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 d1s

目　录

插　图

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/
D1s	Melis	rtos-hal/hal/source/disp2/

2 模块介绍

2.1 模块功能介绍

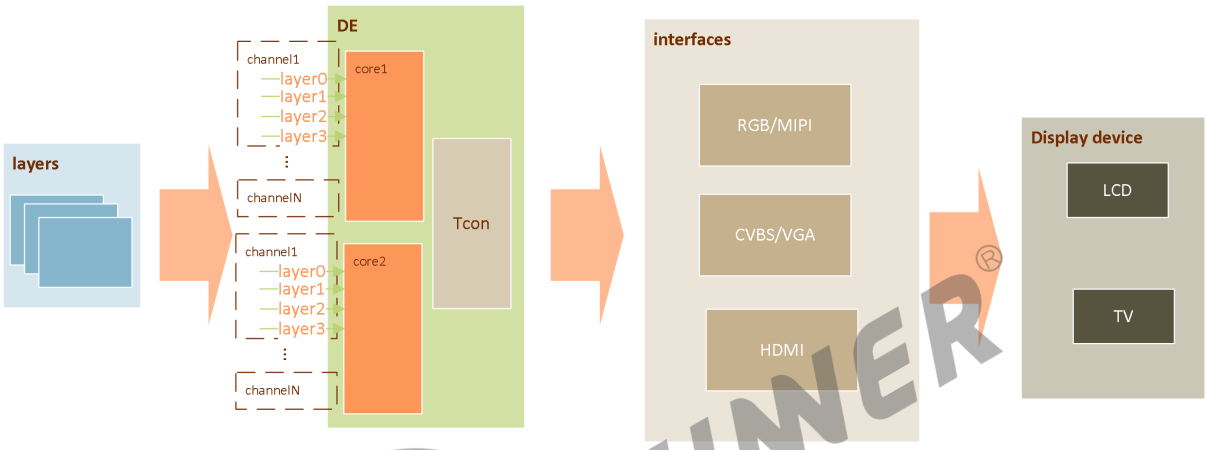

图 2-1: 模块框图

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 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 Operatiing 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 d1s 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_perf1.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_configg_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_INTGER,
        .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_DLDO1,
            .power_vol = 3300000,
        },
```

GPIO:

```
    {
        .name = "lcd_gpio_0",
        .type = PROPERTY_GPIO,
        .v.gpio_list = {
            .gpio = GPIOD(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 d1s-prototype-machine。

```
[lcd0]
lcd_used            = 1

lcd_driver_name     = "VVX07H005A10"
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: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_perf1.c                 --SOC平台配置文件
```

2.5 驱动框架介绍

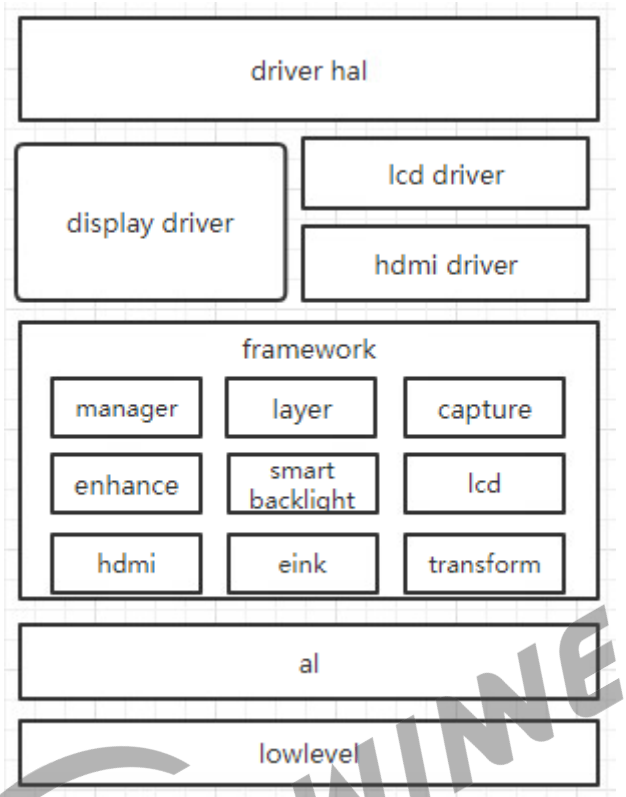

图 2-6: 驱动框图

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

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

```

    //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;
    disp_ioctl( DISP_SET_BKCOLOR, (void*)arg);
```

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
2   //获取屏幕垂直分辨率
3   unsigned int screen_height;
4   unsigned long arg[3];
5   arg[0] = 0;
6   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
2   //获取当前显示输出类型
3   disp_output_type output_type;
4   unsigned long arg[3];
5   arg[0] = 0;
6   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

```
//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;
disp_ioctl( DISP_GET_OUTPUT, (void*)arg);
type = (enum disp_output_type)output.type;
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

```
//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;
    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

```

    //切换
    unsigned long arg[3];
    arg[0] = 0;
    arg[1] =  (unsigned long)DISP_OUTPUT_TYPE_HDMI;
    arg[2] = (unsigned long)DISP_TV_MOD_1080P_60HZ;
    disp_ioctl( 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。
```

3.3.10 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
2   //切换输出类型并设置输出设备的属性参数
3   unsigned long arg[3];
4   struct disp_device_config config;
5   config.type = DISP_OUTPUT_TYPE_HDMI;
6   config.mode = DISP_TV_MOD_1080P_60HZ;
7   config.format = DISP_CSC_TYPE_YUV420;
8   config.bits = DISP_DATA_10BITS;
9   config.eotf = DISP_EOTF_SMPTE2084;
10  config.cs = DISP_BT2020NC;
11  arg[0] = 0;
12  arg[1] =  (unsigned long)&config;
13  disp_ioctl( DISP_DEVICE_SET_CONFIG, (void*)arg);
14  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
2   //获取当前输出类型及相关的属性参数
3   unsigned long arg[3];
4   struct disp_device_config config;
5   arg[0] = 0;
6   arg[1] =  (unsigned long)&config;
7   disp_ioctl( DISP_DEVICE_GET_CONFIG, (void*)arg);
8   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

```
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 = 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

```
//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 = disp_ioctl( DISP_GET_LAYER_CONFIG, (void*)arg);
```

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

```
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 = 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

```
//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 = disp_ioctl( DISP_GET_LAYER_CONFIG2, (void*)arg);
```

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

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

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，用以设置截取的窗口信息和保存图片的信息；

- 返回：
  - DIS_SUCCESS: 成功
  - 其他: 失败号
- 示例

```
1
2   //提交截屏功能，dispfd 为显示驱动句柄
3   unsigned long arg[3];
4   struct disp_capture_info info;
5   arg[0] = 0;
6   screen_width = disp_ioctl( DISP_GET_SCN_WIDTH, (void*)arg);
7   screen_height = disp_ioctl( DISP_GET_SCN_HEIGHT, (void*)arg);
8   info.window.x = 0;
9   info.window.y = 0;
10  info.window.width = screen_width;
11  info.window.y = screen_height;
12  info.out_frame.format = DISP_FORMAT_ARGB_8888;
13  info.out_frame.size[0].width = screen_width;
14  info.out_frame.size[0].height = screen_height;
15  info.out_frame.crop.x = 0;
16  info.out_frame.crop.y = 0;
17  info.out_frame.crop.width = screen_width;
18  info.out_frame.crop.height = screen_height;
19  info.out_frame.addr[0] = fb_address; //buffer address
20  arg[0] = 0;//显示通道0
21  arg[1] = (unsigned long)&info;
22  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
2   //停止截屏功能，dispfd 为显示驱动句柄
3   unsigned long arg[3];
4   arg[0] = 0;//显示通道0
5   disp_ioctl( DISP_CAPTURE_STOP, (void*)arg);
```

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

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

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

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

```
//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 (disp_ioctl( DISP_LCD_SET_GAMMA_TABLE, (void*)arg))
    printf( “set gamma table fail!\n” );
else
    printf( “set gamma table success\n” );
```

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

```
//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 (disp_ioctl( DISP_LCD_GAMMA_CORRECTION_ENABLE, (void*)arg))
    printf( “enable gamma correction fail!\n” );
else
    printf( “enable gamma correction success\n” );
```

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

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

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

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

3.8 Data Structure

3.8.1 disp_fb_info

- 作用：用于描述一个 display frambuffer 的属性信息
- 成员：
  - addr :frame buffer 的内容地址，对于 interleaved 类型，只有 addr[0] 有效；planar 类型，三个都有效；UV combined 的类型 addr[0],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
2     typedef struct
3     {
4         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 */
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.
      1/2/4/8/16/32..)
7         disp_pixel_format format;
8         disp_color_space color_space; //color space
9         unsigned int trd_right_addr[3];/* right address of 3d fb, used when in frame
      packing 3d mode */
10        bool pre_multiply; //true: pre-multiply fb
11        disp_rect64 crop; //crop rectangle boundaries
12        disp_buffer_flags flags; //indicate stereo or non-stereo buffer
13        disp_scan_flags scan; //scan type & scan order
14    }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：

```
- PROTOTYPE

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

3.8.3 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：

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

```
- 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;
```

```
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;
```

```
    }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：

```

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

3.8.8 disp_position

- 作用：用于描述一个坐标的信息
- 结构定义：

```

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

3.8.9 disp_rectsz

- 作用：用于描述一个矩形尺寸的信息
- 结构定义：

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

3.8.10 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：

```
/\* disp_atw_mode - mode for asynchronous time warp
\*
\* @NORMAL_MODE: dual buffer, left eye and right eye buffer is individual
\* @LEFT_RIGHT_MODE: single buffer, the left half of each line buffer
\* is for left eye, the right half is for the right eye
\* @UP_DOWN_MODE: single buffer, the first half of the total buffer
\* is for the left eye, the second half is for the right eye
\*/
enum disp_atw_mode {
    NORMAL_MODE,
    LEFT_RIGHT_MODE,
    UP_DOWN_MODE,
};
/\* disp_atw_info - asynchronous time wrap infomation
\*
\* @used: indicate if the atw funtion is used
\* @mode: atw mode
\* @b_row: the row number of the micro block
\* @b_col: the column number of the micro block
\* @cof_addr: the address of buffer contaied coefficient for atw
\*/
struct disp_atw_info {
    bool used;
    enum disp_atw_mode mode;
    unsigned int b_row;
    unsigned int b_col;
    unsigned long cof_addr;
};
```

3.8.11 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：

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

    /\* SP: semi-planar, P:planar, I:interleaved
    \* UVUV: U in the LSBs; VUVU: V in the LSBs \*/
    DISP_FORMAT_YUV444_I_AYUV = 0x40,//MSB A-Y-U-V LSB
    DISP_FORMAT_YUV444_I_VUYA = 0x41,//MSB V-U-Y-A LSB
    DISP_FORMAT_YUV422_I_YVYU = 0x42,//MSB Y-V-Y-U LSB
    DISP_FORMAT_YUV422_I_YUYV = 0x43,//MSB Y-U-Y-V LSB
    DISP_FORMAT_YUV422_I_UYVY = 0x44,//MSB U-Y-V-Y LSB
    DISP_FORMAT_YUV422_I_VYUY = 0x45,//MSB V-Y-U-Y LSB
    DISP_FORMAT_YUV444_P = 0x46,//MSB P3-2-1-0 LSB, YYYY UUUU VVVV
    DISP_FORMAT_YUV422_P = 0x47,//MSB P3-2-1-0 LSB, YYYY UU VV
    DISP_FORMAT_YUV420_P = 0x48,//MSB P3-2-1-0 LSB, YYYY U V
    DISP_FORMAT_YUV411_P = 0x49,//MSB P3-2-1-0 LSB, YYYY U V
    DISP_FORMAT_YUV422_SP_UVUV = 0x4a,//MSB V-U-V-U LSB
    DISP_FORMAT_YUV422_SP_VUVU = 0x4b,//MSB U-V-U-V LSB
    DISP_FORMAT_YUV420_SP_UVUV = 0x4c,
    DISP_FORMAT_YUV420_SP_VUVU = 0x4d,
```

```
        DISP_FORMAT_YUV411_SP_UVUV = 0x4e,
        DISP_FORMAT_YUV411_SP_VUVU = 0x4f,
        DISP_FORMAT_8BIT_GRAY = 0x50,
        DISP_FORMAT_YUV444_I_AYUV_10BIT = 0x51,
        DISP_FORMAT_YUV444_I_VUYA_10BIT = 0x52,
        DISP_FORMAT_YUV422_I_YVYU_10BIT = 0x53,
        DISP_FORMAT_YUV422_I_YUYV_10BIT = 0x54,
        DISP_FORMAT_YUV422_I_UYVY_10BIT = 0x55,
        DISP_FORMAT_YUV422_I_VYUY_10BIT = 0x56,
        DISP_FORMAT_YUV444_P_10BIT = 0x57,
        DISP_FORMAT_YUV422_P_10BIT = 0x58,
        DISP_FORMAT_YUV420_P_10BIT = 0x59,
        DISP_FORMAT_YUV411_P_10BIT = 0x5a,
        DISP_FORMAT_YUV422_SP_UVUV_10BIT = 0x5b,
        DISP_FORMAT_YUV422_SP_VUVU_10BIT = 0x5c,
        DISP_FORMAT_YUV420_SP_UVUV_10BIT = 0x5d,
        DISP_FORMAT_YUV420_SP_VUVU_10BIT = 0x5e,
        DISP_FORMAT_YUV411_SP_UVUV_10BIT = 0x5f,
        DISP_FORMAT_YUV411_SP_VUVU_10BIT = 0x60,
    }disp_pixel_format;;
```

3.8.12 disp_data_bits

- 作用：用于描述图像的数据宽度
- 结构定义：

```

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

3.8.13 disp_eotf

- 作用：用于描述图像的光电转换特性
- 结构定义：

```

    enum disp_eotf {
        DISP_EOTF_RESERVED = 0x000,
        DISP_EOTF_BT709 = 0x001,
        DISP_EOTF_UNDEF = 0x002,
        DISP_EOTF_GAMMA22 = 0x004, /* SDR */
        DISP_EOTF_GAMMA28 = 0x005,
        DISP_EOTF_BT601 = 0x006,
        DISP_EOTF_SMPTE240M = 0x007,
        DISP_EOTF_LINEAR = 0x008,
```

```
        DISP_EOTF_LOG100 = 0x009,
        DISP_EOTF_LOG100S10 = 0x00a,
        DISP_EOTF_IEC61966_2_4 = 0x00b,
        DISP_EOTF_BT1361 = 0x00c,
        DISP_EOTF_IEC61966_2_1 = 0X00d,
        DISP_EOTF_BT2020_0 = 0x00e,
        DISP_EOTF_BT2020_1 = 0x00f,
        DISP_EOTF_SMPTE2084 = 0x010, /* HDR10 */
        DISP_EOTF_SMPTE428_1 = 0x011,
        DISP_EOTF_ARIB_STD_B67 = 0x012, /* HLG */
    };
```

3.8.14 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：

```

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

3.8.15 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：

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

3.8.16 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：

```
enum disp_color_space
{
    DISP_UNDEF = 0x00,
    DISP_UNDEF_F = 0x01,
    DISP_GBR = 0x100,
    DISP_BT709 = 0x101,
    DISP_FCC = 0x102,
    DISP_BT470BG = 0x103,
    DISP_BT601 = 0x104,
    DISP_SMPTE240M = 0x105,
```

```
        DISP_YCGCO = 0x106,
        DISP_BT2020NC = 0x107,
        DISP_BT2020C = 0x108,
        DISP_GBR_F = 0x200,
        DISP_BT709_F = 0x201,
        DISP_FCC_F = 0x202,
        DISP_BT470BG_F = 0x203,
        DISP_BT601_F = 0x204,
        DISP_SMPTE240M_F = 0x205,
        DISP_YCGCO_F = 0x206,
        DISP_BT2020NC_F = 0x207,
        DISP_BT2020C_F = 0x208,
        DISP_RESERVED = 0x300,
        DISP_RESERVED_F = 0x301,
    };
```

3.8.17 disp_csc_type

- 作用：用于描述图像颜色格式
- 结构定义：

```

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

3.8.18 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：

```

    typedef enum
    {
        DISP_OUTPUT_TYPE_NONE = 0,
```

```
    DISP_OUTPUT_TYPE_LCD = 1,
    DISP_OUTPUT_TYPE_TV = 2,
    DISP_OUTPUT_TYPE_HDMI = 4,
    DISP_OUTPUT_TYPE_VGA = 8,
}disp_output_type;
```

3.8.19 disp_tv_mode

- 作用：用于描述 TV 输出模式
- 结构定义：

```

typedef enum
{
    DISP_TV_MOD_480I = 0,
    DISP_TV_MOD_576I = 1,
    DISP_TV_MOD_480P = 2,
    DISP_TV_MOD_576P = 3,
    DISP_TV_MOD_720P_50HZ = 4,
    DISP_TV_MOD_720P_60HZ = 5,
    DISP_TV_MOD_1080I_50HZ = 6,
    DISP_TV_MOD_1080I_60HZ = 7,
    DISP_TV_MOD_1080P_24HZ = 8,
    DISP_TV_MOD_1080P_50HZ = 9,
    DISP_TV_MOD_1080P_60HZ = 0xa,
    DISP_TV_MOD_1080P_24HZ_3D_FP = 0x17,
    DISP_TV_MOD_720P_50HZ_3D_FP = 0x18,
    DISP_TV_MOD_720P_60HZ_3D_FP = 0x19,
    DISP_TV_MOD_1080P_25HZ = 0x1a,
    DISP_TV_MOD_1080P_30HZ = 0x1b,
    DISP_TV_MOD_PAL = 0xb,
    DISP_TV_MOD_PAL_SVIDEO = 0xc,
    DISP_TV_MOD_NTSC = 0xe,
    DISP_TV_MOD_NTSC_SVIDEO = 0xf,
    DISP_TV_MOD_PAL_M = 0x11,
    DISP_TV_MOD_PAL_M_SVIDEO = 0x12,
    DISP_TV_MOD_PAL_NC = 0x14,
    DISP_TV_MOD_PAL_NC_SVIDEO = 0x15,
    DISP_TV_MOD_3840_2160P_30HZ = 0x1c,
    DISP_TV_MOD_3840_2160P_25HZ = 0x1d,
    DISP_TV_MOD_3840_2160P_24HZ = 0x1e,
    DISP_TV_MODE_NUM = 0x1f,
}disp_tv_mode;
```

3.8.20 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：

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

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：

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

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：

```
/\* disp_device_config - display deivce config
```

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

4 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[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
```

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，应用需要修改图层宽高。

著作权声明

商标声明

免责声明