

D1-H Linux G2D 开发指南

版本号: 2.1 发布日期: 2021.4.10





版本历史

版本号 日期 制/修订人 内容描述 1. 创建该文档。 $^{\circ}1.0$ \$2020.6.30° **A**WA1572≫ 1. 更新适配 linux5.4 2.0 2020.11.18 AWA1639 1. 添加输出宽度限制说明 2.1 2021.4.10 AWA1693

Weiking Weiking Weiking

版权所有。 珠海全志科技股份有限公司。 保留一切权利



目 录

| 1 | 前言 | | 1 |
|----|-------|--|----------------|
| | 101 | 文档简介 | 1 |
| N | 1.2 | 目标读者 🖑 | 1 |
| | 1.3 | 适用范围 | 1 |
| _ | 1#14 | -A/I | _ |
| 2 | 模块 | | 2 |
| | 2.1 | 模块功能介绍 | 2 |
| | | 2.1.1 矩形填充 (fill color rectgngle) | |
| | | 2.1.2 旋转和镜像 (rotate and mirror) | 3 |
| | | 2.1.3 alpha blending | 4 |
| | | 2.1.4 colorkey | 5 |
| | | 2.1.5 缩放 (Stretchblt) | 5 |
| | | 2.1.6 二元光栅操作 (rop2) | 6 |
| | 2.2 | 2.1.7 三元光栅操作 (maskblt rop3) | 6 |
| | | 相关术语介绍 | 7 |
| N | CUTIC | 2.2.1、硬件术语。 | 7 |
| 11 | | 2.2.2 软件术语 | 7 |
| | 2.3 | | 8 |
| | | 2.3.1 Device Tree 配置说明 | 8 |
| | | | 8 |
| | | 33.372.37.37.47.47.47.47.47.47.47.47.47.47.47.47.47 | 9 |
| | 2.5 | 驱动框架介绍 | 9 |
| 3 | 模块 | 接口说明 1 | l1 |
| | | | 11 |
| | | | 11 |
| | | | 12 |
| | | | 12 |
| | | | 14 |
| | , O | | 16 |
| N | OUT; | 3.1.5 g2d_blt_flags_h | 17. |
| | | | 18 |
| | | | 18 |
| | | | 20 |
| | | 0 0 | 21 |
| | | | 21 |
| | | 3.1.11 g2d ck | |
| | | 5 _ | 22 |
| | | 3.1.12 g2d_alpha_mode_enh | 22 22 |
| | | 3.1.12 g2d_alpha_mode_enh 2 3.1.13 g2d_color_gmt 2 | |
| | | 3.1.12 g2d_alpha_mode_enh 2 3.1.13 g2d_color_gmt 2 3.1.14 g2d_scan_order(version 1.0) 2 | 22 |
| | | 3.1.12 g2d_alpha_mode_enh 2 3.1.13 g2d_color_gmt 2 3.1.14 g2d_scan_order(version 1.0) 2 3.1.15 g2d_blt(version 1.0) 2 | 22 23 |
| | | 3.1.12 g2d_alpha_mode_enh 2 3.1.13 g2d_color_gmt 2 3.1.14 g2d_scan_order(version 1.0) 2 3.1.15 g2d_blt(version 1.0) 2 3.1.16 g2d_fillrect(version 1.0) 2 | 22 23 23 |



| ALLWIN | NEA | 文档密级: | 秘密 |
|--------|-----------------------------|--------|-----------|
| | 3.1.18 g2d_blt_h | | 25 |
| | 3.1.19 g2d_bld(version 1.0) | | 26 |
| 3.2 | 函数接口 | | 26 |
| QU4; | 3.2.1 1.0 版本接口 | OUT | 26 |
| Nor | 3.2.1 1.0 版本接口 | Me. | 26 |
| | 3.2.1.2 G2D_CMD_FILLRECT | | |
| | 3.2.1.3 G2D_CMD_STRETCHBLT | | 29 |
| | 3.2.1.4 G2D_CMD_PALETTE_TBL | | 30 |
| | 3.2.2 2.0 版本接口 | | |
| | 3.2.3 G2D_CMD_BITBLT_H | | 31 |
| | 3.2.4 G2D_CMD_BLD_H | | 33 |
| | 3.2.5 G2D_CMD_MASK_H | | 34 |
| 3.3 | 批处理接口 | | 35 |
| | 3.3.1 G2D_CMD_MIXER_TASK | | 36 |
| | 3.3.2 G2D_CMD_CREATE_TASK | | 39 |
| | 3.3.3 G2D_CMD_TASK_APPLY | | |
| | 3.3.4 G2D_CMD_TASK_DESTROY | | 41 |
| witing | 3.3.4 G2D_CMD_TASK_DESTROY | Neitus | 42 |
| 4 FAC | | | 43 |
| 4.1 | 常见问题 | | 43 |
| | 4.1.1 对齐问题 | | 43 |
| | 4.1.2 输出格式显示 | | 43 |
| | 4.1.3 输出宽度 | | 43 |

the neithe neithe neithe neithe neithe neithe neithe neithe neithe

.





插 冬

| | 2-1 | fill rectangle | 3 |
|----------------|----------|--------------------------|---------|
| Q _U | | | |
| SIT | weit 2-3 | alpha blending 1 | 4 weits |
| | 2-4 | alpha blending 2 | 5 |
| | 2-5 | colorkey | 5 |
| | 2-6 | scale and alpha blending | 6 |
| | 2-7 | mask | 7 |
| | 2-8 | menuconfig 5.4 | 8 |
| | 2-9 | G2D 代码框架图 | 10 |
| | 3-1 | mixerpara | 36 |

neixus neixus neixus neixus neixus

版权所有 ② 珠海全志科技股份有限公司。保留一切权利





1.1 文档简介

本文主要介绍 sunxi 平台 G2D 模块的功能、驱动结构及模块的配置和调用方法

1.2 目标读者

- G2D 驱动开发人员/维护人员
- 应用层的 G2D 模块使用者

1.3 适用范围

表 1-1: 适用产品列表

| 产品名称 | | 内核版本 | | | 驱动文件 | | | | |
|-----------|--------|-------|---------|--------|-------|--------|--------|--------|--------|
| D1-H Linu | | | Linux- | 5.4 | | g2d.c | | | |
| | | | | , | | | | | _ |
| Witig | WEITUS | Witig | PUTISIN | Neitig | Witig | WEITIG | neitus | PUTISH | NEITUS |





G2D 驱动主要实现图像旋转/数据格式/颜色空间转换,以及图层合成功能 (包括 alpha、colorkey、rotate、mirror、rop、maskblt) 等加速功能。

2.1 模块功能介绍

G2D 硬件特性如下:

- Input format: iYUV422/PYUV422UVC/PYUV420UVC/PYUV411UVC/ARGB8888/XRGB8888/ARGB4444/ARGB1555/RGB565
- Output format: iYUV422/PYUV422UVC/PYUV420UVC/PYUV411UVC/ARGB8888/XRGB8888/ARGB4444/ARGB1555/RGB565/Y8
- Any format convert function, R/B swap
- 1 channel scaling pipelines for scaling up/down
- Programmalbe source image size up to 2048*2048 pixels
- Programmalbe destination image size up to 2048*2048 pixels
- 4 tap scale filter in horizontal and 2 tap in vertical direction
- 32 programmable coefficients for each tap
- Color space conversion between RGB and YUV
- Clipping support
 - Straight line/Rectangle/Point
 - Block fill
- Rotate and mirror, The state of the sta
 - Rotation 90/180/270 counter-clockwise
 - Mirror horizontal/vertical
- ROP
 - BitBlt
 - StretchBlt
 - MaskBlt
- Colorkey support
 - Source colorkey
 - Destination colorkey



- Alpha blending support
 - Pixel alpha blending
 - Plane alpha blending
- Multi alpha blending
 - Output alpha configurable support

2.1.1 矩形填充 (fill color rectgngle)

填充矩形区域功能可以实现对某块区域进行预订的颜色值填充,如下图就填充了 0xFF0080FF 的 ARGB 值,该功能还可以通过设定数据区域大小实现画点和直线,同时也可以通过设定 flag 实现一种填充颜色和目标做 alpha 运算。



图 2-1: fill rectangle

2.1.2 旋转和镜像 (rotate and mirror)

旋转镜像主要是实现如下 Horizontal、Vertical、Rotate180°、Mirror45°、Rotate90°、Mirror135°、Rotate270°共7种操作。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

ieitus



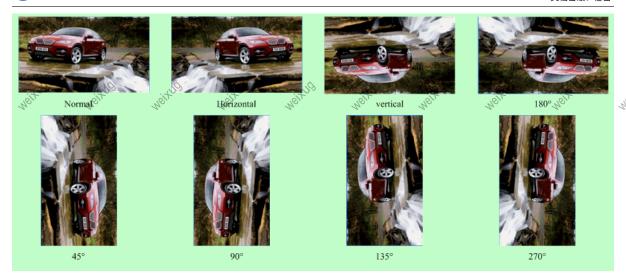


图 2-2: rotate and mirror

2.1.3 alpha blending

不同的图层之间可以做 alpha blending。Alpha 分为 pixel alpha、plane alpha、multi alpha 三种:

pixel alpha 意为每个像素自带有一个专属 alpha 值;

plane alpha 则是一个图层中所有像素共用一个 globe alpha 值;

multi alpha 则每个像素在代入 alpha 运算时的值为 globe alpha*pixel alpha,可以通过 G2D 驱动接口的 flag 去控制。

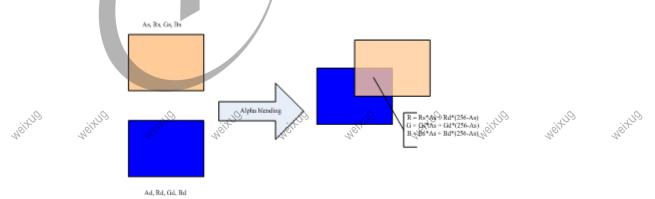


图 2-3: alpha blending 1

PULL

itio outio

版权所有 © 珠海全志科技股份有限公司。保留一切权利

leix119

OUTio,



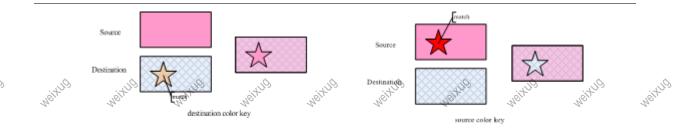
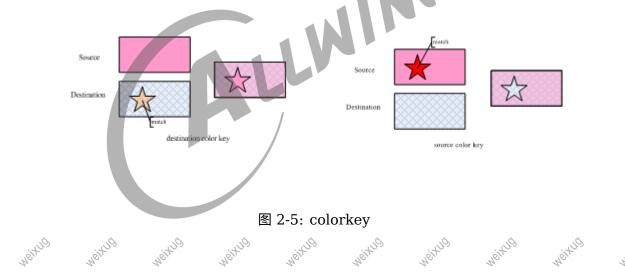


图 2-4: alpha blending 2

2.1.4 colorkey

不同 image 之间可以做 colorkey 效果:

- 左图中 destination 的优先级高于 source, destination 中 match 部分(橙色五角星部分),则被选择透过,显示为 source 与 destination 做 alpha blending 后的效果图。
- 右图中 source 的优先级高于 destination,则 source 中 match 部分(深红色五角星部分),则被选择透过,直接显示 destination 与 source 做 alpha blending 后的效果图。



2.1.5 缩放 (Stretchblt)

Stretchblt 主要是把 source 按照 destination 的 size 进行缩放,并最终与 destination 做 alpha blending、colorkey 等运算或直接旋转镜像后拷贝到目标,此接口在 1.0 版本上使用可以旋转和缩放一起用,但是 2.0 版本以后,缩放和旋转不可以同时操作。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

neitus

outis





2.1.6 二元光栅操作 (rop2)

我们在画线和填充区域的时候将画笔和目标像素组合得到新的目标像素。

2.1.7 三元光栅操作 (maskblt rop3)

对于图像有同样光栅操作用于生成各种特殊效果, 我们要处理的有三种像素: 源图像像素, 目标图像 像素, 画刷像素 (模板图像像素)。如下图所示, 从左上到右下分别是 src ptn mask dst。

版权所有 © 珠海全志科技股份有限公司。保留一切权利











2.2 相关术语介绍

2.2.1 硬件术语

表 2-1: 硬件术语列表

| 术语 | 说明 |
|-----|----------|
| G2D | 2D 图形加速器 |

2.2.2 软件术语

表 2-2: 软件术语列表

| 术语 | 说明 |
|-------------------|------------------------|
| Fill Rectangle | 对某块区域进行预定的颜色值填充 |
| Rotate And mirror | 对图像进行旋转或镜像操作 |
| Alpha Blending | 对两个图像按照预定的比例进行颜色混合 |
| Colorkey | 在两个图像叠加混合的时候,对特殊色做特殊过滤 |

版权所有 ⑥ 珠海全志科技股份有限公司。保留一切权利



2.3 模块配置介绍

2.3.1 Device Tree 配置说明

```
g2d:g2d@01480000{
    compatible = "allwinner,sunxi-g2d";
    reg = <0x0 0x01480000 0x0 0xbffff>;
    interrupts = <GIC_SPI 21 0x0104>;
    clocks = <&clk_g2d>;
    iommus = <&mmu_aw 5 1>;
    status = "okay";
}.
```

2.3.2 kernel menuconfig 配置说明

在命令行中进入 tina 根目录,执行 make kernel_menuconfig 进入配置主界面, 具体配置路径为:

Device Drivers->sunxi g2d driver

图 2-8: menuconfig 5.4



2.4 源码结构介绍

G2d 驱动的源代码位于内核在 drivers/char/sunxi g2d 目录下:

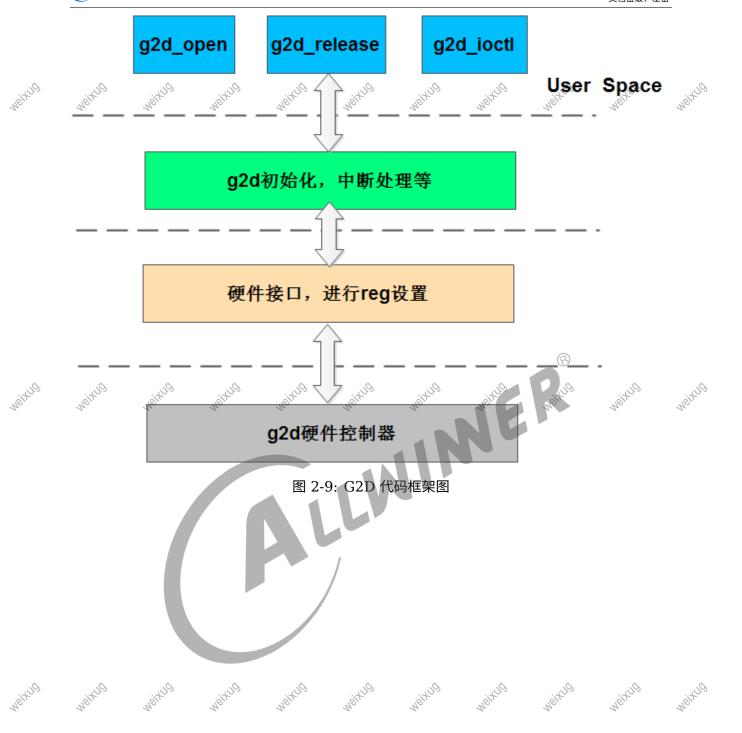
```
drivers/char/sunxi_g2d/g2d_rcq
   g2d_bld.c
   g2d_bld.h
  - g2d_bsp.h
   g2d.c
   g2d_driver_i.h
   g2d mixer.c
   g2d_mixer.h
   g2d_mixer_type.h
  g2d_ovl_u.c
   g2d_ovl_u.h
   g2d_ovl_v.c
   g2d_ovl_v.h
   g2d_rcq.c
   g2d_rcq.h
                                    Meixig Weixig
   g2d rotate.c
   g2d rotate.h
   g2d_rotate_type.h
   g2d_scal.c
   g2d scal.h
   g2d_top.c
  g2d_top.h
  - g2d_top_type.h
  g2d_wb.c
   g2d_wb.h
   Makefile
```

- g2d.c: 为 G2D 驱动顶层文件
- g2d_xxxx.c: 封装了相关功能的实现处理

2.5 驱动框架介绍

其代码框架如下图所示。





版权所有 © 珠海全志科技股份有限公司。探留一切权利 设计 10 10 10



3 模块接口说明

3.1 关键数据结构

3.1.1 g2d_blt_flags

作用

g2d_blt_flags 用于描述一个 bitblt 和 stretchblt 的 flag 属性信息

 $= 0 \times 000000800$,

```
• 定义
     typedef enum {
         G2D BLT NONE
                                       0×00000000,
 2
 3
         G2D_BLT_PIXEL_ALPHA
                                       0×00000001,
         G2D_BLT_PLANE_ALPHA
 4
                                     = 0 \times 000000002
 5
         G2D_BLT_MULTI_ALPHA
                                        0x00000004,
 6
         G2D_BLT_SRC_C0L0RKEY
                                     = 0 \times 000000008,
 7
         G2D_BLT_DST_COLORKEY
                                       0×00000010,
 8
         G2D_BLT_FLIP_HORIZONTAL
                                     = 0 \times 00000020,
         G2D_BLT_FLIP_VERTICAL
 9
                                     = 0 \times 00000040
10
                                       0x00000080,
         G2D BLT ROTATE90
         G2D BLT ROTATE180
                                     = 0 \times 00000100,
11
                                     = 0 \times 00000200,
12
         G2D_BLT_R0TATE270
13
         G2D_BLT_MIRROR45
                                     = 0 \times 00000400,
```

●成员说明

}g2d_blt_flags;

G2D BLT MIRROR135

14

```
G2D BLT NONE
                          - 纯拷贝
   G2D_BLT_PIXEL_ALPHA
                          - 点alpha标志
   G2D BLT PLANE ALPHA
                          - 面alpha标志
   G2D BLT MULTI ALPHA
                          - 混合alpha标志
   G2D_BLT_SRC_COLORKEY
                          - 源colorkey标志
   G2D_BLT_DST_COLORKEY
                          - 目标colorkey标志
   G2D_BLT_FLIP_HORIZONTAL - 水平翻转
   G2D_BLT_FLIP_VERTICAL
                          - 垂直翻转
   G2D_BLT_R0TATE90
9
                          - 逆时针旋转90度
10
   G2D_BLT_R0TATE180
                            逆时针旋转180度
   G2D_BLT_R0TATE270
                            逆时针旋转270度
11
   G2D_BLT_MIRROR45
                          - 镜像45度
```



G2D_BLT_MIRROR135

- 镜像135度

3.1.2 g2d_fillrect_flags

• 作用

g2d fillrect flags 用于描述一个 fillrect 属性信息

定义

```
typedef enum {
       G2D_FIL_NONE
                                  = 0 \times 000000000
3
       G2D_FIL_PIXEL_ALPHA
                                  = 0 \times 00000001
4
                                  = 0 \times 000000002
       G2D_FIL_PLANE_ALPHA
5
                                  = 0 \times 000000004
       G2D_FIL_MULTI_ALPHA
   }g2d_fillrect_flags;
                                                          INE
```

• 成员说明

```
G2D_FIL_NONE
                         纯填充
G2D_FIL_PIXEL_ALPHA - 填充区域和目标做点alpha G2D_FIL_PLANE_ALPHA - 填充区域和目标做面alpha
G2D_FIL_MULTI_ALPHA - 填充区域的alpha值*面alpha值后再和目标做alpha
```

3.1.3 g2d data_fmt(version 1.0)

作用

g2d_data_fmt 用于描述像素格式

定义

1.0 版本支持的图像格式

```
typedef enum {
2
     G2D_FMT_ARGB_AYUV8888
                                 = (0 \times 0),
3
     G2D_FMT_BGRA_VUYA8888
                                 = (0 \times 1),
4
     G2D_FMT_ABGR_AVUY8888
                                 = (0x2),
5
     G2D_FMT_RGBA_YUVA8888
                                 = (0x3),
     G2D_FMT_XRGB8888
                                 = (0x4),
```



```
G2D FMT BGRX8888
   7
                                                                               = (0x5),
  8
                G2D_FMT_XBGR8888
                                                                                   (0x6),
  9
                G2D_FMT_RGBX8888
                                                                                    (0x7),
10
                G2D FMT ARGB4444
                                                                                    (0x8),
11
                G2D_FMT_ABGR4444
                                                                                    (0x9),
12
                G2D_FMT_RGBA4444
                                                                               = (0 \times A)
13
                G2D FMT BGRA4444
                                                                                    (0xB),
14
                G2D_FMT_ARGB1555
                                                                                    (0xC),
15
                G2D_FMT_ABGR1555
                                                                                     (0xD),
16
                G2D FMT RGBA5551
                                                                                     (0xE),
17
                G2D FMT BGRA5551
                                                                                     (0xF),
18
                G2D_FMT_RGB565
                                                                                    (0×10),
19
                G2D_FMT_BGR565
                                                                                    (0 \times 11),
20
                G2D_FMT_IYUV422
                                                                                    (0x12),
                G2D_FMT_8BPP_M0N0
21
                                                                                    (0x13),
22
                                                                               = (0 \times 14),
                G2D_FMT_4BPP_MON0
23
                G2D_FMT_2BPP_MONO
                                                                               = (0x15),
24
                G2D_FMT_1BPP_MONO
                                                                               = (0x16),
25
                G2D_FMT_PYUV422UVC
                                                                               = (0 \times 17),
26
                G2D_FMT_PYUV420UVC
                                                                               = (0x18),
27
                G2D_FMT_PYUV411UVC
                                                                                   (0x19),
28
                                                                                                                      WE HATEL WE WE HATELD IN SECTION OF THE PARTY OF THE PART
29
           //只有输出才有的格式:
30
                      G2D FMT PYUV422
                                                                               = (0 \times 1A),
          G2D_FMT_PYUV420
31
                                                                               = (0 \times 1B)
32
                     G2D_FMT_PYUV411
                                                                               = (0 \times 10)
33
34
           //只有输入才支持的格式:
35
                      G2D_FMT_8BPP_PALETTE
                                                                                     = (0 \times 1D),
                                                                                    = (0x1E),
36
                      G2D_FMT_4BPP_PALETTE
                                                                                     = (0 \times 1F),
37
                      G2D_FMT_2BPP_PALETTE
38
                      G2D_FMT_1BPP_PALETTE
                                                                                     = (0x20)
                     G2D_FMT_PYUV422UVC_MB16 = (0x21),
39
                      G2D_FMT_PYUV420UVC_MB16 = (0x22),
40
41
                      G2D_FMT_PYUV411UVC_MB16 = (0x23),
42
                      G2D_FMT_PYUV422UVC_MB32 = (0x24),
43
                      G2D_FMT_PYUV420UVC_MB32 = (0x25),
44
                      G2D_FMT_PYUV411UVC_MB32 = (0x26),
45
                      G2D_FMT_PYUV422UVC_MB64 = (0x27),
46
                      G2D_FMT_PYUV420UVC_MB64 = (0x28),
47
                      G2D_FMT_PYUV411UVC_MB64 = (0x29),
48
                      G2D_FMT_PYUV422UVC_MB128 = (0x2A),
49
                      G2D_FMT_PYUV420UVC_MB128=(0x2B),
                     G2D_FMT_PYUV411UVC_MB128= (0x2C),
           }g2d data fmt;
```

成员说明

```
G2D FMT ARGB8888
                           : alpha(8bit)R(8bit)G(8bit)B(8bit)
   G2D FMT BGRA8888
                           : B(8bit)G(8bit)R(8bit)alpha(8bit)
   G2D FMT ABGR8888
                           : alpha(8bit)B(8bit)G(8bit)R(8bit)
   G2D FMT RGBA8888
                           : R(8bit)G(8bit)B(8bit)alpha(8bit)
5
6
   G2D_FMT_XRGB8888
                           : 24bit, RGB各8bit, alpha为高位自动填充为0xFF
   G2D_FMT_BGRX8888
                           : 24bit,BGR各8bit,alpha为低位自动填充为0xFF
8
   G2D_FMT_XBGR8888
                           : 24bit,BGR各8bit,alpha为高位自动填充为0xFF
9
   G2D_FMT_RGBX8888
                           : 24bit,RGB各8bit,alpha为低位自动填充为0xFF
10
```

版权所有人的珠海全志科技股份有限公司。保留一切权利人的



```
G2D FMT ARGB4444
11
                             : alpha(4bit)R(4bit)G(4bit)B(4bit)
12
    G2D FMT BGRA4444
                             : B(4bit)G(4bit)R(4bit)alpha(4bit)
13
    G2D_FMT_ABGR4444
                             : alpha(4bit)B(4bit)G(4bit)R(4bit)
14
    G2D FMT RGBA4444
                             : R(4bit)G(4bit)B(4bit)alpha(4bit)
    G2D FMT ARGB1555
                            alpha(1bit)R(5bit)G(5bit)B(5bit)
15
    G2D_FMT_BGRA1555
16
                             : B(5bit)G(5bit)R(5bit)alpha(1bit)
    G2D FMT ABGR1555
17
                             : alpha(1bit)B(5bit)G(5bit)R(5bit)
    G2D_FMT_RGBA1555
18
                             : R(5bit)G(5bit)B(5bit)alpha(1bit)
19
20
    G2D FMT RGB565
                         : R(5bit)G(6bit)B(5bit)
21
    G2D FMT BGR565
                         : B(5bit)G(6bit)R(5bit)
22
23
    G2D_FMT_IYUV422
                         : Interleaved YUV422
24
25
    G2D_FMT_8BPP_MONO
                         : 8bit per pixel mono
    G2D_FMT_4BPP_MONO
26
                         : 4bit per pixel mono
27
    G2D_FMT_2BPP_MONO
                         : 2bit per pixel mono
28
    G2D_FMT_1BPP_MON0
                         : 1bit per pixel mono
29
30
    G2D_FMT_PYUV422UVC : Planar UV combined only
31
    G2D_FMT_PYUV420UVC : Planar UV combined only
32
    G2D_FMT_PYUV411UVC : Planar UV combined only
33
    G2D FMT PYUV422
                         : Planar YUV422
35
    G2D FMT PYUV420
                         : Planar YUV420
36
    G2D_FMT_PYUV411
                         : Planar YUV411
37
    G2D_FMT_8BPP_PALETTE: 8bit per pixel palette only for input
38
    G2D_FMT_4BPP_PALETTE: 4bit per pixel palette only for input
39
    G2D_FMT_2BPP_PALETTE: 2bit per pixel palette only for input
40
    G2D_FMT_1BPP_PALETTE: 1bit per pixel palette only for input
41
42
43
    G2D FMT PYUV422UVC MB16: 16x16 tile base planar uv combined only for input
    G2D_FMT_PYUV420UVC_MB16: 16x16 tile base planar uv combined only for input
    {\tt G2D\_FMT\_PYUV411UVC\_MB16:\ 16\times16\ tile\ base\ planar\ uv\ combined\ only\ for\ input}
45
46
    G2D_FMT_PYUV422UVC_MB32: 16x16 tile base planar uv combined only for input
47
    G2D_FMT_PYUV420UVC_MB32: 16x16 tile base planar uv combined only for input
    G2D_FMT_PYUV411UVC_MB32: 16x16 tile base planar uv combined only for input
48
49
    G2D_FMT_PYUV422UVC_MB64: 16x16 tile base planar uv combined only for input
    G2D_FMT_PYUV420UVC_MB64: 16x16 tile base planar uv combined only for input
51
    G2D_FMT_PYUV411UVC_MB64: 16x16 tile base planar uv combined only for input
    G2D_FMT_PYUV422UVC_MB128: 16x16 tile base planar uv combined only for input
    G2D_FMT_PYUV420UVC_MB128: 16x16 tile base planar uv combined only for input
    G2D_FMT_PYUV411UVC_MB128; 16x16 tile base planar uv combined only for input
```

3.1.4 g2d pixel seq(version 1.0)

作用

g2d pixel seq 用于描述像素序列

定义



```
typedef enum {
 2
         G2D_SEQ_NORMAL
                                        = 0 \times 0.
 3
         G2D_SEQ_VYUY
                                        = 0 \times 1.
 4
    G2D_SEQ_VUVU
                                        = 0x2
 5
                                        = 0x3,
 6
                                        , 0x4 €
 7
         G2D_SEQ_P01
                                        = 0x5,
 8
         G2D_SEQ_P3210
                                        = 0x6,
 9
         G2D_SEQ_P0123
                                        = 0x7
10
         G2D SEQ P76543210
                                        = 0x8,
11
         G2D SEQ P67452301
                                        = 0x9,
12
         G2D_SEQ_P10325476
                                        = 0xA,
13
         G2D_SEQ_P01234567
                                        = 0xB,
14
         G2D_SEQ_2BPP_BIG_BIG
                                        = 0xC,
15
         G2D_SEQ_2BPP_BIG_LITTER
                                        = 0xD,
16
         G2D_SEQ_2BPP_LITTER_BIG
                                        = 0xE,
17
         G2D\_SEQ\_2BPP\_LITTER\_LITTER = 0xF,
18
         G2D_SEQ_1BPP_BIG_BIG
                                        = 0 \times 10,
19
         G2D_SEQ_1BPP_BIG_LITTER
                                        = 0x11,
20
         G2D SEQ 1BPP LITTER BIG
                                        = 0 \times 12,
21
         G2D\_SEQ\_1BPP\_LITTER\_LITTER = 0x13,
22
       }g2d_pixel_seq;
```

• 成员说明

```
G2D SEQ NORMAL
                             Normal sequence
    //for interleaved yuv422
 3
                            : pixel 0在低16位
    G2D SEQ VYUY
 4
    G2D_SEQ_YVYU
                            : pixel 1在低16位
 5
 6
 7
    // for uv_combined yuv420
                              Planar VU combined only
    G2D_SEQ_VUVU
 8
 9
10
    // for 16bpp rgb
11
    G2D_SEQ_P10
                            : pixel 0在低16位
12
    G2D_SEQ_P01
                            : pixel 1在低16位
13
    // planar format or 8bpp rgb
14
                            : pixel 0在低8位
15
    G2D_SEQ_P3210
                         pixel 3在低8位
    G2D_SEQ_P0123
16
17
18
    // for 4bpp rgb
19
    G2D SEQ P76543210
                               7,6,5,4,3,2,1,0
20
    G2D_SEQ_P67452301
                            :
                                6,7,4,5,2,3,0,1
21
    G2D_SEQ_P10325476
                                1,0,3,2,5,4,7,6
22
    G2D_SEQ_P01234567
                                0,1,2,3,4,5,6,7
23
    // for 2bpp rgb
24
    G2D_SEQ_2BPP_BIG_BIG
26
    15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0
27
28
    G2D_SEQ_2BPP_BIG_LITTER :
29
    12,13,14,15,8,9,10,11,4,5,6,7,0,1,2,3
30
31
    G2D_SEQ_2BPP_LITTER_BIG :
    3,2,1,0,7,6,5,4,11,10,9,8,15,14,13,12
```



```
34
   G2D_SEQ_2BPP_LITTER_LITTER :
35
   0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
36
37
   // for 1bpp rgt
38
   G2D_SEQ_1BPP_BIG_BIG
39
   31,30,29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0
40
41
   G2D_SEQ_1BPP_BIG_LITTER
42
   24,25,26,27,28,29,30,31,16,17,18,19,20,21,22,23,8,9,10,11,12,13,14,15,0,1,2,3,4,5,6,7
43
44
   G2D SEQ 1BPP LITTER BIG
45
   7,6,5,4,3,2,1,0,15,14,13,12,11,10,9,8,23,22,21,20,19,18,17,16,31,30,29,28,27,26,25,24
46
47
   G2D_SEQ_1BPP_LITTER_LITTER
```

3.1.5 g2d blt flags h

作用

g2d blt flags h 定义二元光栅操作码

定义

```
Weight Weight
    typedef enum {
2
        G2D_BLT_NONE_0 = 0 \times 0,
3
        G2D_BLT_BLACKNESS,
 4
        G2D_BLT_NOTMERGEPEN,
 5
        G2D BLT MASKNOTPEN,
 6
        G2D_BLT_NOTCOPYPEN,
 7
        G2D BLT MASKPENNOT,
 8
        G2D BLT NOT,
9
        G2D_BLT_XORPEN,
10
        G2D_BLT_NOTMASKPEN,
      G2D_BLT_MASKPEN,
11
        G2D_BLT_NOTXORPEN,
12
13
        G2D_BLT_NOP,
14
        G2D_BLT_MERGENOTPEN,
15
        G2D_BLT_COPYPEN,
16
        G2D_BLT_MERGEPENNOT,
17
        G2D_BLT_MERGEPEN,
18
        G2D_BLT_WHITENESS = 0x000000ff,
19
20
        G2D R0T 90
                           0x00000100,
21
        G2D R0T 180
                           0x00000200,
22
        G2D_R0T_270
                           0x00000300,
23
        G2D_R0T_H
                           0x00001000,
24
        G2D_R0T_V
                           0 \times 00002000,
25
26
        G2D_SM_DTLR_1 = 0 \times 100000000,
      g2d_blt_flags_h;
```



• 成员说明

```
G2D_BLT_NONE
                  单个源操作
   对使用与物理调色板的索引的相关的色彩来填充目标矩形图域,(对缺省的物理调色板,该颜色为黑色的形态。
 3
 6
   G2D_BLT_NOTMERGEPEN dst = \sim (dst+src) :
   G2D_BLT_MASKNOTPEN dst =~src&dst
   G2D_BLT_NOTCOPYPEN dst =~src
9
   G2D_BLT_MASKPENNOT dst =src&~dst
10
11
   //使目标矩形区域颜色取反
   G2D_BLT_NOT dst = -dst
12
   G2D_BLT_X0RPEN dst =src^dst
13
14
   G2D_BLT_NOTMASKPEN dst =~(src&dst)
15
   G2D_BLT_MASKPEN dst =src&dst
16
   G2D_BLT_NOTXORPEN dst =~(src^dst)
17
   G2D_BLT_NOP dst =dst
   G2D_BLT_MERGENOTPEN dst =~src+dst
   G2D BLT COPEPEN dst =src
   G2D BLT_MERGEPENNOT dst =src+~dst
   G2D_BLT_MERGEREN
21
                     dst ∈src+dst
   从使用与物理调色板中索引1有关的颜色填充目标矩形区域(对于缺省物理调色板来说,这
22
   G2D BLT WHITE
                  WHITENESS
```

3.1.6 g2d_image(version 1.0)

作用

g2d_image 用于描述 image 属性信息

定义

```
typedef struct {
                     addr[3];
2
     __u32
3
     __u32
                     w;
4
      _u32
                     h;
     g2d_data_fmt
                     format;
     g2d_pixel_seq
                     pixel_seq;
   }g2d_image;
```

• 成员说明

```
addr[3]:
            图像帧的基地址,对于UV combined,addr[0,1]有效,planar类型addr[0,1,2]有效,其他addr[0]
      有效
            图像帧的宽
  w:
           图像帧的高
3
  h:
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利





```
4 format: 图像帧buffer的像素格式,详见g2d_data_fmt
5 pixel_seq: 图像帧buffer的像素序列,详见g2d_pixel_seq
```

neitus

3.1.7 g2d_image_enh

• 作用

g2d_image_enh 主要描述图片的宽高、存放地址、是否做 Clip 处理,是否为预乘等。

定义

```
typedef struct {
2
     int
                   bbuff;
                                    Weight Weight
3
         _u32
                   color;
4
       g2d_fmt_enh format;
     u32
5
                   laddr[3];
       __u32 __weixus
 6
                   haddr[3];
                   width;
       __u32
 8
                   height;
9
       __u32
                   align[3];
10
       g2d_rect
                   clip_rect;
11
                   gamut;
        u32
12
                   bpremul;
13
                   alpha;
        __u8
14
       g2d_alpha_mode_enh mode;
   } g2d_image_enh;
```

• 成员说明

```
成员
             作用
   format
              : 图格式
              : 起始低位地址。
   laddr
            起始高位地址
   haddr
   width
              : 图宽度 (in pixel)
             : 图高度 (in pixel)
   height
   pitch
             : Buffer的pitch
   clip_rect
             : R0I矩形
   gamut
              : 图的色域
10
   bpremul
              : 是否为预乘
11
   alpha
              : 面alpha值
              : alpha模式设置
   mode
```

3.1.8 g2d_fmt_enh

作用

neitus



g2d fmt enh 用于描述 G2D 模块支持的格式

定义

```
typedef enum{
        G2D_FORMAT_ARGB8888,
 3
        G2D_FORMAT_ABGR8888,
 4
        G2D_FORMAT_RGBA8888,
 5
        G2D_FORMAT_BGRA8888,
 6
        G2D_F0RMAT_XRGB8888,
 7
        G2D_FORMAT_XBGR8888,
 8
        G2D_FORMAT_RGBX8888,
 9
        G2D_FORMAT_BGRX8888,
10
        G2D_FORMAT_RGB888,
11
        G2D_FORMAT_BGR888,
        G2D_FORMAT_RGB565,
12
13
        G2D FORMAT BGR565,
14
        G2D_FORMAT_ARGB4444,
15
        G2D_F0RMAT_ABGR4444,
                                             ASHID WONDS
16
        G2D_FORMAT_RGBA4444,
17
        G2D_FORMAT_BGRA4444,
        G2D_F0RMAT_ARGB1555
18
    G2D_FORMAT_ABGR1555,
19
20
        G2D_F0RMAT_RGBA5551,
21
        G2D_FORMAT_BGRA5551,
22
        G2D_FORMAT_ARGB2101010,
23
        G2D FORMAT ABGR2101010,
24
        G2D FORMAT RGBA1010102,
        G2D_FORMAT_BGRA1010102,
25
26
        /* invailed for UI channel */
27
        G2D_FORMAT_IYUV422_V0Y1U0Y0 = 0x20,
28
        G2D_FORMAT_IYUV422_Y1V0Y0U0,
29
30
        G2D_FORMAT_IYUV422_U0Y1V0Y0,
31
        G2D_F0RMAT_IYUV422_Y1U0Y0V0,
32
        G2D FORMAT YUV422UVC V1U1V0U0,
33
34
        G2D FORMAT YUV422UVC U1V1U0V0,
35
        G2D_F0RMAT_YUV422_PLANAR,
36
37
    G2D_FORMAT_YUV420UVC_V1U1V0U0 =

G2D_FORMAT_YUV420UVC_U1V1U0V0
        G2D_FORMAT_YUV420UVC_V1U1V0U0 = 0x28,
38
39
40
41
        G2D_FORMAT_YUV411UVC_V1U1V0U0 = 0x2c,
42
        G2D_F0RMAT_YUV411UVC_U1V1U0V0,
43
        G2D_FORMAT_YUV411_PLANAR,
44
45
        G2D_FORMAT_Y8 = 0 \times 30,
46
47
        /* YUV 10bit format */
48
        G2D_FORMAT_YVU10_P010 = 0x34,
49
50
        G2D_FORMAT_YVU10_P210 = 0x36,
51
52
        G2D_FORMAT_YVU10_444 = 0x38,
53
        G2D_FORMAT_YUV10_444 = 0x39,
    }g2d_fmt_enh;
```



3.1.9 g2d_rop3_cmd_flag

◆作用

Neit^{us} Neit^{us}

g2d rop3 cmd flag 用于定义三元光栅操作码

● 定义

```
typedef enum {
 2
        G2D_R0P3_BLACKNESS
                               = 0 \times 00,
 3
        G2D_ROP3_NOTSRCERASE = 0x11,
 4
        G2D_ROP3_NOTSRCCOPY = 0x33,
 5
        G2D_R0P3_SRCERASE
                               = 0 \times 44.
 6
        G2D_R0P3_DSTINVERT
                               = 0x55,
 7
        G2D_R0P3_PATINVERT
                               = 0x5A,
 8
        G2D_R0P3_SRCINVERT
                               = 0x66,
                                                   Weithig Weithig
 9
        G2D_R0P3_SRCAND
                               = 0x88,
10
        G2D_ROP3_MERGEPAINT = 0xBB,
11
        G2D_R0P3_MERGECOPY = 0xC0,
12
        G2D_R0R3_SRCCOPY
                               = 0 \times CC
13
        G2D ROP3 SRCPAINT
                               = 0 \times EE
14
        G2D R0P3 PATC0PY
                               = 0 \times F0,
15
        G2D ROP3 PATPAINT
                               = 0xFB
16
        G2D ROP3 WHITENESS
                               = 0xFF,
    }g2d_rop3_cmd_flag;
```

● 成员说明

```
dst = BLACK
   G2D_R0P3_BLACKNESS
                      dst = (NOT src) AND (NOT dst)
   G2D ROP3 NOTSRCERASE
   G2D_R0P3_N0TSRCC0PY
                      dst = (NOT src)
                                           :将源矩形区域颜色取反,拷贝到目标矩形区域
                      dst = src AND (NOT dst )
dst = (NOT dst)
   G2D_R0P3_SRCERASE
   G2D ROP3 DSTINVERT
   G2D_R0P3 PATINVERT
                      dst = pattern XOR dst
                                            :通过使用布尔型的异或(XOR)操作符将特定模式和目标矩形
     区域颜色合并
   G2D_R0P3_SRCINVERT
                      dst = src XOR dst
                                            :通过使用布尔型的异或(XOR)操作符将源和目标矩形区域颜
       色合并
   G2D_R0P3_SRCAND
                      dst = srcAND dst
                                            :通过使用与操作符将源和目标矩形区域颜色值合并
8
9
   G2D_R0P3_MERGEPAINT
                      dst = (NOT src) OR dst
                                             :通过使用布尔型的或(OR)操作符将反向的源矩形区域的颜
       色与目标矩形区域颜色合并
10
   G2D ROP3 MERGECOPY
                      dst = (src AND pattern)
11
   G2D_R0P3_SRCC0PY
                      dst = src
                                             : 将源矩形区域直接拷贝到目标矩形区域
   G2D ROP3 SRCPAINT
                      dst = src OR dst
                                             :通过使用布尔型的或(OR)操作符将源和目标矩形区域颜色
       合并
   G2D ROP3 PATCOPY
                      dst = pattern
   G2D ROP3 PATPAINT
                      dst = DPSnoo
                                             :通过使用布尔型的或(OR)操作符将源矩形区域取反后的颜
       色值与特定模式的颜色合并,然后使用OR操作符与该操作的结果与目标矩形区域内的颜色合并.
   G2D_R0P3_WHITENESS
                      dst = WHITE
```

QU4;

版权所有 © 珠海全志科技股份有限公司。保留一切权利

0/4/6

1¹1⁰ 2



3.1.10 g2d_bld_cmd_flag

作用 g2d bld cmd flag 定义 BLD 操作命令

定义

```
typedef enum {
         G2D BLD CLEAR
                              = 0 \times 00000001,
 2
 3
         G2D_BLD_C0PY
                               = 0 \times 000000002,
 4
         G2D_BLD_DST
                              = 0 \times 000000003,
 5
         G2D_BLD_SRCOVER
                              = 0 \times 00000004,
 6
         G2D_BLD_DSTOVER
                              = 0 \times 00000005,
                                         neikus neikus nekus
         G2D_BLD_SRCIN
                              = 0 \times 000000006,
 8
         G2D_BLD_DSTIN
                              = 0 \times 00000007,
 9
         G2D_BLD_SRCOUT
                              = 0 \times 000000008,
10
         G2D_BLD_DSTOUT
                              = 0 \times 000000009,
       G2D_BLD_SRCATOP
11
                              = 0 \times 00000000a
                             ≥ 0x0000000b,
12
         G2D_BLD_DSTATOP
13
         G2D_BLD_X0R
                              = 0 \times 00000000c
14
         G2D_CK_SRC
                              = 0 \times 00010000,
15
         G2D_CK_DST
                              = 0 \times 00020000,
    }g2d_bld_cmd_flag;
```

3.1.11 g2d ck

作用

g2d ck 定义了 colorkey 操作的参数

```
定义
```

```
typedef struct {
2
       int match_rule;
       __u32 max_color;
3
         _u32 min_color;
   }g2d_ck;
```

• 成员说明

```
match_rule 当match_rule为假时,Color Min=<Color<=Color Max表示满足匹配条件
当match_rule为真时,Color>Color Max or Color <Color Min表示满足匹配条件
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利





```
ck_max_color
                Color Max
ck_min_color
                Color Min
```

3.1.12 g2d_alpha_mode_enh

作用

g2d alpha mode enh 定义进行 alpha blend 操作时,选择的 alpha mode

• 定义

```
typedef enum{
2
       G2D_PIXEL_ALPHA,
3
       G2D_GLOBAL_ALPHA,
       G2D_MIXER_ALPHA,
   }g2d_alpha_mode_enh;
```

• 成员说明

```
MIN
             作用
G2D_PIXEL_ALPHA
             点alpha
G2D GLOBAL ALPHA 面alpha
G2D MIXER ALPHA
            混合alpha
```

3.1.13 g2d_color_gmt

作用

 $g2d_color_gmt$ 定义进行位操作时,选择的颜色空间

定义

```
typedef enum{
2
       G2D_BT601,
3
       G2D_BT709,
4
       G2D_BT2020,
   }g2d_color_gmt;
```



3.1.14 g2d_scan_order(version 1.0)

作用

g2d_scan_order 定义进行 alpha blend 操作时,选择的图像扫行模式

定义

```
1  enum g2d_scan_order {
2    G2D_SM_TDLR = 0x00000000,
3    G2D_SM_TDRL = 0x00000001,
4    G2D_SM_DTLR = 0x00000002,
5    G2D_SM_DTRL = 0x00000003,
6 };
```

• 成员说明

```
1 G2D_SM_TDLR Top to down, Left to right
2 G2D_SM_DTLR Down to top, Left to right
3 G2D_SM_DTRL Top to down, Right to left
4 G2D_SM_DTRL Down to top, Left to right
```

3.1.15 g2d_blt(version 1.0)

作用

g2d blt 用于一个源和目标做 blt 的信息

定义

```
typedef struct {
        g2d\_blt\_flags
                               flag;
3
        g2d_image
                               src_image;
4
        g2d_rect
                               src_rect;
5
        g2d_image
                               dst_image;
6
         _s32
                               dst_x;
        __s32
7
                               dst_y;
        __u32
8
                               color;
9
         u32
                               alpha;
   }g2d_blt;
```

• 成员说明



```
flag
             : block transfer标志,详见g2d blt flags
             : 源图像信息,详见g2d_image
  src_image
3
             : 目标图像信息,详见g2d_image
  dst_image
4
             : 目标矩形左上角x
  dst_x
             : 目标矩形左上角y
5
  ds t√y
  color
            《 : colorkey颜色
  alpha
             : 面alpha值
```

3.1.16 g2d_fillrect(version 1.0)

作用

g2d_fillrect 用于描述一个 fill rectangle 参数信息

定义

```
typedef struct {
      g2d_fillrect_flags
                        flag;
3
      g2d_image
                        dst_image;
4
      g2d rect
                        dst_rect;
5
      __u32
                        color;
       u32
                        alpha;
  }g2d_fillrect;
```

• 成员说明

```
flag : 填充矩形标志,详见g2d_fillrect_flags
dst_image : 目标图像信息,详见g2d_image
dst_rect : 目标矩形信息,x/y/w/h-左上角x/左上角y/宽/高
color : 填充颜色
alpha : 面alpha值
```

3.1.17 g2d stretchblt(version 1.0)

• 作用

g2d stretchblt 用于描述一个 stretchblt 参数信息

● 定义



```
typedef struct {
2
       g2d_blt_flags
                              flag;
3
       g2d_image
                              src_image;
4
       g2d_rect
                              src_rect;
    √°g2d_image√
5
                            %dst_image;>
6
       g2d_rect
                             dst_rect;
7
       __u32
                              color;
8
        __u32
                              alpha;
   } g2d_stretchblt;
```

• 成员说明

```
flag
            : block transfer标志,详见g2d_blt_flags
  src_image
            : 源图像信息,详见g2d_image
            : 源矩形信息,x/y/w/h-左上角x/左上角y/宽/高
  src_rect
            : 目标图像信息,详见g2d_image
  dst_image
  dst_rect
            : 目标矩形信息, x/y/w/h-左上角x/左上角y/宽/高
6
  color
            : colorkey颜色
                                     Weixing Weixing
            : 面alpha值
  alpha
```

3.1.18 g2d blt h

作用

g2d_blt_h 实现对 foreground 带缩放的 ROP2 处理。

定义

```
typedef struct {
2
       g2d_blt_flags_h
                              flag_h;
3
       g2d_image_enh
                              src_image_h;
4
       g2d_image_enh
                             dst_image_h;
5
       __u32 🗳
                             color;
6
         u32
                             alpha;
   }g2d_blt_h;
```

• 成员说明

```
flag_h
                : blt操作flag标志,增强版标志
              : 源图像信息,增强版的图像参数,详见g2d_image_enh
  src_image_h
3
  dst_image_h
              : 目标图像信息,增强版的图像参数
  color
                  : colorkey颜色
5
                  : 面alpha值
  alpha
```



3.1.19 g2d_bld(version 1.0)

作用

Neitus neitus neitus neitus neitus neitus neitus neitus neitus

g2d_bld 实现两幅图的 BLD 和 colorkey 操作。

定义

```
typedef struct {
   g2d_bld_cmd_flag    bld_cmd;
   g2d_image_enh    dst_image_h;
   g2d_image_enh    src_image_h;
   g2d_ck     ck_para;
}g2d_bld;/* blending enhance */
```

成员说明

bld_cmd : blending的操作flag标志,增强版标志 rc_image_h : 源图像信息,增强版的图像参数 dst_image_h : 目标图像信息,增强版的图像参数 ck_para : colorkey参数

3.2 函数接口

3.2.1 1.0 版本接口

3.2.1.1 G2D_CMD_BITBLT

- ◆ 作用: BITBLT 函数实现的是两个图层的运算,比如源拷贝到目标;源旋转放入目标;源和目标 做 alpha blending/colorkey 后拷贝到目标
- 原型:

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

- 参数:
 - fd: G2D 设备文件标识符cmd: G2D_CMD_BITBLT

OUTis



• arg: arg 为 g2d blt 结构体指针

● 返回:

• 0: 成功 • 其他: 失败

MEITU

weitu9

neit10

Meitus

neitur

weit!

weitu

NEIXUS

● 举例:

```
/* 输入/输出image buffer */
    g2d_image image_front,scn;
 3
    g2d_rect src_rect;
 4
    g2d_blt blit;
 5
    __s32 dst_x, dst_y;
 6
 7
    image_front.addr[0]
                            = mem_in;
 8
                            = 800;
    image_front.w
 9
                            = 480;
    image_front.h
    image_front.format
                            = G2D FMT ARGB8888;
10
11
    image_front.pixel_seq = G2D_SEQ_NORMAL;
                                       Weiking Weiking
12
13
    scn.addr[0]
                            = mem_out;
                           = 480; with
                           √∋ 800;
14
    scn
15
    sen.h
16
    scn.format
                            = G2D_FMT_RGBA8888;
17
    scn.pixel_seq
                            = G2D_SEQ_NORMAL;
18
    src rect.x
                            = 0;
                            = 0;
    src rect.y
                             480;
    src rect.w
21
    src_rect.h
                            = 272:
22
23
                            = 0:
    dst_x
24
                            = 0;
    dst_y
25
    /* 设置BITBLT flag标志: 做点alpha和水平翻转 */
26
    blit.flag = G2D_BLT_PIXEL_ALPHA| G2D_BLT_FLIP_HORIZONTAL;
27
28
    blit.color = 0xee8899;
29
    blit.alpha = 0 \times 73;
30
31
    /* 设置源imgae和源rect */
    blit.src_image.addr[0] = image_front.addr[0];
32
33
    blit.src_image.w
                            = image_front.w;
    blit.src_image.h = image_front.h;
blit.src_image.format = image_front.format;
34
35
36
    blit.src_image.pixel_seq= image_front.pixel_seq;
37
    blit.src_rect.x
                           = src rect.x;
38
    blit.src_rect.y
                           = src_rect.y;
39
    blit.src_rect.w
                            = src_rect.w;
40
    blit.src_rect.h
                            = src_rect.h;
41
    /* 设置目标imgae和目标rect */
    blit.dst_image.addr[0] = scn.addr[0];
    blit.dst image.w
                           = scn.w;
45
    blit.dst_image.h
                            = scn.h;
46 blit.dst_image.format = scn.format;
    blit.dst_image.pixel_seq= scn.pixel_seq;
48
    blit.dst_x
                           = dst_x;
49
    blit.dst_y
                            = dst_y;
50
```



```
if(ioctl(g2d_fd, G2D_CMD_BITBLT, &blit)<0)</pre>
52
53
         printf("G2D_CMD_BITBLT failed!\n");
54
    }
```

G2D CMD FILLRECT

- 作用: 用一种颜色的画点画直线及矩形填充,同时也能实现填充颜色和目标做 alpha blending
- 原型:

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

• 参数:

• fd: G2D 设备文件标识符

• cmd: G2D CMD FILLRECTS

• arg: arg 为 g2d fillrect 结构体指针

● 返回:

• 0: 成功

● 其他: 失败

• 举例:

```
Mestrice Washing
    /* 输出image buffer */
    g2d_image scn;
 3
    g2d_rect dst_rect;
    g2d fillrect fillrect;
    /* 设置FILLRECT标志: 做面alpha *
 6
                               = G2D_FIL_PLANE_ALPHA;
 7
    fillrect.flag
 8
    fillrect.color
                               = 0xFF345678;
9
    fillrect.alpha
                               = 0 \times 40;
10
11
    /* 设置目标image和目标rect */
   fillrect.dst_image.addr[0] = scn.addr[0];
12
13
   fillrect.dst_image.w
                              = scn.w;
14
   fillrect.dst_image.h
                              = scn.h;
15 fillrect.dst_image.format = scn.format;
16 fillrect.dst_image.pixel_seq= scn.pixel_seq;
17
   fillrect.dst_rect.x
                              = dst rect.x;
18
   fillrect.dst_rect.y
                              = dst_rect.y;
19
   fillrect.dst_rect.w
                              = dst_rect.w;
20
   fillrect.dst_rect.h
                              = dst_rect.h;
21
22
   if (ioctl(g2d_fd, G2D_CMD_FILLRECT, &fillrect) < 0) {</pre>
23
        printf("G2D_CMD_FILLRECT failed!\n");
24
```



3.2.1.3 G2D_CMD_STRETCHBLT

● 作用: STRETCHBLT 函数实现的是两个图层的运算,比如源缩放到目标大小后拷贝到目标; 源缩放到目标大小旋转放入目标,源缩放到目标大小后和目标做 alpha blending/colorkey 拷 贝到目标

• 原型:

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

- 参数:
 - fd: G2D 设备文件标识符
 - cmd: G2D CMD STRETCHBLT
 - arg: arg 为 g2d_stretchblt 结构体指针
- 返回:
- 0:成功。 其他: 失败
- 举例:

```
Retus Retus
    /* 输出image buffer →
   g2d_image image_front,scn;
    g2d_rect src_rect,dst_rect;
    g2d_stretchblt str;
5
 6
   image_front.addr[0]
                           = mem_in;
 7
    image_front.w
                           = 800;
    image_front.h
                           = 480;
                           = G2D_FMT_PYUV420UVC;
    image_front.format
    image_front.pixel_seq
                           = G2D_SEQ_NORMAL;
10
11
    image_front.addr[1]
                           = mem_in+ image_front.w*image_front.h;
12
   sch.addr[0]
                          ⊬≌ mem_out;
13
                           = 800;
14
    scn.w
15
    scn.h
                           = 480;
                           = G2D_FMT_ARGB8888;
16
    scn.format
                           = G2D SEQ NORMAL;
17
   scn.pixel_seq
18
   src_rect.x
                           = 0;
19
   src_rect.y
                           = 0;
   src rect.w
                           = 480;
   src rect.h
                           = 272;
   dst_rect.x
                           = 17;
23
   dst_rect.y
                           = 100;
24
   dst_rect.w
                           = 480;
25
                           = 272;
   dst_rect.h
26
   /* 设置STRETCHBLT标志:做点alpha和旋转90度 */
27
28
   str.flag = G2D_BLT_PIXEL_ALPHA|G2D_BLT_R0TATE90;
   str.color
                           = 0xee8899;
```





```
str.alpha
                           = 0x73;
31
32
    /* 设置源image和源rect */
33
   str.src_image.addr[0]
                           = image_front.addr[0];
34
   strosrc_image.addr[1]
                           image_front.addr[1];
35
    str.src_image.w
                           = image_front.w;
    str.src_image.h
36
                           = image_front.h;
    str.src_image.format = image_front.format;
37
38
    str.src_image.pixel_seq = image_front.pixel_seq;
    str.src rect.x
                          = src rect.x;
    str.src_rect.y
                           = src_rect.y;
41
    str.src_rect.w
                           = src_rect.w;
42
    str.src_rect.h
                           = src_rect.h;
43
   /* 设置目标image和目标rect */
44
   str.dst_image.addr[0] = scn.addr[0];
45
46
   str.dst_image.w
                           = scn.w;
47
   str.dst_image.h
                           = scn.h;
48
   str.dst_image.format = scn.format;
49
   str.dst_image.pixel_seq = scn.pixel_seq;
   str.dst_rect.x
                           = dst_rect.x;
   str.dst_rect.y
                           = dst_rect.y;
52
   str.dst rect.w
                           = dst rect.w;
                                                        Westing Westing
53
   str.dst_rect.h
                           = dst rect.h;
54
   if(ioctl(g2d_fd, G2D_CMD_STRETCHBLT, &str) < 0)
55
56
57
        printf("G2D_CMD_STRETCHBLT failed!\n");
58
```

3.2.1.4 G2D_CMD_PALETTE_TBL

- 作用: PALETTE_TAL 函数实现的是把查找表写入硬件 SDRAM,也只有在前面接口的源数据 format 设置为 palette 模式时才需要先使用这条命令
- 原型:

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

- 参数:
 - fd: G2D 设备文件标识符
 - cmd: G2D CMD PALETTE TBL
 - arg: arg 为 g2d palette 结构体指针
- 返回:
 - 0: 成功
 - 其他: 失败
- 举例:

OUTIS

版权所有 © 珠海全志科技股份有限公司。保留一切权利

TUS

itus



```
unsigned long length;
2
    /* 查找表数组 */
    unsigned long palette[0x100];
3
4
    g2d_palette pal;
    pal->pbuffer = &palette;
    pal.size = length;
8
    if(ioctl(g2d_fd, G2D_CMD_PALETTE_TBL, &pal)<0)</pre>
10
11
        printf("G2D_CMD_PALETTE_TBL failed!\n");
12
```

3.2.2 2.0 版本接口

3.2.3 G2D CMD BITBLT H

PROTOTYPE

int ioctl(int fd, int cmd, void *arg)

ARGUMENTS

```
cmd
           G2D_CMD_BITBLT_H
           arg为g2d_blt_h结构体指针
arg
```

• RETURNS

成功: 0,失败:失败号

DESCRIPTION

实现单幅图的缩放、格式转换等。实现对 foreground 带缩放的 ROP2 处理。

• DEMO with

```
/* 旋转功能 */
   blit.flag_h = G2D_R0T_90;
   blit.src_image_h.addr[0] = saddr[0];
   blit.src image h.format = G2D FORMAT ARGB8888;
   blit.src image h.mode = G2D GLOBAL ALPHA;
   blit.src_image_h.clip_rect.x = 0;
   blit.src_image_h.clip_rect.y = 0;
   blit.src_image_h.clip_rect.w = 1920;
   blit.src_image_h.clip_rect.h = 1080;
   blit.src_image_h.width = 1920;
11
   blit.src_image_h.height = 1080;
12 blit.src_image_h.alpha = 0xff;
13 blit.dst_image_h.addr[0] = daddr[0];
```



```
blit.dst image h.format = G2D FORMAT ARGB8888;
    blit.dst_image_h.mode = G2D_GL0BAL_ALPHA;
    blit.dst_image_h.clip_rect.x = 0;
17
    blit.dst_image_h.clip_rect.y = 0;
18
    blit.dst_image_h.clip_rect.w = 1920;
19
    blit.dst_image_h.clip_rect.h = 1080;
20
    blit.dst_image_h.alpha = 0xff;
21
    blit.dst_image_h.width = 1920;
22
    blit.dst_image_h.height = 1080;
23
24
    if(ioctl(g2d fd, G2D CMD BITBLT H ,(unsigned long)(&blit)) < 0)</pre>
25
        printf("[%d][%s][%s]G2D_CMD_BITBLT_H failure!\n",
26
     _LINE__, __FILE__,__FUNCTION__);
27
28
                return -1;
29
    }
30
31
    /* 缩放功能 */
32
    blit.flag_h = G2D_BLT_NONE_0;
    blit.src_image_h.addr[0] = saddr[0];
    blit.src_image_h.format = G2D_F0RMAT_ARGB8888;
    blit.src_image_h.mode = G2D_GL0BAL_ALPHA;
                                                       Weiting Weiting
    blit.src_image_h.clip_rect.x = 0;
    blit.src_image_h.clip_rect.y = 0;
38
    blit.src_image_h.clip_rect.w = 1280;
    blit.src_image_h.clip_rect.h = 800;
39
    blit.src_image_h.width = 1280;
40
41
    blit.src_image_h.height = 800;
42.
    blit.src_image_h.alpha = 0xff;
    blit.dst_image_h.addr[0] = daddr[0];
43
    blit.dst_image_h.format = G2D_FORMAT_ARGB8888;
45
    blit.dst_image_h.mode = G2D_GL0BAL_ALPHA;
    blit.dst_image_h.clip_rect.x = 0;
    blit.dst_image_h.clip_rect.y = 0;
48
    blit.dst_image_h.clip_rect.w = 1920;
49
    blit.dst_image_h.clip_rect.h = 1080;
50
    blit.dst_image_h.alpha = 0xff;
    blit.dst_image_h.width = 1920;
51
52
    blit.dst_image_h.height = 1080;
53
54
    if(ioctl(g2d_fd, G2D_CMD_BITBLT_H ,(unsigned long)(&blit)) < 0)</pre>
55
56
        printf("[%d][%s][%s]G2D_CMD_BITBLT_H failure!\n",
      LINE__, __FILE__,__FUNCTION__);
59
60
    /* 格式转换 */
61
62
    blit.flag_h = G2D_BLT_NONE_0;
63
    blit.src_image_h.addr[0] = saddr[0];
    blit.src_image_h.format = G2D_FORMAT_ARGB8888;
64
65
    blit.src_image_h.mode = G2D_GL0BAL_ALPHA;
66
    blit.src_image_h.clip_rect.x = 0;
    blit.src_image_h.clip_rect.y = 0;
    blit.src_image_h.clip_rect.w = 1280;
    blit.src_image_h.clip_rect.h = 800;
    blit.src_image_h.width = 1280;
70
71
    blit.src_image_h.height = 800;
    blit.src_image_h.alpha = 0xff;
72
    blit.dst_image_h.addr[0] = daddr[0];
```



```
blit.dst_image_h.format = G2D_F0RMAT_YUV420UVC_V1U1V0U0;
    blit.dst_image_h.mode = G2D_GLOBAL_ALPHA;
    blit.dst_image_h.clip_rect.x = 0;
    blit.dst_image_h.clip_rect.y = 0;
    blit.dst_image_h.clip_rect.w = 1280;
    blit.dst_image_h.clip_rect.h = 800;
80
    blit.dst_image_h.alpha = 0xff;
81
    blit.dst_image_h.width = 1280;
82
    blit.dst_image_h.height = 800;
84
    if(ioctl(g2d fd, G2D CMD BITBLT H ,(unsigned long)(&blit)) < 0)</pre>
85
86
        printf("[%d][%s][%s]G2D_CMD_BITBLT_H failure!\n",
      _LINE__, __FILE__,__FUNCTION__);
87
88
                return -1;
89
```

3.2.4 G2D CMD BLD H

PROTOTYPE

int ioctl(int fd, int cmd, void *arg)

ARGUMENTS

```
cmd G2D_CMD_BLD_H
arg arg为g2d_bld结构体指针
```

- RETURNS
 - 成功: 0,失败:失败号
- DESCRIPTION

实现两幅图的 BLD(porter-duff) 操作

• NDEMO Noite

```
blend.bld_cmd = G2D_BLD_COPY;
blend.src_image_h.mode = G2D_GL0BAL_ALPHA;
blend.src_image_h.format = G2D_FORMAT_ARGB8888;
blend.src_image_h.alpha = 128;
blend.src_image_h.clip_rect.x = 0;
blend.src_image_h.clip_rect.y = 0;
blend.src_image_h.clip_rect.w = 1280;
blend.src_image_h.clip_rect.h = 800;
blend.src_image_h.width = 1280;
blend.src_image_h.width = 1280;
blend.src_image_h.height = 800;
blend.src_image_h.height = 800;
blend.dst_image_h.mode = G2D_GL0BAL_ALPHA;
blend.dst_image_h.format = G2D_FORMAT_ARGB8888;
blend.dst_image_h.alpha = 128;
```



```
blend.dst_image_h.clip_rect.x = 0;
    blend.dst_image_h.clip_rect.y = 0;
    blend.dst_image_h.clip_rect.w = 1280;
17
    blend.dst_image_h.clip_rect.h = 800;
18
    blend.dst_image_h.width = 1280;
    blend.dst_image_h.height = 800;
19
20
21
    if(ioctl(g2d\_fd,\ G2D\_CMD\_BLD\_H\ ,(unsigned\ long)(\&blend))\ <\ 0)
22
23
    printf("[%d][%s][%s]G2D CMD BLD H failure!\n",
              _LINE__, __FILE__,__FUNCTION__);
24
25
             return -1;
26
```

3.2.5 G2D CMD MASK H

• PROTOTYPE

int ioctl(int fd, int cmd, void *arg)

weith wei

ARGUMENTS

```
cmd G2D_CMD_MASK_H
arg arg为g2d_maskblt结构体指针
```

• RETURNS

成功: 0,失败:失败号

- DESCRIPTION
 - 根据掩膜图和光栅操作码对 src、pattern 和 dst 进行操作,并将结果保存到 dst 中.
- DEMO

```
mask.back_flag = G2D_R0P3_NOTSRCCOPY;
    mask.fore_flag = G2D_R0P3_SRCINVERT;
 3
    mask.src_image_h.clip_rect.x = 0;
    mask.src_image_h.clip_rect.y = 0;
    mask.src_image_h.clip_rect.w = 1280;
    mask.src_image_h.clip_rect.h = 800;
    mask.src_image_h.width = 1280;
    mask.src image h.height = 800;
    mask.src_image_h.mode = G2D_GL0BAL_ALPHA;
    mask.dst_image_h.clip_rect.x = 0;
    mask.dst_image_h.clip_rect.y = 0;
    mask.dst_image_h.clip_rect.w = 1280;
13
   mask.dst_image_h.clip_rect.h = 800;
14
   mask.dst_image_h.width = 1280;
15
   mask.dst_image_h.height = 800;
   mask.dst_image_h.mode = G2D_GL0BAL_ALPHA;
16
   mask.mask_image_h.clip_rect.x = 0;
```



```
mask.mask_image_h.clip_rect.y = 0;
    mask.mask_image_h.clip_rect.w = 1280;
    mask.mask_image_h.clip_rect.h = 800;
21
    mask.mask image h.width = 1280;
22
    mask.mask_image_h.heighto= 800;
    mask.mask_image_h.mode = G2D_GL0BAL_ALPHA;
23
24
    mask.ptn image h.clip rect.x = 0;
25
    mask.ptn_image_h.clip_rect.y = 0;
26
    mask.ptn_image_h.clip_rect.w = 1280;
27
    mask.ptn image h.clip rect.h = 800;
28
    mask.ptn image h.width = 1280;
29
    mask.ptn_image_h.height = 800;
    mask.ptn_image_h.mode = G2D_GL0BAL_ALPHA;
30
    mask.src_image_h.alpha = 0xff;
31
32
    mask.mask_image_h.alpha = 0xff;
33
    mask.ptn_image_h.alpha = 0xff;
34
    mask.dst_image_h.alpha = 0xff;
35
    mask.src_image_h.format = G2D_FORMAT_ARGB8888;
36
    mask.mask_image_h.format = G2D_F0RMAT_ARGB8888;
37
    mask.ptn_image_h.format = G2D_FORMAT_ARGB8888;
38
    mask.dst_image_h.format = G2D_FORMAT_ARGB8888;
39
   if(ioctl(int fd, G2D_CMD_MASK_H ,(unsigned long)(&mask)) < 0)</pre>
40
41
    printf("[%d][%s][%s]G2D_CMD_MASK_H failure!\n, _LINE_____FILE____FUNCTION_);
42
43
              √°return -1√°
44
```

3.3 批处理接口

```
struct mixer_para {
    g2d_operation_flag op_flag;
    g2d_blt_flags_h flag_h;
    g2d_rop3_cmd_flag back_flag;
    g2d_rop3_cmd_flag fore_flag;
    g2d_bld_cmd_flag
                          bld_cmd;
    g2d_image_enh src_image_h;
    g2d_image_enh dst_image_h;
   g2d_image_enh ptn_image_h;
    g2d_image_enh mask_image_h;
    g2d_ck ck_para;
};
typedef enum {
    OP_FILLRECT = 0x1,
    OP_BITBLT = 0x2,
    OP_BLEND = 0x4,
    OP MASK = 0x8,
    OP\_SPLIT\_MEM = 0 \times 10,
} g2d_operation_flag;
```

LLW

struct mixer_para 是 RCQ 批处理的核心结构体,可以看到除了第一个成员,其它成员的类型都是旧驱动里面有的,struct mixer para 是之前驱动接口结构体的一个合集,如图 2 所示:

QU413

dutio Plutio

版权所有 © 珠海全志科技股份有限公司。保留一切权利

ieitus

SUTIS

35 sixur



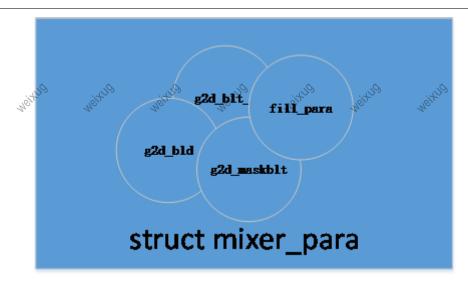


图 3-1: mixerpara

所以你可以用批处理接口完成上面其它接口的功能,只要你设置好对应的成员和g2d_operation_flag即可。

3.3.1 G2D_CMD_MIXER_TASK

• PROTOTYPE

int ioctl(int fd, int cmd, void *arg)

• ARGUMENTS

cmd: G2D_CMD_MIXER_TASK

arg[0]: 设备文件标识符arg指向mixer_para指针,批处理的话就是数组指针。

arg[1]: 指针需要处理的帧的数量,大于等于1

RETURN

【成功: 0,失败:失败号

用户要做的事情,就是填充好 mixer_para 数组,申请好输入输出内存,将要处理的图像写入到输入内存里面,将处理好的图像在输出内存里面取出来。

下面是批处理缩放 16 帧示例,其中 4 帧是 rgb 格式的缩放,6 帧是 Y8 的是缩放,6 帧是 nv12 的缩放。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

itus 3



```
2
    #define RGB_IMAGE_NAME "../../pic/c1080_good.rgb"
 3
     #define Y8_IMAGE_NAME "../../pic/en_dmabuf_bike_1280x720_220_Y8.bin"
     #define NV12_IMAGE_NAME "../../pic/bike_1280x720_220.bin"
 4
                 OUL
 5
    #define FRAME_TO_BE_PROCESS 16,000 14
 6
 7
     /*4 rgb convert 6 Y8 convert 6 yuv420 convert*/
 8
     unsigned int out_width[FRAME_TO_BE_PROCESS] = {
 9
         192, 154, 108, 321, 447, 960, 241, 320,
10
         1920, 1439, 1280, 1920, 2048, 720, 800, 480};
     unsigned int out_height[FRAME_TO_BE_PROCESS] = {108,
11
                                                              87, 70,
                                                                          217, 213, 640,
12
                                                              240, 1080, 777, 800, 1080,
                                                        840.
13
                                                        2048, 480, 480,
                                                                         240};
14
15
    struct test_info_t
16
17
              struct mixer_para info[FRAME_TO_BE_PROCESS];
18
19
    };
20
21
    Int main()
22
    {
23
24
      test_info.info[0].flag_h = G2D_BLT_NONE_H;
25
             test_info.info[0].op_flag = OP_BITBLT;
             test_info.info[0].src_image_h.format = G2D_FORMAT_RGB888;
2.6
27
             test_info.info[0].src_image_h.width = 1920;
28
             test_info.info[0].src_image_h.height = 1080;
29
              test_info.info[0].src_image_h.clip_rect.x = 0;
30
              test_info.info[0].src_image_h.clip_rect.y = 0;
31
              test_info.info[0].src_image_h.clip_rect.w = 1920;
32
              test_info.info[0].src_image_h.clip_rect.h = 1080;
              test info.info[0].src image h.color = 0xee8899;
33
34
              test_info.info[0].src_image_h.mode = G2D_PIXEL_ALPHA;
35
              test_info.info[0].src_image_h.alpha = 0xaa;
              test_info.info[0].src_image_h.align[0] = 0;
36
37
              test_info.info[0].src_image_h.align[1] = 0;
38
             test_info.info[0].src_image_h.align[2] = 0;
39
              test_info.info[0].dst_image_h.format = G2D_FORMAT_RGB888;
40
             test_info.info[0].dst_image_h.width = 800;
test_info.info[0].dst_image_h.height = 480;
41
42
43
              test_info.info[0].dst_image_h.clip_rect.x = 0;
44
             test_info.info[0].dst_image_h.clip_rect.y = 0;
45
              test_info.info[0].dst_image_h.clip_rect.w = 1920;
46
              test_info.info[0].dst_image_h.clip_rect.h = 1080;
47
              test_info.info[0].dst_image_h.color = 0xee8899;
             test_info.info[0].dst_image_h.mode = G2D_PIXEL_ALPHA;
48
49
             test_info.info[0].dst_image_h.alpha = 255;
50
             test_info.info[0].dst_image_h.align[0] = 0;
51
             test_info.info[0].dst_image_h.align[1] = 0;
52
             test_info.info[0].dst_image_h.align[2] = 0;
53
    for (i = 0; i < FRAME_TO_BE_PROCESS; ++i) {</pre>
54
                      memcpy(&test_info.info[i], &test_info.info[0],
55
                             sizeof(struct mixer_para));
56
                      test_info.info[i].dst_image_h.width = out_width[i];
57
                      test_info.info[i].dst_image_h.height = out_height[i];
58
                      test_info.info[i].dst_image_h.clip_rect.w = out_width[i];
59
                      test_info.info[i].dst_image_h.clip_rect.h = out_height[i];
60
                      if (i < 4) {
```



```
61
                              test_info.out_size[i] = test_info.info[i].dst_image_h.width *
         test_info.info[i].dst_image_h.height * 3;
                              test_info.info[i].src_image_h.format = G2D_FORMAT_BGR888;
62
                              test_info.info[i].src_image_h.width = 1920;
63
                                                                                         WEIXIG
64
                             test_info.info[i].src_image_h.height = 1080;
                              test_info.info[i].src_image_b.clip_rect.w = 1920;
65
66
                              test_info.info[i].src_image_h.clip_rect.h = 1080;
67
                              test_info.in_size[i] = 1920*1080*3;
68
                              snprintf(test_info.src_image_name[i], 100, "%s", RGB_IMAGE_NAME);
69
                      } else if (i < 10) {</pre>
                              test info.out size[i] = test info.info[i].dst image h.width *
70
         test_info.info[i].dst_image_h.height;
71
                              test_info.info[i].src_image_h.format = G2D_FORMAT_Y8;
                              test_info.info[i].src_image_h.width = 1280;
72
73
                              test_info.info[i].src_image_h.height = 720;
                              test_info.info[i].src_image_h.clip_rect.w = 1280;
74
75
                              test_info.info[i].src_image_h.clip_rect.h = 720;
76
                              test_info.in_size[i] = 1280*720;
77
                              snprintf(test_info.src_image_name[i], 100,"%s",Y8_IMAGE_NAME);
78
79
                              test_info.out_size[i] = test_info.info[i].dst_image_h.width *
         test_info.info[i].dst_image_h.height * 2;
80
                              test_info.info[i].src_image_h.format =
         G2D_F0RMAT_YUV420UVC_U1V1U0V0;
                          test_info.info[i].src_image_h.width = 1280;
81
82
                              test_info.info[i] src_image n.height = 720;
83
                              test_info.info[i].src_image_h.clip_rect.w = 1280;
                              test_info.info[i].src_image_h.clip_rect.h = 720;
84
                              test_info.in_size[i] = 1280*720*2;
85
                              snprintf(test_info.src_image_name[i], 100,"%s",NV12_IMAGE_NAME);
86
87
88
                      ret = ion_memory_request(&test_info.dst_ion[i], 1, NULL, test_info.
         out size[i]);
89
                      test info.info[i].dst image h.fd = test info.dst ion[i].fd data.fd;//rtos-
         hal中的驱动不支持使用fd,这里请修改为物理地址,并设置好偏移
90
                      test_info.info[i].dst_image_h.format = test_info.info[i].src_image_h.
91
         format:
                      ret = ion_memory_request(&test_info.src_ion[i], 0, test_info.
92
         src_image_name[i], test_info.in_size[i]);
                      test_info.info[i].src_image_h.fd = test_info.src_ion[i].fd_data.fd;//rtos-
93
         hal中的驱动不支持使用fd,这里请修改为物理地址,并设置好偏移
94
95
      arg[0] = (unsigned long)test_info.info;
              arg[1] = FRAME TO BE PROCESS;
96
              if (ioctl(g2d fd, G2D CMD MIXER TASK, (arg)) < 0) {
97
98
                      printf("[%d][%s][%s]G2D CMD MIXER TASK failure!\n", LINE ,
                               _FILE__, __FUNCTION__);
99
100
                      goto FREE_SRC;
101
              printf("[%d][%s][%s]G2D_CMD_MIXER_TASK SUCCESSFULL!\n", __LINE___,
102
103
                     __FILE__, __FUNCTION__);
104
105
106
              printf("save result data to file\n");
107
              char sufix[40] = \{0\};
108
              for (i = 0; i < FRAME_TO_BE_PROCESS; ++i) {</pre>
109
                      if (i < 4) {
                              snprintf(sufix, 40, "rgb888");
110
111
                      } else if (i < 10)
```



```
snprintf(sufix, 40, "y8");
     112
     113
                           else
                                   snprintf(sufix, 40, "nv12");
     114
     115
neixus 116
                           snprintf(test_info_dst_image_name[i], 100,
                                   "../../result/frame%d_%dx%d_to_%dx%d.%s",i,
                                    test_info.info[i].src_image_h.width,
     118
     119
                                    test_info.info[i].src_image_h.height,
     120
                                    test_info.info[i].dst_image_h.width,
     121
                                    test_info.info[i].dst_image_h.height, sufix);
     122
                           if((test_info.dst_fp[i] = fopen(test_info.dst_image_name[i], "wb+")) ==
              NULL) {
     123
                                   printf("open file %s fail.\n", test_info.dst_image_name[i]);
     124
     125
                           } else {
     126
                                   ret = fwrite(test_info.dst_ion[i].virt_addr,
     127
                                                test_info.out_size[i], 1, test_info.dst_fp[i]);
     128
                                   fflush(test_info.src_fp);
     129
                                   printf("Frame %d saved\n", i);
     130
                           }
     131
     132
                   }
     133
     134
        3.3.2 G2D_CMD_CREATE_TASK

• PROTOTYPE
```

```
int ioctl(int fd, int cmd, void *arg)
```

ARGUMENTS

```
cmd
             G2D_CMD_CREATE_TASK
arg[0]
             arg指向mixer_para指针,批处理的话就是数组指针。
arg[1]
             需要处理的帧的数量,大于等于1
```

RETURN

```
成功: task id, 大于等于1, 其它情况则为失败
arg[0]对应的指针所指向的mixer_para内容会被更新。
```

该 ioctl 命令用于创建新的批处理实例,但不做硬件处理, 只是准备好软件。

版权所有 © 珠海全志科技股份有限公司。保留一切权利



0,

这个过程会构造对应帧数的 rcq 队列内存以及进行输入输出图像的 dma map 和 dma umap 操 作,构造完毕之后会更新 mixer para 回应用层。task id 是唯一的,只要不销毁批处理实例,会 一直占据这个 id,根据这个 id 用户可以进一步操作,比如设置,销毁,获取当前 mixer para。

OU, 如下例子,会创建两个不同帧数和输入输出格式的批处理实例,最终得到两个不同的 task id, task0 和 task1。mixer para 如何构造参考 G2D CMD MIXER TASK 的例子。

```
arg[0] = (unsigned long)test_info.info;
        arg[1] = FRAME_TO_BE_PROCESS;
3
        task0 = ioctl(g2d_fd, G2D_CMD_CREATE_TASK, (arg));
        if (task0 < 1) {
 4
5
            printf("[%d][%s][%s]G2D_CMD_CREATE_TASK failure!\n", __LINE___,
 6
                     _FILE__, __FUNCTION__);
 7
            goto FREE_SRC;
 8
9
        printf("[%d][%s][%s]G2D CMD CREATE TASK SUCCESSFULL!\n", LINE ,
10
               __FILE__, __FUNCTION__);
11
12
13
        arg[0] = (unsigned long)test info2.info;
14
        arg[1] = FRAME_TO_BE_PROCESS2;
15
        task1 = ioctl(g2d_fd, G2D_CMD_CREATE_TASK, (arg));
        if (task1 < 1) {
16
            printf("[%d][%s][%s]G2D_CMD_CREATE_TASK failure!\n",
17
18
                   __FILE__, __FUNCTION__);
19
            goto FREE_SRC;
20
        printf("[%d][%s][%s]G2D_CMD_CREATE_TASK_SUCCESSFULL!\n", __LINE___,
21
                _FILE__, __FUNCTION__);
```

3.3.3 G2D CMD TASK APPLY

• PROTOTYPE

```
int ioctl(int fd, int cmd, void *arg)
```

ARGUMENTS

```
\mathsf{cmd}
               G2D CMD TASK APPLY
               task id(由G2D_CMD_CREATE_TASK命令获得)
arg[0]
               arg指向mixer_para指针,批处理的话就是数组指针
arg[1]
```

RETURN

```
成功: 0, 失败: 失败号
```



该 ioctl 命令的作用是执行批处理的硬件操作。

值得注意 arg[1] 中的 mixer_para,必须是 G2D_CMD_CREATE_TASK 之后返回的 mixer_para 或者是通过另外一个 ioctl 命令 G2D_CMD_TASK_GET_PARA 才行, 这里不需要制定帧数的原因是前面的 G2D_CMD_CREATE_TASK 已经指定好帧数,而 G2D_CMD_TASK_APPLY 是基于 task id 来执行的。

```
arg[0] = task0;
 2
        arg[1] = (unsigned long)test_info.info;
3
        if(ioctl(g2d_fd, G2D_CMD_TASK_APPLY, (arg)) < 0) {</pre>
            printf("[%d][%s][%s]G2D_CMD_TASK_APPLY failure!\n", __LINE__,
 4
 5
                      _FILE___, ___FUNCTION___);
 6
            goto FREE_SRC;
 7
        printf("[%d][%s][%s]G2D_CMD_TASK_APPLY SUCCESSFULL!\n", __LINE__,
 8
9
                __FILE__, __FUNCTION__);
10
11
        arg[0] = task1;
12
        arg[1] = (unsigned long)test info2.info;
13
        if(ioctl(g2d_fd, G2D_CMD_TASK_APPLY, (arg)) < 0) {</pre>
14
            printf("[%d][%s][%s]G2D_CMD_TASK_APPLY failure!\n", __LINE___,
                     _FILE__,__FUNCTION__);
15
            goto FREE_SRC;
16
17
18
        printf("[%d][%s][%s]G2D_CMD_TASK_APPLY SUCCESSFULL!\n",
19
                __FILE__, __FUNCTION__);
```

3.3.4 G2D CMD TASK DESTROY

PROTOTYPE

```
int ioctl(int fd, int cmd, void *arg)
```

ARGUMENTS

```
cmd G2D_CMD_TASK_DESTROY
arg[0] task id
```

RETURN

```
成功: 0,失败: 失败号
```

该 ioctl 命令的作用是销毁指定 task id 的批处理实例。

OUTIS

ieitus

itus a

41



```
arg[0] = task0;;
2
        if(ioctl(g2d_fd, G2D_CMD_TASK_DESTROY, (arg)) < 0) {</pre>
3
            printf("[%d][%s][%s]G2D_CMD_TASK_DESTROY failure!\n", __LINE___,
                     _FILE__, __FUNCTION__);
4
            goto FREE_SRC;
5
 6
        printf("[%d][%s][%s]G2D_CMD_TASK_DESTROY SUCCESSFULL!\n", __LINE_
 8
                __FILE__, __FUNCTION__);
9
        arg[0] = task1;;
10
        if(ioctl(g2d fd, G2D CMD TASK DESTROY, (arg)) < 0) {</pre>
11
            printf("[%d][%s][%s]G2D_CMD_TASK_DESTROY failure!\n", __LINE___,
                     _FILE__, __FUNCTION__);
12
            goto FREE_SRC;
13
14
15
        printf("[%d][%s][%s]G2D_CMD_TASK_DESTROY SUCCESSFULL!\n", __LINE__,
                 _FILE__, __FUNCTION__);
16
```

3.3.5 G2D_CMD_TASK_GET_PARA

• PROTOTYPE

int ioctl(int fd, int cmd, void *arg)

ARGUMENTS

```
cmd G2D_CMD_TASK_DESTROY
arg[0] task id
arg[1] 指向mixer_para指针,多帧的话就是数组指针
```

RETURN

成功: 0,失败: 失败号

该 ioctl 命令的作用是获取指定 task id 的 mixer para。

用户必须自行保证传入的指针所指向的内存足够存放这么多帧的参数

OUTIS





4.1 常见问题

4.1.1 对齐问题

- mixer 要 4byte 对齐
- rotate 输出要 8byte 对齐,输入没有要求,底层关心的只是输入的宽和高,以及输出的 pitch 大小

4.1.2 输出格式显示

yuv 格式,做旋转时,输出一律是 yuv420,旋转和缩放不能同时使用,要调用两次接口。

4.1.3 输出宽度

G2D 硬件模块不支持输出宽度等于 1 pixel。

ostus nestus nestus nestus nestus nestus nestus nestus nestus nestus nestus

QU4;



著作权声明

版权所有 © 2022 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护,其著作权由珠海全志科技股份有限公司("全志")拥有并保留 一切权利。

本文档是全志的原创作品和版权财产,未经全志书面许可,任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部,且不得以任何形式传播。

商标声明



举)均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标,产品名称,和服务名称,均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司("全志")之间签署的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所购买或使用的范围内。使用前请认真阅读合同条款和相关说明,并严格遵循本文档的使用说明。您将自行承担任何不当使用行为(包括但不限于如超压,超频,超温使用)造成的不利后果,全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因,本文档内容有可能修改,如有变更,恕不另行通知。全志尽全力在本文档中提供准确的信息,但并不确保内容完全没有错误,因使用本文档而发生损害(包括但不限于间接的、偶然的、特殊的损失)或发生侵犯第三方权利事件,全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中,可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税(专利税)。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。

版权所有 © 珠海全志科技股份有限公司。保留一切权利