13,
24
25     /* SP: semi-planar, P:planar, I:interleaved
26     /* UVUV: U in the LSBs; VUVU: V in the LSBs */
27     DISP_FORMAT_YUV444_I_AYUV = 0x40, //MSB A-Y-U-V LSB
28     DISP_FORMAT_YUV444_I_VUYA = 0x41, //MSB V-U-Y-A LSB
29     DISP_FORMAT_YUV422_I_YVYU = 0x42, //MSB Y-V-Y-U LSB
30     DISP_FORMAT_YUV422_I_YUYV = 0x43, //MSB Y-U-Y-V LSB
31     DISP_FORMAT_YUV422_I_UYVY = 0x44, //MSB U-Y-V-Y LSB
32     DISP_FORMAT_YUV422_I_VYUY = 0x45, //MSB V-Y-U-Y LSB
33     DISP_FORMAT_YUV444_P = 0x46, //MSB P3-2-1-0 LSB, YYYY UUUU VVVV
34     DISP_FORMAT_YUV422_P = 0x47, //MSB P3-2-1-0 LSB, YYYY UU VV
35     DISP_FORMAT_YUV420_P = 0x48, //MSB P3-2-1-0 LSB, YYYY U V
36     DISP_FORMAT_YUV411_P = 0x49, //MSB P3-2-1-0 LSB, YYYY U V
37     DISP_FORMAT_YUV422_SP_UVUV = 0x4a, //MSB V-U-V-U LSB
38     DISP_FORMAT_YUV422_SP_VUVU = 0x4b, //MSB U-V-U-V LSB
39     DISP_FORMAT_YUV420_SP_UVUV = 0x4c,
40     DISP_FORMAT_YUV420_SP_VUVU = 0x4d,

```

```
41     DISP_FORMAT_YUV411_SP_UVUV = 0x4e,  
42     DISP_FORMAT_YUV411_SP_VUVU = 0x4f,  
43     DISP_FORMAT_8BIT_GRAY = 0x50,  
44     DISP_FORMAT_YUV444_I_AYUV_10BIT = 0x51,  
45     DISP_FORMAT_YUV444_I_VUYA_10BIT = 0x52,  
46     DISP_FORMAT_YUV422_I_YVYU_10BIT = 0x53,  
47     DISP_FORMAT_YUV422_I_YUYV_10BIT = 0x54,  
48     DISP_FORMAT_YUV422_I_UYVY_10BIT = 0x55,  
49     DISP_FORMAT_YUV422_I_VYUY_10BIT = 0x56,  
50     DISP_FORMAT_YUV444_P_10BIT = 0x57,  
51     DISP_FORMAT_YUV422_P_10BIT = 0x58,  
52     DISP_FORMAT_YUV420_P_10BIT = 0x59,  
53     DISP_FORMAT_YUV411_P_10BIT = 0x5a,  
54     DISP_FORMAT_YUV422_SP_UVUV_10BIT = 0x5b,  
55     DISP_FORMAT_YUV422_SP_VUVU_10BIT = 0x5c,  
56     DISP_FORMAT_YUV420_SP_UVUV_10BIT = 0x5d,  
57     DISP_FORMAT_YUV420_SP_VUVU_10BIT = 0x5e,  
58     DISP_FORMAT_YUV411_SP_UVUV_10BIT = 0x5f,  
59     DISP_FORMAT_YUV411_SP_VUVU_10BIT = 0x60,  
60 }disp_pixel_format;;
```

### 3.8.12 disp\_data\_bits

- 作用：用于描述图像的数据宽度
- 结构定义：

```
1  
2     enum disp_data_bits {  
3         DISP_DATA_8BITS = 0,  
4         DISP_DATA_10BITS = 1,  
5         DISP_DATA_12BITS = 2,  
6         DISP_DATA_16BITS = 3,  
7     };
```

### 3.8.13 disp\_eotf

- 作用：用于描述图像的光电转换特性
- 结构定义：

```
1  
2     enum disp_eotf {  
3         DISP_EOTF_RESERVED = 0x000,  
4         DISP_EOTF_BT709 = 0x001,  
5         DISP_EOTF_UNDEF = 0x002,  
6         DISP_EOTF_GAMMA22 = 0x004, /* SDR */  
7         DISP_EOTF_GAMMA28 = 0x005,  
8         DISP_EOTF_BT601 = 0x006,  
9         DISP_EOTF_SMPTE240M = 0x007,  
10        DISP_EOTF_LINEAR = 0x008,
```

```

11     DISP_EOTF_LOG100 = 0x009,
12     DISP_EOTF_LOG100S10 = 0x00a,
13     DISP_EOTF_IEC61966_2_4 = 0x00b,
14     DISP_EOTF_BT1361 = 0x00c,
15     DISP_EOTF_IEC61966_2_1 = 0x00d,
16     DISP_EOTF_BT2020_0 = 0x00e,
17     DISP_EOTF_BT2020_1 = 0x00f,
18     DISP_EOTF_SMPTE2084 = 0x010, /* HDR10 */
19     DISP_EOTF_SMPTE428_1 = 0x011,
20     DISP_EOTF_ARIB_STD_B67 = 0x012, /* HLG */
21 };

```

### 3.8.14 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：

```

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

```

### 3.8.15 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:

```

1
2  typedef enum
3  {
4      //for lcd
5      DISP_3D_OUT_MODE_CI_1 = 0x5, //column interlaved 1
6      DISP_3D_OUT_MODE_CI_2 = 0x6, //column interlaved 2
7      DISP_3D_OUT_MODE_CI_3 = 0x7, //column interlaved 3
8      DISP_3D_OUT_MODE_CI_4 = 0x8, //column interlaved 4
9      DISP_3D_OUT_MODE_LIRGB = 0x9, //line interleaved rgb
10     //for hdmi
11     DISP_3D_OUT_MODE_TB = 0x0, //top bottom
12     DISP_3D_OUT_MODE_FP = 0x1, //frame packing
13     DISP_3D_OUT_MODE_SSF = 0x2, //side by side full
14     DISP_3D_OUT_MODE_SSH = 0x3, //side by side half
15     DISP_3D_OUT_MODE_LI = 0x4, //line interleaved
16     DISP_3D_OUT_MODE_FA = 0xa, //field alternative
17 }disp_3d_out_mode;

```

### 3.8.16 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:

```

1
2  enum disp_color_space
3  {
4      DISP_UNDEF = 0x00,
5      DISP_UNDEF_F = 0x01,
6      DISP_GBR = 0x100,
7      DISP_BT709 = 0x101,
8      DISP_FCC = 0x102,
9      DISP_BT470BG = 0x103,
10     DISP_BT601 = 0x104,
11     DISP_SMPTE240M = 0x105,

```

```
12     DISP_YCGCO = 0x106,  
13     DISP_BT2020NC = 0x107,  
14     DISP_BT2020C = 0x108,  
15     DISP_GBR_F = 0x200,  
16     DISP_BT709_F = 0x201,  
17     DISP_FCC_F = 0x202,  
18     DISP_BT470BG_F = 0x203,  
19     DISP_BT601_F = 0x204,  
20     DISP_SMPTE240M_F = 0x205,  
21     DISP_YCGCO_F = 0x206,  
22     DISP_BT2020NC_F = 0x207,  
23     DISP_BT2020C_F = 0x208,  
24     DISP_RESERVED = 0x300,  
25     DISP_RESERVED_F = 0x301,  
26 };
```

### 3.8.17 disp\_csc\_type

- 作用：用于描述图像颜色格式
- 结构定义：

```
1  
2     enum disp_csc_type  
3     {  
4         DISP_CSC_TYPE_RGB = 0,  
5         DISP_CSC_TYPE_YUV444 = 1,  
6         DISP_CSC_TYPE_YUV422 = 2,  
7         DISP_CSC_TYPE_YUV420 = 3,  
8     };
```

### 3.8.18 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：

```
1  
2     typedef enum  
3     {  
4         DISP_OUTPUT_TYPE_NONE = 0,
```

```
5     DISP_OUTPUT_TYPE_LCD = 1,  
6     DISP_OUTPUT_TYPE_TV = 2,  
7     DISP_OUTPUT_TYPE_HDMI = 4,  
8     DISP_OUTPUT_TYPE_VGA = 8,  
9 }disp_output_type;
```

### 3.8.19 disp\_tv\_mode

- 作用：用于描述 TV 输出模式
- 结构定义：

```
1  
2 typedef enum  
3 {  
4     DISP_TV_MOD_480I = 0,  
5     DISP_TV_MOD_576I = 1,  
6     DISP_TV_MOD_480P = 2,  
7     DISP_TV_MOD_576P = 3,  
8     DISP_TV_MOD_720P_50HZ = 4,  
9     DISP_TV_MOD_720P_60HZ = 5,  
10    DISP_TV_MOD_1080I_50HZ = 6,  
11    DISP_TV_MOD_1080I_60HZ = 7,  
12    DISP_TV_MOD_1080P_24HZ = 8,  
13    DISP_TV_MOD_1080P_50HZ = 9,  
14    DISP_TV_MOD_1080P_60HZ = 0xa,  
15    DISP_TV_MOD_1080P_24HZ_3D_FP = 0x17,  
16    DISP_TV_MOD_720P_50HZ_3D_FP = 0x18,  
17    DISP_TV_MOD_720P_60HZ_3D_FP = 0x19,  
18    DISP_TV_MOD_1080P_25HZ = 0x1a,  
19    DISP_TV_MOD_1080P_30HZ = 0x1b,  
20    DISP_TV_MOD_PAL = 0xb,  
21    DISP_TV_MOD_PAL_SVIDEO = 0xc,  
22    DISP_TV_MOD_NTSC = 0xe,  
23    DISP_TV_MOD_NTSC_SVIDEO = 0xf,  
24    DISP_TV_MOD_PAL_M = 0x11,  
25    DISP_TV_MOD_PAL_M_SVIDEO = 0x12,  
26    DISP_TV_MOD_PAL_NC = 0x14,  
27    DISP_TV_MOD_PAL_NC_SVIDEO = 0x15,  
28    DISP_TV_MOD_3840_2160P_30HZ = 0x1c,  
29    DISP_TV_MOD_3840_2160P_25HZ = 0x1d,  
30    DISP_TV_MOD_3840_2160P_24HZ = 0x1e,  
31    DISP_TV_MODE_NUM = 0x1f,  
32 }disp_tv_mode;
```

### 3.8.20 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:

```
1
2 struct disp_output
3 {
4     unsigned int type;
5     unsigned int mode;
6 };
```

### 3.8.21 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:

```
1
2 enum disp_layer_mode
3 {
4     LAYER_MODE_BUFFER = 0,
5     LAYER_MODE_COLOR = 1,
6 };
```

### 3.8.22 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:

```
1
2 /* disp_device_config - display device config
```

```
3  \*
4  \* @type: output type
5  \* @mode: output mode
6  \* @format: data format
7  \* @bits: data bits
8  \* @eotf: electro-optical transfer function
9  \* SDR : DISP_EOTF_GAMMA22
10 \* HDR10: DISP_EOTF_SMPTE2084
11 \* HLG : DISP_EOTF_ARIB_STD_B67
12 \* @cs: color space type
13 \* DISP_BT601: SDR for SD resolution(< 720P)
14 \* DISP_BT709: SDR for HD resolution(>= 720P)
15 \* DISP_BT2020NC: HDR10 or HLG or wide-color-gamut
16 \*/
17 struct disp_device_config {
18     enum disp_output_type type;
19     enum disp_tv_mode mode;
20     enum disp_csc_type format;
21     enum disp_data_bits bits;
22     enum disp_eotf eotf;
23     enum disp_color_space cs;
24     unsigned int reserve1;
25     unsigned int reserve2;
26     unsigned int reserve3;
27     unsigned int reserve4;
28     unsigned int reserve5;
29     unsigned int reserve6;
30 };
```

## 4 FAQ

### 4.1 调试命令

1、查看显示信息进入到系统控制台后，直接输入 `disp` 然后执行，就会打印出当前显示的信息，包括分辨率，fps、图层信息等，示例如下：

```
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[global 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[global 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， global/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 时为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

## 2、其他调试方法 Table:disp 命令的其他使用方法

选项	参数	解释	举例
-c	Screen_id,color 模式	显示 colorbar。共有 8 种模式, 0 到 8	disp -c 0 8
-b	Screen_id, 背光值	调整 lcd 背光, 背光值范围时 0 到 255	disp -b 0 255
-d	Screen_id, 文件路径	抓 DE 图层回写到文件	disp -d 0 /sdmmc/xx.bmp
-s	Screen_id, 显示类型, 显示分辨率	切换显示类型或分辨率	disp -s 0 1 4 打开 LCD 显示

## 4.2 模块使用范例

可参考 display 驱动测试用例 (hal/test/disp2/)。

## 4.3 常见问题

### 4.3.1 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，在板级配置文件上配置正确的参数。

- 步骤三

排查是否 pwm 有问题，可以试跑 pwm 的测试用例并且测量相关引脚的电压确定 pwm 是否正常。

- 步骤四

如果步骤三未解决问题，请排查配置文件。如果还没有解决，可以寻求技术支持。

### 4.3.2 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。

### 4.3.3 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。

### 4.3.4 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。

### 4.3.5 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”章节查看出现问题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，应用需要修改图层宽高。



## 著作权声明

版权所有 © 2021 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

## 商标声明

、 全志科技、（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

## 免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。