

Rockchip

RGA FAQ

发布版本:1.0

日期:2018.12

免责声明

本文档按“现状”提供，福州瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自所有者所有。

版权所有 © 2018 福州瑞芯微电子股份有限公司

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

福州瑞芯微电子股份有限公司

Fuzhou Rockchip Electronics Co., Ltd.

地址：福建省福州市铜盘路软件园 A 区 18 号

网址：www.rock-chips.com

客户服务电话：+86-591-83991906

客户服务传真：+86-591-83951833

客户服务邮箱：service@rock-chips.com

前言

概述

本文档主要介绍 Rockchip RGA 模块的相关问题的处理和回答

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

修订记录

日期	版本	作者	修改说明
12.11	1.0	郑仕相	初版

目录

1 概述	1-1
2 RGA 问题相关 log 获取和说明	2-1
2.1 RGA kernel 驱动的调式节点使用说明	2-1
2.2 RGA 打印 log 说明	2-3
3 RGA_FAQ	3-5
3.1 KernelSpace 层错误处理	3-5
3.2 UserSpacec 层错误处理	3-6

1 概述

本文总结 RK 产品平台上，处理 RGA 相关的各种显示问题的步骤和方法。本文一方面作为对过往工作方法的总结，另一方面也可以作为后续调试工作的参考。

RK 芯片中使用的 RGA，分为 2 两个大版本：

- RGA1 在 RK31xx RK30xx 系列芯片中使用
- RGA2 在 RK32xx RK33xx 系列芯片中使用

RGA 相关的问题主要可以分为两种类型的问题。第一类是 UserSpace 的问题这种类型的问题主要是用户在调用使用 RGA 接口的时候存在的不会使用和错误使用，这累问题是 RGA 相关问题最多也是最主要的问题。关于这类问题的解决方法将在 RGA_FAQ 一章详细介绍。第二类是 KernelSpace 的问题，此类问题主要是驱动存在 bug 和需要优化的地方，反应出来的现象比较严重例如内核 crash 和系统卡死。这类问题需要相关工程可以收集好需要的 log 和前期分析帮助驱动维护的工程师可以快速定位问题。

2 RGA 问题相关 log 获取和说明

用户调用 RGA 驱动有两种方式一种是通过 librga 的相关 hal 层接口调底层驱动，另外一种是直接通过 IOCTL 命令调用底层驱动。第一种方式在 hal 层封装了 RGA 驱动的调用接口使得调用者可以更加方便安全的使用 RGA 工作。关于 librga 的使用请参考 librga 的相关文档。该文档详细介绍了 librga 的使用说明和相关 demo 介绍，以及调试方法。固本文对 librga 的使用不再赘述，主要介绍 kernel 驱动的调试方法。

RGA kernel 驱动的调式节点概述

新版的内核 RGA 驱动在旧版本的基础上将相关的调试 log 进行了相对友好的字符串转换，便于调式人员对 log 的分析。同时通过加入节点的方式使得调式信息的输出可控，且不需要重新编译下载代码。通过节点可以方便通过一些命令跑一些测试 case 方便调试。前文提到 RGA 有两个大版本 RGA1 和 RGA2。两者的调试节点的使用完全相同调式方法也是一样的唯一的区别在于节点的路径做了版本的区别。

Rga1 版本调式节点路径为：d/rga_debug/rga

Rga2 版本调试节点路径为：d/rga2_debug/rga2

2.1 RGA kernel 驱动的调式节点使用说明

1. RGA1 和 RGA2 的调试节点的使用方法完全相同我们以 RGA2 的节点为例子做介绍。
2. cat rga2 该命令可以让串口打印 RGA 节点的使用说明。

```
rk322x_box:/d/rga2_debug # cat rga2
echo reg    > rga2 to open rga reg MSG //开启寄存器配置打印
echo msg    > rga2 to open rga msg MSG //开启传递参数打印
echo time   > rga2 to open rga time MSG //开启耗时打印
echo check  > rga2 to open rga check flag //打开检查 case 功能
echo stop   > rga2 to stop using hardware //停用 rga 驱动
echo int    > rga2 to open interrupt MSG //打开中断信息打印
echo slt    > rga2 to open slt test // 进行内部 slt 测试
```

3. RGA 的通过节点打开的相关打印信息打印级别为 KERNEL_DEBUG。需要敲入 dmesg 才能在串口或者通过 adb 看到相关打印。

4. 打印 log 的开启于关闭命令是一样的，每次输入命令会切换当前的打印状态。可以通过相关的打印信息“open xxx” 或者“close xxx” 来确认。

```
rk322x_box:/d/rga2_debug # echo msg > rga2;dmesg -c //敲入命令
[49578.694343] rga2: open rga2 test MSG! // 打印信息提示当前为打开状态
rk322x_box:/d/rga2_debug # echo msg > rga2;dmesg -c //敲入命令
[49579.462952] rga2: close rga2 test MSG! // 打印信息提示当前为关闭状态
```

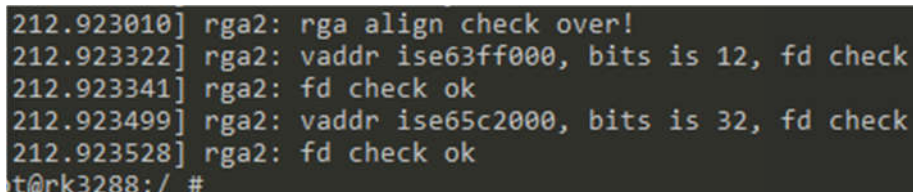
5. echo msg > rga 该命令开关 RGA 上层配置参数信息的打印。打开该打印时，上层调用 rga 驱动传递的参数将被打印出来。

6. echo time > rga 该命令开关 RGA 工作耗时信息的打印。打开该打印时，将会打印每一次的调用 rga 工作的耗时。

7. echo reg > rga 该命令开关 RGA 寄存器配置信息的打印。打开该打印时，将会打印每次 rga 工作寄存器的配置值。

8. echo int > rga 该命令开关 RGA 寄存器中断信息的打印。打开该打印时，将会在 RGA 进入中断后打印中断寄存器和状态基础器的当前值。

9. echo check > rga 该命令开关 RGA 内部的测试 case。打开该打印时，将会在 RGA 每次工作的时候检查相关的参数，主要是内存的检查，和对齐是否满足要求。若输出如下 log 表示通过检查。若内存存在越界的情况，将会导致内核 crash。可以通过 cash 之前的打印 log 确认是 src 数据的问题还是 dst 数据的问题。



```
212.923010] rga2: rga align check over!
212.923322] rga2: vaddr ise63ff000, bits is 12, fd check
212.923341] rga2: fd check ok
212.923499] rga2: vaddr ise65c2000, bits is 32, fd check
212.923528] rga2: fd check ok
t@rk3288:/ #
```

图 1

10. echo stop > rga 该命令开关 RGA 的工作状态。开启时，rga 将不工作直接返回。用于一些特殊情况下的调式。

11. echo slt > rga 该命令让 rga 驱动执行内部 SLT case 测试 rga 硬件是否正常。若输出如下 log 表示正常。

```
41.758185] rga2: ***** RGA_TEST *****
41.758237] rga2: *****
41.758237]
41.758758] rga2: [ num : srcInfo dstInfo ]
41.758958] rga2: [Y00000000 : 0x50505050 0x50505050]
41.774500] rga2: err_count=0, right_count=80000
41.774542] rga2: rga slt success !!
64k3288+ /d/rga2 debug #1
```

图 2

2.2 RGA 打印 log 说明

对于 RGA 的问题调试需要借助相关的 log 来弄清当前 RGA 执行的是什么工作,上文已经说明了相关 log 的开启。接下来将说明如何分析这些 log。

下图是开启 msg 串口输出的 log 示例。

```
[ 6539.002130] rga2: cmd is RGA_GET_VERSION
[ 6539.020400] rga2: cmd is RGA_BLIT_SYNC
[ 6539.020423] rga2: render_mode:bitblt,bitblit_mode=0,rotate_mode:0
[ 6539.020437] rga2: src : y=7 uv=0 v=e1000 aw=1280 ah=720 vw=1280 vh=720 xoff=0 yoff=0 format=RGBA8888
[ 6539.020450] rga2: dst : y=9 uv=0 v=e1000 aw=1280 ah=720 vw=1280 vh=720 xoff=0 yoff=0 format=RGBA8888
[ 6539.020460] rga2: mmu : src=01 src1=00 dst=01 els=00
[ 6539.020470] rga2: alpha : flag 9 mode0=381a mode1=381a
[ 6539.020478] rga2: blend mode is 105 src + (1-src.a)*dst
[ 6539.020486] rga2: yuv2rgb mode is BT.601-range1
[ 6539.020494] rga2: rgb2yuv mode is Bypass
```

图 3 打印 log 示例

1. rga2: cmd is RGA_GET_VERSION //获取版本信息
2. rga2: cmd is RGA_BLIT_SYNC //显示当前传入的工作模式
3. rga2: render_mode:bitblt, //调用的接口
4. blitBlit_mode=0, //合成模式 0 : A+B = B ,1: A+B+C 目前使用的是 A+B=B
5. rotate_mode:270 degree //旋转角度
6. rga2: src : y=7 uv=0 v=e1000 aw=1280 ah 720 vw=1280 wh=720 xoff=0 yoff=0 format=YCbCr420SP
数据源 y:Fd uv : 虚拟地址 v: vw*vh aw , ah 实宽实高 vw ,vh 虚宽虚高 xoff,yoff: x,y 方向偏移 format :数据格式
7. rga2: dst : y=8 uv=0 v=e1000 aw=720 ah 1280 vw=720 wh=1280 xoff=0 yoff=0 format=YCbCr420SP
目的数据:说明同上。
8. rga2: mmu : src=01 src1=00 dst=01 els=00 //mmu 使能位 1 开启 0 关闭
9. rga2: blend mode alpha //alpha 合成模式 105 ; 405
10. rga2: yuv2rgb mode ,rgb2yuv mode //yuv 色域选择位

◆ RGA 的两种工作模式说明：

RGA 驱动对外提供同步和异步接口既 SYNC 和 ASYNC。如图一所示，当配置同步模式时，驱动只有在完成本次工作或者出现异常情况才会返回。如图二所示，当配置异步模式时，驱动会将配置的工作任务加入

Copyright © 2018 Fuzhou Rockchip Electronics Co., Ltd.

到工作队列，开启硬件工作模式不用等待工作完成直接返回。当应用层需要使用 RGA 处理过的 buffer 数据时再配入 FLUSH 命令用于查询硬件工作完成情况。

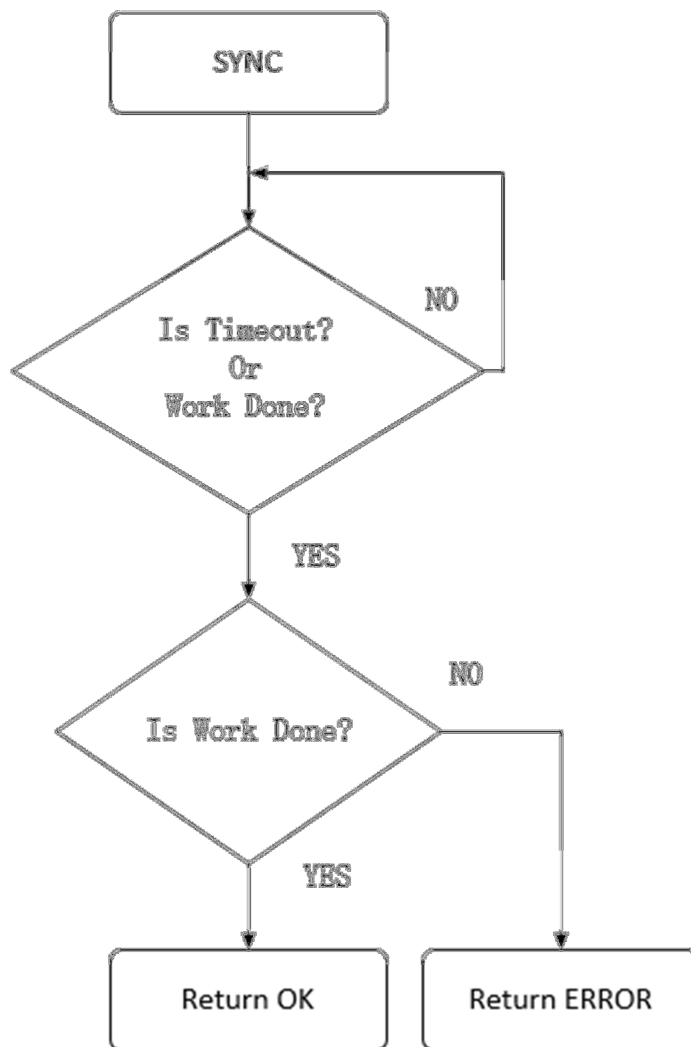


图 4 RGA SYNC 模式

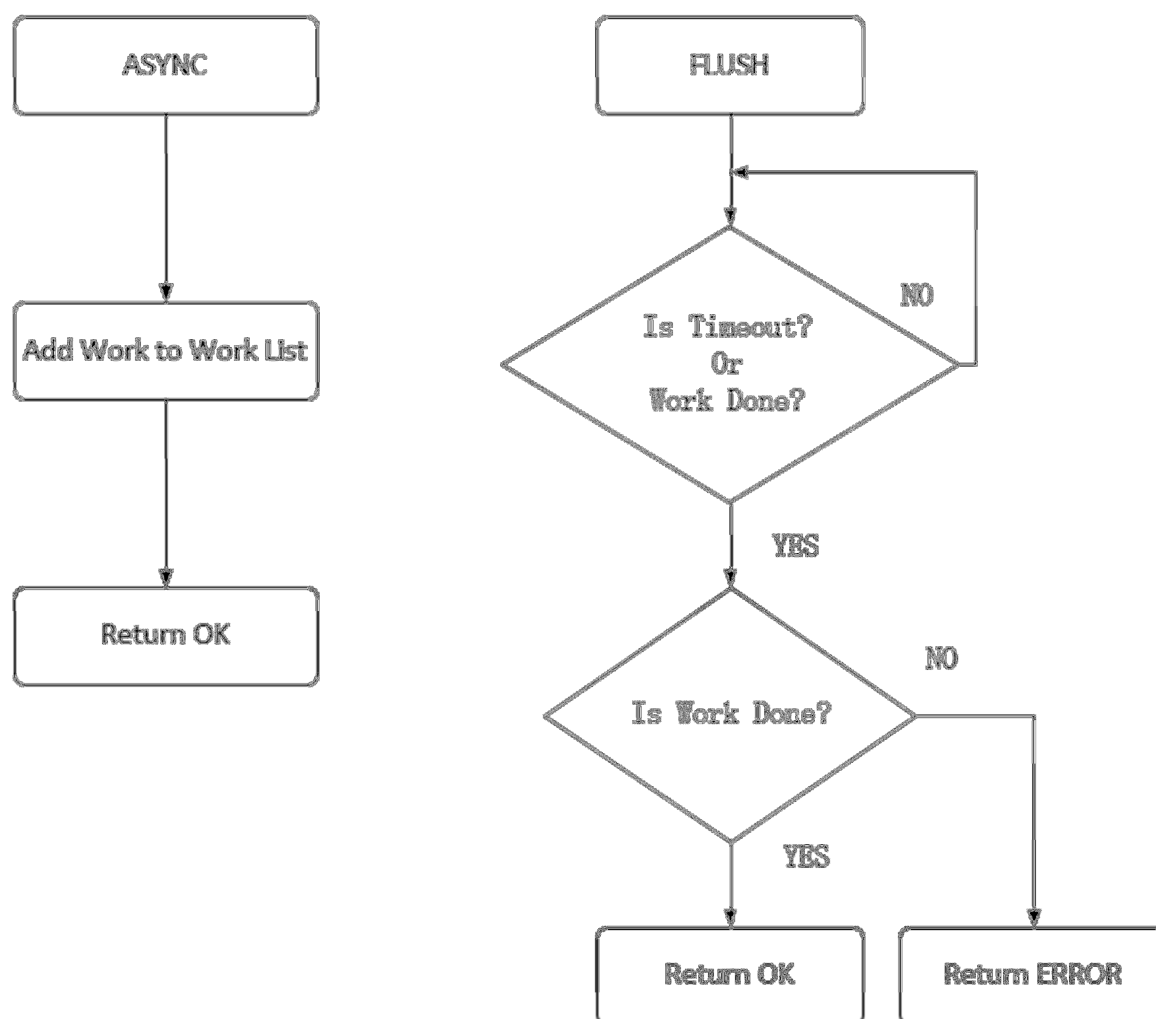


图 5 RGA ASYNC 模式

3 RGA_FAQ

本节将出现的 RGA 相关问题进行分类介绍常用解决办法,如还是不能解决请整理相关 log 和初步分析的信息交由驱动维护工程师处理。

3.1 KernelSpace 层错误处理

当出现系统卡死在 RGA,调用 RGA 工作,输出数据出现类似雪花或者绿条的情况以及内核重启堆栈打在 RGA_FLUSH 上的错误。请先和客户确认版本,更新最新的 RGA 驱动。若问题还存在。确认复现场景,echo msg > rga2; echo time> rga2 ;echo reg > rga2 ;打开 log 打印复现问题。保留 log 交给维护 RGA 驱动的工程处理。

- ◆ 若出现疑似内存越界的问题。比如 RGA 驱动报如下错误

```
265.060398] rga: Rga err irq! INT[701],STATS[1]
root@rk2288:/ #
```

按照如下步骤排查是否内存越界问题：

1. cd d/rga2_deubg
2. echo check > rga2
3. 运行客户程序
4. 若系统 crash 保留系统 crash log 根据 log 可以定位是源数据还是目的数据内存越界

```
rk2288@rk2288:~$ cd d/rga2_deubg
rk2288@rk2288:~/d/rga2_deubg$ echo check > rga2
rk2288@rk2288:~/d/rga2_deubg$ ./rk-rga2-deubg
<7>[ 1349.219274] rga2: rga align check over!
<7>[ 1349.219367] rga2: vaddr ise63ff000, bits is 32, fd check
<7>[ 1349.219401] rga2: fd check ok
<4>[ 1349.219433] func[rga2_convert_dma_buf] line [1236] rk-debug_vaddr is 1152000
<7>[ 1349.219745] rga2: vaddr ise6784000, bits is 32, fd check
<1>[ 1349.219767] Unable to handle kernel paging request at virtual address e6b07000
<1>[ 1349.225689] [e6b07000] *pgd=1a9a5811, *pte=00000000, *ppte=00000000
<0>[ 1349.234242] Internal error: Oops: 7 [#1] PREEMPT SMP ARM
<4>[ 1349.237031] Modules linked in:
<4>[ 1349.241579] CPU: 3 PID: 1681 Comm: rgaBlit Not tainted 3.10.104 #163
<4>[ 1349.246243] task: daad9f80 ti: da1aa000 task.ti: da1aa000
<4>[ 1349.251535] PC is at memcpy+0x48/0x330
<4>[ 1349.257671] pc : [<c02a48c8>] lr : [<00000000>] psr: 200d0013
<4>[ 1349.257671] sp : da1abaac ip : 00000000 fp : 00000000
<4>[ 1349.274043] r7 : 00000000 r6 : 00000000 r5 : 00000000 r4 : 00000000
<4>[ 1349.286837] Flags: nzCv IRQs on FIQs on Mode SVC_32 ISA ARM Segment user
<4>[ 1349.299456]
<4>[ 1349.299456] PC: 0xc02a4848:
<4>[ 1349.315525] 4868 e4d03001 e1330001 1affffff e2400001 13a00000 e1a0f00e e92d4011 e2522004
```

图 6

如上图所示当第二次 check 之后出现了 kernel 报错所以是目的数据存在内存越界的问题。同理若是第一次 check 的时候出现 kernel 报错则是源数据存在问题。

3. 2UserSpacec 层错误处理

1. 询问 RGA 驱动的使用方法和 demo ？

提供文档 Rockchip-Rga-Hal-1v2.pdf。Librga 的 demo。7.0 和以上 SDK 已经集成 librga 的 demo。路径：hardware/rockchip/librga。如果是 Android 4.4 版本的 SDK,demo 的路径有所不同：/hardware/rk29/librga Android7.0 版本及以下的 librga 都是一个版本。可以使用内部最新的更新。Android 8.0 及以上 SDK 已经集成。

2. 询问关于 RGA 对齐相关问题？

RGA 处理 YUV 数据对于数据有对齐要求。如果是 10bit 的 YUV 数据需要虚宽 16 对齐，虚高实高实宽 x 偏移 y 偏移 2 对齐。如果是普通的 YUV 数据需要虚宽 8 对齐，虚高实高实宽 x 偏移 y 偏移 2 对齐。

3. 在 HWC 中调用 RGA 报 RGA 的错误？

建议先更新 hwc，在测试是否还有相关问题。若还是有相关报错请整理相关 log 交由 hwc 维护的工程师

处理。

4. RGA 支持哪些输入输出格式？

RGA 支持输入格式：ARGB8888/888/565/4444/5551, YUV420/YUV422
RGA 支持的输出格式：ARGB8888/888/565/4444/5551, (YUV420/YUV422 (8/10bit)
YUYV 420/422) 其中 RGA1 非 PLUS 版不支持支持 yuv 格式输出主要是 RK3066 和 RK3188
其中 RGA2Enhance 版本比如芯片 RK3399 RV1108 支持 (YUV420/YUV422 (8/10bit)
YUYV 420/422) RGA2_lite 中的 RK3228 RK3228H 支持 YUV420/YUV422 (8/10bit)

5. RGA 能否一次绘制多个矩形区域？RGA 的工作原理？

RGA 在硬件上只能顺序工作即配置的一个任务工作结束和进行下一个配置的工作。因此不能一次绘制多个矩形区域，可以通过 ASYNC 把需要 RGA 做的工作往底层驱动配置，RGA 会将工作存储在驱动自己管理的一个工作队列中按顺序完成。当上层需要处理这块 buffer 时再配置一个 FLUSH 命令来确定 RGA 硬件是否已经完成工作。

6. RGA 工作效率问题？

当给 RGA 的地址是物理地址时且 DDR 带宽足够的前提下 RGA 可以达到理论的处理耗时。
RGA1：理论上每个时钟能执行 1 个像素点，则 300m 的 aclk 每秒能处理 300*1000,000 个像素点：
RGA2：理论上每个时钟能执行 2 个像素点，则 300m 的 aclk 每秒能处理 300*1000,000*2 个像素点：

那么 1920x1080 的像素点需要这么长的理论时间：

1. Rga1: $2073600 / (300 * 1000000) = 0.006912s$

2. Rga2: $2073600 / (300 * 1000000 * 2) = 0.003456s$

注：以上耗时的计算公式只支持数据的 copy。例：1080P 数据进行 copy 操作 RGA2 理论耗时如上文计算所示。RGA 耗时计算的像素点数是源和目的数据最大的一个。若 1080p 缩放到 720p 耗时按照 1080p 的像素点数计算，由于存在缩放，耗时比 1080p 的 copy 耗时更长。下表是 RK3399 使用 librga demo。在 DDR800M RGA aclk=400M 测试的实验数据。

数据 RGBA8888；DDR=800M;ACLK=400M；	耗时 (ms)
3840x2160->3840x2160	11.3
3840x2160->1920x1080	11.4
1920x1080->3840x2160	13.3

7. RGA 的 color_fill 功能对 YUV 格式只有 Y 分量有效？

1) RGA 不支持 YUV420SP 的 color fill 操作 2)如果只是清 0，可以把 YUV420SP 占用的 memory size 用 ARGB 折算一下，然后 dst 用 ARGB 的格式进行清 0 操作。

8. Yuv 格式和 Rgb 数据相互转换输出图像有色差？

这是因为 YUV 的色域选择有差异。
Yuv2Rgb 时 librga 默认选择的是 BT.709-range0
Rgb2Yuv 时 librga 默认选择的是 BT.601-range1。

Yuv2Rgb	BT.601-range1	BT.709-range0
	yuvToRgbMode = 0x0 << 0	yuvToRgbMode = 0x1 << 0
Rgb2Yuv	BT.601-range1	BT.709-range0
	yuvToRgbMode = 0x2 << 4	yuvToRgbMode = 0x3 << 4

若直接通过 ioctl 往驱动配置参数可以按照上面的表格配置。
若是通过调用 librga。需要修改 librga。Normal/NormalRga.cpp 修改方式相同。

9. 数据做 alpha 合成为什么没有效果？

通过 log 查看用户输入的 src 和 dst 是什么格式的数据如果是 YUV 格式的数据确需要使用全局 alpha；如果客户的 scr 和 dst 的数据格式为 RGBA 让客户确认是否 src 和 dst 的数据 alpha 未是否为 ff。若为 ff 没有合成效果是正常的，让客户使用全局 alpha 的合成方式合成。若不是则交由 RGA 驱动维护工程师处理。

10. 做图形旋转图形被拉伸显示异常？

通过让客户打开调式 log 观察 log 当用户做旋转操作旋转方向为 90 度或者 270 度时是 图像旋转后需要交换宽高，才能保证图像不变形。如下图 log 所示 dst 的 aw =src 的 ah dst 的 ah = src 的 aw。vw,vh 类似。

```
>[11821.118679] rga2: yuv2rgb mode is 1
>[11821.118689] rga2: *** rga2_blit_sync proc ***
2>[11872.597273] healthd: battery l=50 v=3 t=2.6 h=2 st=3 chg=au
>[11904.617956] rga2: cmd is RGA_GET_VERSION
>[11904.630796] rga2: cmd is RGA_BLIT_SYNC
>[11904.630822] rga2: render_mode:bitblt,bitblit_mode=0,rotate_mode:270 degree
>[11904.630841] rga2: src : y=7 uv=0 v=e1000 aw=1280 ah=720 vw=1280 vh=720 xoff=0 yoff=0 format=YCbCr420SP
>[11904.630859] rga2: dst : y=8 uv=0 v=e1000 aw=720 ah=1280 vw=720 vh=1280 xoff=0 yoff=0 format=YCbCr420SP
>[11904.630872] rga2: mmu : src=01 src1=00 dst=01 els=00
>[11904.630884] rga2: alpha : flag 0 mode0=0 mode1=0
>[11904.630895] rga2: blend mode is no blend
>[11904.630905] rga2: yuv2rgb mode is 1
>[11904.630915] rga2: *** rga2_blit_sync proc ***

/Users/Administrator/Downloads
```

图 7

11 .关于缩放倍数的支持问题？

RGA1 即 rk31xx 系列 rga 支持缩放倍数为 $1/8 \sim 8$

RGA2 LITE 3368 3328 支持缩放倍数为 $1/8 \sim 8$

其他 RGA2 的版本支持缩放 $1/16 \sim 16$