MPI FAQ

文件标识：RK-PC-YF-560

发布版本：V0.7.0

日期：2022-04-05

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

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： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文档主要描述多媒体处理应用开发过程中遇到的常见问题及解决方法。

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

版本号	作者	修改日期	修改说明
V0.1.0	FXW	2021-09-08	初始版本
V0.2.0	LYH	2021-10-09	添加内存优化
V0.3.0	LXH	2021-10-11	增加VI/VENC FAQ
V0.4.0	HXM	2021-10-11	添加VGS/TDE FAQ
V0.5.0	ZDD	2021-10-11	添加AUDIO FAQ
V0.6.0	JKM	2021-12-10	添加AVS FAQ
V0.7.0	XLM	2022-04-05	更新物理地址获取说明

目录

MPI FAQ

1. SYS
    - 1.1 日志信息
        - 1.1.1 查看MPI日志信息
    - 1.2 内存使用
        - 1.2.1 无法获取物理地址
        - 1.2.2 无法获取缓存块
        - 1.2.3 打开CMA配置
        - 1.2.4 优化媒体内存方法
    - 1.3 性能优化
        - 1.3.1 CPU性能
        - 1.3.2 解码性能
        - 1.3.3 VPSS性能
        - 1.3.4 VO性能
        - 1.3.5 应用性能优化建议
    - 1.4 系统相关
        - 1.4.1 跨进程说明
        - 1.4.2 其他系统相关
2. VDEC
    - 2.1 接口调用错误
    - 2.2 送流接口返回错误
    - 2.3 播放卡顿
    - 2.4 播放无输出
    - 2.5 解码输出帧数不足
    - 2.6 分析VDEC常见异常日志
3. VENC
    - 3.1 编码支持类型及规格参数
    - 3.2 如何排查没有数据输出
    - 3.3 编码输出帧率不足
    - 3.4 输出图像马赛克
    - 3.5 缩放、裁剪编码功能配置
    - 3.6 编码输出分辨率配置
    - 3.7 码率控制异常问题
    - 3.8 压缩格式输入支持
    - 3.9 其他注意事项
4. VI
    - 4.1 如何获取支持的分辨率
    - 4.2 输出帧率不足
    - 4.3 如何配置VI设备
    - 4.4 如何配置VI通道
    - 4.5 无法获取数据流
        - 4.5.1 一帧数据都无法获取
        - 4.5.2 中途出现无法取流
    - 4.6 使能用户图片功能
    - 4.7 其他注意事项
5. VO
    - 5.1 配置VO底色
    - 5.2 显示输出配置
    - 5.3 同显异显输出
    - 5.4 VO图层使用
6. VPSS
    - 6.1 常见问题分析
        - 6.1.1 无法获取Group帧
        - 6.1.2 无法从通道获取帧
        - 6.1.3 解决获取的图像花屏的问题

6.1.4 送帧超时
6.1.5 取帧超时
6.2 最优性能实践
6.2.1 通道模式最佳配置
6.2.2 电子放大
7. VGS
7.1 版本和调试信息
7.1.1 版本信息
7.1.2 调试信息获取
7.1.3 VGS支持的规格
7.2 问题调试
7.2.1 分析Crop处理图像绿屏
7.2.2 分析抠图边缘出现绿边或者花边
7.2.3 ARGB1555/ARGB8888 图像处理与OSD贴图
7.2.4 ARGB1555 格式OSD叠加时未达到期望的透明度
8. TDE
8.1 版本和调试信息
8.1.1 版本信息
8.1.2 调试信息获取
8.1.3 TDE支持的规格
8.2 问题调试
8.2.1 多次调用TDE失败
8.2.2 外部源数据使用TDE处理出现绿边
8.2.3 缩放或者格式转换出现绿屏
8.2.4 压缩图像拷贝出现绿屏
8.2.5 YUV400 格式支持
8.2.6 拷贝是否支持输入输出为同一块内存
8.2.7 调用 RK_TDE_EndJob 出现段错误
8.2.8 调用 RK_TDE_EndJob 出现报错
8.3 性能相关
8.3.1 TDE处理耗时分析
9. AUDIO
9.1 AO/AI对接调试步骤
9.1.1 首先通过命令查看音频驱动注册的声卡
9.1.2 用命令行测试声卡播放和录音功能
9.1.3 AO/AI软件接口对接
9.1.4 AO/AI打开声卡的两种方式
9.2 AO无声或者断音卡顿问题
9.2.1 AO无声
9.2.2 AO断音卡顿
9.3 AI取帧报错
9.4 ADEC/AENC支持格式
10. AVS
10.1 版本和调试信息
10.1.1 版本信息
10.1.2 调试信息获取
10.1.3 AVS支持的规格
11. RGN
11.1 常见问题分析
11.1.1 贴图未生效或显示异常
11.1.2 Overlay贴图存在撕裂或花屏
12. DUMP
12.1 常见问题
12.1.1 dumpsys工具无法使用

1. SYS

1.1 日志信息

1.1.1 查看MPI日志信息

【现象描述】

需要查看日志和调整 log 日志的等级。

【分析解决】

目前日志分为 6 个等级，默认设置为等级 5。等级设置的越高，表示记录到日志中的信息量就越多，当等级为 6 时，此时的信息量非常庞大，会降低系统的整体性能。因此，通常情况下，推荐设置为等级 4 - 5。

目前仅支持设置所有模块日志，可以通过以下几种方式修改日志等级：

- export rt_log_level=x (x为日志等级，值范围1～6），需设置之后启动用户进程方可生效。
- 通过 **RK_MPI_LOG_SetLevelConf** 接口将 **s32Level** 设置为指定等级。

1.2 内存使用

1.2.1 无法获取物理地址

【现象描述】

调用 **RK_MPI_MB_Handle2PhysAddr** 或者 **RK_MPI_MMZ_Handle2PhysAddr** 获取到的物理地址为0。

【分析解决】

无法获取物理地址可能有以下几个原因：

- 仅为 CMA 内存时才能获取到物理地址，需要确认该内存块是否为 CMA 内存，详细方法请见 打开CMA配置。
- 音频内存块均为 OS 分配的内存，不存在物理地址，故无法获取到物理地址。
- VI 内存类型为 **VI_V4L2_MEMORY_TYPE_MMAP** 时，需要增加额外配置。
- VO 通过 **RK_MPI_VO_GetLayerFrame** 获取的buffer，暂时无法获取物理地址。

1.2.2 无法获取缓存块

【现象描述】

调用 **RK_MPI_MB_VirAddr2Handle** 无法获取到对应的 MB_BLK。

【分析解决】

无法获取 MB_BLK 可能有以下几个原因：

- 此虚拟地址必须为未释放的，且不能超出 MB_BLK 中虚拟地址的有效范围。
- 音频内存块均为 OS 分配的内存，暂时无法通过该接口获取对应 MB_BLK。
- 虚拟地址所对应的 MB_BLK 必须为 DMA 内存，即通过 RKMPI SYS/MMZ 模块接口申请的 MMZ 内存块，或通过 VI/VPSS/VDEC 等模块获取的视频帧中携带的 MB_BLK。

1.2.3 打开CMA配置

【现象描述】

默认情况下，我们使用 IOMMU 非连续物理内存，但在某些特定场景下，需要使用 CMA 物理连续内存时，以能够获取物理地址进行操作。

【分析解决】

- 内核驱动修改，配置 CMA 预留内存大小

  以RK356X驱动为例，在DTS中预留CMA内存，大小根据实际需要调整，单位Byte。如果没有添加inactive，将不是媒体模块独占内存，当系统内存不足时，将从这段预留内存申请，修改补丁如下：

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3568-nvr.dtsi
b/arch/arm64/boot/dts/rockchip/rk3568-nvr.dtsi
index a60fa8f0c9d4..c9ddf42a55ee 100644
--- a/arch/arm64/boot/dts/rockchip/rk3568-nvr.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-nvr.dtsi
@@ -65,9 +65,16 @@
compatible = "shared-dma-pool";
inactive;
reusable;
-           size = <0x0 0x20000000>;
+           size = <0x0 0x00000000>;
alignment = <0x0 0x1000>;
};
+       linux_cma: linux-cma {
+           compatible = "shared-dma-pool";
+           reusable;
+           size = <0x0 0x30000000>;
+           alignment = <0x0 0x1000>;
+                       linux,cma-default;
+       };
};
```

- 用户态配置，目前可以通过以下几种方式使能 CMA 内存
  - export rt_mem_heap=0x1200。 输入该命令后， 将会使得各模块（除VO模块输出 Buffer）默认使用CMA内存。
  - 调用 **RK_MPI_MMZ_Alloc** 申请内存时，**u32Flags |= RK_MMZ_ALLOC_TYPE_CMA**。
  - 如果需要将内存池的内存使用为CMA内存，可以在调用 **RK_MPI_MB_CreatePool** 创建内存池时，将传参中的 **enDmaType** 设置为**MB_DMA_TYPE_CMA**。
- 若 VI 模块 VI_ISP_OPT_S **enMemoryType** 配置为 **VI_V4L2_MEMORY_TYPE_MMAP**，此时要获取输出帧物理地址，则需增加如下配置

```
diff --git a/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
b/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
index d4035a360..dae9747be 100644
--- a/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
+++ b/arch/arm64/boot/dts/rockchip/rk3568-evb1-ddr4-v10.dtsi
@@ -399,17 +399,23 @@
+&rkcif_mmu {
+    status = "disabled";
+};

 &rkisp_mmu {
-    status = "okay";
+    status = "disabled";
 };
```

1.2.4 优化媒体内存方法

以下内存优化方法，需要结合具体产品/场景确认对应的参数配置合理值。

- **VO模块配置**

措施	相关接口	收益	影响	注意
像素格式改为NV12	RK_MPI_VO_SetLayerAttr: enPixFormat = RK_FMT_YUV420SP	相比RGBA8888一个素格节省2.5 Byte	GPU合成性能	需根据不同平台的GPU合成效率，配置相应的合成策略。 如在RK356X 双屏4K+1080P场景时，4K屏配置为NV12，1080P屏配置为RGB888
显示缓存队列buffer数设置为3	RK_MPI_VO_SetLayerDispBufLen	默认4个buffer	VO显示的流畅性	-
wbc像素格式配置为NV12	RK_MPI_VO_SetWbcAttr: enPixFormat = RK_FMT_YUV420SP	节省像素占用的内存	-	-
wbc输出缓存buffer数配置为3	RK_MPI_VO_SetWbcDepth	默认4个buffer	-	-

- **VDEC模块配置**

措施	相关接口	收益	影响	注意
配置解码输出帧缓存数	RK_MPI_VDEC_CreateChn: u32FrameBufCnt = 3	默认为6帧	过少可能导致解码卡住	H.264/H.265每个解码通道输出帧缓存至少为参考帧+显示帧+1；JPEG/MJPEG每个解码通道输出帧缓存至少为显示帧+1；
配置解码输入码流缓存数	RK_MPI_VDEC_CreateChn: u32StreamBufCnt = 3	默认为8包	过少会增加解码器等待数据输入耗时	-
H.264/H.265不申请MV buffer	RK_MPI_VDEC_SetChnParam: bTemporalMvpEnable = RK_FALSE	节省MV buffer	需要MV却没创建会导致画面频繁跳转不连续， 内存访问越界，内核日志频繁打印iommu: Page fault	1. 如果 H.264 解码不需要解码 B 帧/ H.265 解码不需要解码支持时域运动矢量预测（sps_temporal_mvp_enabled_flag = 1）的码流， 则配置 bTemporalMvpEnable 为 FALSE；否则配置为TRUE； （若H.265 解码配置为 FALSE，但码流中 sps_temporal_mvp_enabled_flag为1，仍会去创建MV内存。） 2.目前限定需配置为解码序输出，以上才可生效。 RK_MPI_VDEC_SetChnParam：enOutputOrder = VIDEO_OUTPUT_ORDER_DEC
限制解码总/驻留buffer	配置通道内存分配总大小: export rt_vdec_mb_max_alloc=209715200 //200M 配置通道内存驻留总大小: export rt_vdec_mb_max_retain=104857600 //100M	默认不限制总大小， 解码完buffer不驻留	-	如果限制总大小，超出的解码通道将会创建失败。 配置通道驻留内存目的是为了加快通道创建销毁速度。
关闭编解码器packet/frame buffer状态记录	export buf_slot_debug=0x0 (默认值为0x10000000)	一路会开辟16k*2 (一个存packet，一个存frame)	-	-

- **VENC模块配置**

措施	相关接口	收益	影响	注意
配置编码输出包缓存数	RK_MPI_VDEC_CreateChn: u32StreamBufCnt = 3	默认配置为4包	-	-

- **ZRAM内存压缩**

  ZRAM 是 Linux 的一种内存优化技术，可通过开启ZRAM，当系统内存紧张时释放出更多的可用内存。以下脚本示例为开启512MB大小ZRAM内存空间。

```
#!/bin/sh
echo $((512*1024*1024)) > /sys/block/zram0/disksize
mkswap /dev/zram0
swapon  /dev/zram0
sysctl -w vm.swappiness=100
```

1.3 性能优化

1.3.1 CPU性能

- 对精度要求不高的场景下，可调整调度时间片、关闭高精度定时器，来提升CPU Idle。

```
// 修改调度时间片为100Hz
CONFIG_HZ_100=y
// 关闭高精度定时器
#CONFIG_HIGH_RES_TIMERS=y
```

- 分散中断，比如将以太网中断eth0中断迁移出CPU0。
- CPU频率定频、提升运行频率。

1.3.2 解码性能

- H264、H265解码器通道配置为帧压缩模式，以降低 DDR 带宽占用率，提升解码性能。

```
VDEC_CHN_PARAM_S stVdecParam;
stVdecParam.stVdecVideoParam.enCompressMode = COMPRESS_AFBC_16x16;
RK_MPI_VDEC_SetChnParam(VdecChn, &stVdecParam);
```

- MJPEG 不支持帧压缩，不能开启。

1.3.3 VPSS性能

- 通道模式配置策略详见通道模式最佳配置。
- 在RK356X平台，VPSS采用GPU模块实现，需要关注GPU频率配置。

1.3.4 VO性能

以RK356X平台为例：

- 建议HDMI最大支持4KP30，VGA最大支持1080P30。4K/双屏场景下，主副屏输出帧率应配置为30fps，否则性能会不足。
- 屏幕输出格式PixelFormat建议配置RGB888。

```
VO_VIDEO_LAYER_ATTR_S stLayerAttr;
stLayerAttr.enPixFormat = RK_FMT_BGR888;
stLayerAttr.u32DispFrmRt = 30;
RK_MPI_VO_SetLayerAttr(VoLayer, &stLayerAttr);
```

1.3.5 应用性能优化建议

- 减少线程数量以减少调度开销。
- 减少sleep，poll，timer定时器使用或适当调大调度间隔时长。
- 开启 O3/Os 编译优化。

1.4 系统相关

1.4.1 跨进程说明

【现象描述】

目前除 VO 模块外，其他 MPI 模块不支持跨进程操作，但根据项目需求在持续研发。目前 VO 模块视频层是通过 DRM 框架接口操作，如果 UI 在另一个进程操作也是通过DRM接口操作，两个进程可能会出现竞争问题，目前Rockit接口没有直接支持跨进程操作，但可以通过共享 Buffer 方式实现跨进程。

【分析解决】

- UI 初始化为单 Buffer 模式，把 VO 申请的 UI Buffer 转成唯一 ID，在通过 pipe/socket 等跨进程方式将此唯一 ID 传递给 UI 进程，UI 进程收到后再通过查询接口获取Fd，然后 map 出虚拟地址，写入 UI 数据。
- 鼠标跨进程支持也可以用此方法。
- 我们有提供 Sample 示例代码，可以通过我们支持人员获取。

1.4.2 其他系统相关

- NVR 相关产品
  详见文档 NVR SDK 文档《Rockchip_RK356X_NVR_SDK_Release_xxx_CN.pdf》，xxx为版本信息，不断迭代更新。

2. VDEC

VDEC 模块常见问题可借助 dumpsys vdec 工具进行分析排查。

2.1 接口调用错误

2.2 送流接口返回错误

【现象描述】

调用送流接口 **RK_MPI_VDEC_SendStream** 返回 **RK_ERR_VDEC_BUF_FULL** 错误。

【分析解决】

- 送流接口超时时间设置过短，导致内部输入缓冲未及时消耗，而出现概率无法送入数据，建议超时时间设置为 200ms 以上。
- max_input_cnt 被设置过小，导致易出现缓冲队列满，可尝试修改 u32StreamBufCnt 来增加输入缓冲的个数，比如设置8块。
- max_output_cnt 被设置过小，导致解码 Buffer 轮转不够，可尝试修改 u32FrameBufCnt 来增加输出轮转的个数，比如设置8块。

```
id    send_rate   max_input_cnt  left_input_cnt  left_input_size  max_output_cnt
err_status
0     19.35       8              7               41540            8
0
id    input_strm_cnt   output_frm_cnt   error_frm_cnt   unused_buf_cnt
put_buf_cnt
0     77               70               0               0               66
```

- 解码 Buffer 采用轮转模式，当出现后级绑定模块或用户取帧后归还慢，则会导致解码器没有足够的可用 Buffer 解码，从而导致解码变慢，输入缓冲堆积造成无法继续送流，可查看 dumpsys vdec 中的可用解码 Buffer(unused_buf_cnt) 个数是否为零，可能有如下原因:
  - 确认 VDEC 后级模块是否长时间占用解码 Buffer。
    - dumpsys vpss，确认 VPSS 是否所有通道开启 USER 模式，全部开启 USER 模式会导致解码 Buffer 占用时间长。把不需要缩放的通道修改成 PassThrough 模式以降低模块处理开销，加快 Buffer 轮转速度。

```
---------------------- vpss channel attr -----------------------------
grp_id    chn_id    mode      width     height    pixel_format
is_compress    src_rate  dst_rate  depth     align     mirror    flip
0         0         USER      1920      1080      image:yuv420p  N
       -1        -1        1         16        N         N
1         0         USER      1920      1080      image:yuv420p  N
       -1        -1        1         16        N         N
2         0         USER      1920      1080      image:yuv420p  N
       -1        -1        1         16        N         N
```

    - dumpsys vo，确认 VO 的 Cache 缓存帧数是否接近 VDEC 总解码 Buffer个数，此时可先确认 GPU 负载，若 GPU 负载超过90%，确认是否有关键日志出现，可以查看分析VDEC常见异常日志。

```
LAYER clu0 CHANNEL STATUS:
ChnId   Prio    X       Y       W       H       ChnFrt  FgAlpha BgAlpha
Show    Pause   Step    Cache   RevCnt
0       2       265     79      1650    888     13      0       128
Y       N       N       8       93
```

```
cat /sys/devices/platform/fde60000.gpu/utilisation  // 查看gpu负载
```

  - 通过 RK_MPI_VDEC_GetFrame 获取帧后，要确保及时归还，保证归还速率控制在预期帧率范围内。

- 解码能力不足导致输入缓冲堆积而出现概率无法送流
  - 应用业务是否超出该芯片对应的最大解码能力，可查看 VDEC 模块文档概述章节。
  - 系统带宽不足，需结合业务整体分析优化 DDR 带宽。
  - 可通过以下命令确认硬件解码能力耗时，内核日志会输出硬件解码耗时。

```
echo 0x100 > /sys/module/rk_vcodec/parameters/mpp_dev_debug
```

- 确认是否误设置 bEndOfStream 为 RK_TRUE，解码通道在处理完此码流包后，将不再继续解码，导致输入缓冲堆积而无法继续送流。

  VDEC_STREAM_S#bEndOfStream 设置为 RK_TRUE，表示此码流包为当前通道最后一包码流，当解码通道处理完此码流包后，将会输出所有已解码帧，并不再继续解码，若要恢复解码，则需要按如下的调用流程来重置解码器才能正常解码：

```
RK_MPI_VDEC_StopRecvStream -> RK_MPI_VDEC_ResetChn ->
RK_MPI_VDEC_StartRecvStream
```

  当码流为非实时码流且需要获取送入解码通道所有帧时，需要在码流最后一包设置 bEndOfStream 为 RK_TRUE，或者送入一个 bEndOfStream 为 RK_TRUE的空包，通过 RK_MPI_VDEC_GetFrame 获取最后一帧，最后一帧 VIDEO_FRAME_S.stVFrame.u32FrameFlag 会标记上 FRAME_FLAG_SNAP_END。
  - 若是空包设置 bEndOfStream，最后一帧 RK_MPI_MB_GetLength(sFrame.stVFrame.pMbBlk) 等于零。
  - 若是有效码流包设置 bEndOfStream，最后一帧 RK_MPI_MB_GetLength(sFrame.stVFrame.pMbBlk) 大于零。
- Stream 模式输入队列配置过小，无法拼成完整包解码，导致后续也无法继续送入码流。
  当解码通道属性 VIDEO_MODE_E 配置为 VIDEO_MODE_STREAM 时，表示送入的码流可能为非完整帧对应的码流，解码通道内部会进行分帧处理，当码流包被拆分的过小且 max_input_cnt 设置大小不足一个完整包对应的个数，导致解码器无法解码，可尝试修改 u32StreamBufCnt 来增加输入缓冲的个数。

2.3 播放卡顿

【现象描述】

解码输出帧率不足或显示输出时出现播放卡顿现象。

【分析解决】

- 送数据帧率不足
  dumpsys vdec，若 send_rate 的帧率小于预期，且 left_input_size 和 left_input_cnt 为零，表示输入数据不够导致播放卡顿，需要检查输入数据的速率。

```
id      send    send_ok    send_rate   max_input_cnt  left_input_cnt
left_input_size
0       386     385        19.35       8              0               0
```

- **解码能力不足**
  解码能力不足，会导致解码器报错 **RK_ERR_VDEC_BUF_FULL**，可参考送流接口返回错误。

2.4 播放无输出

【现象描述】

解码无输出帧或显示画面黑屏（无解码图像输出）。

【分析解决】

- **VDEC未接收到数据**
  dumpsys vdec，send 值为0，说明 **RK_MPI_VDEC_SendStream** 收到的数据为零，需要检查输入情况。

```
id      send    send_ok    send_rate   max_input_cnt  left_input_cnt
left_input_size
0       0       0           0          8              0               0
```

- **VDEC未解码**
  - 若 input_strm_cnt 大于零，output_frm_cnt 等于零，说明解码没有工作，检查log，看解码器是否有错误产生。
  - 若 input_strm_cnt 大于零，output_frm_cnt 大于零，且不断增长，说明解码器正常，需要检查后级模块是否丢失解码数据。
  - 若 input_strm_cnt 大于零，output_frm_cnt 大于零，且没有不断增长，说明解码器只解码部分，需要检查数据停止输入导致。
  - 若 input_strm_cnt 和 error_frm_cnt 个数接近，且不断增长，说明解码的数据有问题，需要检查输入数据的完整性，比如是否是完整的一帧。
  - 若 output_frm_cnt 不增加，unused_buf_cnt 为零，且 output_frm_cnt 减去 put_buf_cnt 等于 max_output_cnt，说明解码器没有buffer可用，buffer被其他模块占用

```
id              input_strm_cnt output_frm_cnt error_frm_cnt
unused_buf_cnt put_buf_cnt
0               7134           6061           2              0
   6057
```

2.5 解码输出帧数不足

【现象描述】

解码通道输出帧数小于送入解码器的完整码流包个数。

【分析解决】

- dumpsys vdec，确认是否有错帧导致的解码帧数不足，error_frm_cnt 大于零说明有错帧产生，总的帧数等于 error_frm_cnt + output_frm_cnt。

```
id              input_strm_cnt output_frm_cnt error_frm_cnt  unused_buf_cnt
put_buf_cnt
0               7134           6061           2              0
6057
```

- 当输入码流为非实时码流时，可将VDEC_STREAM_S#bEndOfStream 设置为 RK_TRUE，以获取送入解码通道的所有帧。

2.6 分析VDEC常见异常日志

- 当日志出现类似错误信息， 说明进程的 fd 已耗尽，可以使用命令 ulimit -n 4096 来调整进程的 fd 个数。

```
fail to drm_ioctl(fd = 17, req =-1072929747), error: Too many open files
```

- 当日志出现类似错误信息， 说明 GPU 负载较重，合成变慢。

```
mpi_vo_composer  03:29:33-736 {vo_composer_thread:968} Layer 0 compose cost
61607 us, frame interval 40000 us
```

3. VENC

3.1 编码支持类型及规格参数

查看《Rockchip_Developer_Guide_MPI_VENC.md》的“概述”章节部分。

3.2 如何排查没有数据输出

- 首先使用rk_mpi_venc_test测试，若测试正常，检查下与代码示例的差异。
- 检查s32RecvPicNum是否设置为0，或者通过dumpsys venc查看snap_set是否为非-1并且seq已经等于snap_set数值，此时编码会停止输出。
- 检测输出buffer是否设置太小，导致编码输出异常报错。
- 查看串口日志是否有mpp或者其他相关报错。

3.3 编码输出帧率不足

- 确认帧率是否超过实际规格书定义。
- 确认单独使用rk_mpi_venc_test是否帧率正常。如果test测试也异常，查看芯片温度是否过高导致机器进入温控，查看编码频率是否有降低。如果都正常则串口输入以下命令查看内核编码单帧时间是否过长。

```
echo 0x100 > /sys/module/rk_vcodec/parameters/mpp_dev_debug
```

- 确认是否开启帧率控制导致，可以通过dumpsys venc查看。
- 确认是否是VENC通道后级绑定模块归还Buffer数据慢，或用户获取/释放Buffer的速度慢。
- 若单独编码正常，加上其他业务出现帧率低，此时需分析DDR带宽是否充足，可以尝试提高DDR频率，并结合整体业务分析优化带宽占用。

3.4 输出图像马赛克

- 确认输入图像是否正常。
- 确认单独使用rk_mpi_venc_test是否正常。
- 使用rk_mpi_vdec_test解码查看是否正常。
- 检查是否超频使用编码频率，降低对应clk查看是否正常。

```
H264/H265编码: cat sys/kernel/debug/clk/clk_summary | grep venc
JPEG/MJPEG编码(RV1109/RV1126/RK356x): cat sys/kernel/debug/clk/clk_summary |
grep vepu
JPEG/MJPEG编码(RK3588): cat sys/kernel/debug/clk/clk_summary | grep jpeg
```

- 如果只有jpeg/mjpeg编码输出图像异常，查看是否输入为压缩数据。可以通过dumpsys venc中的afbc_mode查看，为0x0为非压缩格式，其余则为压缩格式。
- 尝试使用其他编码器是否正常。如果是mjpeg/h264/h265单独一个编码器异常，查看对应编码器电压供电是否满足需求，尝试抬升25-50mv查看是否有改善。

3.5 缩放、裁剪编码功能配置

支持，目前VENC模块支持裁剪、缩放功能，具体可以查看 test_mpi_venc.cpp 及《Rockchip_Developer_Guide_MPI_VENC.md》的RK_MPI_VENC_SetChnParam 函数说明。

3.6 编码输出分辨率配置

- 当通过RK_MPI_VENC_SetChnParam 设置了缩放功能时，编码器统一按照缩放设置的分辨率输出。
- 编码器默认采用 **自适应分辨率编码**，即当没有开启缩放功能时，按照输入的实际分辨率编码输出，一般JPEG抓图编码可以使用此功能，避免频繁创建销毁编码器。需要注意的是通道输出的 Buffer (u32BufSize) 要按照最大分辨率的 Buffer 大小进行分配，避免因为创建时申请的输出 Buffer 过小导致动态切换大分辨率后无法正常编码。

3.7 码率控制异常问题

- 确认输入数据及输出码流数据都正常，无花屏、马赛克等异常数据。
- 查看码控相关信息设置是否正确，通过 dumpsys venc 关注 venc chn attr 查看 rc_mode 及 gop 信息，及 venc chn rc info 的 qp 及 bps 设置值。
- 保存编码输出码流，从码流qp进行分析，如果 qp 在 qp max 附近说明视频序列比较复杂，这种情况一般真实的码率会大于设置码率，如果 qp 在 qp min 附近，说明视频序列偏简单，这种情况一般真

实码率会小于设置码率的。可以通过调节 RK_MPI_VENC_SetRcParam 中 qp min/qp max（H264/H265）或qfactor(JPEG/MJPEG) 进行调整相应 qp。

- 如果输入源为 sensor 数据，存在超码率问题，需要保证输入图像噪点较小，如果输入图像本身质量较差，码率则一般也会较大。

3.8 压缩格式输入支持

- H264/H265 仅支持YUV420、YUV422压缩格式输入。
- JPEG/MJPEG 硬件编码器本身无法支持 AFBC 压缩格式输入，目前仅 RK3588 平台支持编码通道内部通过扩展模块来支持压缩格式输入，其他平台的 JPEG/MJPEG 压缩数据需要通过 VPSS 模块进行解压缩后再送入 VENC，此处理会额外占用带宽及硬件资源，因此仅建议在抓图等少量帧编码的场景下采用输入压缩图像，否则建议前级模块关闭压缩输出。

3.9 其他注意事项

- 动态参数设置建议在创建通道后、启动编码前设置。避免在编码过程中频繁设置。
- 调用 RK_MPI_VENC_StopRecvFrame 后，建议调用 RK_MPI_VENC_ResetChn 或取走输出码流，否则有可能前级模块无法正常工作。

4. VI

4.1 如何获取支持的分辨率

- 分辨率支持情况可以查看《Rockchip_Developer_Guide_MPI_VI.md》的功能描述章节介绍。
- 是否支持某个分辨率可以通过v4l2-ctl命令去抓流验证，目前VI支持的分辨率跟v4l2-ctl支持的一致。命令示例如下：

```
v4l2-ctl -d /dev/video19 --set-fmt-
video=width=1920,height=1080,pixelformat=NV12 --stream-mmap=3 --stream-skip=3
--stream-count=10 --stream-poll  --stream-to=/tmp/12M.yuv
```

- hdmi_rx输入时可以使用以下命令查看输入源的格式及分辨率：

```
v4l2-ctl -d /dev/video17 --get-fmt-video
```

也可以使用RK_MPI_VI_GetChnConnectInfo函数获取对应信息。

4.2 输出帧率不足

- 确认输入帧率是否正常。ISP 输入可通过cat /proc/rkisp*查看rate信息。
- 确认是否开启帧率控制。可以通过 dumpsys vi 查看。
- 确认是否是VI通道后级绑定模块归还 Buffer 数据慢，或用户获取/释放 Buffer 的速度慢。

- 若VI单独输出帧率正常，而绑定 VO 后输出帧率不足，可提高 VI 的输出 Buffer 个数，建议 Buffer 个数设置不小于 4 个。

4.3 如何配置VI设备

- 全平台可以通过设置 aEntityName 指定节点来选择设备，如 /dev/video0 、/dev/video19 等。
- 全平台带 Sensor 的设备可以通过 dev 号进行区分。device 编号根据 dts 配置中 sensor 连接的属性 “camera-module-index” 值一一对应 ，没有设置时默认dev 0 为 isp0，dev 1 为 isp1，以此类推。

4.4 如何配置VI通道

- 若有指定设备名时，优先按照设备名字指定，aEntityName[MAX_VI_ENTITY_NAME_LEN]; 如 /dev/video0。
- 不指定设备名字时：
  - rv1109/rv1126 chn0 对应 bypass，chn1 对应 scale0，chn2 对应 scale1，chn3 对应 scale2。
  - rk356x chn0 对应 mainpath，chn1 对应 selfpath。
  - rk3588 chn0 对应 mainpath，chn1 对应 selfpath，chn2 对应 fbcpath。

4.5 无法获取数据流

4.5.1 一帧数据都无法获取

- 确认使用 v4l2-ctl 或 rk_mpi_vi_test 测试同分辨率是否正常，如果异常请确认硬件及驱动正常注册。
- 确认打开的分辨率是否是当前通道支持的分辨率。
- 确认打开的格式是否是当前通道支持的格式。
- 通过VI函数返回错误码确认，比如 0xA0088004，则说明没有使能通道。
- 确认 depth 参数配置是否正确，depth 必须配置大于 0。
- 确认配置的 memoryType 参数是否正常，若输入源为不经过isp处理的数据流（hdmi_rx/bt1120），则 memoryType 要配置为1，若输入源为经过isp处理的数据流（sensor）则 memoryType 推荐配置为 4（也可以配置为1，不推荐）。

4.5.2 中途出现无法取流

- 通过 dumpsys vi 查看 get_cnt 及 release_cnt 的信息，如果 get_cnt 与 release_cnt 数值相差为 buf_count 的数值，说明 VI 的 Buffer 都被后级模块引用，没有归还，此时需要通过查看后级模块的 Buffer 使用状态。
- 查看两次 isp/cif 驱动信息中断数是否有增长：
  - 数据流通过 isp 则 cat /proc/rkisp*，查看 “Interrupt Cnt:15047031” 的数值是否有增长。
  - 数据流通过 cif 则 cat /proc/rkcif*，查看 “irq statistics:total:4627756” 的数值是否有增长。

4.6 使能用户图片功能

当出现设备节点没有数据流时，可以通过使能插入用户图片功能（目前支持插入指定 YUV 图或者 RGB 背景色），具体使用可以参考 test_mpi_vi.cpp。

4.7 其他注意事项

- 不支持多个进程/线程同时打开一路数据流，支持一个进程/线程通过bind多个后级或者绑定 VPSS 处理同一路vi数据。同理，VI 或者 v4l2-ctl 同时只能有一个应用能使用同一路数据流，比如 /dev/video0。
- 若数据流未经过 ISP 模块时，VI 模块不支持格式转换及分辨率缩放。如打开设备为 HDMI_RX 并且输入源为1080P RGB888的格式，则 VI 只能按照这个分辨率及格式输出，若需要做格式转换或者分辨率缩放，可通过 VPSS/TDE 等模块处理。

5. VO

5.1 配置VO底色

```
stVoPubAttr.u32BgColor = 0xFF0000;
RK_MPI_VO_SetPubAttr(VoDev, &stVoPubAttr)
```

5.2 显示输出配置

- 目前的HDMI强制输出方案，强制输出后，应用还是能检测到HDMI热拔插事件，可以获取到HDMI的状态。默认配置，uboot，kernel阶段HDMI & VGA强制输出，默认输出分辨率为1024x768。详细配置在arch/arm64/boot/dts/rockchip/rk3568-nvr.dtsi的&route_hdmi 以及&route_edp 节点中。
- 应用通过drm接口可以获取到HDMI接口的状态，上层可以通过RK_S32 RK_MPI_VO_RegCallbackFunc(RK_U32 enIntfType, RK_U32 u32Id, RK_VO_CALLBACK_FUNC_S *pstCallbackFunc) 这个接口注册对应的回调。
- 应用起来后，使能HDMI&VGA对应的VoDev之后对应的设备就是常输出的，建议（HDMI用RK356X_VO_DEV_HD0，VGA用RK356X_VO_DEV_HD1）。

5.3 同显异显输出

以 RK356X 平台为例：

- 同显的模式下，HDMI&VGA 只能输出同一种分辨率。
- 异显的模式下，建议HDMI最大支持4kP30，VGA最大支持1080P30。
- 同显/异显切换，是通过应用层去操作VoDev来实现的。
    - 同显：dts中把edp_in_vp0使能起来 ;应用使能VP0，VoPubAttr.enIntfType = VO_INTF_HDMI | VO_INTF_EDP。

- 异显：应用使能两个VoDev。

5.4 VO图层使用

以RK356X平台为例，对应的默认绑定配置如下：

```
dev 0:   0-video,   4-ui,  6-cursor
dev 1:   2-video,   5-ui,  7-cursor
-------LAYER BIND CONFIG--------
DevId   Video   Gfx     Cursor
0       clu0    esm0    sm0
1       clu1    esm1    sm1
2       unkown  unkown  unkown
```

使用建议：

- 视频层和 UI 层均使用 Cluster 图层， UI 图层数据送往视频层的末尾通道(如VO_MAX_CHN_NUM-1，VO_MAX_CHN_NUM-2通道)。
- 鼠标单独初始化一层32x32大小。

6. VPSS

6.1 常见问题分析

6.1.1 无法获取Group帧

【现象描述】

调用 **RK_MPI_VPSS_GetGrpFrame** 无法获取到帧。

【原因分析】

RK MPI获取Group帧条件较多，需要满足多种条件：

- 该Group存在Passthrough模式的通道。
- 该Group存在与VO绑定的通道。
- backup帧开启。

【解决方法】

解决该问题需要有以下操作：

- 调用 **RK_MPI_VPSS_EnableChn** 开启通道，并且该通道设置为 **VPSS_CHN_MODE_PASSTHROUGH**。
- 调用 **RK_MPI_SYS_Bind** 将条件1的 VPSS 通道同 VO 通道进行绑定。
- 调用 **RK_MPI_VPSS_EnableBackupFrame** 使能backup帧。

6.1.2 无法从通道获取帧

【现象描述】

调用 **RK_MPI_VPSS_GetChnFrame** 无法获取到帧。

【原因分析】

通道无法获取帧的情况有以下几种需要确认：

- 确认该通道是否开启。（可以通过dumpsys vpss确认是否可以查看到该通道的信息，如果查看不到，则是未开启）
- 确认该通道属性 u32Depth 是否不为0。（为0时表示通道不保留帧，全部丢弃）。
- 确认该通道是否绑定了下级。目前不支持绑定下级的同时手动取帧的情况。
- 确认该通道的输出均被 **RK_MPI_VPSS_GetChnFrame** 全部取出后而没有调用 **RK_MPI_VPSS_ReleaseChnFrame** 释放。

【解决方法】

解决该问题需要有以下操作：

- 调用 **RK_MPI_VPSS_EnableChn** 开启通道。
- 调用 **RK_MPI_VPSS_SetChnAttr** 将 **VPSS_CHN_ATTR_S** 中的 **u32Depth** 设置为大于0的值。
- 如果有绑定下级，请调用 **RK_MPI_SYS_UnBind** 进行解绑。
- **RK_MPI_VPSS_GetChnFrame** 和 **RK_MPI_VPSS_ReleaseChnFrame** 必须成对使用。

6.1.3 解决获取的图像花屏的问题

【现象描述】

通过 **RK_MPI_VPSS_GetChnFrame** 或 **RK_MPI_VPSS_GetGrpFrame** 获取到的帧图像花屏。

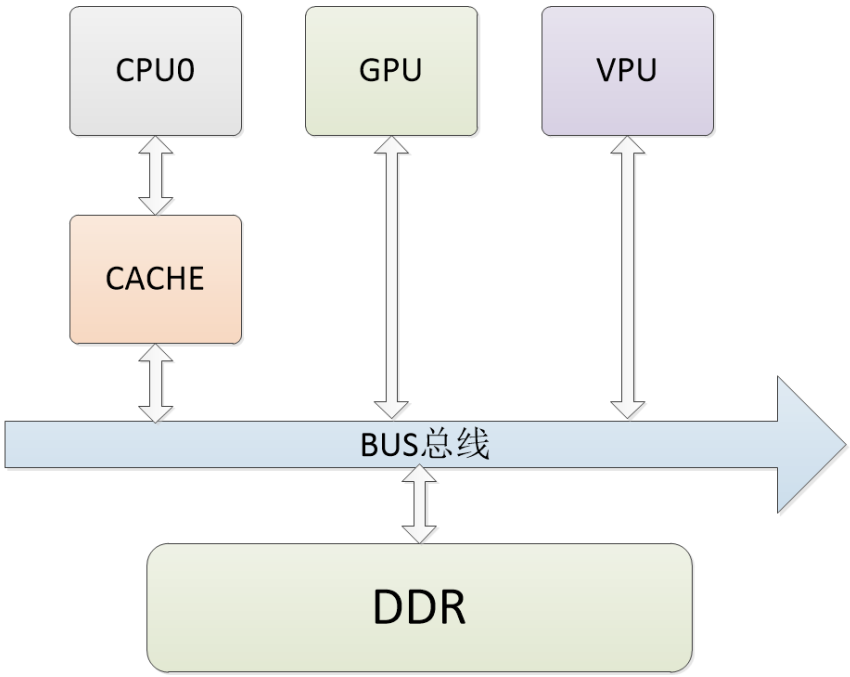
【原因分析】

可能有以下几种原因：

- 图像虚宽高不对齐导致图像数据读取错误。
- CPU cache一致性问题。
- 图像源数据为花屏。
- 调用 **RK_MPI_VPSS_ReleaseChnFrame** 后仍然在使用。

【解决方法】

- 检查图像对齐情况
    - 检查输出图像是否按照 **VIDEO_FRAME_INFO_S** 结构体中的实际宽高、虚宽高（像素对齐宽高）、图像格式、压缩方法使用。各种对齐方式遵循 VPSS 模块开发文档中的对齐方式列表要求。
    - 检查是否将虚宽高当成实宽高使用，如果误设，将会在图像右侧或下方出现黑色条纹（也可能是随机的脏数据）。
- 检查CPU读写图像的cache一致性



这种情况下的典型画面是出现**细条的错误数据**（由于cache整块未被同步，会出现连续的脏数据），原理如上图。解决办法为从图像中读取数据到CPU后调用 **RK_MPI_SYS_MmzFlushCache** 清读写方向的cache（即参数设置为RK_FALSE）。从图像写数据到CPU前调用 **RK_MPI_SYS_MmzFlushCache** 清读方向的cache（即参数设置为RK_TRUE）。

- **检查数据源是否已经花屏**

  此时可以将输入数据写下来，检查输入数据是否花屏。

  ```
  // 录制VPSS 组 0 通道 0 输入数据至/data/out1.nv12，录制10帧
  dumpsys record -m vpss -d 0 -c 0 -i 1 -o /data/out1.nv12 -n 10
  ```

  然后将录制的数据通过7YUV等工具查看。

- **检查是否出现 RK_MPI_VPSS_ReleaseChnFrame 后仍然使用的情况**

  如果在图像被释放仍然读取图像数据，那么这帧图像可能在被重新使用时读取，此时也会表现为花屏，甚至应用直接崩溃。

6.1.4 送帧超时

【现象描述】

调用 **RK_MPI_VPSS_SendFrame** 返回 **RK_ERR_VPSS_BUF_FULL**。

【原因分析】

开启了帧率控制，导致内部送帧阻塞。

【解决方法】

帧率控制开启后的阻塞时间小于送帧超时时间。例如输入帧率设置为30，内部送入阻塞时间将会变为33ms左右，此时送帧超时时间必须大于33ms，否则将可能出现送帧超时的情况。

6.1.5 取帧超时

【现象描述】

调用 **RK_MPI_VPSS_GetChnFrame** 返回 **RK_ERR_VPSS_BUF_EMPTY**。

【原因分析】

取帧超时一般有以下几种可能性：

- 开启了帧率控制。
- 达到性能边界。

【解决方法】

- 开启帧率控制时输出帧率时间小于取帧超时时间。例如设定输出帧率为5帧，那么在没有性能问题情况下取帧间隔为200ms，如果设定取帧超时时间小于200ms，也会出现取帧超时。
- 调用接口超时时间必须大于 VPSS 处理帧的时间，但在系统压力很大的情况下这个时间可能被拉长，需要结合其他模块综合处理性能问题。

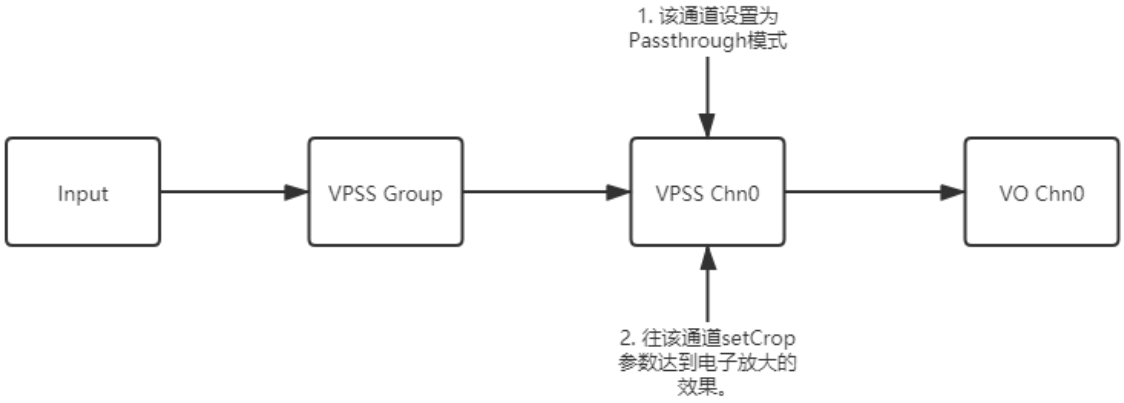
6.2 最优性能实践

6.2.1 通道模式最佳配置

VPSS 模块硬件资源性能受限，在不需要从 VPSS 通道手动获取输出帧（如用于AI算法、用户数据分析处理）情况下，需要将 VPSS 通道的工作模式 **enChnMode** 配置为 **VPSS_CHN_MODE_PASSTHROUGH**。在该模式下，VPSS 模块不做实际数据处理，以透传的方式传递给后级模块，以达到最高效率。

6.2.2 电子放大

电子放大通常使用 VPSS 设置组或通道裁剪，来达到放大缩小的功能。目前RK平台的最优解如下图所示：



如上图所需注意点：

- 通道设置为 **VPSS_CHN_MODE_PASSTHROUGH** 模式。
- 电子放大参数往通道中设，而不是 Group。
- 需要绑定 VO 通道。

7. VGS

7.1 版本和调试信息

7.1.1 版本信息

与 VGS 功能相关的的库包括librockit.so 和 libgraphic_lsf.so。可以通过 dumpsys --version命令一键获取相关版本号。

log相关的版本信息如下：

```
rockit version: git-9625ad8 Tue Sep 28 19:16:44 2021 +0800
rockit building: built-daw 2021-09-29 11:31:52

rk-debug init version=2.93,args[16,16,0], threadId=547547050416
```

7.1.2 调试信息获取

通过dumpsys vgs 获取调试信息， 调试信息包含输入和输出图像的宽高 format、当前和历史处理的job和task的信息等。具体的调试信息参考《Rockchip_Developer_Guide_MPI_DUMP》文档 VGS 章节。

7.1.3 VGS支持的规格

VGS支持常见的图像格式如yuv420、rgb888、rgba8888等, 更多支持规格参考《Rockchip_Developer_Guide_MPI_VGS》文档。

7.2 问题调试

7.2.1 分析Crop处理图像绿屏

- 使用VGS的crop接口进行抠图，抠出来的图像是绿的。出现如下的错误log:

```
rk-debug eglCreateImageKHR NULL dpy=0x7fa8000f00
rk-debug create_egl_img [14,736,480] afbc=0 ,format=35314152,stride=736
,color_space=327f,sample_range=3283
rk-debug eglCreateImageKHR out fail
```

- VGS硬件输入和输出的图像的宽高、虚宽高根据不同的格式有不同的对齐要求。 比如BGRA1555 格式的宽需要32像素对齐，如果输入的图像宽度不是32像素对齐，那么就会输出上述log，抠图失败。同样的输出图像的buffer也需要根据不同格式进行对齐, 否则会出现内存操作越界。

7.2.2 分析抠图边缘出现绿边或者花边

目前VGS硬件处理时，不会对输出buffer进行清空操作。如果输出Buffer是复用的，那么就存在脏数据，在VGS操作区域范围外的数据就可能是绿或者花的。建议输出Buffer在使用前进行清空操作。

7.2.3 ARGB1555/ARGB8888 图像处理与OSD贴图

- 若待处理图像数据排布是小端 ARGB1555/ARGB8888，则需将输入的 VIDEO_FRAME_S#enPixelFormat 设置为 RK_FMT_BGRA5551/RK_FMT_BGRA8888。
- 使用 ARGB1555 格式进行 OSD 贴图时，OSD 填充数据统一按 ARGB1555 小端排布方式，VGS_ADD_OSD_S#enPixelFmt 配置为 RK_FMT_BGRA5551。

7.2.4 ARGB1555 格式OSD叠加时未达到期望的透明度

可通过 VGS_ADD_OSD_S 结构体中 u32BgAlpha 和 u32FgAlpha 进行OSD的透明度控制。

其中 ARGB1555 格式的OSD 合成公式如下
osd图像的a位为0时 目标rgb值 = u32BgAlpha * 前景图像的rgb + (255 - u32BgAlpha) * 背景图像的rgb值，目标a值 = 保留背景图像的a值。
osd图像的a位为1时 目标rgb值 = u32FgAlpha * 前景图像的rgb + (255 - u32FgAlpha) * 背景图像的rgb值，目标a值 = u32FgAlpha。

比如实现OSD图像的背景部分透明，前景部分不透明。可将OSD的u32BgAlpha设置为0，u32FgAlpha设置为255，同时将背景部分的a值填充为0，前景部分的a值填充为1。其他情况可依据合成公式进行前景和背景的a设置。

8. TDE

8.1 版本和调试信息

8.1.1 版本信息

与TDE功能相关的的库 包括librockit.so 和 librga.so。可以通过 dumpsys --version命令一键获取相关版本号。

log中相关版本信息如下：

```
rockit version: git-9625ad8 Tue Sep 28 19:16:44 2021 +0800
rockit building: built-daw 2021-09-29 11:31:52
Rga built version:1.04 9e331f5+2021-04-12 14:19:55
```

8.1.2 调试信息获取

通过dumpsys tde 获取调试信息， 调试信息包含输入和输出图像的宽高 format、当前和历史处理的job和task的信息、task处理耗时等信息。具体的调试信息参考《Rockchip_Developer_Guide_MPI_DUMP》文档TDE章节。

8.1.3 TDE支持的规格

TDE支持常见的图像格式如YUV420、RGB888、RGBA8888等, 更多支持规格参考《Rockchip_Developer_Guide_MPI_TDE》文档。

8.2 问题调试

8.2.1 多次调用TDE失败

- TDE 多次调用begin 和end job，只有第一次调用成功，后面几次都会失败，出现如下报错。

```
Try to use uninit rgaCtx=(nil)
```

- TDE调用RGA 相关接口没有初始化导致的。 问题在新版本的rga库上面已经解决，需要更新librga.so。

8.2.2 外部源数据使用TDE处理出现绿边

Buffer的数据存在虚宽高，如果没有设置虚宽高的话，使用的是图像实际的宽高作为虚宽高，导致硬件取数出现偏差。可以通过RK_MPI_CAL_XXX_GetPicBufferSize（XXX为具体模块名）传入实际的图像宽高获取到具体模块对齐后的虚宽高，再将虚宽高转换为字节为单位的Stride，通过RK_MPI_MB_SetBufferStride 设置到MB_BLK中。目前MPI模块(如VI/VDEC/VPSS)产生的图像数据送入到TDE，TDE模块可以获取到虚宽高，不需要设置Stride，外部图像数据源根据需要设置对应的Stride。

```
PIC_BUF_ATTR_S stPicBufAttr;
MB_PIC_CAL_S stMbPicCalResult;
TDE_SURFACE_S stSrcSurface, stDstSurface;

stPicBufAttr.u32Width = stSrcSurface.u32Width;
stPicBufAttr.u32Height = stSrcSurface.u32Height;
stPicBufAttr.enPixelFormat = stSrcSurface.enColorFmt;
stPicBufAttr.enCompMode = COMPRESS_MODE_NONE;
RK_MPI_CAL_TDE_GetPicBufferSize(&stPicBufAttr, &stMbPicCalResult);
RK_U32 u32HorStride = RK_MPI_CAL_COMM_GetHorStride(stMbPicCalResult.u32VirWidth,
stSrcSurface.enColorFmt);
RK_U32 u32VerStride = stMbPicCalResult.u32VirHeight;
RK_MPI_MB_SetBufferStride(stDstSurface.pMbBlk, u32HorStride, u32VerStride);
```

8.2.3 缩放或者格式转换出现绿屏

- 确认需要处理的格式是不是在模块支持的列表中。
- 确认输入和输出图像的实宽高和虚宽高是否符合对齐的要求。

8.2.4 压缩图像拷贝出现绿屏

压缩图像的拷贝需要调用者知道拷贝的大小，然后根据大小转换成非压缩图像的拷贝，实际使用的是TDE单纯的内存拷贝功能。设置的参数为拷贝的大小转换后的非压缩格式宽高，建议图像格式设置为RGBA8888。

8.2.5 YUV400 格式支持

目前 TDE 支持 yuv400 数据的输入和输出，但由于硬件的限制，YUV400 作为输入格式时，建议 Buffer 大小按 YUV420 大小分配，否则内部处理前会进行一次输入 Buffer 的扩大和拷贝。

8.2.6 拷贝是否支持输入输出为同一块内存

TDE 支持同一块内存中的拷贝。内存的偏移可通过 TDE_RECT_S 结构体中的参数 s32Xpos 和 s32Ypos 进行设置。

如下假定pstVideoFrame为其他模块取出的图像，其大小为 7680x1000，通过如下设置，将 x/y 方向偏移为 (5760, 0) 、宽高为 (1920x1000) 的部分图像拷贝到 x/y 方向偏移为 (0, 0) 的位置。

```
VIDEO_FRAME_INFO_S pstVideoFrame;

pstSrc.pMbBlk = pstVideoFrame->stVFrame.pMbBlk;
pstSrc.u32Width = 7680;
pstSrc.u32Height = 1000;
pstSrc.enColorFmt = pstVideoFrame->stVFrame.enPixelFormat;
pstSrcRect.s32Xpos = 0;
pstSrcRect.s32Ypos = 0;
pstSrcRect.u32Width = 1920;
pstSrcRect.u32Height = 1000;

pstDst.pMbBlk = pstVideoFrame->stVFrame.pMbBlk;
pstDst.u32Width = 7680;
pstDst.u32Height = 1000;
pstDst.enColorFmt = pstVideoFrame->stVFrame.enPixelFormat;
pstDstRect.s32Xpos = 5760;
pstDstRect.s32Ypos = 0;
pstDstRect.u32Width = 1920;
pstDstRect.u32Height = 1000;
```

8.2.7 调用 RK_TDE_EndJob 出现段错误

当调用 RK_TDE_EndJob 必现段错误崩溃时，需检查通过 TDE_SURFACE_S 结构体设置进来的输入和输出 pMbBlk 参数值是否是从 MPI 其他模块获取的或者通过 MPI 提供的 RK_MPI_MB_XXX 等接口构造的，不能直接通过 malloc 申请一个 Buffer 赋值给 pMbBlk，否则 TDE 内部会访问到一个非法的内存而崩溃。

8.2.8 调用 RK_TDE_EndJob 出现报错

- 确认返回的错误码。如报错 0xA00E8001 表示非法的句柄，可检查传入到 endjob 中的 handle 是不是已经被占用或者被 cancel。
- 确认是否有异常的日志输出。如 err ws[100,1280,1280]、Error srcRect 等。可通过dumpsys tde来确认处理的各项参数是否符合硬件的要求，如对齐要求、格式是否支持等。
- 在保证参数都符合要求的情况下，确认输入和输出的内存是否正常，如输出内存是否分配足够、输入的内存是否非法等。

8.3 性能相关

8.3.1 TDE处理耗时分析

TDE处理耗时跟图像大小、硬件模块的频率、DDR带宽频率和使用率有关系。

- 确认TDE处理耗时的时间

通过dumpsys tde 中cost_time 来查看最近几次调用TDE所使用的时间。
打印如下：

```
seq_no    job_hdl   state     task_num  in_size   out_size  cost_time hw_time
0         0         proced    1         345600    921600    2813      0
```

通过如下命令 来确认底层硬件实际使用的时间

```
echo time > /sys/kernel/debug/rga2_debug/rga2
dmesg
```

log如下：

```
[668612.190859] rga2: sync one cmd end time 709
```

- 确认RGA硬件模块的频率

通过如下命令可以 确认RGA运行的频率 和 修改RGA运行的频率。

```
cat /sys/kernel/debug/clk/clk_summary | grep rga  //查看rga频率
echo 400000000 > /sys/kernel/debug/clk/aclk_rga/clk_rate // 修改rga频率
```

- 确认DDR的频率

通过如下命令可以确认和修改DDR的频率

```
cat /sys/kernel/debug/clk/clk_summary | grep ddr

查看DDR频率的可设置范围
cat /sys/class/devfreq/dmc/available_frequencies
设置DDR governors为userspace模式
echo userspace > /sys/class/devfreq/dmc/governor
设置DDR频率
echo 1560000000 > /sys/class/devfreq/dmc/userspace/set_freq
```

- 确认DDR的带宽使用率

通过DDR带宽使用来确认当前的带宽资源是否比较紧张。当带宽资源紧张时，RGA硬件模块访问DDR效率降低，从而导致TDE处理的性能下降。

正常情况下TDE处理一张1080p的图像耗时在4-5ms之间，如果耗时明显多于4-5ms。 首先确认下RGA硬件的频率。RGA正常运行的频率在300M左右，如果明显低于这个频率，可以通过手动修改RGA频率来确认问题。RGA运行频率正常的情况下，确认DDR的运行频率，如果频率明显低于预期频率，可以通过手动修改DDR频率来确认。如果频率都正常的话，需要跑DDR带宽使用率的脚本，确认各个模块的DDR使用率，结合应用业务优化DDR带宽使用率。

9. AUDIO

9.1 AO/AI对接调试步骤

在对接AO/AI之前，先确定SDK默认挂载的声卡驱动是否满足产品场景需求，根据对接客户的经验，一般SDK默认挂载的声卡是不满足的，需要根据产品硬件图调试对接音频驱动。
以具体SDK注册的声卡为例，按以下步骤进行声卡调试。

9.1.1 首先通过命令查看音频驱动注册的声卡

查看注册声卡命令	描述
cat proc/asound/cards	[root@RK356X:/]# cat proc/asound/cards 0 [rockchiphdmi]: rockchip_hdmi - rockchip,hdmi rockchip,hdmi 1 [rockchiprk809co]: rockchip_rk809- - rockchip,rk809-codec rockchip,rk809-codec 2 [ROCKCHIPSPDIF]: ROCKCHIP_SPDIF - ROCKCHIP,SPDIF ROCKCHIP,SPDIF

如果声卡初始化成功，那么通过如上命令，可以看到注册成功后的声卡信息。反之，如果通过如上命令看不到声卡，则说明声卡驱动注册失败，需要检查声卡驱动。

9.1.2 用命令行测试声卡播放和录音功能

- 播放测试

使用aplay命令播放wav文件，测试下声卡的播放功能。

注意：某些codec，比如rk809需要先使用amixer配置Playback Path，以配置codec内部的播放通路。

测试hdmi播放：这里是播放声卡0（rockchiphdmi）：

命令行播放	描述	参数描述
aplay -Dhw:0,0 /userdata/t48.wav	[root@RK356X:/]# aplay -Dhw:0,0 /userdata/t48.wav Playing WAVE '/userdata/t48.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Stereo	-Dhw:**x**,0 : **x**代表的声卡序号

测试rk809播放：需要先设置amixer，再使用aplay播放。这里rk809是声卡1（rockchiprk809co）：

命令行播放	描述	参数描述
amixer cset -c 1 name="Playback Path" "SPK_HP" (非rk809codec不需要设置)	[root@RK356X:/]# amixer cset -c 1 name="Playback Path" "SPK_HP" numid=1,iface=MIXER,name='Playback Path' ;type=ENUMERATED,access=rw----- -,values=1,items=11 ; Item #0 'OFF' ; Item #1 'RCV' ; Item #2 'SPK' ; Item #3 'HP' ; Item #4 'HP_NO_MIC' ; Item #5 'BT' ; Item #6 'SPK_HP' ; Item #7 'RING_SPK' ; Item #8 'RING_HP' ; Item #9 'RING_HP_NO_MIC' ; Item #10 'RING_SPK_HP' : values=6	-c **x** : **x**代表的声卡序号; values=6 : 表示Playback Path设置为SPK_HP
aplay -Dhw:1,0 /userdata/t48.wav	[root@RK356X:/]# aplay -Dhw:1,0 /userdata/t48.wav Playing WAVE '/userdata/t48.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Stereo	-Dhw:**x**,0 : **x**代表的声卡序号

- 录音测试

录音：使用arecord命令录制wav文件，测试声卡的录音功能。

注意：某些codec，比如rk809需要先使用amixer配置Capture MIC Path，以配置codec内部的录音通路。

测试rk809录音，需要先设置amixer，再使用arecord播放。这里是录制声卡1（rockchiprk809co）：

命令行播放	描述	参数描述
amixer cset -c 1 name="Capture MIC Path" "Main Mic" (非rk809codec不需要设置)	[root@RK356X:/]# amixer cset -c 1 name="Capture MIC Path" "Main Mic" numid=2,iface=MIXER,name='Capture MIC Path' ; type=ENUMERATED,access=rw-----,values=1,items=4 ; Item #0 'MIC OFF' ; Item #1 'Main Mic' ; Item #2 'Hands Free Mic' ; Item #2 'Hands Free Mic' ; Item #3 'BT Sco Mic' : values=1	-c **x** : **x**代表的声卡序号; values=1 : 表示Capture MIC Path设置为Main Mic
arecord -Dhw:1,0 -r 16000 -c 2 -f s16_le /userdata/test.wav	[root@RK356X:/]# arecord -Dhw:1,0 -r 16000 -c 2 -f s16_le /userdata/test.wav Recording WAVE '/userdata/test.wav' : Signed 16 bit Little Endian, Rate 16000 Hz, Stereo	-Dhw:**x**,0 : **x**代表的声卡序号 -r sample rate（采样率） -c channels（声道数） -f format（格式）

9.1.3 AO/AI软件接口对接

声卡驱动播放和录音功能调试正常后，再对接AO/AI软件接口。可以先用公版的测试sample编译出来的bin文件（sample源码：test_mpi_ao.cpp/test_mpi_ai.cpp）进行验证测试。下面是常用参数测试用例：

AO/AI	测试参数	参数描述
AO	./rk_mpi_ao_test -i 44100_2.pcm --device_rate 48000 --device_ch 2 --input_ch 2 --input_rate 44100 --sound_card_name hw:1,0	i : 输入文件 device_rate ： 播放声卡采样率 device_ch ： 播放声卡声道数 input_rate ： 播放数据的采样率 input_ch ： 播放数据的声道数 sound_card_name ： 声卡名称（可以是实际物理声卡或者asound.conf的虚拟声卡，不配置默认打开pcm.card0）
AI	./rk_mpi_ai_test --device_rate 16000 --device_ch 2 --out_rate 16000 --out_ch 2 --sound_card_name hw:1,0 -o ./ai	o : 输出文件夹 device_rate ： 打开录制声卡采样率 device_ch ： 打开录制声卡声道数 out_rate ： 录制数据的采样率 out_ch ： 录制数据的声道数 sound_card_name ： 打开声卡（可以是实际物理声卡或者asound.conf的虚拟声卡，不配置默认打开pcm.record0）

Linux版本sdk，音频MPI（AO/AI）基于alsa框架，可以通过下面官方网站了解更多。

描述	网址
Alsa项目的官方网址	http://www.alsa-project.org/
Alsa LIB API Reference	http://www.alsa-project.org/alsa-doc/alsa-lib/
配置文件的语法	http://www.alsa-project.org/alsa-doc/alsa-lib/conf.html
Asoundrc的官方说明文档	http://www.alsa-project.org/main/index.PHP/Asoundrc

AO/AI中 AUDIO_DEV与声卡的映射：

AO/AI使用的虚拟声卡名	描述
pcm.cardx	x表示AoDevId，其取值范围为0~AO_DEV_MAX_NUM
pcm.recordx	x表示AiDevId，其取值范围为0~AI_DEV_MAX_NUM

以AUDIO_DEV AoDevId = 0为例，asound.conf的定义如下：

```
pcm.card0 {
    type asym
    playback.pcm "softvol_play0"
}

pcm.softvol_play0 {
    type softvol
    slave.pcm "dmixer0"
    control {
        name "MasterP Volume"
        card 0
        device 0
    }
    min_dB -40.0
    max_dB -1.8
    resolution 100
}

pcm.dmixer0 {
    type dmix
    ipc_key 5978293 # must be unique for all dmix plugins!!!!
    ipc_key_add_uid yes
    slave {
        pcm "hw:0,0"
    }
}
```

当应用使用 AO AoDevId = 0 作为输出时，AO 会打开虚拟声卡 pcm.card0，并通过 aound.conf 中的定义，逐级打开下一级节点，直至打开真正的物理声卡hw:0,0。如上，pcm.card0 总的输出通路：

```
pcm.card0-->softvol_play0-->dmixer0-->hw:0,0
```

声卡节点	描述
pcm.card0	AO 使用的虚拟声卡名。AO AoDevId=0 时打开声卡
pcm.softvol_play0	alsa标准的softvol插件，用于音量流的音量控制。
pcm.dmixer0	alsa标准的混音插件。用于一个物理声卡上，不同音频流(AoChn)的混音成一路音频输出
"hw:**c**,**d**"	为实际的物理声卡。 **c**：card（可以通过查看声卡确认：cat proc/asound/cards） **d**：device（一般为0） (比如上面的"hw:0,0"表示实际的声卡0（rockchiphdmi）)

- 客户可按照自己的需求，在asound.conf中增加或者减少alsa的插件，以满足自己特定要求。
- 如客户可修改映射关系，比如修改最后一级虚拟声卡"pcm.dmixer0"中的"hw:0,0"修改为"hw:1,0"，那么AO(AoDevId=0)映射的是codec(hw:1,0)。

AI对接情况同理，这里不再赘述。

9.1.4 AO/AI打开声卡的两种方式

- **MPI接口中不配置声卡名**，使用 AUDIO_DEV 映射。如上 AUDIO_DEV=0 的 AO, 会固定打开 asound.conf 中 pcm.card0 这个虚拟声卡（AI固定是pcm.record0），然后数据传递 softvol_play0，再传递给 dmixer0，最后写入 hw:0,0。同理 AUDIO_DEV=1，就是打开 pcm.card1。
  AUDIO_DEV 与 asound.conf 中映射函数如下：

AO/AI	**devId**	固定打开虚拟声卡
AO	AUDIO_DEV=**x**	pcm.card**x**
AI	AUDIO_DEV=**x**	pcm.record**x**

- **MPI接口中配置声卡名**。以 AO 为例，参考测试 Sample中 接口：

```
if (ctx->chCardName) {
    snprintf(reinterpret_cast<char *>(aoAttr.u8CardName),
                sizeof(aoAttr.u8CardName), "%s", ctx->chCardName);
}
```

这种方式打开声卡比较灵活，不受AUDIO_DEV和asound.conf中映射的限制。这里配置的声卡名，可以是任意在asound.conf中定义的声卡名，也可以直接是物理声卡"hw:**c**,**d**"。
比如AUDIO_DEV=0的AO，打开asound.conf中定义的虚拟声卡：pcm.card0，

```
snprintf(reinterpret_cast<char *>(aoAttr.u8CardName), sizeof(aoAttr.u8CardName),
"%s", "pcm.card0");
```

也可以直接打开物理声卡："hw:**c**,**d**"，比如打开物理声卡1："hw:1,0"，

```
snprintf(reinterpret_cast<char *>(aoAttr.u8CardName), sizeof(aoAttr.u8CardName),
"%s", "hw:1,0");
```

AI对接情况同理。

9.2 AO无声或者断音卡顿问题

AO主要有无声和卡顿断音问题。首先确认音频驱动是否正常：使用 aplay 播放 wav 音频是否正常，如果播放无声，需要找硬件驱动工程师确认。

9.2.1 AO无声

- 确认音频驱动是否正常，可以使用 aplay 确认能否正常播放 wav 音频。
- AO日志是否正常，播放过程中声卡是否正常打开：

```
// 查看挂载声卡
cat proc/asound/cards
// 查看场景声卡是否打开
cat proc/asound/cardX/pcm0p/sub0/hw_params /* cardX : X 表示声卡序号 */
```

- 如果声卡打开正常，确认送的pcm数据是否正常，以及音量是否静音。
- 如果声卡close，确认AO接口是否正常对接，可以对比AO测试sample情况。

9.2.2 AO断音卡顿

- **确认上层送数据是否及时**，可以看到是否有underrun的日志，类似如下：

```
card:hw:1,0: underrun
```

- **确认mclk时钟是否正常**，播放的时候打印clk_summary：

```
cat /sys/kernel/debug/clk/clk_summary
```

根据dts配置和硬件图找到对应clk值，查看是否正常。以RK3588 dts配置声卡举例：

```
ALSA device list:
#0: rockchip,hdmi0
#1: rockchip,hdmi1
#2: rockchip,dp-sound1
#3: rockchip,spdif-tx1
#4: rockchip,es8323
```

比如某个声卡(rockchip,spdif-tx1)播放卡顿。确认spdif-tx1的mlck值。如果是mclk_spdif1的频率是12000000是不对的。

```
#cat /sys/kernel/debug/clk/clk_summary
.....
clk_spdif1                 1        1        0    12000000          0     0  50000
  mclk_spdif1              1        2        0    12000000          0     0  50000
.....
```

需要修改对应声卡的dts配置，添加分频：

```
simple-audio-card,mclk-fs = <128>;
```

计算mclk公式: mclk = sample_rate * mclk_fs。

比如打开声卡采样率44100Hz，mclk = 44100 * 128 = 5644800Hz。

修改后mclk打印如下：

```
#cat /sys/kernel/debug/clk/clk_summary
.....
clk_spdif1                1        1        0     5644800          0     0  50000
  mclk_spdif1             1        2        0     5644800          0     0  50000
.....
```

- **如果上述两个点确认没问题，开启xrun的trace event调试信息**，打印hwptr，applptr等，分析buffer生产者和消费者详细的读写信息，方便定位问题。

Trace Event	Description
snd_pcm:applptr	应用指针更新
snd_pcm:hw_ptr_error	dma 指针出错
snd_pcm:xrun	xrun
snd_pcm:hwptr	dma 指针更新

  - 使能 FTRACE CONFIG

```
CONFIG_FUNCTION_TRACER
CONFIG_FUNCTION_GRAPH_TRACER
CONFIG_STACK_TRACER
CONFIG_DYNAMIC_FTRACE
```

  - 打开应用程序播放音频。
  - 执行如下脚本命令(声卡 ID 需要修改成实际调试的声卡)，复现后将 /data/trace_sx.txt 提供给技术支持人员。

```
echo 7 > /proc/asound/card0/pcm0p/xrun_debug  /* card0 ：声卡ID需要修改成  实际调试的声卡 */

cd /sys/kernel/debug/tracing
echo 0 > trace
echo "snd_pcm:applptr" >> set_event
echo "snd_pcm:hwptr" >> set_event
echo "snd_pcm:xrun" >> set_event
echo 1 > tracing_on
cat trace_pipe > /data/trace_sx.txt
```

9.3 AI取帧报错

- 首先确认录音声卡打开声卡是否正常。如果是 close 状态，说明声卡没打开，需要确认应用逻辑。

```
cat proc/asound/cardX/pcm0c/sub0/hw_params /* cardX : X 表示声卡序号 */
```

- 确认声卡正常打开后，可通过 arecord/tinycap 命令确认录音声卡是否可正常录制，若不正常，需要确认声卡驱动，比如测量信号是否正常等。

9.4 ADEC/AENC支持格式

- 已支持的编解码器格式：g711a/g711u/g722/g726/g726le。
  - 由于支持格式的码流没有封装头，编解码内部无法判断码流的正确性，需要应用端保证码流的正确性。
  - g726 格式区分了大小端。RK_AUDIO_ID_ADPCM_G726 表示大端格式，RK_AUDIO_ID_ADPCM_G726LE 表示小端格式。
  - g726/g726le 格式还需要设置u32Bitrate（比特率，见结构体AENC_ATTR_CODEC_S/ADEC_ATTR_CODEC_S）。支持的比特率如下：

```
typedef enum rkAUDIO_G726_BPS_E {
    G726_BPS_16K   = 16000,     // 2bit
    G726_BPS_24K   = 24000,     // 3bit
    G726_BPS_32K   = 32000,     // 4bit
    G726_BPS_40K   = 40000,     // 5bit
    G726_BPS_BUTT,
} AUDIO_G726_BPS;
```

- 通过MPI注册接口扩展自定义编解码器。

10. AVS

10.1 版本和调试信息

10.1.1 版本信息

与AVS功能相关的的库 包括librockit.so、 libpanoStitchApp.so 和 librkgfx_avs.so。可以通过 dumpsys --version命令一键获取相关版本号。

log中相关版本信息如下：

```
---------------- librockit version -----------------
git-3d98064b Wed Mar 30 14:34:10 2022 +0800
built-Col 2022-03-30 14:35:29

---------------- libmali version -----------------
arm_release_ver of this libmali is 'g6p0-01eac0', rk_so_ver is '4'.

---------------- librkgfx_avs version -----------------
v5.2

---------------- libpanoStitchApp version -----------------
Version: 2.8.1.315  date: 2022-03-18 release
```

10.1.2 调试信息获取

通过dumpsys avs 获取调试信息，调试信息包含Group\Channel配置参数等信息。具体的调试信息参考《Rockchip_Developer_Guide_MPI_DUMP》文档 AVS章节。

10.1.3 AVS支持的规格

AVS 支持图像格式yuv420、yuv422，具体支持规格参考《Rockchip_Developer_Guide_MPI_AVS》文档。

11. RGN

11.1 常见问题分析

11.1.1 贴图未生效或显示异常

【现象描述】
在调用 **RK_MPI_RGN_AttachToChn** 后，图像并未成功显示或显示异常。

【分析解决】
未生效的原因可以基本归结于以下几类，需要根据下列项依次确认原因：

- 所贴图模块不支持该类型贴图
  详情请见《Rockchip_Developer_Guide_MPI_RGN_CN》文档中关于模块支持 RGN 的情况。
- Attach chn 失败
  我们可以通过 dumpsys rgn 来观察贴图到通道的情况：

```
---------------------- region chn status of cover --------------------------
-
hdl       type      mod       dev       chn       is_show   x         y
  width     height    color     layer     coord_type
0         1         venc      0         0         true      0         0
  256       256       0xf800    1         ABS
```

上述信息表明了 VENC(0, 0) 上存在 COVER handle 0，则说明通道 Attach 成功，如果有贴图 COVER，但在上述信息未看到相关信息，则可能是贴图失败，需要外部检查参数是否正确。

- 图像数据存放大小端错误，导致贴图图像错误
  RKMPI PIXEL_FORMAT_E 枚举按大端模式定义 (详见《Rockchip_Developer_Guide_MPI_SYS_CN》中数据排布说明)，但 ARGB1555、RGB565 两种格式处理特殊，现做统一约束，要求输入图像为 ARGB1555/RGB565 格式时，图像数据在内存中排列统一按小端存放，对应的 PIXEL_FORMAT_E 枚举值如下：
  - RK_FMT_RGB565：图像内存中数据排布为 BGR565 小端，R在低位。
  - RK_FMT_BGR565：图像内存中数据排布为 RGB565 小端，R在高位。
  - RK_FMT_ARGB1555：图像内存中数据排布为 BGRA5551 小端，A在低位。
  - RK_FMT_BGRA5551：图像内存中数据排布为 ARGB1555 小端，A在高位。
- 图像格式为 RK_FMT_ARGB1555 及 RK_FMT_BGRA5551 时，前背景 alpha 设定错误
  在使用 ARGB1555 格式时， 由于 alpha 值仅为 0 和 1，故我们仅根据 0 和 1 来选择 fgAlpha (前景 alpha) 和 bgAlpha (背景alpha)。
  - 图像的像素 alpha 设定为 0 时，图像根据 bgAlpha 值来确定显示效果，bgAlpha 设定 255 为全不透，0 为全透。
  - 图像的像素 alpha 设定为 1 时，图像根据 fgAlpha 值来确定显示效果，fgAlpha 设定 255 为全不透，0 为全透。

11.1.2 Overlay贴图存在撕裂或花屏

这种情况通常为正在显示的帧被外部写入数据覆盖，导致显示画面不完整。如果是通过 **RK_MPI_RGN_GetCanvasInfo** 和 **RK_MPI_RGN_UpdateCanvas** 来更新贴图，可通过 dumpsys rgn 来进行分析：

```
---------------------- region status of overlay ----------------------------
hdl       type      used      pixel_format    width     height    mb
virt           clut_num
```

- 当 clut_num 等于 1 时，只分配了一块画布 Buffer，更新画布与显示画布是可能同时进行，存在引起撕裂或花屏现象，可配置 OVERLAY_ATTR_S#u32ClutNum 为 2 及以上来解决。
- 当 clut_num 大于等于 2 时，每次更新画布必须完整更新画布的所有内容，否则只会展示更新部分的内容，引起画面花屏。

12. DUMP

12.1 常见问题

12.1.1 dumpsys工具无法使用

dumpsys 命令输出如下错误。

```
connect failed, reason: Connection refused
sendDump: send_message failed
Confirm MPI system has been initialized and running?
-------------------------------------------------------------------------------
DUMP OF SERVICE all:
-------------------------------------------------------------------------------
END DUMP OF SERVICE all:
```

- 确认应用进程已调用RK_MPI_SYS_Init函数进行系统初始化，在系统初始化时，会创建Dumpsys Server来接收命令消息，可通过netstat -nlt来确认是否已初始化成功，确认端口3893连接是否存在，输出如下：

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3893          0.0.0.0:*               LISTEN
```

- 确认应用进程未调用RK_MPI_SYS_Exit，若已调用，将退出Dumpsys Server，无法处理Dumpsys命令消息。
- 确认应用进程处于运行状态，可通过ps命令来确认当前应用是否已退出。
- 确认当前设备是否有127.0.0.1 回环地址，可通过ifconfig来确认，输出如下：

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:756 errors:0 dropped:0 overruns:0 frame:0
          TX packets:756 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:55888 (54.5 KiB)  TX bytes:55888 (54.5 KiB)
```

如果不存在这个配置， 可通过如下命令来手动配置

```
 ifconfig lo 127.0.0.1
```