



HiFB

## API 参考

文档版本      00B05  
发布日期      2018-10-30

**版权所有 © 深圳市海思半导体有限公司 2018。保留一切权利。**

未经本公司书面许可，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



**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

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

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。

## 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](mailto:support@hisilicon.com)



# 前言

## 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，还扩展了一些图形层控制功能，如层间 Alpha、设置原点等。本文档主要介绍 HiFB 的 API 和数据类型以及 Proc 调试信息。



### 说明

- 本文未做特殊说明，Hi3556V100 与 Hi3559V100 一致。
- 本文未做特殊说明，Hi3559CV100 与 Hi3559AV100 一致。
- 本文未做特殊说明，Hi3556AV100、Hi3516CV500、Hi3516DV300 与 Hi3519AV100 一致。
- 本文未做特殊说明，Hi3559V200、Hi3556V200 与 Hi3516CV500 一致。

## 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
Hi3559	V100
Hi3556	V100
Hi3559A	V100ES
Hi3559A	V100
Hi3559C	V100
Hi3519A	V100
Hi3556A	V100
Hi3516C	V500
Hi3516D	V300
Hi3559	V200
Hi3556	V200








## 读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

## 符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 <b>危险</b>	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
 <b>警告</b>	表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 <b>注意</b>	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 <b>窍门</b>	表示能帮助您解决某个问题或节省您的时间。
 <b>说明</b>	表示是正文的附加信息，是对正文的强调和补充。

## 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。

### 文档版本 00B05（2018-10-30）

第 5 次临时版本发布。

2.3 小节，FBIOPUT\_VSCREENINFO 的【注意】涉及修改

2.4 小节，FBIOPUT\_MIRROR\_MODE、FBIOPUT\_COMPRESSION\_HIFB、FBIOPUT\_WAITFOR\_FRESH\_DONE、FBIOPUT\_SCREEN\_SIZE、FBIOPUT\_SCREEN\_SIZE、FBIOPUT\_ROTATE\_MODE、FBIOPUT\_ROTATE\_MODE 的【注意】涉及修改

FBIOPUT\_DYNAMIC\_RANGE\_HIFB、FBIOPUT\_DYNAMIC\_RANGE\_HIFB 新增【差异说明】

3.2 小节，HIFB\_LAYER\_INFO\_S 的【注意】涉及修改



## 文档版本 00B04 (2018-09-04)

第 4 次临时版本发布。

2.4.1 小节, FBIOPUT\_COMPRESSION\_HIFB、FBIOGET\_COLORKEY\_HIFB 【注意】涉及修改

3.2 小节, HIFB\_LAYER\_INFO\_S 【注意】涉及修改

添加 Hi3516CV500 和 Hi3516CV300 的相关内容

## 文档版本 00B03 (2018-08-08)

第 3 次临时版本发布。

2.4.1 小节 FBIOPUT\_SCREEN\_SIZE 的【参数】涉及修改

3.2 小节, HIFB\_LAYER\_INFO\_S 的【成员】涉及修改

## 文档版本 00B02 (2018-06-20)

第 2 次临时版本发布。

删除 Hi3516A/Hi3518EV20X/Hi3519V100/Hi3519V101/Hi3516CV300 相关内容

## 文档版本 00B01 (2018-04-11)

第 1 次临时版本发布

添加 Hi3519AV100 相关内容



# 目 录

前 言.....	i
1 概述.....	1-1
1.1 概述.....	1-1
1.2 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 参考 .....	2-1
2.1 API 类别 .....	2-1
2.2 ioctl 函数.....	2-1
2.3 标准功能.....	2-3
2.4 扩展功能.....	2-10
2.4.1 通用功能 .....	2-10
2.4.2 软鼠标功能 .....	2-43
2.5 错误码.....	2-55
3 数据类型.....	3-1
3.1 在标准中定义的数据类型 .....	3-1
3.2 扩展的数据类型.....	3-6
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 .....	4-3
4.3 数据结构.....	4-6
5 Proc 调试信息.....	5-1
5.1 图形层和 fb 设备号对应关系.....	5-1
5.2 单个图形层调试信息.....	5-1
5.3 图形层的绑定关系.....	5-6



## 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 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 .....	2-55
表 4-1 不同芯片在不同设备间切换图形层.....	4-3





# 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，还扩展了一些图形层控制功能，如层间 Alpha、设置原点、FB 扩展模式等。

## 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

允许设置像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：

```
int ioctl(int fd,
          unsigned long cmd,
          .....
          );
```

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：

```
int ioctl (int fd,
           unsigned long cmd,
```



```
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： <ul style="list-style-type: none"><li>• FBIOGET_VSCREENINFO：获取屏幕可变信息</li><li>• FBIOPUT_VSCREENINFO：设置屏幕可变信息</li><li>• FBIOGET_FSCREENINFO：获取屏幕固定信息</li><li>• FBIOPAN_DISPLAY：设置 PAN 显示</li><li>• FBIOGET_CAPABILITY_HIFB：获取叠加层的支持能力</li><li>• FBIOGET_SCREEN_ORIGIN_HIFB：获取叠加层坐标原点</li><li>• FBIOPUT_SCREEN_ORIGIN_HIFB：设置叠加层坐标原点</li><li>• FBIOGET_SHOW_HIFB：获取叠加层显示状态</li><li>• FBIOPUT_SHOW_HIFB：设置叠加层显示状态</li><li>• FBIOGET_ALPHA_HIFB：获取叠加层 Alpha</li><li>• FBIOPUT_ALPHA_HIFB：设置叠加层 Alpha</li><li>• FBIOGET_COLORKEY_HIFB：获取叠加层的 Colorkey 属性</li><li>• FBIOPUT_COLORKEY_HIFB：设置叠加层的 Colorkey 属性</li><li>• FBIOGET_MDDRDETECT_HIFB：获取内存侦测属性</li><li>• FBIOPUT_MDDRDETECT_HIFB：设置内存侦测属性</li><li>• FBIOPUT_DYNAMIC_RANGE_HIFB：设置叠加层的目标图像动态范围</li><li>• FBIOGET_DYNAMIC_RANGE_HIFB：获取叠加层的目标图像动态范围</li><li>• FBIOPUT_SCREENSIZE：设置叠加层的屏幕输出分辨率</li><li>• FBIOGET_SCREENSIZE：获取叠加层的屏幕输出分辨率</li><li>• 软鼠标的一系列操作</li></ul>	输入



参数名称	描述	输入/输出
cmddata	各 cmd 对应的数据类型分别是： <ul style="list-style-type: none"><li>• 获取或设置屏幕可变信息：struct fb_var_screeninfo *类型</li><li>• 获取屏幕固定信息：struct fb_fix_screeninfo *类型</li><li>• 设置 PAN 显示：struct fb_var_screeninfo *类型</li><li>• 获取叠加层支持能力：HIFB_CAPABILITY_S *类型</li><li>• 获取或设置屏幕叠加层坐标原点：HIFB_POINT_S *类型</li><li>• 获取或设置叠加层显示状态：HI_BOOL *类型</li><li>• 获取或设置叠加层 Alpha：HIFB_ALPHA_S *类型</li><li>• 获取或设置内存侦测属性：HIFB_DDRZONE_S *类型</li><li>• 获取或设置压缩开关状态：HI_BOOL 类型</li><li>• 获取或设置图形层动态范围：HIFB_DYNAMIC_RANGE_E* 类型</li></ul>	输入 输出

## 2.3 标准功能

### FBIOGET\_VSCREENINFO

#### 【目的】

获取屏幕的可变信息。

#### 【语法】

```
int ioctl (int fd,  
           FBIOGET_VSCREENINFO,  
           struct fb_var_screeninfo *var);
```

#### 【描述】

使用此接口获取屏幕的可变信息，主要包括分辨率和像素格式。信息的详细描述请参见“3.1 struct fb\_var\_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_VSCREENINFO	ioctl 号	输入
var	可变信息结构体指针	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】

- 高清设备的图形层默认分辨率为 1280x720，鼠标层的默认分辨率为 128x128，标清设备的图形层的默认分辨率为 720x576，像素格式默认为 ARGB1555。
- 特别说明：对于 Hi3559AV100ES/Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516DV300 芯片，超清和高清设备的图形层默认分辨率为 1920x1080。

### 【举例】

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

### 【相关接口】

[FBIOPUT\\_VSCREENINFO](#)

## FBIOPUT\_VSCREENINFO

### 【目的】

设置 Framebuffer 的屏幕分辨率和像素格式等。

### 【语法】

```
int ioctl (int fd,  
           FBIOPUT_VSCREENINFO,  
           struct fb_var_screeninfo *var);
```

### 【描述】

使用此接口设置屏幕分辨率、像素格式。

### 【参数】

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_VSCREENINFO	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

#### 【差异说明】

芯片	描述
Hi3559V100/Hi3559AV100ES/Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516DV300	支持像素格式 ARGB1555、ARGB4444 和 ARGB8888

#### 【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 [FBIOPUT\\_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。
- 对于隔行显示设备，要求分辨率的高度必须为偶数。
- 除 Hi3559AV100ES、Hi3559AV100、Hi3519AV100 外，在压缩使能时，如果改变实际分辨率，需要先关闭压缩。对于 Hi3559AV100ES、Hi3559AV100、Hi3519AV100，压缩功能在 TDE 搬移过程中完成，不涉及额外压缩数据 Buffer 的申请和销毁，因此，改变实际分辨率不需要关闭压缩。
- 如果图形层支持缩放，可以设置显示分辨率大于设备分辨率，这时候显示图像的一部分。

#### 【举例】

设置实际分辨率为 720x576，虚拟分辨率为 720x576，偏移为 (0, 0)，像素格式为 ARGB1555 的示例代码如下：

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



```
struct fb_bitfield a16 = {15, 1, 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 = 16;
vinfo.xoffset = 0;
vinfo.yoffset = 0;
vinfo.red = r16;
vinfo.green = g16;
vinfo.blue = b16;
vinfo.transp= a16;
if (ioctl(fd, FBIOPUT_VSCREENINFO, &vinfo) < 0)
{
    return -1;
}
```

#### 【相关接口】

#### [FBIOGET\\_VSCREENINFO](#)

### FBIOGET\_FSCREENINFO

#### 【目的】

获取 Framebuffer 的固定信息。

#### 【语法】

```
int ioctl (int fd,
           FBIOGET_FSCREENINFO,
           struct fb_fix_screeninfo *fix);
```

#### 【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 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





参数名称	描述	输入/输出
F BIOGET_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。

## 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】

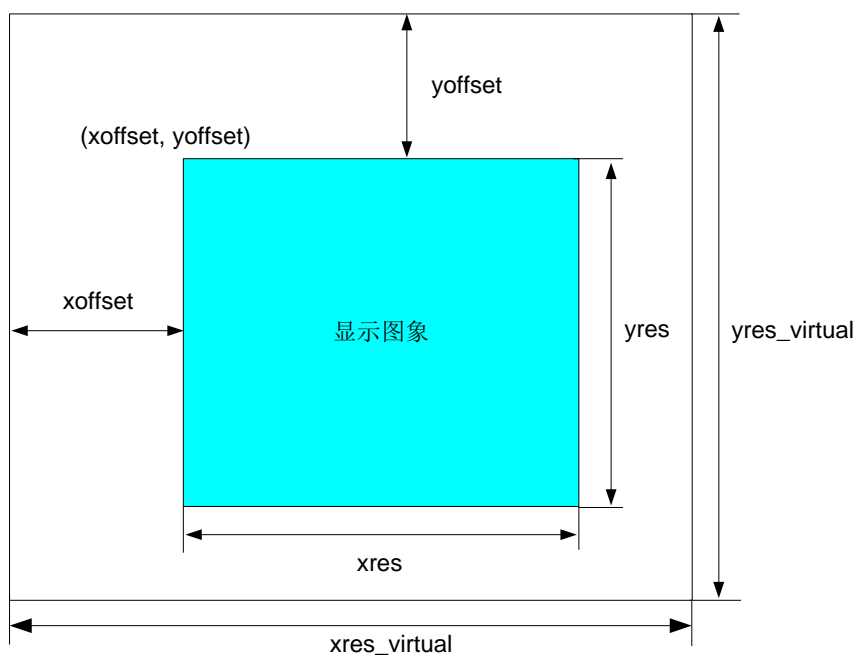
```
int ioctl (int fd,  
           FBIOPAN_DISPLAY,  
           struct fb_var_screeninfo *var);
```

#### 【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】

- 此接口只应在 FB 标准模式中使用，它能把 FB 从扩展模式切换到标准模式。



- 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。另外，最好保证 xoffset 与 yoffset 形成的偏移地址是 16byte 对齐的，否则会将 xoffset 的值减少到能使偏移地址是 16byte 对齐的位置。
- 对于隔行显示设备，要求分辨率的高度必须为偶数。

#### 【举例】

设置实际分辨率为 300x300，虚拟分辨率为 720x576，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)
{
    return -1;
}
vinfo.xoffset = 300;
vinfo.yoffset = 0;
if (ioctl(fd, FBIOPAN_DISPLAY, &vinfo) < 0)
{
    return -1;
}
```



## 2.4 扩展功能

### 2.4.1 通用功能

#### FBIOGET\_CAPABILITY\_HIFB

##### 【目的】

获取叠加层的支持能力。

##### 【语法】

```
int ioctl (int fd,  
           FBIOGET_CAPABILITY_HIFB,  
           HIFB_CAPABILITY_S *pstCap);
```

##### 【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】

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】

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。

## 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】

```
int ioctl (int fd,  
           FBIOGET_SCREEN_ORIGIN_HIFB,  
           HIFB_POINT_S *pstPoint);
```

### 【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】

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】

返回值	描述
0	成功
-1	失败

### 【需求】

头文件：hifb.h

### 【注意】

对软鼠标不适用。

### 【举例】

无。

### 【相关接口】

[FBIOPUT\\_SCREEN\\_ORIGIN\\_HIFB](#)



## 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】

```
int ioctl (int fd,  
           FBIOPUT_SCREEN_ORIGIN_HIFB,  
           HIFB_POINT_S *pstPoint);
```

### 【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减图形层支持的最小分辨率之间。

### 【参数】

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】

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），其中 u32MaxWidth 和 u32MaxHeight 的值是设备时序定义的最大宽高；u32MinWidth 和 u32MinHeight 分别表示可加载的最小图像的宽和高，可通过 FBIOPUT\_SCREEN\_ORIGIN\_HIFB 接口中的 u32MinWidth 和 u32MinHeight 成员获取。
- 对于隔行显示设备，要求坐标原点的纵坐标值为偶数。

### 【举例】

无。



【相关接口】

[FBIOGET\\_SCREEN\\_ORIGIN\\_HIFB](#)

## FBIOGET\_SHOW\_HIFB

【目的】

获取当前叠加层的显示状态。

【语法】

```
int ioctl (int fd,  
           FBIOGET_SHOW_HIFB,  
           HI_BOOL *bShow);
```

【描述】

使用此接口获取当前叠加层显示状态。

【参数】

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： <ul style="list-style-type: none"><li>*bShow = HI_TRUE: 当前叠加层处于显示状态</li><li>*bShow = HI_FALSE: 当前叠加层处于隐藏状态</li></ul>	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】

对软鼠标不适用。

【举例】



无。

#### 【相关接口】

[FBIOPUT\\_SHOW\\_HIFB](#)

## FBIOPUT\_SHOW\_HIFB

#### 【目的】

显示或隐藏该叠加层。

#### 【语法】

```
int ioctl (int fd,  
           FBIOPUT_SHOW_HIFB,  
           HI_BOOL *bShow);
```

#### 【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： <ul style="list-style-type: none"><li>• *bShow = HI_TRUE: 显示当前叠加层</li><li>• *bShow = HI_FALSE: 隐藏当前叠加层</li></ul>	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】

- 为正常显示，在显示之前，应将 bShow 的值设为 HI\_TRUE 调用 ioctl(fd, FBIOPUT\_SHOW\_HIFB, &bShow)，即使能对应图形层。
- 显示时应保证图形层的分辨率不超出设备分辨率。





- 保证显示设备的能力支持所要显示的分辨率。

#### 【举例】

无。

#### 【相关接口】

[FBIOGET\\_SHOW\\_HIFB](#)

## FBIOGET\_MIRROR\_MODE

#### 【目的】

获取当前叠加层的镜像模式。

#### 【语法】

```
int ioctl (int fd,  
           FBIOGET_MIRROR_MODE,  
           HIFB_MIRROR_MODE_E *eMirrorMode);
```

#### 【描述】

使用此接口获取当前叠加层镜像模式。

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_MIRROR_MODE	ioctl 号	输入
eMirrorMode	指示当前叠加层的镜像模式	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】

只用于扩展模式下，HIFB\_LAYER\_BUF\_NONE 模式不支持，对软鼠标不适用。

#### 【举例】



无。

#### 【相关接口】

## FBIOPUT\_MIRROR\_MODE

#### 【目的】

设置当前叠加层的镜像模式。

#### 【语法】

```
int ioctl (int fd,  
           FBIOPUT_MIRROR_MODE,  
           HIFB_MIRROR_MODE_E *eMirrorMode);
```

#### 【描述】

使用此接口获取当前叠加层镜像模式。

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_MIRROR_MODE	ioctl 号	输入
eMirrorMode	叠加层的镜像模式	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】

- 只用于扩展模式下，对软鼠标不适用。
- 不支持镜像模式和压缩同时做。
- 在 HIFB\_LAYER\_BUF\_NONE 刷新模式下不支持镜像操作。

#### 【举例】

无。



## 【相关接口】

### FBIOGET\_ALPHA\_HIFB

#### 【目的】

获取叠加层 Alpha。

#### 【语法】

```
int ioctl (int fd,  
           FBIOGET_ALPHA_HIFB,  
           HIFB_ALPHA_S *pstAlpha);
```

#### 【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](#)



## FBIOPUT\_ALPHA\_HIFB

### 【目的】

设置叠加层的 Alpha。

### 【语法】

```
int ioctl (int fd,  
           FBIOPUT_ALPHA_HIFB,  
           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】

[FBIOPUT\\_ALPHA\\_HIFB](#)

## FBIOPUT\_COLORKEY\_HIFB

### 【目的】



获取叠加层的 colorkey。

#### 【语法】

```
int ioctl (int fd,  
           FBIOGET_COLORKEY_HIFB,  
           HIFB_COLORKEY_S *pstColorKey);
```

#### 【描述】

使用此接口获取叠加层的 colorkey。

#### 【参数】

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】

返回值	描述
0	成功
-1	失败

#### 【差异说明】

芯片	描述
Hi3559V100	只有一个图形层，该图形层支持 colorkey
Hi3559AV100ES	支持两个图形层，图形层 G0 支持 colorkey，图形层 G1 支持 colorkey
Hi3559AV100/Hi3519AV100	支持三个图形层，G0，G1，G3 均支持 colorkey。
Hi3516CV500/Hi3516DV300	只有一个图形层 G0，该图形层支持 colorkey

#### 【需求】

头文件：hifb.h

#### 【注意】

在预乘模式下不支持 colorkey 功能。



【举例】

无。

【相关接口】

[FBIOPUT\\_COLORKEY\\_HIFB](#)

## FBIOPUT\_COLORKEY\_HIFB

【目的】

设置叠加层的 colorkey。

【语法】

```
int ioctl (int fd,  
           FBIOPUT_COLORKEY_HIFB,  
           HIFB_COLORKEY_S *pstColorKey);
```

【描述】

使用此接口设置当前叠加层的 colorkey 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_COLORKEY_HIFB	ioctl 号	输入
pstColorKey	colorkey 结构体指针	输入

【返回值】

返回值	描述
0	成功
-1	失败

【差异说明】

芯片	描述
Hi3559V100	只有一个图形层，该图形层支持 colorkey
Hi3559AV100ES	支持两个图形层，图形层 G0 支持 colorkey，图形层 G1 支持 colorkey
Hi3559AV100/Hi3519AV100	支持三个图形层，G0，G1，G3 均支持 colorkey。



Hi3516CV500/Hi3516DV300	只有一个图形层 G0，该图形层支持 colorkey
-------------------------	----------------------------

#### 【需求】

头文件：hifb.h

#### 【注意】

无。

#### 【举例】

假设当前像素格式为 ARGB8888，则要过滤掉红色分量为 0x1F、绿色分量为 0x2F、蓝色分量为 0x3F 的颜色值，具体设置如下：

```
HIFB_COLORKEY_S stColorKey;

stColorKey.bKeyEnable = HI_TRUE;
stColorKey.u32Key = 0x1F2F3F;
if (ioctl(fd, FBIOPUT_COLORKEY_HIFB, &stColorKey) < 0)
{
    return -1;
}
```

#### 【相关接口】

[FBIOGET\\_COLORKEY\\_HIFB](#)

## FBIOGET\_DEFLICKER\_HIFB

#### 【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_HIFB	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
非 0	失败

【需求】

头文件：hifb.h

【注意】

Hi35xx 芯片不支持抗闪烁操作，所以在 Hi35xx 芯片上调用该接口返回失败。

【举例】

无。

【相关接口】

[FBIOPUT\\_DEFLICKER\\_HIFB](#)

## FBIOPUT\_DEFLICKER\_HIFB

【目的】

设置叠加层的抗闪烁功能。

【语法】

```
int ioctl (int fd,  
           FBIOPUT_DEFLICKER_HIFB,  
           HIFB_DEFLICKER_S *pstDeflicker);
```

【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_HIFB	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】

Hi35xx 芯片不支持抗闪烁操作，所以在 Hi35xx 芯片上调用该接口返回失败。

【举例】

无。

【相关接口】

[FBIOGET\\_DEFLICKER\\_HIFB](#)

## FBIOGET\_VBLANK\_HIFB

【目的】

为了操作显存时不引起撕裂现象，建议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】

```
int ioctl (int fd, FBIOGET_VBLANK_HIFB);
```

【描述】

使用此接口获取当前叠加层的消隐区。

【参数】

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】

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，建议操作时间尽量短，以保证在垂直消隐区结束前完成。

#### 【举例】

无。

#### 【相关接口】

无。

## FBIOFLIP\_SURFACE

#### 【目的】

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

#### 【语法】

```
int ioctl (int fd,  
           FBIOFLIP_SURFACE,  
           HIFB_SURFACEEX_S *pstSurface);
```

#### 【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	输入

#### 【返回值】

返回值	描述
0	成功
-1	失败

#### 【差异说明】



芯片	描述
Hi3559V100	只有一个图形层，该图形层支持 colorkey
Hi3559AV100ES	支持两个图形层，图形层 0 支持 colorkey，图形层 1 支持 colorkey
Hi3559AV100/Hi3519AV100	支持三个图形层，G0，G1，G3 均支持 colorkey。
Hi3516CV500/Hi3516DV300	支持一个图形层 G0，该图形层支持 colorkey

#### 【需求】

头文件：hifb.h。

#### 【注意】

- 此接口只在 FB 标准模式中使用，用于把 FB 从扩展模式切换到标准模式。
- Surface 的物理地址必须在该叠加层配置的显存范围内；而且最好是 16byte 对齐，否则实际上显示的位置与所设置的位置值会有偏差。

#### 【举例】

无。

#### 【相关接口】

[FBIOPAN\\_DISPLAY](#)

## FBIOPUT\_COMPRESSION\_HIFB

#### 【目的】

设置图层启用压缩功能。

#### 【语法】

```
int ioctl (int fd,
           FBIOPUT_COMPRESSION_HIFB,
           HI_BOOL *pbCompress);
```

#### 【描述】

设置图层启动压缩功能。

#### 【参数】

参数名称	描述	输入/输出
Fd	Framebuffer 设备文件描述符	输入
FBIOPUT_COMPRESSION_HIFB	ioctl 号	输入
pbCompress	是否启动压缩功能标识的指针	输入



【返回值】

返回值	描述
0	成功
-1	失败

【差异说明】

芯片	描述
Hi3559V100/Hi3516CV500/Hi3516DV300	不支持压缩功能
Hi3559AV100ES	支持两个图形层，图形层 0 支持压缩功能，图形层 1 支持压缩功能 说明：上述压缩功能由 TDE 完成，解压功能由各图形层完成。
Hi3559AV100	支持三个图形层，图形层 0 支持压缩功能，图形层 1 支持压缩功能，图形层 3 不支持压缩功能。 说明：上述压缩功能由 TDE 完成，解压功能由各图形层完成。
Hi3519AV100	支持三个图形层，图形层 0 支持压缩功能，图形层 1 不支持压缩功能，图形层 3 不支持压缩功能。 说明：上述压缩功能由 TDE 完成，解压功能由各图形层完成。

【需求】

头文件：hifb.h。

【注意】

- 没有设置内存检测区域时，对于启用压缩功能的情况，在绘制完内容后，要调用相应的刷新操作后内容才会真正得以显示出来。其中，FB 标准模式下的刷新操作包括：FBIOPAN\_DISPLAY、FBIOFLIP\_SURFACE；而 FB 扩展模式则包括：FBIO\_REFRESH。另外，通过 FBIOPUT\_SCREEN\_ORIGIN\_HIFB 改变原点坐标位置也会引起刷新操作。
- 设置内存检测区域时（内存检测功能只在 0 buf 模式和标准模式下生效），不需要调用刷新操作，刷新动作由内存检测功能触发。
- 对于 Hi3559AV100ES、Hi3559AV100、Hi3519AV100，压缩功能只针对 ARGB8888、ARGB1555、ARGB4444 格式有效，对其它格式无效。
- 对鼠标层不适用；在启用压缩功能的情况下，不建议使用软鼠标功能。



- 压缩功能默认关闭。
- 对于 Hi3559AV100ES、Hi3559AV100、Hi3519AV100，压缩功能仅支持 HIFB\_LAYER\_BUF\_DOUBLE、HIFB\_LAYER\_BUF\_DOUBLE\_IMMEDIATE 刷新模式的刷新显示。
- 不支持压缩和镜像模式同时做。
- 不支持压缩和旋转同时做。

【举例】

无。

【相关接口】

[FBIOGET\\_COMPRESSION\\_HIFB](#)

## FBIOGET\_COMPRESSION\_HIFB

【目的】

获取图层的压缩功能状态。

【语法】

```
int ioctl (int fd,  
           FBIOGET_COMPRESSION_HIFB,  
           HI_BOOL *pbCompress);
```

【描述】

设置图层是否启动压缩功能。

【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_COMPRESSION_HIFB	ioctl 号	输入
pbCompress	获取状态的指针	-

【返回值】

返回值	描述
0	成功
-1	失败

【差异说明】



芯片	描述
Hi3559V100/Hi3516CV500/Hi3516DV300	不支持压缩功能
Hi3559AV100ES	支持两个图形层，图形层 0 支持压缩功能，图形层 1 支持压缩功能 说明：上述压缩功能由 TDE 完成，解压功能由各图形层完成。
Hi3559AV100	支持三个图形层，图形层 0 支持压缩功能，图形层 1 支持压缩功能，图形层 3 不支持压缩功能。 说明：上述压缩功能由 TDE 完成，解压功能由各图形层完成。
Hi3519AV100	支持三个图形层，图形层 0 支持压缩功能，图形层 1 不支持压缩功能，图形层 3 不支持压缩功能。 说明：上述压缩功能由 TDE 完成，解压功能由各图形层完成。

#### 【需求】

头文件：hifb.h。

#### 【注意】

无。

#### 【举例】

无。

#### 【相关接口】

[FBIOPUT\\_COMPRESSION\\_HIFB](#)

## FBIOPUT\_MDDRDETECT\_HIFB

#### 【目的】

设置图层内存检测属性。

#### 【语法】

```
int ioctl(int fd,  
          FBIOPUT_MDDRDETECT_HIFB,  
          HIFB_DDRZONE_S *stDdrZone);
```

#### 【描述】

设置图层内存检测属性。



#### 【参数】

参数名称	描述	输入/输出
Fd	Framebuffer 设备文件描述符	输入
FBIOPUT_MDDRDETECT_HIFB	ioctl 号	输入
stDdrZone	内存检测属性的指针	输入

#### 【返回值】

返回值	描述
0	成功
-1	失败

#### 【差异说明】

芯片	描述
Hi3559V100/Hi3559AV100ES/Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516DV300	不支持内存侦测

#### 【需求】

头文件：hifb.h。

#### 【注意】

- 内存侦测功能只在 0 buf 模式和标准模式且压缩功能使能条件下生效。
- 压缩功能使能条件下，内存侦测默认打开。内存侦测最多支持同时 32 个内存区域进行变化侦测，G0 默认占用区域 0 至区域 15，G1 默认占用区域 16 至区域 31。
- 内存侦测功能按照内存侦测的区域个数，对显示 buffer 按像素行进行均匀分割，进行内存侦测。
- 当用户设置内存侦测区域数为 0 时，内存侦测功能被关闭。

#### 【举例】

无。

#### 【相关接口】

[FBIOGET\\_COMPRESSION\\_HIFB](#)

FBIOGET\_MDDRDETECT\_HIFB

#### 【目的】



获取图层的内存侦测状态。

#### 【语法】

```
int ioctl (int fd,  
           FBIOGET_MDDRDETECT_HIFB,  
           HIFB_DDRZONE_S *stDdrZone);
```

#### 【描述】

获取图层的内存侦测属性。

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_MDDRDETECT_HIFB	ioctl 号	输入
stDdrZone	获取状态的指针	-

#### 【返回值】

返回值	描述
0	成功
-1	失败

#### 【差异说明】

芯片	描述
Hi3559V100/Hi3559AV100ES/Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516DV300	不支持内存检测功能

#### 【需求】

头文件：hifb.h。

#### 【注意】

无。

#### 【举例】

无。

#### 【相关接口】





## FBIOGET\_MDDRDETECT\_HIFB

## FBIOPUT\_LAYER\_INFO

### 【目的】

设置图层信息，用于完成从 FB 的标准模式到 FB 的扩展模式切换或是 FB 的扩展模式之间的切换，同时能设置扩展模式下的刷新信息。

### 【语法】

```
int ioctl (int fd,  
           FBIOPUT_LAYER_INFO,  
           HIFB_LAYER_INFO_S * pstLayerInfo);
```

### 【描述】

此接口用于设置图层信息，包括刷新模式、抗闪烁级别、屏幕起始点位置、画布分辨率、显存分辨率、屏幕显示分辨率以及是否使能预乘。以上信息的更详细说明见 [HIFB\\_LAYER\\_INFO\\_S](#) 以及 [HIFB\\_LAYER\\_BUF\\_E](#) 的描述。

### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_LAYER_INFO	ioctl 号	输入
pstLayerInfo	图层信息结构体指针	输入

### 【返回值】

返回值	描述
0	成功
-1	失败

### 【差异说明】

芯片	描述
Hi3559V100/Hi3559AV100ES/Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516DV300	<ul style="list-style-type: none"><li>• TDE 支持图形缩放，扩展模式时画布分辨率和显存分辨率可以不相等</li><li>• 所有图形层均支持预乘</li></ul>

### 【需求】



头文件：hifb.h。

#### 【注意】

- 在设置完某项属性后，必须通过设置 pstLayerInfo 的 u32Mask 设置相应的掩码，否则该项设置不会生效；
- 若芯片不支持图形层的缩放，则显存分辨率就是屏幕显示分辨率，改变它们其中之一都会改变最终的显示分辨率，另外要求它们不能大于设备分辨率。
- 对于隔行显示设备，要求显存分辨率与屏幕显示分辨率的高度都必须为偶数。
- 芯片中图层自带的缩放功能可以参考 FBIOPUT\_SCREENSIZE。

#### 【举例】

```
HIFB_LAYER_INFO_S stLayerInfo = {0};
stLayerInfo.BufMode = HIFB_LAYER_BUF_ONE;
stLayerInfo.u32Mask = HIFB_LAYERMASK_BUFMODE;
stLayerInfo.u32DisplayWidth = 360;
stLayerInfo.u32DisplayHeight = 320;
stLayerInfo.s32XPos = 16;
stLayerInfo.s32YPos = 16;
stLayerInfo.u32Mask |= HIFB_LAYERMASK_DISPSIZE | HIFB_LAYERMASK_POS;
s32Ret = ioctl(s32Fd, FBIOPUT_LAYER_INFO, &stLayerInfo);
```

#### 【相关接口】

无。

## FBIOGET\_LAYER\_INFO

#### 【目的】

获取图层信息。

#### 【语法】

```
int ioctl (int fd,
           FBIOGET_LAYER_INFO
           HIFB_LAYER_INFO_S* pstLayerInfo);
```

#### 【描述】

用于获取图层信息，包括刷新模式、抗闪烁级别、屏幕起始点位置、画布分辨率、显存分辨率、屏幕显示分辨率以及是否使能预乘。

#### 【参数】

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



参数名称	描述	输入/输出
pstLayerInfo	图层信息结构体指针	输出

【返回值】

返回值	描述
0	成功
-1	失败

【需求】

头文件：hifb.h。

【注意】

该接口获取到的接口数据 [HIFB\\_LAYER\\_INFO\\_S](#) 中，u32Mask 成员是没有意义的，始终被填充为 HIFB\_LAYERMASK\_BUTT。

【举例】

无。

【相关接口】

无。

## F BIOGET\_CANVAS\_BUFFER

【目的】

获取画布信息。

【语法】

```
int ioctl (int fd,  
           FBIOGET_CANVAS_BUFFER,  
           HIFB_BUFFER_S *pstCanvasBuf)
```

【描述】

获取画布信息。

【参数】

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



参数名称	描述	输入/输出
pstCanvasBuf	画布信息结构体指针	输出

【返回值】

返回值	描述
0	成功
-1	失败

【需求】

头文件：hifb.h。

【注意】

对软鼠标不适用。

【举例】

无。

【相关接口】

无。

## FBIO\_REFRESH

【目的】

用于扩展模式下，刷新显示内容。

【语法】

```
int ioctl (int fd,  
           FBIO_REFRESH,  
           HIFB_BUFFER_S * pstBufInfo);
```

【描述】

扩展模式下,启动刷新操作。

【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIO_REFRESH	ioctl 号	输入
pstBufInfo	HIFB_BUFFER_S 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】

只适用于扩展模式下，对软鼠标不适用。

【举例】

无。

【相关接口】

无。

## FBIO\_WAITFOR\_FREFRESH\_DONE

【目的】

扩展模式下，等待此前启动的刷新操作完成，即刷新的内容显示出来。

【语法】

```
int ioctl (int fd, FBIO_WAITFOR_FREFRESH_DONE)
```

【描述】

等待刷新操作完成。

【参数】

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

【返回值】

返回值	描述
0	成功



返回值	描述
-1	失败

**【需求】**

头文件：hifb.h。

**【注意】**

只适用于扩展模式下，HIFB\_LAYER\_BUF\_NONE 模式不支持，对软鼠标不适用。

**【举例】**

无。

**【相关接口】**

无。

## FBIOPUT\_DYNAMIC\_RANGE\_HIFB

**【目的】**

扩展模式下，设置图层的目标显示动态范围。

**【语法】**

```
int ioctl (int fd, FBIOPUT_DYNAMIC_RANGE_HIFB, HIFB_DYNAMIC_RANGE_E *
enDstDynamicRange);
```

**【描述】**

设置图层的目标显示动态范围。

**【参数】**

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_DYNAMIC_RANGE_HIFB	ioctl 号	输入
enDstDynamicRange	HIFB_DYNAMIC_RANGE_E 类型指针	输入

**【返回值】**

返回值	描述
0	成功



返回值	描述
-1	失败

【差异说明】

芯片	描述
Hi3559AV100ES /Hi3559V100	支持
Hi3519AV100/Hi3516CV500/Hi3516DV300	不支持

【需求】

头文件：hifb.h。

【注意】

只适用于扩展模式下。

【举例】

无。

【相关接口】

[FBIOGET\\_DYNAMIC\\_RANGE\\_HIFB](#)

## FBIOGET\_DYNAMIC\_RANGE\_HIFB

【目的】

扩展模式下，获取图层的目标显示动态范围。

【语法】

```
int ioctl (int fd, FBIOPUT_DYNAMIC_RANGE_HIFB, HIFB_DYNAMIC_RANGE_E *  
enDstDynamicRange);
```

【描述】

获取图层的目标显示动态范围。

【参数】

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



enDstDynamicRange	HIFB_DYNAMIC_RANGE_E 类型指针	输出
-------------------	---------------------------	----

#### 【返回值】

返回值	描述
0	成功
-1	失败

#### 【差异说明】

芯片	描述
Hi3559AV100ES /Hi3559V100	支持
Hi3519AV100/Hi3516CV500/Hi3516DV300	不支持

#### 【需求】

头文件：hifb.h。

#### 【注意】

只适用于扩展模式下。

#### 【举例】

无。

#### 【相关接口】

[FBIOPUT\\_DYNAMIC\\_RANGE\\_HIFB](#)

## FBIOPUT\_SCREENSIZE

#### 【目的】

设置图层在屏幕上的显示大小。

#### 【语法】

```
int ioctl(int fd, FBIOPUT_SCREENSIZE, HIFB_SIZE_S * pstHifbSize);
```

#### 【描述】

设置图层在屏幕上的显示大小。

#### 【参数】





参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_SCREENSIZE	ioctl 号	输入
pstHifbSize	HIFB_SIZE_S 类型指针，宽高要求 2 对齐。	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】

通过设置屏幕显示分辨率，可以对图像进行缩放。如图像大小为 800x600，则设置屏幕显示大小为 1280x720 可将图像放大到 1280x720 显示。

- 对于 Hi3559AV100ES，其中仅 G0 支持缩放，G1 不支持缩放。
- 对于 Hi3559AV100，其中仅 G0 支持缩放，G1、G3 不支持缩放。
- 对于 Hi3519AV100，其中仅 G0 支持缩放，G1、G3 不支持缩放。
- 对于 Hi3516CV500 和 Hi3516DV300，仅 G0 支持缩放。

#### 【举例】

无。

#### 【相关接口】

[FBIOGET\\_SCREENSIZE](#)

## FBIOGET\_SCREENSIZE

#### 【目的】

获取图层在屏幕上的显示大小。

#### 【语法】

```
int ioctl (int fd,
           FBIOPUT_SCREENSIZE,
           HIFB_SIZE_S * pstHifbSize);
```



### 【描述】

获取图层在屏幕上的显示大小。

### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_SCREENSIZE	ioctl 号	输入
pstHifbSize	HIFB_SIZE_S 类型指针	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\\_SCREENSIZE](#)

## FBIOPUT\_ROTATE\_MODE

### 【目的】

扩展模式下，获取图形层的旋转角度。

### 【语法】

```
int ioctl(int fd,
          FBIOPUT_ROTATE_MODE,
          HIFB_ROTATE_MODE_E * penHifbRotGet);
```

### 【描述】

获取图层的旋转角度。



#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_ROTATE_MODE	ioctl 号	输入
penHifbRotGet	HIFB_ROTATE_MODE_E 类型指针	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】

只适用于扩展模式下，HIFB\_LAYER\_BUF\_NONE 模式不支持。

#### 【举例】

无。

#### 【相关接口】

[FBIOPUT\\_ROTATE\\_MODE](#)

## FBIOPUT\_ROTATE\_MODE

#### 【目的】

扩展模式下，设置图层的旋转角度。

#### 【语法】

```
int ioctl (int fd,  
           FBIOPUT_ROTATE_MODE,  
           HIFB_ROTATE_MODE_E * penHifbRotSet);
```

#### 【描述】

设置图层的旋转角度。

#### 【参数】



参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_ROTATE_MODE	ioctl 号	输入
penHifbRotSet	HIFB_ROTATE_MODE_E 类型指针	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】

- 只适用于扩展模式下。
- HIFB\_LAYER\_BUF\_NONE 刷新模式下不支持旋转功能。
- 不支持压缩和旋转同时做。
- 旋转功能的用法：
  - 首先需要将 `struct fb_var_screeninfo` 中成员 `xres`、`yres`（以及 `xres_virtual`、`yres_virtual`）分别设置为旋转后的图像的宽和高；
  - 然后调用 `ioctl (int fd, FBIOPUT_ROTATE_MODE, HIFB_ROTATE_MODE_E * penHifbRotSet)`接口即可；
  - 另外，该功能仅支持 HIFB\_LAYER\_BUF\_ONE、HIFB\_LAYER\_BUF\_DOUBLE 和 HIFB\_LAYER\_BUF\_DOUBLE\_IMMEDIATE 三种刷新模式下（见 [HIFB\\_LAYER\\_BUF\\_E](#)）特定角度（90 度、180 度、270 度）的旋转。

#### 【举例】

无。

#### 【相关接口】

[FBIOGET\\_ROTATE\\_MODE](#)

## FBIO\_RELEASE\_HIFB

#### 【目的】

释放图形层资源。

#### 【语法】



```
int ioctl (int fd, FBIO_RELEASE_HIFB)
```

#### 【描述】

释放图形层资源，关闭图形层的时候使用，主要用于鼠标层。

#### 【参数】

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

- 由于 linux 3.10 以上的内核对 close 机制采用异步方式，这样可能在顺序执行鼠标层关闭和解绑定定时驱动会出现颠倒执行顺序的情况，为确保关闭层和解绑定层能顺序执行，增加此接口，在关闭鼠标层之前调用。
- 执行此接口后，图形层资源会释放，不允许再执行其它 ioctl 接口。

#### 【举例】

无。

#### 【相关接口】

无。

## 2.4.2 软鼠标功能

本小节中描述的接口只有在软鼠标功能启用的情况下才真正生效。如果想启用软鼠标功能，则应在加载 hifb.ko 时把参数 softcursor 置为 “on”。在启用软鼠标功能的情况下，调用 open 函数打开/dev/fb0 后，就可以调用下面的函数进行软鼠标的相关操作（建议只使用下面的函数来使用软鼠标，而不要使用之前介绍的函数）。

## FBIOPUT\_CURSOR\_INFO

#### 【目的】



设置鼠标层的信息。

#### 【语法】

```
int ioctl (int fd, FBIOPUT_CURSOR_INFO, HIFB_CURSOR_S *pstCursor)
```

#### 【描述】

设置鼠标层的信息，包括画布起始地址、大小、跨度以及像素格式。

#### 【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_INFO	ioctl 号	输入
pstCursor	软鼠标层的信息	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】

- 软鼠标的宽与高取值范围为：(0, 128]；
- 软鼠标热点的横坐标与纵坐标都必须大于或等于 0，但不能大于鼠标位图的宽与高。

#### 【举例】

无。

#### 【相关接口】

无。

## FBIOGET\_CURSOR\_INFO

#### 【目的】

获取鼠标层的信息。

#### 【语法】



```
int ioctl (int fd, FBIOGET_CURSOR_INFO, HIFB_CURSOR_S *pstCursor)
```

【描述】

获取鼠标层的信息。

【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_CURSOR_INFO	ioctl 号	输入
pstCursor	软鼠标层的信息	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】

无。

## FBIOPUT\_CURSOR\_ATTCHCURSOR

【目的】

将软鼠标与某一个图形层进行绑定。

【语法】

```
int ioctl (int fd,  
           FBIOPUT_CURSOR_ATTCHCURSOR,  
           HI_U32 *pu32LayerId)
```

【描述】



软鼠标与某一个图形层绑定后，它的内容就通过该图形层显示。

#### 【参数】

参数名称	描述	输入/输出
fd	FB 设备文件描述符	输入
FBIOPUT_CURSOR_ATTCHCURSOR	ioctl 号	输入
pu32LayerId	所要绑定到的图形层标识	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】

- 待绑定的图形层必须处于已打开状态；
- 允许将同一个鼠标与某个图形层绑定多次，但不允许多个鼠标同时绑定到同一图形层；如果一个图形层已与一个鼠标绑定，但想绑定另一个鼠标层，则应先解除之前的绑定关系，否则出错；
- 绑定之前必须设置鼠标层的信息，另外不能将其绑定到其他鼠标层。

#### 【举例】

无。

#### 【相关接口】

无。

## FBIOPUT\_CURSOR\_DETACHCURSOR

#### 【目的】

解除软鼠标与图形层的绑定关系。

#### 【语法】

```
int ioctl (int fd,  
           FBIOPUT_CURSOR_DETACHCURSOR,  
           HI_U32 *pu32LayerId)
```





#### 【描述】

解除软鼠标与图形层的绑定关系后，该鼠标内容将不会显示。

#### 【参数】

参数名称	描述	输入/输出
fd	FB 设备文件描述符	输入
FBIOPUT_CURSOR_DETACHCURSOR	ioctl 号	输入
pu32LayerId	图形层标识	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】

无。

#### 【相关接口】

无。

## FBIOPUT\_CURSOR\_STATE

#### 【目的】

设置软鼠标的显示状态。

#### 【语法】

```
int ioctl(int fd, FBIOPUT_CURSOR_STATE, HI_BOOL *pbShow)
```

#### 【描述】

设置软鼠标的显示状态。

#### 【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_STATE	ioctl 号	输入
pbShow	显示状态标识指针	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】

软鼠标在绑定之后，默认是处于隐藏状态，所以必须要调用此接口将其置于显示状态它才能显示出来。

【举例】

无。

【相关接口】

无。

## FBIOPUT\_CURSOR\_STATE

【目的】

获取软鼠标的显示状态。

【语法】

```
int ioctl (int fd, FBIOPUT_CURSOR_STATE, HI_BOOL *pbShow)
```

【描述】

获取软鼠标的显示状态。

【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_STATE	ioctl 号	输入
pbShow	显示状态标识指针	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】

软鼠标默认是处于隐藏状态。

【举例】

无。

【相关接口】

无。

## FBIOPUT\_CURSOR\_POS

【目的】

设置软鼠标在所绑定图形层上的显示位置。

【语法】

```
int ioctl (int fd, FBIOPUT_CURSOR_POS, HIFB_POINT_S *pstPos)
```

【描述】

设置软鼠标在所绑定图形层上的显示位置。

【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_POS	ioctl 号	输入
pstPos	显示位置信息	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】

无。

【相关接口】

无。

## FBIOGET\_CURSOR\_POS

【目的】

获取软鼠标在所绑定图形层上的显示位置。

【语法】

```
int ioctl (int fd, FBIOGET_CURSOR_POS, HIFB_POINT_S *pstPos)
```

【描述】

获取软鼠标在所绑定图形层上的显示位置。

【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_POS	ioctl 号	输入
pstPos	显示位置信息	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】

无。

## FBIOPUT\_CURSOR\_COLORKEY

【目的】

设置软鼠标的 colorkey 信息。

【语法】

```
int ioctl (int fd, FBIOPUT_CURSOR_COLORKEY, HIFB_COLORKEY_S *  
pstColorKey)
```

【描述】

设置软鼠标的 colorkey 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
FBIOPUT_CURSOR_COLORKEY	ioctl 号	输入
pstColorKey	colorkey 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】

无。

【举例】

无。

【相关接口】

无。

## FBIOGET\_CURSOR\_COLORKEY

【目的】

获取软鼠标的 colorkey 信息。

【语法】

```
int ioctl (int fd, FBIOGET_CURSOR_COLORKEY, HIFB_COLORKEY_S *  
pstColorKey)
```

【描述】

获取软鼠标的 colorkey 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
FBIOGET_CURSOR_COLORKEY	ioctl 号	输入
pstColorKey	colorkey 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】



无。

【举例】

无。

【相关接口】

无。

## FBIOPUT\_CURSOR\_ALPHA

【目的】

设置软鼠标的 alpha 叠加信息。

【语法】

```
int ioctl (int fd, FBIOPUT_CURSOR_ALPHA, HIFB_ALPHA_S *pstAlphaInfo)
```

【描述】

设置软鼠标的 alpha 叠加信息。

【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_CURSOR_ALPHA	ioctl 号	输入
pstAlphaInfo	alpha 叠加信息	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】

无。

【相关接口】



无。

## FBIOGET\_CURSOR\_ALPHA

### 【目的】

获取软鼠标的 alpha 叠加信息。

### 【语法】

```
int ioctl (int fd, FBIOGET_CURSOR_ALPHA, HIFB_ALPHA_S *pstAlphaInfo)
```

### 【描述】

获取软鼠标的 alpha 叠加信息。

### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOGET_CURSOR_ALPHA	ioctl 号	输入
pstAlphaInfo	alpha 叠加信息	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】

无。





## 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

struct fb\_bitfield

### 【说明】

位域信息，用于设置像素格式。

### 【定义】

```
struct fb_bitfield
{
    __u32 offset;          /* beginning of bitfield */
    __u32 length;          /* length of bitfield */
    __u32 msb_right;       /* != 0: Most significant bit is right */
};
```

### 【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。

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;
    __u32 xoffset;              /* offset from virtual to visible */
    __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 */

    __u32 nonstd;               /* != 0 Non standard pixel format */

    __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 */
    __u32 upper_margin;          /* time from sync to picture */
    __u32 lower_margin;
    __u32 hsync_len;             /* length of horizontal sync */
    __u32 vsync_len;             /* length of vertical sync */
    __u32 sync;                  /* see FB_SYNC_* */
}
```



```
__u32 vmode; /* see FB_VMODE_* */
__u32 rotate; /* angle we rotate counter clockwise */
__u32 reserved[5]; /* Reserved for future compatibility */
};
```

### 【成员】

成员名称	描述	支持情况
xres	可见屏幕宽度（像素数）。	支持，fb0，fb1 默认值为 1280；fb2，fb3 默认值为 720。 对于 Hi3559AV100ES、Hi3559AV100、Hi3519AV100、Hi3516CV500、Hi3516DV300，fb0，fb1 默认为 1920
yres	可见屏幕高度（像素数）。	支持。fb0，fb1 默认为 720；fb2，fb3 默认值为 576。 对于 Hi3559AV100ES、Hi3559AV100、Hi3519AV100、Hi3516CV500、Hi3516DV300，fb0，fb1 默认为 1080
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，fb0，fb1 默认值为 1280；fb2，fb3 默认值为 720。 对于 Hi3559AV100ES、Hi3559AV100、Hi3519AV100、Hi3516CV500、Hi3516DV300，fb0，fb1 默认为 1920
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，fb0，fb1 默认为 720；fb2，fb3 默认值为 576。 对于 Hi3559AV100ES、Hi3559AV100、Hi3519AV100、Hi3516CV500、Hi3516DV300，fb0，fb1 默认为 1080
xoffset	在 x 方向上的偏移像素数。	支持，默认为 0。
yoffset	在 y 方向上的偏移像素数。	支持，默认为 0。
bits_per_pixel	每个像素所占的比特数。	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）。



成员名称	描述	支持情况
transp	颜色分量中 alpha 分量的位域信息。	支持，默认为 (15, 1, 0)。
nonstd	是否为标准像素格式。	不支持，缺省值为 0，表示支持标准像素格式。
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。
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、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。	不支持，缺省值为-1。
right_margin		
hsync_len		
upper_margin	分别是上消隐信号、下消隐信号、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。	不支持，缺省值为-1。
lower_margin		
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。

#### 【注意】

- 高清设备图形层默认分辨率为 1280x720；标清设备图形层默认分辨率为 720x576，鼠标层默认分辨率为 128x128。像素格式为 ARGB1555。
- 特别说明：对于 Hi3559AV100ES、Hi3559AV100、Hi3519AV100、Hi3516CV500、Hi3516DV300 芯片，超清和高清设备的图形层默认分辨率为 1920x1080。

#### 【相关数据类型及接口】

- [struct fb\\_bitfield](#)
- [FBIOGET\\_VSCREENINFO](#)



- FBIOPUT\_VSCREENINFO

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

### 【成员】

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_PIXELS，表示像素值紧密排列。
type_aux	附加类型。	不支持，在 FB_TYPE_PACKED_PIXELS 显卡类型下无含义。



成员名称	描述	支持情况
visual	色彩模式。	不支持，默认为 FB_VISUAL_TRUECOLOR，真彩色。
xpanstep	支持水平方向上的 PAN 显示： <ul style="list-style-type: none"><li>• 0：不支持。</li><li>• 非 0：支持，此时该值用于表示在水平方向上每步进的像素值。</li></ul>	固定为 1。
ypanstep	支持垂直方向上的 PAN 显示： <ul style="list-style-type: none"><li>• 0：不支持。</li><li>• 非 0：支持，此时该值用于表示在垂直方向上每步进的像素值。</li></ul>	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。
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。

【注意】

无。

【相关数据类型及接口】

[FBIOGET\\_FSCREENINFO](#)

## 3.2 扩展的数据类型

### HIFB\_ROTATE\_MODE\_E

【说明】

HiFB 支持的图形层旋转角度枚举。

【定义】



```
typedef enum
{
    HIFB_ROTATE_NONE = 0x0,
    HIFB_ROTATE_90 = 0x1,
    HIFB_ROTATE_180 = 0x2,
    HIFB_ROTATE_270 = 0x3,
    HIFB_ROTATE_BUTT
} HIFB_ROTATE_MODE_E;
```

#### 【成员】

成员名称	描述
HIFB_ROTATE_NONE	旋转角度：0 度。
HIFB_ROTATE_90	旋转角度：90 度。
HIFB_ROTATE_180	旋转角度：180 度。
HIFB_ROTATE_270	旋转角度：270 度。
HIFB_ROTATE_BUTT	非法旋转角度。

#### 【注意】

无。

#### 【相关数据类型及接口】

无。

## HIFB\_DYNAMIC\_RANGE\_E

#### 【说明】

HiFB 支持的图形层动态范围类型。

#### 【定义】

```
typedef enum hifbDYNAMIC_RANGE_E
{
    HIFB_DYNAMIC_RANGE_SDR8 = 0,
    HIFB_DYNAMIC_RANGE_SDR10,
    HIFB_DYNAMIC_RANGE_HDR10,
    HIFB_DYNAMIC_RANGE_HLG,
    HIFB_DYNAMIC_RANGE_SLF,
    HIFB_DYNAMIC_RANGE_BUTT
} HIFB_DYNAMIC_RANGE_E;
```

#### 【成员】





成员名称	描述
HIFB_DYNAMIC_RANGE_SDR8	动态范围类型：SDR8。
HIFB_DYNAMIC_RANGE_SDR10	动态范围类型：SDR10。
HIFB_DYNAMIC_RANGE_HDR10	动态范围类型：HDR10。
HIFB_DYNAMIC_RANGE_HLG	动态范围类型：HLG。
HIFB_DYNAMIC_RANGE_SLF	动态范围类型：SLF。

**【注意】**

图形层动态范围仅支持以下种类：SDR8、SDR10 和 HDR10。

**【相关数据类型及接口】**

无。

## HIFB\_COLOR\_FMT\_E

**【说明】**

HiFB 支持的像素格式集合。

**【定义】**

```
typedef enum
{
    HIFB_FMT_1BPP = 0,          /* 1bpp */
    HIFB_FMT_2BPP,              /* 2bpp */
    HIFB_FMT_4BPP,              /* 4bpp */
    HIFB_FMT_8BPP,              /* 8bpp */
    HIFB_FMT_KRGB444,           /* KRGB444 */
    HIFB_FMT_KRGB555,           /* KRGB555 */
    HIFB_FMT_RGB565,            /* RGB565 */
    HIFB_FMT_ARGB4444,          /* ARGB4444 */
    HIFB_FMT_ARGB1555,          /* ARGB1555 */
    HIFB_FMT_KRGB888,           /* KRGB888 */
    HIFB_FMT_ARGB8888,          /* ARGB8888 */
    HIFB_FMT_BUTT
}HIFB_COLOR_FMT_E;
```

**【成员】**

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	非法像素格式。

【注意】

无。

【相关数据类型及接口】

无。

## HIFB\_CAPABILITY\_S

【说明】

各个叠加层的支持能力。

【定义】

```
typedef struct
{
    HI_BOOL bKeyRgb;
    HI_BOOL bKeyAlpha;           /* whether support colorkey alpha */
    HI_BOOL bGlobalAlpha;       /* whether support global alpha */
    HI_BOOL bCmap;               /* whether support color map */
    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 */

HI_BOOL  bDcmp;
HI_BOOL  bPreMul;
HI_BOOL  bGHDR          /* NEW Feature. Is GHDR supported. */
}HIFB_CAPABILITY_S;

```

### 【成员】

成员名称	描述
bKeyRgb	颜色分量是否可进行 colorkey 操作。
bKeyAlpha	是否支持带 Alpha 的 colorkey。
bGlobalAlpha	是否支持全局 Alpha 和像素 Alpha 叠加。
bCmap	是否支持调色板模式。
bColFmt	支持的像素格式。 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。
bDcmp	是否支持压缩模式。
bPreMul	是否支持预乘模式。
bGHDR	是否支持图形层高动态范围设置。

### 【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 和像素 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\\_DYNAMIC\\_RANGE\\_E](#)
- [F BIOGET\\_CAPABILITY\\_HIFB](#)

## HIFB\_POINT\_S

### 【说明】

坐标结构体。

### 【定义】

```
typedef struct
{
    HI_U32 u32PosX;          /* horizontal position */
    HI_U32 u32PosY;          /* vertical position */
}HIFB_POINT_S;
```

### 【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】

- [F BIOGET\\_SCREEN\\_ORIGIN\\_HIFB](#)
- [F BIOPUT\\_SCREEN\\_ORIGIN\\_HIFB](#)

## HIFB\_MIRROR\_MODE\_E

### 【说明】

镜像模式枚举。

### 【定义】

```
typedef enum
{
    HIFB_MIRROR_NONE = 0x0,
    HIFB_MIRROR_HORIZONTAL = 0x1,
    HIFB_MIRROR_VERTICAL = 0x2,
    HIFB_MIRROR_BOTH= 0x3,
    HIFB_MIRROR_BUTT
}HIFB_MIRROR_MODE_E;
```



#### 【成员】

成员名称	描述
HIFB_MIRROR_NONE	不进行镜像操作。
HIFB_MIRROR_HORIZONTAL	水平方向镜像。
HIFB_MIRROR_VERTICAL	垂直方向镜像。
HIFB_MIRROR_BOTH	水平和垂直方向都做镜像操作。
HIFB_MIRROR_BUTT	非法的镜像模式。

#### 【注意】

不支持镜像模式和压缩同时做。

#### 【相关数据类型及接口】

[FBIOGET\\_MIRROR\\_MODE](#)

## HIFB\_ALPHA\_S

#### 【说明】

Alpha 结构体。

#### 【定义】

```
typedef struct
{
    HI_BOOL bAlphaEnable;        /* alpha enable flag */
    HI_BOOL bAlphaChannel;       /* alpha channel enable flag */
    HI_U8 u8Alpha0;              /* alpha0 value */
    HI_U8 u8Alpha1;              /* alpha1 value */
    HI_U8 u8GlobalAlpha;         /* global alpha value */
    HI_U8 u8Reserved;
}HIFB_ALPHA_S;
```

#### 【成员】

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。



成员名称	描述
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 0~255，默认为 255。在 Alpha 通道使能时起作用。
u8Reserved	保留。

#### 【注意】

只有在 Alpha 叠加使能的情况下才进行 Alpha 叠加，否则处于上层的叠加层将覆盖下层的叠加层。

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 和像素 Alpha 叠加的芯片，叠加 Alpha 值的计算公式如下所示： $\alpha = u8GlobalAlpha$
  - 对于支持全局 Alpha 和像素 Alpha 叠加的芯片，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： $\alpha = \alpha_{pixel}$

#### 【相关数据类型及接口】

- [FBIOGET\\_ALPHA\\_HIFB](#)
- [FBIOPUT\\_ALPHA\\_HIFB](#)

## HIFB\_COLORKEY\_S

#### 【说明】

HiFB\_COLORKEY\_S 结构体，用于 colorkey 的属性设置。

#### 【定义】

```
typedef struct
{
    HI_BOOL bKeyEnable;          /* colorkey 是否使能 */
    HI_U32 u32Key;
}HIFB_COLORKEY_S;
```

#### 【成员】



成员	描述
bKeyEnable	Colorkey 是否使能标识。 TRUE: 使能; FALSE: 不使能。
u32Key	Colorkey 的颜色值。

【注意】

无。

【相关数据类型及接口】

- [FBIOGET\\_COLORKEY\\_HIFB](#)
- [FBIOPUT\\_COLORKEY\\_HIFB](#)

## HIFB\_DEFLICKER\_S

【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】

成员	描述
u32HDfLevel	水平抗闪烁阶数。
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。

【注意】

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](#)

## HIFB\_SURFACEEX\_S

【说明】

Surface 结构体，用于双缓冲时两块 Surface 的属性设置。

【定义】

```
typedef struct
{
    HI_U64 u64PhyAddr;
    HIFB\_ALPHA\_S stAlpha;
    HIFB\_COLORKEY\_S stColorkey;
}HIFB_SURFACEEX_S;
```

【成员】

成员	描述
u64PhyAddr	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层配置的显存范围内，而且最好保证是 16byte 对齐。

【相关数据类型及接口】

[FBIOFLIP\\_SURFACE](#)

## HIFB\_LAYER\_INFO\_S

【说明】

图层信息结构体。

【定义】

```
typedef struct
{
    HIFB\_LAYER\_BUF\_E BufMode;
    HIFB\_LAYER\_ANTIFLICKER\_LEVEL\_E eAntiflickerLevel;
    HI_S32 s32XPos;           /**< the x pos of origion point in screen */
    HI_S32 s32YPos;           /**< the y pos of origion point in screen */
}
```





```

HI_U32 u32CanvasWidth;    /**< the width of canvas buffer */
HI_U32 u32CanvasHeight;   /**< the height of canvas buffer */
HI_U32 u32DisplayWidth;   /**< the width of display buf in fb */
HI_U32 u32DisplayHeight;  /**< the height of display buf in fb. */
HI_U32 u32ScreenWidth;    /**< the width of screen */
HI_U32 u32ScreenHeight;   /**< the height of screen */
HI_BOOL bPreMul;          /**< The data drawn in buf is premul data or
not*/
HI_U32 u32Mask;           /**< param modify mask bit*/
}HIFB_LAYER_INFO_S;

```

**【成员】**

成员	描述
BufMode	扩展模式下的刷新模式。
eAntiflickerLevel	图层抗闪烁等级。
s32XPos	图层在屏幕上的原点横坐标。
s32YPos	图层在屏幕上的原点纵坐标。
u32CanvasWidth	画布 buffer 的宽。
u32CanvasHeight	画布 buffer 的高。
u32DisplayWidth	显存分辨率的宽。要求 2 对齐。
u32DisplayHeight	显存分辨率的高。要求 2 对齐。
u32ScreenWidth	屏幕显示分辨率的宽。要求 2 对齐。
u32ScreenHeight	屏幕显示分辨率的高。要求 2 对齐。
bPreMul	FB 中的数据是否为预乘数据。
u32Mask	设置图层信息时参数修改掩码位。

**【注意】**

- 若芯片不支持图形层的缩放，则显存分辨率就是屏幕显示分辨率，改变它们其中之一都会改变最终的显示分辨率，另外要求它们不能大于设备分辨率。  
Hi3559V100/Hi3556V100 的显示设备均不支持图形层的缩放。
- 对于 Hi3559AV100ES、Hi3559AV100、Hi3519AV100、Hi3516CV500、Hi3516DV300，显示设备中仅 G0 支持图形层的缩放，缩放最大支持放大倍数 15，不支持图形层缩小。
- 图形缩放功能的使用方法：上述 Canvas 宽高（u32CanvasWidth, u32CanvasHeight）、Display 宽高（u32DisplayWidth, u32DisplayHeight）和 Screen 宽高（u32ScreenWidth, u32ScreenHeight）之间的关系：



- Canvas 宽高表示实际要显示的内容的宽高，即未经缩放的图像的分辨率；
- Display 宽高表示显示缓冲区的宽高；
- Screen 宽高表示最终显示出来的宽高；
- 从 Canvas 宽高到 Display 宽高使用 TDE 的搬移和缩放功能，将画布内容搬移并缩放到显存；
- 从 Canvas 宽高到 Display 宽高且为放大时，需要将 `struct fb_var_screeninfo` 中成员 `xres` 和 `yres`（以及 `xres_virtual` 和 `yres_virtual`）分别设置为放大后的图像的宽和高；
- 从 Display 宽高到 Screen 宽高使用 G0 中的缩放功能，将显存内容缩放到 Screen 宽高显示。
- Hi3559AV100 当 G0 的缩放输入宽度超过 3840，Hi3519AV100/ Hi3516CV500/ Hi3516DV300 当 G0 的缩放输入宽度大于 1920 时，不会有缩放效果。



### 注意

- 上述缩放功能中，TDE 的缩放能力（放大倍数和缩小倍数）可参考文档《TDE API 参考》。
- 如果仅使用 FBIOPAN\_DISPLAY 接口进行显示，则不涉及 TDE 搬移，在使用 G0 的缩放功能时，HiFB 内部以 Display 宽高作为参考，此时需将 Display 宽高设置成 Canvas 宽高。
- 若设置了 `u32DisplayWidth` 和 `u32Mask`（修改显示宽高的掩码设置参见 [HIFB\\_LAYER\\_INFO\\_MASKBIT](#) 中的 `HIFB_LAYERMASK_DISPSIZE` 项）且该宽度值比系统已存在的设置大，则该设置会修改固定屏幕信息 `struct fb_fix_screeninfo` 中的 `line_length` 项，修改后的大小为“设置的宽度值\*每像素字节数”的 16 字节对齐。
- 若设置了 `u32DisplayWidth`、`u32DisplayHeight` 和 `u32Mask`（修改显示宽高的掩码设置参见 [HIFB\\_LAYER\\_INFO\\_MASKBIT](#) 中的 `HIFB_LAYERMASK_DISPSIZE` 项），则该设置会同步修改 `struct fb_var_screeninfo` 中的 `xres` 和 `yres`；
  - 如果 `u32DisplayWidth` 大于 `xres_virtual`，则设置 `xres_virtual` 为 `u32DisplayWidth`；
  - 如果 `u32DisplayHeight` 大于 `yres_virtual`，则设置 `yres_virtual` 为 `u32DisplayHeight`。
- 当图形层的像素格式为 ARGB1555 或 ARGB4444 时，不支持预乘模式。
- 当图形层全局 `alpha` 为 1 时，不支持预乘模式。
- 当 `colorkey` 使能时，不支持预乘模式。

### 【相关数据类型及接口】

- [FBIOPUT\\_LAYER\\_INFO](#)
- [FBIOGET\\_LAYER\\_INFO](#)



## HIFB\_LAYER\_ANTIFLICKER\_LEVEL\_E

### 【说明】

图层抗闪等级。

### 【定义】

```
typedef enum
{
    HIFB_LAYER_ANTIFLICKER_NONE = 0x0, /**< no antiflicker*/
    HIFB_LAYER_ANTIFLICKER_LOW = 0x1,  /**< low level*/
    HIFB_LAYER_ANTIFLICKER_MIDDLE = 0x2,/**< middle level*/
    HIFB_LAYER_ANTIFLICKER_HIGH = 0x3, /**< high level*/
    HIFB_LAYER_ANTIFLICKER_AUTO = 0x4, /**< auto*/
    HIFB_LAYER_ANTIFLICKER_BUTT
}HIFB_LAYER_ANTIFLICKER_LEVEL_E;
```

### 【成员】

成员	描述
HIFB_LAYER_ANTIFLICKER_NONE	不做抗闪
HIFB_LAYER_ANTIFLICKER_LOW	低等级抗闪
HIFB_LAYER_ANTIFLICKER_MIDDLE	中等级抗闪
HIFB_LAYER_ANTIFLICKER_HIGH	高等级抗闪
HIFB_LAYER_ANTIFLICKER_AUTO	自动抗闪
HIFB_LAYER_ANTIFLICKER_BUTT	无效值

### 【注意】

如果不设置，则默认为自动抗闪。

### 【相关数据类型及接口】

- [FBIOPUT\\_LAYER\\_INFO](#)
- [FBIOGET\\_LAYER\\_INFO](#)

## HIFB\_LAYER\_BUF\_E

### 【说明】

图层刷新类型。

### 【定义】

```
typedef enum
{
```



```
HIFB_LAYER_BUF_DOUBLE = 0x0,  
HIFB_LAYER_BUF_ONE    = 0x1,  
HIFB_LAYER_BUF_NONE    = 0x2,  
HIFB_LAYER_BUF_DOUBLE_IMMEDIATE=0x3,  
HIFB_LAYER_BUF_BUTT  
} HIFB_LAYER_BUF_E;
```

#### 【成员】

成员	描述
HIFB_LAYER_BUF_DOUBLE	2 buffer 模式
HIFB_LAYER_BUF_ONE	1 buffer 模式
HIFB_LAYER_BUF_NONE	0 buffer 模式
HIFB_LAYER_BUF_DOUBLE_IMMEDIATE	2 buffer 立即模式
HIFB_LAYER_BUF_BUTT	无效值

注：各刷新类型的含义具体见《HiFB 开发指南》1.2 中的“图形层刷新类型”小节。

#### 【注意】

- 绘图内容从用户绘制 buffer 到显示 buffer 的过程用 TDE 进行搬移，因此是否支持缩放取决于 TDE；而从显示 buffer 到显示设备的过程是否支持缩放取决于 VO 设备。Hi3559V100/Hi3556V100 的 VO 设备均不支持缩放，所以显存分辨率与屏幕显示分辨率总是相同。
- Hi3559AV100ES、Hi3559AV100、Hi3519AV100、Hi3516CV500、Hi3516DV300 的 VO 设备仅 G0 支持缩放，G1 不支持缩放。对于 G1，显存分辨率与屏幕显示分辨率总是相同。
- HIFB\_LAYER\_BUF\_DOUBLE 与 HIFB\_LAYER\_BUF\_DOUBLE\_IMMEDIATE 的区别在于 HIFB\_LAYER\_BUF\_DOUBLE\_IMMEDIATE 方式的每一次刷新操作都要等到刷新内容真正显示出来之后才会返回，而 HIFB\_LAYER\_BUF\_DOUBLE 方式则不会等待。

#### 【相关数据类型及接口】

- [FBIOPUT\\_LAYER\\_INFO](#)
- [FBIOGET\\_LAYER\\_INFO](#)

## HIFB\_LAYER\_INFO\_MASKBIT

#### 【说明】

标识 HIFB\_LAYER\_INFO\_S 中哪些成员有更新。

#### 【定义】

```
typedef enum  
{
```



```

HIFB_LAYERMASK_BUFMODE = 0x1,
HIFB_LAYERMASK_ANTIFLICKER_MODE = 0x2,
HIFB_LAYERMASK_POS = 0x4,
HIFB_LAYERMASK_CANVASSIZE = 0x8,
HIFB_LAYERMASK_DISPSIZE = 0x10,
HIFB_LAYERMASK_SCREENSIZE = 0x20,
HIFB_LAYERMASK_BMUL = 0x40,
HIFB_LAYERMASK_BUTT
}HIFB_LAYER_INFO_MASKBIT;

```

**【成员】**

成员	描述
HIFB_LAYERMASK_BUFMODE	HIFB_LAYER_INFO_S 中 buf 模式是否有效掩码
HIFB_LAYERMASK_ANTIFLICKER_MODE	抗闪烁模式是否有效掩码
HIFB_LAYERMASK_POS	图层位置是否有效掩码
HIFB_LAYERMASK_CANVASSIZE	canvassize 是否有效掩码
HIFB_LAYERMASK_DISPSIZE	displaysize 是否有效掩码
HIFB_LAYERMASK_SCREENSIZE	screensize 是否有效掩码
HIFB_LAYERMASK_BMUL	预乘是否有效掩码
HIFB_LAYERMASK_BUTT	无效值

**【注意】**

在设置完某项属性之后必须设置相应的掩码，否则该项设置不会生效。

**【相关数据类型及接口】**

- [FBIOPUT\\_LAYER\\_INFO](#)
- [FBIOGET\\_LAYER\\_INFO](#)

**HIFB\_BUFFER\_S****【说明】**

图形层画布信息及更新区域，用于绘制与刷新。

**【定义】**

```

typedef struct
{
    HIFB_SURFACE_S stCanvas;
    HIFB_RECT UpdateRect;    /* refresh region*/
}

```



```
}HIFB_BUFFER_S;
```

### 【成员】

成员	描述
stCanvas	图形层的画布信息。
UpdateRect	图形层的更新区域。

### 【注意】

无。

### 【相关数据类型及接口】

- [FBIO\\_REFRESH](#)
- [FBIOGET\\_CANVAS\\_BUFFER](#)

## HIFB\_SURFACE\_S

### 【说明】

Surface 结构体，设置双缓冲时两块 Surface 的属性。

### 【定义】

```
typedef struct
{
    HI_U64  u64PhyAddr;    /**< start physical address */
    HI_U64  u64GBPhyAddr;  /**< Compressed data part GB's physical
address */
    HI_U32  u32Width;      /**< width pixels */
    HI_U32  u32Height;     /**< height pixels */
    HI_U32  u32Pitch;      /**< line pixels */
    HIFB_COLOR_FMT_E enFmt; /**< color format */
    HIFB_DYNAMIC_RANGE_E enDynamicRange; /**< destination dynamic range.
*/
}HIFB_SURFACE_S;
```

### 【成员】

成员	描述
u64PhyAddr	Surface 的物理地址（非压缩数据时表示整块 surface 的物理地址，压缩数据时表示 AR 部分物理地址）。
u64GBPhyAddr	Surface 的物理地址（压缩数据时表示 GB 部分物理地址）。
u32Width	Surface 的宽度属性。



u32Height	Surface 的高度属性。
u32Pitch	存储区域一行的跨度属性。
enFmt	像素的格式属性。
enDynamicRange	Surface 的动态范围。

【注意】

无。

【相关数据类型及接口】

- [HIFB\\_BUFFER\\_S](#)
- [HIFB\\_CURSOR\\_S](#)

## HIFB\_CURSOR\_S

【说明】

Cursor 结构体，包含软鼠标信息。

【定义】

```
typedef struct
{
    HIFB_SURFACE_S stCursor;
    HIFB_POINT_S stHotPos;
} HIFB_CURSOR_S;
```

【成员】

成员	描述
stCursor	软鼠标的画布信息。
stHotPos	软鼠标的热点位置。

【注意】

软鼠标热点的定义：在通过 FBIOPUT\_CURSOR\_POS 指定在图形层的偏移位置时，以软鼠标位图中的某一点而不是以起始点（0，0）来做偏移操作，这一点就被称为热点。它的横坐标与纵坐标都必须大于或等于 0，但不能大于鼠标位图的宽与高。

【相关数据类型及接口】

- [FBIOPUT\\_CURSOR\\_INFO](#)
- [FBIOGET\\_CURSOR\\_INFO](#)



## HIFB\_DDRZONE\_S

### 【说明】

内存侦测结构体，包含内存侦测设置的开始区域和区域数信息。

### 【定义】

```
typedef struct
{
    HI_U32 u32StartSection;
    HI_U32 u32ZoneNums;
} HIFB_DDRZONE_S;
```

### 【成员】

成员	描述
u32StartSection	内存侦测开始区域。
u32ZoneNums	内存侦测区域数。

### 【注意】

内存侦测区域数最大只有 32 个区域，设置的开始区域与区域数之和不能超过 32。

### 【相关数据类型及接口】

- [FBIOPUT\\_MDDRDETECT\\_HIFB](#)
- [FBIOGET\\_MDDRDETECT\\_HIFB](#)





# 4 图形开发辅助接口

## 4.1 概述

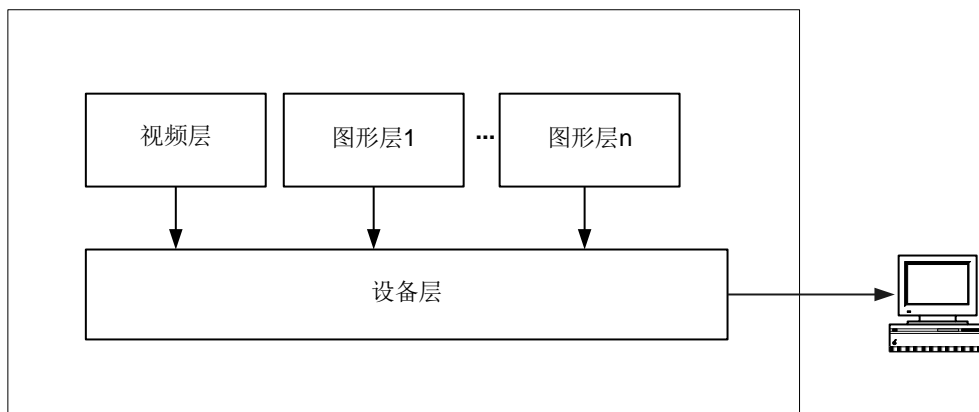
### 4.1.1 简介

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。
- Hi3559AV100ES 芯片一个设备上固定绑定一个视频层和一个图形层。  
当视频层和图形层同时存在时，这两个层的数据会发生叠加，叠加后的数据显示在设备上。
- Hi3559AV100/Hi3519AV100 芯片一个设备上固定绑定一个视频层和一个图形层，此外，图形层 G3 可动态绑定到其中一个设备上。
- Hi3516CV500/Hi3516DV300 芯片一个设备上固定绑定一个视频层和一个图形层。



图4-1 视频输出单元基本结构



## 4.1.2 注意事项

开发图形层时需要注意以下事项：

### 如何看到图形层输出

图形层要在显示设备上正常显示，必须在 `open("/dev/fbn")` 之前先配置并开启设备层。

设备层关闭后，必须对已经执行 `fd=open("/dev/fbn")` 的节点，进行 `close(fd)` 操作，否则，如果设备分辨率发生变化，下一次 `open("/dev/fbn")` 时将不会更新设备分辨率到 `hifb` 而显示异常。

每个显示设备都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。用户需要调用相关接口使能设备层，然后操作图形层，才能看到显示结果。

SDK 使用 VO 模块控制设备层。SDK 的 VO 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`。

注：Hi3559V100 芯片使用“VOU 模块”进行描述，

Hi3559AV100ES/Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516DV300 芯片使用“VO 模块”进行描述。



说明

VO 模块接口详见《HiMPP IPC Vx.0 媒体处理软件开发参考》或《HiMPP V4.0 媒体处理软件开发参考》的“4.视频输出”章节。

### 如何在不同设备间切换图形层

不同芯片在不同设备间切换图形层如表 4-1 所示。



说明

切换之前不需要关闭显示设备，但应先关闭相应的图形层并进行解绑定。



表4-1 不同芯片在不同设备间切换图形层

芯片	描述
Hi3559V100	支持 1 个图形层，G0 G0 层固定绑定在 DSD0 上
Hi3559AV100ES	支持 2 个图形层：G0，G1 G0 层固定绑定在设备 DHD0 上 G1 层固定绑定在设备 DHD1 上
Hi3559AV100/Hi3519AV100	支持 3 个图形层：G0，G1，G3 G0 层固定绑定在设备 DHD0 上 G1 层固定绑定在设备 DHD1 上 G3 层动态绑定在设备 DHD0 或 DHD1 上
Hi3516CV500/Hi3516DV300	支持 1 个图形层，G0 G0 层固定绑定在设备 DHD0 上

## 4.2 API 参考

### HI\_MPI\_VO\_BindGraphicLayer

#### 【目的】

设置图形层绑定关系。

#### 【语法】

```
HI_S32 HI_MPI_VO_BindGraphicLayer(GRAPHIC_LAYER GraphicLayer, VO_DEV  
VoDev)
```

#### 【描述】

此接口实现绑定图形层到指定的 VO 设备。

#### 【参数】

参数名称	描述	输入/输出
GraphicLayer	图形层标识	输入
VoDev	VO 设备号	输入

#### 【差异说明】



芯片	是否支持
Hi3559V100/Hi3559AV100ES/Hi3516C V500/Hi3516DV300	不支持该接口
Hi3559AV100/Hi3519AV100	支持

【返回值】

返回值	描述
0	成功
-1	失败

【需求】

头文件：mpi\_vo.h、hi\_comm\_vo.h

库文件：libmpi.a

【注意】

调用前需保证图形层未使能，而且应先解除之前的绑定关系。

【举例】

无。

【相关接口】

[HI\\_MPI\\_VO\\_UnBindGraphicLayer](#)

## HI\_MPI\_VO\_UnBindGraphicLayer

【目的】

将指定图形层与设备的绑定关系解除。

【语法】

```
HI_S32 HI_MPI_VO_UnBindGraphicLayer(GRAPHIC_LAYER GraphicLayer, VO_DEV VoDev)
```

【描述】

将指定图形层与设备的绑定关系解除。

【参数】

参数名称	描述	输入/输出
GraphicLayer	图形层号。	输入



参数名称	描述	输入/输出
VoDev	视频设备号。	输入

【差异说明】

芯片	是否支持
Hi3559V100/Hi3559AV100ES/Hi3516CV500/Hi3516DV300	不支持该接口
Hi3559AV100/Hi3519AV100	支持

【返回值】

返回值	描述
0	成功
-1	失败

【需求】

- 头文件：mpi\_vo.h、hi\_comm\_vo.h
- 库文件：libmpi.a

【注意】

- 调用前需保证图形层处于关闭状态。
- VoDev 目前没有意义，一般将其赋值为 0 即可。
- 对未绑定的图形层进行解绑定亦会返回成功，即允许对同一个图形层解绑定多次。

【举例】

无。

【相关接口】

[HI\\_MPI\\_VO\\_UnBindGraphicLayer](#)

## HI\_MPI\_VO\_SetPubAttr

【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】

```
HI_S32 HI_MPI_VO_SetPubAttr(VO_DEV VoDev, const VO_PUB_ATTR_S *pstPubAttr)
```



具体请参见《HiMPP IPC Vx.0 媒体处理软件开发参考》或《HiMPP V4.0 媒体处理软件开发参考》。

## HI\_MPI\_VO\_GetPubAttr

### 【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】

```
HI_S32 HI_MPI_VO_GetPubAttr(VO_DEV VoDev, VO_PUB_ATTR_S *pstPubAttr)
```

具体请参见《HiMPP IPC Vx.0 媒体处理软件开发参考》或《HiMPP V4.0 媒体处理软件开发参考》。

## HI\_MPI\_VO\_Enable

### 【目的】

使能视频输出设备。

### 【语法】

```
HI_S32 HI_MPI_VO_Enable (VO_DEV VoDev)
```

### 【注意】

图形层要在显示设备上正常显示，必须在 open("/dev/fbn")之前先调用此接口使能视频输出设备。

具体请参见《HiMPP IPC Vx.0 媒体处理软件开发参考》或《HiMPP V4.0 媒体处理软件开发参考》。

## HI\_MPI\_VO\_Disable

### 【目的】

关闭视频输出设备。

### 【语法】

```
HI_S32 HI_MPI_VO_Disable (VO_DEV VoDev)
```

具体请参见《HiMPP IPC Vx.0 媒体处理软件开发参考》或《HiMPP V4.0 媒体处理软件开发参考》。

## 4.3 数据结构

具体请参见《HiMPP IPC Vx.0 媒体处理软件开发参考》或《HiMPP V4.0 媒体处理软件开发参考》。



## VO\_DEV

具体请参见《HiMPP IPC Vx.0 媒体处理软件开发参考》或《HiMPP V4.0 媒体处理软件开发参考》。

## VO\_PUB\_ATTR\_S

具体请参见《HiMPP IPC Vx.0 媒体处理软件开发参考》或《HiMPP V4.0 媒体处理软件开发参考》。



# 5 Proc 调试信息

## 5.1 图形层和 fb 设备号对应关系

- 对于 Hi3559V100/Hi3516CV500/Hi3516DV300 芯片，HiFB 最多可以管理 1 个叠加图形层：图形层 0，对应的设备文件是/dev/fb0。
- 对于 Hi3559AV100ES 芯片，HiFB 最多可以管理 2 个叠加图形层：图形层 0（G0）和图形层 1（G1），对应的设备文件是/dev/fb0 和/dev/fb1。
- 对于 Hi3559AV100 芯片，HiFB 最多可以管理 3 个叠加图形层：图形层 0（G0）和图形层 1（G1），图形层 2（G3）对应的设备文件是/dev/fb0、/dev/fb1、/dev/fb2。
- 对于 Hi3519AV100 芯片，HiFB 最多可以管理 3 个叠加图形层：图形层 0（G0）和图形层 1（G1），图形层 2（G3）对应的设备文件是/dev/fb0、/dev/fb1、/dev/fb2。
- 可通过命令 `cat /proc/umap/hifbn`（n 为图形层号）查看各个图形层的状态。

## 5.2 单个图形层调试信息

### 【调试信息】

```
# cat /proc/umap/hifb0
layer name           :layer_0
Open count           :0
Show state           :OFF
Start position       :(0, 0)
xres, yres            :(1280, 720)
xres_virtual, yres_virtual :(1280, 1440)
xoffset, yoffset      :(0, 720)
fix.line_length       :2560
Mem size:             :8100 KB
Layer Scale (hw):     :NO
ColorFormat:          :ARGB1555
Alpha Enable         :ON
AlphaChannel Enable   :OFF
Alpha0, Alpha1        :0, 255
```





```
Alpha Global           :0
Colorkey Enable        :OFF
Colorkey value         :0xfc00
Mirror Mode:           :NONE
Dynamic Range:         :SDR8
Deflicker Mode:        :NONE
Rotation mode:         :0
Deflicker Level:       :AUTO
HiFB mode:             :STANDARD
Display Buffer mode (+UsrBuf):unkown
Displaying addr (register) :0x841d2000
display buffer[0] addr  :0x84010000
display buffer[1] addr  :0x841d2000
Be PreMul Mode:        :NO
displayrect            : (1280, 720)
screenrect             : (1280, 720)
device max resolution  :1280, 720
IsNeedFlip(2buf)       :NO
BufferIndexDisplaying(2buf) :1
refresh request num(2buf) :0
switch buf num(2buf)   :0
union rect (2buf)      : (0,0,0,0)
canavas updated addr   :0x841d2000
canavas updated (w, h) :1280,720
canvas width           :1280
canvas height          :720
canvas pitch           :2560
canvas format          :ARGB1555
IsCompress             :NO
Is DDR Dettect         :NO
DDR Detect Zones       :0
```

### 【调试信息分析】

记录当前设备对应的图形层的内存配置信息及显示信息。

### 【参数说明】

参数		描述
图形层基本属性	layer name	G0 ~ G3 对应的名称依次为: layer_0、layer_1、layer_2、layer_3
	Open Count	该图形层打开次数。 在用户调用 open() 时增 1; 调用 close() 时减 1。在第一个用户 open() 时实际打开 VOU 硬件图形层; 在最后一个用户 close() 时才实际关闭 VOU 硬件图



参数		描述
		<p>形层。</p> <p>注：对于 Hi3559AV100ES、Hi3559AV100、Hi3519AV100，上述描述中使用“VO 硬件”代替“VOU 硬件”。</p>
	Show State	<p>该图像层显示状态。</p> <p>取值：{OFF 表示不显示；ON 表示显示}。</p> <p>在用户成功设置可变屏幕信息后，内部自动显示此图层，该状态为 1；用户显式地调用 FBIOPUT_SHOW_HIFB 隐藏/显示该图形层时此状态相应变化。</p>
	Start Position	<p>图形层在显示设备上的起始显示位置，如（100，50）表示起始显示位置 x 为 100，y 为 50。</p> <p>单位：像素。</p> <p>默认为（0,0），用户可调用 FBIOPUT_SCREEN_ORIGIN_HIFB 更新显示位置。</p>
	Layer Scale (hw)	<p>图形层是否支持硬件缩放功能。</p> <p>取值：{NO 表示不支持；YES 表示支持}。</p>
	ColorFormat	<p>图形层格式。</p> <p>取值：</p> <p>Hi3559V100/Hi3559AV100ES/Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516DV300： {ARGB1555、ARGB4444、ARGB8888}</p> <p>系统加载后默认值 ARGB1555。</p> <p>用户设置可变屏幕信息中的格式项后更新。</p>
	Alpha Enable	<p>图形层 alpha 是否使能。</p> <p>取值：{OFF 表示否，ON 表示是}，默认值为 ON。</p> <p>Proc 中所有 alpha 相关信息在用户设置 FBIOPUT_ALPHA_HIFB 时更新。</p> <p>该项关闭，则像素 alpha 配置不生效；</p> <p>该项开启且 AlphaChannel 关闭，则仅像素 alpha 生效（对 ARGB1555 格式，即 Alpha0 和 Alpha1 生效），该项开启且 AlphaChannel 开启，则像素 alpha 和全局 alpha(Alpha Global)都生效。</p>
	AlphaChannel Enable	<p>全局 alpha 是否生效控制开关。</p> <p>取值：{OFF 表示否，ON 表示是}，默认值为 OFF。</p> <p>AlphaChannel Enable 使能，Alpha Global 便生效。</p>



参数		描述
	Alpha0	ARGB1555 格式时，当最高位为 0 时，选择该值作为 Alpha 叠加的 Alpha 值。 取值：0~255，默认为 0。
	Alpha1	ARGB1555 格式时，当最高位为 1 时，选择该值作为 Alpha 叠加的 Alpha 值。 取值：0~255，默认为 255。
	Alpha Global	全局 alpha。 取值：0~255，默认为 255。
	Colorkey Enable	该图层 colorkey 功能是否使能。 取值：{OFF 表示否，ON 表示是}，默认值为 OFF。
	Colorkey Value	被透明掉的像素值，与当前层设置的像素格式一致。
	Mirror Mode	镜像模式，无镜像(NONE)、水平(HORIZONTAL)镜像、垂直(VERTICAL)镜像、水平和垂直镜像(BOTH)。
	Dynamic Range	动态范围，SDR8，SDR10，HDR10，HLG，SLF，Hi3559AV100ES、Hi3559AV100 中，G0 仅支持 SDR8，SDR10，HDR10；G1 仅支持 SDR8。 Hi3519AV100/Hi3516CV500/Hi3516DV300 中，所有的层仅支持 SDR8。
	Deflicker Mode	抗闪烁模式。
	Rotation mode	旋转角度，0，90，180，270 分别表示旋转 0 度，90 度，180 度，270 度。
	Deflicker Level	抗闪烁级别。
	device max resolution	该层所在的显示设备的当前显示分辨率。
	IsCompress	是否启用压缩功能标识。
	DDR Detect Zones	DDR 侦测区域个数。
图形层的显存 buff 信息	fix.smem_start	为该图形层分配的显存的物理起始地址。 Hifb 模块插入后即已分配。
	fix.smem_len	为该图形层分配的实际显存大小。 单位：byte；范围：(最小 256，最大受限于 mmz 内存大小)。 Hifb 显存从 mmz 中分配，mmz 按 4096byte 为 1



参数		描述
		块，显存大小必须为整数块。故用户插入模块时指定 <code>vramX_size=256</code> ,实际分配 4096byte，即 <code>fix.smem_len</code> 显示 4096。
	<code>fix.line_length</code>	显存 stride。 单位：byte； 显存 stride 是根据用户设置的可变屏幕信息的 <code>var.xres_virtual</code> x 每像素所占 byte 数，且自动向上调整为 8byte 对齐。 用户应通过获取固定屏幕信息 <code>fix</code> 来得到显存 stride。
	<code>var.xres_virtual</code>	虚拟屏幕宽度，如图 2-1 所示。 单位：像素；默认值：720 ( <code>xres_virtual</code> , <code>yres_virtual</code> )：称为虚拟屏幕区域，表示用户最大可通过 <code>hifb</code> 操作的区域，而实际显示的区域由 ( <code>xres</code> , <code>yres</code> ) 指定。虚拟屏幕区域的面积当然不能超过显存大小。 ( <code>xres</code> , <code>yres</code> )：表示本次显示的区域的大小，它可以是 ( <code>xres_virtual</code> , <code>yres_virtual</code> ) 中的一部份。 ( <code>xoffset</code> , <code>yoffset</code> )：表示本次显示的区域位于 ( <code>xres_virtual</code> , <code>yres_virtual</code> ) 区域中的起始位置。
	<code>var.yres_virtual</code>	虚拟屏幕高度，如图 2-1 所示。 单位：像素；默认值：576。
	<code>var.xoffset</code>	实际显示区域在虚拟屏幕区域中的 x 起始坐标 单位：像素；默认值：0。 用户可调用 <code>FBIOPAN_DISPLAY</code> 调整要显示区域在显存中的位置。
	<code>var.yoffset</code>	实际显示区域在虚拟屏幕区域中的 y 起始坐标 单位：像素；默认值：0。 用户可调用 <code>FBIOPAN_DISPLAY</code> 调整要显示区域在显存中的位置。
	<code>var.xres</code>	实际显示区域的宽度，如图 2-1 所示。 单位：像素；默认值：720。
	<code>var.yres</code>	实际显示区域的高度，如图 2-1 所示。 单位：像素；默认值：576。
	<code>HiFB mode</code>	HiFB 工作模式。 标准模式：STANDARD； 扩展模式：EXTEND。



参数		描述
	Display Buffer mode(+UsrBuf)	刷新模式。各模式与对应显示的内容关系如下： HIFB_LAYER_BUF_DOUBLE —— triple, HIFB_LAYER_BUF_ONE —— double, HIFB_LAYER_BUF_NONE —— single, DOUBLE_IMMEDIATE —— triple( no frame discarded), HIFB_LAYER_BUF_BUTT —— unkown 注：+UsrBuf 表示该项计数包含用户 buffer。
	Be PreMul Mode	是否预乘模式。 YES：是； NO：不是。 默认值：NO。
	canavas updated addr	画布更新区域的起始地址。

## 5.3 图形层的绑定关系

- 固定绑定关系
  - 对于 Hi3559AV100ES 芯片，图形层与设备的绑定关系是固定不变的，G0 固定绑定设备 DHD0，G1 固定绑定设备 DHD1。
  - 对于 Hi3559AV100 芯片，部分图形层与设备的绑定关系是固定不变的，G0 固定绑定设备 DHD0，G1 固定绑定设备 DHD1。
  - 对于 Hi3519AV100 芯片，部分图形层与设备的绑定关系是固定不变的，G0 固定绑定设备 DHD0，G1 固定绑定设备 DHD1。
  - 对于 Hi3516CV500 和 Hi3516DV300 芯片，图形层与设备的绑定关系是固定不变的，G0 固定绑定设备 DHD0。
- 动态绑定关系
  - 对于 Hi3559AV100 芯片，部分图形层与设备的绑定关系是可动态调整的，G3 动态绑定设备 DHD0 或 DHD1，G3 默认绑定在 DHD1 上。
  - 对于 Hi3519AV100 芯片，部分图形层与设备的绑定关系是可动态调整的，G3 动态绑定设备 DHD0 或 DHD1，G3 默认绑定在 DHD0 上。
  - 对于可动态绑定的图形层的绑定关系，可查看命令 `cat /proc/umap/vo` 输出的最后几行，内容如下：

```
-----GRAPHICS LAYER-----  
Layer  BindDev  
HC0      0
```

### 【参数说明】



参数		描述
GRAPHICS LAYER	Layer	可动态绑定的图形层
	BindDev	图形层所绑定的设备号