



# CV184X GFBG 开发指南

Version: 1.0.0.0

Release date: 2025-4-22

©2022 北京晶视智能科技有限公司  
本文件所含信息归北京晶视智能科技有限公司所有。  
未经授权，严禁全部或部分复制或披露该等信息。

# 目录

<b>1</b>	<b>声明</b>	<b>2</b>
<b>2</b>	<b>GFBG 概述</b>	<b>3</b>
<b>3</b>	<b>GFBG 功能描述</b>	<b>4</b>
3.1	体系结构 . . . . .	4
3.2	标准功能与扩展功能 . . . . .	4
3.2.1	标准功能 . . . . .	4
3.2.2	扩展功能 . . . . .	5
3.3	扩展模式下的图形层刷新类型 . . . . .	5
3.3.1	0 buffer (即 CVI_FB_LAYER_BUF_NONE) . . . . .	5
3.3.2	1 buffer (即 CVI_FB_LAYER_BUF_ONE) 暂不支持 . . . . .	6
3.3.3	2 buffer (即 CVI_FB_LAYER_BUF_DOUBLE) 暂不支持 . . . . .	7
3.3.3.1	图形层压缩暂不支持 . . . . .	7
<b>4</b>	<b>模块加载</b>	<b>9</b>
4.1	原理介绍 . . . . .	9
4.2	参数设置 . . . . .	9
4.2.1	参数 video . . . . .	9
4.2.2	物理显存大小与虚拟内存关系 . . . . .	10
4.2.3	参数默认值 . . . . .	10
<b>5</b>	<b>应用开发流程</b>	<b>11</b>
<b>6</b>	<b>GFBG 接口说明</b>	<b>14</b>
6.1	ioctl 函数 . . . . .	14
6.1.1	F BIOGET_VSCREENINFO . . . . .	15
6.1.2	F BIOPUT_VSCREENINFO . . . . .	16
6.1.3	F BIOGET_FSCREENINFO . . . . .	17
6.1.4	F BIOPAN_DISPLAY . . . . .	17
6.1.5	F BIOGET_SCREEN_ORIGIN_GFBG . . . . .	19
6.1.6	F BIOPUT_SCREEN_ORIGIN_GFBG . . . . .	20
6.1.7	F BIOGET_SHOW_GFBG . . . . .	21
6.1.8	F BIOPUT_SHOW_GFBG . . . . .	22
6.1.9	F BIOGET_COLORKEY_GFBG . . . . .	23
6.1.10	F BIOPUT_COLORKEY_GFBG . . . . .	23
6.1.11	F BIOGET_VER_BLANK_GFBG . . . . .	24
6.1.12	F BIOPUT_LAYER_INFO . . . . .	25
6.1.13	F BIOGET_LAYER_INFO . . . . .	26
6.1.14	F BIOGET_CANVAS_BUF . . . . .	27
6.1.15	F BIO_REFRESH . . . . .	27

6.1.16	FBIOPUT_SCREEN_SIZE . . . . .	28
6.1.17	F BIOGET_SCREEN_SIZE . . . . .	29

修订记录

Revision	Date	Description
0.1	2025/04/21	初稿

# 1 声明

---



## 法律声明

本数据手册包含北京晶视智能科技有限公司（下称“晶视智能”）的保密信息。未经授权，禁止使用或披露本数据手册中包含的信息。如您未经授权披露全部或部分保密信息，导致晶视智能遭受任何损失或损害，您应对因之产生的损失/损害承担责任。

本文件内信息如有更改，恕不另行通知。晶视智能不对使用或依赖本文件所含信息承担任何责任。本数据手册和本文件所含的所有信息均按“原样”提供，无任何明示、暗示、法定或其他形式的保证。晶视智能特别声明未做任何适销性、非侵权性和特定用途适用性的默示保证，亦对本数据手册所使用、包含或提供的任何第三方的软件不提供任何保证；用户同意仅向该第三方寻求与此相关的任何保证索赔。此外，晶视智能亦不对任何其根据用户规格或符合特定标准或公开讨论而制作的可交付成果承担责任。

## 联系我们

**地址** 北京市海淀区丰豪东路 9 号院中关村集成电路设计园（ICPARK）1 号楼

深圳市宝安区福海街道展城社区会展湾云岸广场 T10 栋

**电话** +86-10-57590723 +86-10-57590724

**邮编** 100094（北京）518100（深圳）

**官方网站** <https://www.sophgo.com/>

**技术论坛** <https://developer.sophgo.com/forum/index.html>

## 2 GFBG 概述

---

Framebuffer（简称 GFBG）是数字媒体处理平台提供的模块，用于管理叠加图形层。它不仅具备 Linux Framebuffer 的基本功能，如显示图形和处理像素数据，还提供了多种扩展功能。这些扩展功能包括支持多层叠加、动态调整图层属性、优化的内存管理以及高效的硬件加速能力。通过这些功能，GFBG 模块能够满足复杂图形显示需求，并为开发者提供更大的灵活性和便利性。

# 3 GFBG 功能描述

## 3.1 体系结构

应用程序基于 Linux 系统使用 GFBG，GFBG 体系结构如下图所示。

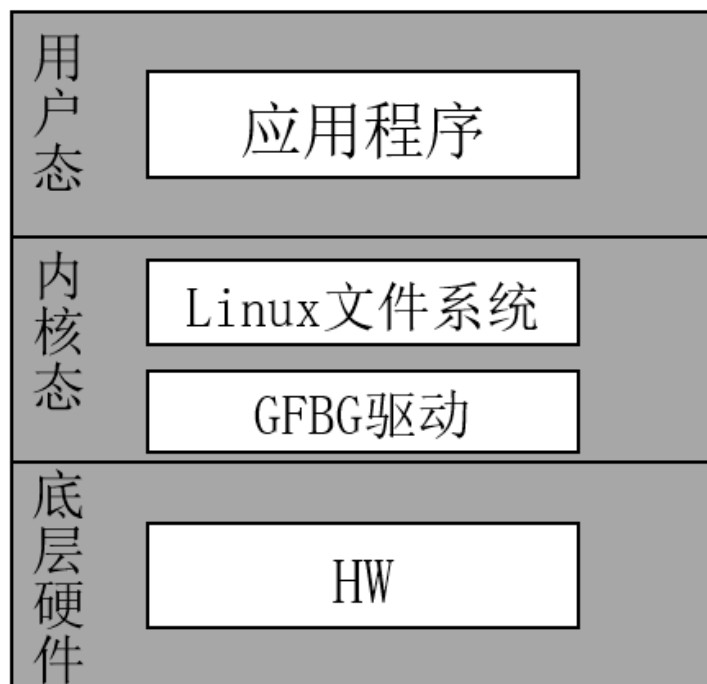


图 3.1: GFBG 体系结构

## 3.2 标准功能与扩展功能

### 3.2.1 标准功能

GFBG 支持以下的 Linux Framebuffer 标准功能：

- 物理显存映射到或者解除映射到虚拟内存空间中，允许用户通过虚拟地址访问物理显存，从而实现更高效的内存管理和操作。

- 像操作普通文件一样操作物理显存，支持通过文件接口对显存进行读写操作，简化了显存的使用方式。
- 设置硬件显示分辨率和象素格式，每个叠加图形层的支持的最大分辨率和象素格式可以通过支持能力接口获取，确保硬件显示的灵活性和兼容性。
- 从物理显存的任何位置进行读写及显示，支持对显存内容的任意位置进行操作，满足复杂显示需求的同时提供更高的自由度。

### 3.2.2 扩展功能

GFBG 新增以下扩展功能：

- 设置和获取叠加图形层的 colorkey 值。
- 设置当前叠加图形层的起始位置（相对于屏幕原点的偏移）。
- 设置和获取当前叠加图形层的显示状态（显示/隐藏）。
- 通过模块加载参数配置 GFBG 的物理显存大小和管理叠加图形层的数目。
- 设置和获取压缩模式的状态，目前仅支持非压缩模式。
- 设置/获取图形层刷新类型（0 buffer、1 buffer 与 2 buffer），目前仅支持 0 buffer 模式。

## 3.3 扩展模式下的图形层刷新类型

GFBG 为上层用户提供了一套完整的刷新方案，称为 FB 的扩展模式。在对系统性能、内存、图形显示效果各方面综合衡量的基础上，可根据需要选择合适的刷新方案。该模式支持灵活的刷新策略，用户可以根据实际需求选择不同的刷新类型，例如 0 buffer、1 buffer 和 2 buffer 模式，以满足不同场景下的性能和显示效果要求。同时，扩展模式还提供了对刷新参数的精细化控制，确保在复杂的显示需求下能够实现高效、稳定的图形渲染。

### 3.3.1 0 buffer（即 CVI\_FB\_LAYER\_BUF\_NONE）

上层用户的绘制 buffer 即为显示 buffer。该刷新类型可以显著节省内存消耗，同时具备最快的刷新速度。然而，由于绘制 buffer 与显示 buffer 是同一个，用户在绘制过程中可能会直接看到图形的绘制过程，例如未完成的图形或中间状态的内容。这种模式适用于对内存和性能要求较高，但对绘制过程的视觉效果要求不高的场景。示意如下图所示。



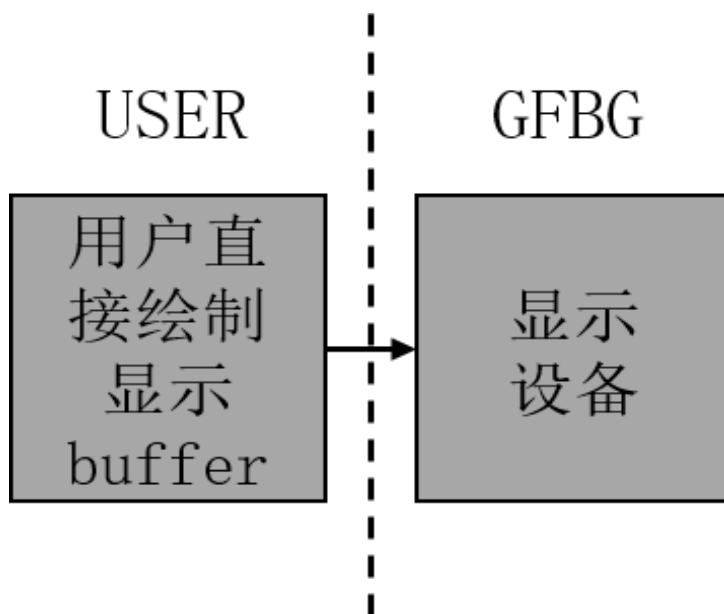


图 3.2: 0 buffer 示意图

### 3.3.2 1 buffer (即 CVI\_FB\_LAYER\_BUF\_ONE) 暂不支持

显示 buffer 由 GFBG 提供，因此需要一定的内存。该刷新类型是对显示效果和内存需求的一种折中考虑。它通过在绘制 buffer 和显示 buffer 之间引入一个缓冲区，避免了绘制过程中直接影响显示内容的问题，从而提升了显示效果的稳定性和一致性。然而，由于需要额外的内存来存储显示 buffer，因此会增加内存的使用量。此外，在某些情况下可能会出现锯齿现象，这需要根据具体的应用场景进行权衡和优化。示意如下图所示。

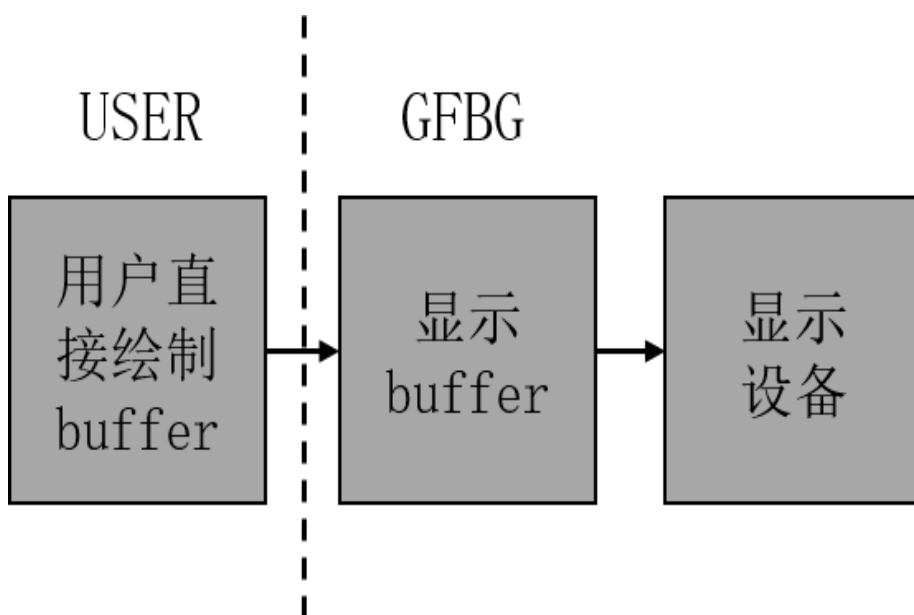


图 3.3: 1 buffer 示意图

### 3.3.3 2 buffer (即 CVI\_FB\_LAYER\_BUF\_DOUBLE) 暂不支持

显示 buffer 由 GFBG 提供。与前面的刷新类型相比，该模式需要占用最多的内存资源，但能够提供最佳的图形显示效果。通过引入双缓冲机制，确保了显示内容的稳定性和一致性，同时避免了绘制过程中对显示内容的直接影响。这种模式适用于对显示效果要求较高的场景，例如需要高质量图形渲染或复杂动画的应用。示意如下图所示。

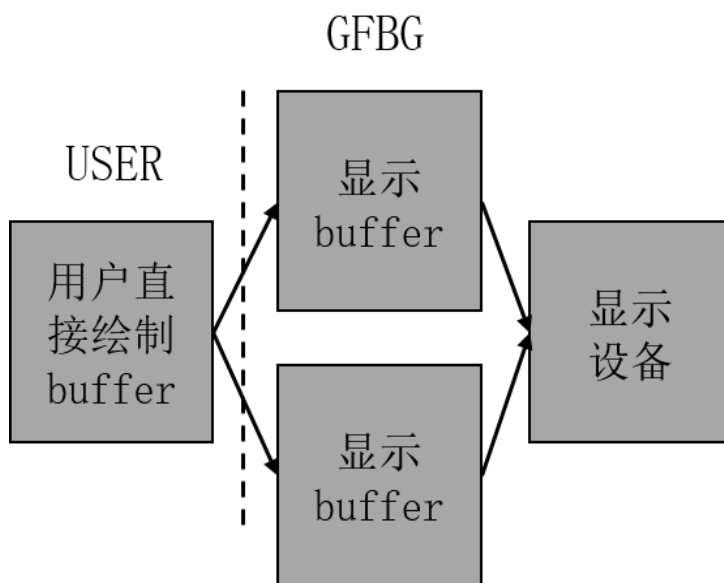


图 3.4: 2 buffer 示意图

#### 3.3.3.1 图形层压缩暂不支持

图形层压缩功能是指图形层能够接收硬件压缩的数据，并基于这些压缩数据进行解压显示。当显示 buffer 的数据未发生变化时，图形层每次都会加载压缩后的数据进行解压显示。启用压缩功能的图形层可以有效降低总线的传输带宽，但需要额外分配一帧压缩数据所需的内存空间。通过这种方式，图形层压缩功能在保证显示效果的同时，能够显著优化系统性能，尤其是在高分辨率和高刷新率的显示场景下，减少了数据传输的压力。此外，该功能还可以在在一定程度上降低系统功耗，使其更适合嵌入式设备和移动设备等对功耗敏感的应用场景。然而，需要注意的是，启用压缩功能可能会增加系统的复杂性，并对硬件和软件的兼容性提出更高的要求，因此在实际应用中需要根据具体需求进行权衡和选择。典型的图形层压缩 buffer 示意图如下图所示。

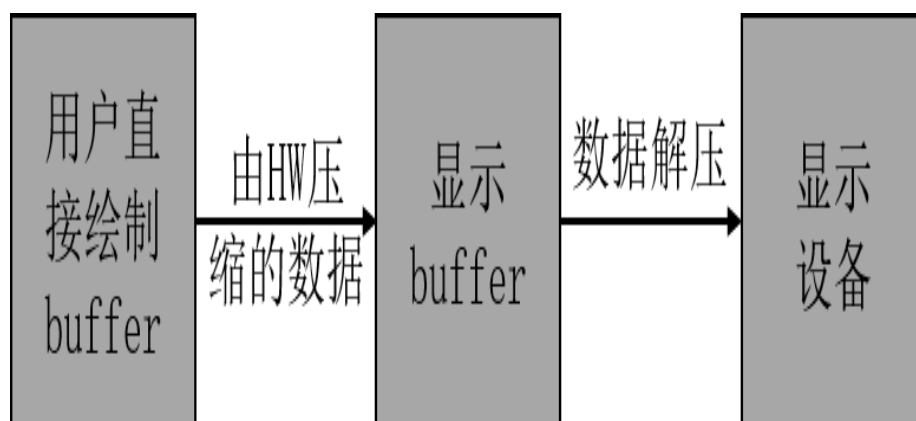


图 3.5: 图形层压缩 buffer 示意图

# 4 模块加载

## 4.1 原理介绍

某些 Linux Framebuffer 驱动（如 versa）不支持在运行期间更改分辨率、颜色深度、时序等显示属性。对此，Linux 系统提供一种机制，允许在内核启动或模块加载时，通过参数将相应选项传递给 Linux Framebuffer。通过这种机制，用户可以在系统启动之前，预先配置显示相关的参数，以满足特定的应用需求。可以在内核加载器中配置内核启动参数，例如指定分辨率、刷新率或颜色深度等。需要注意的是，GFBG 驱动在加载时只能设置物理显存的大小，不允许设置其它选项。因此，在使用 GFBG 驱动时，用户需要根据实际需求提前规划显存大小，并在模块加载时正确传递参数。这意味着用户需要在系统设计阶段充分考虑应用场景的需求，例如是否需要支持高分辨率显示或多层叠加图形的渲染。通过合理规划显存大小，可以确保系统在运行时能够满足性能和功能的要求，同时避免因显存不足导致的显示异常或性能下降。

## 4.2 参数设置

GFBG 可配置其管理的叠加图形层物理显存的大小。物理显存大小决定了 GFBG 可使用的最大物理显存和系统的可设置虚拟分辨率。在加载 GFBG 驱动时通过参数传递设置物理显存大小，物理显存的大小一经设置就不会改变。这意味着用户在配置显存大小时需要充分考虑应用的实际需求，例如是否需要支持高分辨率的显示内容或复杂的图形叠加效果。显存大小的设置直接影响到系统的显示能力和性能，因此建议用户在配置时根据具体的应用场景进行测试和验证。此外，合理的显存规划不仅可以提高系统的稳定性，还可以避免资源浪费，从而优化系统的整体性能。

### 4.2.1 参数 video

video=" gfbg:vram0\_size:xxx, vram1\_size:xxx, ..."

- gfbg: 表示加载 GFBG 驱动。
- vram0\_size: 表示 GFBG 管理的图形层 0 物理显存大小，CV184X 仅有图形层 0，仅支持 vram0\_size。
- vram1\_size: 表示 GFBG 管理的图形层 1 物理显存大小。
- xxx: 表示物理显存的大小，单位为 KB。
- 如果图层不配置物理显存大小，目前驱动默认分配为 8100KB。

## 4.2.2 物理显存大小与虚拟内存关系

对于 GFBG 的标准模式，`vramn_size` 和虚拟分辨率的关系如下： $\text{vramn\_size} * 1024 \geq \text{xres\_virtual} * \text{yres\_virtual} * \text{bpp}$ ；其中  $\text{xres\_virtual} * \text{yres\_virtual}$  是虚拟分辨率，`bpp` 是每个像素所占字节数。

## 4.2.3 参数默认值

CV184X GFBG 驱动器的默认参数为：`video=" gfbg:vram0_size:8100"`，即 1080p 分辨率，32 位色深需要的最小显存。用户需要从全局的角度出发配置 GFBG 需要管理的叠加图形层，并且为每个叠加图形层分配适当的显存。

# 5 应用开发流程

---

GFBG 主要用于显示 2D 图形（以直接操作物理显存的方式），其核心功能是通过高效的显存管理和图形叠加技术，提供流畅的图形显示效果。它能够满足嵌入式系统中对高性能、低延迟图形显示的需求，同时支持灵活的分辨率和色深配置。

GFBG 应用开发流程如下图所示，开发者可以根据实际需求，按照流程图中的步骤逐步完成应用的设计、开发和调试工作。通过合理的显存分配和参数配置，可以充分发挥 GFBG 的性能优势，为用户提供优质的图形显示体验。

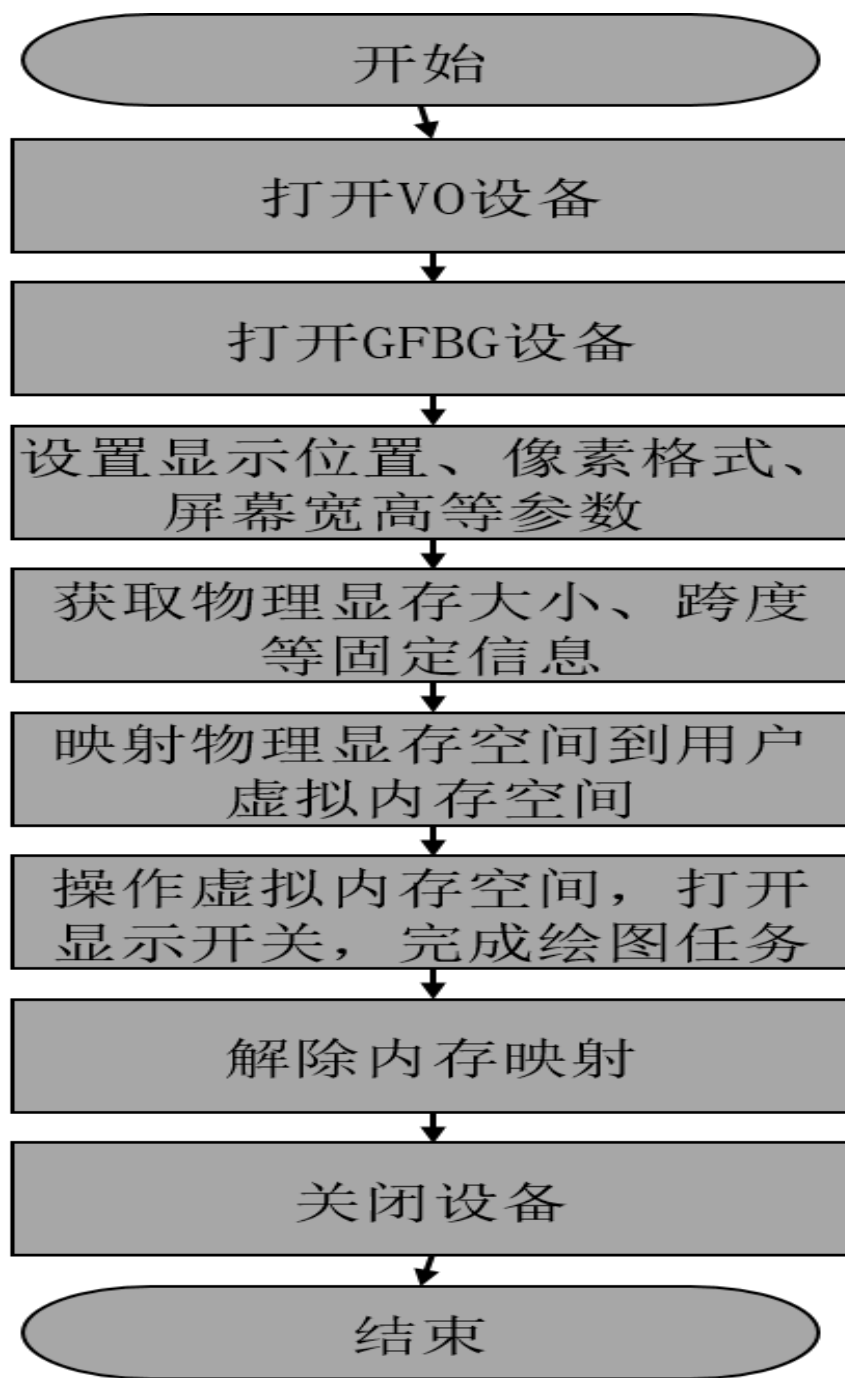


图 5.1: GFBG 应用开发流程图

具体开发步骤如下：

- 基于 VO 的对外 API 打开 VO 设备。
- 调用 open 函数打开 GFBG 设备。
- 调用 ioctl 函数设置 GFBG 显示位置，设置 GFBG 像素格式以及屏幕宽高等参数，具体设置请参考下一节的 GFBG 的 ioctl 接口说明。
- 调用 ioctl 函数获取 GFBG 分配的物理显存大小、跨度等信息，用于后续的显存读写操作。

- 调用 `mmap` 函数将 GFBG 分配的物理显存映射到用户空间，获取虚拟地址。
- 用户操作虚拟内存，完成具体的绘制任务，打开显示开关，标准模式下虚拟显示分辨率可以设置为屏幕显示分辨率的 2 倍，用于实现更高的显示精度。
- 调用 `munmap` 函数解除映射
- 调用 `close` 函数关闭 GFBG 设备。



# 6 GFBG 接口说明

GFBG 的 API 主要分为文件操作类（open、close 等）、显存映射类（mmap、munmap 等）、显存控制和状态查询类（像素格式、颜色深度等）和层间效果控制和状态查询类（显示原点、显示状态等）四大类。每一类的函数都具有特定的功能和用途，开发者可以根据实际需求选择合适的函数进行调用。其中第一类和第二类是 Linux 提供的常用标准接口，本小节不会赘述，主要介绍第三类和第四类的接口函数。

## 6.1 ioctl 函数

GFBG 的用户态接口以 ioctl 形式体现，其形式如下：

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

GFBG 只需要 3 个参数，即形式为：

```
int ioctl(int fd, unsigned long cmd, CMD_DATA_TYPE *cmd_data);
```

该功能模块为用户提供以下 API：

- FBIOGET\_VSCREENINFO：获取屏幕的可变信息。
- FBIOPUT\_VSCREENINFO：设置 Framebuffer 的屏幕分辨率和像素格式等。
- FBIOGET\_FSCREENINFO：获取 Framebuffer 的固定信息。
- FBIOPAN\_DISPLAY：设置从虚拟分辨率中的不同偏移处开始显示。
- FBIOGET\_SCREEN\_ORIGIN\_GFBG：获取叠加层在屏幕上显示的起始点坐标。
- FBIOPUT\_SCREEN\_ORIGIN\_GFBG：设置叠加层在屏幕上显示的起始点坐标。
- FBIOGET\_SHOW\_GFBG：获取当前叠加层的显示状态。
- FBIOPUT\_SHOW\_GFBG：显示或隐藏该叠加层。
- FBIOGET\_COLORKEY\_GFBG：获取叠加层的 colorkey。
- FBIOPUT\_COLORKEY\_GFBG：设置叠加层的 colorkey。
- FBIOGET\_VER\_BLANK\_GFBG：等待叠加层的垂直消隐区到来。
- FBIOPUT\_LAYER\_INFO：设置图层信息。
- FBIOGET\_LAYER\_INFO：获取图层信息。

- `F BIOGET_CANVAS_BUF` : 获取画布信息。
- `F BIO_REFRESH` : 扩展模式下, 启动刷新操作。
- `F BIOPUT_SCREEN_SIZE` : 设置图层在屏幕上的显示大小。
- `F BIOGET_SCREEN_SIZE` : 获取图层在屏幕上的显示大小。

### 6.1.1 FBIOGET\_VSCREENINFO

#### 【描述】

获取屏幕的可变信息。

#### 【语法】

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

#### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIO- GET_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: `cvi_comm_fb.h`

#### 【注意】

#### 【举例】

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

#### 【相关主题】

`FBIOPUT_VSCREENINFO`

## 6.1.2 FBIOPUT\_VSCREENINFO

### 【描述】

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);
```

### 【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIOPUT_VSCREENINFO	INFO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: `cvi_comm_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。
- 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。
- 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】

```
struct fb_var_screeninfo vinfo;
vinfo.xres = 1920;
vinfo.yres = 1080;
vinfo.bits_per_pixel = 32;
vinfo.xres_virtual = 1920;
vinfo.yres_virtual = 2160;

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

### 【相关主题】

[FBIOGET\\_VSCREENINFO](#)

### 6.1.3 FBIOGET\_FSCREENINFO

**【描述】**

获取 Framebuffer 的固定信息。

**【语法】**

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

**【参数】**

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIO- GET_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: cvl\_comm\_fb.h

**【注意】**

无。

**【举例】**

无。

**【相关主题】**

无。

### 6.1.4 FBIOPAN\_DISPLAY

**【描述】**

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);
```

**【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: `cvi_comm_fb.h`

**【注意】**

- 此接口只应在 FB 标准模式中使用。
- 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。
- 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。
- 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。如下图所示: (`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。

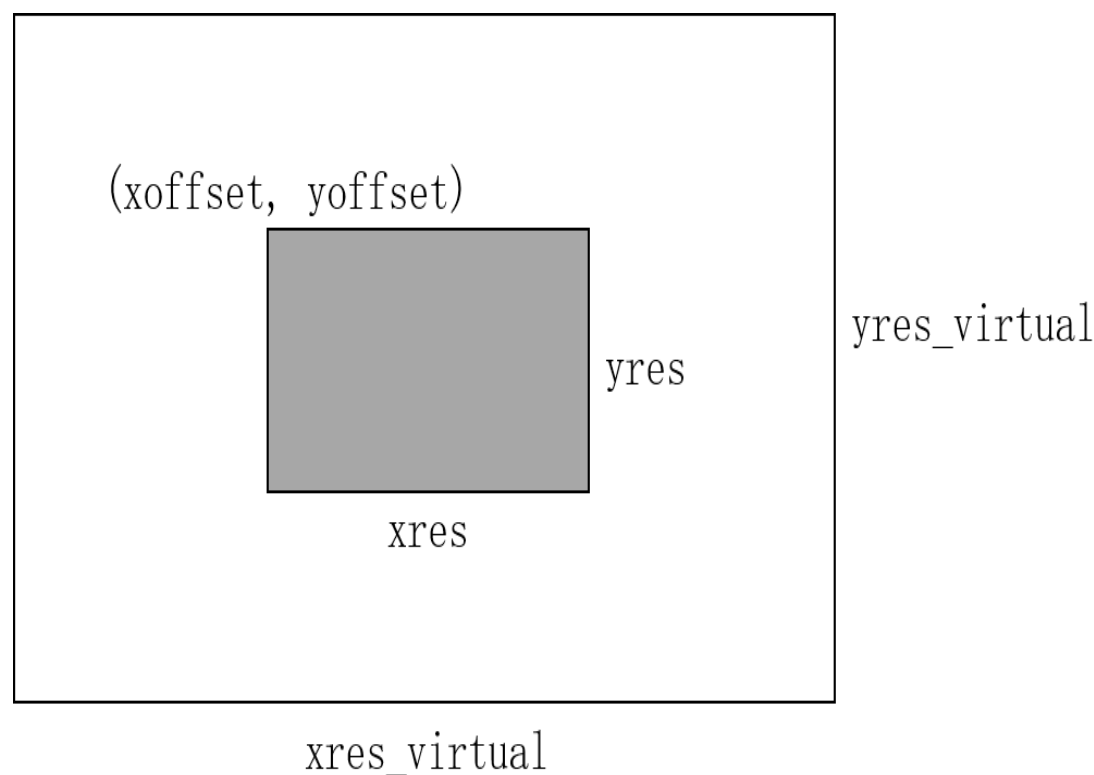


图 6.1: FBIOPAN\_DISPLAY 示意图

【举例】 无。

【相关主题】

无。

### 6.1.5 FBIOGET\_SCREEN\_ORIGIN\_GFBG

【描述】

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_GFBG, cvi_fb_point *point);
```

【参数】

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIO- GET_SCREEN_ORIGIN_GFBG	ioctl 号	输入
point	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: `cvi_comm_gfbg.h`

**【注意】****【举例】**

无。

**【相关主题】**

`FBIOPUT_SCREEN_ORIGIN_GFBG`

## 6.1.6 FBIOPUT\_SCREEN\_ORIGIN\_GFBG

**【描述】**

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_GFBG, cvi_fb_point *point);
```

**【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_GFBG		输入
point	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: `cvi_comm_gfbg.h`

**【注意】**

- 如果叠加层坐标原点超出了范围 ( $x\_pos > (max\_width - min\_width)$  或  $y\_pos > (max\_height - min\_height)$ ), 默认将坐标原点设置为  $(max\_width - min\_width, max\_height - min\_height)$ , 其中 `max_width` 和 `max_height` 的值是设备时序定义的最大宽高; `min_width` 和 `min_height` 分别表示可加载的最小图像的宽和高, 可通过 `FBIOPUT_SCREEN_ORIGIN_GFBG` 接口中的 `min_width` 和 `min_height` 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。

**【相关主题】**

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

## 6.1.7 FBIOGET\_SHOW\_GFBG

**【描述】**

获取当前叠加层的显示状态。

**【语法】**

```
int ioctl(int fd, FBIOGET_SHOW_GFBG, CVI_BOOL *show);
```

**【参数】**

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIO- GET_SHOW_GFBG	ioctl 号	输入
show	该叠加层的显示状态: - * show = CVI_TRUE: 显示当前叠加层 - * show = CVI_FALSE: 隐藏当前叠加层	输出

**【返回值】**

返回值	描述
0	成功。
-1	失败。

**【需求】**

- 头文件: `cvi_comm_gfbg.h`

**【注意】**

- 对软鼠标不适用。

**【举例】**

无。

**【相关主题】**

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



## 6.1.8 FBIOPUT\_SHOW\_GFBG

### 【描述】

显示或隐藏该叠加层。

### 【语法】

```
int ioctl(int fd, FBIOPUT_SHOW_GFBG, CVI_BOOL *show);
```

### 【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_GFBG	GFBG 号	输入
show	该叠加层的显示状态: - * show = CVI_TRUE: 显示当前叠加层 - * show = CVI_FALSE: 隐藏当前叠加层	输入

### 【返回值】

返回值	描述
0	成功。
-1	失败。

### 【需求】

- 头文件: cvl\_comm\_gfbg.h

### 【注意】

- 为正常显示, 在显示之前, 应将 show 的值设为 CVI\_TRUE 调用 ioctl(fd, FBIOPUT\_SHOW\_GFBG, &show)。
- 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。

### 【相关主题】

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

### 6.1.9 FBIOGET\_COLORKEY\_GFBG

**【描述】**

获取叠加层的 colorkey。

**【语法】**

```
int ioctl (int fd, FBIOGET_COLORKEY_GFBG, cvi_fb_colorkey *colorkey);
```

**【参数】**

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIO- GET_COLORKEY_GFBG	ioctl 号	输入
colorkey	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: cvi\_comm\_gfbg.h

**【注意】****【举例】**

无。

**【相关主题】**

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

### 6.1.10 FBIOPUT\_COLORKEY\_GFBG

**【描述】**

设置叠加层的 colorkey。

**【语法】**

```
int ioctl (int fd, FBIOPUT_COLORKEY_GFBG, cvi_fb_colorkey *colorkey);
```

**【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_GFBG	ioctl 号	输入
colorkey	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: `cvi_comm_gfbg.h`

**【注意】****【举例】**

```
cvi_fb_colorkey colorkey;
colorkey.enable = CVI_TRUE;
colorkey.value = 0xFFFF0000;
if (ioctl(fd, FBIOPUT_COLORKEY_GFBG, &colorkey) < 0) {
    return -1;
}
```

**【相关主题】**

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

## 6.1.11 FBIOGET\_VER\_BLANK\_GFBG

**【描述】**

为了操作显存时不引起撕裂现象，建议在该叠加层的消隐区对显存进行操作，通过该接口可以等待该叠加层消隐区的到来，底层原理是等待 vo 的下一个中断到来。

**【语法】**

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

**【参数】**

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

**【返回值】**

返回值	描述
0	成功。
-1	失败。

**【需求】**

- 头文件: `cvi_comm_gfbg.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。

**【相关主题】**

无。

## 6.1.12 FBIOPUT\_LAYER\_INFO

**【描述】**

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, cvi_fb_layer_info *layer_info);
```

**【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	层 ID 号	输入
layer_info	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。

**【需求】**

- 头文件: `cvi_comm_gfbg.h`

**【注意】**

- 在设置完某项属性后，必须通过设置 `layer_info` 的 `mask` 设置相应的掩码，否则该项设置不会生效。

- 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\_SCREEN\_SIZE。

**【举例】**

```

cvi_fb_layer_info layer_info = {0};
layer_info.buf_mode = CVI_FB_LAYER_BUF_NONE;
layer_info.mask = CVI_FB_LAYER_MASK_BUF_MODE;
ret = ioctl(s32Fd, FBIOPUT_LAYER_INFO, &layer_info);

```

**【相关主题】**

无。

### 6.1.13 FBIOGET\_LAYER\_INFO

**【描述】**

获取图层信息，包括刷新模式屏幕起始点位置、画布分辨率、显存分辨率、屏幕显示分辨率。

**【语法】**

```
int ioctl(int fd, FBIOGET_LAYER_INFO, cvi_fb_layer_info *layer_info);
```

**【参数】**

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIO- GET_LAYER_INFO	ioctl 号	输入
layer_info	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：cvi\_comm\_gfbg.h

**【注意】**

- 该接口获取到的接口数据 cvi\_fb\_layer\_info 中，mask 成员是没有意义的，始终被填充为 CVI\_FB\_LAYER\_MASK\_BUTT。

**【举例】**

无。

**【相关主题】**

无。

### 6.1.14 FBIOGET\_CANVAS\_BUF

**【描述】**

获取画布信息。

**【语法】**

```
int ioctl(int fd, FBIOGET_CANVAS_BUF, cvi_fb_buf *canvas_buf);
```

**【参数】**

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIO- GET_CANVAS_BUF	ioctl 号	输入
canvas_buf	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: cvi\_comm\_gfbg.h

**【注意】****【举例】**

无。

**【相关主题】**

无。

### 6.1.15 FBIO\_REFRESH

**【描述】**

扩展模式下，启动刷新操作。

**【语法】**

```
int ioctl(int fd, FBIO_REFRESH, cvi_fb_buf *buf_info);
```

**【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
buf_info	cvi_fb_buf 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: cvi\_comm\_gfbg.h

**【注意】** 无。**【举例】**

无。

**【相关主题】**

无。

## 6.1.16 FBIOPUT\_SCREEN\_SIZE

**【描述】**

设置图层在屏幕上的显示大小。

**【语法】**

```
int ioctl(int fd, FBIOPUT_SCREEN_SIZE, cvi_fb_size *cvi_fb_size);
```

**【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_SIZE	ioctl 号	输入
cvi_fb_size	cvi_fb_size 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: cvi\_comm\_gfbg.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。
- 图形层缩放限制：输入分辨率超过 1920x1080，不支持放大。
- 支持缩放倍数 1~15 倍。

**【举例】**

无。

**【相关主题】**

[FBIOGET\\_SCREEN\\_SIZE](#)

## 6.1.17 FBIOGET\_SCREEN\_SIZE

**【描述】**

获取图层在屏幕上的显示大小。

**【语法】**

```
int ioctl (int fd, FBIOGET_SCREEN_SIZE, cvi_fb_size *cvi_fb_size);
```

**【参数】**

参数名称	描述	输入/输出
fd	Framebuffer 设备文件描述符	输入
FBIO- GET_SCREEN_SIZE	ioctl 号	输入
cvi_fb_size	cvi_fb_size 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：cvi\_comm\_gfbg.h

**【注意】**

无。

**【举例】**

无。

**【相关主题】**

[FBIOPUT\\_SCREEN\\_SIZE](#)