

## **HiFB**

# API 参考

文档版本 06

发布日期 2009-09-30

BOM编码 N/A

深圳市海思半导体有限公司为客户提供全方位的技术支持,用户可与就近的海思办事处联系,也可直接与公 司总部联系。

## 深圳市海思半导体有限公司

地址: 深圳市龙岗区坂田华为基地华为电气生产中心 邮编: 518129

网址: http://www.hisilicon.com

客户服务电话: +86-755-28788858

客户服务传真: +86-755-28357515

客户服务邮箱: support@hisilicon.com

#### 版权所有 © 深圳市海思半导体有限公司 2007-2009。保留一切权利。

非经本公司书面许可,任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部,并不得以任何形式 传播。

#### 商标声明



(上) 、HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标,由各自的所有人拥有。

#### 注意

由于产品版本升级或其他原因,本文档内容会不定期进行更新。除非另有约定,本文档仅作为使用指导, 本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。



# 目 录

刊リ	₽		I
1	概述		1-1
	1.1	概述	1-1
	1.2	2 参考域说明	1-1
		1.2.1 API参考域	1-1
		1.2.2 数据类型参考域	1-2
2 /	API参	<del>发</del> 专	2-1
	2.1	API类别	2-1
	2.2	2 ioctl函数	2-1
	2.3	3 标准功能	2-3
		2.3.1 FBIOGET VSCREENINFO	
		2.3.2 FBIOPUT_VSCREENINFO	2-4
		2.3.3 FBIOGET_FSCREENINFO	2-6
		2.3.4 FBIOPAN_DISPLAY	2-7
		2.3.5 FBIOGETCMAP	2-9
		2.3.6 FBIOPUTCMAP	2-11
	2.4	上扩展功能	2-12
		2.4.1 FBIOGET_CAPABILITY_HIFB	2-12
		2.4.2 FBIOGET_SCREEN_ORIGIN_HIFB	2-13
		2.4.3 FBIOPUT_SCREEN_ORIGIN_HIFB	2-14
		2.4.4 FBIOGET_SHOW_HIFB	2-15
		2.4.5 FBIOPUT_SHOW_HIFB	2-16
		2.4.6 FBIOGET_COLORKEY_HIFB	2-17
		2.4.7 FBIOPUT_COLORKEY_HIFB	2-18
		2.4.8 FBIOGET_ALPHA_HIFB	2-19
		2.4.9 FBIOPUT_ALPHA_HIFB	2-20
		2.4.10 FBIOGET_DEFLICKER_HIFB	2-21
		2.4.11 FBIOPUT_DEFLICKER_HIFB	2-23
		2.4.12 FBIOGET_VBLANK_HIFB	2-24
		2.4.13 FBIOFLIP_SURFACE	
		2.4.14 FBIOGET_CURSOR_HI3511	2-26



	2.4.15 FBIOPUT_CURSOR_HI3511	2-27
	2.5 错误码	2-31
3	数据类型	3-1
	3.1 在标准中定义的数据类型	3-1
	3.1.1 struct fb_bitfield	3-1
	3.1.2 struct fb_var_screeninfo	3-2
	3.1.3 struct fb_fix_screeninfo	3-4
	3.1.4 struct fb_cmap	3-6
	3.2 扩展的数据类型	3-7
	3.2.1 HIFB_COLOR_FMT_E	3-7
	3.2.2 HIFB_CAPABILITY_S	3-8
	3.2.3 HIFB_POINT_S	3-9
	3.2.4 HIFB_COLORKEY_S	3-10
	3.2.5 HIFB_ALPHA_S	3-12
	3.2.6 HIFB_DEFLICKER_S	3-13
	3.2.7 HIFB_SURFACE_S	3-14
	3.2.8 HIFB_YCBCR_S	3-15
	3.2.9 HI3511_CURSOR_E	3-15
	3.2.10 HI3511_CURSOR_S	3-16
4	图形开发辅助接口	4-1
	4.1 概述	4-1
	4.1.1 简介	4-1
	4.1.2 注意事项	4-2
	4.2 API参考	4-4
	4.2.1 VD_SET_GRAPHIC_BIND	4-4
	4.2.2 VD_GET_GRAPHIC_BIND	
	4.2.3 VD_SET_DEV_ATTR	4-7
	4.2.4 VD_GET_DEV_ATTR	4-8
	4.2.5 VD_ENABLE_DEV	4-9
	4.2.6 VD_DISABLE_DEV	4-9
	4.3 数据结构	4-10
	4.3.1 VO_PUB_ATTR_S	4-10
	4.3.2 VD_PUB_ATTR_IOC_S	4-14
	4.3.3 VD_PUB_ATTR_IOC_S	4-14



# 插图目录

图 2-1 设置从虚拟分辨率中的不同偏移处开始显示	2-8
图 4-1 Hi3520 视频输出单元基本结构	4-2



# 表格目录

表 1-1 API参考域说明	1-1
表 1-2 数据类型参考域说明	1-2
表 2-1 ioctl函数的 3 个参数	2-2
表 2-2 错误码	2-31
表 4-1 图形层与输出设备的对应关系表	4-3



# 前言

# 概述

本节介绍本文档的内容、对应的产品版本、适用的读者对象、行文表达约定、历史修订记录等。

# 产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3510 通信媒体处理器	V100
Hi3511 H.264 编解码处理器	V100
Hi3512 H.264 编解码处理器	V100
Hi3520 H.264 编解码处理器	V100

# 读者对象

本文档(本指南)主要适用于以下工程师:

- 电子产品设计维护人员
- 电子产品元器件市场销售人员

# 内容简介

本文档介绍了 Hisilicon Framebuffer(以下简称 HiFB)涉及的 API 函数及数据类型。全书共分为 3 章。

章节	内容
1 概述	简单介绍了关于 HiFB 及 API 参考域和数据类型参考域的说明。
2 API 参考	介绍 HiFB 提供的基于 Linux Framebuffer 的基本功能和扩展功能。

1



章节	内容
3 数据类型	介绍 HiFB 中涉及到标准定义和扩展的数据类型。

# 约定

## 通用格式约定

格式	说明
宋体	正文采用宋体表示。
黑体	一级、二级、三级标题采用黑体。
楷体	警告、提示等内容一律用楷体,并且在内容前后增加线条与正文隔离。
"Terminal Display"格式	"Terminal Display"格式表示屏幕输出信息。此外,屏幕输出信息中夹杂的用户从终端输入的信息采用加粗字体表示。

## 命令行格式约定

格式	意义
粗体	命令行关键字(命令中保持不变、必须照输的部分)采用加粗字体表示。
斜体	命令行参数(命令中必须由实际值进行替代的部分)采用 斜体表示。
[]	表示用"[]"括起来的部分在命令配置时是可选的。
{ x   y   }	表示从两个或多个选项中选取一个。
[x y ]	表示从两个或多个选项中选取一个或者不选。
{ x   y   } *	表示从两个或多个选项中选取多个,最少选取一个,最多 选取所有选项。
[ x   y   ] *	表示从两个或多个选项中选取多个或者不选。

# 表格内容约定

内容	说明
-	表格中的无内容单元。
*	表格中的内容用户可根据需要进行配置。



# 修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2009-09-30	06	● 增加第4章 图形开发辅助接口。
2009-09-03	05	• 3.2.4 HIFB_COLORKEY_S 的【注意】中增加 Hi3520不支持带Alpha的colorkey 的说明。
		• 3.2.5 HIFB_ALPHA_S 的【注意】中增加 Hi3520 芯 片叠加Alpha值的计算公式。
2008-08-22	04	• 增加关于 Hi3512 芯片的说明。
2008-07-30	03	<ul> <li>增加 2.4.13~2.4.15 三个小节来介绍新增接口函数。</li> <li>修改 3.2.4 中【注意】中的描述。</li> <li>增加 3.2.7~3.2.10 四个小节来介绍新增的数据类型。</li> </ul>
2008-02-20	02	<ul> <li>增加 2.4.12 中【注意】中说明。</li> <li>修改 3.1.2 中【成员】pixclock 的描述中单位由"1E-9 秒"改为"ns"。</li> <li>修改 3.2.2 中【注意】中的描述。</li> <li>将文档中关于参数 fd 的描述都由"Framebuffer 设备号"改为"Framebuffer 设备文件描述符"。</li> </ul>
2007-11-30	01	第一次发布。



**1** 概述

# 1.1 概述

Hisilicon Framebuffer(以下简称 HiFB)是海思数字媒体处理平台提供的管理图像叠加层的模块,它基于 Linux Framebuffer 实现,在提供 Linux Framebuffer 基本功能的基础上,还扩展了一些图层控制功能,如层间 Alpha、层间 colorkey 等。

# 1.2 参考域说明

## 1.2.1 API 参考域

本手册使用 9 个参考域描述 API 的相关信息,它们的作用如表 1-1 所示。

表1-1 API 参考域说明

参考域	含义
目的	简要描述 API 的主要功能。
语法	列出调用 API 应包括的头文件以及 API 的原型声明。
参数	列出 API 的参数、参数说明及参数属性。
描述	简要描述 API 的工作过程。
返回值	列出 API 所有可能的返回值及其含义。
需求	列出 API 包含的头文件和 API 编译时要链接的库文件。
注意	列出使用 API 时应注意的事项。
举例	列出使用 API 的实例。
相关接口	列出与本 API 相关联的其他接口。



# 1.2.2 数据类型参考域

本手册使用 5 个参考域描述数据类型的相关信息,它们的作用如表 1-2 所示。

#### 表1-2 数据类型参考域说明

参考域	含义
说明	简要描述数据类型的主要功能。
定义	列出数据类型的定义语句。
成员	列出数据结构的成员及含义。
注意事项	列出使用数据类型时应注意的事项。
相关数据类型和接口	列出与本数据类型相关联的其他数据类型和接口。



# **2** API 参考

# 2.1 API 类别

HiFB 的 API 分为以下几类:

• 文件操作类

提供操作 HiFB 的接口。通过调用这些接口,可以像操作文件一样操作叠加层。这些接口是 Linux 本身提供的标准接口,主要有 open、close、write、read、lseek等。本文档不对这些标准接口进行描述。

● 显存映射类

提供将物理显存映射到用户虚拟内存空间的接口。这些接口是 Linux 本身提供的标准接口,主要有 mmap、munmap 等。本文档不对这些标准接口进行描述。

• 显存控制和状态查询类

允许设置象素格式和颜色深度等属性的接口。这些接口是 Linux 本身提供的标准接口,经常使用。本文档将对其进行简要描述。

• 层间效果控制和状态查询类

HiFB 可以管理多个图形叠加层,每层可以设置 Alpha 值、colorkey 值和原点等。 相对于 Linux Framebuffer,这些是 HiFB 的新增功能。本文档将重点描述该部分。

## 2.2 ioctl 函数

HiFB 的用户态接口以 ioctl 形式体现,其形式如下:

该函数是 Linux 标准接口,具备可变参数特性。但在 HiFB 中,实际只需要 3 个参数。 因此,其语法形式等同于:



CMD\_DATA\_TYPE \*cmddata);

其中,CMD\_DATA\_TYPE 随参数 cmd 的变化而变化。这 3 个参数的详细描述如表 2-1 所示。

#### 表2-1 ioctl 函数的 3 个参数

参数名称	描述	输入/ 输出
fd	Framebuffer 设备文件描述符,是调用 open 函数打开 Framebuffer 设备之后的返回值。	输入
cmd	主要的 cmd(命令控制字)如下:  FBIOGET_VSCREENINFO: 获取屏幕可变信息 FBIOPUT_VSCREENINFO: 读取屏幕可变信息 FBIOGET_FSCREENINFO: 获取屏幕固定信息 FBIOPAN_DISPLAY: 设置 PAN 显示 FBIOGETCMAP: 获取调色板 FBIOGET_CAPABILITY_HIFB: 获取叠加层的支持能力 FBIOGET_CAPABILITY_HIFB: 获取叠加层坐标原点 FBIOPUT_SCREEN_ORIGIN_HIFB: 设置叠加层坐标原点 FBIOPUT_SCREEN_ORIGIN_HIFB: 设置叠加层业标原点 FBIOPUT_SHOW_HIFB: 设置叠加层显示状态 FBIOPUT_SHOW_HIFB: 设置叠加层。colorkey FBIOPUT_COLORKEY_HIFB: 设置叠加层。colorkey FBIOPUT_COLORKEY_HIFB: 设置叠加层和层。colorkey FBIOPUT_ALPHA_HIFB: 设置叠加层和自由。 FBIOPUT_ALPHA_HIFB: 设置叠加层的抗闪烁设置 FBIOPUT_DEFLICKER_HIFB: 获取叠加层的抗闪烁功能 FBIOGET_VBLANK_HIFB: 获取叠加层的垂直消隐区 FBIOFLIP_SURFACE: 实现多个 Surface 交替显示并设置 alpha 和 colorkey 属性 FBIOGET_CURSOR_HI3511: 获取硬件鼠标层的属性	输入
	• FBIOPUT_CURSOR_HI3511: 设置硬件鼠标层属性	



参数名称	描述	输入/ 输出
cmddata	各 cmd 对应的数据类型分别是:	输入
	• 获取或设置屏幕可变信息: struct fb_var_screeninfo *类型	输出
	• 获取屏幕固定信息: struct fb_fix_screeninfo *类型	
	● 设置 PAN 显示: struct fb_var_screeninfo *类型	
	• 获取或设置调色板类型: struct fb_cmap *类型	
	• 获取叠加层支持能力: HIFB_CAPABILITY_S *类型	
	• 获取或设置屏幕叠加层坐标原点: HIFB_POINT_S *类型	
	• 获取或设置叠加层显示状态: HI_BOOL *类型	
	• 获取或设置叠加层 colorkey: HIFB_COLORKEY_S *类型	
	● 获取或设置叠加层 Alpha: HIFB_ALPHA_S *类型	
	• 获取或设置叠加层的抗闪烁功能: HIFB_DEFLICKER_S *类型	
	• 实现多个 Surface 交替显示并设置 alpha 和 colorkey 属性: HIFB_SURFACE_S *类型	
	• 获取或设置 Hi3511/Hi3512 硬件鼠标层属性: HI3511_CURSOR_S *类型	

# 2.3 标准功能

# 2.3.1 FBIOGET\_VSCREENINFO

#### 【目的】

获取屏幕的可变信息。

#### 【语法】

#### 【描述】

使用此接口获取屏幕的可变信息,主要包括分辨率和象素格式。信息的详细描述请参见"3.1.2 struct fb\_var\_screeninfo"。

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入



参数名称	描述	输入/输出
FBIOGET_VSCREENINFO	ioctl 号	输入
var	可变信息结构体指针	输出

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: fb.h

#### 【注意】

系统默认的分辨率为720×576,象素格式为ARGB1555。

#### 【举例】

```
struct fb_var_screeninfo vinfo;
if (ioctl(fd, FBIOGET_VSCREENINFO, &vinfo) < 0)
{
    return -1;
}</pre>
```

#### 【相关接口】

FBIOPUT\_VSCREENINFO

# 2.3.2 FBIOPUT\_VSCREENINFO

#### 【目的】

设置 Framebuffer 的屏幕分辨率和象素格式等。

#### 【语法】

#### 【描述】

使用此接口设置屏幕分辨率、象素格式。



参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_VSCREENINFO	ioctl 号	输入
var	可变信息结构体指针	输入

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: fb.h

#### 【注意】

分辨率的大小必须在各叠加层支持的分辨率范围内,各叠加层支持的最大分辨率和最小分辨率可通过 FBIOGET\_CAPABILITY\_HIFB 获取。

必须保证实际分辨率与偏移的和在虚拟分辨率范围内,否则系统会自动调整实际分辨率的大小让其在虚拟分辨率范围内。

#### 【举例】

设置实际分辨率为  $720 \times 576$ ,虚拟分辨率为  $720 \times 576$ ,偏移为 (0, 0),象素格式为 ARGB8888 的示例代码如下:

```
struct fb_bitfield r32 = {16, 8, 0};
struct fb_bitfield g32 = {8, 8, 0};
struct fb_bitfield b32 = {0, 8, 0};
struct fb_bitfield a32 = {24, 8, 0};
struct fb_bitfield a32 = {24, 8, 0};
struct fb_var_screeninfo vinfo;
if (ioctl(fd, FBIOGET_VSCREENINFO, &vinfo) < 0)
{
    return -1;
}
vinfo.xres_virtual = 720;
vinfo.yres_virtual = 576;
vinfo.xres = 720;
vinfo.yres = 576;
vinfo.activate = FB_ACTIVATE_NOW;
vinfo.bits_per_pixel = 32;
vinfo.xoffset = 0;</pre>
```



```
vinfo.yoffset = 0;
vinfo.red = r32;
vinfo.green = g32;
vinfo.blue = b32;
vinfo.transp= a32;
if (ioctl(fd, FBIOPUT_VSCREENINFO, &vinfo) < 0)
{
    return -1;
}
```

#### 【相关接口】

FBIOGET\_VSCREENINFO

## 2.3.3 FBIOGET\_FSCREENINFO

#### 【目的】

获取 Framebuffer 的固定信息。

#### 【语法】

#### 【描述】

使用此接口获取 Framebuffer 固定信息,包括显存起始物理地址、显存大小和行间距等。信息的详细描述请参见"3.1.3 struct fb\_fix\_screeninfo"。

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_FSCREENINFO	ioctl 号	输入
fix	固定信息结构体指针	输出

#### 【返回值】

返回值	描述
0	成功
-1	失败

#### 【需求】



头文件: fb.h

【注意】

无。

【举例】

无。

【相关接口】

无。

## 2.3.4 FBIOPAN\_DISPLAY

#### 【目的】

设置从虚拟分辨率中的不同偏移处开始显示。

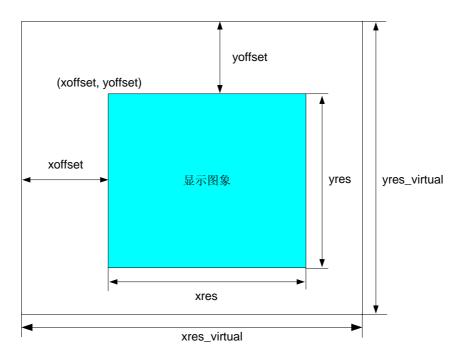
#### 【语法】

#### 【描述】

使用此接口设置从虚拟分辨率中的不同偏移处开始显示,实际的分辨率不变。如图 2-1 所示:(xres\_virtual, yres\_virtual)是虚拟分辨率,(xres, yres)是实际显示的分辨率,(xoffset, yoffset)是显示的偏移。



#### 图2-1 设置从虚拟分辨率中的不同偏移处开始显示



#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPAN_DISPLAY	ioctl 号	输入
var	可变信息结构体指针	输入

#### 【返回值】

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: fb.h

#### 【注意】

必须保证实际分辨率与偏移的和在虚拟分辨率范围内,否则设置不成功。

#### 【举例】



设置实际分辨率为  $300 \times 300$ ,虚拟分辨率为  $720 \times 576$ ,起始偏移为 (50, 50),然后 偏移到 (300, 0) 处开始显示的 PAN 设置代码如下:

```
struct fb_bitfield r32 = \{16, 8, 0\};
struct fb bitfield g32 = \{8, 8, 0\};
struct fb_bitfield b32 = \{0, 8, 0\};
struct fb bitfield a32 = \{24, 8, 0\};
struct fb_var_screeninfo vinfo;
vinfo.xres virtual = 720;
vinfo.yres_virtual = 576;
vinfo.xres = 300;
vinfo.yres = 300;
vinfo.activate = FB_ACTIVATE_NOW;
vinfo.bits per pixel = 32;
vinfo.xoffset = 50;
vinfo.yoffset = 50;
vinfo.red = r32;
vinfo.green = g32;
vinfo.blue = b32;
vinfo.transp= a32;
if (ioctl(fd, FBIOPUT VSCREENINFO, &vinfo) < 0)</pre>
   return -1;
vinfo.xoffset = 300;
vinfo.yoffset = 0;
if (ioctl(fd, FBIOPAN DISPLAY, &vinfo) < 0)</pre>
   return -1;
```

#### 2.3.5 FBIOGETCMAP

#### □ 说明

- Hi3510 芯片不支持调色板模式, 所以在 Hi3510 芯片上调用该接口返回失败。
- Hi3511 芯片不支持调色板模式,所以在 Hi3511 芯片上调用该接口返回失败。
- Hi3512 芯片不支持调色板模式,所以在 Hi3512 芯片上调用该接口返回失败。

#### 【目的】

获取调色板信息。

#### 【语法】



```
struct fb_cmap *cmap);
```

#### 【描述】

使用此接口获取调色板信息。

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGETCMAP	ioctl 号	输入
стар	调色板结构体指针	输出

#### 【返回值】

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: fb.h

#### 【注意】

无。

#### 【举例】



}

#### 【相关接口】

#### **FBIOPUTCMAP**

## 2.3.6 FBIOPUTCMAP

## □ 说明

- Hi3510 芯片不支持调色板模式, 所以在 Hi3510 芯片上调用该接口返回失败。
- Hi3511 芯片不支持调色板模式, 所以在 Hi3511 芯片上调用该接口返回失败。
- Hi3512 芯片不支持调色板模式, 所以在 Hi3512 芯片上调用该接口返回失败。

#### 【目的】

设置调色板信息。

#### 【语法】

#### 【描述】

使用此接口设置调色板信息。

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUTCMAP	ioctl 号	输入
cmap	调色板结构体指针	输入

#### 【返回值】

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: fb.h

#### 【注意】

无。



#### 【举例】

```
unsigned short Red256[256];
                              /* only low 8bits valid */
unsigned short Green256[256]; /* only low 8bits valid */
unsigned short Blue256[256]; /* only low 8bits valid */
struct fb_cmap cmap;
/* create a palette which contains 256 color */
Palette Create(Red256, Green256, Blue256);
cmap.start = 0;
cmap.len = 256;
cmap.red = Red256;
cmap.green = Green256;
cmap.blue = Blue256;
cmap.transp = 0;
if (ioctl(fd, FBIOPUTCMAP, &cmap) < 0)</pre>
   printf("fb ioctl put cmap err!\n");
   return -1;
}
```

#### 【相关接口】

**FBIOGETCMAP** 

# 2.4 扩展功能

# 2.4.1 FBIOGET\_CAPABILITY\_HIFB

#### 【目的】

获取叠加层的支持能力。

#### 【语法】

#### 【描述】

在使用某些接口前,用户可以通过调用此接口查询该叠加层是否支持该功能。



参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_CAPABILITY_HIFB	ioctl 号	输入
pstCap	支持能力结构体指针	输出

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: hifb.h

【注意】

无。

【举例】

无。

【相关接口】

无。

# 2.4.2 FBIOGET\_SCREEN\_ORIGIN\_HIFB

#### 【目的】

获取叠加层在屏幕上显示的起始点坐标。

#### 【语法】

#### 【描述】

使用此接口获取叠加层在屏幕上显示的起始点坐标。



参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_SCREEN_ORIGIN_HIFB	ioctl 号	输入
pstPoint	坐标原点结构体指针	输出

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: hifb.h

【注意】

无。

【举例】

无。

【相关接口】

FBIOPUT\_SCREEN\_ORIGIN\_HIFB

## 2.4.3 FBIOPUT\_SCREEN\_ORIGIN\_HIFB

#### 【目的】

设置叠加层在屏幕上显示的起始点坐标。

#### 【语法】

#### 【描述】

使用此接口设置叠加层在屏幕上显示的起始点坐标,坐标范围从(0,0)到该叠加层支持的最大分辨率和最小分辨率的差值(u32MaxWidth-u32MinWidth, u32MaxHeight – u32MinHeight)之间。



参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_SCREEN_ORIGIN_HIFB	ioctl 号	输入
pstPoint	坐标原点结构体指针	输入

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: hifb.h

#### 【注意】

如果叠加层坐标原点超出了范围,默认将坐标原点设置为(u32MaxWidth – u32MinWidth,u32MaxHeight –u32MinHeight)。

#### 【举例】

无。

#### 【相关接口】

FBIOGET\_SCREEN\_ORIGIN\_HIFB

## 2.4.4 FBIOGET\_SHOW\_HIFB

#### 【目的】

获取当前叠加层的显示状态。

#### 【语法】

#### 【描述】

使用此接口获取当前叠加层显示状态。



参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_SHOW_HIFB	ioctl 号	输入
bShow	指示当前叠加层的状态:	输出
	● *bShow = HI_TRUE: 当前叠加层处于 显示状态	
	● *bShow = HI_FALSE: 当前叠加层处于 隐藏状态	

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: hifb.h

【注意】

无。

【举例】

无。

【相关接口】

FBIOPUT\_SHOW\_HIFB

# 2.4.5 FBIOPUT\_SHOW\_HIFB

#### 【目的】

显示或隐藏该叠加层。

#### 【语法】

#### 【描述】

使用此接口设置叠加层显示状态:显示或隐藏。



#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_SHOW_HIFB	ioctl 号	输入
bShow	该叠加层的显示状态:	输入
	● *bShow = HI_TRUE: 显示当前叠加层	
	● *bShow = HI_FALSE: 隐藏当前叠加层	

#### 【返回值】

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: hifb.h

【注意】

无。

【举例】

无。

【相关接口】

FBIOGET\_SHOW\_HIFB

# 2.4.6 FBIOGET\_COLORKEY\_HIFB

#### 【目的】

获取叠加层的 colorkey。

#### 【语法】

#### 【描述】

使用此接口获取叠加层的 colorkey。



参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_COLORKEY_HIFB	ioctl 号	输入
pstColorKey	colorkey 结构体指针	输出

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: hifb.h

【注意】

无。

【举例】

无。

【相关接口】

FBIOPUT\_COLORKEY\_HIFB

# 2.4.7 FBIOPUT\_COLORKEY\_HIFB

#### 【目的】

设置叠加层的 colorkey。

#### 【语法】

#### 【描述】

使用此接口设置当前叠加层的 colorkey 功能。

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入



参数名称	描述	输入/输出
FBIOPUT_COLORKEY_HIFB	ioctl 号	输入
pstColorKey	colorkey 结构体指针	输入

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: hifb.h

#### 【注意】

无。

#### 【举例】

假设当前象素格式为 ARGB8888,则要过滤掉红色分量落在[0x10,0x1F]、绿色分量落在[0x20,0x2F]、蓝色分量落在[0x30,0x3F]区间的颜色值,具体设置如下:

```
HIFB_COLORKEY_S stColorKey;

stColorKey.bKeyEnable = HI_TRUE;
stColorKey.u32Key = 0x1F2F3F;
stColorKey.bMaskEnable = HI_TRUE;
stColorKey.u8BlueMask = 0x0F;
stColorKey.u8RedMask = 0x0F;
stColorKey.u8GreenMask = 0x0F;
if (ioctl(fd, FBIOPUT_COLORKEY_HIFB, &stColorKey) < 0)
{
    return -1;
}</pre>
```

#### 【相关接口】

FBIOGET\_COLORKEY\_HIFB

# 2.4.8 FBIOGET\_ALPHA\_HIFB

#### 【目的】

获取叠加层 Alpha。



#### 【语法】

#### 【描述】

使用此接口获取当前叠加层的 Alpha 设置。

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_ALPHA_HIFB	ioctl 号	输入
pstAlpha	Alpha 结构体指针	输出

#### 【返回值】

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: hifb.h

#### 【注意】

请参见 HIFB\_ALPHA\_S 的说明。

#### 【举例】

无。

#### 【相关接口】

FBIOPUT ALPHA HIFB

# 2.4.9 FBIOPUT\_ALPHA\_HIFB

#### 【目的】

设置叠加层的 Alpha。

#### 【语法】



HIFB\_ALPHA\_S \*pstAlpha);

#### 【描述】

使用此接口设置当前叠加层的 Alpha 功能。

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_ALPHA_HIFB	ioctl 号	输入
pstAlpha	Alpha 结构体指针	输入

#### 【返回值】

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: hifb.h

#### 【注意】

请参见 HIFB\_ALPHA\_S 的说明。

#### 【举例】

无。

#### 【相关接口】

FBIOGET\_ALPHA\_HIFB

## 2.4.10 FBIOGET\_DEFLICKER\_HIFB

#### □ 说明

- Hi3510 芯片不支持抗闪烁操作,所以在 Hi3510 芯片上调用该接口返回失败。
- Hi3511 芯片不支持抗闪烁操作,所以在 Hi3511 芯片上调用该接口返回失败。
- Hi3512 芯片不支持抗闪烁操作,所以在 Hi3512 芯片上调用该接口返回失败。

#### 【目的】

获取叠加层的抗闪烁设置。

#### 【语法】

int ioctl (int fd,



```
FBIOGET_DEFLICKER_HIFB,
HIFB DEFLICKER S *pstDeflicker);
```

#### 【描述】

使用此接口获取当前叠加层的抗闪烁设置。

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_DEFLICKER_HI FB	ioctl 号	输入
pstDeflicker	抗闪烁结构体指针	输出

#### 【返回值】

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: hifb.h

#### 【注意】

在获取抗闪烁参数时,必须设置能够获取抗闪烁的最大级别,且为抗闪烁系数分配足够的内存。

#### 【举例】

获取水平和垂直抗闪烁最大级别为2的示例代码如下:

```
HI_U8 u8HDefCoef;
HI_U8 u8VDefCoef;
HIFB_DEFLICKER_S stDeflicker;

stDeflicker.u32HDfLevel = 2;
stDeflicker.u32VDfLevel = 2;
stDeflicker.pu8HDfCoef = &u8HDefCoef;
stDeflicker.pu8VDfCoef = &u8VDefCoef;

if (ioctl(fd, FBIOGET_DEFLICKER_HIFB, &stDeflicker) < 0)
{
    return -1;</pre>
```



}

#### 【相关接口】

FBIOPUT\_DEFLICKER\_HIFB

## 2.4.11 FBIOPUT\_DEFLICKER\_HIFB

## □ 说明

- Hi3510 芯片不支持抗闪烁操作,所以在 Hi3510 芯片上调用该接口返回失败。
- Hi3511 芯片不支持抗闪烁操作,所以在 Hi3511 芯片上调用该接口返回失败。
- Hi3512 芯片不支持抗闪烁操作,所以在 Hi3512 芯片上调用该接口返回失败。

#### 【目的】

设置叠加层的抗闪烁功能。

#### 【语法】

#### 【描述】

使用此接口设置当前叠加层的抗闪烁功能。

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_DEFLICKER_HI FB	ioctl 号	输入
pstDeflicker	抗闪烁结构体指针	输入

#### 【返回值】

返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: hifb.h

#### 【注意】

请参见"3.2.6 HIFB\_DEFLICKER\_S"的说明。



#### 【举例】

设置水平和垂直 2 阶抗闪烁的代码如下:

```
HI_U8 u8HDefCoef = 0x80;
HI_U8 u8VDefCoef = 0x80;
HIFB_DEFLICKER_S stDeflicker;

stDeflicker.u32HDfLevel = 2;
stDeflicker.u32VDfLevel = 2;
stDeflicker.pu8HDfCoef = &u8HDefCoef;
stDeflicker.pu8VDfCoef = &u8VDefCoef;

if (ioctl(fd, FBIOPUT_DEFLICKER_HIFB, &stDeflicker) < 0)
{
    return -1;
}</pre>
```

#### 【相关接口】

FBIOGET\_DEFLICKER\_HIFB

### 2.4.12 FBIOGET\_VBLANK\_HIFB

#### □ 说明

- Hi3510 芯片不支持该操作, 所以在 Hi3510 芯片上调用该接口返回失败。
- Hi3511 芯片不支持该操作,所以在 Hi3511 芯片上调用该接口返回失败。
- Hi3512 芯片不支持该操作, 所以在 Hi3512 芯片上调用该接口返回失败。

#### 【目的】

为了操作显存时而不引起撕裂现象,一般可以在该叠加层的垂直消隐区对显存进行操作,通过该接口可以等待该叠加层垂直消隐区的到来。

#### 【语法】

#### 【描述】

使用此接口获取当前叠加层的消隐区。

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_VBLANK_HIFB	ioctl 号	输入



返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: hifb.h。

#### 【注意】

垂直消隐间隔较短,一般在几十毫秒,用户的操作时间也应该尽量的短,保证在垂直消隐区结束前完成。

#### 【举例】

无。

#### 【相关接口】

无。

## 2.4.13 FBIOFLIP\_SURFACE

#### 【目的】

实现多个 Surface 交替显示并设置 alpha 和 colorkey 属性。

#### 【语法】

#### 【描述】

此接口是 FBIOPAN\_DISPLAY 的扩展接口,用于实现多个 Surface 交替显示的同时设置 alpha 和 colorkey 属性。

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOFLIP_SURFACE	ioctl 号	输入
pstSurface	Surface 结构体指针	-

#### 【返回值】



返回值	描述
0	成功
-1	失败

#### 【需求】

头文件: hifb.h。

#### 【注意】

Surface 的物理地址必须在该叠加层配置的显存范围内。

## 【举例】

无。

#### 【相关接口】

无。

# 2.4.14 FBIOGET\_CURSOR\_HI3511

## □ 说明

此接口是 Hi3511/Hi3512 芯片特有的硬件鼠标层操作接口, 其他芯片的鼠标层使用和其他叠加层一致。

#### 【目的】

获取 Hi3511/Hi3512 芯片硬件鼠标层的属性。

#### 【语法】

## 【描述】

使用此接口获取 Hi3511/Hi3512 芯片硬件鼠标层的属性。

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_CURSOR_HI3511	ioctl 号	输入
pstHi3511Cursor	Hi3511/Hi3512 芯片硬件鼠标层属 性指针	输出



#### 【返回值】

返回值	描述
0	成功
-1	失败

## 【需求】

头文件: hifb.h

【注意】

无。

【举例】

无。

【相关接口】

FBIOPUT\_CURSOR\_HI3511

# 2.4.15 FBIOPUT\_CURSOR\_HI3511

## □ 说明

此接口是 Hi3511/Hi3512 芯片特有的硬件鼠标层操作接口, 其他芯片的鼠标层使用和其他叠加层一致。

#### 【目的】

设置 Hi3511/Hi3512 芯片硬件鼠标层的属性。

#### 【语法】

#### 【描述】

使用此接口设置 Hi3511/Hi3512 芯片硬件鼠标层的属性。

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_CURSOR_HI3511	ioctl 号	输入
pstHi3511Cursor	Hi3511/Hi3512 芯片硬件鼠标层属 性指针	输入



#### 【返回值】

返回值	描述
0	成功
-1	失败

## 【需求】

头文件: hifb.h

#### 【注意】

无。

#### 【举例】

假设硬件鼠标层的子设备号是 2,设置两色和透明模式的代码如下:

```
HI_U32 i, j;
HI_S32 s32Fd;
HI U8* pu8Screen;
HIFB_POINT_S stPoint = {10, 10};  /* cursor's position on screen */
HI3511_CURSOR_S stHi3511Cursor;
struct fb_var_screeninfo vinfo;
struct fb fix screeninfo finfo;
/* open cursor */
s32Fd = open("/dev/fb/2", O_RDWR);
if(s32Fd < 0)
   printf("Open cursor failed!\n");
   return HI_FAILURE;
if(ioctl(s32Fd, FBIOGET_VSCREENINFO, &vinfo) < 0)</pre>
   printf("Get vinfo failed!\n");
   close(s32Fd);
   return HI_FAILURE;
}
/* set cursor resolution: 32*32*2bpp */
vinfo.xres_virtual = 32;
vinfo.yres_virtual = 32;
vinfo.xres = 32;
vinfo.yres = 32;
```



```
vinfo.activate = FB_ACTIVATE_NOW;
   vinfo.bits per pixel = 2;
   vinfo.xoffset = vinfo.yoffset = 0;
   if(ioctl(s32Fd, FBIOPUT VSCREENINFO, &vinfo) < 0)</pre>
      printf("Put vinfo failed!\n");
      close(s32Fd);
      return HI_FAILURE;
   }
   /* set cursort attribution */
   stHi3511Cursor.enCursor = HI3511 CURSOR 2COLOR;
   stHi3511Cursor.stYCbCr[0].u8Cr = 226;
   stHi3511Cursor.stYCbCr[0].u8Cb = 25;
   stHi3511Cursor.stYCbCr[0].u8Y = 128;
   stHi3511Cursor.stYCbCr[1].u8Cr = 25;
   stHi3511Cursor.stYCbCr[1].u8Cb = 218;
   stHi3511Cursor.stYCbCr[1].u8Y = 225;
   stHi3511Cursor.stYCbCr[2].u8Cr = 97;
   stHi3511Cursor.stYCbCr[2].u8Cb = 186;
   stHi3511Cursor.stYCbCr[2].u8Y = 225;
   stHi3511Cursor.stYCbCr[3].u8Cr = 0;
   stHi3511Cursor.stYCbCr[3].u8Cb = 85;
   stHi3511Cursor.stYCbCr[3].u8Y = 32;
   if(ioctl(s32Fd, FBIOPUT_CURSOR_HI3511, &stHi3511Cursor) < 0)</pre>
      printf("set cursor failed!\n");
      close(s32Fd);
      return HI FAILURE;
   }
   if(ioctl(s32Fd, FBIOGET_FSCREENINFO, &finfo) < 0)</pre>
      printf("Get finfo failed!\n");
      close(s32Fd);
      return HI FAILURE;
   pu8Screen = (HI_U8*) mmap(NULL, finfo.smem_len, PROT_READ|PROT_WRITE,
MAP_SHARED, s32Fd, 0);
   if (MAP_FAILED == pu8Screen)
      printf("mmap failed!\n");
      close(s32Fd);
```



```
return HI_FAILURE;
}
/* set cursor data */
for(j = 0; j < 32; j++)
   for(i = 0; i < 8; i++)
   if(j < 8)
   *(pu8Screen+j*8 + i) = 0x00;
   else if (j < 16)
   *(pu8Screen+j*8 + i) = 0x55;
   else if(j < 24)
   *(pu8Screen+j*8 + i) = 0xaa;
   *(pu8Screen+j*8 + i) = 0xff;
/* move the cursor */
for(i = 0; i < 576; i += 8)
   stPoint.u32PosX += 8;
   stPoint.u32PosY += 8;
   if(ioctl(s32Fd, FBIOPUT_SCREEN_ORIGIN_HIFB, &stPoint) < 0)</pre>
      printf("set cursor position failed!\n");
      munmap(pu8Screen, finfo.smem_len);
      close(s32Fd);
      return HI_FAILURE;
   }
munmap(pu8Screen, finfo.smem len);
close(s32Fd);
return HI SUCCESS;
```

#### 【相关接口】

FBIOGET\_CURSOR\_HI3511



# 2.5 错误码

表 2-2 列出了当函数返回值小于 0 时有可能出现的所有错误码。这些错误码来自标准的 linux 错误码定义,详细内容请参见 linux 内核原码 errno\_base.h。错误码可以通过打印 Linux 的标准错误码 errno 查看,或者用 strerror(errno)打印错误信息。

表2-2 错误码

错误代码	宏定义	描述
1	EPERM	不支持该操作
12	ENOMEM	内存不够
14	EFAULT	传入参数指针地址无效
22	EINVAL	传入参数无效



# **3** 数据类型

# 3.1 在标准中定义的数据类型

# 3.1.1 struct fb\_bitfield

## 【说明】

位域信息,用于设置象素格式。

## 【定义】

#### 【成员】

成员名称	描述	支持情况
offset	颜色分量起始比特位。	支持。
length	颜色分量所占比特长度。	支持。
msb_right	右边的比特是否为最高有效位。	只支持该位为 0, 即最左 边的 bit 为最高有效位。

#### 【注意】

例如 ARGB1555 格式,其位域信息的赋值如下:

```
struct fb_bitfield a16 = {15, 1, 0};
struct fb_bitfield r16 = {10, 5, 0};
struct fb_bitfield g16 = {5, 5, 0};
struct fb_bitfield b16 = {0, 5, 0};
```



#### 【相关数据类型和接口】

无。

# 3.1.2 struct fb var screeninfo

#### 【说明】

可变的屏幕信息。

#### 【定义】

```
struct fb_var_screeninfo
   __u32 xres;
                              /* visible resolution */
   __u32 yres;
   __u32 xres_virtual;
                              /* virtual resolution */
   __u32 yres_virtual;
                               /* offset from virtual to visible */
   u32 xoffset;
   __u32 yoffset;
                              /* resolution */
   __u32 bits_per_pixel;
                               /* guess what */
   __u32 grayscale;
                              /* != 0 Graylevels instead of colors */
   struct fb bitfield red;
                             /* bitfield in fb mem if true color, */
   struct fb bitfield green;
                             /* else only length is significant */
   struct fb_bitfield blue;
   struct fb bitfield transp; /* transparency */
                              /* != 0 Non standard pixel format */
   u32 nonstd;
   u32 activate;
                              /* see FB ACTIVATE * */
   __u32 height;
                              /* height of picture in mm */
   __u32 width;
                              /* width of picture in mm */
   u32 accel flags;
                              /* (OBSOLETE) see fb info.flags */
   /* Timing: All values in pixclocks, except pixclock (of course) */
   __u32 pixclock;
                              /* pixel clock in ps (pico seconds) */
   __u32 left_margin;
                              /* time from sync to picture */
   __u32 right_margin;
                              /* time from picture to sync */
                              /* time from sync to picture */
   __u32 upper_margin;
   __u32 lower_margin;
                              /* length of horizontal sync */
   __u32 hsync_len;
   __u32 vsync_len;
                              /* length of vertical sync */
   __u32 sync;
                               /* see FB_SYNC_* */
```



成员名称	描述	支持情况
xres	可见屏幕宽度(象素数)。	支持,默认值为 720。
yres	可见屏幕高度(象素数)。	支持。默认为 576。
xres_virtual	虚拟屏幕宽度(显存中图像宽度),当该值小于 xres 时会修改 xres,使 xres 值与该值相等。	支持,默认为 720。
yres_virtual	虚拟屏幕高度(显存中图像高度),当该值小于 yres 时会修改 yres,使 yres 值与该值相等。结合 xres_virtual,可以用来快速水平或 垂直平移图像。	支持,默认为 576。
xoffset	在x方向上的偏移象素数。	支持,默认为0。
yoffset	在y方向上的偏移象素数。	支持,默认为0。
bits_per_pixel	每个象素所占的比特数。	支持,默认为16。
grayscale	灰度级。	不支持,缺省值为0,表示彩色。
red	颜色分量中红色的位域信息。	支持,默认为(10,5,0)。
green	颜色分量中绿色的位域信息。	支持,默认为(5,5,0)。
blue	颜色分量中蓝色的位域信息。	支持,默认为(0,5,0)。
transp	颜色分量中 alpha 分量的位域信息。	支持,默认为(15,1,0)。
nonstd	是否为标准象素格式。	不支持,缺省值为0,表 示支持标准象素格式。
activate	设置生效的时刻。	不支持,缺省值为 FB_ACTIVATE_NOW, 表示设置立刻生效。
height	屏幕高,单位为 mm。	不支持,缺省值为-1。
width	屏幕宽,单位为 mm。	不支持,缺省值为-1。



成员名称	描述	支持情况
accel_flags	加速标志。	不支持,缺省值为-1。
pixclock	显示一个点需要的时间,单位为 ns。	不支持,缺省值为-1。
left_margin	分别是左消隐信号、右消隐信号、	不支持,缺省值为-1。
right_margin	】水平同步时长,这三个值之和等于 】水平回扫时间,单位为点时钟。	
hsync_len	44. L Edd. (41.4). L EDA W. (41.1).	
upper_margin	分别是上消隐信号、下消隐信号、	不支持,缺省值为-1。
lower_margin	】垂直同步时长,这三个值之和等于 】垂直回扫时间,单位为点时钟。	
vsync_len		
sync	同步信号方式。	不支持,缺省值为-1。
vmode	扫描模式。	不支持,缺省值为-1。
rotate	顺时针旋转的角度。	不支持,缺省值为 0,表示无旋转。

系统默认的分辨率为 720×576, 象素格式为 ARGB1555。

## 【相关数据类型及接口】

- struct fb\_bitfield
- FBIOGET\_VSCREENINFO
- FBIOPUT\_VSCREENINFO

# 3.1.3 struct fb\_fix\_screeninfo

#### 【说明】

固定的屏幕信息。

#### 【定义】

```
struct fb_fix_screeninfo
{
   char id[16]; /* identification string eg "TT Builtin" */
   unsigned long smem_start; /* Start of frame buffer mem (physical address) */
   __u32 smem_len; /* Length of frame buffer mem */
   __u32 type; /* see FB_TYPE_* */
   __u32 type_aux; /* Interleave for interleaved Planes */
   __u32 visual; /* see FB_VISUAL_* */
```



```
__u16 xpanstep;
                              /* zero if no hardware panning */
   __u16 ypanstep;
                              /* zero if no hardware panning */
                             /* zero if no hardware ywrap */
   __u16 ywrapstep;
                             /* length of a line in bytes */
   __u32 line_length;
   unsigned long mmio_start; /* Start of Memory Mapped I/O (physical
                                  address) */
                              /* Length of Memory Mapped I/O */
   u32 mmio len;
   __u32 accel; /* Indicate to driver which specific chip/card we have */
   __u16 reserved[3];
                              /* Reserved for future compatibility */
};
```

成员名称	描述	支持情况
id	设备驱动名称。	支持。
smem_start	显存起始物理地址。	支持。
smem_len	显存大小。	支持。
type	显卡类型。	固定为 FB_TYPE_PACKED_PIX ELS,表示象素值紧密排 列。
type_aux	附加类型。	不支持,在 FB_TYPE_PACKED_PIX ELS 显卡类型下无含 义。
visual	色彩模式。	不支持,默认为 FB_VISUAL_TRUECOL OR,真彩色。
xpanstep	支持水平方向上的 PAN 显示:  • 0: 不支持。  • 非 0: 支持,此时该值用于表示在水平方向上每步进的象素值。	固定为 1。
ypanstep	支持垂直方向上的 PAN 显示:  • 0: 不支持。  • 非 0: 支持,此时该值用于表示在垂直方向上每步进的象素值。	固定为 1。
ywrapstep	该方式类似于 ypanstep,不同之处 在于: 当其显示到底部时,能回到 显存的开始处进行显示。	不支持,默认为0。
line_length	每行字节数。	支持。



成员名称	描述	支持情况
mmio_start	显存映射 I/O 首地址。	不支持,默认为0。
mmio_len	显存映射 I/O 长度。	不支持,默认为0。
accel	显示所支持的硬件加速设备。	不支持,默认为 FB_ACCEL_NONE,无 加速设备。
reserved	保留。	不支持,缺省值为0。

无。

【相关数据类型及接口】

FBIOGET\_FSCREENINFO

# 3.1.4 struct fb\_cmap

## 【说明】

调色板结构。

## 【定义】

成员名称	描述	支持情况
start	调色板数组中的起始索引(入口号),从 0~255。	支持。
len	调色板数组中的元素个数,最大支持 256 个。	支持。
red	红色分量,每 16bit 中的低 8bit 有效。	支持。
green	绿色分量,每 16bit 中的低 8bit 有效。	支持。



成员名称	描述	支持情况
blue	蓝色分量,每 16bit 中的低 8bit 有效。	支持。
transp	透明度分量。	不支持,默认为 NULL。

无。

【相关数据类型及接口】

- FBIOGETCMAP
- FBIOPUTCMAP

# 3.2 扩展的数据类型

# 3.2.1 HIFB\_COLOR\_FMT\_E

#### 【说明】

HiFB 支持的象素格式集合。

#### 【定义】

```
typedef enum
   HIFB FMT 1BPP = 0,
                            /* 1bpp */
   HIFB_FMT_2BPP,
                            /* 2bpp */
   HIFB_FMT_4BPP,
                            /* 4bpp */
   HIFB_FMT_8BPP,
                            /* 8bpp */
   HIFB_FMT_KRGB444,
                            /* RGB444 */
                            /* RGB555 */
   HIFB FMT KRGB555,
                            /* RGB565 */
   HIFB_FMT_RGB565,
   HIFB FMT ARGB4444,
                            /* RGB4444 */
                            /* RGB1555 */
   HIFB_FMT_ARGB1555,
   HIFB_FMT_KRGB888,
                            /* RGB888 */
   HIFB_FMT_ARGB8888,
                            /* RGB8888 */
   HIFB_FMT_BUTT
}HIFB_COLOR_FMT_E;
```

成员名称	描述
HIFB_FMT_1BPP	索引格式 1bpp。



成员名称	描述
HIFB_FMT_2BPP	索引格式 2bpp。
HIFB_FMT_4BPP	索引格式 4bpp。
HIFB_FMT_8BPP	索引格式 8bpp。
HIFB_FMT_KRGB444	RGB444 格式。
HIFB_FMT_KRGB555	RGB555 格式。
HIFB_FMT_RGB565	RGB565 格式。
HIFB_FMT_ARGB4444	ARGB4444 格式。
HIFB_FMT_ARGB1555	ARGB1555 格式。
HIFB_FMT_KRGB888	RGB888 格式。
HIFB_FMT_ARGB8888	ARGB8888 格式。
HIFB_FMT_BUTT	非法象素格式。

无。

【相关数据类型及接口】

无。

# 3.2.2 HIFB\_CAPABILITY\_S

#### 【说明】

各个叠加层的支持能力。

#### 【定义】

```
typedef struct
   HI_BOOL bKeyAlpha;
                             /* whether support colorkey alpha */
   HI BOOL bGlobalAlpha;
                             /* whether support global alpha */
                             /* whether support color map */
   HI_BOOL bCmap;
   HI BOOL bColFmt[HIFB FMT BUTT]; /* support which color format */
   HI_U32 u32MaxWidth;
                             /* the max pixels per line */
   HI_U32 u32MaxHeight;
                             /* the max lines */
   HI U32 u32MinWidth;
                             /* the min pixels per line */
   HI_U32 u32MinHeight;
                             /* the min lines */
   HI_U32 u32VDefLevel;
                              /* vertical deflicker level, less than 2
                            means vertical deflicker is unsupported */
```



HI\_U32 u32HDefLevel;

/\* horizontal deflicker level, less than 2
means horizontal deflicker is unsupported \*/

}HIFB\_CAPABILITY\_S;

#### 【成员】

成员名称	描述
bKeyAlpha	是否支持带 Alpha 的 colorkey。
bGlobalAlpha	是否支持全局 Alpha 和象素 Alpha 叠加。
bCmap	是否支持调色板模式。
bColFmt	支持的象素格式。 例如: bColFmt[HIFB_FMT_ARGB1555] = 1,表示支持 ARGB1555 格式。
u32MaxWidth	最大分辨率的宽度。
u32MaxHeight	最大分辨率的高度。
u32MinWidth	最小分辨率的宽度。
u32MinHeight	最小分辨率的高度。
u32VDefLevel	支持的垂直抗闪烁最大级别,小于2为不支持。
u32HDefLevel	支持的水平抗闪烁最大级别,小于2为不支持。

#### 【注意】

• bGlobalAlpha = 1

表示支持全局 Alpha 和像素 Alpha 叠加,当叠加层在处于 Alpha 通道模式时,叠加 Alpha 值来源于全局 Alpha 和象素 Alpha 的叠加。

• bGlobalAlpha = 0

表示不支持全局 Alpha 和像素 Alpha 叠加,当叠加层处于 Alpha 通道模式时,叠加 Alpha 值就等于全局 Alpha。

#### 【相关数据类型及接口】

- HIFB COLOR FMT E
- FBIOGET\_CAPABILITY\_HIFB

# 3.2.3 HIFB\_POINT\_S

#### 【说明】

坐标结构体。

#### 【定义】

typedef struct



#### 【成员】

成员名称	描述
u32PosX	水平坐标。
u32PosY	垂直坐标。

#### 【注意】

无。

## 【相关数据类型及接口】

- FBIOGET\_SCREEN\_ORIGIN\_HIFB
- FBIOPUT\_SCREEN\_ORIGIN\_HIFB

# 3.2.4 HIFB\_COLORKEY\_S

#### 【说明】

colorkey 结构体。

#### 【定义】

成员	描述
bKeyEnable	禁止或打开 colorkey 功能:
	● 1: 打开。
	• 0: 禁止,本结构体的其他成员值都将无效。



成员	描述
bMaskEnable	禁止或打开 key mask 功能:  ● 1: 打开。
	• 0: 禁止,所有 key mask 值无效。
u32Key	被透明掉的象素值,与当前层设置的象素格式一致,例如: • 若当前是调色板模式时,key 值对应一个调色板索引。 • 若当前是 RGB1:5:5:5 格式时,透明掉蓝色,则 key 值是 0x001F。
u8RedMask	红色分量 key mask 值。
u8GreenMask	绿色分量 key mask 值。
u8BlueMask	蓝色分量 key mask 值。
u8Reserved	保留。

key 值与 mask 值的作用如下:

#### key

当该叠加层支持带 Alpha 的 colorkey 时,key 值的最高 8bit 称为 key\_alpha。在图像叠加层中的象素,若有与 key(不包含 key\_alpha)一致的值时,需要根据 key alpha 进行判断。

- 当 key\_alpha 为 0 时,该象素不被显示,但是可以直接看到该层的下一层相应位置的颜色。
- 当 key\_alpha 不为 0 时,只是改变该像素叠加的 alpha 值。该象素的叠加 alpha 值等于 key alpha。

利用 key 只能透明一种颜色,但通过 key 和 mask 的结合可以透明一定范围的颜色。

当该叠加层不支持带 Alpha 的 colorkey 时,key 值的最高 8bit 无效。在图象叠加层中的象素,若有与 key(不包含最高 8bit)一致的值时,该象素不被显示。

#### mask

key mask 中的比特位和颜色分量中的比特位——对应。当 key mask 中的某一比特位为 1 时,说明颜色分量对应的比特位不敏感;当 key mask 中的某一比特位为 0,说明颜色分量对应的比特位敏感。

例如,对于 RGB8:8:8:8 格式,若 key 是  $0x33_4455$ ,即 R、G、B 的值分别为 0x33、0x44、0x55,而 R、G、B 的 mask 分别是 0x07、0x0F、0x0F 时,则所有满足红色分量落在[0x30,0x37],绿色分量落在[0x40,0x4F],蓝色分量落在 [0x50,0x5F]区间的颜色值将被透明掉。



不管当前图像叠加层是不是 RGB8:8:8 格式, mask 都是针对 RGB8:8:8 格式的 颜色值进行运算。例如,对于 RGB1:5:5:5 格式的 key 值 0x1F,其对应的 RGB8:8:8:8 格式的颜色值为 0xFF,则 mask 值针对 0xFF 进行运算。

## □ 说明

- Hi3510 芯片不支持带 Alpha 的 colorkey, 颜色分量的 key mask 值低 4bit 有效, 且蓝色分量和 绿色分量共用一个 mask: u8GreenMask, 即 u8BlueMask 无效。
- Hi3511 芯片不支持带 Alpha 的 colorkey, 颜色分量的 key mask 值低 4bit 有效。
- Hi3512 芯片不支持带 Alpha 的 colorkey, 颜色分量的 key mask 值低 4bit 有效。
- Hi3520 芯片不支持带 Alpha 的 colorkey。

#### 【相关数据类型及接口】

- FBIOGET COLORKEY HIFB
- FBIOPUT\_COLORKEY\_HIFB

# 3.2.5 HIFB\_ALPHA\_S

#### 【说明】

Alpha 结构体。

#### 【定义】

成员名称	描述
bAlphaEnable	Alpha 叠加使能,默认为 1。
bAlphaChannel	Alpha 通道使能,默认为 0。
u8Alpha0	Alpha0 值,范围 0~255,默认为 255。在 RGB1:5:5:5 格式下,当最高位为 0 时,选择该值作为 Alpha 叠加的 Alpha 值。
u8Alpha1	Alpha1 值,范围 0~255,默认为 255。在 RGB1:5:5:5 格式下,当最高位为 1 时,选择该值作为 Alpha 叠加的 Alpha 值。
u8GlobalAlpha	全局 Alpha 值,范围为 $0\sim255$ ,默认为 $255$ 。在 Alpha 通道 使能时起作用。



成员名称	描述
u8Reserved	保留。

只有在 Alpha 叠加使能的情况下才进行 Alpha 叠加, 否则低优先级的叠加层将覆盖高优先级的叠加层。

叠加 Alpha 值的计算公式有以下几种情况:

- 当 Alpha 通道使能时,全局 Alpha 参与叠加。
  - 对于不支持全局 Alpha 和象素 Alpha 叠加的芯片 (例如 Hi3510、Hi3511、Hi3512), 叠加 Alpha 值的计算公式如下所示: α = u8GlobalAlpha
  - 对于支持全局 Alpha 和像素 Alpha 叠加的芯片(例如 Hi3520),叠加 Alpha 值的计算公式如下所示:  $\alpha = u8GlobalAlpha*\alpha_{pixel}$
- 当 Alpha 通道不使能时,叠加 Alpha 值等于像素 Alpha 值,即:  $\alpha = \alpha_{nixel}$

#### 【相关数据类型及接口】

- FBIOGET ALPHA HIFB
- FBIOPUT\_ALPHA\_HIFB

# 3.2.6 HIFB DEFLICKER S

#### □ 说明

- Hi3510 芯片不支持抗闪烁功能, 所以水平和垂直抗闪烁的设置和获取操作都不被允许。
- Hi3511 芯片不支持抗闪烁功能, 所以水平和垂直抗闪烁的设置和获取操作都不被允许。
- Hi3512 芯片不支持抗闪烁功能,所以水平和垂直抗闪烁的设置和获取操作都不被允许。

#### 【说明】

抗闪烁结构体,用于设置或获取叠加层的抗闪烁设置。

#### 【定义】

```
typedef struct hiHIFB_DEFLICKER_S
{
    HI_U32 u32HDfLevel;    /* horizontal deflicker level */
    HI_U32 u32VDfLevel;    /* vertical deflicker level */
    HI_U8 *pu8HDfCoef;    /* horizontal deflicker coefficient */
    HI_U8 *pu8VDfCoef;    /* vertical deflicker coefficient */
}
HIFB DEFLICKER S;
```

成员	描述
u32HDfLevel	水平抗闪烁阶数。



成员	描述
u32VDfLevel	垂直抗闪烁阶数。
pu8HDfCoef	水平抗闪烁系数,个数等于水平抗闪烁阶数减1。
pu8VDfCoef	垂直抗闪烁系数,个数等于垂直抗闪烁阶数减1。

阶数指得到每行(列)处理结果中参与运算的行(列)数。一般而言,抗闪烁阶数越高,得到的效果也越好,但是也会带来图像的模糊。

#### 【相关数据类型及接口】

- FBIOGET DEFLICKER HIFB
- FBIOPUT DEFLICKER HIFB

# 3.2.7 HIFB\_SURFACE\_S

#### 【说明】

Surface 结构体,用于双缓冲时两块 Surfcace 的属性设置。

#### 【定义】

#### 【成员】

成员	描述
pvPhyAddr	Surface 的物理地址。
stAlpha	Surface 的 alpha 属性。
stColorkey	Surface 的 colorkey 属性。

#### 【注意】

Surface 的物理地址必须在该叠加层配置的显存范围内。

【相关数据类型及接口】

FBIOFLIP\_SURFACE



# 3.2.8 HIFB\_YCBCR\_S

## 【说明】

YCbCr 颜色结构体。

## 【定义】

```
typedef struct
{
    HI_U8 u8Cr;
    HI_U8 u8Cb;
    HI_U8 u8Y;
    HI_U8 u8Reserved;
}
HIFB_YCBCR_S;
```

## 【成员】

成员	描述
u8Cr	Cr 分量值。
u8Cb	Cb 分量值。
u8Y	Y 分量值。
u8Reserved	保留值。

## 【注意】

无。

【相关数据类型及接口】

无。

# 3.2.9 HI3511\_CURSOR\_E

#### 【说明】

Hi3511/Hi3512 芯片的硬件鼠标层模式定义。

#### 【定义】

```
typedef enum
{
    HI3511_CURSOR_2COLOR = 0,
    HI3511_CURSOR_3COLOR,
    HI3511_CURSOR_4COLOR
}HI3511_CURSOR_E;
```



成员	描述
HI3511_CURSOR_2COLOR	双色和透明模式。
HI3511_CURSOR_3COLOR	三色和透明模式。
HI3511_CURSOR_4COLOR	四色模式。

无。

【相关数据类型及接口】

无。

# 3.2.10 HI3511\_CURSOR\_S

## 【说明】

Hi3511/Hi3512 芯片的硬件鼠标层属性设置结构体。

## 【定义】

```
typedef struct
{
    HI3511_CURSOR_E enCursor;
    HIFB_YCBCR_S stYCbCr[4];
}HI3511_CURSOR_S;
```

#### 【成员】

成员	描述
enCursor	鼠标层模式。
stYCbCr[4]	调色板数据。

## 【注意】

无。

【相关数据类型及接口】

FBIOFLIP\_SURFACE



# 4 图形开发辅助接口

🔲 说明

本章内容只针对 Hi3520 芯片。

# 4.1 概述

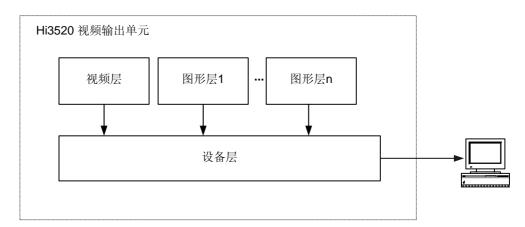
# 4.1.1 简介

Hi3520 视频输出单元主要由设备层、视频层和图形层组成,如图 4-1 所示,具体关系如下:

- 设备层是基础,视频层和若干图形层基于设备层。设备层按照配置,输出一定的时序信号驱动与之相连的显示设备输出视频和图形,同时设备层决定了设备分辨率,即限制了视频层和图形层的显示分辨率。
- 基于以上架构,任何关于设备层的操作,都需要先关闭其上的视频层和所有图形 层,再对设备层进行操作,这样才能正确显示视频和图形。例如:
  - 关闭设备层时,需要先关闭视频层和图形层,再关闭设备层。
  - 设备层属性变化时,如切换设备输出分辨率等,需要先关闭视频层和图形层, 再关闭设备层,最后依次重新配置并开启设备层、视频层和图形层。



#### 图4-1 Hi3520 视频输出单元基本结构



# 4.1.2 注意事项

开发 Hi3520 图形层时需要注意以下事项:

## 如何在 Hi3520 中看到图形层输出

图形层要在显示设备上正常显示,需要先配置并开启设备层。

Hi3520 SDK 中,支持 3 个显示设备,每个显示设备又支持若干种时序输出,SDK 在此不提供默认的设备层配置,也不在 HiFB 模块插入时默认打开设备层。用户需要显式地调用相关接口使能设备层,再操作图形层,才能看到显示结果。

#### □ 说明

Hi3511 由于只有一个显示设备,输出时序选择有限,故在 HiFb 模块插入时,提供了默认的时序并开启设备层。用户不需要显式地配置并启动设备层,仅操作图形层接口 (HiFB 接口),即可看到图形层输出结果。

Hi3520 SDK 提供 2 种方式控制设备层:

- 推荐方式:使用 VOU 模块接口控制设备层。Hi3520 SDK 的 VOU 模块提供设备 层和视频层控制接口,其中操作设备层的接口包括:HI\_MPI\_VO\_Enable/HI\_MPI\_VO\_Disable / HI\_MPI\_VO\_SetPubAttr / HI\_MPI\_VO\_GetPubAttr。
- 备选方式: 在 VOU 模块不存在时,使用 VD 模块控制设备层。

注意:由于以上 2 种方式都可操作设备层,为防止操作冲突,限定设备层不可被重复使能,如先使用 VOU 模块接口  $HI_MPI_VO_E$  使能了 HD 设备,则再使用 VD 接口  $VD_E$  VABLE\_DEV 使能 HD 设备会返回失败。

用户使用时,应只使用一套接口控制设备层(推荐使用 VOU 控制设备接口),并且遵循操作设备应先必须关闭视频和图形层的原则。

#### □ 说明

VOU 模块接口详见《Hi3520 媒体处理软件开发参考》2.2.3 节; VD 模块接口详见"4.2 API 参考"。



## 如何在不同设备间切换图形层

Hi3520 支持 5 个图形层(简称 G0 $\sim$ G4),它们与 3 个输出设备(HD/AD/SD)的约束 关系如表 4-1 所示。

表4-1 图形层与输出设备的对应关系表

图形层	对应显示设备	说明
叠加图行层 0	HD	G0 只能在 HD 设备上显示。
(G0)		
	HD 或 AD	G1 可在 HD 或 AD 设备上显示,调用者可通过 绑定接口指定显示设备。
(G1)		G1 切换为静态切换,即 HD 和 AD 设备会自动 关闭再开启。G1 切换的用户操作如下:先关闭 G1→设置新的绑定关系→配置并开启 G1。
叠加图行层 2 (G2)	AD	G2 只能在 AD 设备上显示。
叠加图行层 3 (G3)	SD	G3 只能在 SD 设备上显示。
叠加图行层 4	HD 或 AD	G4 常用作鼠标显示。
(G4)		G4 可在 HD 或 AD 设备上显示,调用者可通过 绑定接口指定显示设备。
		G4 切换的用户操作和 G1 切换时一致,即先关闭本图形层→设置新的绑定关系→配置并开启本图形层。
		G4 总是处在显示设备叠加层的最高层。如 HD 上有视频层、G0 和 G4,则叠加顺序为 G0 在视 频层之上、G4 在 G0 之上。

G1 和 G4 可在 HD 和 AD 设备间切换,调用者可通过 VD 模块的绑定接口指定图形层与设备间的绑定关系。VD 模块以 ioctl 方式提供用户操作接口,在调用接口前,应打开设备/dev/vd 以获取文件操作句柄。

- G1/G4 切换的用户操作一致,描述如下:即先关闭本图形层→设置新的绑定关系 →配置并开启本图形层。
- G4 切换与 G1 切换的不同点是: G1 在用户设置绑定关系时,会自动关闭再开启 HD 和 AD 设备,故会出现短暂黑屏; G4 切换时 HD 和 AD 设备状态保持不变, 不会关闭再开启,故不会出现短暂黑屏现象。



# 4.2 API 参考

# 4.2.1 VD\_SET\_GRAPHIC\_BIND

## 【目的】

设置图形层绑定关系。

#### 【语法】

#### 【描述】

使用此接口绑定图形层到指定的 VO 设备:可指定图形层 G1 绑定到 HD 或 AD 设备,G4 绑定到 HD 或 AD 设备。其它图形层(包括 G0、G2、G3)具有固定绑定的设备,故不可设置绑带关系。

#### 【参数】

参数名称	描述	输入/输出
Fd	Vd 设备的文件描述符	输入
VD_SET_GRAPHIC_BIND	ioctl 号	输入
pBindAtrr	绑定属性,包括: 绑定的图形层 s32GraphicId: 范围 [0~4],分别对应 G0~G4。本操作 有效的图形层号为 1(即 G1)和 4(即 G4) 被绑定的设备号 DevId: 范围 [0~2],分别对应 HD/AD/SD。本 操作有效的设备号为 0(即 HD)和 1 (即 AD)。	输入

#### 【返回值】

返回值	描述
0	成功
-1	失败

## 【需求】

头文件: mkp\_vd.h



- 图形层 G0 固定绑定于 HD, G2 固定绑定于 AD, G3 固定绑定于 SD, 故这 3 个图 形层不支持设置绑定关系。
- 图形层 G1 可绑定于 HD 或 AD 设备; G4 也可绑定于 HD 或 AD 设备。G1 和 G4 可同时绑定与同一个设备,但同一个图形层不能同时绑定与 2 个设备。
- 各设备上视频和图形层优先级由低到高排序如下:
- HD 设备:视频层→G0—〉G1(如果存在)--〉G4(如果存在)
- HD 设备:视频层→G2—〉G1(如果存在)--〉G4(如果存在)
- HD 设备:视频层→G3
- G1 和 G4 在设备间切换的用户操作相同。以切换 G1 从 HD 到 AD 为例,用户操作如下: 先关闭 G1 层(通过设置 HiFb 的 close 接口),再设置 G1 绑定于 AD 设备,最后重新设置并开启 G1 层(通过设置 HiFb 的接口)。
- G1 切换是静态切换,即设置绑定关系时内部会自动关闭 HD 和 AD 设备,再开启,故存在短暂黑屏现象;而 G4 切换是动态切换,即设置绑定关系时内部不会关闭 HD 和 AD, 故无输出设备黑屏现象。

#### 【举例】

```
VD BIND S stBind;
   int fd;
   /*open vd dev*/
fd = open("/dev/vd", O RDWR, 0);\
   if(fd < 0)
      printf("open vd failed!\n");
      return -1;
}
   /*bind HC to HD*/
   stBind.s32GraphicId = 1; /*G1: 1*/
   stBind.DevId = HD;
   if (ioctl(fd, VD SET GRAPHIC BIND, &stBind) != HI SUCCESS)
      printf("[fd:%d]set bind glayer %d to dev %d failed!\n",fd,HC,HD);
      close(fd);
      return -1;
}
if (ioctl(fd, VD GET GRAPHIC BIND, &stBind) != HI SUCCESS)
{
    printf("[fd:%d]get glayer %d bind failed!\n",fd,Glayer);
     return -1;
}
```



close(fd);

# 4.2.2 VD\_GET\_GRAPHIC\_BIND

## 【目的】

获取指定图形层绑定的设备号。

#### 【语法】

## 【描述】

获取指定图形层绑定的设备号。

## 【参数】

参数名称	描述	输入/输出
fd	Vd 设备的文件描述符	输入
VD_GET_GRAPHIC_BIND	ioctl 号	输入
pBindAtrr	绑定属性,调用者指定图形层号,返回被绑定的设备号: 绑定的图形层 s32GraphicId: 范围 [0~4],分别对应 G0~G4。 被绑定的设备号 DevId: 输出属性,范围[0~2]	输入/输出

## 【返回值】

返回值	描述
0	成功
-1	失败

## 【需求】

头文件: mkp\_vd.h

【注意】

无。

【举例】



请参考 VD\_SET\_GRAPHIC\_BIND 中的用例。

# 4.2.3 VD\_SET\_DEV\_ATTR

## 【目的】

设置视频输出设备的公共属性,指定接口类型、时序等配置。

## 【语法】

## 【描述】

设置输出设备的公共属性,必须在使能设备前指定。

## 【参数】

参数名称	描述	输入/输出
fd	Vd 设备的文件描述符	输入
VD_ENABLE_DEV	ioctl 号	输入
pstPubAttr	输出设备公共属性结构,包括: 输出设备 Id: 范围[0~2],分别对 应 HD/AD/SD 输出设备属性: 见数据结构说明	输入

#### 【返回值】

返回值	描述
0	成功
-1	失败

## 【需求】

头文件: mkp\_vd.h

## 【注意】

使能设备前必须设置设备公共属性。

## 【举例】

无。



# 4.2.4 VD\_GET\_DEV\_ATTR

## 【目的】

获取视频输出设备的公共属性,包括接口类型、时序配置。

## 【语法】

## 【描述】

获取输出设备的公共属性。

## 【参数】

参数名称	描述	输入/输出
fd	Vd 设备的文件描述符	输入
VD_ENABLE_DEV	ioctl 号	输入
pstPubAttr	输出设备公共属性结构。 用户指定设备 Id,返回相应的设备属性: 输出设备 Id:范围[0~2],分别对应 HD/AD/SD 输出设备属性:见数据结构说明	输入/输出

## 【返回值】

返回值	描述
0	成功
-1	失败

## 【需求】

头文件: mkp\_vd.h

#### 【注意】

使能设备前必须设置设备公共属性。

#### 【举例】

无。



# 4.2.5 VD\_ENABLE\_DEV

## 【目的】

使能视频输出设备。

#### 【语法】

#### 【描述】

使用此接口使能 VO 设备,使之输出符合配置的时序信号。

#### 【参数】

参数名称	描述	输入/输出
fd	Vd 设备的文件描述符	输入
VD_ENABLE_DEV	ioctl 号	输入
pVoDevId	待使能的 VO 设备号	输入

## 【返回值】

返回值	描述
0	成功
-1	失败

## 【需求】

头文件: mkp\_vd.h

## 【注意】

- 当系统存在 VOU 模块时,请使用 VOU 模块的接口使能相应的设备 (HI\_MPI\_VO\_Enable 和 HI\_MPI\_VO\_SetPubAttr)。
- 如果视频设备已经使能,则返回错误码 HI\_ERR\_VD\_DEV\_NOT\_DISABLE。

#### 【举例】

无。

# 4.2.6 VD\_DISABLE\_DEV

【目的】



关闭视频输出设备。

#### 【语法】

#### 【描述】

使用此接口关闭 VO 设备, 故视频层和图形层都无法显示。

#### 【参数】

参数名称	描述	输入/输出
fd	Vd 设备的文件描述符	输入
VD_ENABLE_DEV	ioctl 号	输入
pVoDevId	关闭的 VO 设备号	输入

## 【返回值】

返回值	描述
0	成功
-1	失败

## 【需求】

头文件: mkp\_vd.h

## 【注意】

由于视频层和图形层都是基于视频输出设备的,设备关闭前,应先关闭视频(通过 MPP VOU 接口)和图形层(通过 HiFb 接口),再关闭输出设备。

#### 【举例】

无。

# 4.3 数据结构

# 4.3.1 VO\_PUB\_ATTR\_S

## 【说明】

定义视频输出公共属性结构体。



## 【定义】

```
typedef struct hiVO_PUB_ATTR_S
{
    /* background color of device [RGB] */
    HI_U32     u32BgColor;

    /* vo inteface type */
    VO_INTF_TYPE_E enIntfType;

    /* vo interface synchronization */
    VO_INTF_SYNC_E enIntfSync;

    /* video synchronizing information */
    VO_SYNC_INFO_S stSyncInfo;
}
VO_PUB_ATTR_S;
```

成员名称	描述
u32BgColor	设备背景色,表示方法 RGB888。
enIntfType	接口类型典型配置,原型定义:  typedef enum hiVO_INTF_TYPE_E  {      VO_INTF_CVBS = 0,     VO_INTF_BT656 = 1,     VO_INTF_VGA = 2,     VO_INTF_YPBPR = 3,     VO_INTF_BT1120 = 4,     VO_INTF_LCD = 5,     VO_INTF_BUTT  } VO_INTF_TYPE_E;



成员名称	描述
enIntfSync	接口时序典型配置,原型定义:
	typedef enum hiVO_INTF_SYNC_E
	{
	$VO_OUTPUT_PAL = 0,$
	VO_OUTPUT_NTSC = 1,
	$VO_OUTPUT_720P60 = 2,$
	$VO_OUTPUT_1080I60 = 3,$
	$VO_OUTPUT_1080P30 = 4,$
	$VO\_OUTPUT\_800x600\_60 = 5,$
	$VO_OUTPUT_1024x768_60 = 6,$
	$VO_OUTPUT_1280x1024_60 = 7,$
	$VO_OUTPUT_1366x768_60 = 8,$
	$VO_OUTPUT_1440x900_60 = 9,$
	VO_OUTPUT_USER = 10,
	VO_OUTPUT_BUTT
	} VO_INTF_SYNC_E;



成员名称	描述	
stSyncInfo	接口时序结构体,原型定义:	
	typedef struct tagVO_SYNC_INFO_S	
	{	
	HI_BOOL bSynm;	
	HI_BOOL blop;	
	HI_U8 u8Intfb;	
	W 1146	
	HI_U16 u16Vact;	
	HI_U16 u16Vbb;	
	HI_U16 u16Vfb;	
	HI_U16 u16Hact;	
	HI_U16 u16Hbb;	
	HI_U16 u16Hfb;	
	HI_U16 u16Bvact;	
	HI_U16 u16Bvbb;	
	HI_U16 u16Bvfb;	
	HI_U16 u16Hpw;	
	HI_U16 u16Vpw;	
	HI_BOOL bldv;	
	HI_BOOL blhs;	
	HI_BOOL blvs;	
	} VO_SYNC_INFO_S;	

#### 【注意事项】

- 当接口时序配置为 VO\_OUTPUT\_USER 时,stSyncInfo 定义的时序结构才会生效,表示用户自定义的时序结构。
- 接口类型配置为 VO\_INTF\_CVBS 或 VO\_INTF\_BT656 时, enIntfSync 的取值范围 为[0, 1]。
- 接口类型为 VO\_INTF\_VGA、VO\_INTF\_BT1120 或者 VO\_INTF\_LCD 时, enIntfSync 的取值范围为[6, 10]。
- 接口类型为 VO\_INTF\_YPBPR 或 VO\_INTF\_BT1120 时, enIntfSync 的取值范围为 [2, 5]。
- 配置设备属性在设备下一次打开时生效。



# 4.3.2 VD\_PUB\_ATTR\_IOC\_S

#### 【说明】

公共属性的 ioctrl 形式。

#### 【定义】

#### 【成员】

成员	描述
DevId	输出设备 ID 号。
	取值范围:
	0: HD。
	1: AD.
	2: SD.
stPubAttr	VO 公共属性。

## 【注意】

无。

【相关数据类型及接口】

无。

# 4.3.3 VD\_PUB\_ATTR\_IOC\_S

#### 【说明】

公共属性的 ioctrl 形式。

#### 【定义】

```
typedef struct hiVD_BIND_S
{
    /* which graphic layer to bind(0:G0,1:G1,2:G2,3:G3,4:HC)*/
    HI_S32 s32GraphicId;

    /* which device bind to(0:HD,1:AD,2:SD)*/
    VO_DEV DevId;
} VD_BIND_S;;
```



# 【成员】

成员	描述
s32GraphicId	图形层号。
	0: G0.
	1: G1.
	2: G2.
	3: G3.
	4: HC°
DevId	待绑定的设备号。
	0: HD。
	1: AD.
	2: SD <sub>°</sub>

# 【注意】

无。

【相关数据类型及接口】

无。