

密级状态：绝密() 秘密() 内部() 公开(√)

Rockchip_Dr. G_User_Guide

(技术部，图形显示平台中心)

文件状态：	当前版本：	V1.4
<input type="checkbox"/> 正在修改	作 者：	黄德胜、林志雄
<input checked="" type="checkbox"/> 正式发布	完成日期：	2019-07-18
	审 核：	熊伟
	完成日期：	2019-07-20

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Semiconductor Co., Ltd

(版本所有, 翻版必究)

更新记录

版本	修改人	修改日期	修改说明	核定人
V1.4	林志雄	2019-07-18	初始版本	熊伟

目 录

1 主要功能说明	4
2 DR. G 平台依赖	5
2.1 DR. G 适用平台	5
2.2 ANDROID 平台依赖	5
2.3 DR. G 固件说明	6
3 DR. G 使用说明	7
3.1 系统信息收集与 LOG 抓取	7
3.1.1 命令说明	7
3.1.2 脚本说明	7
3.2 系统频率查询和设置	8
3.2.1 设置最大性能模式	8
3.2.2 获取当前频率信息	9
3.2.3 设置 gpu 频率档	9
3.2.4 设置 cpu 频率档	10
3.2.5 设置 ddr 频率档	11
3.2.6 重置为调频状态	12
3.3 总线优先级查询和设置	13
3.3.1 查询总线优先级	13
3.3.2 设置总线优先级	13
3.4 RGA 自动测试	14
3.4.1 RGA 信息打印	14
3.4.2 RGA 自检测测试	15
3.4.3 RGA 效率测试	15
3.4.4 RGA demo 测试	16

3.5 GPU 自动测试.....	18
3.5.1 GPU 最大负载测试.....	18
3.5.2 GPU 最高帧率测试.....	18
3.5.3 GPU 性能测试.....	19
3.5.4 GLES 版本支持测试.....	19
3.5.5 OpenCL 支持测试.....	20
3.5.6 Vulkan 支持测试.....	20
3.6 VOP 自动测试.....	21
3.6.1 VOP 最大 ddr 负载测试.....	21
3.6.2 VOP demo 测试.....	22
3.7 LOG 自动检测功能.....	23
3.7.1 当前 log 分析.....	23
3.7.2 XML 规则文件.....	23

1 主要功能说明

本工具为自动诊断工具，能够很好地帮助使用者 debug 显示相关的问题。

通过日常的归纳与总结，找到工作中效率低的方面，并通过脚本，应用程序等方式解决，最终整合成为 Dr. G 自动诊断工具，工具主要分为：

- 1) 系统信息收集与 log 抓取
- 2) 系统频率查询和设置
- 3) 总线优先级查询和设置
- 4) RGA 自动测试
- 5) GPU 自动测试
- 6) VOP 自动测试
- 7) LOG 自动检测

此工具不仅限于开发人员使用，任何人在阅读完本文档后，也可以自行 debug 解决部分问题，此工具对收集 log 与系统状态信息也有极大的帮助，能够大大减少解决问题前，客户与开发人员的交互工作与代码同步工作，提升工作效率，节约大家时间。

2 Dr. G 平台依赖

2.1 Dr. G 适用平台

本工具主要支持 kernel 4.4 的所有平台，主要有：

- 1) RK3368(7.1 8.1 9.0)
- 2) RK3288(7.1 8.1 9.0)
- 3) RK3399(7.1 8.1 9.0)
- 4) RK3328 (8.1 9.0)
- 5) RK3126C(8.1 9.0)
- 6) RK3326 (8.1 9.0)

可以通过如下命令查到：

```
"dr-g -h"
```

```
rk3399_all:/ # dr-g -h
Version: v1.3
Adapter type:RK3326(8.1 9.0),RK3288(7.1 8.1 9.0),RK3399(7.1 8.1 9.0)
              RK3328(8.1 9.0),RK3126C(8.1 9.0), RK3368(7.1 8.1 9.0)
Usage:
  -h/help Displays this help message
  -v/version Show the version of the dr-g
```

2.2 Android 平台依赖

考虑到多平台的兼容性，因此将 Dr. G 编成 32 位的 binary 来使用。应用建议放置在 /system/bin

下，使用时如果发现找不到依赖库，如：librga.so not found

```
1|generic:/ #
1|generic:/ # dt-g
CANNOT LINK EXECUTABLE "dt-g": library "librga.so" not found
1|generic:/ #
```

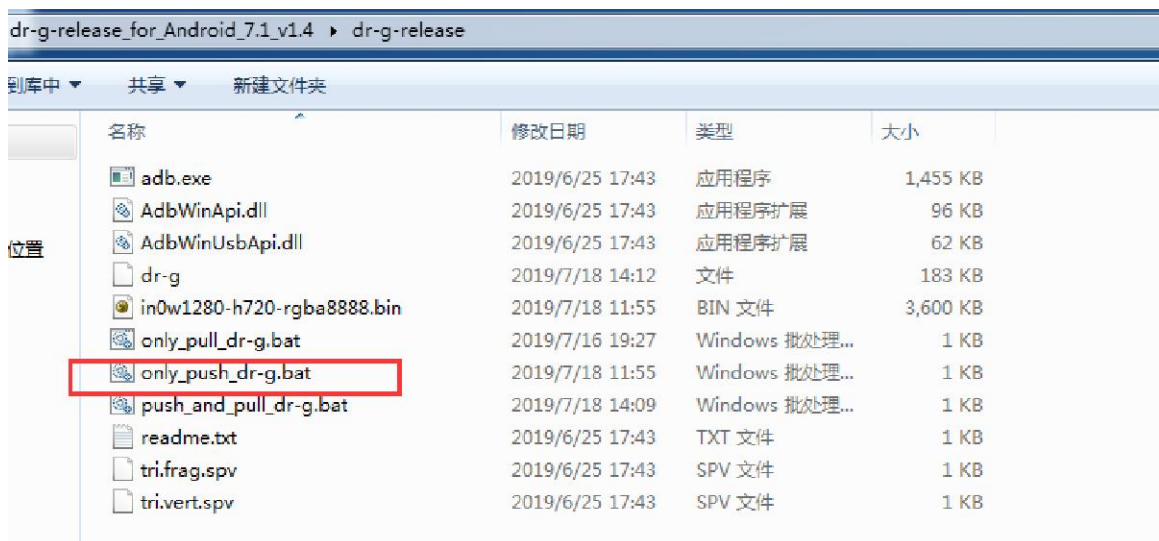
可将 /vendor/lib/ 下的对应 .so 库 拷贝至 /system/lib/ 下

2.3 Dr.G 可执行程序压缩包说明

各个平台的 Dr.G 可执行程序有 3 个压缩包，分别对应 android 平台如下表：

Dr.G 压缩包名称	适用 android 版本
dr-g-release_for_Android_7.1_v1.4.tar.gz	andorid 7.1
dr-g-release_for_Android_8.1_v1.4.tar.gz	andorid 8.1
dr-g-release_for_Android_9.0_v1.4.tar.gz	andorid 9.0

使用者只需解压对应的压缩包，保证 adb 连接成功且能正常使用 adb root、adb remount、adb shell 等基本指令，运行 only_push_dr-g.bat 脚本就可以把 Dr.G 可执行程序安装到目标板子。然后在目标板子，就可以使用 Dr.G 进行排查问题。



3 Dr. G 使用说明

3.1 系统信息收集与 log 抓取

3.1.1 命令说明

```
"dr-g -dump-info"
```

该命令运行后，会从系统中获取平台与各组件版本信息，主要有：manifest.xml、commit_id.xml、build.prop、anr_trace 文件、logcat、dmesg、io 优先级配置、CPU+GPU+DDR 当前频率与电压、CPU+GPU+DDR 最大频率、vop_clk_summary 信息、当前的 fence 状态信息、dumpsys SurfaceFlinger、ps 当前进程。

运行结果： 会在/data/dr-g-file/下生成 dumpInfo.tar

```
rk3328_mid:/ # dr-g -dump-info
DUMP-INFO ..

Dump_times: 1

The num_platform is :10 rk3328 ANDROID8

Please wait about 1 minute...


removing leading '/' from member names
-dump-info END !!! get infomation from system to: /data/dr-g-file/dumpInfo/./dumpInfo.tar
rk3328_mid:/ #
```

3.1.2 脚本说明

```
"push_and_pull_dr-g.bat"
```

当需要收集信息时，只需双击一个脚本，就可以轻松收集信息，并在当前路径生成一个对应时间的文件夹存放 dumpInfo.tar。

运行结果：

rksdk ▶ 3288 ▶ 8_1 ▶ frameworks ▶ native ▶ cmds ▶ diagnose ▶ dr-g-release ▶ 16-04-17				
名称	修改日期	类型	大小	
 dumpInfo.tar	2019/6/25 16:04	360压缩	481 KB	

如果并不需要获取 dumpInfo.tar，想要将 dr-g 推至设备上，使用其他功能，只需双击：

```
"only_push_dr-g.bat"
```

同理，单纯将 dumpInfo.tar 从设备中 pull 出来，可以双击：

```
"only_pull_dr-g.bat"
```

3.2 系统频率查询和设置

3.2.1 设置最大性能模式

```
"dr-g -sys-set perf"
```

设置性能模式，该命令执行后，CPU，GPU，DDR 设置最高频率，温控关闭。回显打印设置后的各频率。可用于调试 APP 性能问题。

运行结果：

```
rk3399_mid:/ # dr-g -sys-set perf
CPU0    freq=1.42 Ghz
CPU4    freq=1.80 Ghz
GPU      freq=800 Mhz
DDR      freq=800 Mhz
GPU utilisation 0~100%:
0
Temperature control :
CPU0 :
user_space
CPU4 :
user_space
rk3399_mid:/ #
```

3.2.2 获取当前频率信息

```
"dr-g -sys-set info"
```

获取当前状态，该命令执行后，回显打印当前 CPU，GPU，DDR 当前频率，温控状态。可用于调试 APP 性能问题。

运行结果：

```
rk3399_mid:/ # dr-g -sys-set info
CPU0    freq=1.42 Ghz
CPU4    freq=1.80 Ghz
GPU     freq=800 Mhz
DDR     freq=800 Mhz
GPU utilisation 0~100%:
0
Temperature control :
CPU0 :
user_space
CPU4 :
user_space
```

3.2.3 设置 gpu 频率档

```
"dr-g -sys-set gpu low"
```

```
"dr-g -sys-set gpu mid"
```

```
"dr-g -sys-set gpu high"
```

设置 GPU 三档频率，常用于 GPU 瓶颈的应用调试。上述 3 条命令运行后，CPU，DDR 设置最高频，分别设置 GPU 频率在频率表里面低档，中档，高档。用于调试 APP 在 3 种 GPU 频率下的表现。

运行结果：

```
rk3399_mid:/ # dr-g -sys-set gpu low
CPU0    freq=1.42 Ghz
CPU4    freq=1.80 Ghz
GPU     freq=200 Mhz
DDR     freq=800 Mhz
GPU utilisation 0~100%:
0
Temperature control :
CPU0 :
user_space
CPU4 :
user_space
```

```
rk3399_mid:/ # dr-g -sys-set gpu mid
CPU0    freq=1.42 Ghz
CPU4    freq=1.80 Ghz
GPU     freq=400 Mhz
DDR     freq=800 Mhz
GPU utilisation 0~100%:
0
Temperature control :
CPU0 :
user_space
CPU4 :
user_space

rk3399_mid:/ # dr-g -sys-set gpu high
CPU0    freq=1.42 Ghz
CPU4    freq=1.80 Ghz
GPU     freq=800 Mhz
DDR     freq=800 Mhz
GPU utilisation 0~100%:
0
Temperature control :
CPU0 :
user_space
CPU4 :
user_space
```

3.2.4 设置 cpu 频率档

```
"dr-g -sys-set cpu low"
```

```
"dr-g -sys-set cpu mid"
```

```
"dr-g -sys-set cpu high"
```

设置 CPU 三档频率，常用于 CPU 瓶颈的应用调试。上述 3 条命令运行后，GPU，DDR 设置最高频，分别设置 CPU 频率在频率表里面低档，中档，高档。用于调试 APP 在 3 种 CPU 频率下的表现。

运行结果：

```
rk3399_mid:/ # dr-g -sys-set cpu low
CPU0      freq=0.60 Ghz
CPU4      freq=0.60 Ghz
GPU       freq=800 Mhz
DDR       freq=800 Mhz
GPU utilisation 0~100%:
0
Temperature control :
CPU0 :
user_space
CPU4 :
user_space
```

```
rk3399_mid:/ # dr-g -sys-set cpu mid
CPU0      freq=1.01 Ghz
CPU4      freq=1.01 Ghz
GPU       freq=800 Mhz
DDR       freq=800 Mhz
GPU utilisation 0~100%:
0
Temperature control :
CPU0 :
user_space
CPU4 :
user space
```

```
rk3399_mid:/ # dr-g -sys-set cpu high
CPU0      freq=1.42 Ghz
CPU4      freq=1.80 Ghz
GPU       freq=800 Mhz
DDR       freq=800 Mhz
GPU utilisation 0~100%:
0
Temperature control :
CPU0 :
user_space
CPU4 :
user_space
```

3.2.5 设置 ddr 频率档

```
"dr-g -sys-set ddr low"
```

```
"dr-g -sys-set ddr mid"
```

```
"dr-g -sys-set ddr high"
```

设置 DDR 三档频率，常用于 DDR 瓶颈的应用调试。上述 3 条命令运行后，CPU，GPU 设置最高频，分别设置 DDR 频率在频率表里面低档，中档，高档。用于调试 APP 在 3 种 DDR 频率下的表现。

运行结果：

```
rk3399_mid:/ # dr-g -sys-set ddr low
CPU0    freq=1.42 Ghz
CPU4    freq=1.80 Ghz
GPU      freq=800 Mhz
DDR      freq=200 Mhz
GPU utilisation 0~100%:
0
Temperature control :
CPU0 :
user_space
CPU4 :
user_space

rk3399_mid:/ # dr-g -sys-set ddr mid
CPU0    freq=1.42 Ghz
CPU4    freq=1.80 Ghz
GPU      freq=800 Mhz
DDR      freq=400 Mhz
GPU utilisation 0~100%:
0
Temperature control :
CPU0 :
user_space
CPU4 :
user_space

rk3399_mid:/ # dr-g -sys-set ddr high
CPU0    freq=1.42 Ghz
CPU4    freq=1.80 Ghz
GPU      freq=800 Mhz
DDR      freq=800 Mhz
GPU utilisation 0~100%:
0
Temperature control :
CPU0 :
user_space
CPU4 :
user_space
```

3.2.6 重置为调频状态

```
"dr-g -sys-set reset"
```

该命令运行后，CPU，DDR，GPU 恢复系统调频模式，温控恢复。

运行结果：

```
rk3399_mid:/ # dr-g -sys-set reset
CPU0      freq=1.42 Ghz
CPU4      freq=0.82 Ghz
GPU        freq=200 Mhz
DDR        freq=800 Mhz
GPU utilisation 0~100%:
0
Temperature control :
CPU0 :
power_allocator
CPU4 :
power_allocator
```

3.3 总线优先级查询和设置

3.3.1 查询总线优先级

```
"dr-g -io-qos get"
```

获取总线优先级，并打印出对应的 name 表格，带有编号，后面可根据此编号进行 set。

运行结果：

```
rk3399_all:/ # dr-g -io-qos get
io_qos get...
set cmds: dr-g -io-qos set [io_num] [pri_num]
demo: dr-g -io-qos set 4 1
demo: dr-g -io-qos set 0 0
[io_num]:      NAME:      io_mem_addr: pri_num
[0]:           gpu_qos:   0xffae0008:   2
[1]:           vop-big_r: 0xffac8008:   3
[2]:           vop-big_w: 0xffac8008:   3
[3]:           vop-little: 0xffad0008:   3
[4]:           rga_r:     0xffab0008:   1
[5]:           rga_w:     0xffab0008:   1
[6]:           video_m0:  0xffab8008:   2
[7]:           cci_m0:    0xffa50008:   1
[8]:           cci_m1:    0xffad8008:   1
rk3399_all:/ #
```

3.3.2 设置总线优先级

```
"dr-g -io-qos set"
```

设置总线优先级，按照提示内容，查找表格对应项设置，pri_num 可选值为 0~3，设置成功后会提示：“io_qos set SUCCESS !!!”。

命令 demo:

```
dr-g -io-qos set 4 1 //设置 get 提示中序号为[4]的优先级为 1 。
dr-g -io-qos set 1 2 //设置 get 提示中序号为[1]的优先级为 2 。
dr-g -io-qos set 0 1 //设置 get 提示中序号为[0]的优先级为 0 。
```

运行结果:

```
rk3399_all:/ # dr-g -io-qos set 0 3
io_qos set... io_num:0 pri_num:3
io_qos set SUCCESS !!!
rk3399_all:/ # dr-g -io-qos get
io_qos get...
set cmds: dr-g -io-qos set [io_num] [pri_num]
demo: dr-g -io-qos set 4 1
demo: dr-g -io-qos set 0 0
[io_num]:      NAME:      io_mem_addr: pri_num
[0]:          gpu_qos:    0xffae0008: 3
[1]:          vop-big_r:  0xffac8008: 3
[2]:          vop-big_w:  0xffac8008: 3
[3]:          vop-little: 0xffad0008: 3
[4]:          rga_r:      0xffab0008: 1
[5]:          rga_w:      0xffab0008: 1
[6]:          video_m0:   0xffab8008: 2
[7]:          cci_m0:     0xffa50008: 1
[8]:          cci_m1:     0xffad8008: 1
rk3399_all:/ #
```

3.4 RGA 自动测试

3.4.1 RGA 信息打印

"dr-g -rga info"

打印 RGA 版本，硬件支持的数据格式，缩放倍数。最大输入输出分辨率信息。

运行结果:

```
rk3399_mid:/ # dr-g -rga info
RGA test ..
librga:RGA_GET_VERSION:3.02,3.020000
ctx=0xeac186e0,ctx->rgaFd=3
version is 3.02,rgav is 5
RGA vesion RGA2_Enhance
input support format : RGBA_8888 RGBA_4444 RGBA_5551 RGB_565 RGB_888 YUV420SP/P YUV422SP/P YUV420/YUV422 10bitYUV
output support format : RGBA_8888 RGBA_4444 RGBA_5551 RGB_565 RGB_888 YUV420SP/P YUV422SP/P YUYV420/422 UYVY420/422
Max input 8192x8192 Max output 4096x4096
Scale limit 1/16 ~ 16
```


3.4.2 RGA 自检测测试

```
"dr-g -rga test"
```

根据 RGA 支持的格式做格式转换，缩放，旋转，合成的测试。所有转换的结果生成 crc 数据和预先保存的 crc 数据做比较。如果机器 RGA 本身有问题，那么生成结果与预先 crc 数据不一致，那么认为该系统 RGA 本身有问题。

运行结果：

```
rk3399_mid:/ # dr-g -rga test
RGA test ..
librga:RGA_GET_VERSION:3.02,3.020000
ctx=0xe92186e0,ctx->rgaFd=3
version is 3.02,rgav is 5
format conver ok
rotate test ok
alpha test ok
scale test ok
```

3.4.3 RGA 效率测试

```
"dr-g -rga perf"
```

RGA 效率测试。根据 获取到的 ddr 频率以及 rga 的 aclk 计算出 720p 1080p 4k 数据理论下的耗时与当前测试耗时做比较，从而得出当前运行的 RGA 性能是否符合预期。

运行结果：

```
rk3399_mid:/ # dr-g -rga perf
RGA test ..
librga:RGA_GET_VERSION:3.02,3.020000
ctx=0xe77986e0,ctx->rgaFd=3
version is 3.02,rgav is 5
CPU0    freq=1.42 Ghz
CPU4    freq=1.80 Ghz
GPU      freq=800 Mhz
DDR      freq=800 Mhz
GPU utilisation 0~100%:
0
Temperature control :
CPU0 :
user_space
CPU4 :
user_space
rgba8888 1280x720 -> rgba8888 1280x720 test ok cost time 1.34
rgba8888 1920x1080 -> rgba8888 1920x1080 test ok cost time 3.03
rgba8888 3840x2160 -> rgba8888 3840x2160 test ok cost time 11.72
```


3.4.4 RGA demo 测试

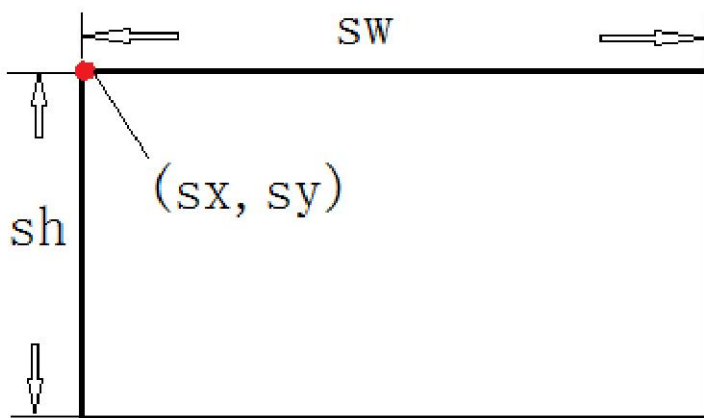
`"dr-g -rga demo"`

通过命令参数让 rga 执行对应的功能。输出的源数据和目标数据放在 data 目录下。

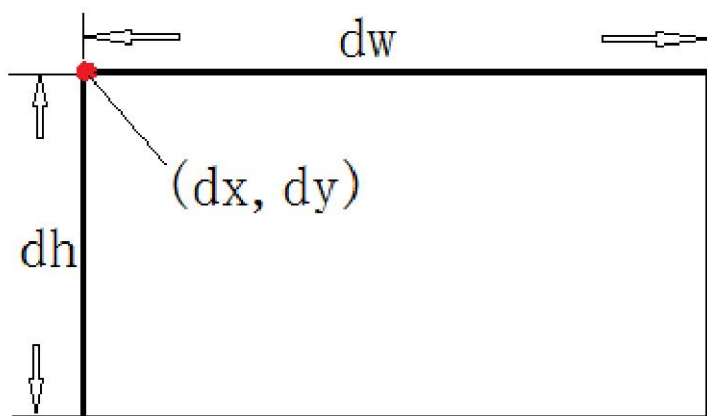
命令格式: `dr-g -rga demo sw1280sh720sf0dw1280dh720df0rt2sx0sy0dx0dy0`

注: `-rga demo` 后的参数串需要连续不能有空格

源数据:



目标数据:



参数说明:

alphah 模式	blit 类型	混合公式
bd1	105	$src + dst * (1 - \alpha)$
bd2	405	$src * \alpha + dst * (1 - \alpha)$

旋转参数	rt1	rt2	rt3	rt4	rt5
对应旋转方向	rotate 90	rotate180	rotate270	mirrorH	mirrorV

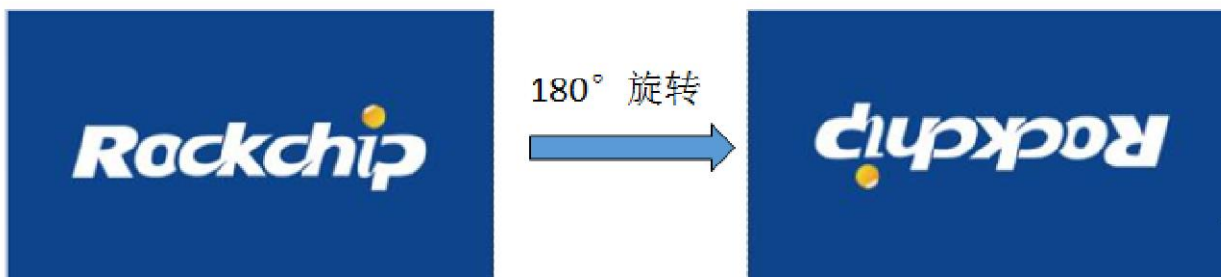
格式参数	sf0	sf1	sf2	sf3	sf4	sf5
对应格式	RGBA8888	RGBX8888	BGRA8888	RGB888	RGB565	NV12

格式参数	sf6	sf7	sf8	sf9	sf10	sf11	sf12
对应格式	NV21	YV12	YV21	NV16	NV61	YV16	YV61

运行结果:

源文件 /data/in0w1280-h720-rgba8888.bin, 对该图像进行 180 旋转, 把旋转后的数据写到 /data/out0w1280-h720-rgba8888.bin 。可以把此 raw 数据拉出来, 用工具查看。

```
rk3399_mid:/data # dr-g -rga demo sw1280sh720sf0dw1280dh720df0rt2sx0sy0dx0dy0
RGA test ..
librga:RGA_GET_VERSION:3.02,3.020000
ctx=0xf3a986e0,ctx->rgaFd=3
version is 3.02,rgav is 5
src:1280x720 rgba8888 dst:1280x720 rgba8888 cost time2.06
open /data/out0w1280-h720-rgba8888.bin and write ok
```



3.5 GPU 自动测试

3.5.1 GPU 最大负载测试

```
"dr-g -gpu max-load"
```

最大负载测试，同时可以给定 run 的时间，后面直接加时间（单位秒）。如 `dr-g -gpu max-load 100`，最大负载运行 100 秒。该测试运行后，GPU 负载接近 100%。可用于老化测试，增加系统负载等测试。

运行结果：

```
rk3399_mid:/ # cat /sys/devices/platform/ff9a0000.gpu/utilisation
100
rk3399_mid:/ # cat /sys/devices/platform/ff9a0000.gpu/utilisation
100
rk3399_mid:/ # cat /sys/devices/platform/ff9a0000.gpu/utilisation
100
```

```
"dr-g -gpu max-load-bg"
```

后台运行最大负载测试，不显示在前端，不影响当前显示内容。其他功能与 `max-load` 一致。

3.5.2 GPU 最高帧率测试

```
"dr-g -gpu max-fps"
```

运行成功后，shell 打印 GPU 应用在当前系统下可以达到的最高帧率。最高帧率就是屏的刷新率，如果打印的值与 60fps 相差较大，那么需要排除屏的刷新率配置。

运行结果：

```
130|rk3399_mid:/data # dr-g -gpu max-fps
GPU test ..
EGL version 1.4

The gpu test can reach to max 59.94 fps
```

3.5.3 GPU 性能测试

```
"dr-g -gpu perf"
```

性能测试，对当前系统 GPU 性能是否满足预期测试，测试完成会打印本次测试消耗的时间，当前平台完成的参考时间。测试前需要把系统 CPU,DDR,GPU 频率定最高(dr-g -sys-set perf)。如果测试消耗时间比参考时间大 10% 以上，那么认为该机器 GPU 性能不能充分发挥。

运行结果：

```
rk3399_mid:/data # dr-g -gpu perf
GPU test ..
The GPU performance is OK ,use time =13395 ms, refs time =16019 ms,[1536x2048]
```

3.5.4 GLES 版本支持测试

```
"dr-g -gpu gles11"
```

对 GLES 1.1 测试，运行后，shell 打印 egl 信息表示 GLES1.0 运行成功。

运行结果：

```
rk3399_all:/ # dr-g -gpu gles11
GPU test ..
```

```
GLES1.1 API test OK
```

```
"dr-g -gpu gles20"
```

对 GLES 2.0 测试，运行后，shell 打印 egl 信息表示 GLES2.0 运行成功。

运行结果：

```
GLES1.1 API test OK
rk3399_all:/ # dr-g -gpu gles20
GPU test ..
```

```
GLES2.0 test OK
```

```
"dr-g -gpu gles30"
```

对 GLES 3.0 测试，运行后，shell 打印 egl 信息表示 GLES3.0 运行成功。

运行结果：

```
rk3399_mid:/data # dr-g -gpu gles30
GPU test ..
GLES3.0 Test OK
```

3.5.5 OpenCL 支持测试

```
"dr-g -cl"
```

系统是否支持 openCL，运行后如果提示"OpenCL test OK !" 表示该平台支持 opencl。

运行结果：

```
rk3399_mid:/ # dr-g -cl
CL test ..
load 64bit
OpenCL version[1.2]
OpenCL DV version[OpenCL 1.2 v1.r14p0-01re10.bbe559ee339d53ef73edfec755f4120e]
OpenCL test OK !
rk3399_mid:/ #
```

3.5.6 Vulkan 支持测试

```
"dr-g -vk info"
```

打印 vulkan 版本信息，检测支持的 vk 扩展，检测支持的 vk 功能。

运行结果：

```
rk3399_mid:/ # dr-g -vk info
VK test ..
vk_info
Instance Extensions [4]:
    VK_KHR_surface (v25)
    VK_KHR_android_surface (v6)
    VK_EXT_debug_report (v2)
    VK_ANDROID_native_buffer (v1)
Instance Layers [0]:
PhysicalDevices [1]:
    "Mali-T860" (INTEGRATED_GPU) 1.0.18/0x89622354 [13b5:8602000]
```

```
"dr-g -vk test"
```

vulkan sample, 绘制一个简单的三角形。

运行结果:

```
rk3399_mid:/ # dr-g -vk test
VK test ..
vk test
w=1536,h=2048,xdpi=320.000000,ydpi=320.000000,fps=60.000004,ds=1.750000
->createSwapChain
Got 3 formats
<-createSwapChain end
size = 792
size = 528
vulkan test ok
```

运行后屏幕显示一个三角形，命令行打印 vulkan test ok。表示系统 vulkan 驱动环境是 OK 的。

3.6 VOP 自动测试

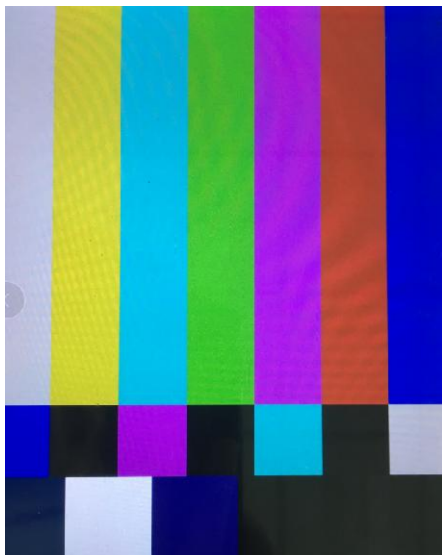
3.6.1 VOP 最大 ddr 负载测试

```
"dr-g -vop -c chip-type"
```

该命令设置后，dr-g 会配置 vop，并增加系统 ddr 负载，用以测试在预设模式下 ddr 条件是否可以正确输出图像。 chip-type 可设置的参数有：

"rk3399", "rk3326", "rk3368", "rk3328", "rk3288", "rk3128", "rk3188"。

运行结果:



3.6.2 VOP demo 测试

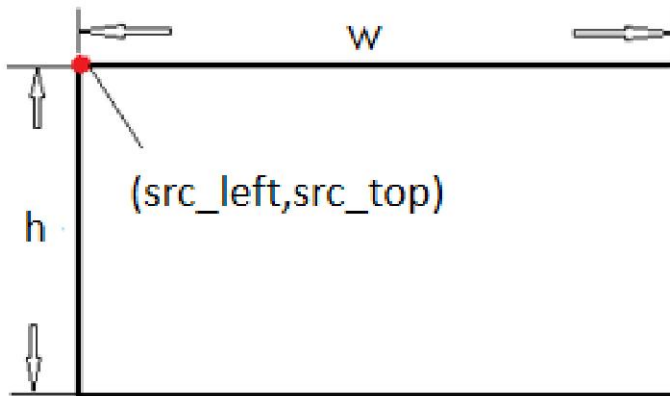
```
"dr-g -vop -w <win_id>@<crtc_id>:<src_left>,<src_top>,<src_w>,<src_h>:<dst_left>,<dst_top>,<dst_w>,<dst_h>[#zpos][@<format>]"
```

通过命令参数配置 vop 的对应 win_id 输出图像，提高调屏时的工作效率。

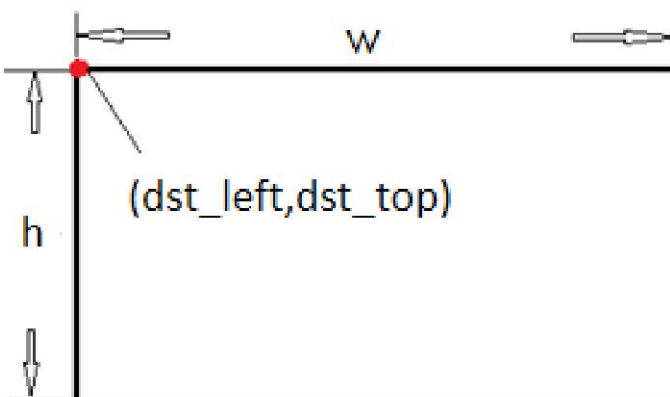
命令 demo: dr-g -vop -w 0@0:0, 1536, 2048:0, 0, 1536, 2048#1@AR24 -f 1536x2048@AR24

运行结果:

源数据:



目标数据:



3.7 LOG 自动检测功能

3.7.1 当前 log 分析

```
"dr-g -log-ans"
```

对当前 logcat 存储的 log 与预设的报错日志规则进行匹配，若检测到对映 log，则报出警告，并给出相映的修改意见。前期收集与整理报错日志多花时间，在后期使用中能大大提高工作效率。用机器提前匹配一次 log，并打印出某行，直接定位，无需人工在复杂的 log 中定位报错。

运行结果：

```
rk3328_mid:/ # dr-g -log-ans
Searching for typical Mali err logs.
Found a line of typical err log :
    06-25 15:59:21.612 1384 1384 E          : Runtime and build version of gralloc private structure does no
match for handle
Please ensure that in your source project hardware/rockchip/libgralloc and (null) are both the same as the SDK ve
sion you base on.
```

3.7.2 XML 规则文件

log_id: 规则 log 的 id 号

match_str: 规则匹配字符串

advice: 给出相应的建议

```
<collection instructions="This is the rule of log analysis">
  <rule>
    <log_id>1</log_id>
    <match_str>Kernel module may not have been loaded</match_str>
    <advice>此报错是由于kernel以module的形式加载mali驱动，且加载失败了，/dev/mali0节点并未被生成。建议：
    请检查/system/lib/module下是否有该模块的.ko文件。或采用build_in的方式编译该驱动。</advice>
  </rule>
  <rule>
    <log_id>2</log_id>
    <match_str>Runtime and build version of gralloc private structure does not match for handle</match_str>
    <advice>Gralloc版本与预期版本对不上。建议：请同步SDK对应匹配的Gralloc版本。</advice>
  </rule>
  <rule>
    <log_id>3</log_id>
    <match_str>Failed to set damage region on surface</match_str>
    <advice>此报错多发生在安卓8.1系统，是hwui框架代码，在abandon之后，无返回直接报fatal导致的。可以参考redmine:189373</advice>
  </rule>
  <rule>
    <log_id>4</log_id>
    <match_str>beginning of crash</match_str>
    <advice>发现有Android Crash信息，可以搜索关键字：beginning of crash定位，查看堆栈</advice>
  </rule>
</collection>
```