H616

显示模块使用文档

1.5

2019.12.30

文档履历

版本号	日期	制/修订人	内容描述
1.5	2019.12.30	AW	Initial

目录

1. 概述

1.1 编写目的

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

1.2 适用范围

本模块设计适用于 A83T/V3/H8/H3/A64/H64/B100/A20E/V40/A63/H6/V5/T7/A50/H616/T507 平台。

1.3 相关人员

与显示相关的应用开发人员，及与显示相关的其他模块的开发人员，以及新人。

2. 模块介绍

2.1 模块功能介绍

本模块主要处理显示相关功能，主要功能如下：

- 支持 linux 标准的 framebuffer 接口
- 支持 lcd(hv/lvds/cpu/dsi) 输出
- 支持多图层叠加混合处理
- 支持多种显示效果处理（alpha, colorkey, 图像增强，亮度/对比度/饱和度/色度调整）
- 支持智能背光调节
- 支持多种图像数据格式输入 (arg,yuv)
- 支持图像缩放处理
- 支持截屏
- 支持图像转换

2.2 相关术语介绍

- Framebuffer device : A framebuffer device is an abstraction for the graphic hardware. It represents the frame buffer of some video hardware, and allows application software to access the graphic hardware through a well-defined interface, so that the software doesn't need to know anything about the low-level interface。
- Layer: a display resource of video hardware,the image buffer of a layer can be display on the screen.
- alpha compositing: it is the process of combining an image with a background to create the appearance of partial or full transparency。
- Capture: 截屏，controller2memory 的处理。
- transform: 图像变换

2.3 模块配置介绍

2.3.1 menuconfig 配置说明

在命令行中进入内核根目录，执行 make ARCH=arm(64) menuconfig 进入配置主界面。并按以下步骤操作：

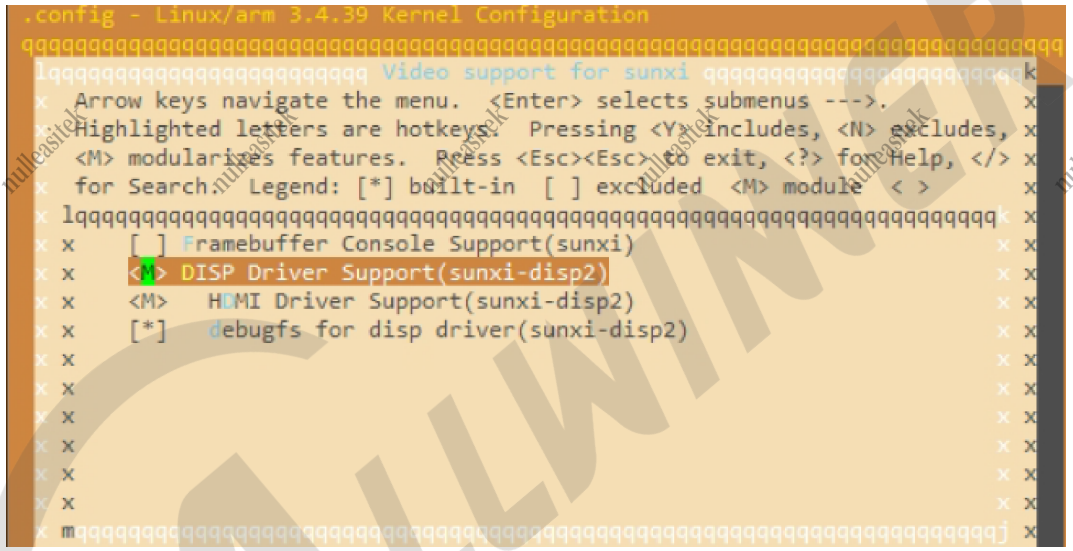

图 1: menuconfig

具体配置目录为：Device Drivers->Graphics support->Support for frame buffer devices->Video Support for sunxi -> DISP Driver Support(sunxi-disp2) / HDMI Drivers Support

2.4 源码结构介绍

```
drivers
├─video 显示驱动目录
| ├─fbmem.c framebuffer core
| ├─sunxi display driver for sunxi
```

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

3. 图层操作说明

显示驱动中最重要的显示资源为图层，sunxi 中支持 1 到 2 路显示通道，0 路显示一般支持 16 个图层（其中视频图层 4 个），3 个 blending 通道；1 路一般支持 8 个图层（其中视频图层 4 个），1 个 Blending 通道，所有图层都支持缩放。对图层的操作如下所示。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)。

- 设置图层参数并使能，接口为 DISP_LAYER_SET_CONFIG，图像格式，buffer size，buffer 地址，alpha 模式，enable，图像帧 id 号等参数。
- 关闭图层，依然通过 DISP_LAYER_SET_CONFIG，将 enable 参数设置为 0 关闭。

4. 显示输出设备操作说明

Disp2 支持多种显示输出设备，LCD、TV、HDMI。开启显示输出设备有几种方式，第一种是在 sys_config 或 dts 中配置 [disp] 的初始化参数，显示模块在加载时将会根据配置初始化选择的显示输出设备；第二种是在 kernel 启动后，调用驱动模块的 ioctl 接口去开启或关闭指定的输出设备，以下是操作的说明：

- 开启或切换到某个具体的显示输出设备，ioctl(DISP_DEVICE_SWITCH…)，参数设置为特定的输出设备类型，DISP_OUTPUT_TYPE_LCD/TV/HDMI。
- 关闭某个设备，ioctrl(DISP_DEVICE_SWITCH…)，参数设置为 DISP_OUTPUT_TYPE_NONE。

5. 接口参数更改说明

项目平台	Disp2	Disp1
图层标识	以 disp, chennel, layer_id 唯一标识	以 disp, layer_id 唯一标识
图层开关	将开关当成参数放置于 DISP_LAYER_SET_CONFIG 中	独立图层开关接口
图层 size	每个分量都需要设置 1 个 size	一个 buffer 只有 1 个 size
图层 align	针对每个分量需要设置其 align，单位为 byte	无
图层 Crop	为 64 位定点小数，高 32 位为整数，低 32 位为小数	为 32 位参数，不支持小数
YUV MB 格式支持	不再支持	支持
PALETTE 格式支持	不再支持	支持
单色模式 (无 buffer)	支持	不支持
Pipe 选择	Pipe 对用户透明，无需选择，只要配 channel	用户设置
zorder	用户设置且保证 zorder 不重复，0 到 N-1	用户不能设置
设置图层信息接口	一次可设置多个图层的信息, 增加图层数目参数	一次设置 1 个图层信息

6. 图层主要参数介绍

6.1 Size 与 crop

Fb 有两个与 size 有关的参数，分别是 size 与 crop。Size 表示 buffer 的完整尺寸，crop 则表示 buffer 中需要显示裁减区。如下图所示，完整的图像以 size 标识，而矩形框住的部分为裁减区，以 crop 标识，在屏幕上只能看到 crop 标识的部分，其余部分是隐藏的，不能在屏幕上显示出来的。

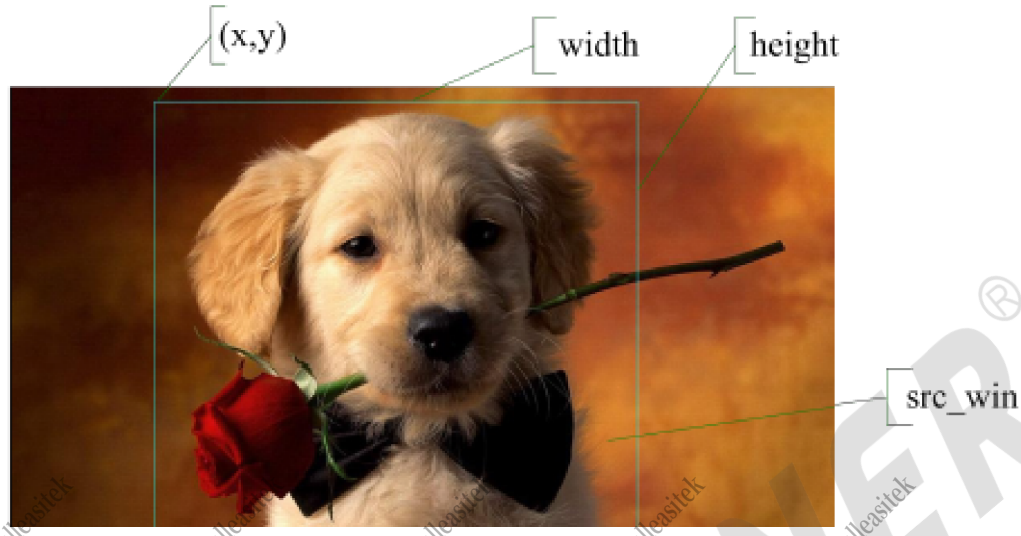

图 2: menuconfig

6.2 crop 和 screen_win

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

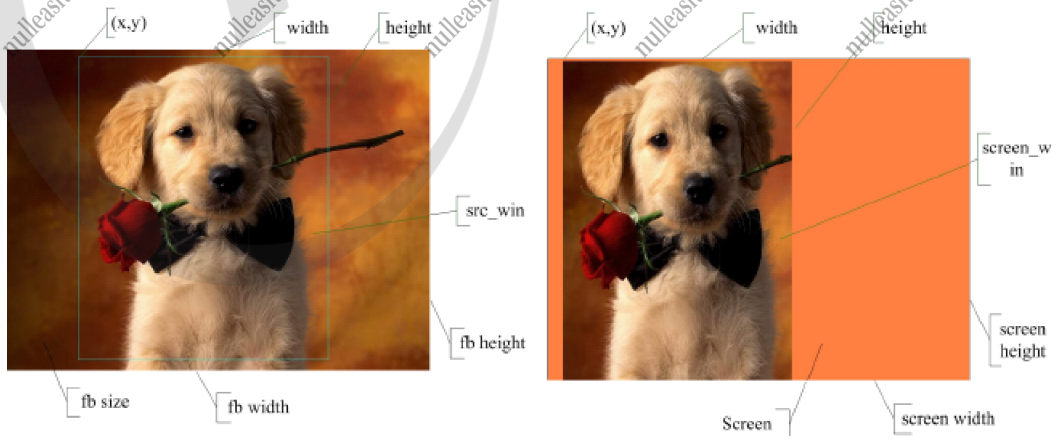

图 3: menuconfig

6.3 alpha

Alpha 模式有三种:

- Gloabal alpha: 全局 alpha, 也叫面 alpha，即整个图层共用一个 alpha，统一的透明度
- Pixel alpha: 点 alpha，即每个像素都有自己单独的 alpha，可以实现部分区域全透，部分区
- 域半透，部分区域不透的效果
- Global_pixel alpha: 可以是说以上两种效果的叠加，在实现 pxiel alpha 的效果的同时，还可以做淡入浅出的效果。

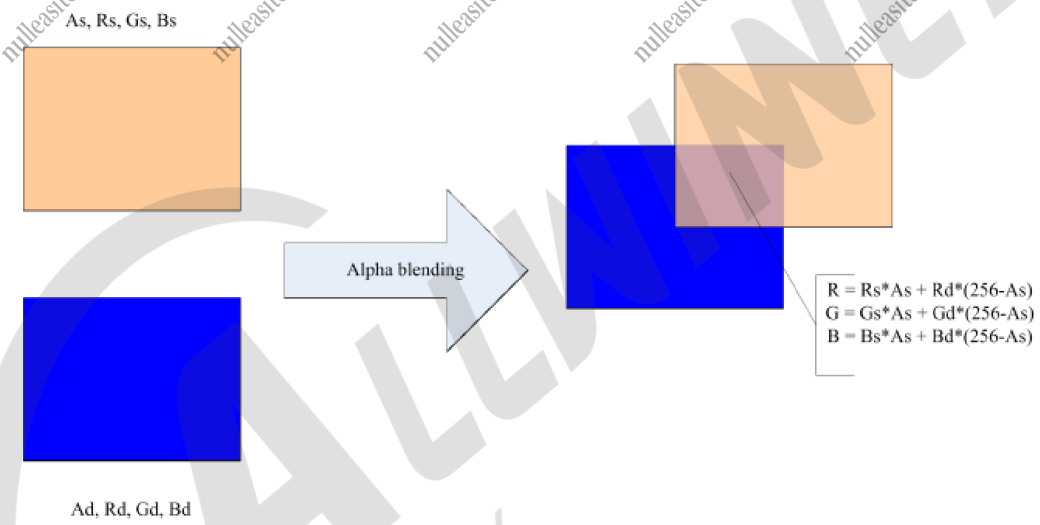

图 4: menuconfig

6.4 Format 支持

- Ui 通道支持的格式：

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

- Video 通道支持的格式：

```
DISP_FORMAT_ARGB_8888
DISP_FORMAT_ABGR_8888
DISP_FORMAT_RGBA_8888
DISP_FORMAT_BGRA_8888
DISP_FORMAT_XRGB_8888
DISP_FORMAT_XBGR_8888
DISP_FORMAT_RGBX_8888
DISP_FORMAT_BGRX_8888
DISP_FORMAT_RGB_888
DISP_FORMAT_BGR_888
DISP_FORMAT_RGB_565
DISP_FORMAT_BGR_565
DISP_FORMAT_ARGB_4444
DISP_FORMAT_ABGR_4444
DISP_FORMAT_RGBA_4444
DISP_FORMAT_BGRA_4444
DISP_FORMAT_ARGB_1555
```

```
DISP_FORMAT_ABGR_1555
DISP_FORMAT_RGBA_5551
DISP_FORMAT_BGRA_5551
DISP_FORMAT_YUV444_I_AYUV
DISP_FORMAT_YUV444_I_VUYA
DISP_FORMAT_YUV422_I_YVYU
DISP_FORMAT_YUV422_I_YUYV
DISP_FORMAT_YUV422_I_UYVY
DISP_FORMAT_YUV422_I_VYUY
DISP_FORMAT_YUV444_P
DISP_FORMAT_YUV422_P
DISP_FORMAT_YUV420_P
DISP_FORMAT_YUV411_P
DISP_FORMAT_YUV422_SP_UVUV
DISP_FORMAT_YUV422_SP_VUVU
DISP_FORMAT_YUV420_SP_UVUV
DISP_FORMAT_YUV420_SP_VUVU
DISP_FORMAT_YUV411_SP_UVUV
DISP_FORMAT_YUV411_SP_VUVU
DISP_FORMAT_YUV444_I_AYUV_10BIT
DISP_FORMAT_YUV444_I_VUYA_10BIT
```

7. 输出设备介绍

该平台支持屏以及 HDMI 输出，及二者同时显示。

7.1 屏

屏的接口很多，请参考 IC 规格书。

7.2 HDMI

HDMI 全名是：High-Definition Multimedia Interface。可以提供 DVD,audio device,set-top boxes, television sets, and other video displays 之间的高清互联。可以承载音，视频数据，以及其他的控制，数据信息。支持热插拔，内容保护，模式是否支持的查询。

7.3 同显

驱动支持双路显示。屏（主）+ HDMI（辅）。同显或异显，差别只在于显示内容，如果显示内容一样，则为同显；反之，则为异显。

- 如果是 android 系统，4.2 版本以上版本，原生框架已经支持多显（同显，异显，虚拟显示设备），实现同显则比较简单，在 android hal 与上层对接好即可。
- 如果是 android 4.1 以下版本，同显需要自行实现，参考做法为主屏内容由 android 原生提供，辅屏需要 android hal 在合适的时机（比如 HDMI 插入时）打开辅屏，并且将主屏的内容（存放于 FB0 中），拷贝至辅屏的显示后端 buffer 中，然后将辅屏的后端 buffer 切换到前端 buffer。注意问题为，两路显示的显示 buffer 的同步，如果同步不好，会产生图像撕裂，错位的现象。
- 如果是 Linux 系统，做法与上一个做法类似。

8. IOCTL 接口描述

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。

8.1 Global Interface

8.1.1 DISP_SHADOW_PROTECT

- PROTOTYPE

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

- ARGUMENTS

```
hdle 显示驱动句柄；
cmd DISP_SHADOW_PROTECT；
arg arg[0]为显示通道0/1；
arg[1]为protect 参数，1 表示protect, 0:表示not protect
```

- RETURNS

```
如果成功，返回DIS_SUCCESS，否则，返回失败号；
```

- DESCRIPTION

DISP_SHADOW_PROTECT（1）与DISP_SHADOW_PROTECT（0）配对使用，在两个接口调用之间的接口调用将不马上执行,而等到调用DISP_SHADOW_

- DEMO

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

8.1.2 DISP_SET_BKCOLOR

- PROTOTYPE

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

- ARGUMENTS

```
hdle 显示驱动句柄；
cmd DISP_SET_BKCOLOR；
arg arg[0]为显示通道0/1；
arg[1]为backcolor 信息，指向disp_color 数据结构指针；
- RETURNS
如果成功，返回DIS_SUCCESS，否则，返回失败号；
```

- DESCRIPTION

该函数用于设置显示背景色。

- DEMO

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

8.1.3 DISP_GET_BKCOLOR

- PROTOTYPE

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

- ARGUMENTS

```
hdle 显示驱动句柄；
cmd DISP_GET_BKCOLOR；
arg arg[0]为显示通道0/1
arg[1]为backcolor 信息，指向disp_color 数据结构指针；
```

- RETURNS

如果成功，返回DIS_SUCCESS，否则，返回失败号；

- DESCRIPTION

该函数用于获取显示背景色。

- DEMO

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

8.1.4 DISP_GET_SCN_WIDTH

- PROTOTYPE

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

- ARGUMENTS

hdle 显示驱动句柄；
cmd DISP_GET_SCN_WIDTH；
arg 显示通道0/1；

- RETURNS

如果成功，返回当前屏幕水平分辨率，否则，返回失败号；

- DESCRIPTION

该函数用于获取当前屏幕水平分辨率。

- DEMO

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

8.1.5 DISP_GET_SCN_HEIGHT

- PROTOTYPE

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

- ARGUMENTS

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

- RETURNS

```
如果成功，返回当前屏幕垂直分辨率，否则，返回失败号；
```

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

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

8.1.6 DISP_GET_OUTPUT_TYPE

- PROTOTYPE

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

- ARGUMENTS

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

- RETURNS

```
如果成功，返回当前显示输出类型，否则，返回失败号；
```

- DESCRIPTION

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

- DEMO

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

8.1.7 DISP_GET_OUTPUT

- PROTOTYPE

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

- ARGUMENTS

```
hdle 显示驱动句柄；
cmd DISP_GET_OUTPUT
arg arg[0]为显示通道0/1；
Arg[1]为指向disp_output 结构体的指针，用于保存返回值
```

- RETURNS

```
如果成功，返回0，否则，返回失败号；
```

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

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

8.1.8 DISP_VSYNC_EVENT_EN

- PROTOTYPE

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

- ARGUMENTS

```
hdle 显示驱动句柄；
cmd DISP_VSYNC_EVENT_EN；
arg arg[0]为显示通道0/1；
arg[1]为enable 参数，0：disable, 1:enable
```

- RETURNS

```
如果成功，返回DIS_SUCCESS，否则，返回失败号；
```

- DESCRIPTION

```
该函数开启/关闭vsync 消息发送功能。
```

- DEMO

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

8.1.9 DISP_DEVICE_SWITCH

- PROTOTYPE

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

- ARGUMENTS

```
hdle 显示驱动句柄；
cmd DISP_DEVICE_SWITCH；
arg arg[0]为显示通道0/1；
arg[1]为输出类型
arg[2]为输出模式，在输出类型不为LCD 时有效
```

- RETURNS

```
如果成功，返回DIS_SUCCESS，否则，返回失败号；
```

- DESCRIPTION

```
该函数用于切换输出类型
```

- DEMO

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

```
ioctl(disphd, 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。
```

8.1.10 DISP_DEVICE_SET_CONFIG

- PROTOTYPE

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

- ARGUMENTS

```
hdle 显示驱动句柄；
cmd DISP_DEVICE_SET_CONFIG；
arg arg[0]为显示通道0/1；
arg[1]为指向disp_device_config 的指针
```

- RETURNS

```
如果成功，返回DIS_SUCCESS，否则，返回失败号；
```

- DESCRIPTION

```
该函数用于切换输出类型并设置输出设备的属性参数
```

- DEMO

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

8.1.11 DISP_DEVICE_GET_CONFIG

- PROTOTYPE

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

- ARGUMENTS

```
hdle 显示驱动句柄；
cmd DISP_DEVICE_GET_CONFIG；
arg arg[0]为显示通道0/1；
arg[1]为指向disp_device_config 的指针
```

- RETURNS

如果成功，返回DIS_SUCCESS，否则，返回失败号；

- DESCRIPTION

该函数用于获取当前输出类型及相关的属性参数

- DEMO

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

8.2 Layer Interface

8.2.1 DISP_LAYER_SET_CONFIG

- PROTOTYPE

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

- ARGUMENTS

hdle 显示驱动句柄；
cmd DISP_SET_LAYER_CONFIG
arg arg[0]为显示通道0/1；
arg[1]为图层配置参数指针；
arg[2]为需要配置的图层数目

- RETURNS

如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。

- DESCRIPTION

该函数用于设置多个图层信息。

- DEMO

```
struct
{
    disp_layer_info info,
    bool enable;
    unsigned int channel,
    unsigned int layer_id,
}disp_layer_config;
//设置图层参数，disphd 为显示驱动句柄
unsigned long arg[3];
struct disp_layer_config config;
unsigned int width = 1280;
unsigned int height = 800;
unsigned int ret = 0;
memset(&config, 0, sizeof(struct disp_layer_config));
config.channnel = 0;//blending channel
```

```
config.layer_id = 0;//layer index in the blending channel
config.info.enable = 1;
config.info.mode = LAYER_MODE_BUFFER;
config.info.fb.addr[0] = (unsigned long long)mem_in; //FB 地址
config.info.fb.size[0].width = width;
config.info.fb.align[0] = 4;//bytes
config.info.fb.format = DISP_FORMAT_ARGB_8888; //DISP_FORMAT_YUV420_P
config.info.fb.crop.x = 0;
config.info.fb.crop.y = 0;
config.info.fb.crop.width = ((unsigned long)width) << 32;//定点小数。高32bit 为整数，低32bit 为小数
config.info.fb.crop.height= ((uunsigned long)height)<<32;//定点小数。高32bit 为整数，低32bit 为小数
config.info.fb.flags = DISP_BF_NORMAL;
config.info.fb.scan = DISP_SCAN_PROGRESSIVE;
config.info.alpha_mode = 2; //global pixel alpha
config.info.alpha_value = 0xff;//global alpha value
config.info.screen_win.x = 0;
config.info.screen_win.y = 0;
config.info.screen_win.width = width;
config.info.screen_win.height= height;
config.info.id = 0;
arg[0] = 0;//screen 0
arg[1] = (unsigned long)&config;
arg[2 = 1; //one layer
ret = ioctl(disphd, DISP_LAYER_SET_CONFIG, (void*)arg);
```

8.2.2 DISP_LAYER_GET_CONFIG

- PROTOTYPE

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

- ARGUMENTS

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

arg[1]为图层配置参数指针；
arg[2]为需要获取配置的图层数目；

- RETURNS

如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。

- DESCRIPTION

该函数用于获取图层参数。

- DEMO

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

8.2.3 DISP_LAYER_SET_CONFIG2

- PROTOTYPE

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

- ARGUMENTS

```
hdle 显示驱动句柄；
cmd DISP_SET_LAYER_CONFIG2
arg arg[0]为显示通道0/1；
arg[1]为图层配置参数(disp_layer_config2)的指针；
arg[2]为需要配置的图层数目
```

- RETURNS

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

- DESCRIPTION

```
该函数用于设置多个图层信息，注意该接口只接受disp_layer_config2 的信息。
```

- DEMO

```
struct
{
    disp_layer_info info,
    bool enable;
    unsigned int channel,
```

```
    unsigned int layer_id,
}disp_layer_config2;
//设置图层参数，disphd 为显示驱动句柄
unsigned long arg[3];
struct disp_layer_config2 config;
unsigned int width = 1280;
unsigned int height = 800;
unsigned int ret = 0;
memset(&config, 0, sizeof(struct disp_layer_config2));
config.channnel = 0;//blending channel
config.layer_id = 0;//layer index in the blending channel
config.info.enable = 1;
config.info.mode = LAYER_MODE_BUFFER;
config.info.fb.addr[0] = (unsigned long long)mem_in; //FB 地址
config.info.fb.size[0].width = width;
config.info.fb.align[0] = 4;//bytes
config.info.fb.format = DISP_FORMAT_ARGB_8888; //DISP_FORMAT_YUV420_P
config.info.fb.crop.x = 0;
config.info.fb.crop.y = 0;
config.info.fb.crop.width = ((unsigned long)width) << 32;//定点小数。高32bit 为整数，低32bit 为小数
config.info.fb.crop.height= ((uunsigned long)height)<<32;//定点小数。高32bit 为整数，低32bit 为小数
config.info.fb.flags = DISP_BF_NORMAL;
config.info.fb.scan = DISP_SCAN_PROGRESSIVE;
config.info.fb.eotf = DISP_EOTF_SMPTE2084; //HDR
config.info.fb.metadata_buf = (unsigned long long)mem_in2;
config.info.alpha_mode = 2; //global pixel alpha
config.info.alpha_value = 0xff;//global alpha value
config.info.screen_win.x = 0;
config.info.screen_win.y = 0;
config.info.screen_win.width = width;
config.info.screen_win.height= height;
config.info.id = 0;
arg[0] = 0;//screen 0
arg[1] = (unsigned long)&config;
arg[2 = 1; //one layer
ret = ioctl(disphd, DISP_LAYER_SET_CONFIG2, (void*)arg);
```

8.2.4 DISP_LAYER_GET_CONFIG2

- PROTOTYPE

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

- ARGUMENTS

hdle 显示驱动句柄；
cmd DISP_LAYER_GET_CONFIG2
arg arg[0]为显示通道0/1；
arg[1]为图层配置参数(disp_layer_config2)的指针；
arg[2]为需要获取配置的图层数目；

- RETURNS

如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。

- DESCRIPTION

该函数用于获取图层参数。

- DEMO

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

8.3 capture interface

8.3.1 DISP_CAPTURE_START

- PROTOTYPE

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

- ARGUMENTS

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

- RETURNS

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

- DESCRIPTION

```
该函数启动截屏功能。
```

- DEMO

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

8.3.2 DISP_CAPTURE_COMMIT

- PROTOTYPE

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

- ARGUMENTS

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

- RETURNS

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

- DESCRIPTION

```
该函数提交截屏信息，提交后才走在启动截屏功能。
```

- DEMO

```
//提交截屏功能，disphd 为显示驱动句柄
unsigned long arg[3];
struct disp_capture_info info;
arg[0] = 0;
screen_width = ioctl(disphd, DISP_GET_SCN_WIDTH, (void*)arg);
screen_height = ioctl(disphd, DISP_GET_SCN_HEIGHT, (void*)arg);
info.window.x = 0;
```

```
info.window.y = 0;
info.window.width = screen_width;
info.window.y = screen_height;
info.out_frame.format = DISP_FORMAT_ARGB_8888;
info.out_frame.size[0].width = screen_width;
info.out_frame.size[0].height = screen_height;
info.out_frame.crop.x = 0;
info.out_frame.crop.y = 0;
info.out_frame.crop.width = screen_width;
info.out_frame.crop.height = screen_height;
info.out_frame.addr[0] = fb_address; //buffer address
arg[0] = 0;//显示通道0
arg[1] = (unsigned long)&info;
ioctl(disphd, DISP_CAPTURE_COMMIT, (void*)arg);
```

8.3.3 DISP_CAPTURE_STOP

- PROTOTYPE

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

- ARGUMENTS

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

- RETURNS

如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。

- DESCRIPTION

该函数停止截屏功能。

- DEMO

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

8.3.4 DISP_CAPTURE_QUERY

- PROTOTYPE

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

- ARGUMENTS.

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

- RETURNS

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

- DESCRIPTION

```
该函数查询刚结束的图像帧是否截屏成功。
```

- DEMO

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

8.4 LCD Interface

8.4.1 DISP_LCD_SET_BRIGHTNESS

- PROTOTYPE

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

- ARGUMENTS

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

- RETURNS

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

- DESCRIPTION

```
该函数用于设置LCD 亮度。
```

- DEMO

```
//设置LCD 的背光亮度，disphd 为显示驱动句柄
unsigned long arg[3];
unsigned int bl = 197;
arg[0] = 0;//显示通道0
arg[1] = bl;
ioctl(disphd, DISP_LCD_SET_BRIGHTNESS, (void*)arg);
```

8.4.2 DISP_LCD_GET_BRIGHTNESS

- PROTOTYPE

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

- ARGUMENTS

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

- RETURNS

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

- DESCRIPTION

```
该函数用于获取LCD 亮度。
```

- DEMO

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

8.4.3 DISP_LCD_SET_GAMMA_TABLE

- PROTOTYPE

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

- ARGUMENTS

```
hdle 显示驱动句柄；
cmd DISP_LCD_SET_GAMMA_TABLE
arg arg[0]为显示通道0/1；
arg[1]为gamma table 的首地址
arg[2]为gamma table 的size，字节为单位，建议为1024，不能超过这个值。
```

- RETURNS

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

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

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

8.4.4 DISP_LCD_GAMMA_CORRECTION_ENABLE

- PROTOTYPE

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

- ARGUMENTS

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

- RETURNS

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

- DESCRIPTION

```
该函数用于使能lcd 的gamma 校正功能。
```

- DEMO

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

8.4.5 DISP_LCD_GAMMA_CORRECTION_DISABLE

- PROTOTYPE

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

- ARGUMENTS

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

- RETURNS

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

- DESCRIPTION

```
该函数用于关闭lcd 的gamma 校正功能。
```

- DEMO

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

8.5 smart backlight

8.5.1 DISP_SMBL_ENABLE

- PROTOTYPE

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

- ARGUMENTS

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

- RETURNS

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

- DESCRIPTION

```
该函数用于使能智能背光功能。
```

- DEMO

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

8.5.2 DISP_SMBL_DISABLE

- PROTOTYPE

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

- ARGUMENTS

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

- RETURNS

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

- DESCRIPTION

```
该函数用于关闭智能背光功能。
```

- DEMO

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

8.5.3 DISP_SMBL_SET_WINDOW

- PROTOTYPE

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

- ARGUMENTS

```
hdle 显示驱动句柄；
cmd DISP_SMBL_SET_WINDOW
arg arg[0]为显示通道0/1；
arg[1]为指向struct disp_rect 的指针
- RETURNS
如果成功，则返回DIS_SUCCESS；如果失败，则返回失败号。
```

- DESCRIPTION

```
该函数用于设置智能背光开启效果的窗口，智能背光在设置的窗口中有效。
```

- DEMO

```
//设置智能背光窗口，disphd 为显示驱动句柄
unsigned long arg[3];
unsigned int screen_width, screen_height;
struct disp_rect window;
screen_width = ioctl(disphd, DISP_GET_SCN_WIDTH, (void*)arg);
screen_height = ioctl(disphd, DISP_GET_SCN_HEIGHT, (void*)arg);
window.x = 0;
window.y = 0;
window.width = screen_width / 2;
widnow.height = screen_height;
arg[0] = 0;//显示通道0
arg[1] = (unsigned long)&window;
ioctl(disphd, DISP_SMBL_SET_WINDOW, (void*)arg);
```

8.6 hdmi interface

8.6.1 DISP_HDMI_SUPPORT_MODE

- PROTOTYPE

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

- ARGUMENTS

```
hdle 显示驱动句柄；
cmd DISP_HDMI_SUPPORT_MODE
arg arg[0]为显示通道0/1；
arg[1]为需要查询的模式，详见disp_tv_mode
```

- RETURNS

```
如果支持，则返回1；如果失败，则返回0。
```

- DESCRIPTION

```
该函数用于查询指定的HDMI 模式是否支持。
```

- DEMO

```
//查询指定的HDMI 模式是否支持
unsigned long arg[3];
arg[0] = 0;//显示通道0
arg[1] = (unsigned long)DISP_TV_MOD_1080P_60HZ;
ioctl(disphd, DISP_HDMI_SUPPORT_MODE, (void*)arg);
```

8.6.2 DISP_HDMI_GET_HPD_STATUS

- PROTOTYPE

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

- ARGUMENTS

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

- RETURNS

```
如果HDMI 插入，则返回1；如果未插入，则返回0。
```

- DESCRIPTION

该函数用于指定HDMI 是否处于插入状态。

- DEMO

```
//查询HDMI 是否处于插入状态
unsigned long arg[3];
arg[0] = 0;//显示通道0
if (ioctl(disphd, DISP_HDMI_GET_HPD_STATUS, (void*)arg) == 1)
    printf(“hdmi plug in\n”);
else
    printf(“hdmi plug out\n”);
```

8.6.3 DISP_HDMI_GET_EDID

- PROTOTYPE

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

- ARGUMENTS

```
hdle 显示驱动句柄；
cmd DISP_HDMI_GET_EDID
arg arg[0]为显示通道0/1；
arg[1]为保存EDID 数据的buffer 地址
arg[2]为buffer 的大小，单位：byte
```

- RETURNS

返回EDID 的长度，如果buffer 的长度比1024 大，则返回1024，否则返回buffer 的长度。

- DESCRIPTION

该函数用于获取EDID 的信息。

- DEMO

```
//获取HDMI 的EDID 信息
unsigned long arg[3];
unsigned char buf[1024];
int ret;
arg[0] = 0;//显示通道0
arg[1] = （unsigned long)buf;
arg[2] = 1024;
if(ret = ioctl(disphd, DISP_HDMI_GET_EDID, (void*)arg)) == 0) {
      printf(“get edid fail\n”);
}
```

9. sysfs 接口描述

以下两个函数在下面接口的 demo 中会使用到。

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

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

9.1 enhance

9.1.1 enhance_mode

- SYSFS NODE

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

- ARGUMENTS

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

- RETURNS

```
no
```

- DESCRIPTION

```
该接口用于设置色彩增强的模式
```

- DEMO

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

```
echo 0 > /sys/class/disp/disp/attr/disp;
echo 3 > /sys/class/disp/disp/attr/enhance_mode;
c/c++代码：
char sysfs_path[MAX_LENGTH];
char sysfs_data[MAX_DATA];
unisgned int disp = 0
unsigned int enhance_mode = 1;
snprintf(sysfs_path,sizeof(sysfs_full_path),"sys/class/disp/disp/attr/disp");
snprintf(sysfs_data, sizeof(sysfs_data),"%d",disp);
write_data(sysfs_path, sys_data, strlen(sysfs_data));
snprintf(sysfs_path,sizeof(sysfs_full_path),
"/sys/class/disp/disp/attr/enhance_mode");
snprintf(sysfs_data, sizeof(sysfs_data), "%d",enhance_mode);
write_data(sysfs_path, sys_data, strlen(sysfs_data));
```

9.1.2 enhance_bright/contrast/saturation/edge/detail/denoise

- SYSFS NODE

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

- ARGUMENTS

```
disp display channel，比如0: disp0, 1：disp1
enhance_xxx： 范围：0~100，数据越大，调节幅度越大。
```

- RETURNS

no

- DESCRIPTION

该接口用于设置图像的亮度/对比度/饱和度/边缘锐度/细节增强/降噪的调节幅度。

- DEMO

```
//设置disp0 的图像亮度为80
echo 0 > /sys/class/disp/disp/attr/disp;
echo 80 > /sys/class/disp/disp/attr/enhance_bright;
//设置disp1 的饱和度为50
echo 1 > /sys/class/disp/disp/attr/disp;
echo 50 > /sys/class/disp/disp/attr/enhance_saturation;
c/c++代码：
char sysfs_path[MAX_LENGTH];
char sysfs_data[MAX_DATA];
unisgned int disp = 0
unsigned int enhance_bright = 80;
snprintf(sysfs_path,sizeof(sysfs_full_path),"sys/class/disp/disp/attr/disp");
snprintf(sysfs_data, sizeof(sysfs_data),"%d",disp);
write_data(sysfs_path, sys_data, strlen(sysfs_data));
snprintf(sysfs_path,sizeof(sysfs_full_path),
"/sys/class/disp/disp/attr/enhance_bright");
snprintf(sysfs_data, sizeof(sysfs_data), "%d",enhance_bright);
write_data(sysfs_path, sys_data, strlen(sysfs_data));
```

9.2 hdmi edid

9.2.1 edid

- SYSFS NODE

```
/sys/class/hdmi/hdmi/attr/edid
```

- ARGUMENTS

```
no
```

- RETURNS

```
Edid data(1024 bytes)
```

- DESCRIPTION

```
该接口用于读取EDID 的裸数据
```

- DEMO

```
//读取edid 数据
cat /sys/class/hdmi/hdmi/attr/edid
c/c++代码：
#define EDID_MAX_LENGTH 1024
char sysfs_path[MAX_LENGTH];
char sysfs_data[EDID_MAX_LENGTH];
ssize_t edid_length;
snprintf(sysfs_path,sizeof(sysfs_full_path),"/sys/class/hdmi/hdmi/attr/edid");
edid_length = read_data(sysfs_path, sys_data);
```

9.2.2 hpd

- SYSFS NODE

```
/sys/class/switch/hdmi/state
```

- ARGUMENTS

```
no
```

- RETURNS

```
Hdmi hotplut state, 0: unplug; 1: plug in
```

- DESCRIPTION

```
该接口用于读取HDMI 的热插拔状态
```

- DEMO

```
//读取HDMI 热插拔状态
cat /sys/class/switch/hdmi/state
c/c++代码：
char sysfs_path[MAX_LENGTH];
char sysfs_data[MAX_DATA];
int hpd;
snprintf(sysfs_path,sizeof(sysfs_full_path),"/sys/class/hdmi/hdmi/attr/edid");
read_data(sysfs_path, sys_data);
hpd = atoi(sys_data);
```

```
If (hpd)
    printf("hdmi plug in\n");
else
    printf("hdmi unplug \n");
```

9.2.3 hdcp_enable

- SYSFS NODE

```
/sys/class/hdmi/hdmi/attr/hdcp_enable
```

- ARGUMENTS

```
@enable: 0: disable hdmi hdcp function；1：enable hdmi hdcp function
```

- RETURNS

```
No returns.
```

- DESCRIPTION

```
该接口用于使能、关闭hdmi hdcp 功能
```

- DEMO

```
//开启hdmi hdcp 功能
echo 1 > /sys/class/hdmi/hdmi/attr/hdcp_enable
//关闭hdmi hdcp 功能
echo 0 > /sys/class/hdmi/hdmi/attr/hdcp_enable
c/c++代码：
char sysfs_path[MAX_LENGTH];
char sysfs_data[MAX_DATA];
snprintf(sysfs_path,sizeof(sysfs_full_path),"/sys/class/hdmi/hdmi/attr/hdcp_enable");
snprintf(sysfs_data, sizeof(sysfs_data),"%d",1);
write_data(sysfs_path, sys_data, strlen(sysfs_data));
```

10. Data Structure

10.1 disp_fb_info

- PROTOTYPE

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

- MEMBERS

```
addr :frame buffer 的内容地址，对于interleaved 类型，只有addr[0]
有效；planar 类型，三个都有效；UV combined 的类型addr[0],addr[1]有效
size :size of framebuffer,单位为pixel
align : 对齐位宽，为2 的指数
format :pixel format,详见disp_pixel_format
color_space :color space mode,详见disp_cs_mode
b_trd_src: 1:3D source; 0：2D source
trd_mode :source 3D mode, 详见disp_3d_src_mode
trd_right_addr :used when in frame packing 3d mode
crop :用于显示的buffer 裁减区
flags ： 标识2D 或3D 的buffer
scan :标识描述类型，progress, interleaved
```

- DESCRIPTION

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

10.2 disp_layer_info

- PROTOTYPE

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

- MEMBERS

```
mode :图层的模式，详见disp_layer_mode
zorder :layer zorder,优先级高的图层可能会覆盖优先级低的图层;
alpha_mode :0:pixel alpha, 1:global alpha, 2:global pixel alpha
alpha_value :layer global alpha value，valid while alpha_mode(1/2)
screenn_win :screen window，图层在屏幕上显示的矩形窗口
fb :framebuffer 的属性，详见disp_fb_info,valid when BUFFER_MODE
color :display color, valid when COLOR_MODE
b_trd_out :if output in 3d mode,used for scaler layer
out_trd_mode:output 3d mode,详见disp_3d_out_mode
```

id :frame id,
设置给驱动的图像帧号，可以通过DISP_LAYER_GET_FRAME_ID获取当前显示的帧号，以做一下特定的处理，比如释放掉已经显示完成的图像帧bu

- DESCRIPTION

disp_layer_info 用于描述一个图层的属性信息。

10.3 disp_layer_config

- PROTOTYPE

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

- MEMBERS

info :图像的信息属性
enable :使能标志
channel :图层所在的通道id（0/1/2/3）
layer_id :图层的id，此id是在通道内的图层id。即(channel,layer_id)=(0,0)表示通道0中的图层0之意。

- DESCRIPTION

disp_layer_config 用于描述一个图层配置的属性信息。

10.4 disp_layer_config2

- PROTOTYPE

```
/\* disp_fb_info2 - image buffer info v2
\*
\* @addr: buffer address for each plane
\* @size: size<width,height> for each buffer, unit:pixels
\* @align: align for each buffer, unit:bytes
\* @format: pixel format
\* @color_space: color space
\* @trd_right_addr: the right-eye buffer address for each plane,
\* valid when frame-packing 3d buffer input
\* @pre_multiply: indicate the pixel use premultiplied alpha
\* @crop: crop rectangle for buffer to be display
\* @flag: indicate stereo/non-stereo buffer
\* @scan: indicate interleave/progressive scan type, and the scan order
\* @metadata_buf: the phy_address to the buffer contained metadata for
fbc/hdr
\* @metadata_size: the size of metadata buffer, unit:bytes
\* @metadata_flag: the flag to indicate the type of metadata buffer
\* 0 : no metadata
\* 1 << 0: hdr static metadata
\* 1 << 1: hdr dynamic metadata
\* 1 << 4: frame buffer compress(fbc) metadata
\* x : all type could be "or" together
\*/
struct disp_fb_info2 {
    unsigned long long addr[3];
    struct disp_rectsz size[3];
    unsigned int align[3];
    enum disp_pixel_format format;
    enum disp_color_space color_space;
    unsigned int trd_right_addr[3];
    bool pre_multiply;
    struct disp_rect64 crop;
    enum disp_buffer_flags flags;
    enum disp_scan_flags scan;
    enum disp_eotf eotf;
    unsigned long long metadata_buf;
```

```
        unsigned int metadata_size;
        unsigned int metadata_flag;
};
/\* disp_layer_info2 - layer info v2
\*
\* @mode: buffer/clolor mode, when in color mode, the layer is widthout buffer
\* @zorder: the zorder of layer, 0~max-layer-number
\* @alpha_mode:
\* 0: pixel alpha;
\* 1: global alpha
\* 2: mixed alpha, compositing width pixel alpha before global alpha
\* @alpha_value: global alpha value, valid when alpha_mode is not pixel alpha
\* @screen_win: the rectangle on the screen for fb to be display
\* @b_trd_out: indicate if 3d display output
\* @out_trd_mode: 3d output mode, valid when b_trd_out is true
\* @color: the color value to be display, valid when layer is in color mode
\* @fb: the framebuffer info related width the layer, valid when in buffer mode
\* @id: frame id, the user could get the frame-id display currently by
\* DISP_LAYER_GET_FRAME_ID ioctl
\* @atw: asynchronous time wrap information
\*/
struct disp_layer_info2 {
        enum disp_layer_mode mode;
        unsigned char zorder;
        unsigned char alpha_mode;
        unsigned char alpha_value;
        struct disp_rect screen_win;
        bool b_trd_out;
        enum disp_3d_out_mode out_trd_mode;
        union {
                unsigned int color;
                struct disp_fb_info2 fb;
        };
        unsigned int id;
        struct disp_atw_info atw;
};
/\* disp_layer_config2 - layer config v2
\*
\* @info: layer info
\* @enable: indicate to enable/disable the layer
\* @channel: the channel index of the layer, 0~max-channel-number
\* @layer_id: the layer index of the layer widthin it's channel
\*/

struct disp_layer_config2 {
        struct disp_layer_info2 info;
        bool enable;
        unsigned int channel;
        unsigned int layer_id;
};
```

- MEMBERS

这里仅介绍相对disp_layer_config 新增的成员
format： 数据宽度会在format 中体现出来
atw： 异步时移信息，详细见struct disp_atw_info
eotf: : 光电转换特性信息，HDR 图像时需要，定义见disp_eotf
metadata_buf:指向携带metadata 的buffer 的地址
metadata_size:metadata buffer 的大小
metadata_flag:标识metadata buffer 中携带的信息类型

- DESCRIPTION

disp_layer_config2 用于描述一个图层配置的属性信息，与disp_layer_config 的差别在于支持的功能更多，支持ATW/FBD/HDR 功能。该结构体只能使用DISP_LAYER_SET_CONFIG2命令接口。

10.5 disp_color_info

- PROTOTYPE

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

- MEMBERS

alpha :颜色的透明度
red :红
green :绿
blue :蓝

- DESCRIPTION

disp_color_info 用于描述一个颜色的信息。

10.6 disp_rect

- PROTOTYPE

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

- MEMBERS

x :起点x 值
y :起点y 值
width :宽
height :高

- DESCRIPTION

disp_rect 用于描述一个矩形窗口的信息。

10.7 disp_rect64

- PROTOTYPE

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

- MEMBERS

x :起点x 值, 定点小数，高32bit 为整数，低32bit 为小数
y :起点y 值, 定点小数，高32bit 为整数，低32bit 为小数
width :宽, 定点小数，高32bit 为整数，低32bit 为小数
height :高, 定点小数，高32bit 为整数，低32bit 为小数

- DESCRIPTION

disp_rect64 用于描述一个矩形窗口的信息。

10.8 disp_position

- PROTOTYPE

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

- MEMBERS

```
x :x
y :y
```

- DESCRIPTION

```
disp_position 用于描述一个坐标的信息。
```

10.9 disp_rectsz

- PROTOTYPE

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

- MEMBERS

width :宽
height :高

- DESCRIPTION

disp_rectsz 用于描述一个矩形尺寸的信息。

10.10 disp_atw_info

- PROTOTYPE

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

```
    unsigned int b_col;
    unsigned long cof_addr;
};
```

- MEMBERS

```
used :是否开启
mode :ATW 的模式，左右或上下模式
b_row :宏块的行数
b_col :宏块的列数
cof_addr ： ATW 系数buffer 的地址
```

- DESCRIPTION

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

10.11 disp_pixel_format

- PROTOTYPE

```
typedef enum
{
    DISP_FORMAT_ARGB_8888 = 0x00,//MSB A-R-G-B LSB
    DISP_FORMAT_ABGR_8888 = 0x01,
    DISP_FORMAT_RGBA_8888 = 0x02,
    DISP_FORMAT_BGRA_8888 = 0x03,
    DISP_FORMAT_XRGB_8888 = 0x04,
    DISP_FORMAT_XBGR_8888 = 0x05,
    DISP_FORMAT_RGBX_8888 = 0x06,
    DISP_FORMAT_BGRX_8888 = 0x07,
```

```
    DISP_FORMAT_RGB_888 = 0x08,
    DISP_FORMAT_BGR_888 = 0x09,
    DISP_FORMAT_RGB_565 = 0x0a,
    DISP_FORMAT_BGR_565 = 0x0b,
    DISP_FORMAT_ARGB_4444 = 0x0c,
    DISP_FORMAT_ABGR_4444 = 0x0d,
    DISP_FORMAT_RGBA_4444 = 0x0e,
    DISP_FORMAT_BGRA_4444 = 0x0f,
    DISP_FORMAT_ARGB_1555 = 0x10,
    DISP_FORMAT_ABGR_1555 = 0x11,
    DISP_FORMAT_RGBA_5551 = 0x12,
    DISP_FORMAT_BGRA_5551 = 0x13,

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

- MEMBERS

```
DISP_FORMAT_ARGB_8888： 32bpp, A 在最高位，B 在最低位
DISP_FORMAT_YUV420_P： planar yuv 格式，分三块存放，需三个地址，P3 在最
高位。
DISP_FORMAT_YUV422_SP_UVUV：semi-planar yuv 格式，分两块存放，需两个地址，
UV 的顺序为U 在低位， DISP_FORMAT_YUV420_SP_UVUV 类似
DISP_FORMAT_YUV422_SP_VUVU：semi-planar yuv 格式，分两块存放，需两个地址，
UV 的顺序为V 在低位， DISP_FORMAT_YUV420_SP_VUVU 类似
```

- DESCRIPTION

```
disp_pixel_format 用于描述像素格式。
```

10.12 disp_data_bits

- PROTOTYPE

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

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

10.13 disp_eotf

- PROTOTYPE

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

- DESCRIPTION

```
disp_eotf 用于描述图像的光电转换特性。
```

10.14 disp_buffer_flags

- PROTOTYPE

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

```
    DISP_BF_STEREO_LI = 1 << 4,//stereo line interlace
}disp_buffer_flags;
```

- MEMBERS

```
DISP_BF_NORMAL : 2d
DISP_BF_STEREO_TB： top bottom 模式
DISP_BF_STEREO_FP : framepacking
DISP_BF_STEREO_SSF : side by side full,左右全景
DISP_BF_STEREO_SSH : side by side half,左右半景
DISP_BF_STEREO_LI : line interleaved,行交错模式
```

- DESCRIPTION

```
disp_buffer_flags 用于描述3D 源模式。
```

10.15 disp_3d_out_mode

- PROTOTYPE

```
typedef enum
{
    //for lcd
    DISP_3D_OUT_MODE_CI_1 = 0x5,//column interlaved 1
    DISP_3D_OUT_MODE_CI_2 = 0x6,//column interlaved 2
    DISP_3D_OUT_MODE_CI_3 = 0x7,//column interlaved 3
    DISP_3D_OUT_MODE_CI_4 = 0x8,//column interlaved 4
    DISP_3D_OUT_MODE_LIRGB = 0x9,//line interleaved rgb
    //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;
```

- MEMBERS

```
//for lcd
DISP_3D_OUT_MODE_CI_1 : 列交织
DISP_3D_OUT_MODE_CI_2 : 列交织
DISP_3D_OUT_MODE_CI_3 : 列交织
DISP_3D_OUT_MODE_CI_4 : 列交织
DISP_3D_OUT_MODE_LIRGB : 行交织
//for hdmi
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
```

- DESCRIPTION

```
disp_3d_out_mode 用于描述3D 输出模式。
```

10.16 disp_color_space

- PROTOTYPE

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

- MEMBERS

```
DISP_BT601：用于标清视频，SDR 模式
DISP_BT709：用于高清视频，SDR 模式
DISP_BT2020NC：用于HDR 模式
```

- DESCRIPTION

```
disp_color_space 用于描述颜色空间类型。
```

10.17 disp_csc_type

- PROTOTYPE

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

- DESCRIPTION

```
disp_csc_type 用于描述图像颜色格式。
```

10.18 disp_output_type

- PROTOTYPE

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

- MEMBERS

```
DISP_OUTPUT_TYPE_NONE： 无显示输出
DISP_OUTPUT_TYPE_LCD： LCD 输出
DISP_OUTPUT_TYPE_TV： TV 输出
```

DISP_OUTPUT_TYPE_HDMI： HDMI 输出
DISP_OUTPUT_TYPE_VGA： VGA 输出

- DESCRIPTION

disp_output_type 用于描述显示输出类型。

10.19 disp_tv_mode

- PROTOTYPE

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

```
    DISP_TV_MOD_PAL_NC_SVIDEO = 0x15,
    DISP_TV_MOD_3840_2160P_30HZ = 0x1c,
    DISP_TV_MOD_3840_2160P_25HZ = 0x1d,
    DISP_TV_MOD_3840_2160P_24HZ = 0x1e,
    DISP_TV_MODE_NUM = 0x1f,
}disp_tv_mode;
```

- DESCRIPTION

```
disp_tv_mode 用于描述TV 输出模式。
```

10.20 disp_output

- PROTOTYPE

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

- MEMBERS

```
Type:输出类型
Mode:输出模式，480P/576P, etc.
```

- DESCRIPTION

disp_output 用于描述显示输出类型，模式

10.21 disp_layer_mode

- PROTOTYPE

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

- MEMBERS

LAYER_MODE_BUFFER: buffer 模式，带buffer 的图层
LAYER_MODE_COLOR: 单色模式，无buffer 的图层，只需要一个颜色值表示图像内容

- DESCRIPTION

disp_layer_mode 用于描述图层模式。

10.22 disp_scan_flags

- PROTOTYPE

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

- MEMBERS

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

- DESCRIPTION

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

10.23 disp_device_config

- PROTOTYPE

```
/\* disp_device_config - display deivce config
\*
\* @type: output type
\* @mode: output mode
\* @format: data format
\* @bits: data bits
\* @eotf: electro-optical transfer function
\* SDR : DISP_EOTF_GAMMA22
\* HDR10: DISP_EOTF_SMPTE2084
```

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

- MEMBERS

```
type: 设备类型，如HDMI/TV/LCD 等
mode: 分辨率
format: 输出的数据格式，比如RGB/YUV444/422/420
bits: 输出的数据位宽，8/10/12/16bits
eotf: 光电特性信息
cs: 输出的颜色空间类型
```

- DESCRIPTION

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

10.24 disp_device_config

- PROTOTYPE

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

- MEMBERS

```
type: 设备类型，如HDMI/TV/LCD 等
mode: 分辨率
format: 输出的数据格式，比如RGB/YUV444/422/420
bits: 输出的数据位宽，8/10/12/16bits
eotf: 光电转换特性信息
cs: 输出的颜色空间类型
```

- DESCRIPTION

disp_device_config 用于描述输出设备的属性信息。

11. 调试

11.1 查看显示模块的状态

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

PAD 方案状态示例如下：

```
# cat /sys/class/disp/disp/attr/sys
screen 0: /* 0 路显示通道*/
de_rate 297000000 Hz /* de 的时钟频率*/, ref_fps=60 /* 输出设备的参考刷新率*/
lcd output backlight( 61) fps:60.6 1280x 800 /* 屏输出| 背光值（61）| 屏刷新率:60.6Hz | 分辨率为：1280x800 */
err: 0 /* de 缺数的次数*/ skip: 1 /* de 跳帧的次数*/ irq:1350 /* tcon 中断的次数*/ vsync:601 /* 已发送的vsync 消息个数*/
BUF enable ch[0] lyr[0] z[0] prem[Y] a[globl 255] fmt[ 0] fb[1280, 800;1280, 800;1280, 800] crop[ 0, 0,1280, 800] frame[ 0, 0,1280, 800]
addr[60fa1000, 0, 0] flags[0x 0] trd[0,0] /* 图层信息*/
screen 1: /* 1 路显示通道*/
de_rate 297000000 Hz /* de 的时钟频率*/, ref_fps=60 /* 输出设备的参考刷新率*/
hdmi output mode(10) fps:60.6 1920x1080 /* HDMI 输出| 模式为（10：1080P@60Hz）| 刷新率为：60.6Hz | 分辨率为：1920x1080 */
err: 0 /* de 缺数的次数*/ skip: 2 /* de 跳帧的次数*/ irq:270 /* 中断的次数*/ vsync:206 /* 已发送的vsync 消息个数*/
BUF enable ch[0] lyr[0] z[0] prem[Y] a[globl 255] fmt[ 0] fb[1920,1080;1920,1080;1920,1080] crop[ 0, 0,1920,1080] frame[ 0, 0,1920,1080]
addr[63b11000, 0, 0] flags[0x 0] trd[0,0] /* 图层信息*/
acquire: 292, 60.0 fps hwc 送下来的帧数，帧率
release: 291, 59.1 fps de 返还的帧数，帧率
display: 287, 59.5 fps de 显示的帧数，帧率
```

BOX 方案状态示例如下：

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

acquire: 225, 2.6 fps
release: 224, 2.6 fps
display: 201, 2.5 fps

图层各信息描述如下：

BUF: 图层类型，BUF/COLOR，一般为BUF，即图层是带BUFFER 的。COLOR 意思是显示一个纯色的画面，不带BUFFER。
enable: 显示处于enable 状态
ch[0]: 该图层处于blending 通道0
lyr[0]: 该图层处于当前blending 通道中的图层0
z[0]: 图层z 序，越小越在底部，可能会被z 序大的图层覆盖住
prem[Y]: 是否预乘格式，Y 是，N 否
a: alpha 参数， globl/pixel/; alpha 值
fmt: 图层格式，值64 以下为RGB 格式；以上为YUV 格式，常见的72 为YV12,76 为NV12
fb: 图层buffer 的size，width,height，三个分量
crop: 图像buffer 中的裁减区域，[x,y,w,h]
frame: 图层在屏幕上的显示区域，[x,y,w,h]
addr: 三个分量的地址
flags: 一般为0， 3D SS 时0x4, 3D TB 时为0x1, 3D FP 时为0x2;
trd: 是否3D 输出，3D 输出的类型（HDMI FP 输出时为1）各counter 描述如下：
err： de 缺数的次数，de 缺数可能会出现屏幕抖动，花屏的问题。de 缺数一般为带宽不足引起。
skip: 表示de 跳帧的次数，跳帧会出现卡顿问题。跳帧是指本次中断响应较慢，de
模块判断在本次中断已经接近或者超过了消隐区，将放弃本次更新图像的机会，选择继续显示原有的图像。
irq: 表示该通路上垂直消隐区中断执行的次数，一直增长表示该通道上的timing
controller 正在运行当中。
vsync:表示显示模块往用户空间中发送的vsync 消息的数目，一直增长表示正在不断地发送中。

acquire/release/display 含义如下,只在android 方案中有效：
acquire： 是hw composer 传递给disp driver 的图像帧数以及帧率，帧率只要有在有图像更新时才有效，静止时的值是不准确的
release: 是disp driver 显示完成之后，返还给android 的图像帧数以及帧率，帧率只要有在有图像更新时才有效，静止时的值是不准确的
display： 是disp 显示到输出设备上的帧数以及帧率，帧率只要有在有图像更新时才有效，静止时的值是不准确的如果acquire 与release
不一致，说明disp 有部分图像帧仍在使用，未返还，差值在1~2
之间为正常值。二者不能相等，如果相等，说明图像帧全部返还，显示将会出
现撕裂现象。如果display 与release 不一致，说明在disp 中存在丢帧情况，原因为在一个active 区内hwcomposer
传递多于一帧的图像帧下来

调试说明：

1. 对于android 系统，可以dumpsys SurfaceFlinger 打印surface 的信息，如果信息与disp 中sys 中的信息不一致，很大可能是hwc
的转换存在问题。
2. 如果发现图像刷新比较慢，存在卡顿问题，可以看一下输出设备的刷新率，对比一下ref_fps 与fps 是否一致，如果不一致，说明tcon
的时钟频率或timing 没配置正确。如果ref_fps 与屏的spec 不一致，则需要检查sys_config
中的时钟频率和timing配置是否正确。屏一般为60Hz，而如果是TV 或HDMI，则跟模式有关，比较常见的为60/50/30/24Hz。
如果是android 方案，还可以看一下display 与release 的counter 是否一致，如果相差太大，说明android 送帧不均匀，造成丢帧。
3. 如果发现图像刷新比较慢，存在卡顿问题，也需要看一下skip counter，如果skip

counter 有增长，说明现在的系统负荷较重，对vblank 中断的响应较慢，出现跳帧，导致了图像卡顿问题。
4. 如果屏不亮，怀疑背光时，可以看一下屏的背光值是否为0。
如果为0，说明上层传递下来的背光值不合理；如果不为0，背光还是不亮，则为驱动或硬件问题了。硬件上可以通过测量bl_en 以及pwm 的电压值来排查问题。
5. 如果花屏或图像抖动，可以查看err counter，如果err counter 有增长，则说明de缺数，有可能是带宽不足，或者瞬时带宽不足问题。

11.2 查看 hdmi 热插拔状态

```
cat /sys/class/switch/hdmi/state
```

0：表示 unplug 1：表示 plug in

11.3 hdmi 软件模拟热插拔

```
echo 0xnm > /sys/class/hdmi/hdmi/attr/hpd_mask
n：0：表示使用硬件的真实状态； 1：表示使用软件模拟的状态
m：表示软件模拟的状态，0：拔出；1：插入
/* 以下表示模拟拔出状态，而不管硬件真实状态*/
# echo 0x10 > /sys/class/hdmi/hdmi/attr/hpd_mask
/* 以下表示模拟插入状态，而不管硬件真实状态*/
# echo 0x11 > /sys/class/hdmi/hdmi/attr/hpd_mask
/* 以下表示真实反应硬件真实状态*/
# echo 0x0m > /sys/class/hdmi/hdmi/attr/hpd_mask
```

模拟插入或拔出信号发出后，驱动以及用户空间都会将其当成真实状态在运行。即模拟拔出时，驱动会认为 HDMI 线是拔出的，用户空间也会收到 HDMI 拔出的消息。

11.4 hdmi 模块的 debug 开关

```
/* 打开debug 开关*/
# echo on > /sys/class/hdmi/hdmi/attr/debug
/* 关闭debug 开关*/
# echo off > /sys/class/hdmi/hdmi/attr/debug
```

打开开关后，驱动流程会有 debug 信息打印出来，如下所示：

```
[[ 1638.504349] [HDMI] plugin
[ 1638.710091] [HDMI] HDMI_State_EDID_Parse
[ 1638.714576] [HDMI] ParseEDID
[ 1638.717857] [HDMI] DDC_Read
[ 以下为BANK0 的数据。如果出现EDID block 0 checksum error 或者EDID block0 header error 的打印说明DDC 通信失败]
[ 1638.855238] [HDMI] Sink : EDID bank 0:
[ 1638.859405] [HDMI] 0 1 2 3 4 5 6 7 8 9 A B C D E F
[ 1638.876511] [HDMI] ================================================================================
[ 1638.888208] [HDMI] 00 ff ff ff ff ff ff 00 22 45 9b 06 01 00 00 00
[ 1638.896637] [HDMI] 0f 17 01 03 80 ba 69 78 0a ee 91 a3 54 4c 99 26
[ 1638.905104] [HDMI] 0f 50 54 bd c8 00 81 00 81 40 81 80 95 00 a9 40
[ 1638.913557] [HDMI] 01 01 01 01 01 01 04 74 00 30 f2 70 5a 80 b0 58
[ 1638.922033] [HDMI] 8a 00 44 17 74 00 00 1e 02 3a 80 18 71 38 2d 40
[ 1638.930459] [HDMI] 58 2c 45 00 a0 5a 00 00 00 1e 00 00 00 fd 00 18
[ 1638.938868] [HDMI] 4b 1a 51 1e 00 0a 20 20 20 20 20 20 00 00 00 fc
[ 1638.947308] [HDMI] 00 48 41 49 45 52 20 34 4b 32 4b 55 48 44 01 ea
[ 1638.955741] [HDMI] ================================================================================
[ 1638.967353] [HDMI] EDID version: 1.3
[ 1638.971238] [HDMI] PCLK=297000000 Xsize=3840 Ysize=2160 Frame_rate=30
[ 1638.978616] [HDMI] PCLK=148500000 Xsize=1920 Ysize=1080 Frame_rate=60
[ 1638.985800] [HDMI] DDC_Read
[ 以下为BANK1 的数据。如果出现EDID block 1 checksum error或者EDID block1 header error 的打印说明DDC 通信失败]
[ 1639.117296] [HDMI] Sink : EDID bank 1:
[ 1639.121490] [HDMI] 0 1 2 3 4 5 6 7 8 9 A B C D E F
[ 1639.129800] [HDMI] ================================================================================
[ 1639.141423] [HDMI] 02 03 27 f1 4b 90 9f 04 13 05 14 03 12 20 21 22
[ 1639.149834] [HDMI] 23 09 07 07 83 01 00 00 6e 03 0c 00 30 00 b8 3c
[ 1639.158264] [HDMI] 20 80 80 01 02 03 04 01 1d 00 bc 52 d0 1e 20 b8
[ 1639.166699] [HDMI] 28 55 40 c4 8e 21 00 00 1e 02 3a 80 d0 72 38 2d
[ 1639.175120] [HDMI] 40 10 2c 45 80 a0 5a 00 00 00 1f 01 1d 80 d0 72
[ 1639.183557] [HDMI] 1c 16 20 10 2c 25 80 a0 5a 00 00 00 9f 00 00 00
[ 1639.191975] [HDMI] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[ 1639.200442] [HDMI] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 8f
[ 1639.208851] [HDMI] ================================================================================
[ 1639.220477] [HDMI] device support YCbCr44 output
[ 1639.225605] [HDMI] Parse_VideoData_Block: VIC 16(native) support
[ 1639.232304] [HDMI] Parse_VideoData_Block: VIC 31(native) support
[ 1639.238983] [HDMI] Parse_VideoData_Block: VIC 4 support
[ 1639.244813] [HDMI] Parse_VideoData_Block: VIC 19 support
[ 1639.250731] [HDMI] Parse_VideoData_Block: VIC 5 support
[ 1639.256535] [HDMI] Parse_VideoData_Block: VIC 20 support
```

```
[ 1639.262457] [HDMI] Parse_VideoData_Block: VIC 3 support
[ 1639.268256] [HDMI] Parse_VideoData_Block: VIC 18 support
[ 1639.274204] [HDMI] Parse_VideoData_Block: VIC 32 support
[ 1639.280129] [HDMI] Parse_VideoData_Block: VIC 33 support
[ 1639.286029] [HDMI] Parse_VideoData_Block: VIC 34 support
[ 1639.291937] [HDMI] Parse_AudioData_Block: max channel=2
[ 1639.297769] [HDMI] Parse_AudioData_Block: SampleRate code=7
[ 1639.303987] [HDMI] Parse_AudioData_Block: WordLen code=7
[ 1639.309883] [HDMI] Find HDMI Vendor Specific DataBlock
[ 1639.315598] [HDMI] 3D_present [ 支持3D 模式]
[ 1639.318888] [HDMI] Parse_HDMI_VSDB: VIC 1 support [ 支持4K 显示, 3840x2160@30Hz ]
[ 1639.324148] [HDMI] Parse_HDMI_VSDB: VIC 2 support [ 支持4K 显示, 3840x2160@25Hz ]
[ 1639.329366] [HDMI] Parse_HDMI_VSDB: VIC 3 support [ 支持4K 显示, 3840x2160@24Hz ]
[ 1639.334600] [HDMI] Parse_HDMI_VSDB: VIC 4 support [ 支持4K 显示, 4096x2160@24Hz ]
[ 1639.339821] [HDMI] PCLK=74250000 Xsize=1280 Ysize=720 Frame_rate=50
[ 1639.346835] [HDMI] PCLK=148500000 Xsize=1920 Ysize=1080 Frame_rate=50
[ 1639.354026] [HDMI] PCLK=74250000 Xsize=1920 Ysize=540 Frame_rate=50
[ 1639.361147] [HDMI] switch_set_state 1 [ 往switch 结点汇报插入消息（1）；如果是拔出则汇报拔出消息（0） ]
[ 1639.363495] [HDMI] set_video_enable = 1!
[ 1639.363506] [HDMI] video_on @ set_video_enable = 0!
[ cts_enable：表示cts_compatible 配置选项的值； isHDMI 为1 表明当前连接的显示器是HDMI 设备，0 则为DVI 设备；YCbCr444_Support 为1
    表示当前连接的显示器支持ycbcr 输入;hdcp_enable 表示hdmi_hdcp_enable 配置选项的值，1 表示需要开启hdcp 功能, 0
    表示不需要开启hdcp 功能]
[ 1639.363515] [HDMI] video_config, vic:19,cts_enable:0,isHDMI:1,YCbCr444_Support:1,hdcp_enable:0
[ 1639.363527] [HDMI] hdmi video + audio
[ 1639.363533] [HDMI] video_on @ video_config = 0!
[ is_hdmi:1 表示hdmi 控制器输出hdmi 模式，0 表示输出dvi 模式；is_yuv：1 表示hdmi 控制器输出yuv 格式，0 则为rgb格式；is_hcts：1
    表示hdmi 控制器开启hdcp 功能，0 表示关闭hdcp 功能]
[ 1639.363540] [HDMI] set_video_enable, vic:19,is_hdmi:1,is_yuv:1,is_hcts:0
```

11.5 hdmi edid 数据

```
# /sys/class/hdmi/hdmi/attr/edid
```

只读，大小是 1024 bytes。

11.6 显示模块 debugfs 接口

11.6.1 总述

```
目录：
# /sys/kernel/debug/dispdbg;
/* mount debugfs */
# mount -t debugfs none /sys/kernek/debug;
/* 结点*/
# ls
# name command param start info
/* name: 表示操作的对象名字
command: 表示执行的命令
param: 表示该命令接收的参数
start: 输入1 开始执行命令
info: 保存命令执行的结果
*/
只读，大小是1024 bytes。
```

11.6.2 切换显示输出设备

```
name: disp0/1/2 //表示显示通道0/1/2
command: switch
param: type mode
参数说明：type:0(none),1(lcd),2(tv),4(hdmi),8(vga)
mode 详见disp_tv_mode 定义
例子：
/* 显示通道0 输出HDMI 720P@50Hz */
echo disp0 > name;echo switch > command;echo 4 4 > param;echo 1 > start;
/* 显示通道0 输出LCD */
echo disp0 > name;echo switch > command;echo 1 0 > param;echo 1 > start;
/* 显示通道1 输出HDMI 1080@60Hz */
echo disp1 > name;echo switch > command;echo 4 10 > param;echo 1 > start;
/* 关闭显示通道0 的输出*/
echo disp0 > name;echo switch > command;echo 0 0 > param;echo 1 > start;
```

11.6.3 开关显示输出设备

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

11.6.4 电源管理 (suspend/resume) 接口

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

11.6.5 调节 lcd 屏幕背光

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

11.6.6 vsync 消息开关

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

11.6.7 查看 enhance 的状态

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

11.6.8 查看智能背光的状态

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

11.6.9 查看 hdmi 电视对分辨率的支持情况

```
name: hdmi0/1/2 //表示hdmi0/1/2
command: is_support //获取smart backlight 的状态
param: x
info: y
参数说明：x: 分辨率，详见disp_tv_mod
返回值说明：y:0（表示不支持），1（表示支持）
例子：
/* 查看hdmi 电视对720P@50Hz 分辨率是否支持*/
echo hdmi0 > name;echo is_support > command;echo 4 > param;echo 1 > start;cat
info;
/* 查看hdmi 电视对1080P@60Hz 分辨率是否支持*/
echo hdmi0 > name;echo is_support > command;echo 10 > param;echo 1 >
start;cat info;
```

11.7 查看 android 帧率

```
# setprop debug.hwc.showfps 1
# logcat -s hwcomposer
结果示例如下：
D/hwcomposer( 1551): >>>fps:: 40
D/hwcomposer( 1551): >>>fps:: 57
D/hwcomposer( 1551): >>>fps:: 21
```

11.8 查看 android 图层信息

```
# setprop debug.hwc.showfps 2
# logcat -s hwcomposer
```

结果示例如下：

```
+Disp info:
+ Fram Num|DP|Vsy|STP|CA(U)|Ad|IST|FPS (20,40,60)|MAX_LMTD|Cur_LMTD|TR_LMTD
      |All_Used|DSP_Used|CHNNEL_0|CHNNEL_1|CHNNEL_2|CHNNEL_3|timestamp
+---------+--+---+---+-----+--+---+-------------+--------+--------+--------+--------+--------+--------+--------+--------+--------+----------+
+ 4322| 0|Yes| No|1 (1)| 2|Yes|58.9,59.1,59.3|37324800|37324800| 2280000| 6021120| 6021120| 2170880| 3850240| 0| 0|0
+---------+--+---+---+-----+--+---+-------------+--------+--------+--------+--------+--------+--------+--------+--------+--------+----------+
+
+Type|CH|V|SC H|SC W|PL|S| Handle | Phyaddr | Usage | Flags |Tr|Bld| Format | Source Crop | Frame Crop | Reason
+----+--+-+----+----+--+-+----------------+----------------+--------+--------+--+---+---------+----------------------+----------------------+-------------
+ HWC| 0|N|1.00|1.00|ff|0| 0000000081457e00| 0000000072300000| 00000f02| 00000000|00|100| 00000004|[ 0, 0, 752, 1280]|[ 48, 0, 800,
      1280]|I_OVERLAY
+ HWC| 1|N|1.00|1.00|ff|0| 0000000081457aa0| 0000000073a00000| 00000f02| 00000000|00|105| 00000001|[ 0, 0, 752, 1280]|[ 48, 0, 800,
      1280]|I_OVERLAY
+ HWC| 0|N|1.00|1.00|ff|0| 0000000081dce500| 00000000700c0000| 00000f02| 00000000|00|105| 00000001|[ 0, 0, 48, 1280]|[ 0, 0, 48,
      1280]|I_OVERLAY
+ FB|-2|N|0.00|0.00|ff|0| 0000000081c4bfa0| 0000000070c00000| 00001e02| 00000000|00|105| 00000005|[ 0, 0, 800, 1280]|[ 0, 0, 800,
      1280]|NOT_ASSIGNED
+----+--+-+----+----+--+-+----------------+----------------+--------+--------+--+---+---------+----------------------+----------------------+-------------
```

各字段表示的意义如下：

字段	意义
Fram Num	帧号，从 0 开始，一直往上增长
DP	真实的 display 屏幕，0/1/..
Vsy	hw vsync 是否开启
STP	是否 stop hwc，如果 stop，所有的 surface 都会走 GPU 合成通道
CA(U)	为存储图层信息而分配的链表结点数目，（U）表示已经使用的个数，剩下的为空闲的个数
Ad	链表结点中由于空闲个数过多，将要舍弃释放的结点个数，不包含在（U）之中。
IST	是否是在稳定期，在唤醒后的前 60 帧以及从静止转为动态的前 60 帧为非稳定期，该值为 0。主要是休眠或者静态画面后 ddr 会动态调整至较低的频率，如果一下子开放所有的 DE 图层，可能会导致瞬间带宽的不足，所以在非稳定期，会限制一部分的 DE 图层资源。
FPS (20,40,60)	每 20 帧计算的帧率，该值用于计算当前的分配策略下的功耗以及全部使用 GPU 合成的功耗数据，取功耗较小的合成方式，以节省功耗。
MAX_LMTD	DE 可用的带宽限制（包含所有的显示屏幕），如果送给 DE 的图层消耗带宽超出该限制，可能会出现显示异常情况
Cur_LMTD	当前屏幕下的带宽限制，如果送给 DE 该屏幕的图层消耗带宽超出该限制，可能会出现显示异常情况
TR_LMTD	rotate 硬件的带宽限制

字段	意义
All_Used	当前场景下 DE 总共消耗的带宽（包含所有的显示屏幕）
DSP_Used	当前屏幕下 DE 消耗的带宽
CHNNEL_0~3	channel0 到 channe3 消耗的带宽
timestamp	当前最新的 hw vysnc 触发的时间戳

图层信息

字段	意义
Type	``GLES": GPU 合成,``HWC": DE 合成, ``BKGD": 背景, ``FB": framebuffer target,``SIDE":（暂未使用）,``CURS": 硬件鼠标;
CH	使用的 blending channel 索引号
V	是否是 video 图像，以格式（yuv）判断
SC	H height 的放大倍数
SC	W width 的放大倍数
PL	plane alpha，面 alpha
S	是否需要 sync memory，是否需要刷 cpu 的 cache
Handle	buffer 的 handle，与 sf 传递下来的 Handle 一致
Phyaddr	该 buffer 的物理地址
Usage	handle 的 usage
Flags	layer 的 flag，与 sf 传递下来的 flag 一致
Tr	是否带旋转，旋转的角度是多少，0 表示不旋转，1：水平翻转，2：垂直翻转，3：180 度旋转，4：90 度旋转，7：270 度旋转
Bld	alpha blending，与 sf 传递下来的 blend 一致，0x100：HWC_BLENDING_NONE，0x105:HWC_BLENDING_PREMULT，0x405: HWC_BLENDING_COVERAGE
	Format 图像的颜色格式
Source Crop	图像源的裁减区，只有裁减区内的内容会显示在屏幕上，
Frame Crop	图像源的裁减区显示在屏幕上的矩形窗口，x_left, y_top, x_right, y_bottom

不能使用 DE 合成的原因:

I_OVERLAY(HWC 合成)
D_NULL_BUF（handle 是null）
D_CONTIG_MEM（非连续内存）
D_VIDEO_PD（视频保护）
D_CANNT_SCALE（由于不支持的图像格式）
D_SKIP_LAYER（surfaceflinger 不让DE 合成）
D_NO_FORMAT（DE 不支持的格式）
D_BACKGROUND（由于是背景）
D_TR_N_0（没有旋转硬件模块）
D_ALPHA（不支持的alpha blending）
D_X_FB（跟FB 相交）
D_SW_OFTEN（由于CPU 频繁读写）
D_SCALE_OUT（scale 能力不够导致）
D_STOP_HWC（通过系统属性设置停止使用HWC）
D_NO_PIPE（没有blending channel 导致）
D_NO_MEM（DE 运算能力不足）
D_MEM_CTRL（GPU 和DE 合成的能耗控制导致，如果GPU 合成功耗较低，则走GPU 合成）

12. Declaration

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