

RK Rockit MPI Developer Guide

文件标识：RK-KF-YF-576

发布版本：V2.13.3

日期：2024.10

文件密级：绝密 秘密 内部资料 公开

免责声明

本文档按“现状”提供，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，由其各自拥有者所有。

版权所有 © 2024 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。

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

本文档主要介绍MPI API和数据类型。

产品版本

芯片名称	内核版本
RK3506	linux 6.1 / rt-thread 4.1
RK3308	linux 5.10 / rt-thread 4.1
RV1106/RV1103	linux 5.10
RV1106B/RV1103B	linux 5.10

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
V1.1	Aaron.sun	2021-12-11	合并所有的MPI文档
V1.2	Aaron.sun	2021-12-18	更新所有的MPI文档
V1.2.1	qingxu.zhou	2021-12-24	合并所有的MPI文档
V1.2.2	qingxu.zhou	2022-01-06	更新VI模块MPI文档
V1.3.0	qingxu.zhou	2022-01-08	合并所有的MPI文档
V1.3.1	qingxu.zhou	2022-01-22	更新RGN、VENC模块MPI文档
V1.3.2	qingxu.zhou	2022-01-27	更新VPSS、AVS模块MPI文档
V1.4.8	Aaron.sun	2022-03-28	更新所有的MPI文档
V1.4.15	Aaron.sun	2022-05-26	更新所有的MPI文档
V1.4.25	Aaron.sun	2022-10-12	更新所有的MPI文档
V1.4.26	Aaron.sun	2022-10-25	增加VI VENC卷绕配置说明
V1.4.27	Aaron.sun	2022-11-08	增加RGN相关限制的说明
V1.4.44	Aaron.sun	2023-03-24	更新所有的MPI文档
V1.4.56	Aaron.sun	2023-07-27	更新部分MPI文档
V1.4.85	Aaron.sun	2024-04-28	更新部分MPI文档
V2.13.3	Aaron.sun	2024-10-16	ROCKIT-C API说明

总目录

[系统控制](#)

[内存管理](#)

[视频输入](#)

[视频处理子系统](#)

[视频解码](#)

[视频编码](#)

[智能视频监控](#)

[视频输出](#)

[视频图形子系统](#)

[图形处理](#)

[音频子系统](#)

[区域管理](#)

[全景拼接](#)

[DUMP调试信息说明](#)

系统控制

[系统概述](#)

[功能描述](#)

[API 参考](#)

[数据类型](#)

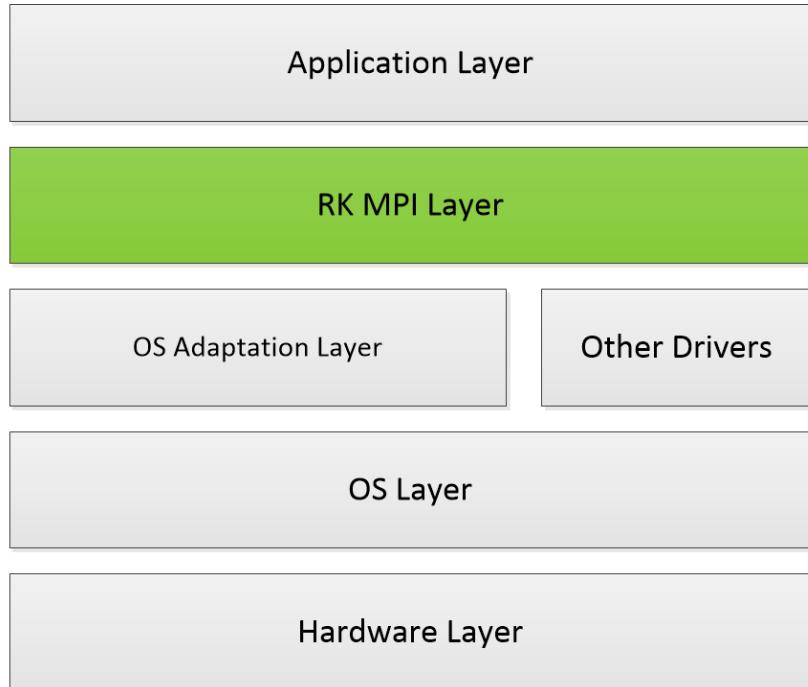
[SYS错误码](#)

1. 系统概述

Rockchip提供的媒体处理接口(Rockchip Media Process Interface,简称 RK MPI)，可支持应用软件快速开发。该平台整合了RK的硬件资源，对应用软件屏蔽了芯片相关的复杂的底层处理，并对应用软件直接提供接口完成相应功能。该平台支持应用软件快速开发以下功能：输入视频捕获、H.265/H.264/JPEG 编码、H.265/H.264/JPEG 解码、视频输出显示、视频图像前处理（包括裁剪、缩放、旋转）、智能、音频捕获及输出、音频编解码等功能。

1.1 系统架构

图1-1 系统架构图



- **应用层**

基于RK MPI及其他驱动，由用户开发的应用软件系统。

- **RK MPI层**

基于适配层（已有的RK芯片适配封装接口），完成媒体处理功能。它对外屏蔽了硬件处理细节，并对应用提供通用化的媒体处理功能接口。

- **操作系统适配层**

基于RK芯片已有的硬件模块对外提供的驱动接口。

- **操作系统层**

基于Linux和Android系统。

- **硬件层**

硬件层由 RK芯片加上必要的外围器件构成。

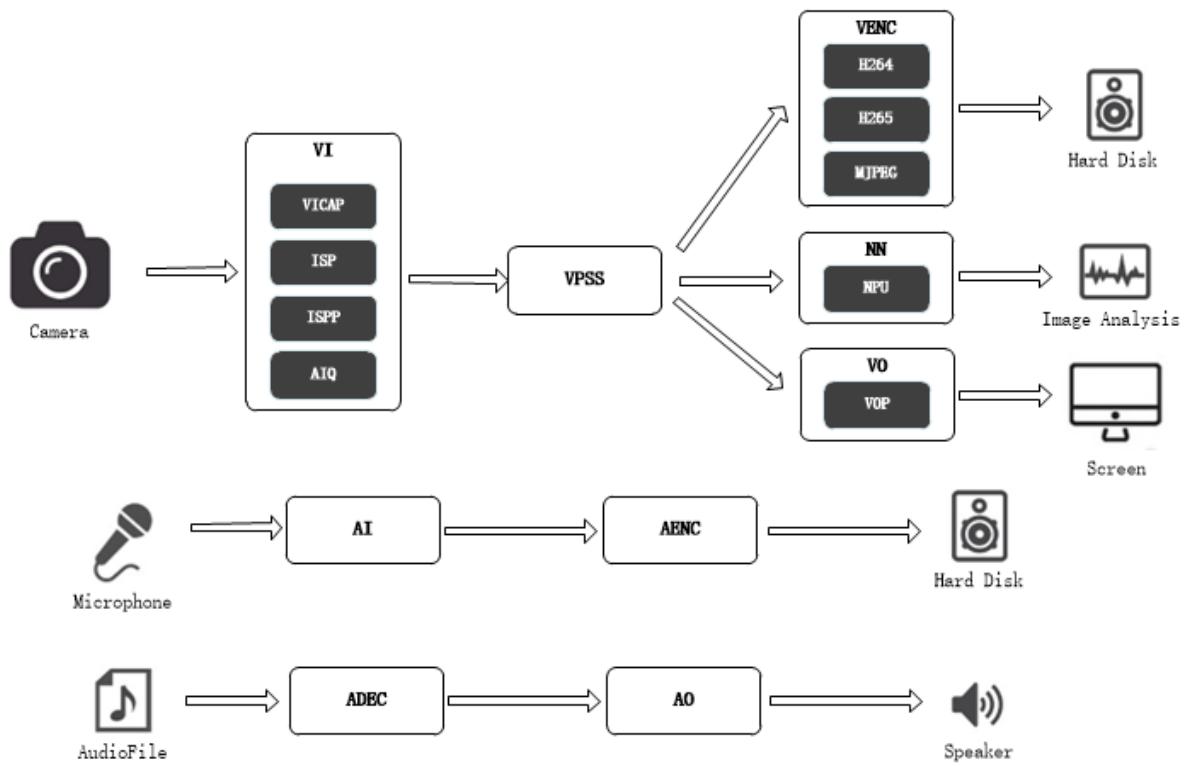
1.2 模块介绍

简称	全称	功能概述
SYS	system	实现RK MPI系统初始化、反初始化、数据流管理及平台性的功能接口。
MB	Media Buffer & Pool	实现通用化内存接口，内存和内存池管理。
VPSS	Video Process Sub-System	视频处理子系统，支持的具体图像处理功能包括FRC、CROP、Scale、像素格式转换、角度旋转、Cover、Overlay、Mosaic、Mirror/Flip、压缩解压等。
VI	Video Input	视频输入（VI）模块实现Sensor、HDMI IN等图像数据采集，VI将接收到的数据存入到指定的内存区域。
VDEC	Video Decoder	提供视频硬件解码接口，实现视频解码功能。
VENC	Video Encoder	提供视频硬件编码接口，实现视频编码功能。
VO	Video Output	模块主动从内存相应位置读取视频和图形数据，并通过相应的显示设备输出视频和图形。
RGN	Region Manager	REGION 模块用于统一管理OSD、遮挡、马赛克、画线等区域资源，将这些区域信息提供至各模块中去。
VGS	Video Graphics System	视频图形系统，由GPU接口实现各种图像处理功能。
TDE	Two Dimensional Engine	二维图像处理引擎，由RGA接口实现2D图像处理功能。
AI	Audio Input	音频采集模块。
ADEC	Audio Decoder	音频解码模块，RK芯片平台通常为软件音频解码。
AENC	Audio Encoder	音频编码模块，RK芯片平台通常为软件音频编码。
AO	Audio Output	音频输出模块，将音频PCM数据输出至各个声卡硬件。
AVS	Any View Stitching	全景拼接，对多路图像进行全景拼接，并且按照指定的投影模式输出图。
PVS	Planar Video Stitching	二维多路图像拼接。

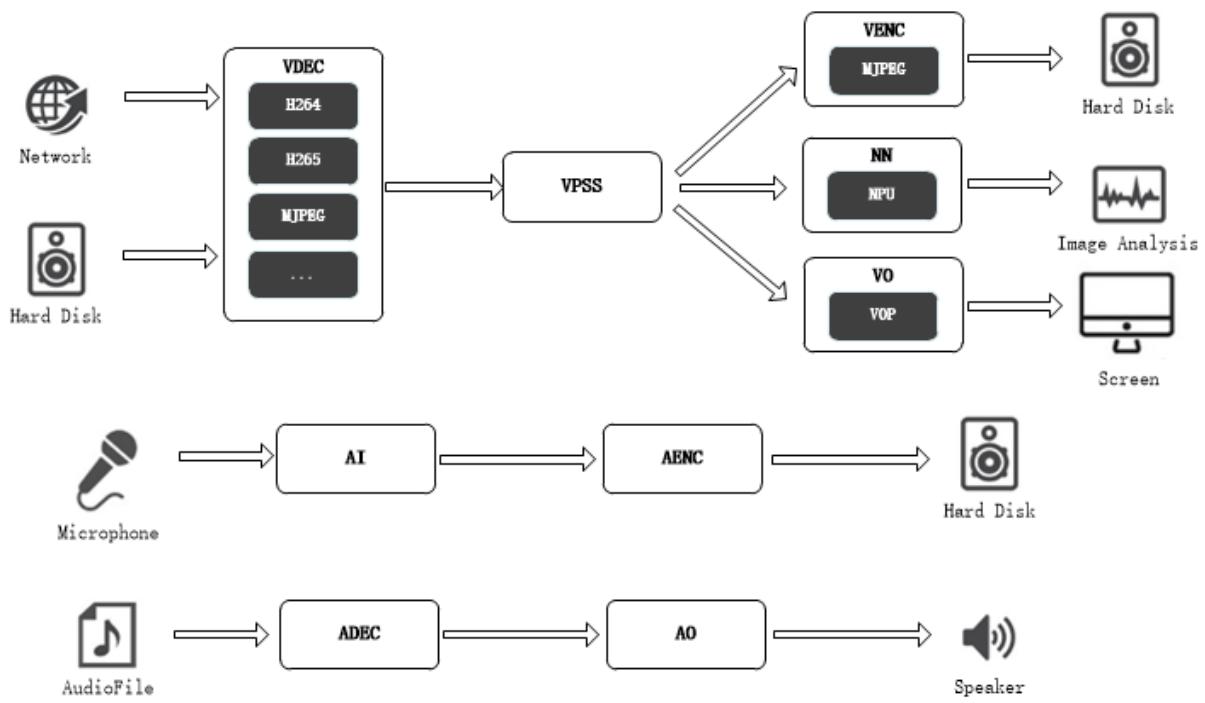
1.3 媒体处理平台典型应用

RK MPI产品的典型应用产品有IPC、NVR、显控等，这里简单描述各种典型产品的使用流程。

1.3.1 IPC典型应用



1.3.2 NVR典型应用



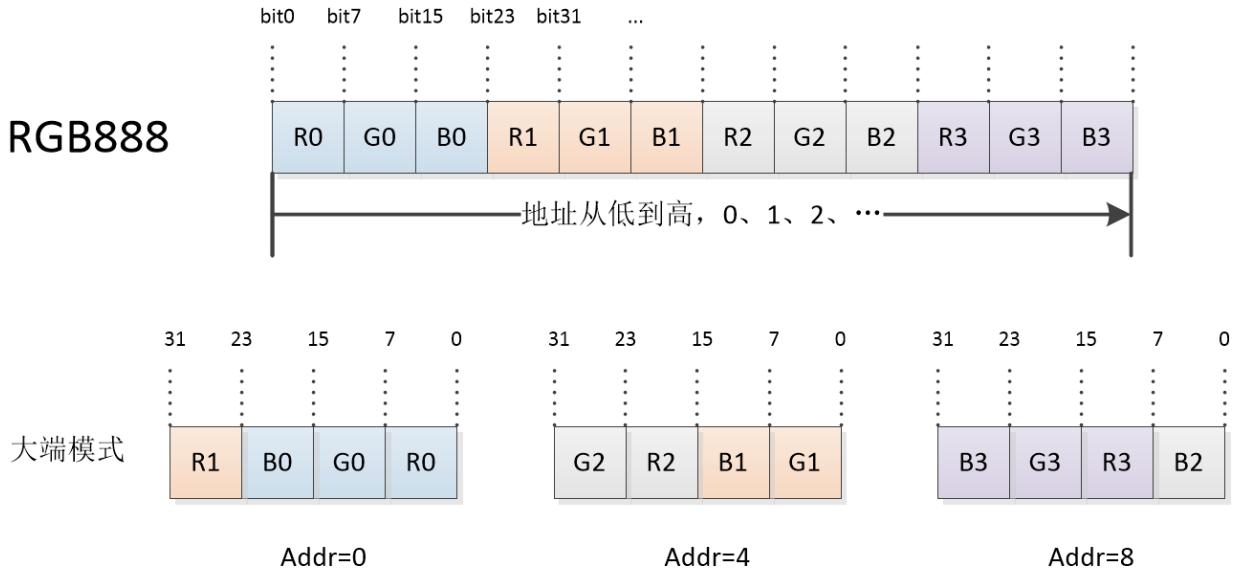
- VI 模块捕获视频图像，可对其做剪切、缩放、镜像等处理，并输出多路不同分辨率的图像数据。
- VDEC 模块对编码后的视频码流进行解码，并将解析后的图像数据送 VPSS 进行图像处理或直接送 VO 显示。可对 H.264/H.265/MJPEG/MPEG2/MPEG4 等多种格式的视频码流进行解码。
- VPSS 模块接收 VI 和解码模块发送过来的图像，可对图像进行裁剪、缩放、旋转、镜像、贴图等处理，并实现同源输出多路不同分辨率的图像数据用于编码、预览、智能分析或抓拍。
- VENC 模块接收 VI 捕获并经 VPSS 处理后输出的图像数据，可叠加用户通过 Region 模块设置的 OSD 图像，然后按不同协议进行编码并输出相应码流。
- VO 模块接收 VPSS 处理后的输出图像，可进行播放控制等处理，最后按用户配置的输出协议输出给外围视频设备。
- AI 模块捕获音频数据，然后 AENC 模块支持按多种音频协议对其进行编码，最后输出音频码流。
- 用户从网络或外围存储设备获取的音频码流可直接送给 ADEC 模块，ADEC 支持解码多种不同的音频格式码流，解码后数据送给 AO 模块即可播放声音。

1.4 概念描述

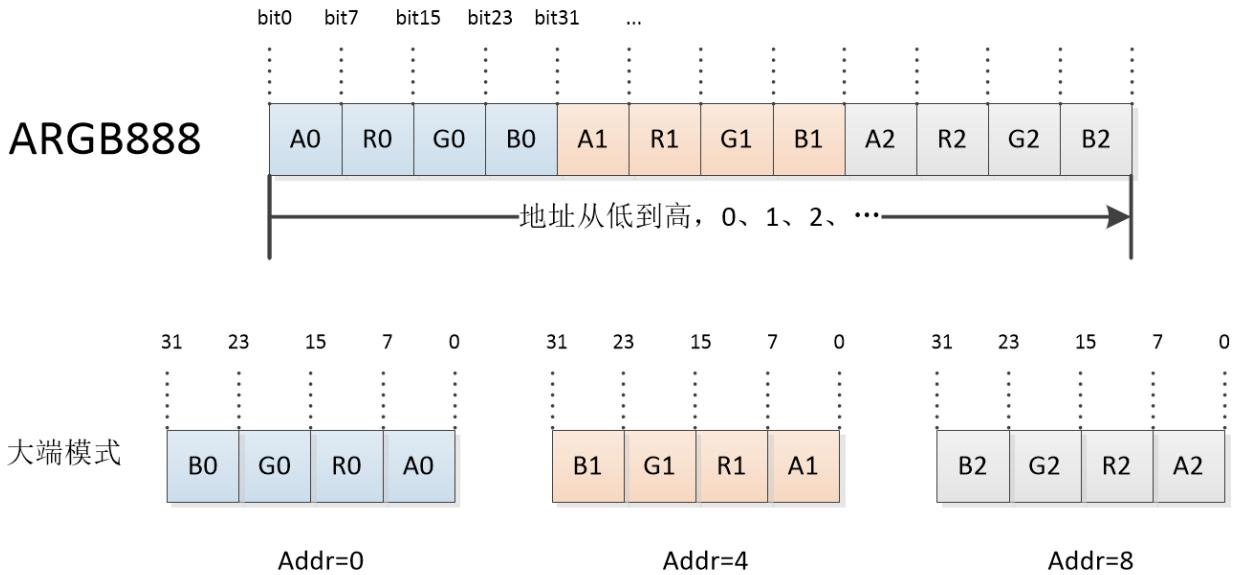
1.4.1 像素格式排布

RKMPI_PIXEL_FORMAT_E 枚举按大端模式定义，即枚举定义代表的数据存储排布是按大端模式排布，以 RGB888/ARGB8888 格式的图像存储为例：

- 图1-2 RGB888格式的图像存储示意图（大端模式）



• 图1-3 ARGB8888格式的图像存储示意图（大端模式）



【注意】

PIXEL_FORMAT_E 枚举按大端模式定义，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：

枚举	存储说明
RK_FMT_RGB565	图像数据在内存中排布为 BGR565 小端，R在低位
RK_FMT_BGR565	图像数据在内存中排布为 RGB565 小端，R在高位
RK_FMT_ARGB1555	图像数据在内存中排布为 BGRA5551 小端，A在低位
RK_FMT_BGRA5551	图像数据在内存中排布为 ARGB1555 小端，A在高位

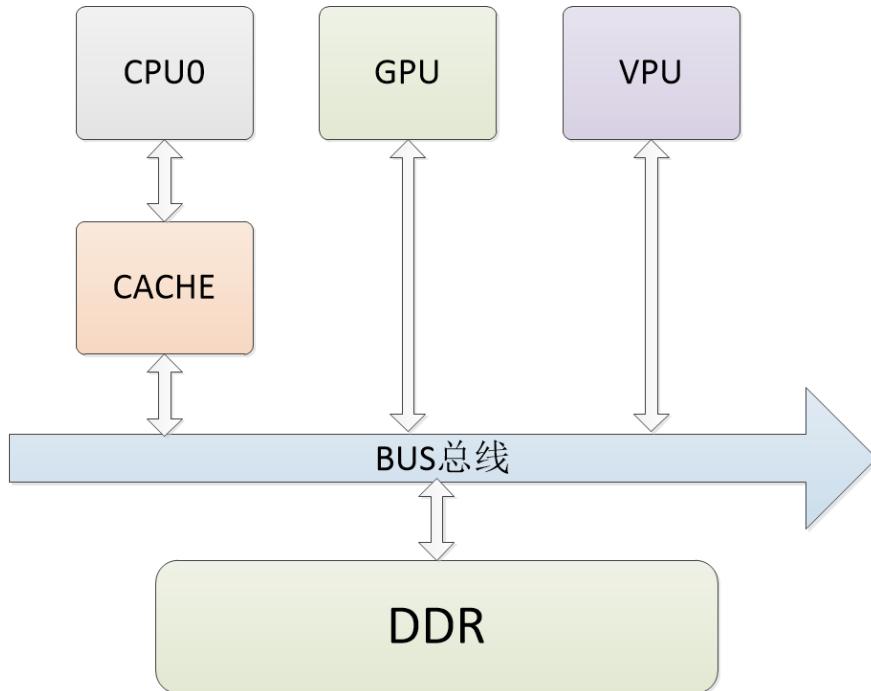
其他RGB格式建议按大端排布，若待处理图像数据是按小端排布，则对应的 PIXEL_FORMAT_E 枚举值也需按照反向配置。如待处理图像为 ARGB8888 小端存储(A在高位)，枚举值要配置为 RK_FMT_BGRA8888。

1.4.2 DMA内存与Cache的一致性

在应用调试及功能实现的过程中，经常会涉及到CPU与GPU、RKVDEC等多个硬件模块对同一个DMA内存数据的读写，此时通常情况会引入cache一致性问题（如果申请的non cache的DMA内存则不存在该问题，但实际上这种内存读写效率较低，在有CPU参与访问数据的情况下，通常不使用）。在对DMA内存读写过程中，通过DDR控制器对DDR存储器进行访问，为了加快访问速度，常常将一些数据缓存在cache中，而且不是针对一个数据缓存，而是一批，这样的好处是下次访问速度会加快，但是坏处也很明显，cache数据发生变化，不能马上反映到DDR中，反之亦然，当通过GPU、RGA等硬件IP修改DDR数据时，CPU可能还不知道发生了什么，拿到的数据还是cache中没有修改的数据，导致读写数据的错误。

CPU与GPU、RGA等硬件IP对DDR数据交互如图所示。

- 图1-4 CPU与GPU、RGA等硬件IP对DDR数据交互图



基于上面所述的cache特点，在使用DMA内存进行数据读写需要满足以下准则即可：

- 当CPU往DMA内存写数据后，若GPU、RGA等其他硬件IP需要访问该DMA内存时，需先调用 [RK_MPI_SYS_MmzFlushCache](#)，将cache写方向flush，将CPU端的Cache数据flush到DDR中。
- 当CPU从DMA内存读数据前，若在此之前有GPU、RGA等其他硬件IP有修改该DMA内存时，若需要先调用 [RK_MPI_SYS_MmzFlushCache](#)，将cache读方向flush，将Cache置为失效，保证与DDR中数据一致性。

2. 功能描述

2.1 内存缓存池

内存缓存池主要向媒体业务提供内存管理功能，负责内存的分配和回收，充分发挥内存缓存池的作用，让内存资源在各个媒体处理模块中合理使用。

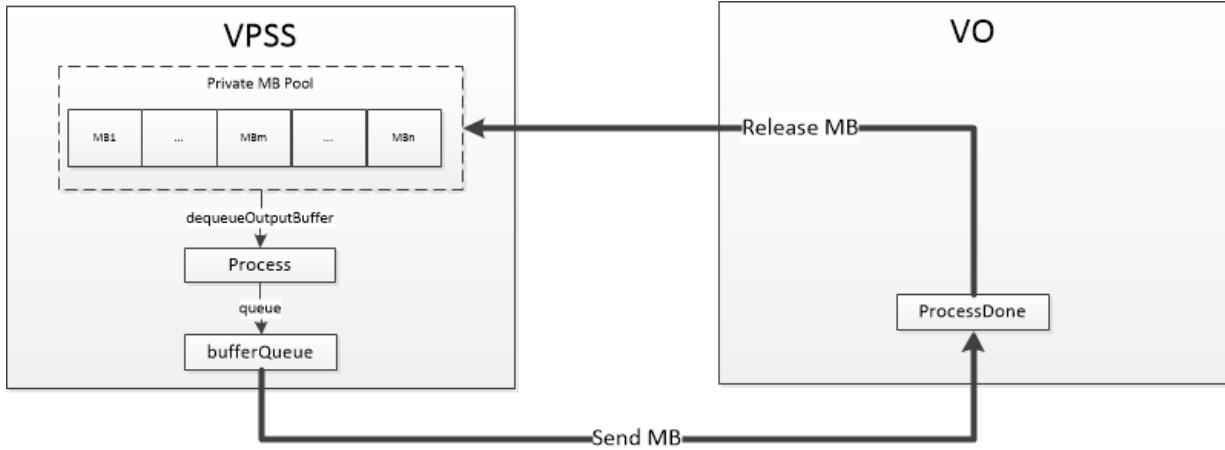
一组大小相同的内存缓存块组成一个内存缓存池。根据业务的不同，申请的缓存池的数量、缓存块的大小和数量不同。

RK提供四种内存缓存池模式（详见[MB_SOURCE_E](#)）：私有缓冲池(MB_SOURCE_PRIVATE)、公有缓冲池(MB_SOURCE_COMMON)、模块缓冲池 (MB_SOURCE_MODULE) 和用户缓冲池 (MB_SOURCE_USER)。并非所有模块都支持这几种模式，详细请看各模块描述，模块中未描述时，默认认为模块使用私有缓冲池。

2.1.1 私有缓冲池

私有缓冲池时，内存缓存池由模块各通道自行申请，申请大小，个数由通道内部根据所设参数及默认参数自行决定。如图2-1所示，VPSS通道内按需分配私有缓冲池，VPSS从私有缓冲池中获取内存块后将目标图像输出至该内存块，送至VO显示，VO显示完毕后释放回私有缓冲池。

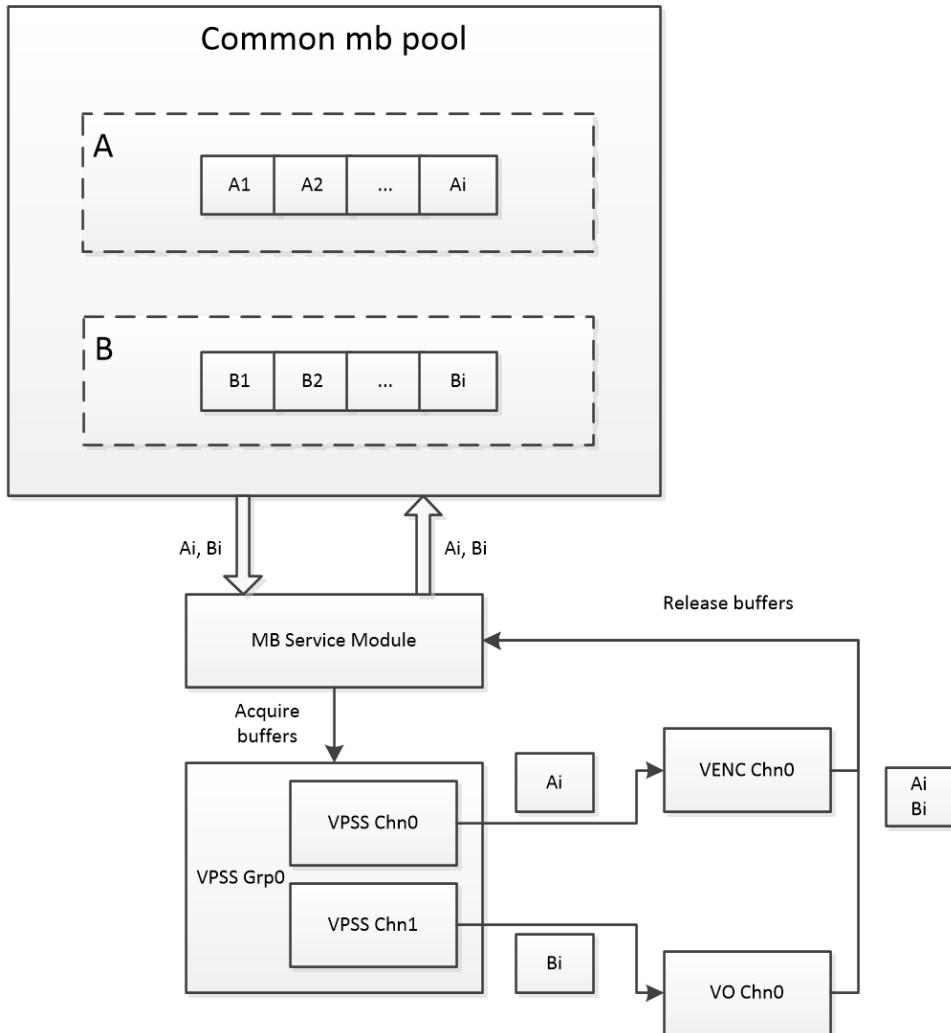
• 图2-1 典型的私有缓冲池数据流图



2.1.2 公共缓冲池

所有的视频输入通道都可以从公共视频缓存池中获取视频缓存块用于保存采集的图像，如图 2-2 中所示，假设VPSS 通道的工作模式是 USER，则 VPSS 通道 0 从公共视频缓存池 A 中获取缓存块 Ai作为输出图像缓存 buffer 发送给 VENC，VPSS 通道 1 从公共视频缓存池 B 中获取缓存块 Bi 作为输出图像缓存 buffer 发送给 VO，Ai 经 VENC 编码完之后释放回公共视频缓存池，Bi 经 VO 显示完之后释放回公共视频缓存池。

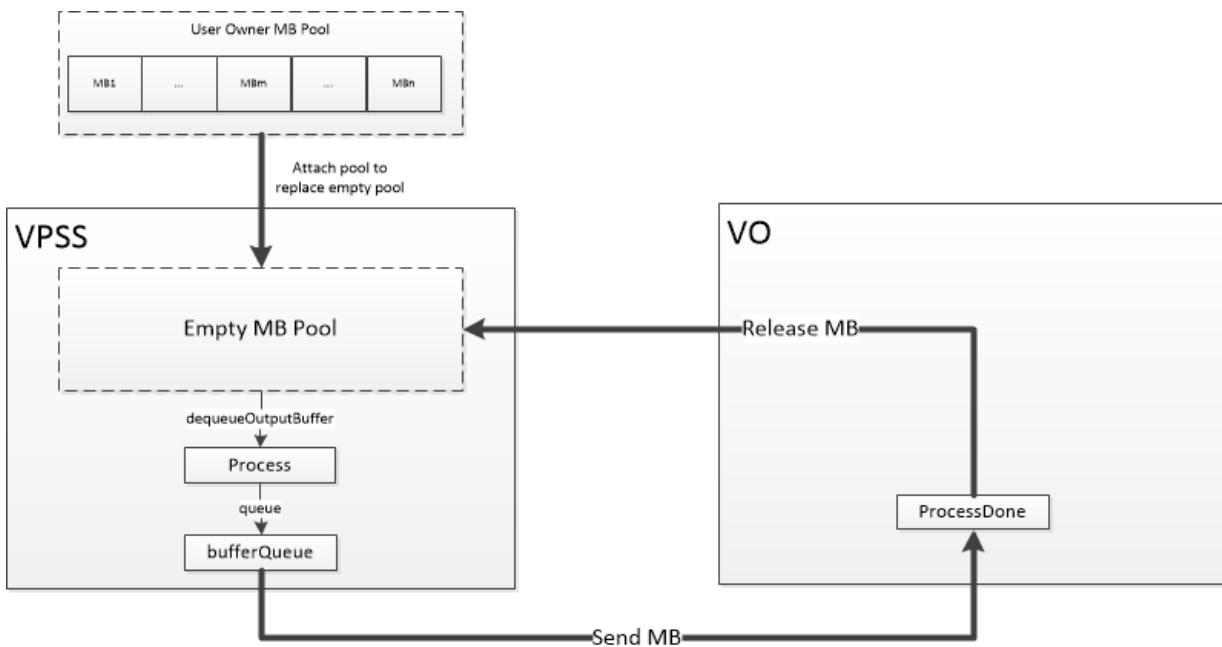
• 图2-2 典型的公共缓冲池数据流图



2.1.3 用户缓冲池

用户内存缓冲池由用户通过模块的attach pool接口注入外部内存缓存池。

- 图2-3 典型的用户缓冲池数据流图



2.1.4 内存池单块MB大小计算规则

为了保证注入的内存池可被正常使用，内存池内存计算需要满足一定规则，直接参考rk_mpi_cal.h。

rk_mpi_cal.h定义了内存大小计算接口，接口介绍如下：

- 表2-1 图像buffer大小及虚宽高计算方法

接口名	用途
RK_MPI_CAL_COMM_GetPicBufferSize	计算通用的图像buffer大小（不被单独列出接口的模块均可以使用该接口确认大小）。
RK_MPI_CAL_TDE_GetPicBufferSize	计算TDE输出所需图像buffer大小。
RK_MPI_CAL_VGS_GetPicBufferSize	计算VGS、VPSS输出所需图像buffer大小。
RK_MPI_CAL_VDEC_GetPicBufferSize	计算VDEC输出所需图像buffer大小。
RK_MPI_CAL_VGS_GetPicVirWidth	获取VGS输出图像像素对齐后的宽度，以像素为单位。
RK_MPI_CAL_VGS_GetPicVirHeight	获取VGS输出图像像素对齐后的高度，以像素为单位。
RK_MPI_CAL_VDEC_GetVirWidth	获取VDEC输出图像像素对齐后的宽度，以像素为单位。
RK_MPI_CAL_VDEC_GetVirHeight	获取VDEC输出图像像素对齐后的高度，以像素为单位。
RK_MPI_CAL_COMM_GetHorStride	实现虚宽(以像素为单位)到Stride(以字节为单位)的转换。
RK_MPI_CAL_COMM_GetVirWidth	实现Stride(以字节为单位)到虚宽(以像素为单位)的转换。

2.2 系统绑定

RK MPI提供系统绑定接口（RK_MPI_SYS_Bind），即通过数据接收者绑定数据源来建立两者之间的关联关系（只允许数据接收者绑定数据源）。绑定后，数据源生成的数据将自动发送给接收者。目前RK MPI支持的绑定关系如下表：

- 表2-2 RK MPI 支持的绑定关系

数据源	数据接收者
VI	VO
VI	VENC
VI	VPSS
VI	AVS
AVS	VENC
AVS	VO
AVS	VPSS
AVS	AVS
VPSS	AVS
VPSS	VO
VPSS	VENC
VPSS	VPSS
VDEC	VPSS
VDEC	VO
VDEC	VENC
WBC	VO
WBC	VENC
WBC	VPSS
VENC	VDEC
AI	AENC
AI	AO
ADEC	AO

2.3 帧率控制

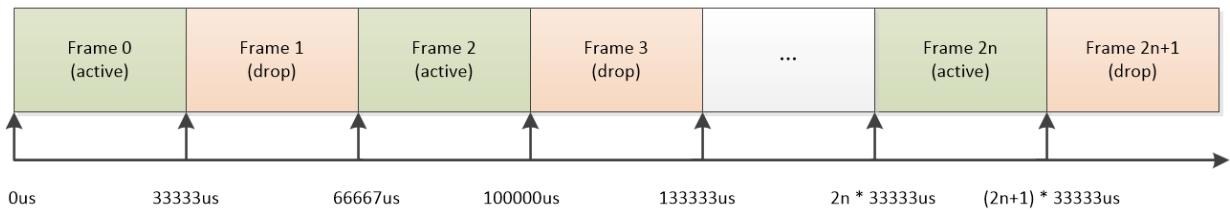
模块帧率控制属性包括输入帧率和输出帧率两部分，同时会根据输入帧率控制是否允许帧输入，例如输入帧率为30时，每33.33ms才允许输入一帧。

为了方便理解，我们通常规定了以下几种模式：

2.3.1 减帧模式

当输入帧率大于或等于输出帧率时，会根据设定值均匀丢弃某些帧。例如，输入帧率为30，输出帧率为15时，每2帧将会丢弃一帧，具体如图所示：

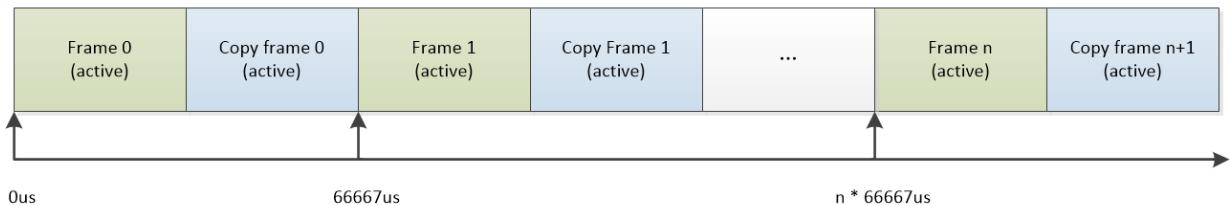
- 图2.4 减帧模式示意图



2.3.2 增帧模式

当输入帧率小于输出帧率时，会根据设定值均匀插入相同图像帧。例如，输入帧率为15，输出帧率为30时，每两帧之间会插入相同的图像帧，具体如图所示：

- 图2.5 增帧模式示意图



2.3.3 帧率控制功能支持列表

- RK356X/RK3588支持情况

模块	增帧模式	减帧模式	送帧控制方式
VI	支持	支持	阻塞式
VENC	支持	支持	阻塞式
VPSS（组）	支持	支持	阻塞式
VPSS（通道）	不支持	支持	非阻塞式
VO	不支持	支持	非阻塞式

3. API 参考

系统控制实现RK MPI系统绑定解绑、创建视频缓存池、提供系统时钟、内存管理等功能。

该功能模块为用户提供以下API：

- [RK_MPI_SYS_Init](#): 初始化RK MPI系统。
- [RK_MPI_SYS_Exit](#): 反初始化RK MPI系统。
- [RK_MPI_SYS_Bind](#): 数据源到数据接收者绑定。
- [RK_MPI_SYS_UnBind](#): 数据源到数据接收者解绑定。
- [RK_MPI_SYS_MmzAlloc](#): 在用户态分配 MMZ 内存。
- [RK_MPI_SYS_MmzAlloc_Cached](#): 在用户态分配 MMZ 内存，该内存支持 cache 缓存。
- [RK_MPI_SYS_MmzAllocEx](#): 在用户态分配 MMZ 内存，可配置内存标志，如是否带cache/是否是物理连续内存。
- [RK_MPI_SYS_MmzFree](#): 在用户态释放 MMZ 内存。
- [RK_MPI_SYS_MmzFlushCache](#): 刷新 cache 里的内容到内存并且使 cache 里的内容无效。
- [RK_MPI_SYS_Malloc](#): 申请malloc内存。
- [RK_MPI_SYS_Free](#): 释放malloc申请的内存。
- [RK_MPI_SYS_CreateMB](#): 创建一个内存缓存快。
- [RK_MPI_SYS_GetCurPTS](#): 获取当前时间戳。

- [RK_MPI_SYS_InitPTSBase](#): 初始化 RK MPI 时间戳。
- [RK_MPI_SYS_SyncPTS](#): 同步 RK MPI 时间戳。
- [RK_MPI_MB_CreatePool](#): 创建一个内存缓存池。
- [RK_MPI_MB_DestroyPool](#): 销毁一个内存缓存池。
- [RK_MPI_MB_GetMB](#): 获取一个缓存块。
- [RK_MPI_MB_ReleaseMB](#): 释放一个已经获取的缓存块。
- [RK_MPI_MB_Handle2PhysAddr](#): 获取一个缓存块的物理地址。
- [RK_MPI_MB_Handle2VirAddr](#): 获取一个内存缓存池中的缓存块的用户态虚拟地址。
- [RK_MPI_MB_Handle2UniqueId](#): 获取一个缓存块的系统全局唯一ID。
- [RK_MPI_MB_Handle2Fd](#): 用户态通过缓存块的句柄。
- [RK_MPI_MB_Handle2PoolId](#): 获取一个缓存块所在缓存池的 ID。
- [RK_MPI_MB.GetSize](#): 获取一个缓存块的内存大小。
- [RK_MPI_MB.GetLength](#): 获取一个缓存块的实际数据大小。
- [RK_MPI_MB.GetOffset](#): 获取一个缓存块的实际内存偏移大小。
- [RK_MPI_MB_SetOffset](#): 设置一个缓存块的内存偏移大小。
- [RK_MPI_MB_VirAddr2Handle](#): 根据虚拟地址获取对应的缓存块。
- [RK_MPI_MB_UniqueId2Fd](#): 根据系统全局唯一ID映射出fd。
- [RK_MPI_MB_InquireUserCnt](#): 查询缓存块使用计数信息。
- [RK_MPI_MB_SetModPoolConfig](#): 设置模块公共视频缓存池属性。
- [RK_MPI_MB_GetModPoolConfig](#): 获取模块公共视频缓存池属性。
- [RK_MPI_MB_SetConfig](#): 设置视频公共缓存池属性。
- [RK_MPI_MB_GetConfig](#): 获取视频公共缓存池属性。
- [RK_MPI_MB_Init](#): 初始化视频公共缓冲池。
- [RK_MPI_MB_Exit](#): 反初始化视频公共缓冲池。
- [RK_MPI_MB_SetBufferStride](#): 设置缓存块用于存储图像时该图像的虚宽高Stride。

3.1 RK_MPI_SYS_Init

【描述】

初始化RK MPI系统。

【语法】

```
RK_S32 RK_MPI_SYS_Init(RK_VOID);
```

【参数】

无

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 运行RK MPI系统前必须调用该接口进行初始化。
- 不建议多次调用该接口, 多次调用仍会返回成功, 需与[RK_MPI_SYS_Exit](#)成对使用。
- 多个进程暂无法共享资源使用各模块, 仅MB模块相关接口可多进程使用。

3.2 RK_MPI_SYS_Exit

【描述】

反初始化RK MPI系统。

【语法】

```
RK_S32 RK_MPI_SYS_Exit(RK_VOID);
```

【参数】

无

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 退出RK MPI系统前必须调用该函数。
- 多次调用仍会返回成功, 需与[RK_MPI_SYS_Init](#)成对使用, 否则会造成泄漏。

3.3 RK_MPI_SYS_Bind

【描述】

数据源到数据接收者绑定的接口。

【语法】

```
RK_S32 RK_MPI_SYS_Bind(const MPP\_CHN\_S pstSrcChn, const MPP\_CHN\_S pstDestChn);
```

【参数】

参数名	描述	输入/输出
pstSrcChn	源通道指针。	输入
pstDestChn	目的通道指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 系统目前支持的绑定关系, 请参见[系统绑定](#)章节。
- 同一个数据接收者只能绑定一个数据源。
- 绑定是指数据源和数据接收者建立关联关系。绑定后, 数据源生成的数据将自动发送给接收者。
- VI 和 VDEC 作为数据源, 是以通道为发送者, 向其他模块发送数据, 用户将设备号置为 0, SDK 不检查输入的设备号。
- VPSS 作为数据接收者时, 是以设备 (GROUP) 为接收者, 接收其他模块发来的数据, 用户将通道号置为 0。
- 模块作为数据发送者时, 没有图像队列深度(DEPTH)设置的模块, 绑定后无法调用自身模块的接口获取图像, 能够设置图像深度的模块, 必须图像深度不为0才可手动获取图像。

3.4 RK_MPI_SYS_UnBind

【描述】

数据源到数据接收者解绑定。

【语法】

```
RK_S32 RK_MPI_SYS_UnBind(const MPP\_CHN\_S pstSrcChn, const MPP\_CHN\_S pstDestChn);
```

【参数】

参数名	描述	输入/输出
pstSrcChn	源通道指针。	输入
pstDestChn	目的通道指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- `pstDestChn`如果找不到绑定的源通道或者找到了绑定的源通道, 但是绑定的源通道和 `pstSrcChn`不匹配, 则返回失败。

3.5 RK_MPI_SYS_MmzAlloc

【描述】

在用户态分配 MMZ 内存。

【语法】

```
RK_S32 RK_MPI_SYS_MmzAlloc(MB\_BLK pBlk, const RK_CHAR*pstrMmb, const RK_CHAR *pstrZone, RK_U32 u32Len);
```

【参数】

参数名	描述	输入/输出
pBlk	缓存块指针。	输出
pstrMmb	Mmb 名称的字符串指针。填写RK_NULL。	输入
pstrZone	MMZ zone 名称的字符串指针。填写RK_NULL。	输入
u32Len	缓存块大小。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 无

3.6 RK_MPI_SYS_MmzAlloc_Cached

【描述】

在用户态分配 MMZ 内存, 该内存支持 cache 缓存。

【语法】

```
RK_S32 RK_MPI_SYS_MmzAlloc_Cached(MB\_BLK pBlk, const RK_CHAR*pstrMmb, const RK_CHAR *pstrZone, RK_U32 u32Len);
```

【参数】

参数名	描述	输入/输出
pBlk	缓存块指针。	输出
pstrMmb	Mmb 名称的字符串指针。填写RK_NULL。	输入
pstrZone	MMZ zone 名称的字符串指针。填写RK_NULL。	输入
u32Len	缓存块大小。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 无

3.7 RK_MPI_SYS_MmzAllocEx

【描述】

在用户态分配 MMZ 内存, 可配置内存标志, 如是否带cache/是否是物理连续内存。

【语法】

```
RK_S32 RK_MPI_SYS_MmzAllocEx(MB\_BLK pBlk, const RK_CHAR*pstrMmb, const RK_CHAR *pstrZone, RK_U32 u32Len, RK_U32 u32HeapFlags);
```

【参数】

参数名	描述	输入/输出
pBlk	缓存块指针。	输出
pstrMmb	Mmb 名称的字符串指针。填写RK_NULL。	输入
pstrZone	MMZ zone 名称的字符串指针。填写RK_NULL。	输入
u32Len	缓存块大小。	输入
u32HeapFlags	内存标志, 可同时带上多个标记。当前支持如下标志: MB_REMAP_MODE_NOCACHE: 不支持Cache缓存 MB_REMAP_MODE_CACHED: 支持Cache缓存 MB_DMA_TYPE_CMA: 带上此标志表示申请物理连续内存, 物理连续内存需要内核提前配置预留媒体专用内存, 当前默认SDK是不支持。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 无

3.8 RK_MPI_SYS_MmzFree

【描述】

在用户态释放 MMZ 内存。

【语法】

```
RK_S32 RK_MPI_SYS_MmzFree(MB\_BLK blk);
```

【参数】

参数名	描述	输入/输出
blk	缓存块ID。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 无

3.9 RK_MPI_SYS_MmzFlushCache

【描述】

刷新 cache 里的内容到内存并且使 cache 里的内容无效。

【语法】

```
RK_S32 RK_MPI_SYS_MmzFlushCache(MB\_BLK blk, RK_BOOL bReadOnly);
```

【参数】

参数名	描述	输入/输出
blk	缓存块ID。	输入
bReadOnly	True: 使 cache 里的内容无效。等同于DMA_BUF_SYNC_READ; False: 将cache内容回写内存并使cache无效。DMA_BUF_SYNC_RW	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 无

3.10 RK_MPI_SYS_Malloc

【描述】

申请malloc内存。

【语法】

```
RK_S32 RK_MPI_SYS_Malloc(MB\_BLK *pBlk, RK_U32 u32Len);
```

【参数】

参数名	描述	输入/输出
pBlk	缓存块指针。	输出
u32Len	缓存块大小。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 无。

3.11 RK_MPI_SYS_Free

【描述】

释放malloc申请的内存。

【语法】

```
RK_S32 RK_MPI_SYS_Free(MB\_BLK blk);
```

【参数】

参数名	描述	输入/输出
blk	缓存块ID。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 无

3.12 RK_MPI_SYS_CreateMB

【描述】

创建一个内存缓存快。

【语法】

```
RK_S32 RK_MPI_SYS_CreateMB(MB\_BLK pBlk, MB\_EXT\_CONFIG\_S*pstMbExtConfig);
```

【参数】

参数名	描述	输入/输出
pBlk	缓存块指针。	输出
pstMbExtConfig	缓存块参数配置信息	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 无

3.13 RK_MPI_SYS_GetCurPTS

【描述】

获取当前时间戳。

【语法】

```
RK_S32 RK_MPI_SYS_GetCurPTS(RK_U64 *pu64CurPTS);
```

【参数】

参数名	描述	输入/输出
pu64CurPTS	当前时间戳指针。单位：微秒。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 SYS错误码 。

【注意】

- 无

3.14 RK_MPI_SYS_InitPTSBase

【描述】

初始化 RK MPI 时间戳。

【语法】

```
RK_S32 RK_MPI_SYS_InitPTSBase(RK_U64 u64PTSBase);
```

【参数】

参数名	描述	输入/输出
u64PTSBase	时间戳基准。单位：微秒。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 SYS错误码 。

【注意】

- 初始化时间戳基准会将当前系统的时间戳强制置成 u64PTSBase，与系统原有时间戳没有任何约束。因此，建议在媒体业务没有启动时（例如操作系统刚启动），调用这个接口。

3.15 RK_MPI_SYS_SyncPTS

【描述】

同步 RK MPI 时间戳。

【语法】

```
RK_S32 RK_MPI_SYS_SyncPTS(RK_U64 u64PTSBase);
```

【参数】

参数名	描述	输入/输出
u64PTSBase	时间戳基准。单位：微秒。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 SYS错误码 。

【注意】

- 无

3.16 RK_MPI_SYS_SetChnInputMode

【描述】

设置通道输入(接收)流模式。用户可根据需要配置通道输入(接收)流模式，可选择保留最新一帧，也可选择丢弃。当用户启用某模块通道时，没有绑定下级模块也没有消耗通道数据，将导致buffer无法正常轮转使用，此时可将此通道输入流模式配置为丢弃模式，保证通道buffer可正常轮转。

【语法】

```
RK_S32 RK_MPI_SYS_SetChnInputMode(const MPP_CHN_S *pstChn, CHN_INPUT_MODE_E mode);
```

【参数】

参数名	描述	输入/输出
pstChn	通道指针	输入
mode	通道输入流模式	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 SYS错误码 。

【注意】

- 无

3.17 RK_MPI_MB_CreatePool

【描述】

创建一个内存缓存池。

【语法】

```
MB_POOL RK_MPI_MB_CreatePool(MB_POOL_CONFIG_S *pstMbPoolCfg);
```

【参数】

参数名	描述	输入/输出
pstMbPoolCfg	缓存池配置属性参数指针。	输入

【返回值】

返回值	描述
非 MB_INVALID_POOLID	有效的缓存池 ID 号。
MB_INVALID_POOLID	创建缓存池失败，可能是参数非法或者保留内存不足。

【注意】

- 无

3.18 RK_MPI_MB_DestroyPool

【描述】

销毁一个内存缓存池。

【语法】

```
RK_S32 RK_MPI_MB_DestroyPool(MB\_POOL pool);
```

【参数】

参数名	描述	输入/输出
pool	缓存池 ID 号。 取值范围：[0, MB_MAX_POOLS]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 SYS错误码 。

【注意】

- GROUP 必须已创建。

3.19 RK_MPI_MB_GetMB

【描述】

获取一个缓存块。

【语法】

```
MB_BLK RK_MPI_MB_GetMB(MB\_POOL pool, RK_U64 u64Size, RK_BOOL block = RK_FALSE);
```

【参数】

参数名	描述	输入/输出
pool	缓存池ID号。 取值范围：[0, MB_MAX_POOLS]。	输入
u64Size	缓存块大小。 取值范围：数据类型全范围，以 byte 为单位。	输入
block	获取缓存块是否等待缓存块使用完毕返回缓存池。 RK_TRUE：阻塞。RK_FALSE：非阻塞。默认非阻塞	输入

【返回值】

返回值	描述
非 MB_INVALID_HANDLE	有效的缓存块句柄。
MB_INVALID_HANDLE	获取缓存块失败。

【注意】

- 无

3.20 RK_MPI_MB_ReleaseMB

【描述】

释放一个已经获取的缓存块。

【语法】

RK_S32 RK_MPI_MB_ReleaseMB([MB_BLK](#) mb);

【参数】

参数名	描述	输入/输出
mb	缓存块ID。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 无

3.21 RK_MPI_MB_Handle2PhysAddr

【描述】

获取一个缓存块的物理地址。

【语法】

RK_U64 RK_MPI_MB_Handle2PhysAddr([MB_BLK](#) mb);

【参数】

参数名	描述	输入/输出
mb	缓存块ID。	输入

【返回值】

返回值	描述
0	失败
非0	物理地址

【注意】

- 默认使用非连续物理内存, 不支持物理地址获取。
- 若要支持物理地址获取, 需预先配置CMA物理连续内存; 在创建MB/MB_POOL时, 需指定类型为 MB_DMA_TYPE_CMA, 详见FAQ配置说明。

3.22 RK_MPI_MB_Handle2VirAddr

【描述】

获取一个内存缓存池中的缓存块的用户态虚拟地址。

【语法】

```
RK_VOID *RK_MPI_MB_Handle2VirAddr(MB\_BLK mb);
```

【参数】

参数名	描述	输入/输出
mb	缓存块ID。	输入

【返回值】

返回值	描述
RK_NULL	获取虚拟地址失败。
非RK_NULL	有效的虚拟地址。

【注意】

- 无

3.23 RK_MPI_MB_Handle2UniqueId

【描述】

获取一个缓存块的系统全局唯一ID，可用于跨进程操作。

【语法】

```
RK_S32 RK_MPI_MB_Handle2UniqueId(MB\_BLK mb);
```

【参数】

参数名	描述	输入/输出
mb	缓存块ID。	输入

【返回值】

返回值	描述
负数	获取系统全局唯一ID失败。
非负数	有效的系统全局唯一ID。

【注意】

- 仅支持MMZ缓存块获取唯一ID。
- 通过[RK_MPI_SYS_CreateMB](#)创建的MB_BLK无法获取系统全局唯一ID。
- 获取的唯一ID可以通过跨进程传输到另一进程中，再通过[RK_MPI_MB_UniqueId2Fd](#)转为fd使用，然后map出虚拟地址进行操作。详细看下列示例。

【示例】

- 源进程示例：

```
MB_BLK MbBlk = MB_INVALID_HANDLE;
RK_S32 s32Ret = RK_SUCCESS;
RK_S32 s32UniqueId = -1;
RK_S32 s32Size = 4 * 1024 * 1024;
```

```

s32Ret = RK_MPI_SYS_MmzAlloc_Cached(&MbBlk, RK_NULL, RK_NULL, s32Size);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}
s32UniqueId = RK_MPI_MB_Handle2UniqueId(MbBlk);
if (s32UniqueId < 0) {
    return RK_FAILURE;
}

// send unique id cross process
s32Ret = send_info_to_dst_process(s32UniqueId, s32Size);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}

```

- 目标进程示例：

```

RK_S32 s32UniqueId = -1;
RK_S32 s32DupFd = -1;
RK_VOID *pVirtAddr = RK_NULL;
RK_S32 s32Ret = RK_SUCCESS;

s32Ret = recv_info_by_src_process(&s32UniqueId, &s32Size);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}

s32DupFd = RK_MPI_MB_UniqueId2Fd(s32UniqueId);
if (s32DupFd < 0) {
    return RK_FAILURE;
}

pVirtAddr = mmap64(NULL, s32Size, PROT_READ | PROT_WRITE, MAP_SHARED, s32DupFd, 0);
if (pVirtAddr == RK_NULL) {
    return RK_FAILURE;
}

```

3.24 RK_MPI_MB_Handle2Fd

【描述】

用户态通过缓存块的句柄。

【语法】

RK_S32 RK_MPI_MB_Handle2Fd([MB_BLK](#) mb);

【参数】

参数名	描述	输入/输出
mb	缓存块ID。	输入

【返回值】

返回值	描述
负数	获取缓存块句柄失败。
非负数	有效的缓存块句柄。

【注意】

- 缓存块句柄只有内存类型为DMA时才可获取有效的句柄。

3.25 RK_MPI_MB_Handle2PoolId

【描述】

获取一个缓存块所在缓存池的 ID。

【语法】

```
MB_POOL RK_MPI_MB_Handle2PoolId(MB\_BLK mb);
```

【参数】

参数名	描述	输入/输出
mb	缓存块ID。	输入

【返回值】

返回值	描述
非MB_INVALID_POOLID	有效的缓存池 ID 号。
MB_INVALID_POOLID	无效的缓存池 ID 号。

【注意】

- 指定的缓存块应该是从RK MPI内存缓存池中获取的有效缓存块，否则无法获取到缓存池ID号。

3.26 RK_MPI_MB_GetSize

【描述】

获取一个缓存块的内存大小。

【语法】

```
RK_U64 RK_MPI_MB_GetSize(MB\_BLK mb);
```

【参数】

参数名	描述	输入/输出
mb	缓存块ID。	输入

【返回值】

返回值	描述
任意值	缓存块的内存大小。

【注意】

- 无

3.27 RK_MPI_MB_GetLength

【描述】

获取一个缓存块的实际数据大小。

【语法】

```
RK_U64 RK_MPI_MB_GetLength(MB\_BLK mb);
```

【参数】

参数名	描述	输入/输出
mb	缓存块ID。	输入

【返回值】

返回值	描述
任意值	缓存块的实际数据大小。

【注意】

- 无

3.28 RK_MPI_MB_GetOffset

【描述】

获取一个缓存块的实际内存偏移大小。

【语法】

```
RK_U32 RK_MPI_MB_GetOffset(MB\_BLK mb);
```

【参数】

参数名	描述	输入/输出
mb	缓存块ID。	输入

【返回值】

返回值	描述
非负数	缓存块的实际内存偏移大小。

【注意】

- 无

3.29 RK_MPI_MB_SetOffset

【描述】

设置一个缓存块的内存偏移大小。

【语法】

```
RK_S32 RK_MPI_MB_SetOffset(MB\_BLK mb, RK_U32 u32Offset);
```

【参数】

参数名	描述	输入/输出
mb	缓存块ID。	输入
u32Offset	内存偏移大小。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 无

3.30 RK_MPI_MB_VirAddr2Handle

【描述】

根据虚拟地址获取对应的缓存块。

【语法】

```
MB_BLK RK_MPI_MB_VirAddr2Handle(RK_VOID *pstVirAddr);
```

【参数】

参数名	描述	输入/输出
pstVirAddr	虚拟地址。	输入

【返回值】

返回值	描述
MB_INVALID_HANDLE	获取缓存块ID失败。
非MB_INVALID_HANDLE	有效的缓存块ID。

【注意】

- 目前仅支持DMA内存从虚拟地址查找到缓存块。

3.31 RK_MPI_MB_UniqueId2Fd

【描述】

获取一个缓存块的系统全局唯一ID。

【语法】

```
RK_S32 RK_MPI_MB_UniqueId2Fd(RK_S32 s32UniqueId);
```

【参数】

参数名	描述	输入/输出
s32UniqueId	系统全局唯一ID。	输入

【返回值】

返回值	描述
负值	获取句柄失败。
非负值	有效的句柄。

【注意】

- 无

3.32 RK_MPI_MB_InquireUserCnt

【描述】

查询缓存块使用计数信息。

【语法】

```
RK_S32 RK_MPI_MB_InquireUserCnt(MB\_BLK mb);
```

【参数】

参数名	描述	输入/输出
mb	缓存块ID。	输入

【返回值】

返回值	描述
非负数	获取缓存块的使用计数。
负数	无效的缓存块计数。

【注意】

- 无

3.33 RK_MPI_MB_SetModPoolConfig

【描述】

设置模块公共视频缓存池属性。

【语法】

```
RK_S32 RK_MPI_MB_SetModPoolConfig(MB\_UID\_E enMbUid, const MB\_CONFIG\_S *pstMbConfig);
```

【参数】

参数名	描述	输入/输出
enMbUid	使用模块公共视频缓冲池的模块 ID。	输入
pstMbConfig	模块公共视频缓存池属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 需要在调用[RK_MPI_SYS_Init](#)前设置该属性。

3.34 RK_MPI_MB_GetModPoolConfig

【描述】

获取模块公共视频缓存池属性。

【语法】

```
RK_S32 RK_MPI_MB_GetModPoolConfig(MB\_UID\_E enMbUid, MB\_CONFIG\_S *pstMbConfig);
```

【参数】

参数名	描述	输入/输出
enMbUid	使用模块公共视频缓冲池的模块 ID。	输入
pstMbConfig	模块公共视频缓存池属性指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 无

3.35 RK_MPI_MB_SetConfig

【描述】

设置视频公共缓存池属性。

【语法】

```
RK_S32 RK_MPI_MB_SetConfig(const MB\_CONFIG\_S *pstMbConfig);
```

【参数】

参数名	描述	输入/输出
pstMbConfig	视频缓冲池设置属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 只能在系统处于未初始化的状态下, 才可以设置缓存池属性, 否则会返回失败。
- 公共缓存池中每个缓存块的大小应根据当前图像像素格式以及图像是否压缩而有所不同。具体分配大小请参考[内存池单块MB大小计算规则](#)中的描述。

3.36 RK_MPI_MB_GetConfig

【描述】

获取视频公共缓存池属性。

【语法】

```
RK_S32 RK_MPI_MB_GetConfig(MB\_CONFIG\_S *pstMbConfig);
```

【参数】

参数名	描述	输入/输出
pstMbConfig	视频缓冲池设置属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 必须先调用[RK_MPI_MB_SetConfig](#)设置公共视频缓存池属性, 才能再获取属性, 否则返回**RK_ERR_MB_NOT_PERM**。

3.37 RK_MPI_MB_Init

【描述】

初始化视频公共缓存池。

【语法】

```
RK_S32 RK_MPI_MB_Init();
```

【参数】

参数名	描述	输入/输出
mb	缓存块ID。	输入
u32HorStride	图像虚宽, 以字节为单位。	输入
u32VerStride	图像虚高, 以字节为单位。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 必须先调用[RK_MPI_MB_SetConfig](#)配置缓存池属性, 再初始化缓存池, 否则返回**RK_ERR_MB_NOT_PERM**。
- 可反复初始化, 不返回失败。
- 初始化成功后再次设置[RK_MPI_MB_SetConfig](#)会保存设置参数, 但不会立即生效。

3.38 RK_MPI_MB_Exit

【描述】

反初始化视频公共缓存池。

【语法】

```
RK_S32 RK_MPI_MB_Exit();
```

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 初始化缓存池成功后, 必须再调用该接口[RK_MPI_SYS_Exit](#)去初始化 MPP 系统, 再去初始化缓存池, 否则返回失败。
- 可以反复反初始化, 不返回失败。
- 反初始化不会清除先前对缓存池的配置。

3.39 RK_MPI_MB_SetBufferStride

【描述】

设置缓存块用于存储图像时该图像的虚宽高Stride。

【语法】

```
RK_S32 RK_MPI_MB_SetBufferStride(MB\_BLK mb, RK_U32 u32HorStride, RK_U32 u32VerStride);
```

【参数】

参数名	描述	输入/输出
mb	缓存块ID。	输入
u32HorStride	图像虚宽，以字节为单位。	输入
u32VerStride	图像虚高，以字节为单位。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 SYS错误码 。

【注意】

- 无

3.40 RK_MPI_SYS_WaitFreeMB

【描述】

调用这个函数可以等待所有异步释放内存操作完成

【语法】

```
RK_S32 RK_MPI_SYS_WaitFreeMB();
```

【参数】

无

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 SYS错误码 。

【注意】

- 无

3.41 RK_MPI_SYS_RelasePhyMemory

【描述】

将指定的媒体BUFFER，释放到系统中使用

【语法】

```
RK_S32 RK_MPI_SYS_RelasePhyMemory();
```

【参数】

无

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 调用这个函数之前必须先删除所有的媒体对象，并且调用RK_MPI_SYS_WaitFreeMB
- 调用之后，媒体模块无法继续工作，重启后可以恢复
- 释放的内存参考KO相关的文档说明
- 如果ROCKIT使用KO加载，需指定释放的区域，具体请参见
《Rockchip_Rockit_Runtime_Library_Developer_Guide.pdf》

3.42 RK_MPI_SYS_SetForceLostFrame

【描述】

将指定的模块强制丢掉指定的帧数

【语法】

```
RK_S32 RK_MPI_SYS_SetForceLostFrame();
```

【参数】

参数名	描述	输入/输出
pstChn	指定通道, 请参见 MPP CHN S	输入
frameCnt	丢帧数	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 SYS错误码 。

【注意】

- 无

4. 数据类型

基本数据类型定义如下：

4.1 公共数据类型

```
typedef unsigned char      RK_UCHAR;
typedef uint8_t            RK_U8;
typedef uint16_t           RK_U16;
typedef uint32_t           RK_U32;
typedef uint32_t           RK_UL;
typedef uintptr_t          RK_UINTPTR_T;

typedef char               RK_CHAR;
```

```

typedef int8_t      RK_S8;
typedef int16_t     RK_S16;
typedef int32_t     RK_S32;
typedef int32_t     RK_SL;

typedef float       RK_FLOAT;
typedef double      RK_DOUBLE;

typedef uint64_t    RK_U64;
typedef int64_t     RK_S64;

typedef uint32_t    RK_SIZE_T;
typedef uint32_t    RK_LENGTH_T;

typedef unsigned int RK_HANDLE;

typedef enum {
    RK_FALSE = 0,
    RK_TRUE  = 1,
} RK_BOOL;

#ifndef NULL
#define NULL      0L
#endif

#define RK_NULL      0L
#define RK_SUCCESS    0
#define RK_FAILURE   (-1)

#define RK_VOID      void

```

4.2 MOD_ID_E

【说明】

定义模块 ID 枚举类型。

【定义】

```

typedef enum rkMOD_ID_E {
    RK_ID_CMPI  = 0,
    RK_ID_MB    = 1,
    RK_ID_SYS   = 2,
    RK_ID_RGN   = 3,
    RK_ID_VENC  = 4,
    RK_ID_VDEC  = 5,
    RK_ID_VPSS  = 6,
    RK_ID_VGS   = 7,
    RK_ID_VI    = 8,
    RK_ID_VO    = 9,
    RK_ID_AI    = 10,
    RK_ID_AO    = 11,
    RK_ID_AENC  = 12,
    RK_ID_ADEC  = 13,
    RK_ID_TDE   = 14,
    RK_ID_ISP   = 15,
    RK_ID_BUTT,
} MOD_ID_E;

```

【注意事项】

无

4.3 MB_UID_E

【说明】

定义媒体内存缓冲池的模块 ID 枚举类型。

【定义】

```
typedef enum rkMB_UID_E {
    MB_UID_VI = 0,
    MB_UID_VO = 1,
    MB_UID_VGS = 2,
    MB_UID_VENC = 3,
    MB_UID_VDEC = 4,
    MB_UID_VPSS = 5,
    MB_UID_AI = 6,
    MB_UID_AENC = 7,
    MB_UID_ADEC = 8,
    MB_UID_AVIS = 9,
    MB_UID_BUTT = 10
} MB_UID_E;
```

【注意事项】

无

4.4 RK_CODEC_ID_E

【说明】

定义音视频编码类型枚举。

【定义】

```
typedef enum rkCODEC_ID_E {
    RK_VIDEO_ID_Unused,           /**< Value when coding is N/A */
    RK_VIDEO_ID_AutoDetect,       /**< Autodetection of coding type */
    RK_VIDEO_ID_MPEG1VIDEO,
    RK_VIDEO_ID_MPEG2VIDEO,       /**< AKA: H.262 */
    RK_VIDEO_ID_H263,             /**< H.263 */
    RK_VIDEO_ID_MPEG4,             /**< MPEG-4 */
    RK_VIDEO_ID_WMV,              /**< Windows Media Video (WMV1,WMV2,WMV3) */
    RK_VIDEO_ID_RV,               /**< all versions of Real Video */
    RK_VIDEO_ID_AVC,               /**< H.264/AVC */
    RK_VIDEO_ID_MJPEG,             /**< Motion JPEG */
    RK_VIDEO_ID_VP8,               /**< VP8 */
    RK_VIDEO_ID_VP9,               /**< VP9 */
    RK_VIDEO_ID_HEVC,              /**< ITU H.265/HEVC */
    RK_VIDEO_ID_DolbyVision,       /**< Dolby Vision */
    RK_VIDEO_ID_ImageHEIC,         /**< HEIF image encoded with HEVC */
    RK_VIDEO_ID_VC1 = 0x01000000,  /**< Windows Media Video (WMV1,WMV2,WMV3) */
    RK_VIDEO_ID_FLV1,              /**< Sorenson H.263 */
    RK_VIDEO_ID_DIVX3,              /**< DIVX3 */
    RK_VIDEO_ID_VP6,
    RK_VIDEO_ID_AVSPPLUS,          /**< AVS+ profile=0x48 */
    RK_VIDEO_ID_AVIS,               /**< AVS profile=0x20 */
/* * < Reserved region for introducing Khronos Standard Extensions */
    RK_VIDEO_ID_KhronosExtensions = 0x2F000000,
/* * < Reserved region for introducing Vendor Extensions */
    RK_VIDEO_ID_VendorStartUnused = 0x3F000000,
    RK_VIDEO_ID_Max = 0x3FFFFFFF,
    RK_AUDIO_ID_Unused = 0x40000000, /**< Placeholder value when coding is N/A */
    RK_AUDIO_ID_AutoDetect,        /**< auto detection of audio format */
    RK_AUDIO_ID_PCM_ALAW,          /**< g711a */
    RK_AUDIO_ID_PCM_MULAW,         /**< g711u */
    RK_AUDIO_ID_PCM_S16LE,         /**< Any variant of PCM_S16LE coding */
}
```

```

RK_AUDIO_ID_PCM_S24LE, /*< Any variant of PCM_S24LE coding */
RK_AUDIO_ID_PCM_S32LE, /*< Any variant of PCM_S32LE coding */
RK_AUDIO_ID_ADPCM_G722, /*< Any variant of ADPCM_G722 encoded data */
RK_AUDIO_ID_ADPCM_G726, /*< Any variant of ADPCM_G726 encoded data */
RK_AUDIO_ID_ADPCM_IMA_QT, /*< Any variant of ADPCM_IMA encoded data */
RK_AUDIO_ID_AMR_NB, /*< Any variant of AMR_NB encoded data */
RK_AUDIO_ID_AMR_WB, /*< Any variant of AMR_WB encoded data */
RK_AUDIO_ID_GSMFR, /*< Any variant of GSM fullrate (i.e. GSM610) */
RK_AUDIO_ID_GSMEFR, /*< Any variant of GSM Enhanced Fullrate encoded data*/
RK_AUDIO_ID_GSMHR, /*< Any variant of GSM Halfrate encoded data */
RK_AUDIO_ID_PDCFR, /*< Any variant of PDC Fullrate encoded data */
RK_AUDIO_ID_PDCEFR, /*< Any variant of PDC Enhanced Fullrate encoded data */
RK_AUDIO_ID_PDCHR, /*< Any variant of PDC Halfrate encoded data */
RK_AUDIO_ID_TDMAFR, /*< Any variant of TDMA Fullrate encoded data (TIA/EIA-136-420) */
RK_AUDIO_ID_TDMAEFR, /*< Any variant of TDMA Enhanced Fullrate encoded data (TIA/EIA-136-410) */
RK_AUDIO_ID_QCELP8, /*< Any variant of QCELP 8kbps encoded data */
RK_AUDIO_ID_QCELP13, /*< Any variant of QCELP 13kbps encoded data */
RK_AUDIO_ID_EVRC, /*< Any variant of EVRC encoded data */
RK_AUDIO_ID_SMV, /*< Any variant of SMV encoded data */
RK_AUDIO_ID_G729, /*< Any variant of G.729 encoded data */
RK_AUDIO_ID_AAC, /*< Any variant of AAC encoded data */
RK_AUDIO_ID_MP3, /*< Any variant of MP3 encoded data */
RK_AUDIO_ID_SBC, /*< Any variant of SBC encoded data */
RK_AUDIO_ID_VORBIS, /*< Any variant of VORBIS encoded data */
RK_AUDIO_ID_WMA, /*< Any variant of WMA encoded data */
RK_AUDIO_ID_RA, /*< Any variant of RA encoded data */
RK_AUDIO_ID_MIDI, /*< Any variant of MIDI encoded data */
RK_AUDIO_ID_FLAC, /*< Any variant of FLAC encoded data */
RK_AUDIO_ID_APE = 0x50000000,
/*< Reserved region for introducing Khronos Standard Extensions */
RK_AUDIO_CodingKhronosExtensions = 0x6F000000,
/*< Reserved region for introducing Vendor Extensions */
RK_AUDIO_CodingVendorStartUnused = 0x7F000000,
RK_AUDIO_ID_WMAV1,
RK_AUDIO_ID_WMAV2,
RK_AUDIO_ID_WMAPRO,
RK_AUDIO_ID_WMALOSSLESS,
RK_AUDIO_ID_MP1,
RK_AUDIO_ID_MP2,
/*< add audio bitstream Codec ID define for RT> */
RK_AUDIO_ID_DTS,
RK_AUDIO_ID_AC3,
RK_AUDIO_ID_EAC3,
RK_AUDIO_ID_DOLBY_TRUEHD,
RK_AUDIO_ID_MLP,
RK_AUDIO_ID_DTS_HD,
RK_AUDIO_CodingMax = 0x7FFFFFFF,

/* subtitle codecs */
RK_SUB_ID_Unused = 0x80000000, //< A dummy ID pointing at the start of subtitle codecs.
RK_SUB_ID_DVD,
RK_SUB_ID_DVB,
RK_SUB_ID_TEXT, //< raw UTF-8 text
RK_SUB_ID_XSUB,
RK_SUB_ID_SSA,
RK_SUB_ID_MOV_TEXT,
RK_SUB_ID_HDMV_PGS,
RK_SUB_ID_DVB_TELETEXT,
RK_SUB_ID_SRT,

RK_SUB_ID_MICRODVD = 0x80000800,
RK_SUB_ID_EIA_608,
RK_SUB_ID_JACOSUB,
RK_SUB_ID_SAMI,
RK_SUB_ID_REALTEXT,
RK_SUB_ID_STL,
RK_SUB_ID_SUBVIEWER1,
RK_SUB_ID_SUBVIEWER,

```

```
RK_SUB_ID_SUBRIP,
RK_SUB_ID_WEBVTT,
RK_SUB_ID_MPL2,
RK_SUB_ID_VPLAYER,
RK_SUB_ID_PJS,
RK_SUB_ID_ASS,
RK_SUB_ID_HDMV_TEXT,
RK_SUB_CodingMax
} RK_CODEC_ID_E;
```

【注意事项】

无

4.5 ROTATION_E

【说明】

定义旋转角度枚举。

【定义】

```
typedef enum rkROTATION_E {
    ROTATION_0      = 0,
    ROTATION_90     = 1,
    ROTATION_180    = 2,
    ROTATION_270    = 3,
    ROTATION_BUTT
} ROTATION_E;
```

【注意事项】

无

4.6 POINT_S

【说明】

定义坐标信息结构体。

【定义】

```
typedef struct rkPOINT_S {
    RK_S32 s32X;
    RK_S32 s32Y;
} POINT_S;
```

【注意事项】

无

4.7 RECT_S

【说明】

定义矩形区域信息结构体。

【定义】

```
typedef struct rkRECT_S {
    RK_S32 s32X;
    RK_S32 s32Y;
    RK_U32 u32Width;
    RK_U32 u32Height;
} RECT_S;
```

【注意事项】

无

4.8 CROP_INFO_S

【说明】

定义 CROP 属性结构体

【定义】

```
typedef struct rkCROP_INFO_S {  
    RK_BOOL bEnable;  
    RECT_S stRect;  
} CROP_INFO_S;
```

【注意事项】

无

4.9 ROTATION_EX_S

【说明】

定义任意角度旋转属性。

【定义】

```
typedef struct rkROTATION_EX_S {  
    /* RW;Range: [0, 2];Rotation mode */  
    ROTATION_VIEW_TYPE_E enViewType;  
    /* RW;Range: [0,360];Rotation Angle:[0,360] */  
    RK_U32          u32Angle;  
    /*  
     * RW;Range: [-511, 511];Horizontal offset of the image  
     * distortion center relative to image center.  
     */  
    RK_S32          s32CenterXOffset;  
    /*  
     * RW;Range: [-511, 511];Vertical offset of the image  
     * distortion center relative to image center.  
     */  
    RK_S32          s32CenterYOffset;  
    /* RW;Dest size of any angle rotation */  
    SIZE_S          stDestSize;  
} ROTATION_EX_S;
```

【注意事项】

无

4.10 VIDEO_PROC_DEV_TYPE_E

【说明】

定义硬件设备类型枚举。

【定义】

```

typedef enum rkVIDEO_PROC_DEV_TYPE_E {
    VIDEO_PROC_DEV_GPU = 0x0, /* GPU device */
    VIDEO_PROC_DEV_RGA = 0x1, /* RGA device */
    VIDEO_PROC_DEV_ISP = 0x2, /* ISP device */
    VIDEO_PROC_DEV_VPSS = 0x3, /* VPSS device */

    VIDEO_PROC_DEV_BUTT
} VIDEO_PROC_DEV_TYPE_E;

```

【成员】

成员名称	描述
VIDEO_PROC_DEV_GPU	GPU
VIDEO_PROC_DEV_RGA	RGA
VIDEO_PROC_DEV_ISP	ISP
VIDEO_PROC_DEV_VPSS	VPSS

【注意事项】

- 模块 VPSS 支持配置 GPU / RGA / ISP / VPSS 作为其视频图像处理实现的硬件设备，具体描述请参考“视频处理子系统”章节，相关调用接口 RK_MPI_VPSS_SetVProcDev 和 RK_MPI_VPSS_GetVProcDev。

4.11 MB_SOURCE_E

【说明】

定义MB来源选择。

【定义】

```

typedef enum rkMB_SOURCE_E {
    MB_SOURCE_COMMON = 0,
    MB_SOURCE_MODULE = 1,
    MB_SOURCE_PRIVATE = 2,
    MB_SOURCE_USER = 3,
    MB_SOURCE_BUTT
} MB_SOURCE_E;

```

【成员】

成员名称	描述
MB_SOURCE_COMMON	公共缓冲池。
MB_SOURCE_MODULE	模块公共缓冲池。
MB_SOURCE_USER	用户缓冲池。
MB_SOURCE_PRIVATE	私有缓冲池。

【注意事项】

- 详细描述请见[内存缓冲池](#)。

4.12 MPP_CHN_S

【说明】

定义模块设备通道结构体。

【定义】

```
typedef struct rkMPP_CHN_S {
    MOD_ID_E enModId;
    RK_S32 s32DevId;
    RK_S32 s32ChnId;
} MPP_CHN_S;
```

【成员】

成员名称	描述
enModId	模块号。
s32DevId	设备号。
s32ChnId	通道号。

【注意事项】

无

4.13 CHN_INPUT_MODE_E

【说明】

定义通道输入流模式。

【定义】

```
typedef enum rkCHN_INPUT_MODE_E {
    CHN_INPUT_MODE_NORMAL, /* CHN receive all packet */
    CHN_INPUT_MODE_REMAIN_NEWEST, /* CHN remain newest packet */
    CHN_INPUT_MODE_DROP_ALWAYS, /* CHN drop all packet */
    CHN_INPUT_MODE_BUTT
} CHN_INPUT_MODE_E;
```

【注意事项】

无

4.14 MB_POOL

【说明】

定义MB内存池的句柄。

【定义】

```
typedef RK_U32 MB_POOL;
```

【注意事项】

无

4.15 MB_BLK

【说明】

定义MB内存的句柄。

【定义】

```
typedef void * MB_BLK;
```

【注意事项】

无

4.16 MB_MAX_POOLS

【说明】

定义MB内存池的最大个数。

【定义】

```
#define MB_MAX_POOLS      512
```

【注意事项】

无

4.17 MB_POOL_CONFIG_S

【说明】

定义内存缓存池属性结构体。

【定义】

```
typedef struct rkMB_POOL_CONFIG_S {
    RK_U64 u64MBSize;
    RK_U32 u32MBCnt;
    MB_REMAP_MODE_E enRemapMode;
    MB_ALLOC_TYPE_E enAllocType;
    MB_DMA_TYPE_E enDmaType;
    RK_BOOL bPreAlloc;
} MB_POOL_CONFIG_S;
```

【成员】

成员名称	描述
u64MBSize	缓存块大小，以 Byte 位单位。
u32MBCnt	每个缓存池的缓存块个数。取值范围：(0, 10240]
enRemapMode	内核态虚拟地址映射模式。
enAllocType	申请的内存类型。
enDmaType	申请的DMA物理内存类型，如配置是否物理连续
bPreAlloc	是否在缓存池创建时申请好缓存块。

【注意事项】

无

4.18 MB_REMAP_MODE_E

【说明】

定义 MB 内核态虚拟地址映射模式。

【定义】

```
typedef enum rkMB_REMAP_MODE_E {
    MB_REMAP_MODE_NONE = 0, /* no remap */
    MB_REMAP_MODE_NOCACHE = 1 << 8, /* no cache remap */
    MB_REMAP_MODE_CACHED = 1 << 9, /* cache remap, if you use this mode, you should flush cache by yourself */
    MB_REMAP_MODE_BUTT
} MB_REMAP_MODE_E;
```

【成员】

成员名称	描述
MB_REMAP_MODE_NOCACHE	MB 映射 nocache 属性的内核态虚拟地址。
MB_REMAP_MODE_CACHED	MB 映射 cached 属性的内核态虚拟地址。

【注意事项】

无

4.19 MB_DMA_TYPE_E

【说明】

定义申请的物理内存类型。

【定义】

```
typedef enum rkMB_DMA_TYPE_E {
    MB_DMA_TYPE_NONE = 0, /* Physically Non-Continuous memory default */
    MB_DMA_TYPE_CMA = 1 << 12, /* Physically Continuous memory */
    MB_DMA_TYPE_BUTT
} MB_DMA_TYPE_E;
```

【成员】

成员名称	描述
MB_DMA_TYPE_NONE	申请的物理内存为非连续物理地址内存。
MB_DMA_TYPE_CMA	申请的物理内存为连续物理地址内存。

【注意事项】

无

4.20 MB_ALLOC_TYPE_E

【说明】

定义MB申请的内存类型。

【定义】

```
typedef enum rkMB_ALLOC_TYPE {
    MB_ALLOC_TYPE_UNUSED = -1,
    MB_ALLOC_TYPE_DMA = 0,
    MB_ALLOC_TYPE_MALLOC,
    MB_ALLOC_TYPE_MAX,
} MB_ALLOC_TYPE_E;
```

【成员】

成员名称	描述
MB_ALLOC_TYPE_DMA	申请内核态可用的DMA内存。
MB_ALLOC_TYPE_MALLOC	申请malloc内存。

【注意事项】

无

4.21 MB_EXT_CONFIG_S

【说明】

定义申请外部MB内存的信息。

【定义】

```
typedef struct _rkMB_EXT_CONFIG_S {  
    RK_U8      *pu8VirAddr;  
    RK_U64     u64PhyAddr;  
    RK_S32     s32Fd;  
    RK_U64     u64Size;  
    RK_MPI_MB_FREE_CB pFreeCB;  
    RK_VOID    *pOpaque;  
} MB_EXT_CONFIG_S;
```

【成员】

成员名称	描述
pu8VirAddr	外部申请的虚拟地址。
u64PhyAddr	外部申请的物理地址。
s32Fd	外部申请的内存句柄。
u64Size	外部申请的内存大小。
pFreeCB	申请内存的释放回调方法。
pOpaque	申请内存的释放回调所需的上下文。

【注意事项】

- 使用此结构体时，建议先将结构体memset为0后使用。

4.22 MB_CONFIG_S

【说明】

定义媒体内存缓存池属性结构体。

【定义】

```
typedef struct rkMB_CONFIG_S {  
    RK_U32 u32MaxPoolCnt;  
    MB_POOL_CONFIG_S astCommPool[MB_MAX_COMM_POOLS];  
} MB_CONFIG_S;
```

【成员】

成员名称	描述
u32MaxPoolCnt	整个系统中可容纳的缓存池个数。 静态属性。 保留属性，目前未生效。
astCommPool	公共缓存池属性结构体。 静态属性。

【注意事项】

- u64MBSIZE 等于 0 或 u32MBCnt 等于 0，则对应的缓存池不会被创建。
- 建议整个结构体先 memset 为 0 再按需赋值，否则出现随机值内部无法完全过滤。

5. SYS错误码

5.1 系统控制SYS错误码

系统控制 API SYS错误码如下所示：

错误代码	宏定义	描述
0xA0028006	RK_ERR_SYS_NULL_PTR	空指针错误
0xA0028010	RK_ERR_SYS_NOTREADY	系统控制属性未配置
0xA0028009	RK_ERR_SYS_NOT_PERM	操作不允许
0xA002800C	RK_ERR_SYS_NOMEM	分配内存失败，如系统内存不足
0xA0028003	RK_ERR_SYS_ILLEGAL_PARAM	参数设置无效
0xA0028012	RK_ERR_SYS_BUSY	系统忙
0xA0028008	RK_ERR_SYS_NOT_SUPPORT	不支持的功能

5.2 内存缓存池SYS错误码

错误代码	宏定义	描述
0xA0018003	RK_ERR_MB_ILLEGAL_PARAM	参数设置无效
0xA0018005	RK_ERR_MB_UNEXIST	内存缓存池不存在
0xA0018006	RK_ERR_MB_NULL_PTR	参数空指针错误
0xA0018009	RK_ERR_MB_NOT_PERM	操作不允许
0xA001800C	RK_ERR_MB_NOMEM	分配内存失败
0xA001800D	RK_ERR_MB_NOBUF	分配缓存失败
0xA0018010	RK_ERR_MB_NOTREADY	系统控制属性未配置
0xA0018012	RK_ERR_MB_BUSY	系统忙
0xA0018013	RK_ERR_MB_SIZE_NOT_ENOUGH	MB 块大小不够
0xA0018040	RK_ERR_MB_2MPOOLS	创建缓存池太多

内存管理

1. API 参考

MMZ该功能模块为用户提供以下API：

- [RK_MPI_MMZ_Alloc](#): 申请用户缓存。
- [RK_MPI_MMZ_Free](#): 释放用户缓存。
- [RK_MPI_MMZ_Handle2PhysAddr](#): 获取用户缓存的物理地址。
- [RK_MPI_MMZ_Handle2VirAddr](#): 获取用户缓存的虚地址。
- [RK_MPI_MMZ_Handle2Fd](#): 获取用户缓存的fd。
- [RK_MPI_MMZ_GetSize](#): 获取用户缓存的大小。
- [RK_MPI_MMZ_Fd2Handle](#): 通过fd找到对应的用户缓存。
- [RK_MPI_MMZ_VirAddr2Handle](#): 通过虚地址查找对应的用户缓存。
- [RK_MPI_MMZ_PhysAddr2Handle](#): 通过物理查找对应的用户缓存。
- [RK_MPI_MMZ_IsCacheable](#): 查询用户缓存是否为cache缓存。
- [RK_MPI_MMZ_FlushCacheStart](#): 刷新cache里的内容到内存并且使cache里的内容无效，在cpu访问前调用，当offset和length都等于0时候，执行full sync，否则执行partial sync。
- [RK_MPI_MMZ_FlushCacheEnd](#): 刷新cache里的内容到内存并且使cache里的内容无效，在cpu访问结束后调用，当offset和length都等于0时候，执行full sync，否则执行partial sync。
- [RK_MPI_MMZ_FlushCacheVaddrStart](#): 刷新cache里的内容到内存并且使cache里的内容无效，在cpu访问前调用，指定待刷新内存的虚拟地址及其长度，只支持partial sync。
- [RK_MPI_MMZ_FlushCacheVaddrEnd](#): 刷新cache里的内容到内存并且使cache里的内容无效，在cpu访问结束后调用，指定待刷新内存的虚拟地址及其长度，只支持partial sync。
- [RK_MPI_MMZ_FlushCachePaddrStart](#): 刷新cache里的内容到内存并且使cache里的内容无效，在cpu访问前调用，指定待刷新内存的物理地址及其长度，只支持partial sync。
- [RK_MPI_MMZ_FlushCachePaddrEnd](#): 刷新cache里的内容到内存并且使cache里的内容无效，在cpu访问结束后调用，指定待刷新内存的物理地址及其长度，只支持partial sync。

1.1 RK_MPI_MMZ_Alloc

【描述】

申请用户缓存。

【语法】

```
RK_S32 RK_MPI_MMZ_Alloc(MB\_BLK *pBlk, RK_U32 u32Length, RK_U32 u32Flags);
```

【参数】

参数名	描述	输入/输出
pBlk	缓存块指针	输出
u32Length	缓存块的大小	输入
u32Flags	内存标志，可同时或者分别带上是否支持Cache标记和物理内存是否连续标记，当前支持如下标志： RK_MMZ_ALLOC_CACHEABLE: 支持Cache缓存 RK_MMZ_ALLOC_UNCACHEABLE: 不支持Cache缓存 RK_MMZ_ALLOC_TYPE_IOMMU: 带上此标志表示申请物理不连续内存 RK_MMZ_ALLOC_TYPE_CMA: 带上此标志表示申请物理连续内存	输入

【返回值】

返回值	描述
0	成功
负值	失败

【注意】

- 无。

1.2 RK_MPI_MMZ_Free

【描述】

释放用户缓存。

【语法】

```
RK_S32 RK_MPI_MMZ_Free(MB\_BLK blk);
```

【参数】

参数名	描述	输入/输出
blk	缓存块ID	输入

【返回值】

返回值	描述
0	成功
负值	失败

【注意】

- 无。

1.3 RK_MPI_MMZ_Handle2PhysAddr

【描述】

获取用户缓存的物理地址。

【语法】

```
RK_U64 RK_MPI_MMZ_Handle2PhysAddr(MB\_BLK blk);
```

【参数】

参数名	描述	输入/输出
blk	缓存块ID	输入

【返回值】

返回值	描述
大于0	获取有效物理地址
0	失败

【注意】

- 无。

1.4 RK_MPI_MMZ_Handle2VirAddr

【描述】

获取用户缓存的虚地址。

【语法】

```
RK_VOID *RK_MPI_MMZ_Handle2VirAddr(MB\_BLK blk);
```

【参数】

参数名	描述	输入/输出
blk	缓存块ID	输入

【返回值】

返回值	描述
非RK_NULL	有效的虚拟地址
RK_NULL	获取虚拟地址失败

【注意】

- 无

1.5 RK_MPI_MMZ_Handle2Fd

【描述】

获取用户缓存的fd。

【语法】

```
RK_S32 RK_MPI_MMZ_Handle2Fd(MB\_BLK blk);
```

【参数】

参数名	描述	输入/输出
blk	缓存块ID	输入

【返回值】

返回值	描述
非负	有效的fd
负数	失败

【注意】

- 无

1.6 RK_MPI_MMZ_GetSize

【描述】

获取用户缓存的大小。

【语法】

```
RK_U64 RK_MPI_MMZ_GetSize(MB\_BLK blk);
```

【参数】

参数名	描述	输入/输出
blk	缓存块ID	输入

【返回值】

返回值	描述
非负	缓存块的大小
负数	失败

【注意】

- 无

1.7 RK_MPI_MMZ_Fd2Handle

【描述】

通过fd查找到对应的用户缓存。

【语法】

```
MB_BLK RK_MPI_MMZ_Fd2Handle(RK_S32 u32Fd);
```

【参数】

参数名	描述	输入/输出
u32Fd	fd值	输入

【返回值】

返回值	描述
非空	缓存块ID
RK_NULL	失败

【注意】

- 无

1.8 RK_MPI_MMZ_VirAddr2Handle

【描述】

通过虚拟地址查找对应的用户缓存。

【语法】

```
MB_BLK RK_MPI_MMZ_VirAddr2Handle(RK_VOID *pVirAddr);
```

【参数】

参数名	描述	输入/输出
pVirAddr	虚拟地址	输入

【返回值】

返回值	描述
非空	缓存块ID
RK_NULL	失败

1.9 RK_MPI_MMZ_PhysAddr2Handle

【描述】

通过物理地址查找对应的用户缓存。

【语法】

```
MB_BLK RK_MPI_MMZ_PhysAddr2Handle(RK_U64 u64phyAddr);
```

【参数】

参数名	描述	输入/输出
u64phyAddr	虚拟地址	输入

【返回值】

返回值	描述
非空	缓存块ID
RK_NULL	失败

1.10 RK_MPI_MMZ_IsCacheable

【描述】

查询用户缓存是否为cache缓存。

【语法】

```
RK_S32 RK_MPI_MMZ_IsCacheable(MB\_BLK blk);
```

【参数】

参数名	描述	输入/输出
blk	缓存块ID	输入

【返回值】

返回值	描述
1	当前缓存是cache缓存
0	当前缓存非cache缓存
负值	查询失败

1.11 RK_MPI_MMZ_FlushCacheStart

【描述】

刷新cache里的内容到内存并且使cache里的内容无效，在cpu访问前调用，当offset和length都等于0时候，执行full sync，否则执行partial sync。以[start](#)开始，以[end](#)结束，其间是CPU对该内存的操作。

【语法】

RK_S32 RK_MPI_MMZ_FlushCacheStart(MB_BLK blk, RK_U32 u32Offset, RK_U32 u32Length, RK_U32 u32Flags);

【参数】

参数名	描述	输入/输出
blk	缓存块ID	输入
u32Offset	指定刷新内存的偏移地址	输入
u32Length	需要刷新cache的长度	输入
u32Flags	读写标志, flag在start和end中需要保持一致: RK_MMZ_SYNC_READONLY: 在start和end之间的代码, 对指定内存cpu只做读操作 RK_MMZ_SYNC_WRITEONLY: 在start和end之间的代码, 对指定内存cpu只做写操作 RK_MMZ_SYNC_RW: 在start和end之间的代码, 对指定内存cpu即有读也有写	输入

【返回值】

返回值	描述
0	成功
负值	失败

1.12 RK_MPI_MMZ_FlushCacheEnd

【描述】

刷新cache 里的内容到内存并且使 cache 里的内容无效, 在cpu访问结束后调用, 当offset和length都等于0时候, 执行full sync, 否则执行partial sync。以[start](#)开始, 以[end](#)结束, 其间是CPU对该内存的操作。

【语法】

RK_S32 RK_MPI_MMZ_FlushCacheEnd(MB_BLK blk, RK_U32 u32Offset, RK_U32 u32Length, RK_U32 u32Flags);

【参数】

参数名	描述	输入/输出
blk	缓存块ID	输入
u32Offset	指定刷新内存的偏移地址	输入
u32Length	需要刷新内存的长度	输入
u32Flags	读写标志, flag在start和end中需要保持一致: RK_MMZ_SYNC_READONLY: 在start和end之间的代码, 对指定内存cpu只做读操作 RK_MMZ_SYNC_WRITEONLY: 在start和end之间的代码, 对指定内存cpu只做写操作 RK_MMZ_SYNC_RW: 在start和end之间的代码, 对指定内存cpu即有读也有写	输入

【返回值】

返回值	描述
0	成功
负值	失败

1.13 RK_MPI_MMZ_FlushCacheVaddrStart

【描述】

刷新cache 里的内容到内存并且使 cache 里的内容无效，在cpu访问前调用，指定待刷新内存的虚拟地址及其长度，只支持 partial sync。以[start](#)开始，以[end](#)结束， 其间是CPU对该内存的操作。

【语法】

```
RK_S32 RK_MPI_MMZ_FlushCacheVaddrStart(RK_VOID *pVirAddr, RK_U32 u32Length, RK_U32 u32Flags);
```

【参数】

参数名	描述	输入/输出
pVirAddr	指定刷新内存的虚拟地址	输入
u32Length	需要刷新内存的长度	输入
u32Flags	读写标志，flag在start和end中需要保持一致： RK_MMZ_SYNC_READONLY: 在start和end之间的代码，对指定内存cpu只做读操作 RK_MMZ_SYNC_WRITEONLY: 在start和end之间的代码，对指定内存cpu只做写操作 RK_MMZ_SYNC_RW: 在start和end之间的代码，对指定内存cpu即有读也有写	输入

【返回值】

返回值	描述
0	成功
负值	失败

1.14 RK_MPI_MMZ_FlushCacheVaddrEnd

【描述】

刷新cache 里的内容到内存并且使 cache 里的内容无效，在cpu访问结束后调用，指定待刷新内存的虚拟地址及其长度，只支持 partial sync。以[start](#)开始，以[end](#)结束， 其间是CPU对该内存的操作。

【语法】

```
RK_S32 RK_MPI_MMZ_FlushCacheVaddrEnd(RK_VOID *pVirAddr, RK_U32 u32Length, RK_U32 u32Flags);
```

【参数】

参数名	描述	输入/输出
pVirAddr	指定刷新内存的虚拟地址	输入
u32Length	需要刷新内存的长度	输入
u32Flags	读写标志，flag在start和end中需要保持一致： RK_MMZ_SYNC_READONLY: 在start和end之间的代码，对指定内存cpu只做读操作 RK_MMZ_SYNC_WRITEONLY: 在start和end之间的代码，对指定内存cpu只做写操作 RK_MMZ_SYNC_RW: 在start和end之间的代码，对指定内存cpu即有读也有写	输入

【返回值】

返回值	描述
0	成功
负值	失败

1.15 RK_MPI_MMZ_FlushCachePaddrStart

【描述】

刷新cache 里的内容到内存并且使 cache 里的内容无效，在cpu访问前调用，指定待刷新内存的物理地址及其长度，只支持 partial sync。以[start](#)开始，以[end](#)结束， 其间是CPU对该内存的操作。

【语法】

```
RK_S32 RK_MPI_MMZ_FlushCachePaddrStart(RK_U64 u64phyAddr, RK_U32 u32Length, RK_U32 u32Flags);
```

【参数】

参数名	描述	输入/输出
u64phyAddr	指定刷新内存的物理地址	输入
u32Length	需要刷新内存的长度	输入
u32Flags	读写标志, flag在start和end中需要保持一致： RK_MMZ_SYNC_READONLY: 在start和end之间的代码，对指定内存cpu只做读操作 RK_MMZ_SYNC_WRITEONLY: 在start和end之间的代码，对指定内存cpu只做写操作 RK_MMZ_SYNC_RW: 在start和end之间的代码，对指定内存cpu即有读也有写	输入

【返回值】

返回值	描述
0	成功
负值	失败

1.16 RK_MPI_MMZ_FlushCachePaddrEnd

【描述】

刷新cache 里的内容到内存并且使 cache 里的内容无效，在cpu访问结束后调用，指定待刷新内存的物理地址及其长度，只支持 partial sync。以[start](#)开始，以[end](#)结束， 其间是CPU对该内存的操作。

【语法】

```
RK_S32 RK_MPI_MMZ_FlushCachePaddrEnd(RK_U64 u64phyAddr, RK_U32 u32Length, RK_U32 u32Flags);
```

【参数】

参数名	描述	输入/输出
u64phyAddr	指定刷新内存的物理地址	输入
u32Length	需要刷新内存的长度	输入
u32Flags	读写标志, flag在start和end中需要保持一致： RK_MMZ_SYNC_READONLY: 在start和end之间的代码，对指定内存cpu只做读操作 RK_MMZ_SYNC_WRITEONLY: 在start和end之间的代码，对指定内存cpu只做写操作 RK_MMZ_SYNC_RW: 在start和end之间的代码，对指定内存cpu即有读也有写	输入

【返回值】

返回值	描述
0	成功
负值	失败

2. 数据类型

基本数据类型定义如下：

2.1 RK_MMZ_ALLOC_TYPE_IOMMU

【说明】

申请的物理内存为非连续物理地址内存。

【定义】

```
#define RK_MMZ_ALLOC_TYPE_IOMMU 0x00000000
```

2.2 RK_MMZ_ALLOC_TYPE_CMA

【说明】

申请的物理内存为连续物理地址内存。

【定义】

```
#define RK_MMZ_ALLOC_TYPE_CMA 0x00000001
```

2.3 RK_MMZ_ALLOC_CACHEABLE

【说明】

映射cached属性的用户态虚拟地址，需要用户自己去刷新cache。

【定义】

```
#define RK_MMZ_ALLOC_CACHEABLE 0x00000000
```

2.4 RK_MMZ_ALLOC_UNCACHEABLE

【说明】

映射非cached属性的用户态虚拟地址。

【定义】

```
#define RK_MMZ_ALLOC_UNCACHEABLE 0x00000010
```

2.5 RK_MMZ_SYNC_READONLY

【说明】

内存读操作。

【定义】

```
#define RK_MMZ_SYNC_READONLY 0x00000000
```

2.6 RK_MMZ_SYNC_WRITEONLY

【说明】

内存写操作。

【定义】

```
#define RK_MMZ_SYNC_WRITEONLY 0x00000001
```

2.7 RK_MMZ_SYNC_RW

【说明】

内存读和写操作。

【定义】

```
#define RK_MMZ_SYNC_RW      0x00000002
```

视频输入

[基本概念](#)

[重要概念](#)

[功能描述](#)

[举例](#)

[API 参考](#)

[数据类型](#)

[环境变量](#)

[VII 错误码](#)

1. 基本概念

视频输入（VI）模块实现的功能：通过 MIPI Rx(含 MIPI 接口、LVDS 接口)，BT.1120，BT.656，BT.601，DC 等接口接收视频数据。VI 将接收到的数据存入到指定的内存区域，实现视频数据的采集。

2. 重要概念

• 2.1 DEV设备

视频输入设备支持若干种时序输入，负责对时序进行解析。

• 2.2 PIPE管道

视频输入 PIPE 绑定在设备后端，负责设备解析后的数据再处理，DEV为在线模式时PIPE对应为bypass状态，DEV为离线模式时Pipe会接收DEV端数据并处理。

• 2.3 CHANNEL通道

视频输入最后一级的获取通道，如dev为isp时输出有四个channel。

• 2.4 DIS 电子防抖

DIS (Digital Image Stabilization) 模块通过比较当前图像与前两帧图像采用不同自由度的防抖算法计算出当前图像在各个轴方向上的抖动偏移向量，然后根据抖动偏移向量对当前图像进行校正，从而起到防抖的效果。

• 2.5 CAC 色差纠正

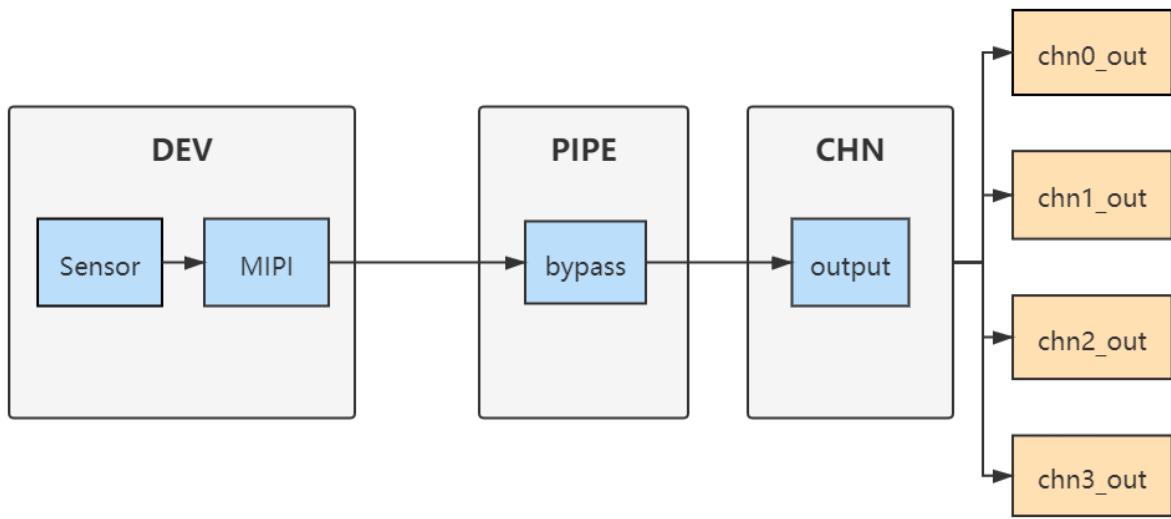
CAC (Chromatic Aberration Correction) 模块用于修正图像中的色差问题，色差通常由于透镜和光学元件的制造或设计不完美引起。通过调整颜色补偿系数微调图像中的不同颜色，以改善颜色准确性，从而提高整体图像质量，特别适用于消除边缘区域和高对比度区域的紫边。

3. 功能描述

3.1 功能框图

VI 在软件层次上划分 3个部分，如图[1-1](#)所示。

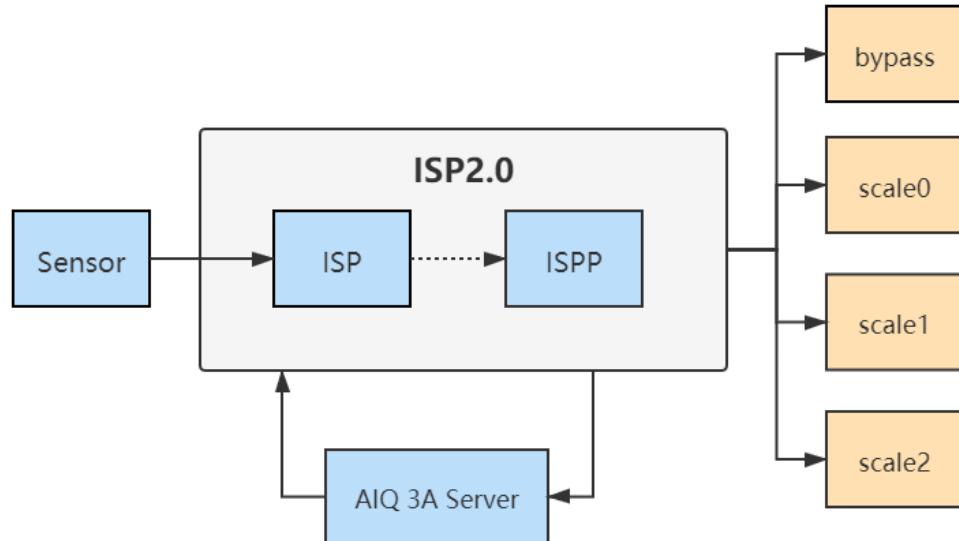
3.1.1 图1-1 VI处理流程框图



VI 从软件上划分了输入设备 (dev) , 输入 pipe、输出通道 (channel) 三个层级。各芯片的设备、PIPE、通道个数差异如下所示。

芯片	dev	pipe	channel
RV1126/RV1109	3	3	4
RK356X	3	3	2
RK3588	8	8	3
RV1106/RV1103	2	2	6

3.1.2 图1-2 RV1126/RV1109 VI硬件处理流程框图



【注意】

- 如vi使用过程无赋值aEntityName (参考 [VI_ISP_OPT_S](#)结构体定义) , 则默认chn0_out对应bypass, chn1_out对应scale0, chn2_out对应scale1, chn3_out对应scale2。

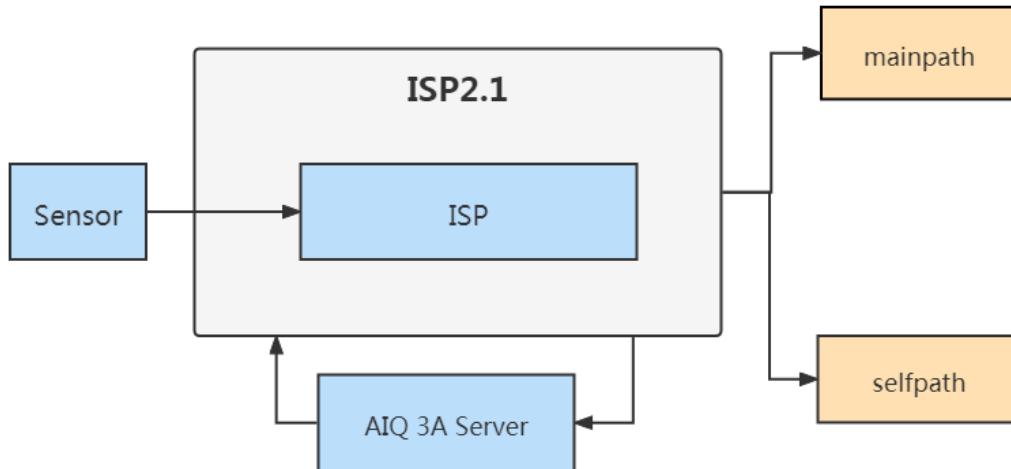
各路输出分辨率限制如下表:

aEntityName	name	max output	support output fmt
rkispp_m_bypass 或/dev/video18	bypass	默认输出ispp最大分辨率，不支持设置分辨率，不支持缩放。	NV12/NV16/YUYV/UYVY/FBC0/FBC2
rkispp_scale0 或/dev/video19	scale0	最大输出同bypass分辨率。 缩放 nv16:max width: 2080; nv12:max width:3264;最大支持8倍，最小2倍缩放	NV12/NV16/YUYV/UYVY
rkispp_scale1 或/dev/video20	scale1	max width: 1280, 最大支持8倍缩放， 最小2倍缩放	NV12/NV16/YUYV/UYVY
rkispp_scale2 或/dev/video21	scale2	max width: 1280, 最大支持8倍缩放， 最小2倍缩放	NV12/NV16/YUYV/UYVY

【注意】

- aEntityName 对应的 /dev/videoX 节点 X 不是固定的，可以使用 media-ctl 查看对应的节点。
- rkispp_scaleX 节点有最大宽度和缩放倍数的限制，若不满足 Sensor 原始分辨率的输出倍数关系会导致不出流。
- 可以通过 v4l2-ctl 命令去获取对应节点及分辨率是否能够正常输出，VI 同 v4l2 支持列表一致。
- 如需了解更多 isp 相关资料可以查看《Rockchip_Development_Guide_ISP2x_CN_v1.6.3.pdf》
《Rockchip_Instruction_Linux_Application_ISP20_CH.pdf》相关文档说明。

3.1.3 图1-3 RK356x VI硬件处理流程框图



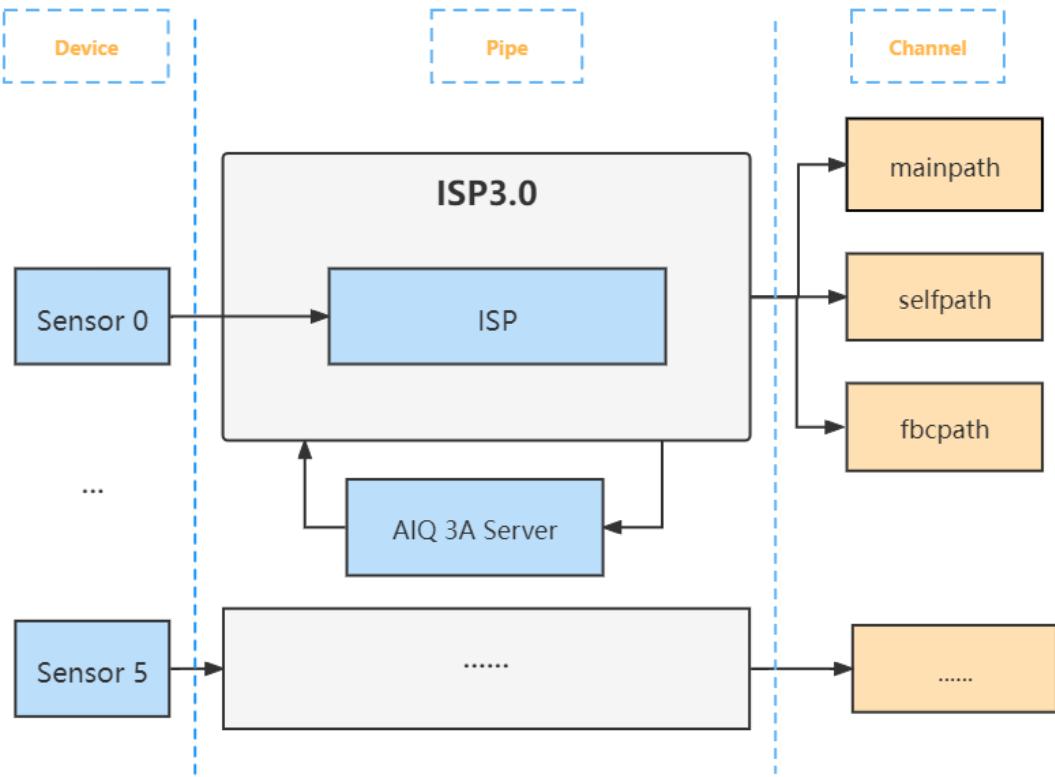
【注意】

- 如 vi 使用过程无赋值 aEntityName (参考 [VI_ISP_OPT_S](#) 结构体定义)，则默认 chn0_out 对应 mainpath，chn1_out 对应 selfpath。
- aEntityName 对应的 /dev/videoX 节点中，X不是固定的，可以使用 media-ctl 查看对应的节点。

各路输出分辨率限制如下表：

aEntityName	name	max output	support output fmt
/dev/video0	rkisp_mainpath	4096*2304, 最大支持8倍缩放	NV12/NV16/YUYV/UYVY
/dev/video1	rkisp_selfpath	1920*1080, 最大支持8倍缩放	NV12/NV16/YUYV/UYVY

3.1.4 图1-4 RK3588 VI硬件处理流程框图



【注意】

- 如 vi 使用过程无赋值 aEntityName (参考 [VI_ISP_OPT_S](#) 结构体定义) , 则默认 chn0_out 对应 mainpath, chn1_out 对应 selfpath, chn2_out 对应 fbcpath。多路isp使用时建议通过 dev/pipe/channel 使用默认定义。
- aEntityName 对应的 /dev/videoX 节点中, X 不是固定的, 可以使用 media-ctl 查看对应的节点。

各路输出分辨率限制如下表:

aEntityName	name	max output	support output fmt
/dev/video32	rkisp_mainpath	4672*3504, 最大支持8倍缩放	NV12/NV16/YUYV/UYVY
/dev/video33	rkisp_selfpath	1920*1080, 最大支持8倍缩放	NV12/NV16/YUYV/UYVY
/dev/video34	rkisp_fbcpath	fbc输出, sensor最大分辨率	AFBC

【注意】

- 以上分辨率限制是单路ISP 限制, RK3588支持两路 ISP 组合输出, 此时 rkisp_mainpath 最大分辨率支持到8064x6048。
- device 编号根据dts配置中sensor连接的属性 camera-module-index 值一一对应。
- rkisp_fbcpath 仅支持压缩格式输出, 其他不支持。
- 更多ISP 细节参看 RK3588 datasheet 中 ISP 章节说明。

3.1.5 图1-5 RV1106/RV1103 VI硬件处理流程框图

【注意】

- 如 vi 使用过程无赋值 aEntityName (参考 [VI_ISP_OPT_S](#) 结构体定义) , 则默认 chn0_out 对应 mainpath, chn1_out 对应 selfpath, chn2_out 对应 bypasspath。多路 ISP 使用时建议通过 dev/pipe/channel 使用默认定义。
- aEntityName 对应的 /dev/videoX 节点中, X 不是固定的, 可以使用 media-ctl 查看对应的节点。

各路输出分辨率限制如下表:

aEntityName	name	max output	support output fmt
/dev/video8	rkisp_mainpath	支持sensor最大输出分辨率，支持缩放	NV12/NV16/UYVY
/dev/video9	rkisp_selfpath	支持sensor最大输出分辨率，支持缩放	NV12/NV16/UYVY/RGB565/XBGR
/dev/video10	rkisp_bypasspath	支持sensor最大输出分辨率，支持缩放	NV12/NV16/UYVY
/dev/video11	rkisp_mainpath_4x4sampling	输出分辨率是rkisp_mainpath的宽与高的四分之一	NV12/NV16/UYVY
/dev/video12	rkisp_bypasspath_4x4sampling	输出分辨率是rkisp_bypasspath的宽与高的四分之一	NV12/NV16/UYVY

【注意】

- 以上分辨率输出限制是单路 ISP 限制。
- sensor输入最大输入3072x1728 (5M) @30fps，最小256x256。
- device编号根据dts配置中 sensor 连接的属性 camera-module-index 值一一对应。
- sensor输出最大分辨率根据 sensor 的宽高比不同而不同，支持sensor最大输出分辨率，并支持缩放。
- mainpath通道支持卷绕模式，selfpath和bypass理论不支持卷绕模式；同时只能支持一路码率和编码器卷绕。
- 只有selfpath支持XBGR8888格式输出。
- 支持对 rkisp_mainpath 与 rkisp_bypasspath 通道的下采样采集。仅支持4x4的下采样。故各下采样通道的分辨率是对应通道分辨率的四分之一。**下采样通道采集时，需在被采样对象通道开启后开启，在被采样通道停止前停止。**
- 更多 ISP 细节参看 rv1106 datasheet 中 ISP 章节说明。

3.1.6 图1-6 RK3576 VI硬件处理流程框图

【注意】

- 如vi使用过程无赋值 aEntityName (参考 [VI_ISP_OPT_S](#) 结构体定义)，则默认 chn0_out 对应 mainpath，chn1_out 对应 selfpath，chn2_out 对应 ldcpath。多路 ISP 使用时建议通过 dev/pipe/channel 使用默认定义。
- aEntityName 对应的 /dev/videoX 节点中，X不是固定的，可以使用 media-ctl 查看对应的节点。

各路输出分辨率限制如下表：

aEntityName	name	max output	support output fmt
/dev/video11	rkisp_mainpath	4672*3504, 最大支持8倍缩放	NV12/NV16/YUYV/UYVY
/dev/video12	rkisp_selfpath	1920*1080, 最大支持8倍缩放	NV12/NV16/YUYV/UYVY
/dev/video13	rkisp_ldcpath	sensor最大分辨率	NV12/NV16

【注意】

- device编号根据dts配置中sensor连接的属性 camera-module-index 值一一对应。
- rkisp_ldcpath 是 ldcv 和 ldch 开启后，输出畸变校正后的图像。
- 更多 ISP 细节参看 RK3576 datasheet 中 ISP 章节说明。

3.2 RV1106/RV1103起流、停流流程

按使用场景、sensor 数目介绍起流、停流不同应用下的使用流程，方便使用者调用。具体使用方式可参考 SDK/media/samples 中的 demo。

3.2.1 VI起停流流程

- 多摄初始化销毁停流流程

```
...// do aiq-start
RK_MPI_VI_EnableChn(pipe0, chn0)
RK_MPI_VI_EnableChn(pipe1, chn0)
RK_MPI_VI_StartPipe(pipe0)
RK_MPI_VI_startPipe(pipe1)
... // do other aiq ops
RK_MPI_VI_StopPipe(pipe0)
RK_MPI_VI_StopPipe(pipe1)
RK_MPI_VI_DisableChn(pipe0, chn0)
RK_MPI_VI_DisableChn(pipe1, chn0)
...// do aiq-destroy
```

- Pipe起流停流流程

```
...
RK_MPI_VI_StopPipe(pipe0)
RK_MPI_VI_StopPipe(pipe1)
... // do aiq-restart
RK_MPI_VI_StartPipe(pipe0)
RK_MPI_VI_StartPipe(pipe1)
...
```

- Chn起流停流流程

```
...
RK_MPI_VI_PauseChn(pipe0, chn0)
RK_MPI_VI_PauseChn(pipe0, chn1)
... // do aiq-restart
RK_MPI_VI_ResumeChn(pipe0, chn0)
RK_MPI_VI_ResumeChn(pipe0, chn1)
...
```

3.2.2 dev离线模式配置注意事项

- Vi Dev离线模式仅支持RV1106平台
- 当Dev离线模式时会接管Cif和Isp之间数据流, 和aiq存在冲突, 若要同时使用两者需aiq支持
rk_aiq_uapi2_sysctl_preInit_rkrawstream_info接口配置, 此接口用于告诉aiq数据流由外部接管, aiq只执行图像效果功能。配置流程如下

```
rk_aiq_static_info_t aiq_static_info = {0};
rk_aiq_uapi2_sysctl_enumStaticMetasByPhyId(s32CamId, &aiq_static_info);
rk_aiq_rkrawstream_info_t info = {0};
info.width = u32SensorWidth;
info.height = u32SensorHeight;
info.format = V4L2_PIX_FMT_SBGGR10;
info.mode = RK_ISP_RKRAWSTREAM_MODE_HALF_ONLINE;
rk_aiq_uapi2_sysctl_preInit_rkrawstream_info(aiq_static_info.sensor_info.sensor_name, &info);
```

【注意】

- 涉及VI初始化和反初始化流程, 如果有使用aiq, 在vi反初始化后aiq也需反初始化。

4. 举例

4.1 VI在线模式

```
TEST_VI_CTX_S *ctx;
ctx = reinterpret_cast<TEST_VI_CTX_S *>(malloc(sizeof(TEST_VI_CTX_S)));
memset(ctx, 0, sizeof(TEST_VI_CTX_S));

ctx->width = 1920;
ctx->height = 1080;
ctx->devId = 0;
ctx->pipeId = ctx->devId;
ctx->channelId = 1;
ctx->loopCountSet = 100;

//0. get dev config status
s32Ret = RK_MPI_VI_GetDevAttr(ctx->devId, &ctx->stDevAttr);
if (s32Ret == RK_ERR_VI_NOT_CONFIG) {
    //0-1.config dev
    s32Ret = RK_MPI_VI_SetDevAttr(ctx->devId, &ctx->stDevAttr);
    if (s32Ret != RK_SUCCESS) {
        RK_LOGE("RK_MPI_VI_SetDevAttr %x", s32Ret);
        goto __FAILED1;
    }
} else {
    RK_LOGE("RK_MPI_VI_SetDevAttr already");
}

//1.get dev enable status
s32Ret = RK_MPI_VI_GetDevsEnable(ctx->devId);
if (s32Ret != RK_SUCCESS) {
    //1-2.enable dev
    s32Ret = RK_MPI_VI_EnableDev(ctx->devId);
    if (s32Ret != RK_SUCCESS) {
        RK_LOGE("RK_MPI_VI_EnableDev %x", s32Ret);
        goto __FAILED1;
    }
    //1-3.bind dev/pipe
    ctx->stBindPipe.u32Num = 1;
    ctx->stBindPipe.PipeId[0] = ctx->pipeId;
    ctx->stBindPipe.bDataOffline = RK_FALSE; //dev work online
    ctx->stBindPipe.bUserStartPipe[0] = RK_FALSE; // vi start pipe
    s32Ret = RK_MPI_VI_SetDevBindPipe(ctx->devId, &ctx->stBindPipe);
    if (s32Ret != RK_SUCCESS) {
        RK_LOGE("RK_MPI_VI_SetDevBindPipe %x", s32Ret);
        goto __FAILED2;
    }
} else {
    RK_LOGE("RK_MPI_VI_EnableDev already");
}

//2.config channel
ctx->stChnAttr.stSize.u32Width = ctx->width;
ctx->stChnAttr.stSize.u32Height = ctx->height;
s32Ret = RK_MPI_VI_SetChnAttr(ctx->pipeId, ctx->channelId, &ctx->stChnAttr);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_VI_SetChnAttr %x", s32Ret);
    goto __FAILED2;
}

//3.enable channel
s32Ret = RK_MPI_VI_EnableChn(ctx->pipeId, ctx->channelId);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_VI_EnableChn %x", s32Ret);
    goto __FAILED2;
}

//4.save debug file
if (ctx->stDebugFile.bCfg) {
    s32Ret = RK_MPI_VI_ChnSaveFile(ctx->pipeId, ctx->channelId, &ctx->stDebugFile);
    RK_LOGE("RK_MPI_VI_ChnSaveFile %x", s32Ret);
}
```

```

while (loopCount < ctx->loopCountSet) {
    //5.get the frame
    s32Ret = RK_MPI_VI_GetChnFrame(ctx->pipeld, ctx->channelId, &ctx->stViFrame, waitTime);
    if (s32Ret == RK_SUCCESS) {
        void *data = RK_MPI_MB_Handle2VirAddr(ctx->stViFrame.pMbBlk);
    }
    //6.get the channel status
    s32Ret = RK_MPI_VI_QueryChnStatus(ctx->pipeld, ctx->channelId, &ctx->stChnStatus);
    //7.release the frame
    s32Ret = RK_MPI_VI_ReleaseChnFrame(ctx->pipeld, ctx->channelId, &ctx->stViFrame);
    if (s32Ret != RK_SUCCESS) {
        RK_LOGE("RK_MPI_VI_ReleaseChnFrame fail %x", s32Ret);
    }
    loopCount++;
} else {
    RK_LOGE("RK_MPI_VI_GetChnFrame timeout %x", s32Ret);
}
usleep(10*1000);
}

//8. disable one chn
s32Ret = RK_MPI_VI_DisableChn(ctx->pipeld, ctx->channelId);
RK_LOGE("RK_MPI_VI_DisableChn %x", s32Ret);
//9.disable dev(will disabled all chn)
__FAILED2:
s32Ret = RK_MPI_VI_DisableDev(ctx->devId);
RK_LOGE("RK_MPI_VI_DisableDev %x", s32Ret);

```

4.2 VI离线模式

```

TEST_VI_CTX_S *ctx;
ctx = reinterpret_cast<TEST_VI_CTX_S *>(malloc(sizeof(TEST_VI_CTX_S)));
memset(ctx, 0, sizeof(TEST_VI_CTX_S));

ctx->width = 1920;
ctx->height = 1080;
ctx->devId = 0;
ctx->pipeld = ctx->devId;
ctx->channelId = 1;
ctx->loopCountSet = 100;

//0. get dev config status
s32Ret = RK_MPI_VI_GetDevAttr(ctx->devId, &ctx->stDevAttr);
if (s32Ret == RK_ERR_VI_NOT_CONFIG) {
    //0-1.config dev
    s32Ret = RK_MPI_VI_SetDevAttr(ctx->devId, &ctx->stDevAttr);
    if (s32Ret != RK_SUCCESS) {
        RK_LOGE("RK_MPI_VI_SetDevAttr %x", s32Ret);
        goto __FAILED1;
    }
} else {
    RK_LOGE("RK_MPI_VI_SetDevAttr already");
}
//1.get dev enable status
s32Ret = RK_MPI_VI_GetDevsEnable(ctx->devId);
if (s32Ret != RK_SUCCESS) {
    //1-2.enable dev
    s32Ret = RK_MPI_VI_EnableDev(ctx->devId);
    if (s32Ret != RK_SUCCESS) {
        RK_LOGE("RK_MPI_VI_EnableDev %x", s32Ret);
        goto __FAILED1;
    }
    //1-3.bind dev/pipe
    ctx->stBindPipe.u32Num = 1;
    ctx->stBindPipe.Pipeld[0] = ctx->pipeld;
    ctx->stBindPipe.bDataOffline = RK_TRUE; //dev work offline
}

```

```

ctx->stBindPipe.bUserStartPipe[0] = RK_TRUE; // user start pipe
s32Ret = RK_MPI_VI_SetDevBindPipe(ctx->devId, &ctx->stBindPipe);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_VI_SetDevBindPipe %x", s32Ret);
    goto __FAILED2;
}
} else {
    RK_LOGE("RK_MPI_VI_EnableDev already");
}

//2.config pipe
VI_PIPE_ATTR_S stPipeAttr;
memset(&stPipeAttr, 0, sizeof(stPipeAttr));
stPipeAttr.u32MaxW = ctx->width;
stPipeAttr.u32MaxH = ctx->height;
stPipeAttr.enPixFmt = RK_FMT_RGB_BAYER_SBGR_10BPP;
stPipeAttr.enMemMode = VI_RAW_MEM_WORD_LOW_ALIGN;
s32Ret = RK_MPI_VI_CreatePipe(ctx->pipeId, &stPipeAttr);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_VI_CreatePipe failed %#X !!!", s32Ret);
    goto __FAILED2;
}

//3.start pipe
s32Ret = RK_MPI_VI_StartPipe(ctx->pipeId);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_VI_StartPipe failure %#x !!!", s32Ret);
    goto __FAILED1;
}

//4.config channel
ctx->stChnAttr.stSize.u32Width = ctx->width;
ctx->stChnAttr.stSize.u32Height = ctx->height;
s32Ret = RK_MPI_VI_SetChnAttr(ctx->pipeId, ctx->channelId, &ctx->stChnAttr);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_VI_SetChnAttr %x", s32Ret);
    goto __FAILED2;
}

//5.enable channel
s32Ret = RK_MPI_VI_EnableChn(ctx->pipeId, ctx->channelId);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_VI_EnableChn %x", s32Ret);
    goto __FAILED2;
}

//6.save debug file
if (ctx->stDebugFile.bCfg) {
    s32Ret = RK_MPI_VI_ChnSaveFile(ctx->pipeId, ctx->channelId, &ctx->stDebugFile);
    RK_LOGE("RK_MPI_VI_ChnSaveFile %x", s32Ret);
}

while (loopCount < ctx->loopCountSet) {
//7.get the frame
    s32Ret = RK_MPI_VI_GetChnFrame(ctx->pipeId, ctx->channelId, &ctx->stViFrame, waitTime);
    if (s32Ret == RK_SUCCESS) {
        void *data = RK_MPI_MB_Handle2VirAddr(ctx->stViFrame.pMbBlk);

//8.get the channel status
        s32Ret = RK_MPI_VI_QueryChnStatus(ctx->pipeId, ctx->channelId, &ctx->stChnStatus);

//9.release the frame
        s32Ret = RK_MPI_VI_ReleaseChnFrame(ctx->pipeId, ctx->channelId, &ctx->stViFrame);
        if (s32Ret != RK_SUCCESS) {
            RK_LOGE("RK_MPI_VI_ReleaseChnFrame fail %x", s32Ret);
        }
        loopCount++;
    } else {
        RK_LOGE("RK_MPI_VI_GetChnFrame timeout %x", s32Ret);
    }
    usleep(10*1000);
}
}

```

```

//10. disable one chn
s32Ret = RK_MPI_VI_DisableChn(ctx->pipeId, ctx->channelId);
RK_LOGE("RK_MPI_VI_DisableChn %x", s32Ret);

//11. stop pipe
s32Ret = RK_MPI_VI_StopPipe(ctx->pipeId);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_VI_StopPipe fail %x", s32Ret);
}

__FAILED1:
//12. destroy pipe
s32Ret = RK_MPI_VI_DestroyPipe(ctx->pipeId);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_VI_DestroyPipe fail %x", s32Ret);
}

//13.disable dev(will disabled all chn)
__FAILED2:
s32Ret = RK_MPI_VI_DisableDev(ctx->devId);
RK_LOGE("RK_MPI_VI_DisableDev %x", s32Ret);

```

详细测试DEMO，请参考发布文件：test_mpi_vi.cpp。

5. API 参考

该功能模块为用户提供以下API：

- [RK_MPI_VI_SetModParam](#): 设置 VI模块模式。
- [RK_MPI_VI_GetModParam](#): 获取 VI模块模式。
- [RK_MPI_VI_SetDevAttr](#): 设置 VI设备属性。
- [RK_MPI_VI_GetDevAttr](#): 获取 VI 设备属性。
- [RK_MPI_VI_EnableDev](#): 启用 VI 设备。
- [RK_MPI_VI_DisableDev](#): 禁用 VI 设备。
- [RK_MPI_VI_GetDevIsEnable](#): 获取 VI 设备是否使能。
- [RK_MPI_VI_QueryDevStatus](#): 查询VI设备当前状态。
- [RK_MPI_VI_GetDevFrame](#): 获取指定Dev端帧数据。
- [RK_MPI_VI_ReleaseDevFrame](#): 释放指定Dev获取帧数据。
- [RK_MPI_VI_CreatePipe](#): 创建PIPE。
- [RK_MPI_VI_DestroyPipe](#): 销毁PIPE。
- [RK_MPI_VI_StartPipe](#): 启用pipe中的所有通道。
- [RK_MPI_VI_StopPipe](#): 禁用pipe中的所有通道。
- [RK_MPI_VI_PipeSendFrame](#): 送帧数据给指定pipe。
- [RK_MPI_VI_GetPipelsEnable](#): 获取Pipe是否使能。
- [RK_MPI_VI_SetDevBindPipe](#): 绑定dev跟pipe。
- [RK_MPI_VI_GetDevBindPipe](#): 获取dev绑定的pipe属性。
- [RK_MPI_VI_SetChnAttr](#): 设置VI通道属性。
- [RK_MPI_VI_GetChnAttr](#): 获取VI通道属性。
- [RK_MPI_VI_EnableChn](#): 启用VI通道。
- [RK_MPI_VI_DisableChn](#): 禁用VI通道。
- [RK_MPI_VI_EnableChnExt](#): 启用VI通道扩展接口。
- [RK_MPI_VI_DisableChnExt](#): 禁用VI通道扩展接口。
- [RK_MPI_VI_PauseChn](#): 暂停指定VI通道的数据流。
- [RK_MPI_VI_ResumeChn](#): 恢复指定VI通道的数据流。
- [RK_MPI_VI_GetChnFrame](#): 获取VI通道一帧数据。
- [RK_MPI_VI_ReleaseChnFrame](#): 释放VI通道一帧数据。
- [RK_MPI_VI_ChnSaveFile](#): 保存VI通道数据。
- [RK_MPI_VI_QueryChnStatus](#): 获取VI通道状态。
- [RK_MPI_VI_GetChnFd](#): 获取VI通道对应的设备文件句柄。
- [RK_MPI_VI_CloseChnFd](#): 关闭VI通道对应的设备文件句柄。

- [RK_MPI_VI_SetChnFreeze](#): 设置VI通道视频输出冻结。
- [RK_MPI_VI_GetChnFreeze](#): 获取VI通道视频输出冻结。
- [RK_MPI_VI_SetUserPic](#): 设置VI通道使用用户自定义的图像。
- [RK_MPI_VI_EnableUserPic](#): 使能使用用户自定义图像的功能。
- [RK_MPI_VI_DisableUserPic](#): 禁止使用用户自定义图像的功能。
- [RK_MPI_VI_GetChnConnectInfo](#): 获取VI通道连接信息。
- [RK_MPI_VI_SetChnEdid](#): 设置VI通道EDID信息。
- [RK_MPI_VI_GetChnEdid](#): 获取VI通道EDID信息。
- [RK_MPI_VI_SetChnStreamCodec](#): 设置VI通道 stream codec。
- [RK_MPI_VI_GetChnStreamCodec](#): 获取VI通道 stream codec。
- [RK_MPI_VI_GetChnStream](#): 获取VI通道 stream 码流。
- [RK_MPI_VI_ReleaseChnStream](#): 释放VI通道 stream 码流。
- [RK_MPI_VI_RegChnEventCallBack](#): 注册VI通道事件回调接口。
- [RK_MPI_VI_SetChnDISConfig](#): 设置VI通道的 DIS 配置信息。
- [RK_MPI_VI_GetChnDISConfig](#): 获取VI通道的 DIS 配置信息。
- [RK_MPI_VI_SetChnDISAttr](#): 设置VI通道的 DIS 属性。
- [RK_MPI_VI_GetChnDISAttr](#): 获取VI通道的 DIS 属性。
- [RK_MPI_VI_SetSwcacConfig](#): 设置VI通道的 CAC 配置信息。
- [RK_MPI_VI_GetSwcacConfig](#): 获取VI通道的 CAC 配置信息。
- [RK_MPI_VI_GetChnWrapBufAttr](#): 获取VI通道卷绕模式buffer属性。
- [RK_MPI_VI_SetChnWrapBufAttr](#): 设置VI通道卷绕模式Buffer属性。
- [RK_MPI_VI_AttachMbPool](#): 附加MB buffer pool到VI通道。
- [RK_MPI_VI_DetachMbPool](#): 解除附加到VI通道的MB buffer pool。
- [RK_MPI_VI_GetChnMirrorFlip](#): 获取当前通道的镜像翻转属性。
- [RK_MPI_VI_SetChnMirrorFlip](#): 设置当前通道的镜像翻转属性。
- [RK_MPI_VI_GetEptz](#): 获取当前通道裁剪缩放属性。
- [RK_MPI_VI_SetEptz](#): 设置当前通道裁剪缩放属性。
- [RK_MPI_VI_SetPreAiispAttr](#): 设置Raw域Aiisp工作属性。
- [RK_MPI_VI_GetPreAiispAttr](#): 获取Raw域Aiisp工作属性。

5.1 RK_MPI_VI_SetModParam

【描述】

设置VI 模块模式，如Dev和Pipe工作模式，卷绕模式。

【语法】

```
RK_S32 RK_MPI_VI_SetModParam(const VI\_PARAM\_MOD\_S *pstModParam);
```

【参数】

参数名	描述	输入/输出
pstModParam	模块属性。 静态属性	输入

【注意】

- 仅RV1103B平台支持。
- 配置VI_DEV_PIPE_MODE带有离线模式时，不支持使用RK_MPI_VI_SetDevBindPipe设置用户态离线。

5.2 RK_MPI_VI_GetModParam

【描述】

设置VI 模块模式。

【语法】

```
RK_S32 RK_MPI_VI_GetModParam(VI\_PARAM\_MOD\_S *pstModParam);
```

【描述】

参数名	描述	输入/输出
pstModParam	模块属性。 静态属性	输出

5.3 RK_MPI_VI_SetDevAttr

【描述】

设置VI设备参数。

【语法】

```
RK_S32 RK_MPI_VI_SetDevAttr(VI\_DEV ViDev, const VI\_DEV\_ATTR\_S *pstDevAttr);
```

【参数】

参数名	描述	输入/输出
ViDev	VI设备号。 取值范围：[0, VI_MAX_DEV_NUM)。	输入
pstDevAttr	VI设备属性。 静态属性	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 同一个进程在设置设备属性之前需要先查询VI的属性是否已经设置过，同一个进程只能设置一次属性，如果要重新设置需要先禁用后再重新设置。

5.4 RK_MPI_VI_GetDevAttr

【描述】

获取VI设备属性。

【语法】

```
RK_S32 RK_MPI_VI_GetDevAttr(VI\_DEV ViDev, VI\_DEV\_ATTR\_S *pstDevAttr);
```

【参数】

参数名	描述	输入/输出
ViDev	音频设备号。 取值范围：[0, VI_MAX_DEV_NUM)。	输入
pstDevAttr	VI设备属性。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 获取的属性为前一次配置的属性。
- 如果从未配置过属性，则返回属性未配置的错误。

5.5 RK_MPI_VI_EnableDev

【描述】

启用VI设备。

【语法】

```
RK_S32 RK_MPI_VI_EnableDev(VI\_DEV ViDev);
```

【参数】

参数名	描述	输入/输出
ViDev	VI设备号。 取值范围: [0, VI_MAX_DEV_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 要求在启用前配置 VI 设备属性，否则会返回属性未配置的错误。
- 如果 VI 设备已经启用，重复启用，则返回正在使用中的错误。

5.6 RK_MPI_VI_DisableDev

【描述】

禁用VI设备。

【语法】

```
RK_S32 RK_MPI_VI_DisableDev(VI\_DEV ViDev);
```

【参数】

参数名	描述	输入/输出
ViDev	VI设备号。 取值范围: [0, VI_MAX_DEV_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为VI错误码。

【注意】

- 如果 VI 设备未启用，则返回未启用的VI错误码。
- 禁用 VI 设备会禁用设备下所有 VI 通道。如果只有关闭一个通道，调用关闭通道函数即可。

5.7 RK_MPI_VI_SetDevBindPipe

【描述】

设置 VI 设备与 PIPE 的绑定关系。

【语法】

```
RK_S32 RK_MPI_VI_SetDevBindPipe(VI\_DEV ViDev, const VI\_DEV\_BIND\_PIPE\_S *pstDevBindPipe);
```

【参数】

参数名	描述	输入/输出
ViDev	VI设备号。 取值范围: [0, VI_MAX_DEV_NUM)。	输入
pstDevBindPipe	绑定到 Dev 的PIPE 信息的结构体指针。 静态属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- 必须先使能 VI 设备后才能绑定PIPE。
- 不支持动态绑定。
- 调用此接口时, 当[VI_DEV_BIND_PIPE_S](#)中bUserStartPipe[ViPipe]属性为RK_FALSE, vi默认启动pipe(即用户不需要调用此接口启动pipe); 当bUserStartPipe[ViPipe]属性为RK_TRUE时, 用户需要调用[RK_MPI_VI_StartPipe](#)接口启动pipe流。
- Dev离线时, 若要Vi启动Pipe(即中bUserStartPipe[ViPipe] = RK_TRUE), 则调用此接口前需先将Pipe创建[RK_MPI_VI_CreatePipe](#).

5.8 RK_MPI_VI_GetDevBindPipe

【描述】

获取 VI 设备与 PIPE 的绑定关系。

【语法】

```
RK_S32 RK_MPI_VI_GetDevBindPipe(VI\_DEV ViDev, VI\_DEV\_BIND\_PIPE\_S *pstDevBindPipe);
```

【参数】

参数名	描述	输入/输出
ViDev	VI设备号。 取值范围: [0, VI_MAX_DEV_NUM)。	输入
pstDevBindPipe	绑定到 Dev 的PIPE 信息的结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- 使用本接口前，需先配置 DEV 属性，使能设备并绑定设备跟PIPE，否则返回失败。

5.9 RK_MPI_VI_GetDevIsEnable

【描述】

获取设备是否使能

【语法】

```
RK_S32 RK_MPI_VI_GetDevIsEnable(VI\_DEV ViDev);
```

【参数】

参数名	描述	输入/输出
ViDev	VI设备号。 取值范围：[0, VI_MAX_DEV_NUM)。	输入

【返回值】

返回值	描述
0	已经使能。
非0	未使能。

5.10 RK_MPI_VI_QueryDevStatus

【描述】

查询Sensor设备的状态。

【语法】

```
RK_S32 RK_MPI_VI_QueryDevStatus(VI\_DEV Videv, VI\_DEV\_STATUS\_S *pstDevStatus);
```

【参数】

参数名	描述	输入/输出
ViDev	VI设备号。 取值范围：[0, VI_MAX_DEV_NUM)。	输入
pstDevStatus	Sensor在线状态的结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 该接口通过是否成功获取到Sensor名称来判断Sensor是否在线。
- 不支持定时查询Sensor的状态。即Sensor一开始在线后，再次调用该接口仍获取的是在线的状态，无法实时获取到Sensor离线状态。
- 该接口需在VI设备使能后调用，否则获取设备状态会失败。
- 该接口可获取sensor最大原始输出分辨率大小

5.11 RK_MPI_VI_GetDevFrame

【描述】

获取指定Dev端帧数据

【语法】

```
RK_S32 RK_MPI_VI_GetDevFrame(VI\_DEV Videv, VIDEO\_FRAME\_INFO\_S *pstVideoFrame, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
ViDev	VI设备号。 取值范围: [0, VI_MAX_DEV_NUM)。	输入
pstVideoFrame	输出图像的帧信息结构指针。	输出
s32MilliSec	获取数据的超时时间, -1表示阻塞模式; 0表示非阻塞模式; >0表示阻塞 s32MilliSec毫秒, 超时则报错返回。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- 此接口适用于dev和pipe非绑定状态下使用(dev, pipe均离线), 即在不调用RK_MPI_VI_SetDevBindPipe时可用此接口获取dev端数据。
- 此接口需要配合[RK_MPI_VI_ReleaseDevFrame](#)使用, 释放获取到的帧数据
- 此接口仅在dev为非HDR模式时使用

5.12 RK_MPI_VI_ReleaseDevFrame

【描述】

释放指定Dev端帧数据

【语法】

```
RK_S32 RK_MPI_VI_ReleaseDevFrame(VI\_DEV Videv, VIDEO\_FRAME\_INFO\_S *pstVideoFrame);
```

参数名	描述	输入/输出
ViDev	VI设备号。 取值范围: [0, VI_MAX_DEV_NUM)。	输入
pstVideoFrame	输入图像的帧信息结构指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- 此接口配合[RK_MPI_VI_GetDevFrame](#)使用.

5.13 RK_MPI_VI_CreatePipe

【描述】

创建Pipe

【语法】

```
RK_S32 RK_MPI_VI_CreatePipe(VI\_PIPE ViPipe, const VI_PIPE_ATTR_S *pstPipeAttr);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
pstPipeAttr	pipe属性, 静态属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

5.14 RK_MPI_VI_DestroyPipe

【描述】

销毁当前pipe。

【语法】

```
RK_S32 RK_MPI_VI_DestroyPipe(VI\_PIPE ViPipe);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

5.15 RK_MPI_VI_StartPipe

【描述】

启动当前pipe数据流。

【语法】

```
RK_S32 RK_MPI_VI_StartPipe(VI\_PIPE ViPipe);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- 此接口调用和[VI_DEV_BIND_PIPE_S](#) 中的bUserStartPipe[ViPipe]相关, 当bUserStartPipe[ViPipe]为RK_TRUE时, 需调用[RK_MPI_VI_StartPipe](#)启用Pipe流; 当bUserStartPipe[ViPipe]为RK_FALSE时, 则由Vi模块在调用[RK_MPI_VI_SetDevBindPipe](#)时内部启用Pipe流; Dev初始化时若没有调RK_MPI_VI_SetDevBindPipe来绑定pipe(Dev, Pipe离线状态), 同需用户去调用RK_MPI_VI_StartPipe启Pipe流.
- 和[RK_MPI_VI_StopPipe](#)接口成对使用

5.16 RK_MPI_VI_StopPipe

【描述】

暂定当前pipe数据流。

【语法】

RK_S32 RK_MPI_VI_StopPipe([VI_PIPE](#) ViPipe);

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- 和[RK_MPI_VI_StartPipe](#)接口成对使用

5.17 RK_MPI_VI_PipeSendFrame

【描述】

送帧数据给指定pipe

【语法】

RK_S32 RK_MPI_VI_PipeSendFrame([VI_PIPE](#) ViPipe, [VIDEO_FRAME_INFO_S](#) *pstVideoFrame, RK_S32 s32MilliSec)

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
pstVideoFrame	输入图像的帧信息结构指针。	输入
s32MilliSec	获取数据的超时时间, -1表示阻塞模式; 0表示非阻塞模式; >0表示阻塞 s32MilliSec毫秒, 超时则报错返回。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- 适用于dev和pipe非绑定时使用

5.18 RK_MPI_VI_GetPipesEnable

【描述】

获取指定Pipe是否使能。

【语法】

RK_S32 RK_MPI_VI_GetPipesEnable([VI_PIPE](#) ViPipe)

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

5.19 RK_MPI_VI_SetPreAiispAttr

【描述】

设置raw域aiisp 属性

【语法】

RK_S32 RK_MPI_VI_SetPreAiispAttr(VI_PIPE ViPipe, const VI_AIISP_INFO_S *pstAiispAttr)

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
pstAiispAttr	aiisp属性结构体指针。 静态属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 仅在Pipe离线时支持

5.20 RK_MPI_VI_GetPreAiispAttr

【描述】

获取raw域aiisp属性

【语法】

```
RK_S32 RK_MPI_VI_GetPreAiispAttr(VI_PIPE ViPipe, VI_AIISP_INFO_S *pstAiispAttr);
```

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
pstAiispAttr	aiisp属性结构体指针。 静态属性。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 仅在Pipe离线时支持

5.21 RK_MPI_VI_SetChnAttr

【描述】

设置 VI 通道属性。

【语法】

```
RK_S32 RK_MPI_VI_SetChnAttr(VI_PIPE ViPipe, VI_CHN ViChn, const VI_CHN_ATTR_S *pstChnAttr);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM)。	输入
pstChnAttr	VI 通道属性结构体指针。 静态属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

通道属性的各项配置限制如下：

- 目标图像大小 stSize：
必须配置，且大小需要在 VI 支持的范围内且不超过 VI 的缩放倍数限制。
- 通道buff类型enBufType：
当采集数据类型为外部申请的内存时，配置为VI_ALLOC_BUF_TYPE_EXTERNAL。内部申请内存时配置为VI_ALLOC_BUF_TYPE_INTERNAL。
- 采集数据选项stIspOpt，当采集的数据为isp输入或者直通时，需要配置：
 1. aEntityName：当数据类型为isp直通型时，需要配置。eg: /dev/video0
 2. stMaxSize：必须配置，ISP采集通道输出的最大分辨率。
- 通道buff数量
RV1106/RV1103 平台，非卷绕模式下，各码流通道的buffer个数需大于等于2个，否则可能会造成撕裂现象。
- PIPE 必须已绑定到设备，否则会返回失败。
- Mirror/Flip：
RV1106/RV1103 芯片，若通道0 (mainpath 通道) 设置了卷绕模式，则不支持 Mirror/Flip 功能。
Mirror 功能是所有通道共用的配置，是全局的。
Flip 功能可以不同通道独立配置。
- 帧率控制stFrameRate：
原始帧率 s32SrcFrameRate：如果不进行帧率控制，则该值设置为-1。
目标帧率 s32DstFrameRate：如果不进行帧率控制，则该值设置为-1，否则必须设置原始帧率 s32SrcFrameRate 的值，且目标帧率必须小于或等于原始帧率。

5.22 RK_MPI_VI_GetChnAttr

【描述】

获取 VI 通道属性。

【语法】

```
RK_S32 RK_MPI_VI_GetChnAttr(VI\_PIPE ViPipe, VI\_CHN ViChn, VI\_CHN\_ATTR\_S *pstChnAttr);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM]。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 启用VI通道前，必须先启用其所属的VI设备，否则返回设备未启动的VI错误码。
- 通道属性需先设置后，才可获取。

5.23 RK_MPI_VI_EnableChn

【描述】

启用VI通道。

【语法】

```
RK_S32 RK_MPI_VI_EnableChn(VI\_PIPE ViPipe, VI\_CHN ViChn);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM]。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 启用VI通道前，必须先启用其所属的VI设备，否则返回设备未启动的VI错误码。

5.24 RK_MPI_VI_DisableChn

【描述】

禁用 VI 通道。

【语法】

```
RK_S32 RK_MPI_VI_DisableChn(VI\_PIPE ViPipe, VI\_CHN ViChn);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM]。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- PIPE 必须已绑定设备，否则会返回失败。

5.25 RK_MPI_VI_EnableChnExt

【描述】

启用VI通道扩展接口。

【语法】

```
RK_S32 RK_MPI_VI_EnableChnExt(VI\_PIPE ViPipe, VI\_CHN ViChn);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 该接口只配置了当前通道的相关资源与上下出流环境，并未真正起流。起流需配合调用 RK_MPI_VI_StartPipe 接口，在本接口之后调用 RK_MPI_VI_StartPipe 接口。
- 启用VI通道前，必须先启用其所属的 VI 设备，否则返回设备未启动的VI错误码。

5.26 RK_MPI_VI_DisableChnExt

【描述】

禁用 VI 通道扩展接口。

【语法】

```
RK_S32 RK_MPI_VI_DisableChnExt(VI\_PIPE ViPipe, VI\_CHN ViChn);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 该接口只关闭或停止当前通道的相关资源与上下出流环境，并未真正停流操作。停流需配合调用 RK_MPI_VI_StopPipe 接口，需在调用该接口之前调用 RK_MPI_VI_StopPipe 接口。
- PIPE 必须已绑定设备，否则会返回失败。

5.27 RK_MPI_VI_PauseChn

【描述】

暂停指定VI通道的数据流。

【语法】

```
RK_S32 RK_MPI_VI_PauseChn(VI\_PIPE ViPipe, VI\_CHN ViChn);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- PIPE 必须已绑定设备，否则会返回失败。

5.28 RK_MPI_VI_ResumeChn

【描述】

恢复指定VI通道的数据流。

【语法】

```
RK_S32 RK_MPI_VI_ResumeChn(VI\_PIPE ViPipe, VI\_CHN ViChn);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- PIPE 必须已绑定设备，否则会返回失败。

5.29 RK_MPI_VI_GetChnFrame

【描述】

从VI通道获取采集的图像。

【语法】

```
RK_S32 RK_MPI_VI_GetChnFrame(VI\_PIPE ViPipe, VI\_CHN ViChn, VIDEO\_FRAME\_INFO\_S *pstFrameInfo, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM)。	输入
pstFrameInfo	输出图像的帧信息结构指针。	输入
s32MilliSec	获取数据的超时时间, -1表示阻塞模式； 0表示非阻塞模式；>0表示阻塞 s32MilliSec毫秒， 超时则报错返回。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- PIPE 必须已绑定设备，否则会返回失败。
- s32MilliSec 的值必须大于等于-1，等于-1 时采用阻塞模式发送数据，等于 0 时采用非阻塞模式发送数据，大于 0 时，阻塞 s32MilliSec 毫秒后，则返回超时并报错。
- 获取的缓存信息来自VI内部使用的 MediaBuffer，因此使用完之后，必须要调用 RK_MPI_VI_ReleaseChnFrame接口释放其内存。
- 如果通过RK_MPI_SYS_Bind将VI绑定到了其他设备，则此接口将获取不到数据。

5.30 RK_MPI_VI_ReleaseChnFrame

【描述】

释放一帧从VI通道获取的图像。

【语法】

```
RK_S32 RK_MPI_VI_ReleaseChnFrame(VI\_PIPE ViPipe, VI\_CHN ViChn, const VIDEO\_FRAME\_INFO\_S *pstFrameInfo);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
pstFrameInfo	输出图像的帧信息结构指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- PIPE 必须已绑定设备, 否则会返回失败。
- 此接口必须与RK_MPI_VI_GetChnFrame配对使用。

5.31 RK_MPI_VI_ChnSaveFile

【描述】

保存 VI 通道数据。

【语法】

```
RK_S32 RK_MPI_VI_ChnSaveFile(VI\_PIPE ViPipe, VI\_CHN ViChn, VI\_SAVE\_FILE\_INFO\_S*pstSaveFileInfo);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
pstSaveFileInfo	VI通道数据保存信息结构指针。	

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- PIPE 必须已绑定设备, 否则会返回失败。
- 如果通过RK_MPI_SYS_Bind将VI绑定到了其他设备, 则此接口将保存不了数据。

5.32 RK_MPI_VI_QueryChnStatus

【描述】

查询 VI 通道的状态。

【语法】

```
RK_S32 RK_MPI_VI_QueryChnStatus(VI\_PIPE ViPipe, VI\_CHN ViChn, VI\_CHN\_STATUS\_S*pstChnStatus);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
pstChnStatus	VI 通道状态的指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- PIPE 必须已绑定设备, 否则会返回失败。
- 如果通过RK_MPI_SYS_Bind将VI绑定到了其他设备, 则此接口获取不到帧率数据、丢帧数等统计数据。

5.33 RK_MPI_VI_GetChnFd

【描述】

获取VI通道对应的设备文件句柄。通过此接口返回值可以使用select/poll查询对应通道的数据状态。

【语法】

```
RK_S32 RK_MPI_VI_GetChnFd(VI\_PIPE ViPipe, VI\_CHN ViChn);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
非负数	成功, VI通道的设备文件句柄。
负数	失败, 其值为 VI错误码 。

【注意】

- 关闭文件句柄需要调用[RK_MPI_VI_CloseChnFd](#)。
- 推荐在启用通道前进行获取fd。
- 不推荐在不关数据流的情况下频繁开关fd。

5.34 RK_MPI_VI_CloseChnFd

【描述】

关闭VI通道对应的设备文件句柄。

【语法】

```
RK_S32 RK_MPI_VI_CloseChnFd(VI\_PIPE ViPipe, VI\_CHN ViChn);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 无

5.35 RK_MPI_VI_SetChnFreeze

【描述】

设置VI通道视频输出冻结。VI模块输出的YUV画面保持不变，输出帧率等同于sensor输出帧率。

【语法】

```
RK_S32 RK_MPI_VI_SetChnFreeze(VI\_PIPE ViPipe, VI\_CHN ViChn, RK_BOOL bFreeze);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
bFreeze	是否使能视频输出冻结功能。 RK_TRUE: 使能 RK_FALSE: 不使能 动态属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 无

5.36 RK_MPI_VI_GetChnFreeze

【描述】

获取VI通道视频输出冻结使能状态。

【语法】

```
RK_S32 RK_MPI_VI_GetChnFreeze(VI\_PIPE ViPipe, VI\_CHN ViChn, RK_BOOL *pbFreeze);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
pbFreeze	是否使能视频输出冻结功能。 RK_TRUE: 使能 RK_FALSE: 不使能	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- 无

5.37 RK_MPI_VI_SetUserPic

【描述】

设置VI通道使用用户自定义的图像。

【语法】

```
RK_S32 RK_MPI_VI_SetUserPic(VI\_PIPE ViPipe, VI\_CHN ViChn, const VI\_USERPIC\_ATTR\_S*pstUsrPic);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
pstUsrPic	用户图片信息结构指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 本接口设置的用户图片对所有 PIPE 均有效。
- 此接口支持两种用户图片模式，一种是 YUV 图像模式，另一种是纯色背景模式。
- 设置 VI 用户图片，用于将 VI 通道输出的视频帧图像修改为用户配置的指定 YUV 图像数据或纯色背景，而不是 codec 的视频数据；此接口需要与[RK_MPI_VI_EnableUserPic](#)、[RK_MPI_VI_DisableUserPic](#)配合使用。
- 设置完用户图片后，即可调用[RK_MPI_VI_EnableUserPic](#)接口对指定 VI PIPE 启用插入用户图片，此时 VI PIPE 的通道输出的数据即为所配置的图片 YUV 数据或纯色背景；一般用于视频信号丢失时，VI PIPE 的通道输出 NoVideo 图片。配置的用户图片的宽高一般应该与 VI PIPE 大小相一致；如果不一致，VI 内部会自动将其缩放为 VI PIPE 大小。
- 用户图片的帧率目前暂固定为 30 帧。
- VI 模块在启用插入用户图片时，会直接使用设置的用户图片帧信息中的物理地址，因此设置完用户图片后，不应该释放或销毁其视频缓存块，除非确认不再使用。可以通过再次调用此接口设置另外一块 VideoBuffer 以修改图片信息。
- 图像模式送下来的 YUV 帧信息必须为真实有效的 VB。
- 图像模式 YUV 数据像素格式支持 RK_FMT_YUV420SP。
- 图像模式 YUV 数据 enCompressMode 必须为非压 [COMPRESS_MODE_NONE](#)。
- 图像模式 YUV 数据宽度必须 2 对齐，高度必须 2 对齐。

5.38 RK_MPI_VI_EnableUserPic

【描述】

使能使用用户自定义图像的功能。

【语法】

```
RK_S32 RK_MPI_VI_EnableUserPic((VI\_PIPE ViPipe, VI\_CHN ViChn)
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM]。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 启用插入用户图片之前，需要先设置用户图片帧信息
- 启用插入用户图片之后，当前 VI PIPE 的通道即输出配置的用户图片帧数据。
- PIPE 必须已创建。
- 此接口可以重复调用。

5.39 RK_MPI_VI_DisableUserPic

【描述】

禁止使用用户自定义图像的功能。

【语法】

```
RK_S32 RK_MPI_VI_DisableUserPic((VI\_PIPE ViPipe, VI\_CHN ViChn)
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- VI 通道不再需要输出用户图片时, 应该调用此接口以恢复输出 AD 的原始视频数据。
- 此接口可以重复调用。
- PIPE 必须已创建。

5.40 RK_MPI_VI_GetChnConnectInfo

【描述】

获取VI通道连接信息。

【语法】

```
RK_S32 RK_MPI_VI_GetChnConnectInfo((VI\_PIPE ViPipe, VI\_CHN ViChn, VI\_CONNECT\_INFO\_S *pstConnectInfo)
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
pstConnectInfo	VI通道连接信息结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- connect状态目前仅RK3588 hdmi rx支持查询，其他情况均返回VI_CONNECT_STATE_UNKNOWN。
 - 可查询hdmi rx插拔后重新获取输入帧率的数据。
 - 必须在[RK_MPI_VI_SetChnAttr](#)函数调用后才支持正常获取，否则返回RK_ERR_VI_NOT_CONFIG。
 - 此接口可以重复调用。
 - PIPE 必须已创建。

5.41 RK_MPI_VI_SetChnEdid

【描述】

设置VI通道EDID信息。

【语法】

RK_S32 RK_MPI_VI_SetChnEdid((VI_PIPE ViPipe, VI_CHN ViChn, const VI_EDID_S *pstEdid))

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
pstEdid	VI通道EDID信息结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- EDID信息仅hdmi rx支持设置。
 - 必须在[RK_MPI_VI_SetChnAttr](#)函数调用后才支持正常获取，否则返回RK_ERR_VI_NOT_CONFIG。
 - 此接口可以重复调用。
 - PIPE 必须已创建。

【举例】

```

};

VI_EDID_S stEdid;
memset(&stEdid, 0, sizeof(VI_EDID_S));
stEdid.u32Blocks = 2;
stEdid.pu8Edid = (RK_U8 *)test_edid;
s32Ret = RK_MPI_VI_SetChnEdid(ctx->pipeId, ctx->channelId, &stEdid);

```

5.42 RK_MPI_VI_GetChnEdid

【描述】

获取VI通道EDID信息。

【语法】

```
RK_S32 RK_MPI_VI_GetChnEdid((VI\_PIPE ViPipe, VI\_CHN ViChn, VI\_EDID\_S *pstEdid)
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM)。	输入
pstEdid	VI通道EDID信息结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- EDID信息仅hdmi rx支持查询。
- pstEdid的pu8Edid需要按照需要查询的u32Blocks个数申请，每个block的内存大小为128字节，总内存大小为u32Blocks*128。
- 必须在[RK_MPI_VI_SetChnAttr](#)函数调用后才支持正常获取，否则返回RK_ERR_VI_NOT_CONFIG。
- 此接口可以重复调用。
- PIPE 必须已创建。

【举例】

```

VI_EDID_S stEdid;
memset(&stEdid, 0, sizeof(VI_EDID_S));
stEdid.u32Blocks = 3;
stEdid.pu8Edid = (RK_U8 *)calloc(128 * stEdid.u32Blocks, 1);
s32Ret = RK_MPI_VI_GetChnEdid(ctx->pipeId, ctx->channelId, &stEdid);
free(stEdid.pu8Edid);

```

5.43 RK_MPI_VI_SetChnStreamCodec

【描述】

设置VI通道 stream codec。

【语法】

```
RK_S32 RK_MPI_VI_SetChnStreamCodec(VI\_PIPE ViPipe, VI\_CHN ViChn, const RK_CODEC_ID_E enCodecId);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
enCodecId	stream codec码流格式。 取值范围: RK_VIDEO_ID_Unused: 不设置 RK_VIDEO_ID_MJPEG: MJPEG 码流 RK_VIDEO_ID_AVC: H264 码流 RK_VIDEO_ID_HEVC: H265 码流	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- PIPE 必须已绑定设备, 否则会返回失败。
- 必须在RK_MPI_VI_SetChnAttr之后, RK_MPI_VI_EnableChn之前设置。
- 此功能用于设置需要获取UVC设备的编码码流格式。

5.44 RK_MPI_VI_GetChnStreamCodec

【描述】

恢复指定VI通道的数据流。

【语法】

```
RK_S32 RK_MPI_VI_GetChnStreamCodec(VI\_PIPE ViPipe, VI\_CHN ViChn, RK_CODEC_ID_E *penCodecId);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
penCodecId	stream codec码流格式。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- PIPE 必须已绑定设备, 否则会返回失败。

5.45 RK_MPI_VI_GetChnStream

【描述】

从VI通道获取采集的编码码流。

【语法】

```
RK_S32 RK_MPI_VI_GetChnStream(VI\_PIPE ViPipe, VI\_CHN ViChn, VI\_STREAM\_S *pstStream, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
pstStream	输出码流的帧信息结构指针。	输出
s32MilliSec	获取数据的超时时间, -1表示阻塞模式; 0表示非阻塞模式; >0表示阻塞 s32MilliSec毫秒, 超时则报错返回。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- PIPE 必须已绑定设备, 否则会返回失败。
- 必须通过[RK_MPI_VI_SetChnStreamCodec](#)设置enCodecId为RK_VIDEO_ID_MJPEG、RK_VIDEO_ID_AVC、RK_VIDEO_ID_HEVC中的一种格式, 才能调用此接口获取码流数据。不是所有的UVC设置都支持以上三种格式获取, 具体能够的支持的格式及分辨率需要查看UVC设备支持的属性。如果设置了UVC不支持的格式, 返回的数据会是UVC默认的格式数据。如果设置了UVC不支持的分辨率, 返回的是UVC能够支持的最接近的分辨率数据。使用v4l2-ctl命令可以查询相应节点支持的格式及分辨率信息, 这里以/dev/video0为例, 查询命令为: v4l2-ctl -d /dev/video0 --list-formats-ext。
- s32MilliSec 的值必须大于等于-1, 等于-1 时采用阻塞模式发送数据, 等于 0 时采用非阻塞模式发送数据, 大于 0 时, 阻塞 s32MilliSec 毫秒后, 则返回超时并报错。
- 获取的码流使用完之后, 必须要调用 [RK_MPI_VI_ReleaseChnStream](#)接口释放其内存。
- 如果通过[RK_MPI_SYS_Bind](#)将VI绑定到了其他设备, 需要设置[VI_CHN_ATTR_S](#)的u32Depth大于0, 否则此接口将获取不到数据。

5.46 RK_MPI_VI_ReleaseChnStream

【描述】

释放VI通道采集的编码码流内存。

【语法】

```
RK_S32 RK_MPI_VI_ReleaseChnStream(VI\_PIPE ViPipe, VI\_CHN ViChn, const VI\_STREAM\_S *pstStream);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
pstStream	码流的帧信息结构指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- PIPE 必须已绑定设备, 否则会返回失败。
- 此接口必须与RK_MPI_VI_GetChnStream配对使用。

5.47 RK_MPI_VI_RegChnEventCallback

【描述】

注册VI通道事件回调接口。

【语法】

```
RK_S32 RK_MPI_VI_RegChnEventCallback(VI\_PIPE ViPipe, VI\_CHN ViChn, const VI\_EVENT\_CALL\_BACK\_S *pstCallback);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
pstCallback	回调函数结构指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【注意】

- PIPE 必须已绑定设备, 否则会返回失败。
- 此接口实现hdmirx的插拔事件 (VI_CHANGE_EVENT_CONNECT_CHANGE) 及输入源切换事件 (VI_CHANGE_EVENT_SOURCE_CHANGE) 的回调, 收到回调事件后, 用户通过u32Event确认触发回调的事件类型, 通过调用[RK_MPI_VI_GetChnConnectInfo](#)函数获取当前的连接状态及源分辨率格式信息。
- 收到输入源切换事件后需要做一次复位或者销毁重建vi, 否则会导致无法收到数据流。如果此时源分辨率及格式跟之前设置一样, 则只需要调用[RK_MPI_VI_PauseChn](#)及[RK_MPI_VI_ResumeChn](#)即可恢复出流。如果此时源分辨率或者格式跟之前不一样, 则需要销毁重建vi。
- 回调函数中不建议处理耗时任务, 不能处理阻塞任务, 否则会影响后续相关事件通知。

【举例】

```

RK_S32 s32Ret = RK_FAILURE;
VI_EVENT_CALL_BACK_S stCallbackFunc;
stCallbackFunc.pfnCallback = test_vi_event_call_back;
stCallbackFunc.pPrivateData = ctx;
s32Ret = RK_MPI_VI_RegChnEventCallBack(ctx->pipeId, ctx->channelId, &stCallbackFunc);
...
// 回调函数实现
static void test_vi_event_call_back(RK_VOID *pPrivateData, VI_CB_INFO_S *info) {
    TEST_VI_CTX_S *ctx = (TEST_VI_CTX_S *)pPrivateData;
    if (info) {
        if (info->u32Event & VI_CHANGE_EVENT_CONNECT_CHANGE)
            RK_LOGI("event connect change");
        if (info->u32Event & VI_CHANGE_EVENT_SOURCE_CHANGE) {
            RK_LOGI("event source change");
            test_vi_event_source_change(ctx);
        }
    }
}

```

5.48 RK_MPI_VI_SetChnDISConfig

【描述】

设置 VI 通道的 DIS 配置信息。

【语法】

RK_S32 RK_MPI_VI_SetChnDISConfig(([VI_PIPE](#) ViPipe, [VI_CHN](#) ViChn, const [DIS_CONFIG_S](#) *pstDISConfig)

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
pstDISConfig	DIS 配置信息结构体。静态参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【芯片差异】

芯片类型	是否支持 DIS_MODE_GYRO
RK3588	支持
RK356x RV1109/RV1126	不支持。

【注意】

- 仅 RK3588 平台支持该接口。
- PIPE 必须已创建，否则会返回失败。
- 在 DIS 使能且 bScale 为 TRUE 时，crop_ratio 参数可以动态修改，其他情况要修改 pstDISConfig 中参数时，必须先关闭DIS 再修改参数，不可动态修改 pstDISConfig 中参数，不可动态修改。

【相关主题】

- [RK_MPI_VI_GetChnDISConfig](#)

【需求】

- 头文件：rk_comm_dis.h
- 库文件：librkAlgoDis.so

5.49 RK_MPI_VI_GetChnDISConfig

【描述】

获取 VI 通道的 DIS 配置信息。

【语法】

```
RK_S32 RK_MPI_VI_GetChnDISConfig((VI\_PIPE ViPipe, VI\_CHN ViChn, const DIS\_CONFIG\_S *pstDISConfig)
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM)。	输入
pstDISConfig	DIS 配置信息结构体。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【芯片差异】

芯片类型	是否支持 DIS_MODE_GYRO
RK3588	支持
RK356x RV1109/RV1126	不支持。

【注意】

- 仅 RK3588 平台支持该接口。
- PIPE 必须已创建，否则会返回失败。

【相关主题】

- [RK_MPI_VI_SetChnDISConfig](#)

5.50 RK_MPI_VI_SetChnDISAttr

【描述】

设置 VI 通道的 DIS 属性。

【语法】

```
RK_S32 RK_MPI_VI_SetChnDISAttr((VI\_PIPE ViPipe, VI\_CHN ViChn, const DIS\_ATTR\_S *pstDISAttr)
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
pstDISAttr	DIS 配置信息结构体。动态参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【芯片差异】

芯片类型	是否支持 DIS_MODE_GYRO
RK3588	支持
RK356x RV1109/RV1126	不支持。

【注意】

- 仅 RK3588 平台支持该接口。
- PIPE 必须已创建，否则会返回失败。
- 必须先使能 VI 通道和设置 DIS 配置后，才能设置 DIS 属性。

【相关主题】

- [RK_MPI_VI_SetChnDISConfig](#)
- [RK_MPI_VI_GetChnDISConfig](#)

【需求】

- 头文件: rk_comm_dis.h
- 库文件: librkAlgoDis.so

5.51 RK_MPI_VI_GetChnDISAttr

【描述】

获取 VI 通道的 DIS 属性。

【语法】

```
RK_S32 RK_MPI_VI_GetChnDISConfig(VI\_PIPE ViPipe, VI\_CHN ViChn, const DIS\_CONFIG\_S * pstDISConfig)
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
pstDISConfig	DIS 配置信息结构体。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【芯片差异】

芯片类型	是否支持 DIS_MODE_GYRO
RK3588	支持
RK356x RV1109/RV1126	不支持。

【注意】

- 仅 RK3588 平台支持该接口。
- PIPE 必须已创建，否则会返回失败。
- 建议先设置 DIS 属性后，再使用该接口。

【相关主题】

- [RK_MPI_VI_SetChnDISAttr](#)

5.52 RK_MPI_VI_SetSwcacConfig

【描述】

设置 VI 通道的 CAC 配置信息。

【语法】

```
RK_S32 RK_MPI_VI_SetSwcacConfig(VI\_PIPE ViPipe, VI\_CHN ViChn, const SWCAC\_CONFIG\_S *pstSwcacCfg)
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM)。	输入
pstSwcacCfg	CAC 配置信息结构体。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 仅 RK3588 平台支持该接口。
- PIPE 必须已创建，否则会返回失败。
- 仅支持 RK_FMT_YUV420SP 像素格式。
- 不支持 COMPRESS_AFBC_16x16 压缩格式。
- pstSwcacCfg 中的各项参数均需要合法，否则会返回失败。

【相关主题】

- [RK_MPI_VI_GetSwcacConfig](#)

【需求】

- 头文件：rk_comm_swcac.h。
- 库文件：libRkSwCac.so。

5.53 RK_MPI_VI_GetSwcacConfig

【描述】

获取 VI 通道的 CAC 配置信息。

【语法】

```
RK_S32 RK_MPI_VI_GetSwcacConfig(VI\_PIPE ViPipe, VI\_CHN ViChn, SWCAC\_CONFIG\_S *pstSwcacCfg)
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM)。	输入
pstSwcacCfg	CAC 配置信息结构体。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【注意】

- 仅 RK3588 平台支持该接口。
- PIPE 必须已创建，否则会返回失败。

【相关主题】

- [RK_MPI_VI_SetSwcacConfig](#)

5.54 RK_MPI_VI_GetChnWrapBufAttr

【描述】

获取VI通道卷绕buffer的属性。

【语法】

```
RK_S32 RK_MPI_VI_GetChnWrapBufAttr(VI\_PIPE ViPipe, VI\_CHN ViChn, VI\_CHN\_BUF\_WRAP\_S *pstViWrap);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
VI_CHN_BUF_WRAP_S	VI通道buffer卷绕属性。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【芯片差异】

芯片	是否支持该功能
RV1126/RV1109	不支持
RK356x	不支持
RK3588	不支持
RV1106/RV1103	支持

【注意】

- 仅支持对VI rkisp_mainpath通道操作。

5.55 RK_MPI_VI_SetChnWrapBufAttr

【描述】

设置VI通道卷绕buffer的属性。

【语法】

```
RK_S32 RK_MPI_VI_SetChnWrapBufAttr(VI\_PIPE ViPipe, VI\_CHN ViChn, const VI\_CHN\_BUF\_WRAP\_S*pstViWrap);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
VI_CHN_BUF_WRAP_S	VI通道buffer卷绕属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【芯片差异】

芯片	是否支持该功能
RV1126/RV1109	不支持
RK356x	不支持
RK3588	不支持
RV1106/RV1103	支持

【注意】

- 仅支持对VI rkisp_mainpath 通道进行操作。
- 卷绕模式中的 BufLine 需小于输出图像高度，建议最小卷绕行数不小于128行。
- 卷绕模式中的 BufLine 必须能被输出图像的高度整除。
- 卷绕模式中的 BufLine 和 CPU负载，SENSOR VBLANK（建议7MS左右），子码流大小，子码流个数，DDR带宽有关。
当出现编码器OVERFLOW时，请先检查以上因素，必要时开大BufLine。
- 使能卷绕时，u32WrapBufferSize 必须大于 0。
- 使用卷绕模式编码时，不能同时从VI通道中获取视频流。
- 使用多个 sensor 时，不支持开启卷绕模式。
- 设置卷绕属性满足VI、VENC同时设置卷绕，VI需要 bind VENC。
- 使用卷绕模式编码时，BufLine 卷绕行数、BufferSize 卷绕大小配置需要和VENC卷绕属性的配置保持一致。
- 使用卷绕模式时，不支持设置 isp 的输出帧率。

5.56 RK_MPI_VI_AttachMbPool

【描述】

将 VI PIPE 的通道绑定到某个视频缓存 MB 池中。

【语法】

```
RK_S32 RK_MPI_VI_AttachMbPool(VI\_PIPE ViPipe, VI\_CHN ViChn, MB_POOL hMbPool);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM)。	输入
hMbPool	视频缓存MB池信息	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【芯片差异】

芯片	是否支持该功能
RV1126/RV1109	不支持
RK356x	不支持
RK3588	不支持
RV1106/RV1103	支持

【注意】

- PIPE 必须已创建。
- 用户必须调用接口 RK_MPI_MB_CreatePool 创建一个视频缓存 MB 池，再通过调用接口 RK_MPI_VI_AttachMbPool 把当前组的通道绑定到固定 PoolId 的 MB 池中。
- hMbPool 必须保证是已创建 MB 池的有效 PoolId。

5.57 RK_MPI_VI_DetachMbPool

【描述】

将 VI PIPE 的通道从某个视频缓存 VB 池中解绑定。

【语法】

```
RK_S32 RK_MPI_VI_DetachMbPool(VI\_PIPEViPipe, VI\_CHNViChn);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【芯片差异】

芯片	是否支持该功能
RV1126/RV1109	不支持
RK356x	不支持
RK3588	不支持
RV1106/RV1103	支持

【注意】

PIPE 必须已创建。

5.58 RK_MPI_VI_GetChnMirrorFlip

【描述】

获取当前 VI 通道镜像、翻转的信息。

【语法】

```
RK_S32 RK_MPI_VI_GetChnMirrorFlip(VI\_PIPEViPipe, VI\_CHNViChn, VI\_ISP\_MIRROR\_FLIP\_S *pstMirrFlip);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
*pstMirrFlip	镜像旋转结构体。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【芯片差异】

芯片	是否支持该功能
RV1126/RV1109	不支持
RK356x	不支持
RK3588	不支持
RV1106/RV1103	支持

【注意】

- PIPE 必须已创建。
- VI Channel必须已创建。
- 该接口需动态调用。

5.59 RK_MPI_VI_SetChnMirrorFlip

【描述】

设置当前 VI 通道镜像、翻转的信息。

Mirror / Flip 接口功能是对Sensor原生不支持这类操作，不能通过 rkAiq 接口配置时，通过 ISP 来实现这类功能的补充。

【语法】

```
RK_S32 RK_MPI_VI_SetChnMirrorFlip(VI\_PIPEViPipe, VI\_CHNViChn, VI\_ISP\_MIRROR\_FLIP\_STRUCT*stMirrFlip);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
stMirrFlip	镜像旋转结构体。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【芯片差异】

芯片	是否支持该功能
RV1126/RV1109	不支持
RK356x	不支持
RK3588	不支持
RV1106/RV1103	支持

【注意】

- PIPE 必须已创建。
- VI Channel 必须已创建。
- 若通道0(mainpath通道)开启了卷绕模式，则不支持 Mirror / Flip 的设置。
- Mirror 镜像功能是全局的属性配置，即某个通道设置后，其他通道均会有效。
- Flip 翻转功能是独立的属性配置，即只对某个通道设置后有效。其他通道无效。
- 该接口需动态调用。

5.60 RK_MPI_VI_GetEptz

【描述】

获取当前 VI 通道裁剪与缩放的信息。

【语法】

```
RK_S32 RK_MPI_VI_GetEptz(VI_PIPE ViPipe, VI_CHN ViChn, VI_CROP_INFO_S *stCropInfo);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围：[0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围：[0, VI_MAX_CHN_NUM)。	输入
*stCropInfo	裁剪缩放信息结构体。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【芯片差异】

芯片	是否支持该功能
RV1126/RV1109	不支持
RK356x	不支持
RK3588	不支持
RV1106/RV1103	支持

【注意】

- PIPE 必须已创建。
- VI Channel必须创建。
- 该接口需动态调用，不支持静态调用。
- 不支持直通模式情况下的调用，支持双摄/多摄模式下的回读模式。

5.61 RK_MPI_VI_SetEptz

【描述】

设置当前 VI 通道裁剪与缩放的信息。

【语法】

```
RK_S32 RK_MPI_VI_SetEptz(VI_PIPE ViPipe, VI_CHN ViChn, VI_CROP_INFO_S stCropInfo);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
stCropInfo	裁剪缩放信息结构体。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【芯片差异】

芯片	是否支持该功能
RV1126/RV1109	不支持
RK356x	不支持
RK3588	不支持
RV1106/RV1103	支持

【注意】

- PIPE 必须已创建。
- VI Channel必须创建。
- 该接口需动态调用，不支持静态调用。
- 不支持直通模式情况下的调用，支持双摄/多摄模式下的回读模式。

5.62 RK_MPI_VI_DevEnableLight

【描述】

设置当前设备是否打开补光灯。

【语法】

```
RK_S32 RK_MPI_VI_DevEnableLight(VI\_PIPEViPipe, VI\_CHNViChn, VI\_LIGHT\_CTL\_PARAM\_S *lightCtlParam);
```

【参数】

参数名	描述	输入/输出
ViPipe	VI PIPE号。 取值范围: [0, VI_MAX_PIPE_NUM)。	输入
ViChn	VI通道号。 取值范围: [0, VI_MAX_CHN_NUM)。	输入
lightCtlParam	补光灯参数信息结构体。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 VI错误码 。

【芯片差异】

芯片	是否支持该功能
RV1126/RV1109	不支持
RK356x	不支持
RK3588	不支持
RV1106/RV1103	支持

【注意】

- PIPE 必须已创建。
- VI Channel必须创建。
- 该接口需动态调用, 不支持静态调用。

5.63 RK_MPI_VI_DevEnableSingleFrame

【描述】

使能当前 VI 设备获取单帧。在AOV模式下, 使能应用获取一帧数据。

【语法】

```
RK_S32 RK_MPI_VI_DevEnableSingleFrame(VI\_DEVVidev, int frmNum);
```

【参数】

参数名	描述	输入/输出
Videv	VI DEV号。 取值范围: [0, VI_MAX_DEV_NUM)。	输入
frmNum	裁剪缩放信息结构体。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 VI错误码 。

【芯片差异】

芯片	是否支持该功能
RV1126/RV1109	不支持
RK356x	不支持
RK3588	不支持
RV1106/RV1103	支持

【注意】

- DEV必须已创建。
- 该接口需动态调用，不支持静态调用。

6. 数据类型

VI 模块相关数据类型定义如下：

- [DIS_MODE_E](#)：定义 DIS 防抖算法模式。
- [DIS_MOTION_LEVEL_E](#)：定义镜头的运动级别。
- [DIS_PDT_TYPE_E](#)：定义 DIS 防抖支持的产品形态。
- [DIS_CONFIG_S](#)：定义 DIS 的配置信息。
- [DIS_ATTR_S](#)：定义 DIS 的属性参数。
- [CAC_EFFECT_ATTR_S](#)：定义 CAC 的效果参数。
- [SWCAC_CONFIG_S](#)：定义 CAC 的配置参数。

6.1 VI_DEV

【说明】

定义 VI 设备句柄。

【定义】

```
typedef RK_S32 VI_DEV;
```

6.2 VI_PIPE

【说明】

定义 VI PIPE。

【定义】

```
typedef RK_S32 VI_PIPE;
```

6.3 VI_CHN

【说明】

定义 VI 通道。

【定义】

```
typedef RK_S32 VI_CHN;
```

6.4 VI_MAX_DEV_NUM

【说明】

定义设备最大个数。

【定义】

```
#define VI_MAX_DEV_NUM 3
```

6.5 VI_MAX_PIPE_NUM

【说明】

定义PIPE最大个数。

【定义】

```
#define VI_MAX_PIPE_NUM VI_MAX_DEV_NUM
```

6.6 VI_MAX_CHN_NUM

【说明】

定义 VI 物理通道和扩展通道的总个数。

【定义】

```
#define VI_MAX_CHN_NUM (VI_MAX_PHY_CHN_NUM + VI_MAX_EXT_CHN_NUM)
```

6.7 DIS_MAX_IMAGE_WIDTH

【说明】

定义 DIS 的最大图像宽度。

【定义】

```
#define DIS_MAX_IMAGE_WIDTH (4096)
```

【相关数据类型及接口】

无。

6.8 DIS_MAX_IMAGE_HEIGHT

【说明】

定义 DIS 的最大图像高度。

【定义】

```
#define DIS_MAX_IMAGE_HEIGHT (4096)
```

【相关数据类型及接口】

无。

6.9 DIS_MIN_IMAGE_WIDTH

【说明】

定义 DIS 的最小图像宽度。

【定义】

```
#define DIS_MIN_IMAGE_WIDTH (1080)
```

【相关数据类型及接口】

无。

6.10 DIS_MIN_IMAGE_HEIGHT

【说明】

定义 DIS 的最小图像高度。

【定义】

```
#define DIS_MIN_IMAGE_HEIGHT (1080)
```

【相关数据类型及接口】

无。

6.11 DIS_MODE_E

【说明】

定义 DIS 防抖算法模式。

【定义】

```
typedef enum rkDIS_MODE_E {
    DIS_MODE_4_DOF_GME = 0, /* Only use with GME in 4 dof */
    DIS_MODE_6_DOF_GME,   /* Only use with GME in 6 dof */
    DIS_MODE_GYRO,        /* Only use with gyro in 6 dof */
    DIS_MODE_DOF_BUTT
} DIS_MODE_E;
```

【成员】

成员名称	描述
DIS_MODE_4_DOF_GME	四自由度 GME 算法，不使用陀螺仪。
DIS_MODE_6_DOF_GME	六自由度 GME 算法，不使用陀螺仪。
DIS_MODE_GYRO	陀螺仪算法，不使用 GME。

【相关数据类型及接口】

- [RK_MPI_VI_SetChnDISConfig](#)
- [RK_MPI_VI_GetChnDISConfig](#)

6.12 DIS_MOTION_LEVEL_E

【说明】

定义镜头的运动级别。

【定义】

```
typedef enum rkDIS_MOTION_LEVEL_E {
    DIS_MOTION_LEVEL_LOW = 0,
    DIS_MOTION_LEVEL_NORMAL,
    DIS_MOTION_LEVEL_HIGH,
    DIS_MOTION_LEVEL_BUTT
} DIS_MOTION_LEVEL_E;
```

【成员】

成员名称	描述
DIS_MOTION_LEVEL_LOW	低级别运动，镜头小幅度运动。
DIS_MOTION_LEVEL_NORMAL	正常级别运动，镜头正常幅度运动。
DIS_MOTION_LEVEL_HIGH	高级别运动，镜头大幅度运动。

【相关数据类型及接口】

- [RK_MPI_VI_SetChnDISConfig](#)
- [RK_MPI_VI_GetChnDISConfig](#)

6.13 DIS_PDT_TYPE_E

【说明】

定义 DIS 防抖支持的产品形态。

【定义】

```
typedef enum rkDIS_PDT_TYPE_E {
    DIS_PDT_TYPE_IPC = 0,
    DIS_PDT_TYPE_DV,
    DIS_PDT_TYPE_DRONE,
    DIS_PDT_TYPE_BUTT
} DIS_PDT_TYPE_E;
```

【成员】

成员名称	描述
DIS_PDT_TYPE_IPC	IPC。
DIS_PDT_TYPE_DV	DV。
DIS_PDT_TYPE_DRONE	无人机。

【相关数据类型及接口】

- [RK_MPI_VI_SetChnDISConfig](#)
- [RK_MPI_VI_GetChnDISConfig](#)

6.14 VI_RAW_MEMORY_TYPE_E

```
typedef enum {
    VI_RAW_MEM_COMPACT = 0,
    VI_RAW_MEM_WORD_LOW_ALIGN = 1,
    VI_RAW_MEM_WORD_HIGH_ALIGN = 2,
} VI_RAW_MEMORY_TYPE_E;
```

【说明】

定义raw数据内存排列格式。

【成员】

成员名称	描述
VI_RAW_MEM_COMPACT	紧凑型
VI_RAW_MEM_WORD_LOW_ALIGN	低字节对齐
VI_RAW_MEM_WORD_HIGH_ALIGN	高字节对齐

6.15 VI_HDR_MODE_E

```
typedef enum {
    VI_MODE_NORMAL    = 0,
    VI_MODE_ISP_HDR2  = 0x10,
    VI_MODE_ISP_HDR3  = 0x20,
} VI_HDR_MODE_E;
```

【说明】

定义dev端pipeline模式。

成员名称	描述
VI_MODE_NORMAL	普通模式
VI_MODE_ISP_HDR2	两帧模式
VI_MODE_ISP_HDR3	三帧模式

6.16 DATA_BITWIDTH_E

【说明】

数据bit数

【定义】

```

typedef enum rkDATA_BITWIDTH_E {
    DATA_BITWIDTH_8 = 0,
    DATA_BITWIDTH_10,
    DATA_BITWIDTH_12,
    DATA_BITWIDTH_14,
    DATA_BITWIDTH_16,
    DATA_BITWIDTH_BUTT
} DATA_BITWIDTH_E;

```

【成员】

成员名称	描述
DATA_BITWIDTH_8	8 bit
DATA_BITWIDTH_10	10 bit
DATA_BITWIDTH_12	12 bit
DATA_BITWIDTH_14	14 bit
DATA_BITWIDTH_16	16 bit
DATA_BITWIDTH_BUTT	保留

6.17 VI_DEV_ATTR_S

【说明】

定义VI设备属性结构体。

【定义】

```

/* The attributes of a VI device */
typedef struct rkVI_DEV_ATTR_S {
    /* RW;Interface mode */
    VI_INTF_MODE_E    enIntfMode;
    /* RW;Work mode */
    VI_WORK_MODE_E    enWorkMode;
    /* The below members must be configured in BT.601 mode or DC mode and are invalid in other modes */
    /* RW;Input data sequence (only the YUV format is supported) */
    VI_YUV_DATA_SEQ_E enDataSeq;
    /* RW;RGB: CSC-709 or CSC-601, PT YUV444 disable; YUV: default yuv CSC coef PT YUV444 enable. */
    VI_DATA_TYPE_E    enInputDataType;
    /* RW;Input max size */
    SIZE_S             stMaxSize;
    /* RW;Data rate of Device */
    DATA_RATE_E        enDataRate;
    /* RW;Pixel format */
    PIXEL_FORMAT_E     enPixFmt;
    /* RW;Dev mem type */
    VI_RAW_MEMORY_TYPE_E enMemMode;
    /* RW;Dev buff type */
    VI_V4L2_MEMORY_TYPE enBufType;
    /* RW;Dev buff cnt */
    RK_U32             u32BufCount;
    /* RW;Dev Hdr Mode */
    VI_HDR_MODE_E      enHdrMode;
    /* RW;Dev Set Crop size*/
    RECT_S              stCropRect;
    /* RW; Dev Long Frame raw buffer wrap line */
    RK_U32             u32LongFrameWrapLine;
} VI_DEV_ATTR_S;

```

【成员】

成员名称	描述
enIntfMode	接口模式。 (暂未使用, 可不设置)
enWorkMode	1、2、4 路复合工作模式。 (暂未使用, 可不设置)
enDataSeq	输入数据顺序 (仅支持 yuv 格式), 当 enIntfMode 为 VI_MODE_BT656 或者 VI_MODE_BT601 时取值范围为 [VI_DATA_SEQ_UYVY, VI_DATA_SEQ_YVYU], 当 enIntfMod 为 VI_MODE_BT1120_STANDARD, VI_MODE_MIPI_YUV420_NORMAL, VI_MODE_MIPI_YUV420_LEGACY, VI_MODE_MIPI_YUV422 时取值必须为 VI_DATA_SEQ_VUVU 或者 VI_DATA_SEQ_UVUV。 (暂未使用, 可不设置)
enInputDataType	输入数据类型, Sensor 输入一般为 RGB, AD 输入一般为 YUV。 (暂未使用, 可不设置)
stMaxSize	捕获图像的最大宽高。 (暂未使用, 可不设置)
enDataRate	设备的速率。 (暂未使用, 可不设置)
enPixFmt	数据像素格式, 默认RK_FMT_RGB_BAYER_SBGR_10BPP, dev离线模式下生效。
enMemMode	数据内存排布格式, 默认VI_RAW_MEM_COMPACT, dev离线模式下生效。
enBufType	buff类型, 默认VI_V4L2_MEMORY_TYPE_MMAP, dev离线模式下生效。
u32BufCount	buff个数, 默认3, dev离线模式下生效
enHdrMode	hdr模式, 默认VI_MODE_NORMAL, dev离线时生效
stCropRect	设置dev端裁剪。 (暂未使用, 可不设置)
u32LongFrameWrapLine	直通模式hdr帧卷绕行配置

【注意事项】

- u32LongFrameWrapLine 配置仅支持RV1103B平台且isp工作在直通模式下, 不支持4k unite和离线模式, 建议配置为 height/4, 最小需大于sensor长短帧间隔行, 为0时功能关闭; 可通过echo或者cat /sys/module/video_rkisp/parameters/hdr_wrap_line节点配置或查看卷绕行。

6.18 VI_DEV_STATUS_S

【说明】

定义VI Sensor的探测状态。用来检测当前sensor是否在线。

【定义】

```
typedef struct rkVI_DEV_STATUS_S {
    SIZE_S stSize;          /* RO;chn output size */
    RK_BOOL bProbeOk;       /* RO;whether sensor is probed success */
} VI_DEV_STATUS_S;
```

【成员】

成员名称	描述
stSize	获取sensor原始图像输出宽高大小。
bProbeOk	sensor是否被成功探测到。

6.19 VI_DEV_BIND_PIPE_S

【说明】

定义 VI DEV 与 PIPE 的绑定关系。

【定义】

```
/* Information of pipe binded to device */
typedef struct rkVI_DEV_BIND_PIPE_S {
    RK_U32 u32Num;           /* RW;Range [1,VI_MAX_PHY_PIPE_NUM] */
    VI_PIPE PipeId[MAX_VI_BIND_PIPE_NUM]; /* RW;Array of pipe ID */
    RK_BOOL bDataOffline;    /* RW;RK_FALSE: dev work online, RK_TRUE: dev work offline */
    RK_BOOL bUserStartPipe[MAX_VI_BIND_PIPE_NUM]; /* RW;RK_FALSE: vi start pipe, RK_TRUE: user start pipe */
} VI_DEV_BIND_PIPE_S;
```

【成员】

成员名称	描述
u32Num	该 VI Dev 所绑定的 PIPE 数目，取值范围[1, VI_MAX_PIPE_NUM]。
PipeId	该 VI Dev 绑定的 PIPE 号。
bDataOffline	定义dev,pipe工作为在离线或在线，RK_FALSE：在线，RK_TRUE：离线，默认RK_FALSE.
bUserStartPipe	定义在绑定时是否启动pipe，RK_FALSE: vi启动pipe，RK_TRUE: 用户启动pipe， 默认RK_FALSE

【注意事项】

6.20 VI_AIISP_EXP_PARAM_S

【说明】

Aiisp所需曝光参数，用于aiisp回调函数获取曝光参数

【定义】

```
typedef struct rkVIAIISP_EXP_PARAM {
    RK_FLOAT fAnalogGain; /* sensor analog gain */
    RK_FLOAT fDigitalGain; /* sensor digital gain */
    RK_FLOAT flspDgain; /* isp digital gain */
} VI_AIISP_EXP_PARAM_S;
```

【成员】

成员名称	说明
fAnalogGain	sensor 模拟增益
fDigitalGain	sensor 数字增益
flspDgain	isp 数字增益

6.21 VIAIISP_CALLBACK_FUNC_S

【说明】

定义Aiisp回调函数注册数据结构。

【定义】

```

typedef struct rkVIAIISP_CALLBACK_S {
    PREAIISP_CALLBACK    pfUpdateCallback;
    RK_VOID             *pPrivateData; /* Not yet used */
} VIAIISP_CALLBACK_S;

```

【成员】

成员名称	说明
pfUpdateCallback	用户注册用户函数，用于aiisp获取iso
pPrivateData	保留参数

6.22 VI_AIISP_INFO_S

【说明】

Aiisp参数定义

【定义】

```

typedef struct rkVI_AIISP_INFO_S {
    RK_BOOL      bEnable;      /* Whether AIISP is enable */
    VIAIISP_CALLBACK_S  callBack;   /* Aiisp get sensor iso and private by userCallBack */
} VI_AIISP_INFO_S;

```

【成员】

成员名称	描述
bEnable	是否是能aiisp
callBack	注册回调

6.23 VI_PIPE_ATTR_S

【说明】

定义pipe属性

```

typedef struct rkVI_PIPE_ATTR_S {
    /* RW;Range:[0,1];ISP bypass enable */
    RK_BOOL      blspBypass;
    /* RW;Range:[0,1];Range[VI_PIPE_MIN_WIDTH,VI_PIPE_MAX_WIDTH];Maximum width */
    RK_U32       u32MaxW;
    /* RW;Range[VI_PIPE_MIN_HEIGHT,VI_PIPE_MAX_HEIGHT];Maximum height */
    RK_U32       u32MaxH;
    /* RW;Pixel format */
    PIXEL_FORMAT_E  enPixFmt;
    /* RW;Range:[0,4];Compress mode.*/
    COMPRESS_MODE_E  enCompressMode;
    /* RW;Range:[0,4];Bit width*/
    DATA_BITWIDTH_E  enBitWidth;
    /* RW;Frame rate */
    FRAME_RATE_CTRL_S  stFrameRate;
    /* RW;Mem mode, used when dev works on offline mode */
    VI_RAW_MEMORY_TYPE_E  enMemMode;
    /* RW;Pipe Hdr Mode */
    VI_HDR_MODE_E    enHdrMode;
} VI_PIPE_ATTR_S;

```

【成员】

成员名称	描述
bIspBypass	配置pipe是否为bypass, TRUE:enable, FALSE:disable
u32MaxW	pipe 宽大小
u32MaxH	pipe 高大小
enPixFmt	pipe处理帧像素格式
enCompressMode	pipe处理帧压缩格式
enBitWidth	pipe处理帧bit数
stFrameRate	pipe处理帧率, 暂未使用
enMemMode	pipe处理帧内存排列格式, 默认VI_RAW_MEM_COMPACT, Pipe离线下生效
enHdrMode	pipe 处理多帧模式, 默认VI_MODE_NORMAL, Pipe离线下生效

6.24 VI_CHN_ATTR_S

【说明】

定义 VI 通道属性。

【定义】

```
/* The attributes of channel */
typedef struct rkVI_CHN_ATTR_S {
    SIZE_S      stSize;        /* RW;Channel out put size */
    PIXEL_FORMAT_E  enPixelFormat;  /* RW;Pixel format */
    DYNAMIC_RANGE_E  enDynamicRange; /* RW;Dynamic Range */
    VIDEO_FORMAT_E  enVideoFormat; /* RW;Video format */
    COMPRESS_MODE_E  enCompressMode; /* RW;256B Segment compress or no compress.*/
    RK_BOOL      bMirror;       /* RW;Mirror enable */
    RK_BOOL      bFlip;         /* RW;Flip enable */
    RK_U32       u32Depth;      /* RW;Range [0,8];Depth */
    FRAME_RATE_CTRL_S stFrameRate; /* RW;Frame rate */
    VI_BUF_TYPE_E  enBufType;    /* RW;channel buf opt */
    VI_ISP_OPT_S   stIspOpt;     /* RW;isp opt */
    MPP_CHN_S      stShareBufChn; /* RW; enAllocBufType=VI_ALLOC_BUF_TYPE_CHN_SHARE, share buffer*/
} VI_CHN_ATTR_S;
```

【成员】

成员名称	描述
stSize	获取通道图像宽高大小。
enPixelFormat	目标图像像素格式。详情参看rk_comm_video.h中PIXEL_FORMAT_E定义。
enDynamicRange	目标图像动态范围。（暂未使用，可不设置）
enVideoFormat	目标图像视频数据格式。（暂未使用，可不设置）
enCompressMode	目标图像压缩格式。
bMirror	Mirror 使能开关。 RK_FALSE: 不使能； RK_TRUE: 使能。（暂未使用，可不设置）
bFlip	Flip 使能开关。 RK_FALSE: 不使能； RK_TRUE: 使能。（暂未使用，可不设置）
u32Depth	用户获取图像的队列深度。取值范围：[0,8]；
stFrameRate	帧率控制。 源帧率取值范围：(0, 输入最大帧率]，以及-1。目标帧率取值范围：[-1, 源帧率]。当源帧率为-1时，目标帧率必须为-1(不进行帧率控制)，其他情况下，目标帧率不能大于源帧率。
enAllocBufType	图像内存申请类型。 VI_ALLOC_BUF_TYPE_INTERNAL: 内部申请。 VI_ALLOC_BUF_TYPE_EXTERNAL: 外部申请。 VI_ALLOC_BUF_TYPE_CHN_SHARE: VI内部共享BUF。 当图像类型为mmap方式获取时需要设置为外部申请。
stIspOpt	获取图像为isp处理/直通数据的参数设置。
stShareBufChn	enAllocBufType设置为VI_ALLOC_BUF_TYPE_CHN_SHARE时指定的共享内存通道，其中 stShareBufChn.enModId 必须设置RK_ID_VI, (stShareBufChn.s32DevId, stShareBufChn.s32ChnId) 代表具体的通道，其中stShareBufChn.s32DevId为PIPE ID, stShareBufChn.s32ChnId 为CHN ID, 如果 stShareBufChn.s32ChnId 为 -1, stShareBufChn.s32DevId 代表设备ID

【注意事项】

- u32Depth的设置参考如下：
 1. 当vi后级没有绑定其他模块时建议u32Depth设置跟VI_ISP_OPT_S的u32BufCount个数一致，如果此时设置0则获取不到输出buf。
 2. 当vi后级绑定有其他模块时：
通过RK_MPI_VI_GetChnFrame获取图像，建议u32Depth设置比VI_ISP_OPT_S的u32BufCount至少要小2，如u32BufCount设置为3，u32Depth设置为1。此时u32Depth设置过高会影响正常输出数据或者导致后级帧率不足。
不通过RK_MPI_VI_GetChnFrame获取图像，u32Depth设置为0。
- stShareBufChn的设置参考如下：
 1. enAllocBufType 必须设置VI_ALLOC_BUF_TYPE_CHN_SHARE，否则无效。
 2. 成员enModId必须为 RK_ID_VI。
 3. 成员 s32DevId, s32ChnId 指定的通道必须已经创建，并且创建时enAllocBufType 必须设置 VI_ALLOC_BUF_TYPE_INTERNAL。
 4. 如果s32ChnId 设置 -1， s32DevId所对应的设备必须已经创建，并且创建时enBufType必须设置为 VI_V4L2_MEMORY_TYPE_DMABUF。

6.25 SIZE_S

【说明】

定义图像大小信息结构体。

【定义】

```
typedef struct rkSIZE_S {  
    RK_U32 u32Width;  
    RK_U32 u32Height;  
} SIZE_S;
```

【成员】

成员名称	描述
u32Width	宽度。
u32Height	高度。

6.26 COMPRESS_MODE_E

【说明】

定义数据压缩格式结构体。

【定义】

```
typedef enum rkCOMPRESS_MODE_E {  
    COMPRESS_MODE_NONE = 0, /* No compress */  
    COMPRESS_AFBC_16x16,  
    COMPRESS_MODE_BUTT  
} COMPRESS_MODE_E;
```

【成员】

成员名称	描述
COMPRESS_MODE_NONE	无压缩
COMPRESS_AFBC_16x16	AFBC压缩
COMPRESS_MODE_BUTT	无

6.27 VI_ALLOC_BUF_TYPE_E

【说明】

定义VI图像内存分配类型。

【定义】

```
typedef enum rkVI_ALLOC_BUF_TYPE_E {  
    VI_ALLOC_BUF_TYPE_INTERNAL,  
    VI_ALLOC_BUF_TYPE_EXTERNAL,  
    VI_ALLOC_BUF_TYPE_CHN_SHARE  
} VI_ALLOC_BUF_TYPE_E;
```

【成员】

成员名称	描述
VI_ALLOC_BUF_TYPE_INTERNAL	VI内部分配
VI_ALLOC_BUF_TYPE_EXTERNAL	VI外部分配
VI_ALLOC_BUF_TYPE_CHN_SHARE	VI内部共享BUF

6.28 VI_ISP_OPT_S

【说明】

获取图像为isp处理/直通数据的参数设置。

【定义】

```
typedef struct rkVI_ISP_OPT_S {
    RK_U32      u32BufCount;    /* RW;isp buf count */
    RK_U32      u32BufSize;     /* R;isp buf size */
    VI_V4L2_CAPTURE_TYPE enCaptureType; /* RW;isp capture type */
    VI_V4L2_MEMORY_TYPE enMemoryType; /* RW;isp buf memory type */
    RK_CHAR     aEntityName[MAX_VI_ENTITY_NAME_LEN]; /* RW;isp capture entity name*/
    RK_BOOL     bNoUseLibV4L2;   /* RW;is use libv4l2 */
    SIZE_S      stMaxSize;     /* RW;isp bypass resolution */
    RECT_S      stWindow;      /* RW;isp output window*/
} VI_ISP_OPT_S;
```

【成员】

成员名称	描述
u32BufCount	输出通道总的缓存块数。
u32BufSize	每个缓存块申请的缓存大小。
enCaptureType	获取图像的V4L2录制类型。
enMemoryType	获取图像的V4L2内存类型。
aEntityName	通道设备名字。
bNoUseLibV4L2	是否使用libV4L2（暂时只支持开启）。
stMaxSize	通道获取图像的最大分辨率。
stWindow	通道在显存中的显示窗口

【注意事项】

- u32BufSize的大小必须能够存下stMaxSize所指定图像数据。
- stWindow不允许越界，即 stWindow.s32X + stWindow.u32Width <= stMaxSize.u32Width; stWindow.s32Y + stWindow.u32Height <= stMaxSize.u32Height;
- RK3588平台，因底层硬件需要做缓存处理，u32BufCount 至少需要3个或以上buffer才能获取到满足帧率需求的帧数。
- RV1126平台，因底层驱动已有做缓存处理，u32BufCount 可以配置更少的个数即可满足帧率需求，建议最少配置2个。

6.29 VI_V4L2_CAPTURE_TYPE

【说明】

获取图像的V4L2录制类型。

【定义】

```
typedef enum rkVI_V4L2_CAPTURE_TYPE {
    VI_V4L2_CAPTURE_TYPE_VIDEO_CAPTURE = 1,
    VI_V4L2_CAPTURE_TYPE_VBI_CAPTURE = 4,
    VI_V4L2_CAPTURE_TYPE_SLICED_VBI_CAPTURE = 6,
    VI_V4L2_CAPTURE_TYPE_VIDEO_CAPTURE_MPLANE = 9,
    VI_V4L2_CAPTURE_TYPE_SDR_CAPTURE = 11,
    VI_V4L2_CAPTURE_TYPE_META_CAPTURE = 13,
    /* Deprecated, do not use */
    VI_V4L2_CAPTURE_TYPE_PRIVATE = 0x80,
} VI_V4L2_CAPTURE_TYPE;
```

【注意事项】

- 同V4L2_CAPTURE_TYPE_VIDEO_CAPTURE设置。

6.30 VI_V4L2_MEMORY_TYPE

【说明】

获取图像的V4L2内存类型。

【定义】

```
typedef enum rkVI_V4L2_MEMORY_TYPE {
    VI_V4L2_MEMORY_TYPE_MMAP      = 1,
    VI_V4L2_MEMORY_TYPE_USERPTR   = 2,
    VI_V4L2_MEMORY_TYPE_OVERLAY   = 3,
    VI_V4L2_MEMORY_TYPE_DMABUF    = 4,
} VI_V4L2_MEMORY_TYPE;
```

【成员】

成员名称	描述
VI_V4L2_MEMORY_TYPE_MMAP	MMAP内存类型。
VI_V4L2_MEMORY_TYPE_USERPTR	USERPTR内存类型。
VI_V4L2_MEMORY_TYPE_OVERLAY	OVERLAY内存类型。
VI_V4L2_MEMORY_TYPE_DMABUF	DMA分配内存类型。

6.31 VI_MODTYPE_E

【说明】

vi工作模式

【定义】

```
typedef enum rkVI_MODTYPE_E {
    VI_DEV_PIPE_MODE = 1, /* DEV_PIPE_MODE */
    VI_CHN_WRAP_MODE, /* CHN_WRAP_MODE */
    VI_BUTT
} VI_MODTYPE_E;
```

【成员】

成员名称	描述
VI_DEV_PIPE_MODE	dev, pipe配置工作模式
VI_CHN_WRAP_MODE	卷绕配置模式

6.32 rkVI_DEV_PIPE_MODTYPE_E

【说明】

VI_DEV_PIPE_MODE模式下dev和pipe之间在驱动层数据流通模式

【定义】

```

typedef enum rkVI_DEV_PIPE_MODTYPE_E {
    VI_DEV_PIPE_OFFLINE = 1, /* devx->ddr->pipex*/
    VI_DEV_PIPE_ONLINE = 2, /* devx->pipex*/
    VI_DEV_PIPE_LEFT_HALF_ONLINE = 3, /* dev0->pipe0, dev1->ddr->pipe1 */
    VI_DEV_PIPE_RIGHT_HALF_ONLINE = 4, /* dev1->pipe1, dev0->ddr->pipe0 */
    VI_DEV_PIPE_UNITE_HALF_ONLINE = 5, /* unite mode */
    VI_DEV_PIPE_BUTT
} VI_DEV_PIPE_MODTYPE_E;

```

【成员】

成员名称	描述
VI_DEV_PIPE_OFFLINE	dev, pipe均为离线状态，支持单双摄配置
VI_DEV_PIPE_ONLINE	dev, pipe均为在线状态，支持单双摄配置
VI_DEV_PIPE_LEFT_HALF_ONLINE	dev0-pipe0为在线状态, dev1-pipe1为离线状态
VI_DEV_PIPE_RIGHT_HALF_ONLINE	dev0-pipe0为离线状态, dev1-pipe1为在线状态
VI_DEV_PIPE_UNITE_HALF_ONLINE	4k分辨率输入配置半直通

【特别说明】

- VI_DEV_PIPE_MODTYPE_E 配置仅在RV1103B上可生效。
- 使用 VI_DEV_PIPE_MODTYPE_E 配置离线模式时，可通过 VI_DEV_ATTR_S 中的 u32BufCount 指定对应的离线buff个数。
- 上述配置可使用以下命令查看

1. 单摄查看模式

`cat /sys/module/video_rkisp/parameters/rdbk_auto`

Y 表示离线, N 表示在线

2. 单摄查看离线buff个数

`cat /sys/module/video_rkisp/parameters/vicap_raw_buf`

3. 双摄查看模式

`cat /sys/module/video_rkisp/parameters/m_online`

Y,N Y 表示在线, N 表示离线, 这里Y,N 表示isp0为在线, isp1为离线

4. 双摄查看离线buff个数

`cat /sys/module/video_rkisp/parameters/vicap_raw_buf`

3,0,0,0 表示isp0离线buff为3个

6.33 VI_MODTYPE_E

【说明】

卷绕类型配置

【定义】

```

/* vi chn wrap mode type */
typedef enum rkVI_CHN_WRAP_MODTYPE_E {
    VI_CHN_WRAP_HARD = 1, /* vi venc use hardware wrap */
    VI_CHN_WRAP_SOFT = 2, /* vi venc use software wrap */
    VI_CHN_WRAP_BUTT
} VI_CHN_WRAP_MODTYPE_E;

```

【成员】

成员名称	描述
VI_CHN_WRAP_HARD	硬件卷绕
VI_CHN_WRAP_SOFT	软件卷绕

【特别说明】

- 仅RV1103B平台支持

6.34 VIDEO_FRAME_S

【说明】

视频帧信息结构体。

【定义】

```
typedef struct rkVIDEO_FRAME_S {
    MB_BLK      pMbBlk;
    RK_U32      u32Width;
    RK_U32      u32Height;
    RK_U32      u32VirWidth;
    RK_U32      u32VirHeight;
    VIDEO_FIELD_E   enField;
    PIXEL_FORMAT_E  enPixelFormat;
    VIDEO_FORMAT_E  enVideoFormat;
    COMPRESS_MODE_E enCompressMode;
    DYNAMIC_RANGE_E enDynamicRange;
    COLOR_GAMUT_E   enColorGamut;

    RK_VOID      *pVirAddr[RK_MAX_COLOR_COMPONENT];

    RK_U32      u32TimeRef;
    RK_U64      u64PTS;

    RK_U64      u64PrivateData;
    RK_U32      u32FrameFlag; /* FRAME_FLAG_E, can be OR operation. */
} VIDEO_FRAME_S;

typedef struct rkVIDEO_FRAME_INFO_S {
    VIDEO_FRAME_S stVFrame;
} VIDEO_FRAME_INFO_S;
```

【成员】

成员名称	描述
pMbBlk	输出图像buf的mediabuff结构体。
u32Width	图像宽度
u32Height	图像高度。
u32VirWidth	图像虚宽。
u32VirHeight	图像虚高。
enField	帧场模式。
enPixelFormat	视频图像像素格式。
enVideoFormat	视频图像格式。
enCompressMode	视频压缩格式。
enDynamicRange	动态范围。
enColorGamut	色域范围。
pVirAddr	图像数据虚拟地址。
u32TimeRef	图像帧序列号。
u64PTS	输出图像的时戳。
u64PrivateData	私有数据。
u32FrameFlag	当前帧的标记，使用 FRAME_FLAG_E 里面的值标记，可以按位或操作。

6.35 VI_CHN_STATUS_S

【说明】

VI通道状态结构体。

【定义】

```
typedef struct rkVI_CHN_STATUS_S {
    RK_BOOL bEnable;           /* RO;Whether this channel is enabled */
    RK_U32 u32FrameRate;      /* RO;Current frame rate */
    RK_U32 u32CurFrameID;     /* RO;Current frame id */
    RK_U32 u32InputLostFrame; /* RO;Input lost frame count */
    RK_U32 u32OutputLostFrame; /* RO;Output lost frame count */
    RK_U32 u32VbFail;         /* RO;Video buffer malloc failure */
    SIZE_S stSize;             /* RO;Channel output size */
} VI_CHN_STATUS_S;
```

【成员】

成员名称	描述
bEnable	当前通道是否使能。0：不使能；1：使能。
u32FrameRate	当前通道输出帧率。
u32CurFrameID	当前通道图像的帧号。
u32InputLostFrame	当前通道输入图像的丢失帧数。
u32OutputLostFrame	当前通道输出图像的丢失帧数。
u32VbFail	当前通道获取图像失败次数。
stSize	当前通道图像大小。

6.36 VI_SAVE_FILE_INFO_S

【说明】

VI通道数据保存信息结构体。

【定义】

```
typedef struct rkVI_SAVE_FILE_INFO_S {
    RK_BOOL    bCfg;
    RK_CHAR    aFilePath[MAX_VI_FILE_PATH_LEN];
    RK_CHAR    aFileName[MAX_VI_FILE_NAME_LEN];
    RK_U32    u32FileSize; /*in KB*/
} VI_SAVE_FILE_INFO_S;
```

【成员】

成员名称	描述
bCfg	是否保存通道输出数据。0：不保存；1：保存。
aFilePath	保存通道输出数据文件路径。
aFileName	保存通道输出数据文件名。
u32FileSize	保存通道输出数据文件大小，单位kB。

6.37 VI_USERPIC_BGC_S

【说明】

纯色背景模式下的用户图片相关信息。

【定义】

```
typedef struct rkVI_USERPIC_BGC_S {
    RK_U32 u32BgColor;
} VI_USERPIC_BGC_S;
```

【成员】

成员名称	描述
u32BgColor	填充数据，与颜色的RGB值对应。取值范围：[0, 0xFFFFFFFF]。 其中R分量16~23bit，G分量8~15bit，B分量0~7bit。

6.38 VI_USERPIC_MODE_E

【说明】

用户图片类型。

【定义】

```
typedef enum rk_VI_USERPIC_MODE_E {
    VI_USERPIC_MODE_PIC = 0,
    VI_USERPIC_MODE_BGC,
    VI_USERPIC_MODE_BUTT,
} VI_USERPIC_MODE_E;
```

【成员】

成员名称	描述
VI_USERPIC_MODE_PIC	YUV 图像模式
VI_USERPIC_MODE_BGC	纯色背景图像模式。

【注意事项】

无

6.39 VI_USERPIC_ATTR_S

【说明】

用户图片信息。

【定义】

```
typedef struct rkVI_USERPIC_ATTR_S {
    VI_USERPIC_MODE_E enUsrPicMode;
    union {
        VIDEO_FRAME_INFO_S stUsrPicFrm;
        VI_USERPIC_BGC_S stUsrPicBg;
    } unUsrPic;
} VI_USERPIC_ATTR_S;
```

【成员】

成员名称	描述
enUsrPicMode	用户图片模式
stUsrPicFrm	YUV 图像模式下的用户图片信息
stUsrPicBg	纯色背景模式下的用户图片信息。 取值范围 [0x0, 0xFFFFFFF]。 其中R分量0~7bit, G分量 8~15bit, B分量16~23bit。

【注意事项】

无

6.40 VI_CONNECT_INFO_S

【说明】

通道连接信息。

【定义】

```
typedef struct rkVI_CONNECT_INFO_S {  
    RK_U32 u32Width;  
    RK_U32 u32Height;  
    RK_FLOAT f32FrameRate;  
    PIXEL_FORMAT_E enPixFmt;  
    VI_CONNECT_STATE_E enConnect;  
} VI_CONNECT_INFO_S;
```

【成员】

成员名称	描述
u32Width	通道接入图像宽度
u32Height	通道接入图像高度
f32FrameRate	通道输入帧率
enPixFmt	通道接入图像格式
enConnect	通道接入状态

【注意事项】

无

6.41 VI_CONNECT_STATE_E

【说明】

通道接入状态。

【定义】

```
typedef enum rkVI_CONNECT_STATE_E {  
    VI_CONNECT_STATE_UNKNOWN = 0,  
    VI_CONNECT_STATE_CONNECT,  
    VI_CONNECT_STATE_DISCONNECT,  
    VI_CONNECT_STATE_BUTT  
} VI_CONNECT_STATE_E;
```

【成员】

成员名称	描述
VI_CONNECT_STATE_UNKNOWN	未知
VI_CONNECT_STATE_CONNECT	连接
VI_CONNECT_STATE_DISCONNECT	未连接
VI_CONNECT_STATE_BUTT	最大值

【注意事项】

无

6.42 VI_EDID_S

【说明】

通道EDID信息结构体。

【定义】

```
typedef struct rkVI_EDID_S {  
    RK_U32 u32Pad;  
    RK_U32 u32StartBlock;  
    RK_U32 u32Blocks;  
    RK_U32 au32Reserved[5];  
    RK_U8 *pu8Edid;  
} VI_EDID_S;
```

【成员】

成员名称	描述
u32Pad	pad值，预留参数，设置0即可。
u32StartBlock	EDID 起始block。 设置EDID时只能设置0； 获取EDID时取值范围[0, u32Blocks)。
u32Blocks	EDID的block个数 设置时表示需要设置EDID的block个数； 获取时表示通道的EDID的block个数。
au32Reserved	预留参数，设置为0即可。
pu8Edid	EDID值。 每个block的EDID个数为128，第一个block起始地址pu8Edid[0]，第二个block起始地址pu8Edid[128]，以此类推。

【注意事项】

无

6.43 VI_STREAM_S

【说明】

stream 编码码流结构体。

【定义】

```
typedef struct rkVI_STREAM_S {  
    MB_BLK pMbBlk;  
    RK_U32 u32Len;  
    RK_U32 u32Seq;  
    RK_U64 u64PTS;  
} VI_STREAM_S;
```

【成员】

成员名称	描述
pMbBlk	码流内存句柄。
u32Len	码流长度。
u32Seq	码流序列号。
u64PTS	码流时间戳。单位us。

【注意事项】

无

6.44 VI_EVENT_CALL_BACK_S

【说明】

事件回调结构体。

【定义】

```
typedef struct rkVI_EVENT_CALL_BACK_S {
    RK_VI_EventCallBack pfnCallback;
    RK_VOID *pPrivateData;
} VI_EVENT_CALL_BACK_S;
```

【成员】

成员名称	描述
pfnCallback	事件回调函数指针。
pPrivateData	私有数据指针。

【注意事项】

无

6.45 RK_VI_EventCallBack

【说明】

事件回调函数。

【定义】

```
typedef void (*RK_VI_EventCallback)(RK_VOID *pPrivateData, VI_CB_INFO_S *pstInfo);
```

【注意事项】

无

6.46 VI_CB_INFO_S

【说明】

事件回调函数信息结构体。

【定义】

```
typedef struct rkVI_CB_INFO_S {  
    RK_U32 u32Event;  
} VI_CB_INFO_S;
```

【成员】

成员名称	描述
u32Event	事件类型。

【注意事项】

无

6.47 VI_EVENT_E

【说明】

事件类型结构体。

【定义】

```
typedef enum rkVI_EVENT_E {  
    VI_EVENT_CONNECT_CHANGE = 1 << 0,  
    VI_EVENT_SOURCE_CHANGE = 1 << 1,  
} VI_EVENT_E;
```

【成员】

成员名称	描述
VI_EVENT_CONNECT_CHANGE	插拔事件
VI_EVENT_SOURCE_CHANGE	输入源切换事件

6.48 DIS_CONFIG_S

【说明】

定义 DIS 的配置信息。

【定义】

```
typedef struct rkDIS_CONFIG_S {  
    DIS_MODE_E enMode;  
    DIS_MOTION_LEVEL_E enMotionLevel;  
    DIS_PDT_TYPE_E enPdtType;  
    RK_U32 u32BufNum;  
    RK_U32 u32CropRatio;  
    RK_U32 u32FrameRate;  
    RK_U32 u32GyroOutputRange;  
    RK_U32 u32GyroDataBitWidth;  
    RK_BOOL bCameraSteady;  
} DIS_CONFIG_S;
```

【成员】

成员名称	描述
enMode	DIS 中使用不同自由度防抖算法，共有 3 种不同算法。
enMotionLevel	Camera 的运动级别。
enPdtType	使用 DIS 的产品形态。
u32BufNum	DIS 用于缓存图像的 buf 数目，在 DIS 输出帧率偶尔出现丢帧时，可以增加缓存 buf 数。取值范围是 [0, 5]。
u32CropRatio	DIS 输出图像的裁剪比例。 取值范围为 [50, 98]。
u32FrameRate	设置帧率。无效参数，暂未使用。
u32GyroOutputRange	陀螺仪输出的角度范围。 取值范围为 [0, 360]。无效参数，暂未使用。
u32GyroDataBitWidth	陀螺仪数据的位宽。 取值范围为 [0, 32]。 通常设置为 15。无效参数，暂未使用。
bCameraSteady	镜头是否固定静止的。

【注意事项】

- enMode 目前只支持 DIS_MODE_4_DOF_GME。
- enPdtType 目前只支持 DIS_PDT_TYPE_IPC。
- bCameraSteady 只在 IPC 产品形态下有效，在 DV 和 DRONE 无人机产品形态下会默认设置为 RK_FALSE。
- [DIS_CONFIG_S](#) 中的参数都不可动态修改，如果需要修改，必须先关闭 DIS，再动态修改 [DIS_CONFIG_S](#) 中的参数。
- 对于参数未有任何修改，但调用 RK_MPI_VI_SetChnDISConfig 重复设置时，接口返回成功。
- 开启DIS 功能会对裁剪后图像进行放大操作，使输出图像与输入图像相同尺寸。

【相关数据类型及接口】

- [RK_MPI_VI_SetChnDISConfig](#)
- [RK_MPI_VI_GetChnDISConfig](#)
- [RK_MPI_VI_SetChnDISAttr](#)
- [RK_MPI_VI_GetChnDISAttr](#)

6.49 DIS_ATTR_S

【说明】

定义 DIS 的属性参数。

【定义】

```
typedef struct rkDIS_ATTR_S {
    RK_BOOL bEnable;
    RK_U32 u32MovingSubjectLevel;
    RK_S32 s32RollingShutterCoef;
    RK_S32 s32Timelag;
    RK_U32 u32ViewAngle;
    RK_U32 u32HorizontalLimit;
    RK_U32 u32VerticalLimit;
    RK_BOOL bStillCrop;
} DIS_ATTR_S;
```

【成员】

成员名称	描述
bEnable	DIS 使能开关。 RK_FALSE: 不使能； RK_TRUE: 使能。 动态参数。
u32MovingSubjectLevel	用于判断物体是否是运动的级别，取值范围为 [0,6]，级别越大，防抖效果越弱。值越小，运动过程中越稳定，但更容易出现偏移情况。值较大时，运动过程中背景抖动幅度大，但能够改善图像偏移现象。 请根据具体应用场景设置级别大小。 无效参数，未使用。
s32RollingShutterCoef	校正 rolling shutter 现象的校正参数。 取值范围为 [0, 1000]。 无效参数，未使用。
s32TimeLag	帧起始时间和陀螺仪数据采集时间的时间差，取值范围 [-2000000, 2000000]，单位为微秒。 无效参数，未使用。
u32ViewAngle	镜头水平视场角参数。 无效参数，未使用。
u32HorizontalLimit	水平偏移限制，取值范围 [0, 1000]。当大面积物体经过引起背景拖拽的水平偏移超过一定幅度时就不进行防抖。偏移幅度计算：2047 * u32HorizontalLimit / 1000。 无效参数，未使用。
u32VerticalLimit	垂直偏移限制，取值范围 [0, 1000]。当大面积物体经过引起背景拖拽的垂直偏移超过一定幅度时就不进行防抖。偏移幅度计算：2047 * u32VerticalLimit / 1000。 无效参数，未使用。
bStillCrop	关闭 DIS 防抖效果，但图像依旧保持裁剪比例输出的使能开关。 动态参数。

【注意事项】

- u32HorizontalLimit 和 u32VerticalLimit 只在 bCameraSteady 为 RK_TRUE 时生效，当 bCameraSteady 为 RK_FALSE 时，会默认设置为最大值。

【相关数据类型及接口】

- [RK_MPI_VI_SetChnDISAttr](#)
- [RK_MPI_VI_GetChnDISAttr](#)

6.50 CAC_EFFECT_ATTR_S

【说明】

定义 CAC 的效果参数。

【定义】

```
typedef struct rkCAC_EFFECT_ATTR_S {
    RK_U32 u32AutoHighLightDetect;      // {0, 1}
    RK_U32 u32AutoHighLightOffset;      // [0, 127]
    RK_U32 u32FixHighLightBase;        // [0, 255]
    RK_FLOAT fYCompensate;             // [0.0f, 1.0f]
    RK_FLOAT fAutoStrengthU;           // [0.0f, 1.0f]
    RK_FLOAT fAutoStrengthV;           // [0.0f, 1.0f]
    RK_FLOAT fGrayStrengthU;           // [0.0f, 1.0f]
    RK_FLOAT fGrayStrengthV;           // [0.0f, 1.0f]

    // Reserved params
    RK_FLOAT fRes1Params[8];          // [i&o] Reserved, do not care, set to 0
}
```

```

RK_FLOAT fResNParams[8];      // [i&o] Reserved, do not care, set to 0
} CAC_EFFECT_ATTR_S;

```

【成员】

成员名称	描述
u32AutoHighLightDetect	是否使用自适应亮度检测高光紫边。0为关闭，1为打开，推荐设置为1。
u32AutoHighLightOffset	为1时生效，亮度域（8bit）去紫边作用范围的偏移值，只有在附近亮度超过基准值加上偏移值的区域才会做去紫边（基准值是自适应计算的）。
u32FixHighLightBase	为0时生效，亮度域（8bit）去紫边作用范围的固定基准值，只有在附近亮度超过基准值的区域才会做去紫边。
fYCompensate	无效参数，预留参数。
fAutoStrengthU	根据检测到的紫边，自动判断所在点的实际颜色对U通道进行修复，0.0时为完全关闭，1.0时为完全打开，推荐值为1.0，尽可能打开到最大。
fAutoStrengthV	根据检测到的紫边，自动判断所在点的实际颜色对V通道进行修复，0.0时为完全关闭，1.0时为完全打开，推荐值为1.0，尽可能打开到最大。
fGrayStrengthU	根据检测到的紫边，按比例修改所在点的U通道的值，0.0时为完全关闭，1.0时为完全打开，推荐值 [0.3, 0.6]，过大的时候可能会造成紫边周围的图像不连贯。
fGrayStrengthV	根据检测到的紫边，按比例修改所在点的V通道的值，0.0时为完全关闭，1.0时为完全打开，推荐值 [0.0, 0.2]，过大的时候可能会造成紫边周围的图像不连贯。
fRes1Params	无效参数，预留参数。
fResNParams	无效参数，预留参数。

【注意事项】

- u32AutoHighLightOffset 设置越大，越检测亮处边缘的紫边；该参数设置越小，就更多检测暗处边缘的紫边。非 HDR 模式下，这个值推荐设置为96这种比较大的值（保证只在接近过曝的区域去紫边）。HDR 模式下，推荐设置0。
- u32FixHighLightBase 设置越大，越检测亮处边缘的紫边；该参数设置越小，就更多检测暗处边缘的紫边。

【相关数据类型及接口】

- [SWCAC_CONFIG_S](#)

6.51 SWCAC_CONFIG_S

【说明】

定义 CAC 的配置参数。

【定义】

```

typedef struct rkSWCAC_CONFIG_S {
    RK_BOOL    bEnable;        /* RW; SWCAC enable */
    CAC_EFFECT_ATTR_S stCacEffectAttr;
} SWCAC_CONFIG_S;

```

【成员】

成员名称	描述
bEnable	CAC 使能开关。 RK_FALSE: 不使能； RK_TRUE: 使能。 动态参数。
stCacEffectAttr	CAC 效果参数。 动态参数。

【注意事项】

【相关数据类型及接口】

- [RK_MPI_VI_SetSwcacConfig](#)
- [RK_MPI_VI_GetSwcacConfig](#)

6.52 VI_CHN_BUF_WRAP_S

【说明】

定义VI通道buffer的卷绕属性。

【定义】

```
typedef** *struct* rkVI_CHN_BUF_WRAP_S {
    RK_BOOL bEnable;
    RK_U32 u32BufLine;
    RK_U32 u32WrapBufferSize;
} VI_CHN_BUF_WRAP_S;
```

【成员】

成员名称	描述
bEnable	VI通道 Buffer 卷绕开关。
u32BufLine	卷绕 Buffer 行高。
u32WrapBufferSize	卷绕 Buffer 大小。

6.53 VI_ISP_MIRROR_FLIP_S

【说明】

定义VI通道镜像、翻转属性。

【定义】

```
typedef struct rkISP_MIRROR_FLIP_S {
    RK_U8 mirror;
    RK_U8 flip;
} VI_ISP_MIRROR_FLIP_S;
```

【成员】

成员名称	描述
mirror	镜像模式 (取值0、1)。
flip	翻转模式 (取值0、1)。

6.54 VI_CROP_COORDINATE_E

【说明】

VI 裁剪坐标类型枚举。

【定义】

```
typedef enum rkVI_CROP_COORDINATE_E {
    VI_CROP_RATIO_COOR = 0,
    VI_CROP_ABS_COOR,
    VI_CROP_BUTT
} VI_CROP_COORDINATE_E;
```

【成员】

成员名称	描述
VI_CROP_RATIO_COOR	相对坐标
VI_CROP_ABS_COOR	绝对坐标

6.55 VI_CROP_INFO_S

【说明】

VI 裁剪信息结构体。

【定义】

```
typedef struct rkVI_CROP_INFO_S {
    RK_BOOL bEnable;
    VI_CROP_COORDINATE_E enCropCoordinate;
    RECT_S stCropRect;
} VI_CROP_INFO_S;
```

【成员】

成员名称	描述
bEnable	VI使能裁剪功能开关。
enCropCoordinate	裁剪坐标系类型。
stCropRect	裁剪区域矩形结构体。

6.56 VI_LIGHT_TYPE_E

【说明】

补光灯类型枚举。

【定义】

```
typedef enum _rkVILightType{
    LIGHT_TYPE_PWM,
    LIGHT_TYPE_GPIO,
} VI_LIGHT_TYPE_E;
```

【成员】

成员名称	描述
LIGHT_TYPE_PWM	补光灯是 pwm 类型
LIGHT_TYPE_GPIO	补光灯是 gpio 类型

6.57 VI_LIGHT_CTL_PARAM_S

【说明】

VI 补光灯控制参数信息结构体。

【定义】

```
typedef struct rkVILightParam{  
    RK_U8 light_type;  
    RK_U8 light_enable;  
    RK_U64 duty_cycle;  
    RK_U64 period;  
    RK_U32 polarity;  
} VI_LIGHT_CTL_PARAM_S;
```

【成员】

成员名称	描述
light_type	补光灯类型。参考 VI_LIGHT_TYPE_E
light_enable	是否使能补光灯功能。
duty_cycle	pwm 占空比周期数
period	pwm 周期
polarity	pwm 极性

7. 环境变量

7.1 rt_vi_rawstream_en

【描述】

是否使能VI接管RawStream功能, 使用场景如手动灌raw数据用于isp效果调式时需要关闭VI的RawStream功能, 默认是打开状态。

【变量值说明】

变量值	描述
0	禁用
1	使能

8. VI错误码

视频输入API VI 错误码如下所示。

错误代码	宏定义	描述
0xA008001	RK_ERR_VI_INVALID_DEVID	VI设备号无效
0xA008002	RK_ERR_VI_INVALID_CHNID	VI通道号无效
0xA008003	RK_ERR_VI_INVALID_PARA	VI参数设置无效
0xA008004	RK_ERR_VI_EXIST	VI通道已存在
0xA008005	RK_ERR_VI_UNEXIST	不存在通道对象
0xA008006	RK_ERR_VI_INVALID_NULL_PTR	VI空指针错误
0xA008007	RK_ERR_VI_NOT_CONFIG	VI参数未配置
0xA008008	RK_ERR_VI_NOT_SUPPORT	操作不允许
0xA008009	RK_ERR_VI_NOT_PERM	无权限操作
0xA00800A	RK_ERR_VI_INVALID_PIPEID	VI PIPE号无效
0xA00800C	RK_ERR_VI_NOMEM	无内存
0xA00800E	RK_ERR_VI_BUF_EMPTY	VI输入缓冲为空
0xA00800F	RK_ERR_VI_BUF_FULL	VI输入缓存为满
0xA008010	RK_ERR_VI_SYS_NOTREADY	系统未准备好
0xA008012	RK_ERR_VI_BUSY	VI设备忙
0xA008014	RK_ERR_VI_DEV_EXIST	VI Dev或Dev资源已存在
0xA008015	RK_ERR_VI_DEV_UNEXIST	VI Dev或Dev资源不存在
0xA008016	RK_ERR_VI_PIPE_EXIST	VI Pipe或Pipe资源已存在
0xA008017	RK_ERR_VI_PIPE_UNEXIST	VI Pipe或Pipe资源不存在

视频处理子系统

[功能描述](#)

[API 参考](#)

[数据类型](#)

[VPSS错误码](#)

1. 功能描述

1.1 基本概念

1.1.1 GROUP

VPSS 对用户提供组（GROUP）的概念。最大个数请参见VPSS_MAX_GRP_NUM 定义，各 GROUP 分时复用硬件设备，硬件依次处理各个组提交的任务。

1.1.2 CHANNEL

VPSS 组的通道。提供多个通道，每个通道具有缩放、裁剪等功能。把图像裁剪、缩放成用户设置的目标分辨率输出。

1.1.3 CROP

裁剪，分为 2 种：组裁剪、通道裁剪。

- 组裁剪，VPSS 对输入图像进行裁剪。
- 通道裁剪，VPSS 利用硬件设备对各个通道的输出图像进行裁剪。

1.1.4 像素格式转换

支持输入输出图像的数据格式转换，例如NV12->RGB565等。

1.1.5 Scale

缩放，对图像进行缩小放大。组水平、垂直最大支持32倍放大、缩小；通道水平、垂直最大支持32倍放大、缩小。

1.1.6 Mirror/Flip

Mirror 即水平镜像，Flip 即垂直翻转。可使用 Mirror+Flip 实现 180°旋转。

1.1.7 FRC（帧率控制）

帧率控制分为两种：组帧率控制和通道帧率控制。

- 组帧率控制：用于控制各 Group 对输入图像的接收。
- 通道帧率控制：用于控制各个通道图像的处理。

1.1.8 Cover

视频遮挡区域，调用 VGS 对 VPSS 通道的输出图像填充纯色块。

1.1.9 Overlay

视频叠加区域，调用 VGS 对 VPSS 通道的输出图像叠加位图。

1.1.10 Mosaic

马赛克，调用 VGS 对 VPSS 通道的输出图像填充马赛克块。

1.1.11 RGN通道配置

RK356X/RK3588 芯片VPSS Group和通道均支持Overlay/Cover/Mosaic/Line等功能，使用RGN AttachChn时MPP_CHN_S->s32ChnId参数如果设置为[0, [VPSS_MAX_CHN_NUM](#))，则表示贴图至指定通道上；如果设置为[VPSS_MAX_CHN_NUM](#)，则表示贴图至Group上。

1.1.12 固定角度旋转

支持 0 度、90 度、180 度以及 270 度固定角度的旋转功能。

1.1.13 压缩

支持通道输出图像进行压缩，支持AFBC压缩。

1.1.14 解压

对输入的压缩图像进行解压缩处理，支持AFBC解压。

1.1.15 Aspect Ratio

幅形比。指定输出画面相对于输入画面的宽高纵横比。

1.1.16 通道工作模式

VPSS通道工作模式主要分为两种，通过改变[VPSS_CHN_MODE_E](#)切换模式，模式主要有两种：

- USER模式(VPSS_CHN_MODE_USER)：用户需要手动获取通道处理输出图像时设置，会带来性能开销。
- PAST模式(VPSS_CHN_MODE_PASSTHROUGH)：在绑定VO时使用，将会把处理工作放于VO中一并处理，不会带来额外开销，预览、回放时建议设置为此模式。

1.1.17 硬件设备类型

VPSS 可以使用的硬件设备有四种，通过[RK_MPI_VPSS_SetVProcDev](#) 接口设置 VIDEO_PROC_DEV_TYPE_E 类型切换硬件设备，在默认情况下，根据芯片种类有以下区分：

芯片名称	硬件设备类型
RV1109/RV1126	RGA（默认设备）、ISP
RV1103/RV1106	RGA（默认设备）
RK356X	GPU（默认设备）、RGA
RK3588	GPU（默认设备）、RGA
RK3567	GPU（默认设备）、RGA、VPSS

【注意】

- RGA

当硬件设备选择 RGA 时，有如下使用限制：

- 不支持 Mosaic (MOSAIC_RGN) 功能。
- 当底图为 YUV 时，不支持叠加带 alpha 格式（如：RK_FMT_RGBA8888、RK_FMT_BGRA5551）的 Overlay (OVERLAY_RGN) 功能。

- ISP

当硬件设备选择ISP时，有如下使用限制：

- 只支持Scale, Mirror/Flip、Rotation、Crop、Cover/Mosaic 均不支持
- 不支持OVERLAY_RGN，仅支持OVERLAY_EX_RGN形式的Overlay功能。
- ISPP分为Sensor和Input模式，VPSS使用的是Input模式，如果需要在两种模式中切换，需要使用media-ctl 命令设置ISPP模式、以及ISP和ISPP pipeline的分辨率。
- 通道0对应rkispp_m_bypass，通道1对应rkispp_scale0，通道2对应rkispp_scale1，通道3对应rkispp_scale2，使用限制遵循ISPP的限制。

- VPSS

当硬件设备选择 VPSS 时，有如下使用限制：

- 分辨率
 - 最小分辨率32x32，最大分辨率8192x6144，当输入图像分辨率大于4672x3504，小于8192x6144 时，自动切换为 unite 模式。
 - unite 模式不支持宽放大，高放大不限制。
 - unite 模式不支持 Tile 和 RKFBDCD 模式的输入格式。
 - unite 模式只支持NV12、NV16、NV21、NV61 的输出格式。
 - unite 模式不支持aspt。

- Format
 - 输入Format
 - Line 模式：NV12/NV16/RGB565/RGB24/XRGB32 (同时U/V swap 和 R/B swap 的也支持，包括 NV21/NV61/RGB565X/BGR24/XBGR32)。
 - Tile 模式：支持YUV422/YUV420。
 - RKFBDCD 模式：支持YUV444/YUV422/YUV420。
 - 输出Format
 - 所有channel都支持：NV12/NV16/UYVY/GREY(同时U/V swap的也支持，包括 NV21/NV61/UYVY，注意：U/V swap对所有channel生效)。
 - channel0额外支持：RGB565/RGB24/XBGR32 (同时 R/B swap 的也支持，包括 RGB565X/BGR24/XRGB32)。
 - 当 RKFBDCD 格式为 yuv444 时，其输出为 yuv422。

- Scale
 - channel1：支持横向8倍，纵向6倍的缩小或放大；其它channel：横向和纵向均支持32倍缩小或放大。
 - channel2、channel3：缩放输出宽度限制为1920，高度不限；如果纵向bypass，则横向缩放输出宽度不受1920的限制。

- Mirror/Flip
 - Mirror 作用于整个 vpss group。
 - Flip 4个chaannel 单独作用。

- Rotation
 - 只有 Tile 模式的输入图像支持旋转，其他图像格式不支持。

- 对齐

- Tile 模式的图像格式要求宽和虚宽4字节对齐。
- 其他图像格式虚宽4字节对齐，宽2字节对齐
- 高统一要求2对齐。
- 不支持 OVERLAY_RGN，仅支持 OVERLAY_EX_RGN 形式的 Overlay 功能。

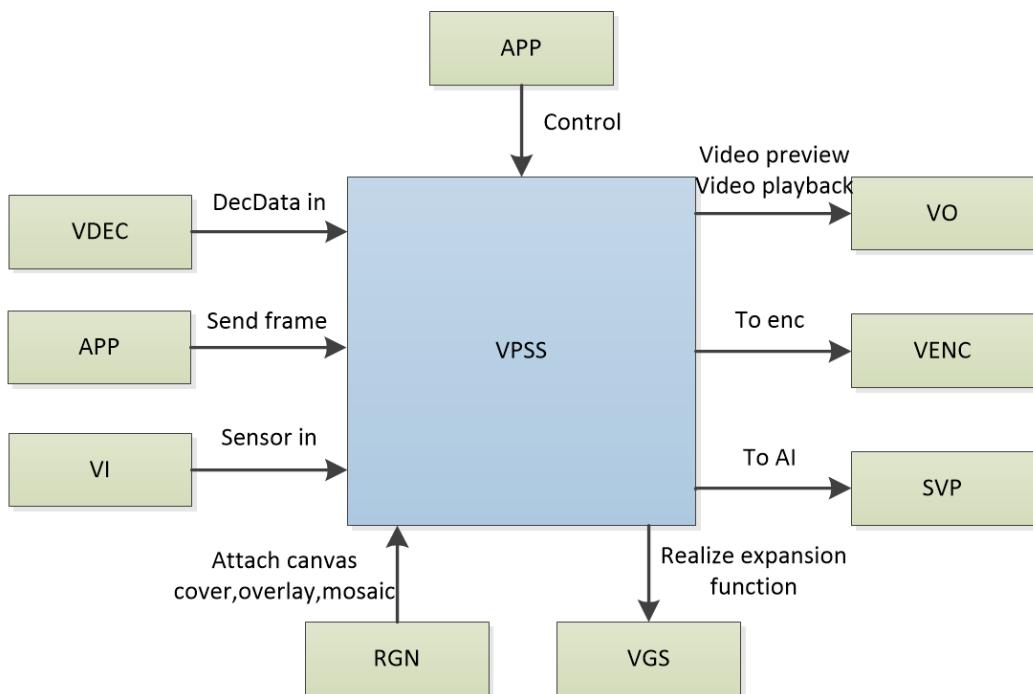
1.1.18 AI ISP

RV1106/RV1126/RV1109 芯片VPSS Group 支持通过AI ISP 对VI 输出图片进行微光降噪及智能增强处理，可在暗光弱光环境下，依然呈现出无拖影、低噪点、更清晰的画面。

1.2 应用方式

VPSS 在系统中的定位如图1-1所示。

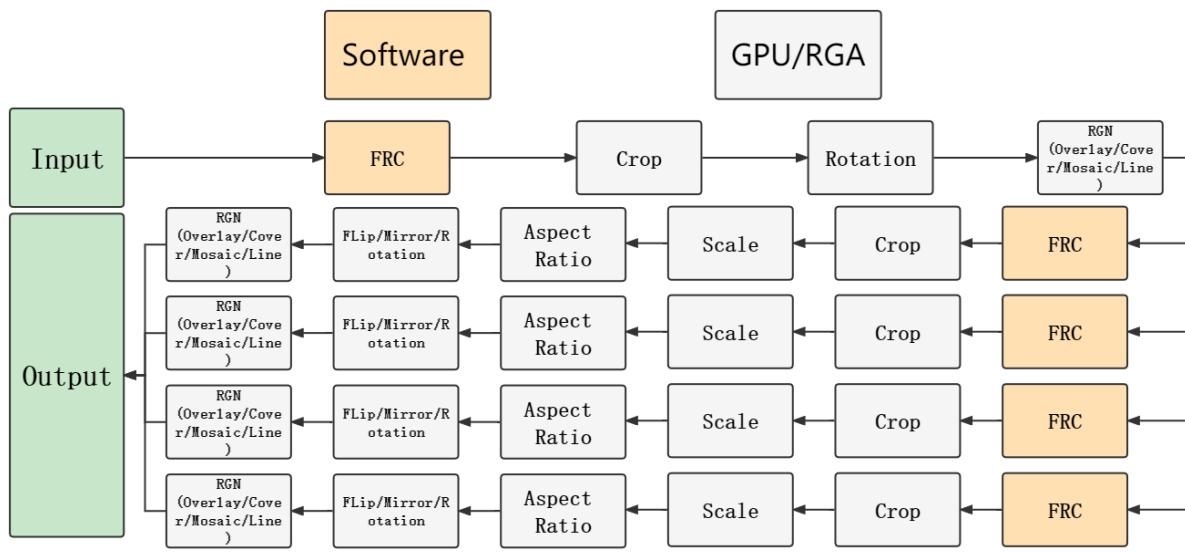
1.2.1 图1-1 VPSS上下文关系



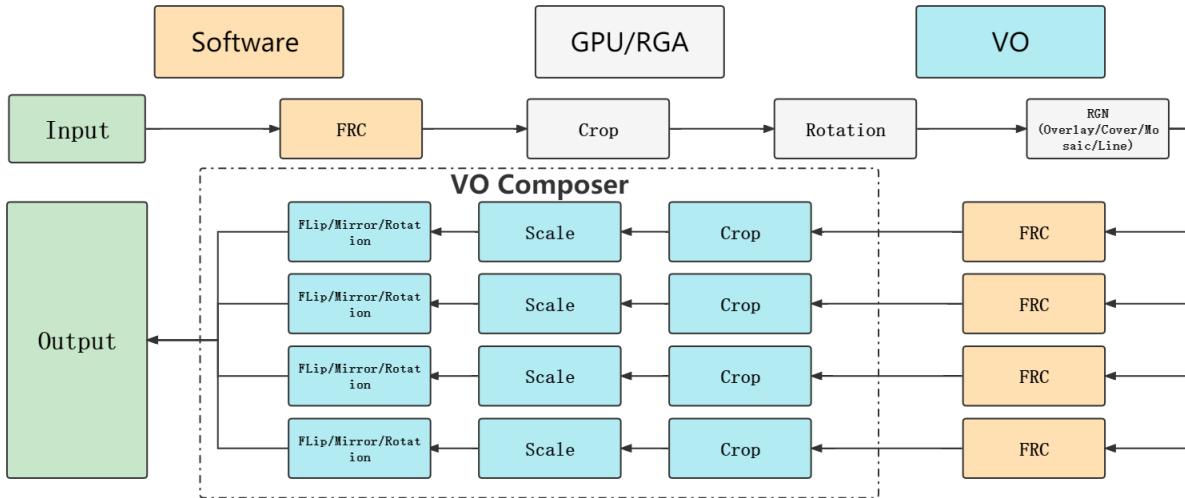
通过调用 SYS 模块的绑定接口，可与 VDEC/VI 和 VO/VENC/SVP 等模块进行绑定，其中前者为 VPSS 的输入源，后者为 VPSS 的接收者。用户可通过 VPSS MPI 接口对 Group 进行管理。每个 Group 仅可与一个输入源绑定。Group 的通道有两种工作模式：USER 和 PASSTHROUGH，两种模式间可动态切换。USER 模式下主要用于非绑定模式下，从通道手动取帧，用于AI算法、用户数据分析处理等场景，PASSTHROUGH 模式主要用于预览和回放场景下做播放控制，如电子放大。两种模式下各通道均可与多个接收者绑定。

1.3 芯片处理流程

1.3.1 RK356X/RK3588 VPSS USER模式数据流图



1.3.2 RK356X/RK3588 VPSS PAST模式数据流图



VPSS通道工作于PAST模式下，大部分数据处理（如上图虚线框内）是VPSS将处理参数透传给VO模块，由VO模块来一次性完成实际处理，进而节省硬件资源占用。

【注意事项】

- 在PAST模式时，如果当前为绑定状态时（自动送帧给下级时），RGN Attach到通道的贴图无效。

1.4 输入输出特性

1.4.1 输出图像格式及对齐方式

VPSS对齐方式均采用像素对齐。各芯片支持详情见下表：

1.4.1.1 RK356X/RK3588输入/输出图像支持列表

数据格式	宽度对齐像素	高度对齐像素	像素字节宽度 (bit)
RK_FMT_YUV420SP RK_FMT_YUV420SP_VU	16	1	8
RK_FMT_YUV420P	64	1	8
RK_FMT_YUV422SP	16	1	16
RK_FMT_YUV422_YUYV	32	1	16
RK_FMT_YUV400SP	64	1	8
RK_FMT_YUV420SP_10BIT	32	1	10
RK_FMT_RGB565 RK_FMT_BGR565 RK_FMT_RGBA5551 RK_FMT_BGRA5551	32	1	16
RK_FMT_RGB888 RK_FMT_BGR888	64	1	24
RK_FMT_BGRA8888 RK_FMT_RGBA8888	16	1	32

1.4.1.2 RV1109/RV1126输出图像支持列表

数据格式	宽度对齐像素	高度对齐像素	像素字节宽度 (bit)
RK_FMT_YUV420SP RK_FMT_YUV420SP_VU RK_FMT_YUV420P	4	2	8
RK_FMT_YUV422SP RK_FMT_YUV422P	4	2	8
RK_FMT_YUV400SP	4	2	8
RK_FMT_YUV420SP_10BIT RK_FMT_YUV422SP_10BIT	16	2	10
RK_FMT_YUV422_YUYV	4	2	8
RK_FMT_RGB888 RK_FMT_BGR888	4	1	24
RK_FMT_RGB565 RK_FMT_BGR565	2	1	16
RK_FMT_RGBA5551 RK_FMT_BGRA5551	2	1	16
RK_FMT_BGRA8888 RK_FMT_RGBA8888	1	1	32
RK_FMT_BGRA4444	2	1	16

1.4.1.3 RV1103/RV1106输出图像支持列表

数据格式	宽度对齐像素	高度对齐像素	像素字节宽度 (bit)
RK_FMT_YUV420SP RK_FMT_YUV420SP_VU RK_FMT_YUV420P	4	2	8
RK_FMT_YUV422SP RK_FMT_YUV422P	4	2	8
RK_FMT_YUV400SP	4	2	8
RK_FMT_YUV422_YUYV	4	2	8
RK_FMT_RGB888 RK_FMT_BGR888	4	1	24
RK_FMT_RGB565 RK_FMT_BGR565	2	1	16
RK_FMT_BGRA8888 RK_FMT_RGBA8888	1	1	32
RK_FMT_BGRA4444	2	1	16

1.5 举例

```

RK_S32 s32Ret = RK_SUCCESS;
VPSS_GRP VpssGrp = 0;
VPSS_CHN VpssChn[VPSS_MAX_CHN_NUM] = { VPSS_CHN0, VPSS_CHN1, VPSS_CHN2, VPSS_CHN3 };
VPSS_GRP_ATTR_S stGrpVpssAttr;
VPSS_CHN_ATTR_S stVpssChnAttr;

stGrpVpssAttr.u32MaxW = SRC_WIDTH;
stGrpVpssAttr.u32MaxH = SRC_HEIGHT;
stGrpVpssAttr.enPixelFormat = PIXEL_FORMAT_YUV_SEMIPLANAR_420;
stGrpVpssAttr.enCompressMode = COMPRESS_AFBC_16x16;
stGrpVpssAttr.stFrameRate.s32SrcFrameRate = -1;
stGrpVpssAttr.stFrameRate.s32DstFrameRate = -1;
s32Ret = RK_MPI_VPSS_CreateGrp(VpssGrp, &stGrpVpssAttr);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}

stCropInfo.bEnable = RK_TRUE;
stCropInfo.enCropCoordinate = VPSS_CROP_ABS_COOR;
stCropInfo.stCropRect.s32X = 640;
stCropInfo.stCropRect.s32Y = 360;
stCropInfo.stCropRect.u32Width = 640;
stCropInfo.stCropRect.u32Height = 360;
s32Ret = RK_MPI_VPSS_SetGrpCrop(VpssGrp, &stCropInfo);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}
s32Ret = RK_MPI_VPSS_GetGrpCrop(VpssGrp, &stCropInfo);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}
memset(&stVpssChnAttr, 0, sizeof(VPSS_CHN_ATTR_S));
stVpssChnAttr.enChnMode = VPSS_CHN_MODE_USER;
stVpssChnAttr.enCompressMode = COMPRESS_MODE_NONE;
stVpssChnAttr.enDynamicRange = DYNAMIC_RANGE_SDR8;
stVpssChnAttr.enPixelFormat = PIXEL_FORMAT_YUV_SEMIPLANAR_420;
stVpssChnAttr.stFrameRate.s32SrcFrameRate = -1;
stVpssChnAttr.stFrameRate.s32DstFrameRate = -1;
stChnCropInfo.bEnable = RK_TRUE;

```

```

stChnCropInfo.enCropCoordinate = VPSS_CROP_RATIO_COOR;
stChnCropInfo.stCropRect.s32X = 500;
stChnCropInfo.stCropRect.s32Y = 500;
stChnCropInfo.stCropRect.u32Width = 500;
stChnCropInfo.stCropRect.u32Height = 500;
for (RK_S32 i = 0; i < VPSS_MAX_CHN_NUM; i++) {
    stVpssChnAttr.u32Width = SRC_WIDTH / VPSS_MAX_CHN_NUM * (i + 1);
    stVpssChnAttr.u32Height = SRC_HEIGHT / VPSS_MAX_CHN_NUM * (i + 1);
    s32Ret = RK_MPI_VPSS_SetChnAttr(VpssGrp, VpssChn[i], &stVpssChnAttr);
    if (s32Ret != RK_SUCCESS) {
        return s32Ret;
    }
    s32Ret = RK_MPI_VPSS_GetChnAttr(VpssGrp, VpssChn[i], &stVpssChnAttr);
    if (s32Ret != RK_SUCCESS) {
        return s32Ret;
    }
    s32Ret = RK_MPI_VPSS_SetChnCrop(VpssGrp, VpssChn[i], &stChnCropInfo);
    if (s32Ret != RK_SUCCESS) {
        return s32Ret;
    }
    s32Ret = RK_MPI_VPSS_GetChnCrop(VpssGrp, VpssChn[i], &stChnCropInfo);
    if (s32Ret != RK_SUCCESS) {
        return s32Ret;
    }
    s32Ret = RK_MPI_VPSS_EnableChn(VpssGrp, VpssChn[i]);
    if (s32Ret != RK_SUCCESS) {
        return s32Ret;
    }
}
s32Ret = RK_MPI_VPSS_StartGrp(VpssGrp);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}
s32Ret = RK_MPI_VPSS_StopGrp(VpssGrp);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}
for (RK_S32 i = 0; i < VPSS_MAX_CHN_NUM; i++) {
    s32Ret = RK_MPI_VPSS_DisableChn(VpssGrp, VpssChn[i]);
    if (s32Ret != RK_SUCCESS) {
        return s32Ret;
    }
}
s32Ret = RK_MPI_VPSS_DestroyGrp(VpssGrp);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}

```

2. API 参考

该功能模块为用户提供以下 API：

- [RK MPI VPSS CreateGrp](#): 创建一个 VPSS GROUP。
- [RK MPI VPSS DestroyGrp](#): 销毁一个 VPSS GROUP。
- [RK MPI VPSS StartGrp](#): 启用 VPSS GROUP。
- [RK MPI VPSS StopGrp](#): 禁用 VPSS GROUP。
- [RK MPI VPSS ResetGrp](#): 重置一个 VPSS GROUP。
- [RK MPI VPSS GetGrpAttr](#): 获取 VPSS GROUP 属性。
- [RK MPI VPSS SetGrpAttr](#): 设置 VPSS GROUP 属性。
- [RK MPI VPSS SetGrpCrop](#): 设置 VPSS GROUP CROP 功能属性。
- [RK MPI VPSS GetGrpCrop](#): 获取 VPSS GROUP CROP 功能属性。
- [RK MPI VPSS SetGrpRotation](#): 设置 VPSS GROUP 图像固定角度旋转属性
- [RK MPI VPSS GetGrpRotation](#): 获取 VPSS GROUP 图像固定角度旋转属性。

- [RK_MPI_VPSS_SendFrame](#): 用户向 VPSS GROUP 发送数据。
- [RK_MPI_VPSS_GetGrpFrame](#): 用户从 VPSS GROUP 获取一帧原始图像。
- [RK_MPI_VPSS_ReleaseGrpFrame](#): 用户释放一帧原始图像。
- [RK_MPI_VPSS_EnableBackupFrame](#): 使能 backup 帧。
- [RK_MPI_VPSS_DisableBackupFrame](#): 不使能 backup 帧。
- [RK_MPI_VPSS_SetChnAttr](#): 设置 VPSS 通道属性。
- [RK_MPI_VPSS_GetChnAttr](#): 获取 VPSS 通道属性。
- [RK_MPI_VPSS_EnableChn](#): 启用 VPSS 通道。
- [RK_MPI_VPSS_DisableChn](#): 禁用 VPSS 通道。
- [RK_MPI_VPSS_SetChnCrop](#): 设置 VPSS 通道裁剪功能属性。
- [RK_MPI_VPSS_GetChnCrop](#): 获取 VPSS 通道裁剪功能属性。
- [RK_MPI_VPSS_SetChnRotation](#): 设置 VPSS 通道图像固定角度旋转属性。
- [RK_MPI_VPSS_GetChnRotation](#): 获取 VPSS 通道图像固定角度旋转属性。
- [RK_MPI_VPSS_SetChnRotationEx](#): 设置 VPSS 的任意角度旋转属性。
- [RK_MPI_VPSS_GetChnRotationEx](#): 获取 VPSS 的任意角度旋转属性。
- [RK_MPI_VPSS_GetChnFrame](#): 用户获取一帧通道图像。
- [RK_MPI_VPSS_ReleaseChnFrame](#): 用户释放一帧通道图像。
- [RK_MPI_VPSS_AttachMbPool](#): 将 VPSS 的通道绑定到某个视频缓存 MB 池中。
- [RK_MPI_VPSS_DetachMbPool](#): 将 VPSS 的通道从某个视频缓存 MB 池中解绑定。
- [RK_MPI_VPSS_GetChnFd](#): 获取 VPSS 通道对应的设备文件句柄。
- [RK_MPI_VPSS_CloseFd](#): 关闭设备和通道的文件描述符。
- [RK_MPI_VPSS_SetVProcDev](#): 设置 VPSS 的硬件设备类型。
- [RK_MPI_VPSS_GetVProcDev](#): 获取 VPSS 的硬件设备类型。
- [RK_MPI_VPSS_SetModParam](#): 设置 VPSS 的模块参数。
- [RK_MPI_VPSS_GetModParam](#): 获取 VPSS 的模块参数。
- [RK_MPI_VPSS_GetGrpAIISPAttr](#): 获取 VPSS 的 AI ISP 参数。
- [RK_MPI_VPSS_SetGrpAIISPAttr](#): 设置 VPSS 的 AI ISP 参数。

2.1 RK_MPI_VPSS_CreateGrp

【描述】

创建一个 VPSS GROUP。

【语法】

```
RK_S32 RK_MPI_VPSS_CreateGrp(VPSS\_GRP VpssGrp, const VPSS\_GRP\_ATTR\_S *pstGrpAttr);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
pstGrpAttr	VPSS GROUP 属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 不支持重复创建。重复创建返回[RK_ERR_VPSS_NOT_PERM](#)错误。

2.2 RK_MPI_VPSS_DestroyGrp

【描述】

销毁一个 VPSS GROUP。

【语法】

```
RK_S32 RK_MPI_VPSS_DestroyGrp(VPSS\_GRP VpssGrp);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- 不支持重复销毁, 重复销毁返回**RK_ERR_VPSS_UNEXIST**错误。
- 调用此接口之前, 必须先调用[RK_MPI_VPSS_StopGrp](#)禁用此 GROUP。
- 调用此接口时, 会一直等待此 GROUP 当前任务处理结束才会真正销毁。

2.3 RK_MPI_VPSS_StartGrp

【描述】

启用 VPSS GROUP。

【语法】

```
RK_S32 RK_MPI_VPSS_StartGrp(VPSS\_GRP VpssGrp);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- 同组调用该函数仅第一次调用成功, 重复调用返回失败。

2.4 RK_MPI_VPSS_StopGrp

【描述】

禁用 VPSS GROUP。

【语法】

```
RK_S32 RK_MPI_VPSS_StopGrp(VPSS\_GRP VpssGrp);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- 同组调用该函数仅第一次调用成功, 重复调用返回失败。

2.5 RK_MPI_VPSS_ResetGrp

【描述】

复位 VPSS GROUP。

【语法】

```
RK_S32 RK_MPI_VPSS_ResetGrp(VPSS\_GRP VpssGrp);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- 组及通道中待处理或待消耗的数据将会被释放。

2.6 RK_MPI_VPSS_GetGrpAttr

【描述】

获取 VPSS GROUP 属性。

【语法】

```
RK_S32 RK_MPI_VPSS_GetGrpAttr(VPSS\_GRP VpssGrp, VPSS\_GRP\_ATTR\_S *pstGrpAttr);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
pstGrpAttr	VPSS GROUP 属性指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- GROUP 属性必须合法, 其中部分静态属性不可动态设置。

2.7 RK_MPI_VPSS_SetGrpAttr

【描述】

设置 VPSS GROUP 属性。

【语法】

```
RK_S32 RK_MPI_VPSS_SetGrpAttr(VPSS\_GRP VpssGrp, const VPSS\_GRP\_ATTR\_S *pstGrpAttr);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
pstGrpAttr	VPSS GROUP 属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- GROUP 属性必须合法, 其中部分静态属性不可动态设置。

2.8 RK_MPI_VPSS_SetGrpCrop

【描述】

设置 VPSS CROP 功能属性。

【语法】

```
RK_S32 RK_MPI_VPSS_SetGrpCrop(VPSS\_GRP VpssGrp, const VPSS\_CROP\_INFO\_S *pstCropInfo);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
pstCropInfo	CROP 功能参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- 相对模式裁剪时, 裁剪区域坐标取值范围为[0, 999], 裁剪区域宽高取值范围为[1,1000]。
- CROP 区域的尺寸不能小于 VPSS 最小尺寸, 不能超过 VPSS 支持的最大输入分辨率; 裁剪区域起始点不支持负坐标, 裁剪区域右边界不能超出 VPSS 支持的最大输入宽度, 裁剪区域下边界不能超出 VPSS 支持的最大输入高度。
- 如果裁剪区域超出图像范围, 裁剪坐标向原点方向移动, 优先保证裁剪出的宽高与所设置的参数相同。
- 如果裁剪宽度大于输入图像宽度, 则裁剪输出宽度调整为输入图像宽度。
- 如果裁剪高度大于输入图像高度, 则裁剪输出高度调整为输入图像高度。
- 在有绑定VO时, 不推荐使用此接口做电子放大, 建议通道模式设置为PAST模式, 并使用[RK_MPI_VPSS_SetChnCrop](#)做电子放大功能。
- 通道设置为PAST时, 只有在绑定了VO的情况下才生效。
- RV1103/RV1106 不支持该接口。

2.9 RK_MPI_VPSS_GetGrpCrop

【描述】

获取 VPSS CROP 功能属性。

【语法】

```
RK_S32 RK_MPI_VPSS_GetGrpCrop(VPSS\_GRP VpssGrp, VPSS\_CROP\_INFO\_S *pstCropInfo);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
pstCropInfo	CROP 功能参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- RV1103/RV1106 不支持该接口。

2.10 RK_MPI_VPSS_SetGrpRotation

【描述】

设置 VPSS GROUP 图像固定角度旋转属性。

【语法】

```
RK_S32 RK_MPI_VPSS_SetGrpRotation(VPSS\_GRP VpssGrp, ROTATION\_E enRotation);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
enRotation	旋转属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- 仅支持 0 度、90 度、180 度、270 度的旋转, 不支持任意角度旋转。

2.11 RK_MPI_VPSS_GetGrpRotation

【描述】

获取 VPSS GROUP 图像固定角度旋转属性。

【语法】

```
RK_S32 RK_MPI_VPSS_GetGrpRotation(VPSS\_GRP VpssGrp, ROTATION\_E*penRotation);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
penRotation	旋转属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。

2.12 RK_MPI_VPSS_SendFrame

【描述】

用户向 VPSS 发送数据。

【语法】

```
RK_S32 RK_MPI_VPSS_SendFrame(VPSS\_GRP VpssGrp, VPSS\_GRP\_PIPE VpssGrpPipe, const VIDEO\_FRAME\_INFO\_S*pstVideoFrame, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssGrpPipe	VPSS 组的管道号。取值只能为 0。	输入
pstVideoFrame	待发送的图像信息。	输入
s32MilliSec	超时参数 s32MilliSec 设为-1 时, 为阻塞接口; 0 时为非阻塞接口; 大于 0 时为超时等待时间, 超时时间的单位为毫秒 (ms)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- 用户使用此接口时, 可以自行进行帧率控制。
- VPSS 通道设置为 PAST 模式时使用此接口前, VPSS 应正常启动, 推荐与VO绑定, 否则用户通过获取图像接口获取的图像将会使用VGS处理; 或 VPSS 通道设置为 USER 模式, 用户通过获取图像接口获取图像。

2.13 RK_MPI_VPSS_GetGrpFrame

【描述】

用户从 GROUP 获取一帧原始图像。主要应用场景：高清设备解码回放，要求暂停、步进时，PIP 层和普通视频层上的两个通道显示同一帧图像。通过本接口和RK_MPI_VPSS_SendFrame 等接口的配合使用，可实现该功能。

【语法】

```
RK_S32 RK_MPI_VPSS_GetGrpFrame(VPSS\_GRP VpssGrp, VPSS\_GRP\_PIPE VpssGrpPipe, VIDEO\_FRAME\_INFO\_S*pstVideoFrame);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssGrpPipe	VPSS 组的管道号。取值只能为 0。	输入
pstVideoFrame	图像信息。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- 获取的图像要及时释放, 否则将造成 MB 不够或解码回放时停止, 建议与RK_MPI_VPSS_ReleaseGrpFrame 接口配对使用。
- 使能了 backup 帧时才能获取。
- 有绑定VO时才能获取。
- 通道设置为PAST模式时才可获取。

2.14 RK_MPI_VPSS_ReleaseGrpFrame

【描述】

用户释放一帧源图像。

【语法】

```
RK_S32 RK_MPI_VPSS_ReleaseGrpFrame(VPSS\_GRP VpssGrp, VPSS_GRP_PIPE VpssGrpPipe, const VIDEO_FRAME_INFO_S *pstVideoFrame);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssGrpPipe	VPSS 组的管道号。取值只能为 0。	输入
pstVideoFrame	图像信息。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 实际上, 此接口的 VpssGrp 参数并无实际用途, 可在取值范围内任意设置。注意PIPE 号取值只能为 0。
- 此接口需与 [RK_MPI_VPSS_GetGrpFrame](#) 配对使用。

2.15 RK_MPI_VPSS_EnableBackupFrame

【描述】

使能 Backup 帧。

【语法】

```
RK_S32 RK_MPI_VPSS_EnableBackupFrame(VPSS\_GRP VpssGrp);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。

2.16 RK_MPI_VPSS_DisableBackupFrame

【描述】

不使能 Backup 帧。

【语法】

RK_S32 RK_MPI_VPSS_DisableBackupFrame([VPSS_GRP](#) VpssGrp)

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- RV1103/RV1106 不支持该接口。

2.17 RK_MPI_VPSS_SetChnAttr

【描述】

设置 VPSS 通道属性。

【语法】

RK_S32 RK_MPI_VPSS_SetChnAttr([VPSS_GRP](#) VpssGrp, [VPSS_CHN](#) VpssChn, const [VPSS_CHN_ATTR_S](#) *pstChnAttr);

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入
pstChnAttr	VPSS 通道属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- 具体成员取值及功能限制参见[VPSS_CHN_ATTR_S](#) 结构体说明。

2.18 RK_MPI_VPSS_GetChnAttr

【描述】

获取 VPSS 通道属性。

【语法】

```
RK_S32 RK_MPI_VPSS_GetChnAttr(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn, VPSS\_CHN\_ATTR\_S*pstChnAttr);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入
pstChnAttr	VPSS 通道属性。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。

2.19 RK_MPI_VPSS_EnableChn

【描述】

启用 VPSS 通道。

【语法】

```
RK_S32 RK_MPI_VPSS_EnableChn(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 多次使能仅第一次返回成功, 后续调用返回失败。
- GROUP 必须已创建。

2.20 RK_MPI_VPSS_DisableChn

【描述】

禁用 VPSS 通道。

【语法】

```
RK_S32 RK_MPI_VPSS_DisableChn(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 多次禁用仅第一次返回成功, 后续调用返回失败。
- GROUP 必须已创建。

2.21 RK_MPI_VPSS_SetChnCrop

【描述】

设置 VPSS 通道裁剪功能属性。

【语法】

```
RK_S32 RK_MPI_VPSS_SetChnCrop(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn, const VPSS\_CROP\_INFO\_S *pstCropInfo);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入
pstCropInfo	CROP 功能参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- 如果裁剪区域超出图像范围, 裁剪坐标向原点方向移动, 优先保证裁剪出的宽高与所设置的参数相同。
- 如果裁剪宽度大于输入图像宽度, 则裁剪输出宽度调整为输入图像宽度。
- 如果裁剪高度大于输入图像高度, 则裁剪输出高度调整为输入图像高度。
- 其他限制与[RK_MPI_VPSS_SetGrpCrop](#)相同。

2.22 RK_MPI_VPSS_GetChnCrop

【描述】

获取 VPSS 通道裁剪功能属性。

【语法】

```
RK_S32 RK_MPI_VPSS_GetChnCrop(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn, VPSS\_CROP\_INFO\_S*pstCropInfo);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入
pstCropInfo	CROP 功能参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。

2.23 RK_MPI_VPSS_SetChnRotation

【描述】

设置 VPSS 通道图像固定角度旋转属性。

【语法】

```
RK_S32 RK_MPI_VPSS_SetChnRotation(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn, ROTATION_E enRotation);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入
enRotation	旋转属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- 通道属性必须已设置。
- 仅支持 0 度、90 度、180 度、270 度的旋转, 不支持任意角度旋转。
- 通道为 PAST 模式时不生效。
- RV1126 旋转角度的优先级比使能水平镜像、使能垂直翻转的优先级都低, 以下是具体情况:

旋转角度	使能水平镜像	使能垂直翻转	输出效果
0 度 90 度 270 度	使能	禁止	水平镜像
180 度	使能	禁止	垂直反转
0 度 90 度 270 度	禁止	使能	垂直反转
180 度	禁止	使能	水平镜像
0 度 90 度 180 度 270 度	使能	使能	180 度 270 度 0 度 90 度

- RK356X/RK3588 旋转角度和使能水平镜像、使能垂直翻转均可叠加操作, 以下是具体情况:

旋转角度	使能水平镜像	使能垂直翻转	输出效果
0 度 90 度 270 度	使能	禁止	水平镜像+旋转角度叠加
180 度	使能	禁止	垂直反转
0 度 90 度 270 度	禁止	使能	垂直反转+旋转角度叠加
180 度	禁止	使能	水平镜像
0 度 90 度 180 度 270 度	使能	使能	180 度 270 度 0 度 90 度

2.24 RK_MPI_VPSS_GetChnRotation

【描述】

获取 VPSS 通道图像固定角度旋转属性。

【语法】

```
RK_S32 RK_MPI_VPSS_GetChnRotation(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn, ROTATION_E *penRotation);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围：[0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围：[0, VPSS_MAX_CHN_NUM)。	输入
penRotation	旋转属性。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。

2.25 RK_MPI_VPSS_SetChnRotationEx

【描述】

设置 VPSS 的任意角度旋转属性。

【语法】

```
RK_S32 RK_MPI_VPSS_SetChnRotationEx(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn, const VPSS\_ROTATION\_EX\_ATTR\_S* pstRotationExAttr);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入
pstRotationExAttr	任意角度旋转属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- 必须在设置通道属性后才能设置此属性。
- 此接口与 [RK_MPI_VPSS_SetChnRotation](#) 接口不能同时使用。

2.26 RK_MPI_VPSS_GetChnRotationEx

【描述】

获取 VPSS 的任意角度旋转属性。

【语法】

```
RK_S32 RK_MPI_VPSS_GetChnRotationEx(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn, VPSS\_ROTATION\_EX\_ATTR\_S*  
pstRotationExAttr);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入
pstRotationExAttr	任意角度旋转属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。

2.27 RK_MPI_VPSS_GetChnFrame

【描述】

用户从通道获取一帧处理完成的图像。

【语法】

```
RK_S32 RK_MPI_VPSS_GetChnFrame(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn, const VIDEO_FRAME_INFO_S
*pstVideoFrame, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入
pstVideoFrame	处理完成的图像信息。	输出
s32MilliSec	超时参数 s32MilliSec 设为-1 时, 为阻塞接口; 0 时为非阻塞接口; 大于 0 时为超时等待时间, 超时时间的单位为毫秒 (ms) 。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- GROUP 必须已创建。
- 队列深度不为 0, 才能获取到图像。
- 调用该接口获取图像, 不会对后端绑定的模块有影响。如后端绑定 VO 显示, 可以在显示过程中获取图像, VO 仍正常显示, 不会受到影响。
- 当 s32MilliSec 设为-1 时, 表示阻塞模式, 程序一直等待, 直到获取到图像才返回。如果 s32MilliSec 等于 0 时, 表示非阻塞模式。如果 s32MilliSec 大于 0 时, 表示超时等待模式, 参数的单位是毫秒, 指超时时间, 在此时间内如果没有获取到图像, 则超时返回。
- 通道为PAST模式时, 调用该接口时会使用VGS进行图像处理。
- 默认在私有MB模式下, 通道为USER模式时, 最大输出帧个数限制到3个。
- 默认在私有MB模式下, 通道为PAST模式时, 不限制最大输出帧个数。
- 推荐队列深度设置为3个以内, 否则在队列输出图像超过3个时, VPSS将会停止工作, 同时同一Group的其他通道也将停止工作。

2.28 RK_MPI_VPSS_ReleaseChnFrame

【描述】

用户释放一帧通道图像。

【语法】

```
RK_S32 RK_MPI_VPSS_ReleaseChnFrame(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn, const VIDEO_FRAME_INFO_S
*pstVideoFrame);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围[0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入
pstVideoFrame	图像信息。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 此接口需与[RK_MPI_VPSS_GetChnFrame](#)配对使用。
- 接口调用同[RK_MPI_VPSS_GetChnFrame](#)调用次数一一对应, 不允许同一帧多次调用。
- 通过[RK_MPI_VPSS_GetChnFrame](#)获取的帧, 需要及时调用该接口释放, 否则达到最大输出帧个数后无法获取新的帧。

2.29 RK_MPI_VPSS_AttachMbPool

【描述】

将 VPSS 的通道绑定到某个视频缓存 MB 池中。

【语法】

```
RK_S32 RK_MPI_VPSS_AttachMbPool(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn, MB_POOL hMbPool);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入
hMbPool	视频缓存 MB 池信息。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 必须保证组已创建。
- 仅在通道为USER模式时生效, 通道为PAST时不受影响。
- 用户必须调用接口[RK_MPI_MB_CreatePool](#)创建一个视频缓存 MB 池, 再通过调用接口[RK_MPI_VPSS_AttachMbPool](#)把当前组的通道绑定到固定 PoolId 的 MB 池中。支持多个组的多个通道绑定到同一个 MB 池中。
- 当要切换当前组绑定的 MB 池时, 只需再调一次接口[RK_MPI_VPSS_AttachMbPool](#)正确配置需要绑定到的 MB 池即可。
- hMbPool 必须保证是已创建 MB 池的有效 PoolId。
- 在调用[RK_MPI_VPSS_DetachMbPool](#)后, 销毁创建的 MB 之前, 需要保证 MB 没有被 VPSS 后端绑定的模块使用, 可以通过 sleep 或清除后端模块通道缓存的方式先把 MB 都释放, 再销毁缓存 MB 池。
- 绑定后, 通道申请的 MB 均是从此 MB 池中获取。
- RV1103/RV1106 不支持该接口。

2.30 RK_MPI_VPSS_DetachMbPool

【描述】

将 VPSS 的通道从某个视频缓存 MB 池中解绑定。

【语法】

```
RK_S32 RK_MPI_VPSS_DetachMbPool(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 必须保证组已创建。
- RV1103/RV1106 不支持该接口。

2.31 RK_MPI_VPSS_GetChnFd

【描述】

获取 VPSS 通道对应的设备文件句柄。

【语法】

```
RK_S32 RK_MPI_VPSS_GetChnFd(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 必须保证组已创建。

2.32 RK_MPI_VPSS_CloseFd

【描述】

关闭设备和通道的文件描述符。

【语法】

```
RK_S32 RK_MPI_VPSS_CloseFd(VPSS\_GRP VpssGrp, VPSS\_CHN VpssChn);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号。 取值范围: [0, VPSS_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 必须保证组已创建。
- 允许重复调用, 均返回成功。

2.33 RK_MPI_VPSS_SetVProcDev

【描述】

设置 VPSS 的硬件设备类型。

【语法】

```
RK_S32 RK_MPI_VPSS_SetVProcDev(VPSS\_GRP VpssGrp, VIDEO\_PROC\_DEV\_TYPE\_E enVProcDev);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
enVProcDev	硬件设备类型。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 必须保证组已创建。
- 不调用接口, 默认 GPU 作为模块 VPSS 的硬件设备。

2.34 RK_MPI_VPSS_GetVProcDev

【描述】

获取 VPSS 的硬件设备类型。

【语法】

```
RK_S32 RK_MPI_VPSS_GetVProcDev(VPSS\_GRP VPSS_GRP VpssGrp, VIDEO\_PROC\_DEV\_TYPE\_E *enVProcDev);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
enVProcDev	硬件设备类型。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 必须保证组已创建。

2.35 RK_MPI_VPSS_SetModParam

【描述】

设置 VPSS 的模块参数。

【语法】

```
RK_S32 RK_MPI_VPSS_SetModParam(const VPSS\_MOD\_PARAM\_S *pstModParam);
```

【参数】

参数名	描述	输入/输出
pstModParam	模块参数	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 通常在所有VPSS组未被创建前调用。
- 允许VPSS组已被创建调用, 但在正在使用的组中并不会立即生效, 重新释放和申请该组后生效。

2.36 RK_MPI_VPSS_GetModParam

【描述】

获取 VPSS 的模块参数。

【语法】

```
RK_S32 RK_MPI_VPSS_GetModParam(VPSS\_MOD\_PARAM\_S *pstModParam);
```

【参数】

参数名	描述	输入/输出
pstModParam	模块参数、	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 无

2.37 RK_MPI_VPSS_SetGrpAIISPAttr

【描述】

设置 VPSS 的 AI ISP 参数。

【语法】

```
RK_S32 RK_MPI_VPSS_SetGrpAIISPAttr(VPSS_GRP VpssGrp, const AIISP_ATTR_S *pstAilspAttr);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
pstAilspAttr	AIISP 参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 该接口仅RV1103/RV1106/RV1109/RV1126 平台支持。
- 重复调用接口使能 AI ISP 功能将报错返回。

【参考代码】

- AI ISP 功能设置及开启。

```
#include "rkpostisp.h" // for struct rk_ainr_param

/* after rkaiq init */
static RK_S32 aiisp_callback(RK_VOID *pAinrParam, RK_VOID *pPrivateData) {
    if (pAinrParam == RK_NULL) {
        return RK_FAILURE;
    }
    memset(pAinrParam, 0, sizeof(rk_ainr_param));
    SAMPLE_COMM_ISP_GetAINrParams(0, pAinrParam);

    return RK_SUCCESS;
}

RK_S32 s32Ret = RK_FAILURE;
VPSS_GRP vpssGrpId = 0;
AIISP_ATTR_S stAIISPAttr;

/* after vpss vpssGrpId create */
memset(&stAIISPAttr, 0, sizeof(AIISP_ATTR_S));
stAIISPAttr.bEnable = RK_TRUE;
```

```

stAIISPAttr.stAilspCallback.pfUpdateCallback = aiisp_callback;
stAIISPAttr.stAilspCallback.pPrivateData = RK_NULL;
stAIISPAttr.pModelFilePath = "/oem/usr/lib/";
stAIISPAttr.u32FrameBufCnt = 1;

s32Ret = RK_MPI_VPSS_SetGrpAIISPAttr(vpssGrpId, &stAIISPAttr);
if (RK_SUCCESS != s32Ret) {
    return s32Ret;
}

```

- 更新AI ISP 参数(伪代码)。

```

RK_S32 s32Ret = RK_FAILURE;
AIISP_ATTR_S stAIISPAttr;

/* after vpss vpssGrpId create */
memset(&stAIISPAttr, 0, sizeof(AIISP_ATTR_S));
stAIISPAttr.bEnable = RK_TRUE;
stAIISPAttr.stAilspCallback.pfUpdateCallback = aiisp_callback;
stAIISPAttr.stAilspCallback.pPrivateData = RK_NULL;
stAIISPAttr.pModelFilePath = "/oem/usr/lib/";
stAIISPAttr.u32FrameBufCnt = 1;

s32Ret = RK_MPI_VPSS_SetGrpAIISPAttr(vpssGrpId, &stAIISPAttr);
if (RK_SUCCESS != s32Ret) {
    return s32Ret;
}

stAIISPAttr.bEnable = RK_FALSE;
s32Ret = RK_MPI_VPSS_SetGrpAIISPAttr(vpssGrpId, &stAIISPAttr);
if (RK_SUCCESS != s32Ret) {
    return s32Ret;
}

stAIISPAttr.bEnable = RK_TRUE;
stAIISPAttr.stAilspCallback.pfUpdateCallback = aiisp_callback_0;
stAIISPAttr.stAilspCallback.pPrivateData = RK_NULL;
stAIISPAttr.pModelFilePath = "/usr/lib";
stAIISPAttr.u32FrameBufCnt = 2;

s32Ret = RK_MPI_VPSS_SetGrpAIISPAttr(vpssGrpId, &stAIISPAttr);
if (RK_SUCCESS != s32Ret) {
    return s32Ret;
}

```

2.38 RK_MPI_VPSS_GetGrpAIISPAttr

【描述】

获取 VPSS 的 AI ISP 参数。

【语法】

```
RK_S32 RK_MPI_VPSS_GetGrpAIISPAttr(VPSS_GRP VpssGrp, AIISP_ATTR_S *pstAilspAttr);
```

【参数】

参数名	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围: [0, VPSS_MAX_GRP_NUM)。	输入
pstAilspAttr	AIISP 参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VPSS错误码 。

【注意】

- 该接口仅RV1103/RV1106/RV1109/RV1126 平台支持。

3. 数据类型

VPSS 模块相关数据类型定义如下：

- [VPSS_MAX_GRP_NUM](#): 定义 VPSS GROUP 的最大个数。
- [VPSS_MAX_GRP_PIPE_NUM](#): 定义 VPSS GROUP 上最大的 PIPE 个数。
- [VPSS_MAX_CHN_NUM](#): 定义 VPSS 通道的最大个数。
- [VPSS_MIN_IMAGE_WIDTH](#): 定义 VPSS 图像的最小宽度。
- [VPSS_MIN_IMAGE_HEIGHT](#): 定义 VPSS 图像的最小高度。
- [VPSS_MAX_IMAGE_WIDTH](#): 定义 VPSS 图像的最大宽度。
- [VPSS_MAX_IMAGE_HEIGHT](#): 定义 VPSS 图像的最大高度。
- [VPSS_GRP](#): 定义 VPSS 组号。
- [VPSS_GRP_PIPE](#): 定义 VPSS 组的管道号。
- [VPSS_CHN](#): 定义 VPSS 通道号。
- [VPSS_CROP_COORDINATE_E](#): 定义 CROP 起点坐标的模式。
- [VPSS_CROP_INFO_S](#): 定义 CROP 功能所需信息。
- [VPSS_ROTATION_EX_ATTR_S](#): 定义 VPSS 的任意角度旋转属性。
- [VPSS_MOD_PARAM_S](#): 定义设置模块参数。
- [VPSS_GRP_ATTR_S](#): 定义 VPSS GROUP 属性。
- [VPSS_CHN_ATTR_S](#): 定义 VPSS 通道属性。
- [VPSS_CHN_MODE_E](#): 定义 VPSS CHN 工作模式。
- [ASPECT_RATIO_E](#): 定义 VPSS 帧型比工作模式。
- [ASPECT_RATIO_S](#): 定义 VPSS 输出图像幅形比的设置参数。
- [AIISP_CALLBACK](#): 定义 AI ISP 更新 AI NR 参数回调函数。
- [AIISP_CALLBACK_FUNC_S](#): 定义 AI ISP 属性。
- [AIISP_ATTR_S](#): 定义 AI ISP 属性。

3.1 VPSS_MAX_GRP_NUM

【说明】

定义 VPSS GROUP 的最大个数。

【定义】

```
#define VPSS_MAX_GRP_NUM      256
```

【注意事项】

- 芯片不同，可能有所差异，具体指参数值定义请看rkDefines.h。

3.2 VPSS_MAX_GRP_PIPE_NUM

【说明】

定义 VPSS GROUP 的最大个数。

【定义】

```
#define VPSS_MAX_GRP_PIPE_NUM    1
```

【注意事项】

- 只能设置为 0。

3.3 VPSS_MAX_CHN_NUM

【说明】

定义 VPSS 通道的最大个数。

【定义】

```
#define VPSS_MAX_CHN_NUM    4
```

【注意事项】

- 芯片不同，可能有所差异，具体指参数值定义请看rk_defines.h。

3.4 VPSS_MIN_IMAGE_WIDTH

【说明】

定义 VPSS 图像的最小宽度。

【定义】

```
#define VPSS_MIN_IMAGE_WIDTH    64
```

【注意事项】

- 芯片不同，可能有所差异，具体指参数值定义请看rk_defines.h。

3.5 VPSS_MIN_IMAGE_HEIGHT

【说明】

定义 VPSS 图像的最小高度。

【定义】

```
#define VPSS_MIN_IMAGE_HEIGHT    64
```

【注意事项】

- 芯片不同，可能有所差异，具体指参数值定义请看rk_defines.h。

3.6 VPSS_MAX_IMAGE_WIDTH

【说明】

定义 VPSS 图像的最大宽度。

【定义】

```
#define VPSS_MAX_IMAGE_WIDTH    8192
```

【注意事项】

- 芯片不同，可能有所差异，具体指参数值定义请看rk_defines.h。

3.7 VPSS_MAX_IMAGE_HEIGHT

【说明】

定义 VPSS 图像的最大高度。

【定义】

```
#define VPSS_MAX_IMAGE_HEIGHT 8192
```

【注意事项】

- 芯片不同，可能有所差异，具体指参数值定义请看rkDefines.h。

3.8 VPSS_GRP

【说明】

定义 VPSS 组号。

【定义】

```
typedef RK_S32 VPSS_GRP;
```

【注意事项】

无

3.9 VPSS_GRP_PIPE

【说明】

定义 VPSS 组的管道号。

【定义】

```
typedef RK_S32 VPSS_GRP_PIPE;
```

【注意事项】

- VPSS_GRP_PIPE 取值只能为 0。

3.10 VPSS_CHN

【说明】

定义 VPSS 通道号。

【定义】

```
typedef RK_S32 VPSS_CHN;
```

【注意事项】

无

3.11 VPSS_CROP_COORDINATE_E

【说明】

定义 CROP 起点坐标的模式。

【定义】

```
typedef enum rkVPSS_CROP_COORDINATE_E {
    VPSS_CROP_RATIO_COOR = 0,
    VPSS_CROP_ABS_COOR
} VPSS_CROP_COORDINATE_E;
```

【成员】

成员名称	描述
VPSS_CROP_RATIO_COOR	相对坐标。
VPSS_CROP_ABS_COOR	绝对坐标。

【注意事项】

- 相对坐标，即起始点的坐标值是以与当前图像宽高的比率来表示，使用时需做转换，具体请参见[VPSS_CROP_INFO_S](#)。

3.12 VPSS_CROP_INFO_S

【说明】

定义 CROP 功能所需信息。

【定义】

```
typedef struct rkVPSS_CROP_INFO_S {
    RK_BOOL          bEnable;
    VPSS_CROP_COORDINATE_E enCropCoordinate;
    RECT_S           stCropRect;
} VPSS_CROP_INFO_S;
```

【成员】

成员名称	描述
bEnable	CROP 使能开关。
enCropCoordinate	CROP 起始点坐标模式。
stCropRect	CROP 的矩形区域。

【注意事项】

- 若 enCropCoordinate 为 VPSS_CROP_RATIO_COOR（相对坐标模式），使用 stCropRect 的成员时应做转换，计算公式为：

s32X = 起始点坐标 x 原始图像宽度/1000，合法取值范围：[0, 999]，计算完成后会进行取整操作和对齐操作。公式同样适用于纵坐标计算。u32Width = 区域宽度 x 实际图像宽度/1000，区域宽度取值范围：[1, 1000]。计算完成后会进行取整操作和对齐操作。公式同样适用于区域高度计算。

- 坐标和宽高要求 2 像素对齐。

3.13 VPSS_ROTATION_EX_ATTR_S

【说明】

定义 VPSS 的任意角度旋转属性。

【定义】

```
typedef struct rkVPSS_ROTATION_EX_ATTR_S {
    RK_BOOL      bEnable;
    ROTATION_EX_S stRotationEx;
} VPSS_ROTATION_EX_ATTR_S;
```

【成员】

成员名称	描述
bEnable	Enable/Disable 任意角度旋转功能。
stRotationEx	任意角度旋转的详细属性。具体描述请参考“系统控制”章节。

【注意事项】

- RK356X/RK3588不支持任意角旋转。

3.14 VPSS_MOD_PARAM_S

【说明】

定义设置模块参数。

【定义】

```
typedef struct rkVPSS_MOD_PARAM_S {  
    MB_SOURCE_E enVpssMBSource;  
} VPSS_MOD_PARAM_S;
```

【成员】

成员名称	描述
enVpssMBSource	视频缓存池类型。 0：公共缓冲池。 1：模块缓冲池。保留，暂不支持。 2：私有缓冲池。 3：用户缓冲池。 默认为私有缓冲池。

【注意事项】

- 无

3.15 VPSS_GRP_ATTR_S

【说明】

定义 VPSS GROUP 属性。

【定义】

```
typedef struct rkVPSS_GRP_ATTR_S {  
    RK_U32          u32MaxW;  
    RK_U32          u32MaxH;  
    PIXEL_FORMAT_E   enPixelFormat;  
    DYNAMIC_RANGE_E  enDynamicRange;  
    FRAME_RATE_CTRL_S stFrameRate;  
    COMPRESS_MODE_E  enCompressMode;  
} VPSS_GRP_ATTR_S;
```

【成员】

成员名称	描述
u32MaxW	输入图像的最大宽度。静态属性，创建 Group 时设定，不可更改，暂未被使用。
u32MaxH	输入图像的最大高度。静态属性，创建 Group 时设定，不可更改，暂未被使用。
enPixelFormat	输入图像像素格式。静态属性，创建 Group 时设定，不可更改，暂未被使用。
enDynamicRange	输入图像动态范围。静态属性，创建 Group 时设定，不可更改，暂未被使用。
stFrameRate	组帧率控制。动态属性，立即生效。
enCompressMode	group取的图像的压缩方式。静态属性，创建 Group 时设定，不可更改。

【注意事项】

- 无效参数无需设置，不做异常参数检查。
- 若要开启帧率控制，源帧率必须大于0，目标帧率可以为0，此时会根据源帧率进行阻塞式的帧率控制，并将所有输入帧丢弃。

3.16 VPSS_CHN_ATTR_S

【说明】

定义 VPSS 通道的属性。

【定义】

```
typedef struct rkVPSS_CHN_ATTR_S {
    VPSS_CHN_MODE_E enChnMode;
    RK_U32          u32Width;
    RK_U32          u32Height;
    VIDEO_FORMAT_E  enVideoFormat;
    PIXEL_FORMAT_E   enPixelFormat;
    DYNAMIC_RANGE_E enDynamicRange;
    COMPRESS_MODE_E enCompressMode;
    FRAME_RATE_CTRL_S stFrameRate;
    RK_BOOL          bMirror;
    RK_BOOL          bFlip;
    RK_U32          u32Depth;
    ASPECT_RATIO_S  stAspectRatio;
    RK_U32          u32FrameBufCnt;
} VPSS_CHN_ATTR_S;
```

【成员】

成员名称	描述
enChnMode	通道工作模式。动态属性。
u32Width	目标图像宽度。动态属性。
u32Height	目标图像高度。动态属性。
enVideoFormat	目标图像视频格式。动态属性。
enPixelFormat	目标图像像素格式。动态属性。
enDynamicRange	目标图像动态范围。动态属性，暂未被使用。
enCompressMode	目标图像压缩模式。动态属性。
stFrameRate	帧率控制信息。动态属性，立即生效。
bMirror	水平镜像使能。动态属性。
bFlip	垂直翻转使能。动态属性。
u32Depth	用户获取通道图像的队列长度。 取值范围：[0, 8]。动态属性。
stAspectRatio	幅形比参数。动态属性。
u32FrameBufCnt	最大目标图像缓冲个数 仅在通道缓存模式为private模式时生效。 静态属性。

【注意事项】

- 源帧率与目标帧率都为-1，则不进行帧率控制。
- 若要开启帧率控制，源帧率必须大于0，目标帧率可以为0，此时会将所有通道待处理图像丢弃。
- u32Depth深度为0时，表示不保留通道图像，全部丢弃。
- RV1103/RV1106 平台上用户获取通道图像的优先级高于绑定。常见异常情况：u32FrameBufCnt 和 u32Depth 同时配置为1时，用户无获取通道输出图像的行为发生则该通道输出卡顿。
- bMirror 与 bFlip 可同时生效，并与通道旋转角度参数（固定旋转角度 或 任意旋转角度）共同决定通道图像的输出效果，具体详见：[RK_MPI_VPSS_SetChnRotation](#) 接口。
- 动态属性，如做不特殊说明，均延后一帧生效。
- u32FrameBufCnt 最大目标图像缓冲个数设置小于VPSS 最小目标图像缓存个数时，内部强制为VPSS 最小目标图像缓存个数，以保证VPSS输出满足最小可用缓存。

芯片平台	VPSS 最小目标图像缓存个数
RV1103/RV1106	1
RV1109/RV1126	4
RK356x	4
RK3588	4

- u32FrameBufCnt 最大目标图像缓冲个数仅在通道缓存模式为private模式 (MB_SOURCE_PRIVATE) 时生效，具体详见：[RK_MPI_VPSS_SetModParam](#) 接口。

【相关数据类型及接口】

- [RK_MPI_VPSS_SetChnRotation](#)。
- [RK_MPI_VPSS_SetModParam](#)。
- [RK_MPI_VPSS_SetChnRotation](#)。

3.17 VPSS_CHN_MODE_E

【说明】

定义 VPSS CHN 工作模式。

【定义】

```
typedef enum rkVPSS_CHN_MODE_E {  
    VPSS_CHN_MODE_USER = 0,  
    VPSS_CHN_MODE_AUTO = 1,  
    VPSS_CHN_MODE_PASSTHROUGH = 2  
} VPSS_CHN_MODE_E;
```

【成员】

成员名称	描述
VPSS_CHN_MODE_USER	用户设置模式。
VPSS_CHN_MODE_AUTO	自动模式。（弃用模式，不推荐使用）
VPSS_CHN_MODE_PASSTHROUGH	穿透模式。使用该模式时，VPSS硬件本身不处理数据，将传递解析数据至下级处理。

【注意事项】

- 通道工作模式推荐使用USER模式和PAST模式，AUTO模式弃用，工作模式的工作流程详见[RK356X/RK3588 VPSS USER 模式数据流图](#)和[RK356X/RK3588 VPSS PAST模式数据流图](#)。
- 通道模式为PAST时，需要与VO绑定才可发挥模式作用。

3.18 ASPECT_RATIO_E

【说明】

定义幅型比工作模式。

【定义】

```
typedef enum rkASPECT_RATIO_E {  
    ASPECT_RATIO_NONE = 0,  
    ASPECT_RATIO_AUTO = 1,  
    ASPECT_RATIO_MANUAL = 2,  
    ASPECT_RATIO_BUTT  
} ASPECT_RATIO_E;
```

【成员】

成员名称	描述
ASPECT_RATIO_NONE	不开启幅型比功能。
ASPECT_RATIO_AUTO	开启幅形比自动适配功能，输出图像保持和输入图像宽高比一致。
ASPECT_RATIO_MANUAL	手动调整输出画面在显示区域中的位置。

【注意事项】

详见[ASPECT_RATIO_S](#)。

【相关数据类型及接口】

[ASPECT_RATIO_S](#)

3.19 ASPECT_RATIO_S

【说明】

定义 VPSS 输出图像幅形比的设置参数。

【定义】

```
typedef struct rkASPECT_RATIO_S {  
    ASPECT_RATIO_E enMode;  
    RK_U32      u32BgColor;  
    RECT_S     stVideoRect;  
} ASPECT_RATIO_S;
```

【成员】

成员名称	描述
enMode	设置幅形比的类型。
u32BgColor	开启幅形比后，非视频区域的填充颜色（暂不支持该参数）。
stVideoRect	ASPECT_RATIO_MANUAL 模式下，视频区域在显示区域的位置。 MANUAL 模式下，坐标值必须满足 x: 大于等于 0，小于通道图像宽； y: 大于等于 0，小于通道图像高； w: 大于等于 64，小于等于通道图像宽度； h: 大于等于 64，小于等于通道图像高度，并且 x+w 不大于通道图像宽，y+h 不大于通道图像高。

【注意事项】

- 如果输出图像存在缩放，则图像会缩放至幅形比视频区域的大小。
- 幅形比参数在通道属性中设置。只有在 VPSS 通道设置为 USER 模式时才生效，PAST 模式下幅形比参数由 VO 设置。
- 设定为 ASPECT_RATIO_MANUAL 模式时，用户手动设置图像的起始坐标和宽高，VPSS 会根据用户的设置来输出图像并添加黑边。图像显示的区域不允许超过输出图像的宽高。
- 设定为 ASPECT_RATIO_AUTO 模式时，VPSS 会保持输出画面的宽高比和输入宽高比一致，自动在画面上下或者左右加上黑边。

3.20 AIISP_CALLBACK

【说明】

AI ISP 更新 AI NR 参数回调函数。

【需求】

- 头文件：rk_comm_aiisp.h

【定义】

```
/** AIISP update AINR params callback function */  
typedef RK_S32 (*AIISP_CALLBACK)(RK_VOID *pAinrParam, RK_VOID *pPrivateData);
```

【注意事项】

无。

3.21 AIISP_CALLBACK_FUNC_S

【说明】

定义 AI NR 参数更新回调函数结构体。

【定义】

```

typedef struct AIISP_CALLBACK_FUNC_S {
    AIISP_CALLBACK    pfUpdateCallback;
    RK_VOID        *pPrivateData; /* Not yet used */
} AIISP_CALLBACK_FUNC_S;

```

【成员】

成员名称	描述
pfUpdateCallback	回调函数。
pPrivateData	回调函数私有变量。

【注意事项】

- pPrivateData 暂未使用。

3.22 AIISP_ATTR_S

【说明】

定义 AI ISP 属性。

【需求】

- 头文件：rk_comm_aiisp.h
- 库文件：librkpostisp.so、librkMotionDetect.so
- 模型文件：*.aiisp (如：rkmodel_1080_1920.aiisp)

【定义】

```

typedef struct rkAIISP_ATTR_S {
    RK_BOOL      bEnable; /* Whether AIISP is enable */
    AIISP_CALLBACK_FUNC_S stAilspCallback; /* AIISP callback function */
    const RK_CHAR   *pModelFilePath; /* AIISP model file path */
    RK_U32       u32FrameBufCnt; /* RW; frame buffer cnt */
} AIISP_ATTR_S;

```

【成员】

成员名称	描述
bEnable	AI ISP 功能使能开关。 动态属性。
stAilspCallback	AI NR 参数更新回调函数结构体。
pModelFilePath	AI ISP 模型路径。
u32FrameBufCnt	最大目标图像缓冲个数。

【注意事项】

- 无特殊说明均为静态属性。
- bEnable 不可重复配置为 RKTURE, 否则 [RK_MPI_VPSS_SetGrpAIISPAttr](#) 接口报错。
- pModelFilePath 为模型所在的目录，将由AI ISP 根据输入图像尺寸自动加载适合的模型；也可以为空字符串，模型路径将由系统环境变量指定。
- u32FrameBufCnt 为 0 时，内部强制设置为 1，以保证AI ISP 输出满足最小可用缓存。

4. VPSS错误码

视频处理子系统 API VPSS 错误码如下所示：

错误代码	宏定义	描述
0xA0068001	RK_ERR_VPSS_INVALID_DEVID	VPSS GROUP 号无效
0xA0068002	RK_ERR_VPSS_INVALID_CHNID	VPSS 通道号无效
0xA0068003	RK_ERR_VPSS_ILLEGAL_PARAM	VPSS 参数设置无效
0xA0068004	RK_ERR_VPSS_EXIST	VPSS GROUP 已创建
0xA0068005	RK_ERR_VPSS_UNEXIST	VPSS GROUP 未创建
0xA0068006	RK_ERR_VPSS_NULL_PTR	输入参数空指针错误
0xA0068008	RK_ERR_VPSS_NOT_SUPPORT	操作不支持
0xA0068009	RK_ERR_VPSS_NOT_PERM	操作不允许
0xA006800C	RK_ERR_VPSS_NOMEM	分配内存失败
0xA006800D	RK_ERR_VPSS_NOBUF	分配 BUF 池失败
0xA006800E	RK_ERR_VPSS_BUF_EMPTY	图像队列为空
0xA006800F	RK_ERR_VPSS_BUF_FULL	图像队列已满
0xA0068010	RK_ERR_VPSS_NOTREADY	VPSS 系统未初始化
0xA0068012	RK_ERR_VPSS_BUSY	VPSS 系统忙
0xA0068013	RK_ERR_VPSS_SIZE_NOT_ENOUGH	MB 块大小不够

视频解码

[概述](#)

[重要概念](#)

[模块参数](#)

[API 参考](#)

[数据类型](#)

[VDEC 错误码](#)

1. 概述

VDEC模块是提供视频硬件解码的接口，不同芯片平台支持的解码协议、分辨率、性能等有所差别。

各芯片的VDEC特性如表1-1所示：

芯片	硬解模块	最大通道	支持协议	支持分辨率	最大性能
RK3568	RKVDEC VDPU JPEGD	64	RKVDEC: H.264/H.265 VDPU: MPEG2/MPEG4 JPEGD: MJPEG/JPEG	RKVDEC: - H.264: 16x16 to 4096x2304 - H.265: 64x64 to 4096x2304 VDPU: - MPEG2: 48x48 to 1920x1088 - MPEG4: 48x48 to 1920x1088 JPEGD: - MJPEG: 48x48 to 65536x65536 - JPEG: 48x48 to 65536x65536	H264: 4096x2304@30fps H265: 4096x2304@60fps MPEG2/MPEG4: 1920x1088@60fps MPEG4: 1920x1088@60fps MJPEG/JPEG: 1920x1080@120fps
RK3588	RKVDEC VDPU JPEGD	256	RKVDEC: H.264/H.265 VDPU: MPEG2/MPEG4 JPEGD: MJPEG/JPEG	RKVDEC: - H.264: 16x16 to 65520x65520 - H.265: 64x64 to 65472x65472 VDPU: - MPEG2: 48x48 to 1920x1088 - MPEG4: 48x48 to 1920x1088 JPEGD: - MJPEG: 48x48 to 65536x65536 - JPEG: 48x48 to 65536x65536	H264: 7680x4320@30fps H265: 7680x4320@60fps MPEG2/MPEG4: 1920x1088@60fps MPEG4: 1920x1088@60fps MJPEG/JPEG: 1920x1080@280fps

2. 重要概念

• 码流发送模式

解码器码流发送模式包括以下两种：

- 流式发送 (VIDEO_MODE_STREAM)

用户每次可发送任意长度码流到解码器，由解码器内部完成码流分帧，解码器需要在收到下一帧码流才能识别当前帧码流的结束，因此在该发送模式下，无法立即输出当前帧解码图像。

- 按帧发送 (VIDEO_MODE_FRAME)

用户每次发送完整一帧码流到解码器，解码器认为该帧码流包含一帧完整的图像码流，内部不再做分帧，开始解码图像，因此需保证每次调用发送接口发送的码流必须为一帧，否则会出现解码错误。通过该发送方式可以达到快速解码的目的。

码流发送模式配置方法：

码流发送模式可通过接口 [RK_MPI_VDEC_CreateChn](#) 配置置通道属性 [VDEC_CHN_ATTR_S](#) 》 enMode 来配置。

• 码流包创建方式

通过码流发送接口发送码流时，承载码流数据的数据类型统一为 MB_BLK 类型的内存块，该内存块的创建方式包括：

- 基于用户已有的虚拟内存构建

通过接口 [RK_MPI_SYS_CreateMB](#) 创建内存块，需配置 [MB_EXT_CONFIG_S](#) 参数。若外部码流在读取或解析后，已经存放于一块虚拟内存，可直接基于该虚拟内存地址构建一块MB内存块，内部仅做数据结构类型封装，不做拷贝。若该虚拟内存外部需要循环使用时，则不需要配置释放回调方法 pFreeCB，并在发送码流时以通过拷贝模式发送。若该虚拟内存是动态分配的，可通过配置释放回调方法，交由解码器管理，并在使用完成后调用回调释放该虚拟内存。

- 基于用户已有的DMA-BUF构建

若用于已通过其他方式创建了 DMA Buffer，并将码流存放在该 Buffer 内，可通过基于虚拟内存相同的构建方法，创

建 MB 内存块，每个 DMA Buffer 均有对应的访问句柄FD，在配置 [MB_EXT_CONFIG_S](#) 参数时需填写对应的句柄 FD。

- 从内存池申请

通过接口 [RK_MPI_MB_CreatePool](#) 先创建内存池，然后通过 [RK_MPI_MB_GetMB](#) 获取一个内存块，在使用完成后调用 [RK_MPI_MB_ReleaseMB](#) 释放回内存池。采用此方式时，建议在送码流时，采用直通模式发送。

- 直接申请

直接调用 [SYS](#) 模块提供的接口申请MB内存块，然后将码流数据拷贝到申请的内存块。通过接口 [RK_MPI_SYS_Malloc](#) 申请虚拟内存，通过接口 [RK_MPI_SYS_MmzAlloc](#) 申请物理内存。采用此方式时，建议在送码流时，采用直通模式发送。

- **解码器码流接收模式**

用户通过码流发送接口发送码流给解码器时，解码器接收码流方式包括以下两种：

- 拷贝模式

通过码流发送接口送入的码流数据会在解码器内部做拷贝，解码器内外部码流 Buffer 已无关联，用户可根据需要自行管理 [VDEC_STREAM_S](#)》 pMbBlk 相关的内存块。

- 直通模式

通过码流发送接口送入的码流数据会在解码器内部不做拷贝，只是对 [VDEC_STREAM_S](#)》 pMbBlk 增加一次引用，在处理完成后释放该引用。若该 pMbBlk 是通过 [RK_MPI_SYS_CreateMB](#) 构建的外部内存块，在释放 pMbBlk 时，会同时调用用户构建 pMbBlk 时设置的释放回调 pFreeCB 释放相关内存。若该 pMbBlk 是通过 [RK_MPI_MB_GetMB](#) 构建的内存块，则会释放回 MB 内存池，循环使用。

码流接收模式配置方法：

通过接口 [RK_MPI_VDEC_SendStream](#) 发送码流，并配置码流参数 [VDEC_STREAM_S](#)》 bBypassMbBlk，配置为 RK_FALSE 时开启拷贝模式，配置为 RK_TRUE 时开启直通模式。以上两种方式在调用 [RK_MPI_VDEC_SendStream](#) 之后，都需要调用 [RK_MPI_MB_ReleaseMB](#) 来释放，否则会出现内存泄漏。

- **解码帧存分配方式**

解码帧存是存放解码后的图像数据内存，[VDEC](#) 模块支持的解码内存池分配方式包括：

- 私有池 (MB_SOURCE_PRIVATE)

私有池是指由 VDEC 模块内部按通道创建并被该通道独占的 MB 内存池，可从该内存池申请 MB 作为该通道的图像 Buffer。用户可以在创建通道接口 [RK_MPI_VDEC_CreateChn](#) 中设置私有 MB 内存池的个数 u32FrameBufCnt 及单个 MB 内存块大小 u32FrameBufSize，MB 块的大小计算较为复杂，建议将参数 u32FrameBufSize 配置为 0，由内部根据配置的宽高计算所需的图像 Buffer 大小。

- 用户池 (MB_SOURCE_USER)

用户池是指在创建解码通道时不创建内存池也不分配解码图像 Buffer，而是由用户调用接口 [RK_MPI_MB_CreatePool](#) 创建一个 MB 内存池，再通过调用接口 [RK_MPI_VDEC_AttachMbPool](#) 把该 MB 内存池绑定到某个解码通道。

- 模块内存池 (MB_SOURCE_MODULE)

模块内存池是指在创建解码通道时从公共内存池获取一个符合大小和个数要求的内存池，若不存在，则会重新创建一个内存池，获取的内存池也是该解码通道独占。在通道销毁后，将该内存池还回公共内存池，若此时未配置内存池驻留大小，该内存池将被销毁。用户可通过配置驻留内存大小，来减少内存的频繁分配释放，也可通过配置总的内存大小来限定总内存使用，如果未配置，则内部会按需分配内存供解码通道使用。

- 通过设置环境变量来限制使用内存总大小： export rt_mb_max_alloc=209715200 (限制200M，超过200M将分配不到解码 bufer)，注意总内存的大小包含非解码部分的内存，在计算阈值时需要考虑非解码内存大小。

- 通过设置环境变量来限制驻留内存总大小： export rt_mb_max_retain=104857600 (限制100M，最多驻留100M为下次解码使用)

解码内存池分配方式配置方法：

内存池分配方式可通过接口 [RK_MPI_VDEC_SetModParam](#) 配置模块参数 [VDEC_MOD_PARAM_S](#)》 enVdecMBSource 来配置，如未配置，默认为 MB_SOURCE_MODULE 模式。

- **解码图像输出方式**

解码图像输出方式包括以下两种：

- 解码序 (VIDEO_OUTPUT_ORDER_DEC)

解码图像按照解码输入包的先后顺序解码输出，解码后的图像能够立即快速输出，解码图像显示输出的先后顺序与解码顺序一致。当码流不存在 B 帧的情况下建议使用此输出方式，可实现快速解码出帧，且能减少解码所需内存块数。

- 显示序 (VIDEO_OUTPUT_ORDER_DISP)

解码图像在解码后按照各协议语法要求排序后输出，解码后的图像可能不会在解码后立即输出，解码图像显示输出的先后顺序与解码顺序可能不一致。比如当码流存在 B 帧且需要参考前后的 P 帧(原始帧顺序 IPBP)，解码该 B 帧前，

需先解码B帧后的P帧，但P帧不会立即输出，需要在B帧解码后才能输出，此时显示序(IPBP)与解码序(IPPB)不一致。

解码图像输出方式配置方法：

图像输出方式可通过接口 [RK_MPI_VDEC_SetChnParam](#) 配置通道参数 [VDEC_CHN_PARAM_S](#)》

[VDEC_PARAM_VIDEO_S](#)》enOutputOrder 来配置，JPEG/MJPEG码流不支持该配置。

• 显示时间戳处理

按帧模式 (VIDEO_MODE_FRAME) 发送码流时，解码输出的图像显示时间戳 PTS 为发送码流接口 (RK_MPI_VDEC_SendStream) 中用户送入的 PTS，解码器不会更改此值，但可能会更改输出顺序，比如码流存在B帧时，输出图像 PTS 与输入的码流 PTS 不一致。如果用户送入的 PTS 值为 -1，则表示此图像不会被视频输出模块 (VO) 显示；若用户配置的 PTS 值为 0，则表示用户不进行帧率控制，而是由视频输出模块 VO 进行帧率控制；按流式模式 (VIDEO_MODE_STREAM) 发送码流时，解码输出图像的 PTS 应统一设为 0，用户不进行帧率控制，而是由视频输出模块 VO 进行帧率控制。

• 用户图片插入支持

用户可直接向 VDEC 模块插入图片数据，VDEC 模块不对该图片做任何处理，在绑定模式下，直接送往下一级。在码流源端出现某种异常时，用户可通过该方法插入图片提示。比如当网络异常断开，前端没有再继续送码流，用户可通过设置插入图片，并输出显示到 VO 模块，以提示当前网络异常或没有码流可解码。VDEC 模块插入用户图片方式包括：

- 立即插入图片

VDEC模块会先清空解码器内部的码流和图像，然后插入用户图片。

- 延迟插入图片

VDEC模块会先将解码器内部的码流全部解码输出后，然后再插入用户图片。

用户图片插入方法：

先通过接口 [RK_MPI_VDEC_SetUserPic](#) 设置需要插入的用户图片，再通过接口 [RK_MPI_VDEC_EnableUserPic](#)

使能插入的用户图片，设置该接口参数 bInstant 为 RK_TRUE 时，则会立即插入并输出到后级模块；设置为 RK_FALSE 时，则延迟插入图片，但通道内剩余数据解码完成后再插入用户图片。

3. 模块参数

• VDEC_MAX_CHN_NUM

VDEC 模块支持的最大通道数。目前支持的通道数是固定的，后续将增加配置文件让用户可根据业务场景需求配置一个合理的大解码通道数，以减少通道上下文相关的内存占用。

4. API 参考

该功能模块为用户提供以下 API：

- [RK_MPI_VDEC_SetModParam](#)：设置解码模块参数。
- [RK_MPI_VDEC_GetModParam](#)：获取解码模块参数。
- [RK_MPI_VDEC_CreateChn](#)：创建视频解码通道。
- [RK_MPI_VDEC_DestroyChn](#)：销毁视频解码通道。
- [RK_MPI_VDEC_ResetChn](#)：复位解码通道。
- [RK_MPI_VDEC_GetChnAttr](#)：获取视频解码通道属性。
- [RK_MPI_VDEC_SetChnAttr](#)：设置视频解码通道属性。
- [RK_MPI_VDEC_StartRecvStream](#)：解码器开始接收用户发送的码流。
- [RK_MPI_VDEC_StopRecvStream](#)：解码器停止接收用户发送的码流。
- [RK_MPI_VDEC_QueryStatus](#)：查询解码通道状态。
- [RK_MPI_VDEC_SendStream](#)：向视频解码通道发送码流数据。
- [RK_MPI_VDEC_GetFrame](#)：获取视频解码通道的解码图像。
- [RK_MPI_VDEC_ReleaseFrame](#)：释放视频解码通道的解码图像。
- [RK_MPI_VDEC_SetChnParam](#)：设置解码通道参数。
- [RK_MPI_VDEC_GetChnParam](#)：获取解码通道参数。
- [RK_MPI_VDEC_SetDisplayMode](#)：设置显示模式。
- [RK_MPI_VDEC_GetDisplayMode](#)：获取显示模式。
- [RK_MPI_VDEC_GetFd](#)：获取视频解码通道的设备文件句柄。

- [RK_MPI_VDEC_CloseFd](#): 关闭视频解码的设备文件句柄。
- [RK_MPI_VDEC_AttachMbPool](#): 将解码通道绑定到某个视频缓存 MB 池中。
- [RK_MPI_VDEC_DetachMbPool](#): 将解码通道从某个视频缓存 MB 池中解绑定。
- [RK_MPI_VDEC_SetUserPic](#): 设置用户图片属性。
- [RK_MPI_VDEC_EnableUserPic](#): 使能插入用户图片。
- [RK_MPI_VDEC_DisableUserPic](#): 禁止使能插入用户图片。

4.1 RK_MPI_VDEC_SetModParam

【描述】

设置解码通道参数。

【语法】

```
RK_S32 RK_MPI_VDEC_SetModParam(const VDEC\_MOD\_PARAM\_S*pstModParam)
```

【参数】

参数名	描述	输入/输出
pstModParam	模块参数结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VDEC错误码 。

【注意】

- 此接口必须在所有解码通道创建前调用, 否则不生效。
- 传入的pstModParam不能为空, 否则返回空指针错误RK_ERR_VDEC_NULL_PTR。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_GetModParam](#)。

4.2 RK_MPI_VDEC_GetModParam

【描述】

获取解码通道参数

【语法】

```
RK_S32 RK_MPI_VDEC_GetModParam(VDEC\_MOD\_PARAM\_S*pstModParam)
```

【参数】

参数名	描述	输入/输出
pstModParam	模块参数结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VDEC错误码 。

【注意】

传入的pstModParam不能为空，否则返回空指针错误RK_ERR_VDEC_NULL_PTR。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_SetModParam](#)。

4.3 RK_MPI_VDEC_CreateChn

【描述】

创建视频解码通道。

【语法】

RK_S32 RK_MPI_VDEC_CreateChn(VDEC_CHN VdChn, const [VDEC_CHN_ATTR_S](#) *pstAttr)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围：[0, VDEC_MAX_CHN_NUM)。	输入
pstAttr	解码通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VDEC错误码 。

【注意】

- 通道号必须合法，不能超出最大通道号 VDEC_MAX_CHN_NUM。
- 参数pstAttr不能为空，否则将返回空指针错误 RK_ERR_VDEC_NULL_PTR。
- 在创建视频解码通道之前必须保证通道未创建（或者已经销毁），否则将返回通道已存在错误 RK_ERR_VDEC_EXIST。
- 通道属性 pstAttr 中的宽高必须大于0，否则将返回非法参数错误 RK_ERR_VDEC_ILLEGAL_PARAM。
- 通道属性 pstAttr 中的宽高不能超硬件解码能力限制，否则解码通道会初始化失败。
- 解码通道内存池分配方式只支持使用私有池和用户池方式，使用方法参考重要概念章节中的解码帧存分配方式。
- 解码帧存分配方式使用的是私有池方式，则用户需要根据解码码流配置解码所需的内存块个数 u32FrameBufCnt 及内存块大小 u32FrameBufSize，若配置为0，则会使用VDEC模块内部的默认参数。建议 u32FrameBufCnt 根据码流情况以及系统内存情况配置，建议 u32FrameBufSize 配置为0，由 VDEC 模块内部根据配置的宽高计算所使用的 Buffer 大小，根据个数和大小创建相应的MB私有池。
- 解码帧存分配方式使用的是用户池方式，则通道属性参数 u32FrameBufSize、u32FrameBufCnt 无效。由用户通过 rk_mpi_mb.h 接口申请内存，然后绑定到解码通道。用户创建的内存池需要保证 MB 块的大小和个数满足当前解码通道所需图像 Buffer 的大小和个数。不同协议解码所需的图像 MB 内存块个数和大小不同，其中H264、H265、MPEG2、MPEG4 每个通道解码所需 MB 内存块个数至少为参考帧数+2，JPEG/MJPEG 解码每个解码通道所需 VB 个数至少为2 个。MB 内存块计算大小较复杂，具体计算方法可参见 rk_mpi_cal.h 中的函数RK_MPI_CAL_VDEC_GetPicBufferSize。
- 解码的 H.264 码流有 B 帧，或者解码的 H.265 码流支持时域运动矢量预测（**sps_temporal_mvp_enabled_flag = 1**），则创建通道时需要设置此通道支持时域运动矢量预测（bTemporalMvpEnable 设置为 1），VDEC 模块内部将按照参考帧数量分配运动矢量 MV 存储所需的内存空间。反之，可关闭时域运动矢量预测（bTemporalMvpEnable 设置为 0），可节省内存分配。若无法同时满足各协议限制条件，可根据协议分别判断并进行设置开关时域运动矢量预测。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_DestroyChn](#)。

4.4 RK_MPI_VDEC_DestroyChn

【描述】

销毁视频解码通道。

【语法】

```
RK_S32 RK_MPI_VDEC_DestroyChn(VDEC_CHN VdChn)
```

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围：[0, VDEC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VDEC错误码 。

【注意】

- 通道必须已创建，否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_CreateChn](#)。

4.5 RK_MPI_VDEC_ResetChn

【描述】

复位视频解码通道，将清除解码通道缓存数据。

【语法】

```
RK_S32 RK_MPI_VDEC_ResetChn(VDEC_CHN VdChn)
```

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围：[0, VDEC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VDEC错误码 。

【注意】

- 通道必须已创建，否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。

【举例】

无。

【相关主题】

无。

4.6 RK_MPI_VDEC_GetChnAttr

【描述】

获取视频解码通道属性

【语法】

```
RK_S32 RK_MPI_VDEC_GetChnAttr(VDEC_CHN VdChn, VDEC\_CHN\_ATTR\_S *pstAttr)
```

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围：[0, VDEC_MAX_CHN_NUM)。	输入
pstAttr	解码通道属性指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VDEC错误码 。

【注意】

- 通道必须已创建，否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 传入的pstAttr不能为空，否则会返回空指针错误 RK_ERR_VDEC_NULL_PTR。

【举例】

无。

【相关主题】

无。

4.7 RK_MPI_VDEC_SetChnAttr

【描述】

设置视频解码通道属性

【语法】

```
RK_S32 RK_MPI_VDEC_SetChnAttr(VDEC_CHN VdChn, const VDEC\_CHN\_ATTR\_S *pstAttr)
```

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围：[0, VDEC_MAX_CHN_NUM)。	输入
pstAttr	解码通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VDEC错误码 。

【注意】

- 通道必须已创建，否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。

- 必须在调用 [RK_MPI_VDEC_StartRecvStream](#) 之前设置，不支持动态切换属性。
- 切换通道属性之前必须先停止接收码流，再次启动接收码流方可生效。
- 解码器改变属性之后，再次启动接收码流会自动复位解码通道。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_StopRecvStream](#)。

4.8 RK_MPI_VDEC_StartRecvStream

【描述】

解码器开始接收用户发送的码流。

【语法】

RK_S32 RK_MPI_VDEC_StartRecvStream(VDEC_CHN VdChn)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围：[0, VDEC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VDEC错误码 。

【注意】

- 通道必须已创建，否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 启动接收码流之后，才能调用 [RK_MPI_VDEC_SendStream](#) 发送码流成功。
- 重复调用启动接收码流接口时，返回成功。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_StopRecvStream](#)。

4.9 RK_MPI_VDEC_StopRecvStream

【描述】

解码器停止接收用户发送的码流。

【语法】

RK_S32 RK_MPI_VDEC_StopRecvStream(VDEC_CHN VdChn)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围：[0, VDEC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VDEC错误码 。

【注意】

- 通道必须已创建, 否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 调用此接口后, 调用发送码流接口 RK_MPI_VDEC_SendStream 会返回失败。
- 重复调用停止接收码流接口时, 返回成功。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_StartRecvStream](#)。

4.10 RK_MPI_VDEC_QueryStatus

【描述】

查询解码通道状态。

【语法】

RK_S32 RK_MPI_VDEC_QueryStatus(VDEC_CHN VdChn, [VDEC_CHN_STATUS_S](#) *pstStatus)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围: [0, VDEC_MAX_CHN_NUM)。	输入
pstStatus	视频解码通道状态结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VDEC错误码 。

【注意】

- 通道必须已创建, 否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 传入的pstStatus不能为空, 否则返回空指针错误RK_ERR_VDEC_NULL_PTR。

【举例】

无。

【相关主题】

无。

4.11 RK_MPI_VDEC_SendStream

【描述】

解码器停止接收用户发送的码流。

【语法】

RK_S32 RK_MPI_VDEC_SendStream(VDEC_CHN VdChn, const [VDEC_STREAM_S](#) *pstStream, RK_S32 s32MilliSec)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围: [0, VDEC_MAX_CHN_NUM)。	输入
pstStream	解码码流数据指针。	输入
s32MilliSec	送码流方式标志。 取值范围: -1: 阻塞。 0: 非阻塞。 正值: 超时时间, 以 ms 为单位, 没有上限值, 可动态设置。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VDEC错误码 。

【注意】

- 通道必须已创建, 否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 发送码流前必须保证已调用 RK_MPI_VDEC_StartRecvStream 接口启动接收码流, 否则返回该操作不允许错误 RK_ERR_VDEC_NOT_PERM。
- 发送码流时要按照创建解码通道时设置的发送方式进行发送。按帧发送时, 必须一次性发送完整的一帧码流, 否则解码会出现错误。按流式发送则无此限制。
- 当码流发送结束后, 可随最后一个码流包带上 bEndOfStream 为 1, 或者发送 bEndOfStream 为 1 的空码流包 (pMbBlk 置为 MB_INVALID_HANDLE 或 u32Len 置为 0), 解码器会把所有码流全部解完并输出全部图像。其它情况应把 bEndOfStream 置为 0。
- 不允许发送 bEndOfStream 为 0 的空码流包, 否则返回操作不允许错误 RK_ERR_VDEC_NOT_PERM。
- 按帧模式发送码流时, 解码图像的时间戳 PTS 等于传入参数 pstStream 结构体中的时间戳 PTS; 按流发送时, 解码图像的时间戳等于 0。
- 通过设置 s32MilliSec 值可支持阻塞方式、非阻塞方式、超时方式发送码流。
- 以阻塞方式发送码流, 则会永久阻塞直至码流发送成功。
- 以非阻塞方式发送码流, 如果码流缓冲区已满, 则返回错误 RK_ERR_VDEC_BUF_FULL。可通过调整通道属性参数 u32StreamBufCnt 来改变缓冲区存储的码流包个数。
- 以超时方式发送码流, 到达设定的超时时间还不能成功发送码流, 则返回错误 RK_ERR_VDEC_BUF_FULL。用户在收到错误 RK_ERR_VDEC_BUF_FULL 时, 应重送该码流包, 否则将出现解码错误。为避免出现 RK_ERR_VDEC_BUF_FULL 错误, 建议将超时时间设置为通道帧间隔的4倍, 如通道输出帧率为25fps, 应设置超时时间为160ms以上。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_StartRecvStream](#)。

4.12 RK_MPI_VDEC_GetFrame

【描述】

获取视频解码通道的解码图像。

【语法】

RK_S32 RK_MPI_VDEC_GetFrame(VDEC_CHN VdChn, [VIDEO_FRAME_INFO_S](#) *pstFrameInfo, RK_S32 s32MilliSec)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围: [0, VDEC_MAX_CHN_NUM)。	输入
pstFrameInfo	获取的解码图像信息。	输出
s32MilliSec	获取图像方式标志。 取值范围: -1: 阻塞。 0: 非阻塞。 正值: 超时时间, 以ms为单位, 没有上限值, 可动态设置。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VDEC错误码 。

【注意】

- 通道必须已创建, 否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 传入的pstFrameInfo不能为空, 否则会返回空指针错误 RK_ERR_VDEC_NULL_PTR。
- 通过 RK_MPI_VDEC_GetFrame 获取解码图像数据后, 需通过 RK_MPI_VDEC_ReleaseFrame 来释放, 否则会导致解码过程阻塞等待资源。
- 通过设置 s32MilliSec 值支持阻塞方式、非阻塞方式、超时方式获取解码图像。
- 以阻塞方式发送码流, 则会永久阻塞直至成功获取到解码图像。
- 以非阻塞方式获取解码图像, 如果缓冲区内无图像, 则返回错误 RK_ERR_VDEC_BUF_EMPTY。
- 以超时方式获取解码图像, 到达设定的超时时间还不能获取到图像则会返回错误 RK_ERR_VDEC_BUF_EMPTY。超时时间建议设置为10ms的倍数, 并大于解码输出帧率间隔, 如通道输出帧率为25fps, 则超时时间应设置为50ms。
- 通过 RK_MPI_VDEC_GetFd 获取通道句柄fd, 通过 select 方式查询缓冲区是否有图像, 然后再调用此接口获取解码图像。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_ReleaseFrame](#)。

[RK_MPI_VDEC_GetFd](#)。

4.13 RK_MPI_VDEC_ReleaseFrame

【描述】

释放视频解码通道的图像。

【语法】

RK_S32 RK_MPI_VDEC_ReleaseFrame(VDEC_CHN VdChn, const [VIDEO_FRAME_INFO_S](#) *pstFrameInfo)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围: [0, VDEC_MAX_CHN_NUM)。	输入
pstFrameInfo	解码后的图像信息指针, 由 RK_MPI_VDEC_GetFrame 接口获取。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VDEC错误码 。

【注意】

- 通道必须已创建, 否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 必须与 RK_MPI_VDEC_GetFrame 配对使用, 获取的数据应当在使用完之后立即释放, 否则会导致解码过程阻塞等待资源。
- 允许通道销毁后, 再释放获取的通道图像。
- 允许用户非顺序释放, 即不按照获取图像的顺序释放图像。
- 释放的数据必须是通过接口 RK_MPI_VDEC_GetFrame 从该通道获取的数据, 不得对数据信息结构体做任何修改。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_GetFrame](#)。

4.14 RK_MPI_VDEC_SetUserPic

【描述】

设置用户图片属性。

【语法】

RK_S32 RK_MPI_VDEC_SetUserPic(VDEC_CHN VdChn, const [VIDEO_FRAME_INFO_S](#) *pstUsrPic)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围: [0, VDEC_MAX_CHN_NUM)。	输入
pstUsrPic	用户图片属性结构指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VDEC错误码 。

【注意】

- 通道必须已创建, 否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 传入的 pstUsrPic 不能为空, 否则会返回空指针错误 RK_ERR_VDEC_NULL_PTR。
- 用户可直接通过 SYS 接口 RK_MPI_SYS_MmzAlloc 申请一块 MMZ 内存, 也可以通过 RK_MPI_MB_CreatePool 接口创建一个私有 MB 池, 从这个私有 MB 池里获取一块 MB 内存块来存放用户图片。
- 用户图片像素格式 enPixelFormat 应为后级模块能支持的格式, 如 NV12 / RGB888 等, 用户图片的 PTS 设置为 0。
- 在用户图片已使能的情况下, 不能再设置用户图片, 必须先调用接口 RK_MPI_VDEC_DisableUserPic 禁止使能用户图片, 否则会返回操作不允许错误 RK_ERR_VDEC_NOT_PERM。
- 用户图片属性设置成功之后不会生效, 需要调用 RK_MPI_VDEC_EnableUserPic 生效。
- 解码通道不会对用户图片做任何处理, 因此用户图片的宽高不受解码通道宽高的限制, 但受后级模块的限制。
- 用户图片属性设置成功之后 VDEC 将一直占着这块用户图片 MB, 直到销毁解码通道时才释放。
- 用户图片属性支持重复设置, 重复设置时, VDEC 会先释放之前的用户图片 MB 块, 然后占用当前的用户图片 MB 块。
- 同一个用户图片的属性可用于不同的通道, 如在网络异常情况下提示异常, 只要一个用户图片属性, 对所有通道设置。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_EnableUserPic。](#)

[RK_MPI_VDEC_DisableUserPic。](#)

4.15 RK_MPI_VDEC_EnableUserPic

【描述】

使能插入用户图片。

【语法】

RK_S32 RK_MPI_VDEC_EnableUserPic(VDEC_CHN VdChn, RK_BOOL bInstant)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围：[0, VDEC_MAX_CHN_NUM)。	输入
bInstant	使能用户图片方式。 取值范围： 0：使能延迟插入用户图片； 1：使能立刻插入用户图片；	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VDEC错误码 。

【注意】

- 通道必须已创建，否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 使能插入用户图片前必须先设置用户图片属性，否则返回操作不允许错误 RK_ERR_VDEC_NOT_PERM。
- 使能插入用户图片前必须先停止接收码流，否则返回操作不允许错误 RK_ERR_VDEC_NOT_PERM。
- 参数 bInstant 为 RK_TRUE 时，将立刻插入用户图片，此时 VDEC 会先复位解码通道，清楚通道缓存数据，然后插入用户图片。因此当禁止使能用户图片后送入新码流时，可能无法立即开始正确解码，必须等到下一个 I 帧到来才能开始正确解码。
- 参数 bInstant 为 RK_FALSE 时，将延迟插入用户图片，此时 VDEC 会先等待解码器把所有码流解完并输出全部图像后再插入用户图片。
- 使能插入用户图片之后，必须调用接口 RK_MPI_VDEC_DisableUserPic 禁止使能插入用户图片，才能重新设置新的用户图片属性。
- 重复使能插入用户图片将返回成功，但不会重复插入用户图片。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_SetUserPic。](#)

[RK_MPI_VDEC_DisableUserPic。](#)

4.16 RK_MPI_VDEC_DisableUserPic

【描述】

禁止使能插入用户图片。

【语法】

RK_S32 RK_MPI_VDEC_DisableUserPic(VDEC_CHN VdChn)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围: [0, VDEC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VDEC错误码 。

【注意】

- 通道必须已创建, 否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_EnableUserPic](#)。

4.17 RK_MPI_VDEC_SetChnParam

【描述】

设置解码通道参数。

【语法】

RK_S32 RK_MPI_VDEC_SetChnParam(VDEC_CHN VdChn, const [VDEC_CHN_PARAM_S](#) *pstParam)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围: [0, VDEC_MAX_CHN_NUM)。	输入
pstParam	通道参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VDEC错误码 。

【注意】

- 通道必须已创建, 否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 传入的 pstParam 不能为空, 否则会返回空指针错误 RK_ERR_VDEC_NULL_PTR。
- 通道参数设置不支持动态立即生效, 必须先停止接口码流, 然后再重新启动接收码流才可生效, 此过程解码通道会被复位。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_GetChnParam](#)。

4.18 RK_MPI_VDEC_GetChnParam

【描述】

获取解码通道参数。

【语法】

```
RK_S32 RK_MPI_VDEC_GetChnParam(VDEC_CHN VdChn, VDEC\_CHN\_PARAM\_S *pstParam)
```

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围: [0, VDEC_MAX_CHN_NUM)。	输入
pstParam	通道参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VDEC错误码 。

【注意】

- 通道必须已创建, 否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 传入的 pstParam 不能为空, 否则会返回空指针错误 RK_ERR_VDEC_NULL_PTR。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_SetChnParam](#)。

4.19 RK_MPI_VDEC_SetDisplayMode

【描述】

设置显示模式。

【语法】

```
RK_S32 RK_MPI_VDEC_SetDisplayMode(VDEC_CHN VdChn, VIDEO\_DISPLAY\_MODE\_E enDisplayMode)
```

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围: [0, VDEC_MAX_CHN_NUM)。	输入
enDisplayMode	显示模式枚举。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VDEC错误码 。

【注意】

- 通道必须已创建, 否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。

- 预览模式 (VIDEO_DISPLAY_MODE_PREVIEW)：实时性强，数据流转可能出现丢帧，可实现实时预览。VDEC 绑定的直接后级模块以非阻塞方式接收解码图像。比如直接后级为 VPSS，当解码帧存个数多于 VPSS 缓存队列个数 (VPSS Buffer队列满) 时，VPSS 将丢弃 VDEC 发送过来的图像，以达到不反压 VDEC 解码，实现实时预览。但当解码帧存个数少于 VPSS 缓存队列个数时，即使开启预览模式，VPSS 还是会反压解码。
- 回放模式 (VIDEO_DISPLAY_MODE_PLAYBACK)：实时性弱，数据流转不会丢帧，延迟长短与数据通路各模块缓存队列相关。VDEC 绑定的直接后级模块以阻塞方式接收解码图像，VDEC 绑定的直接后级模块能够反压 VDEC 解码。比如直接后级为 VPSS，当 VPSS 的图像缓存队列满时，不接收 VDEC 发送过来的图像，VDEC 发送图像失败后，启动重新发送机制，直到图像发送成功为止。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_SetDisplayMode](#)。

4.20 RK_MPI_VDEC_GetDisplayMode

【描述】

设置显示模式。

【语法】

RK_S32 RK_MPI_VDEC_GetDisplayMode(VDEC_CHN VdChn, [VIDEO_DISPLAY_MODE_E](#) *penDisplayMode)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围：[0, VDEC_MAX_CHN_NUM)。	输入
penDisplayMode	显示模式枚举指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VDEC错误码 。

【注意】

- 通道必须已创建，否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 传入的 penDisplayMode 不能为空，否则会返回空指针错误 RK_ERR_VDEC_NULL_PTR。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_SetDisplayMode](#)。

4.21 RK_MPI_VDEC_GetFd

【描述】

获取视频解码通道的设备文件句柄，用于查询解码通道数据有产生解码图像。

【语法】

RK_S32 RK_MPI_VDEC_GetFd(VDEC_CHN VdChn)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围: [0, VDEC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
正数值	有效返回值。
非正值	无效返回值。

【注意】

- 通道必须已创建，否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 获取到通道fd后，可通过 select 接口查询是否新的解码图像输出，然后再调用 RK_MPI_VDEC_GetFrame 获取解码图像，避免盲查询获取图像。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_GetFrame](#)。

4.22 RK_MPI_VDEC_CloseFd

【描述】

关闭视频解码的设备文件句柄。

【语法】

RK_S32 RK_MPI_VDEC_CloseFd(VDEC_CHN VdChn)

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围: [0, VDEC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	关闭成功。
-1	关闭失败。

【注意】

- 通道必须已创建，否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_GetFd](#)。

4.23 RK_MPI_VDEC_AttachMbPool

【描述】

将解码通道绑定到某个视频缓存 MB 池中。

【语法】

```
RK_S32 RK_MPI_VDEC_AttachMbPool(VDEC_CHN VdChn, MB_POOL hMbPool)
```

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围：[0, VDEC_MAX_CHN_NUM)。	输入
hMbPool	视频缓存 MB 池信息。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VDEC错误码 。

【注意】

- 通道必须已创建，否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 参数 hMbPool 必须是已创建且有效的 MB 内存池，否则返回错误 RK_ERR_VDEC_ILLEGAL_PARAM。
- 用户通过调用接口 RK_MPI_MB_CreatePool 创建一个视频缓存 MB 池，再通过调用接口 RK_MPI_VDEC_AttachMbPool 将创建 MB 内存池绑定到该解码通道中，不允许把同一个解码通道绑定到多个 MB 池中。
- 如果要切换当前解码通道绑定的 MB 池时，只需再调一次接口 RK_MPI_VDEC_AttachMbPool，即可绑定到新的 MB 内存池。
- 如果当前解码帧存分配方式使用的不是解码用户池（MB_SOURCE_USER），则返回错误 RK_ERR_VDEC_NOT_SUPPORT。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_DetachMbPool](#)。

4.24 RK_MPI_VDEC_DetachMbPool

【描述】

将解码通道从某个视频缓存 MB 池中解绑定。

【语法】

```
RK_S32 RK_MPI_VDEC_DetachMbPool(VDEC_CHN VdChn)
```

【参数】

参数名	描述	输入/输出
VdChn	视频解码通道号。 取值范围：[0, VDEC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VDEC错误码 。

【注意】

- 通道必须已创建, 否则会返回通道不存在的错误 RK_ERR_VDEC_UNEXIST。
- 如果当前解码帧存分配方式使用的不是解码用户池 (MB_SOURCE_USER) , 则返回VDEC错误码 RK_ERR_VDEC_NOT_SUPPORT。

【举例】

无。

【相关主题】

[RK_MPI_VDEC_AttachMbPool](#)。

5. 数据类型

视频解码相关数据类型、数据结构定义如下：

- [VDEC_CHN_ATTR_S](#): 定义解码通道属性。
- [VDEC_ATTR_VIDEO_S](#): 定义视频解码视频通道属性。
- [VIDEO_MODE_E](#): 定义码流发送方式枚举。
- [VDEC_CHN_STATUS_S](#): 定义通道状态结构体。
- [VDEC_DECODE_ERROR_S](#): 定义解码错误信息结构体。
- [VDEC_CHN_PARAM_S](#): 定义解码通道高级参数结构体。
- [VDEC_PARAM_VIDEO_S](#): 定义视频解码高级参数结构体。
- [VDEC_PARAM_PICTURE_S](#): 定义图片解码高级参数结构体。
- [VIDEO_DEC_MODE_E](#): 定义解码模式枚举。
- [VIDEO_OUTPUT_ORDER_E](#): 定义解码输出顺序枚举。
- [COMPRESS_MODE_E](#): 定义解码图像压缩模式枚举。
- [VDEC_PRTCL_PARAM_S](#): 定义与协议相关的内存分配参数结构体。
- [VDEC_STREAM_S](#): 定义解码码流结构体。
- [VIDEO_DISPLAY_MODE_E](#): 定义显示模式枚举。
- [VDEC_CHN_POOL_S](#): 定义解码通道绑定的 MB 池结构体。
- [VDEC_VIDEO_MOD_PARAM_S](#): 定义视频解码模块参数结构体。
- [VDEC_PICTURE_MOD_PARAM_S](#): 定义图片解码模块参数结构体。
- [VDEC_MOD_PARAM_S](#): 定义解码模块参数结构体。

5.1 VDEC_MAX_CHN_NUM

【说明】

定义解码通道最大个数。

【定义】

```
#define VDEC_MAX_CHN_NUM      64
```

【注意事项】

无

5.2 VDEC_CHN_ATTR_S

【说明】

定义解码通道属性

【定义】

```
typedef struct rkVDEC_CHN_ATTR_S {  
    PAYLOAD_TYPE_E enType;  
    VIDEO_MODE_E enMode;  
    RK_U32 u32PicWidth;  
    RK_U32 u32PicHeight;  
    RK_U32 u32PicVirWidth;  
    RK_U32 u32PicVirHeight;  
    RK_U32 u32StreamBufSize;  
    RK_U32 u32FrameBufSize;  
    RK_U32 u32FrameBufCnt;  
    RK_U32 u32StreamBufCnt;  
    union {  
        VDEC\_ATTR\_VIDEO\_S stVdecVideoAttr;  
    };  
} VDEC_CHN_ATTR_S;
```

【成员】

成员名称	描述	属性
enType	解码协议类型枚举值。	静态属性。
enMode	码流发送方式。	静态属性。
u32PicWidth	视频的宽度（以像素为单位）。	静态属性。
u32PicHeight	视频的高度（以像素为单位）。	静态属性。
u32PicVirWidth	不支持。	静态属性。
u32PicVirHeight	不支持。	静态属性。
u32StreamBufSize	不支持。	静态属性。
u32FrameBufSize	默认配置为0，内部按需计算大小。 若解码10bit码流，则需要设置为 RK_MPI_CAL_VDEC_GetPicBufferSize计算的解码buffer大小	静态属性。
u32FrameBufCnt	解码图像帧存个数。 取值范围：大于0，默认值为6，但是用户必须保证所配置的帧存个数满足解码码流帧存个数要求，否则无法正常解码。 H.264/H.265解码所需帧存个数为参考帧+2。 JPEG/MJPEG解码所需帧存个数为显示帧+1。仅PrivateMB模式有效。	静态属性。
u32StreamBufCnt	码流包存储个数。 取值范围：大于等于2，默认值为8	静态属性。

【注意事项】

- 帧存个数分配方法：

- 如果u32FrameBufCnt > 0，帧存个数按照u32FrameBufCnt来分配
- 如果u32FrameBufCnt == 0 && u32RefFrameNum > 0，帧存个数按照u32RefFrameNum + 2来分配
- 如果u32FrameBufCnt == 0 && u32RefFrameNum == 0，帧存个数按默认值分配
 - 帧存buffer大小，按照宽度和高度自动计算分配，用户必须保证配置的正确的宽高
 - 如果u32StreamBufCnt == 0，缓冲码流个数按默认值分配
 - 建议u32StreamBufCnt 配置最少为3

【举例】

无。

【相关主题】

无。

5.3 VDEC_ATTR_VIDEO_S

【说明】

定义视频解码视频通道属性。

【定义】

```
typedef struct rkVDEC_ATTR_VIDEO_S {  
    RK_U32 u32RefFrameNum; /RW, Range: [0, 16]; reference frame num./  
    RK_BOOL bTemporalMvpEnable; /RW;/  
    /specifies whether temporal motion vector predictors can be used for inter prediction/  
    RK_U32 u32TmvBufSize; /RW; tmv buffer size(Byte)/  
} VDEC_ATTR_VIDEO_S;
```

【成员】

成员名称	描述	属性
u32RefFrameNum	<p>参考帧的数目。</p> <p>取值范围：[0, 16]，以帧为单位。</p> <p>参考帧的数目决定解码时需要的参考帧个数，会较大的影响内存 MB 块占用，根据实际情况设置合适的值。</p> <p>-IPC码流：推荐设为 5。</p> <p>-测试码流：推荐设为 16。</p>	静态属性。
bTemporalMvpEnable	<p>解码的 H.264 码流有 B 帧，或者解码的 H.265 码流支持时域运动矢量预测（<code>sps_temporal_mvp_enabled_flag = 1</code>），则创建通道时需要设置此通道支持时域运动矢量预测，即需要设置 <code>bTemporalMvpEnable</code> 设置为 1。VDEC 模块内部将按照参考帧数量分配运动矢量 MV 存储所需的内存空间。反之，<code>bTemporalMvpEnable</code> 设置为 0。</p>	静态属性。
u32TmvBufSize	不支持	静态属性。

【注意事项】

无

【举例】

无。

【相关主题】

无。

5.4 VIDEO_MODE_E

【说明】

定义码流发送方式。

【定义】

```
typedef enum rkVIDEO_MODE_E {  
    VIDEO_MODE_STREAM = 0, /send by stream/  
    VIDEO_MODE_FRAME, /send by frame/  
    VIDEO_MODE_COMPAT, /One frame supports multiple packets sending./  
    /The current frame is considered to end when bEndOfFrame is equal to RK_TRUE/  
}
```

```
VIDEO_MODE_BUTT  
} VIDEO_MODE_E;
```

【成员】

成员名称	描述	属性
VIDEO_MODE_STREAM	按流方式发送码流。JPEG/MJPEG 解码不支持此模式。	静态属性。
VIDEO_MODE_FRAME	按帧方式发送码流。以帧为单位。	静态属性。
VIDEO_MODE_COMPAT	不支持	静态属性。

【注意事项】

当一帧码流的最后一包没有把 bEndOfFrame 为 RK_TRUE 时，还可以再发送一个bEndOfFrame 为 RK_TRUE 的空包给解码器内部标识当前帧码流已发送完毕。

【举例】

无

【相关主题】

无

5.5 VDEC_CHN_STATUS_S

【说明】

定义通道状态

【定义】

```
typedef struct rkVDEC_CHN_STATUS_S {  
    RK_CODEC_ID_E enType; /R; video type to be decoded/  
    RK_U32 u32LeftStreamBytes; /R; left stream bytes waiting for decode/  
    RK_U32 u32LeftStreamFrames; /R; left frames waiting for decode,only valid for VIDEO_MODE_FRAME/  
    RK_U32 u32LeftPics; /R; pics waiting for output/  
    RK_BOOL bStartRecvStream; /R; had started recv stream?/  
    RK_U32 u32RecvStreamFrames; /R; how many frames of stream has been received. valid when send by frame./  
    RK_U32 u32DecodeStreamFrames; /R; how many frames of stream has been decoded. valid when send by frame./  
    VDEC\_DECODE\_ERROR\_S stVdecDecErr; /R; information about decode error/  
    RK_U32 u32Width;  
    RK_U32 u32Height;  
} VDEC_CHN_STATUS_S;
```

【成员】

成员名称	描述
enType	解码协议类型枚举值。
u32LeftStreamBytes	码流 buffer 中待解码的 byte 数，包括正在解码的当前帧中未解码的 byte 数。
u32LeftStreamFrames	码流 buffer 中待解码的帧数，不包括正在解码的当前帧。 -1 表示无效。 流模式发送时无效。码流有错丢帧时可能计数不准。
u32LeftPics	图像 buffer 中剩余的 pic 数目。
bStartRecvStream	解码器是否已经启动接收码流。
u32RecvStreamFrames	码流 buffer 中已接收码流帧数。 -1 表示无效。 流模式发送时无效。
u32DecodeStreamFrames	码流 buffer 中已解码帧数。
stVdecDecErr	解码错误信息。
u32Width	图像宽度。
u32Height	图像高度。

【注意事项】

无

【举例】

无。

【相关主题】

无。

5.6 VDEC_DECODE_ERROR_S

【说明】

定义解码错误信息结构体。

【定义】

```
typedef struct rkVDEC_DECODE_ERROR_S {
    RK_S32 s32FormatErr; /R; format error. eg: do not support filed/
    RK_S32 s32PicSizeErrSet; /R; picture width or height is larger than channel width or height/
    RK_S32 s32StreamUnsprt; /R; unsupport the stream specification/
    RK_S32 s32PackErr; /R; stream package error/
    RK_S32 s32PrtclNumErrSet; /R; protocol num is not enough. eg: slice, pps, sps/
    RK_S32 s32RefErrSet; /R; reference num is not enough/
    RK_S32 s32PicBufSizeErrSet; /R; the buffer size of picture is not enough/
    RK_S32 s32StreamSizeOver; /R; the stream size is too big and force discard stream/
    RK_S32 s32VdecStreamNotRelease; /R; the stream not released for too long time/
} VDEC_DECODE_ERROR_S;
```

【成员】

成员名称	描述
s32FormatErr	不支持的格式。设置的fmt不在RK_CODEC_ID_E支持列表中
s32PicSizeErrSet	不支持。
s32StreamUnsprt	不支持的规格。设置的宽高小于0
s32PackErr	码流有错。
s32PrtclNumErrSet	不支持。
s32RefErrSet	不支持。
s32PicBufSizeErrSet	不支持。
s32StreamSizeOver	不支持。
s32VdecStreamNotRelease	不支持。

【注意事项】

- 所有错误信息均以累加计数的形式表现。比如解码每发现一次码流有错， s32PackErr 数值就加 1。
- 复位通道后所有错误信息计数清零

【举例】

无。

【相关主题】

无。

5.7 VDEC_CHN_PARAM_S

【说明】

定义解码通道高级参数。

【定义】

```
typedef struct rkVDEC_CHN_PARAM_S {
    PAYLOAD_TYPE_E enType;
    RK_U32 u32DisplayFrameNum;
    union {
        VDEC\_PARAM\_VIDEO\_S stVdecVideoParam;
        VDEC\_PARAM\_PICTURE\_S stVdecPictureParam;
    };
} VDEC_CHN_PARAM_S;
```

【成员】

成员名称	描述	属性
enType	解码协议。	静态属性。
u32DisplayFrameNum	不支持。	静态属性。
stVdecVideoParam	视频(H.264/H.265)解码高级参数。	静态属性。
stVdecPictureParam	图片(JPEG/MJPEG)解码高级参数。	静态属性。

【注意事项】

- 如果需要获取可以正常显示的解码数据，需要设置图像输出为非压缩：COMPRESS_MODE_NONE

【举例】

无。

【相关主题】

无

5.8 VDEC_PARAM_VIDEO_S

【说明】

定义视频解码高级参数。

【定义】

```
typedef struct rkVDEC_PARAM_VIDEO_S {  
    RK_S32 s32ErrThreshold;  
    VIDEO_DEC_MODE_E enDecMode;  
    VIDEO\_OUTPUT\_ORDER\_E enOutputOrder;  
    COMPRESS\_MODE\_E enCompressMode;  
    VIDEO_FORMAT_E enVideoFormat;  
} VDEC_PARAM_VIDEO_S;
```

【成员】

成员名称	描述	属性
s32ErrThreshold	不支持	静态属性。
enDecMode	不支持	静态属性。
enOutputOrder	解码图像输出顺序。 Default: VIDEO_OUTPUT_ORDER_DISP	静态属性。
enCompressMode	解码图像压缩模式。 取值范围：仅支持 COMPRESS_MODE_NONE 和COMPRESS_AFBC_16x16。 Default: COMPRESS_MODE_NONE	静态属性。
enVideoFormat	不支持	静态属性。

【注意事项】

- 如果需要获取可以正常显示的解码数据，需要设置图像输出为非压缩：COMPRESS_MODE_NONE

【举例】

无。

【相关主题】

无

5.9 VDEC_PARAM_PICTURE_S

【说明】

定义图形解码高级参数。

【定义】

```
typedef struct rkVDEC_PARAM_PICTURE_S {  
    PIXEL_FORMAT_E enPixelFormat; /RW; output pixel format/  
    RK_U32 u32Alpha; /RW, Range: [0, 255]; value 0 is transparent./  
    /*[0,127] is deemed to transparent when enPixelFormat is ARGB1555 or ABGR1555  
     [128,256] is deemed to non-transparent when enPixelFormat is ARGB1555 or ABGR1555  
     */  
} VDEC_PARAM_PICTURE_S;
```

【成员】

成员名称	描述	属性
enPixelFormat	JPEG(MJPEG)解码输出格式。 取值范围：仅支持以下几种输出格式 RK_FMT_BGR565、 RK_FMT_RGB888、 RK_FMT_YUV420SP RK_FMT_YUV420SP_VU、 RK_FMT_YUV422_YUYV、 RK_FMT_YUV422_UYVY、 Default: RK_FMT_YUV420SP	静态属性。
u32Alpha	不支持	静态属性。

【注意事项】

无

【举例】

无。

【相关主题】

无

5.10 VIDEO_DEC_MODE_E

【说明】

定义视频解码模式枚举。

【定义】

```
typedef enum rkVIDEO_DEC_MODE_E {
    VIDEO_DEC_MODE_IPB = 0,
    VIDEO_DEC_MODE_IP,
    VIDEO_DEC_MODE_I,
    VIDEO_DEC_MODE_BUTT
} VIDEO_DEC_MODE_E;
```

【成员】

成员名称	描述	属性
VIDEO_DEC_MODE_IPB	IPB 模式，即 I、P、B 帧都解码。	静态属性。
VIDEO_DEC_MODE_IP	不支持	静态属性。
VIDEO_DEC_MODE_I	不支持	静态属性。

【注意事项】

无

【举例】

无。

【相关主题】

无

5.11 VIDEO_OUTPUT_ORDER_E

【说明】

定义视频解码输出顺序枚举。

【定义】

```
typedef enum rkVIDEO_OUTPUT_ORDER_E {
    VIDEO_OUTPUT_ORDER_DISP = 0,
    VIDEO_OUTPUT_ORDER_DEC,
    VIDEO_OUTPUT_ORDER_BUTT
} VIDEO_OUTPUT_ORDER_E;
```

【成员】

成员名称	描述	属性
VIDEO_OUTPUT_ORDER_DISP	显示序输出。	静态属性。
VIDEO_OUTPUT_ORDER_DEC	解码序输出。	静态属性。

【注意事项】

解码有B帧的码流应设置为显示序输出。
没有B帧需要设置为解码序，加快解码输出

【举例】

无。

【相关主题】

无

5.12 COMPRESS_MODE_E

【说明】

定义解码图像压缩模式枚举。

【定义】

```
typedef enum rkCOMPRESS_MODE_E {
    COMPRESS_MODE_NONE = 0, /no compress/
    COMPRESS_AFBC_16x16,
    COMPRESS_MODE_BUTT
} COMPRESS_MODE_E;
```

【成员】

成员名称	描述	属性
COMPRESS_MODE_NONE	不压缩。（解码支持不压缩）。	静态属性。
COMPRESS_AFBC_16x16	帧压缩。（解码支持此压缩模式）	静态属性。

【注意事项】

无

【举例】

无。

【相关主题】

无

5.13 VDEC_STREAM_S

【说明】

定义视频解码的码流结构体

【定义】

```

typedef struct rkVDEC_STREAM_S {
    MB_BLK pMbBlk;
    RK_U32 u32Len;
    RK_U64 u64PTS;
    RK_BOOL bEndOfStream;
    RK_BOOL bEndOfFrame;
    RK_BOOL bBypassMbBlk;
} VDEC_STREAM_S;

```

【成员】

成员名称	描述
pMbBlk	码流包的地址。
u32Len	码流包的长度。以 byte 为单位。
u64PTS	码流包的时间戳。以 μs 为单位。
bEndOfStream	是否发完所有码流。
bEndOfFrame	不支持。
bBypassMbBlk	pMbBlk 是否需要拷贝

【注意事项】

- 按帧模式发送时，解码图像的时间戳等于码流包中的时间戳。
- 按流发送时，解码图像的时间戳等于 0
- 当发完所有码流后，把 bEndOfStream 置为 1，表示码流文件结束，这时解码器会

解完发送下来的所有码流并输出所有图像。如果发完所有码流后把 bEndOfStream 置为 0，解码器内部可能残余大于等于一帧的图像未解码输出，因为解码器必须等到下一帧码流到来才能知道当前帧已经结束，送入解码。

- 用户分配的buffer需要统一封装成MB_BLK类型，可通过RK_MPI_SYS_CreateMB来封装
- 拷贝模式，即设置bBypassMbBlk为RK_FALSE。在该模式下，用户data送入解码器后，解码器内部会做数据拷贝，用户data内存空间交由用户控制，用户可以循环复用此内存空间。
- 直通模式，即设置bBypassMbBlk为RK_TRUE，并设置释放回调函数user_data_callback，推荐使用此模式。在该模式下，用户data内存空间由解码器来释放，减少内存的消耗和cpu占用率

【举例】

- 直通模式

```

static RK_S32 user_data_callback(void *opaque) {
    if (RK_NULL != opaque) {
        free(opaque);
    }
    opaque = RK_NULL;
    return 0;
}
int main(int argc, const char **argv) {
    VDEC_STREAM_S stStream;
    MB_EXT_CONFIG_S pstMbExtConfig;
    MB_BLK buffer = RK_NULL;
    RK_S32 usersize = 1024;

    memset(&stMbExtConfig, 0, sizeof(MB_EXT_CONFIG_S));
    memset(&stStream, 0, sizeof(VDEC_STREAM_S));

    RK_S8 *userdata = (RK_S8 *)malloc(usersize);
    pstMbExtConfig.pFreeCB = user_data_callback;
    pstMbExtConfig.pOpaque = userdata;
    pstMbExtConfig.pu8VirAddr = userdata;
    pstMbExtConfig.u64Size = usersize;

    RK_MPI_SYS_CreateMB(&buffer, &pstMbExtConfig);
}

```

```

pstStream.u64PTS = userpts;
pstStream.pMbBlk = buffer;
pstStream.u32Len = usersize;
pstStream.bEndOfStream = RK_FALSE;
pstStream.bBypassMbBlk = RK_TRUE;

__RETRY:
s32Ret = RK_MPI_VDEC_SendStream(chn, &stStream, MAX_TIME_OUT_MS);
if (s32Ret < 0) {
    usleep(1000llu);
    goto __RETRY;
} else {
    RK_MPI_MB_ReleaseMB(stStream.pMbBlk);
}
.....
}

```

- 拷贝模式

```

int main(int argc, const char **argv) {
VDEC_STREAM_S stStream;
MB_EXT_CONFIG_S pstMbExtConfig;
MB_BLK buffer = RK_NULL;
RK_S32 usersize = 1024;

memset(&stMbExtConfig, 0, sizeof(MB_EXT_CONFIG_S));
memset(&stStream, 0, sizeof(VDEC_STREAM_S));

RK_S8 *userdata = (RK_S8 *)malloc(usersize);
pstMbExtConfig.pFreeCB = RK_NULL;
pstMbExtConfig.pOpaque = userdata;
pstMbExtConfig.pu8VirAddr = userdata;
pstMbExtConfig.u64Size = usersize;

RK_MPI_SYS_CreateMB(&buffer, &pstMbExtConfig);

pstStream.u64PTS = userpts;
pstStream.pMbBlk = buffer;
pstStream.u32Len = usersize;
pstStream.bEndOfStream = RK_FALSE;
pstStream.bBypassMbBlk = RK_FALSE;

__RETRY:
s32Ret = RK_MPI_VDEC_SendStream(chn, &stStream, MAX_TIME_OUT_MS);
if (s32Ret < 0) {
    usleep(1000llu);
    goto __RETRY;
} else {
    RK_MPI_MB_ReleaseMB(stStream.pMbBlk);
}
.....
}

```

【相关主题】

无。

5.14 VIDEO_DISPLAY_MODE_E

【说明】

定义显示模式枚举。

【定义】

```

typedef enum rkVIDEO_DISPLAY_MODE_E {
    VIDEO_DISPLAY_MODE_PREVIEW = 0x0,
    VIDEO_DISPLAY_MODE_PLAYBACK = 0x1,
    VIDEO_DISPLAY_MODE_BUTT
} VIDEO_DISPLAY_MODE_E;

```

【成员】

成员名称	描述
VIDEO_DISPLAY_MODE_PREVIEW	预览模式。
VIDEO_DISPLAY_MODE_PLAYBACK	回放模式。

【注意事项】

无

【举例】

无。

【相关主题】

无。

5.15 VDEC_VIDEO_MOD_PARAM_S

【说明】

定义视频解码模块参数结构体。

【定义】

```

typedef struct rkVDEC_VIDEO_MOD_PARAM_S {
    RK_U32 u32MaxPicWidth;
    RK_U32 u32MaxPicHeight;
    RK_U32 u32MaxSliceNum;
    RK_U32 u32VdhMsgNum;
    RK_U32 u32VdhBinSize;
    RK_U32 u32VdhExtMemLevel;
} VDEC_VIDEO_MOD_PARAM_S;

```

【成员】

成员名称	描述
u32MaxPicWidth	不支持。
u32MaxPicHeight	不支持。
u32MaxSliceNum	不支持。
u32VdhMsgNum	不支持。
u32VdhBinSize	不支持。
u32VdhExtMemLevel	不支持。

【注意事项】

无

【举例】

无。

【相关主题】

无。

5.16 VDEC_PICTURE_MOD_PARAM_S

【说明】

定义图片解码模块参数结构体。

【定义】

```
typedef struct rkVDEC_PICTURE_MOD_PARAM_S {  
    RK_U32 u32MaxPicWidth;  
    RK_U32 u32MaxPicHeight;  
    RK_BOOL bSupportProgressive;  
    RK_BOOL bDynamicAllocate;  
} VDEC_PICTURE_MOD_PARAM_S;
```

【成员】

成员名称	描述
u32MaxPicWidth	jpeg/mjpeg解码支持的最大宽度。不支持
u32MaxPicHeight	jpeg/mjpeg视频解码支持的最大高度。不支持
bSupportProgressive	不支持。
bDynamicAllocate	不支持。

【注意事项】

无

【举例】

无。

【相关主题】

无。

5.17 VDEC_MOD_PARAM_S

【说明】

定义解码模块参数结构体。

【定义】

```
typedef struct rkVDEC_MOD_PARAM_S {  
    MB_SOURCE_E enVdecMBSource; /RW, Range: [1, 3]; frame buffer mode/  
    RK_U32 u32MiniBufMode; /RW, Range: [0, 1]; stream buffer mode/  
    RK_U32 u32ParallelMode; /RW, Range: [0, 1]; VDPU working mode/  
    VDEC\_VIDEO\_MOD\_PARAM\_S stVideoModParam;  
    VDEC\_PICTURE\_MOD\_PARAM\_S stPictureModParam;  
} VDEC_MOD_PARAM_S;
```

【成员】

成员名称	描述
enVdecMBSource	解码帧存 MB 来源。 取值范围：仅支持 MB_SOURCE_MODULE、 MB_SOURCE_PRIVATE、 MB_SOURCE_USER Default: MB_SOURCE_MODULE
u32MiniBufMode	不支持。
u32ParallelMode	不支持。
stVideoModParam	不支持。

【注意事项】

- enVdecMBSource模式必须在所有通道未被启动前设置，否则不生效。
- MB_SOURCE_USER 模式时，启动解码通道前必须调用[RK_MPI_VDEC_AttachMbPool](#)将用户申请的内存池注入使用，否则该通道无解码缓存可用。

【举例】

无。

【相关主题】

无。

5.18 FRAME_FLAG_E

【说明】

定义帧的类型。

【定义】

```
typedef enum rkFRAME_FLAG_E {
    FRAME_FLAG_SNAP_FLASH = 0x1 << 0,
    FRAME_FLAG_SNAP_CUR = 0x1 << 1,
    FRAME_FLAG_SNAP_REF = 0x1 << 2,
    FRAME_FLAG_SNAP_END = 0x1 << 31,
    FRAME_FLAG_BUTT
} FRAME_FLAG_E;
```

【成员】

成员名称	描述
FRAME_FLAG_SNAP_FLASH	不支持。
FRAME_FLAG_SNAP_CUR	不支持。
FRAME_FLAG_SNAP_REF	不支持。
FRAME_FLAG_SNAP_END	解码最后一帧数据。

【注意事项】

无。

【举例】

无。

【相关主题】

无。

5.19 VIDEO_FRAME_INFO_S

【说明】

定义视频图像帧信息结构体。

【定义】

```
typedef struct rkVIDEO_FRAME_S {  
    MB_BLK      pMbBlk;  
    RK_U32      u32Width;  
    RK_U32      u32Height;  
    RK_U32      u32VirWidth;  
    RK_U32      u32VirHeight;  
    VIDEO_FIELD_E   enField;  
    PIXEL_FORMAT_E  enPixelFormat;  
    VIDEO_FORMAT_E  enVideoFormat;  
    COMPRESS_MODE_E enCompressMode;  
    DYNAMIC_RANGE_E enDynamicRange;  
    COLOR_GAMUT_E   enColorGamut;  
  
    RK_VOID      *pVirAddr[RK_MAX_COLOR_COMPONENT];  
  
    RK_U32      u32TimeRef;  
    RK_U64      u64PTS;  
  
    RK_U64      u64PrivateData;  
    RK_U32      u32FrameFlag; /* FRAME_FLAG_E, can be OR operation. */  
} VIDEO_FRAME_S;  
  
typedef struct rkVIDEO_FRAME_INFO_S {  
    VIDEO_FRAME_S stVFrame;  
} VIDEO_FRAME_INFO_S;
```

【成员】

成员名称	描述
pMbBlk	图像数据
u32Width	图像实际宽度。
u32Height	图像实际高度。
u32VirWidth	图像虚宽。
u32VirHeight	图像虚高。
enField	帧场模式。
enVideoFormat	不支持。
enPixelFormat	目标图像像素格式。
enCompressMode	目标图像压缩模式。
enDynamicRange	不支持。
enColorGamut	不支持。
	图像Y和UV的地址。
pVirAddr	取值范围: Y地址:pVirAddr[RK_COLOR_YUV_Y_PLANE] uv地址:pVirAddr[RK_COLOR_YUV_UV_PLANE]
u32TimeRef	不支持。
u64PTS	图像时间戳。
u64PrivateData	不支持。
u32FrameFlag	当前帧的标记，使用 FRAME_FLAG_E 里面的值标记，可以按位或操作。

【注意事项】

无。

【举例】

- 用户获取YUV图像数据，可以通过RK_MPI_MB_Handle2VirAddr转换pMbBlk成虚拟地址来使用

```
data = RK_MPI_MB_Handle2VirAddr(sFrame.stVFrame.pMbBlk);
fwrite(data, 1, sFrame.stVFrame.u32Width * sFrame.stVFrame.u32Height * 3 / 2, fp);
fflush(fp);
```

【相关主题】

无。

5.20 VIDEO_FIELD_E

【说明】

定义视频图像帧场类型。

【定义】

```
typedef enum rkVIDEO_FIELD_E {
    VIDEO_FIELD_TOP    = 0x1, /* even field */
    VIDEO_FIELD_BOTTOM  = 0x2, /* odd field */
    VIDEO_FIELD_INTERLACED = 0x3, /* two interlaced fields */
    VIDEO_FIELD_FRAME   = 0x4, /* frame */

    VIDEO_FIELD_BUTT
} VIDEO_FIELD_E;
```

【成员】

成员名称	描述
VIDEO_FIELD_TOP	顶场类型。 (不支持)
VIDEO_FIELD_BOTTOM	底场类型。 (不支持)
VIDEO_FIELD_INTERLACED	两场间插类型
VIDEO_FIELD_FRAME	帧类型。

【注意事项】

如果bDeiEn==RK_TRUE，且片源是隔行扫描，做Deinterlace处理，帧数翻倍

如果bDeiEn==RK_FALSE，且片源是隔行扫描，顶场和底场合成一帧输出

【举例】

无。

【相关主题】

[VIDEO FRAME INFO S](#)

6. VDEC错误码

视频解码 API VDEC错误码如下所示：

错误代码	宏定义	描述
0xA0058002	RK_ERR_VDEC_INVALID_CHNID	VDEC 通道号无效
0xA0058003	RK_ERR_VDEC_ILLEGAL_PARAM	VDEC 参数设置无效
0xA0058004	RK_ERR_VDEC_EXIST	VDEC 通道已创建
0xA0058005	RK_ERR_VDEC_UNEXIST	VDEC 通道未创建
0xA0058006	RK_ERR_VDEC_NULL_PTR	输入参数空指针错误
0xA0058008	RK_ERR_VDEC_NOT_SUPPORT	操作不支持
0xA0058009	RK_ERR_VDEC_NOT_PERM	操作不允许
0xA005800C	RK_ERR_VDEC_NOMEM	分配内存失败
0xA005800D	RK_ERR_VDEC_NOBUF	分配 BUF 池失败
0xA005800E	RK_ERR_VDEC_BUF_EMPTY	图像队列为空
0xA005800F	RK_ERR_VDEC_BUF_FULL	图像队列满状态
0xA0058010	RK_ERR_VDEC_NOTREADY	VDEC 系统未初始化
0xA0058012	RK_ERR_VDEC_BUSY	VDEC 系统忙
0xA0058013	RK_ERR_VDEC_BADADDR	错误的地址

视频编码

[概述](#)

[功能描述](#)

[举例](#)

[API 参考](#)

[数据类型](#)

[VENC 错误码](#)

1. 概述

VENC 模块，即视频编码模块，主要支持H264、H265、JPEG、MJPEG。本模块支持多路实时编码，且每路编码独立，编码协议和编码 profile 可以不同。本模块支持视频编码同时，调用 Region 模块对编码图像内容进行叠加和遮挡。

VENC 模块的输入源包括以下几种：

- 用户态读取图像文件向编码模块发送数据；
- 视频输入（VI）模块采集的图像经视频处理子系统（VPSS）发送到编码模块；
- 视频输入（VI）模块采集的图像直接发送到编码模块；
- 视频解码（VDEC）模块解码图像经视频处理子系统（VPSS）发送到编码模块；

不同型号的芯片支持不同的编码规格，芯片支持的编码规格如下表所示。

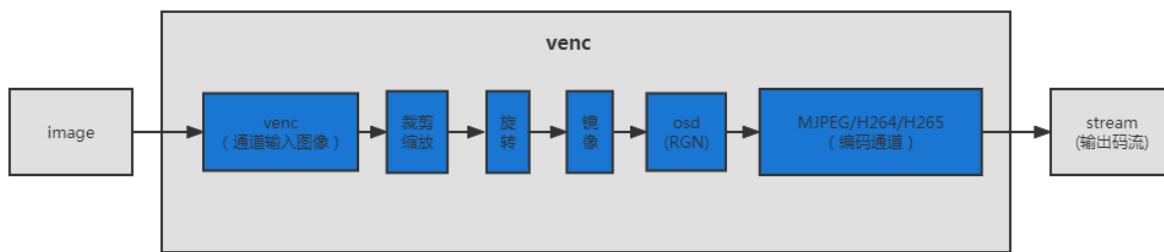
芯片	支持的编码类别	支持的编码格式	支持的编码规格
RV1109/RV1126	JPEG、MJPEG	YCbCr formats: YCbCr 4:2:0 planar YCbCr 4:2:0 semi-planar YCbYCr 4:2:2 CbYCrY 4:2:2 Interleaved RGB formats: RGB444 to BGR444 RGB555 to BGR555 RGB565 to BGR565 ARGB8888 to BRGA8888 RGB101010 BRG 101010	96x96 to 8192x8192(64 million pixels) Step size 4 pixels Up to 90 million pixels per second
RV1109/RV1126	H264	ARGB, RGB, YUV422/420P/SP Arm AFBC YUV422/420	high profile encoding, up to level 5.1; 64x64 -- 4096x4096 4096x2304@30fps
RV1109/RV1126	H265	ARGB, RGB, YUV422/420P/SP Arm AFBC YUV422/420	main profile encoding, up to level 5.0; 64x64 -- 4096x4096 4096x2304@30fps
RK356x	JPEG、MJPEG	YCbCr formats: YCbCr 4:2:0 planar YCbCr 4:2:0 semi-planar YCbYCr 4:2:2 CbYCrY 4:2:2 Interleaved RGB formats: RGB444 to BGR444 RGB555 to BGR555 RGB565 to BGR565 ARGB8888 to BRGA8888 RGB101010 BRG 101010	96x96 to 8192x8192(64 million pixels) Step size 4 pixels Up to 90 million pixels per second
RK356x	H264	ARGB, RGB, YUV422/420P/SP Arm AFBC YUV422/420	high profile encoding, up to level 5.1; 64x64 -- 1920x1080 1920x1080@60fps
RK356x	H265	ARGB, RGB, YUV422/420P/SP Arm AFBC YUV422/420	main profile encoding, up to level 5.0; 64x64 -- 1920x1080 1920x1080@60fps 注：支持tile编码
RK3588	JPEG、MJPEG	YCbCr formats: YCbCr 4:2:0 planar YCbCr 4:2:0 semi-planar YCbYCr 4:2:2 CbYCrY 4:2:2 Interleaved RGB formats: RGB444 to BGR444 RGB555 to BGR555 RGB565 to BGR565 ARGB8888 to BRGA8888 RGB101010 BRG 101010	96x96 to 8192x8192(64 million pixels) Step size 4 pixels Up to 90 million pixels per second 1920x1080@60fps 注：以上为单核编码数据，共有四核JPEG/MJPEG编码器

芯片	支持的编码类别	支持的编码格式	支持的编码规格
RK3588	H264	ARGB, RGB, YUV422/420P/SP Arm AFBC YUV422/420	High Profile, level 6.0 Resolution upto 16384x8192 1920x1080@240fps 注：以上单核编码数据，共有俩核H264/H265编码器。
RK3588	H265	ARGB, RGB, YUV422/420P/SP Arm AFBC YUV422/420	Main Profile, Level 6.0 High Tier Resolution upto 16384x8192 1920x1080@240fps 注：以上单核编码数据，共有俩核H264/H265编码器； 支持tile编码。

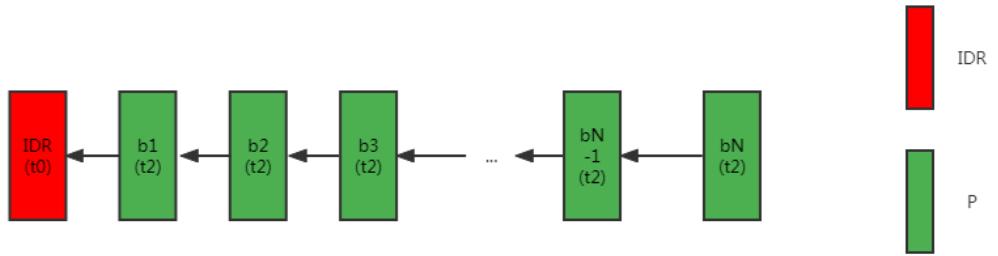
2. 功能描述

2.1 编码数据流程图

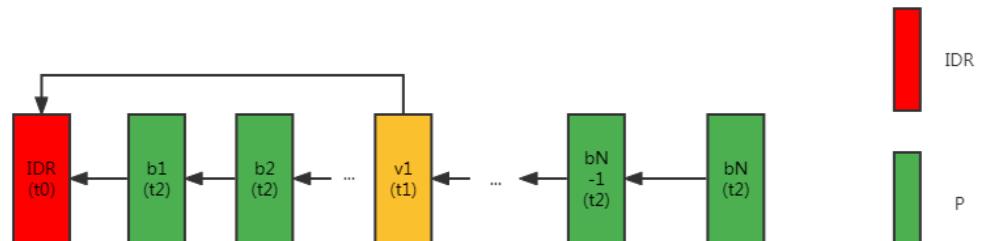
典型的编码流程包括了输入图像的接收、图像的选择、镜像、裁剪缩放、图像内容的遮挡和覆盖、图像的编码、以及码流的输出等过程。内部数据流程处理如下图。



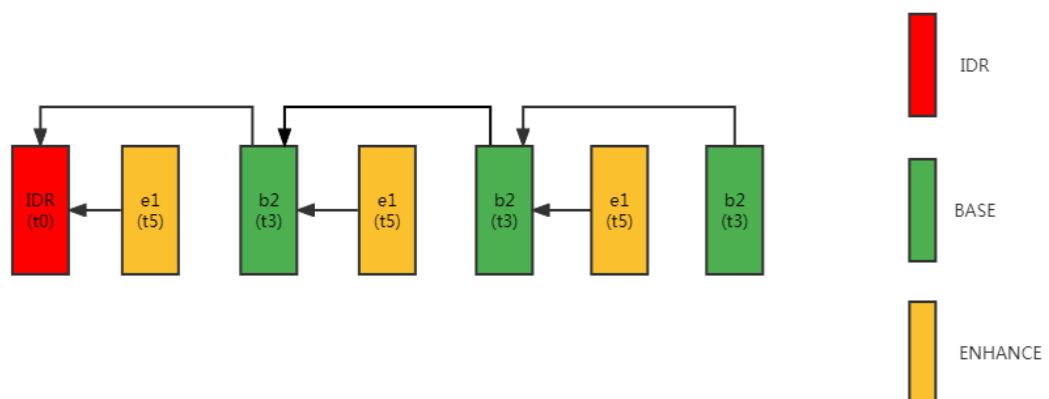
2.2 GOP结构



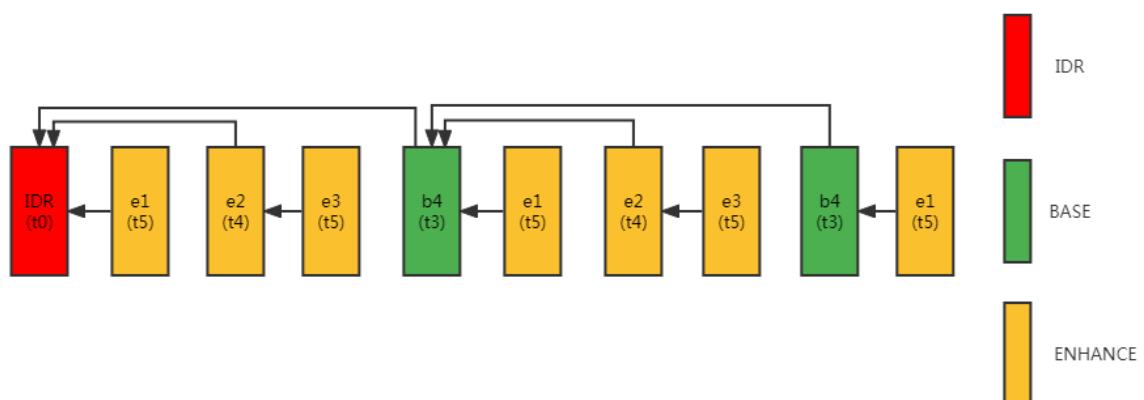
NORMALP参考关系



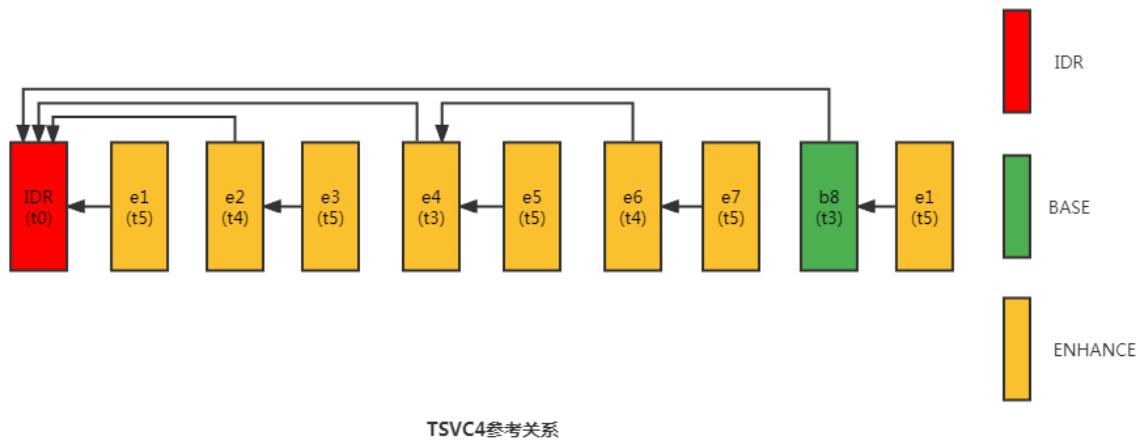
SMARTP参考关系



TSVC2参考关系



TSVC3参考关系



【注意】

- t0-t5对应[H264E_REF_TYPE_E](#)中BASE_IDRSlice到ENHANCE_PSlice_NOTFORREF的定义。

3. 举例

```

VENC_RECV_PIC_PARAM_S pstRecvParam;
VENC_CHN_ATTR_S pstAttr;
MB_POOL vencPool

pstAttr.stVencAttr.enType = RK_VIDEO_ID_AVC;
pstAttr.stVencAttr.u32PicWidth = 720;
pstAttr.stVencAttr.u32PicHeight = 576;
pstAttr.stVencAttr.u32MaxPicWidth = 720;
pstAttr.stVencAttr.u32MaxPicHeight = 576;
pstAttr.stVencAttr.u32StreamBufCnt = 4;
pstAttr.stVencAttr.u32BufSize = 720 * 576 * 3 / 2;
pstAttr.stVencAttr.enPixelFormat = RK_FMT_YUV420SP;

RK_MPI_VENC_CreateChn(0, &gVencCtx->pstAttr);
RK_MPI_VENC_StartRecvFrame(0, &pstRecvParam);

MB_POOL_CONFIG_S pstMbPoolCfg;
memset(&pstMbPoolCfg, 0, sizeof(MB_POOL_CONFIG_S));
pstMbPoolCfg.u64MBSize = 720 * 576 * 2;
pstMbPoolCfg.u32MBCnt = 10;
pstMbPoolCfg.enAllocType = MB_ALLOC_TYPE_DMA;

vencPool = RK_MPI_MB_CreatePool(&pstMbPoolCfg);

RK_MPI_VENC_SendFrame(0, pstFrame, -1);
RK_MPI_VENC_GetStream(0, pstStream, -1);
RK_MPI_VENC_ReleaseStream(0, pstStream);

RK_MPI_VENC_StopRecvFrame(0);
RK_MPI_VENC_DestroyChn(0);
RK_MPI_MB_DestroyPool(vencPool);

```

4. API 参考

视频编码模块主要提供视频编码通道的创建和销毁、视频编码通道的复位、开启和停止接收图像、设置和获取编码通道属性、获取和释放码流等功能。

该功能模块为用户提供以下 API：

- [RK_MPI_VENC_SetModParam](#): 设置编码相关的模块参数。
- [RK_MPI_VENC_GetModParam](#): 获取编码相关的模块参数。
- [RK_MPI_VENC_CreateChn](#): 创建编码通道。
- [RK_MPI_VENC_DestroyChn](#): 销毁编码通道。
- [RK_MPI_VENC_ResetChn](#): 复位编码通道。
- [RK_MPI_VENC_StartRecvFrame](#): 开启编码通道接收输入图像。
- [RK_MPI_VENC_StopRecvFrame](#): 停止编码通道接收输入图像。
- [RK_MPI_VENC_QueryStatus](#): 查询编码通道状态。
- [RK_MPI_VENC_SetChnAttr](#): 设置编码通道的编码属性。
- [RK_MPI_VENC_GetChnAttr](#): 获取编码通道的编码属性。
- [RK_MPI_VENC_SendFrame](#): 用户发送原始图像进行编码。
- [RK_MPI_VENC_SendFrameEx](#): 支持用户发送原始图像及该图的QpMap表信息进行编码。
- [RK_MPI_VENC_GetStream](#): 获取编码码流。
- [RK_MPI_VENC_ReleaseStream](#): 释放码流缓存。
- [RK_MPI_VENC_SetJpegParam](#): 设置 JPEG 编码的参数集合。
- [RK_MPI_VENC_GetJpegParam](#): 获取 JPEG 编码的参数集合。
- [RK_MPI_VENC_GetRcParam](#): 获取通道码率控制高级参数。
- [RK_MPI_VENC_SetRcParam](#): 设置通道码率控制高级参数。
- [RK_MPI_VENC_RequestIDR](#): 请求 IDR 帧。
- [RK_MPI_VENC_GetRoiAttr](#): 获取编码通道的兴趣区域编码配置。
- [RK_MPI_VENC_SetRoiAttr](#): 设置编码通道的兴趣区域编码配置。
- [RK_MPI_VENC_SetChnParam](#): 设置通道参数。
- [RK_MPI_VENC_GetChnParam](#): 获取通道参数。
- [RK_MPI_VENC_InsertUserData](#): 插入用户数据。
- [RK_MPI_VENC_SetRcAdvParam](#): 设置 RC 模块的高级参数。
- [RK_MPI_VENC_GetRcAdvParam](#): 获取 RC 模块的高级参数。
- [RK_MPI_VENC_GetFd](#): 获取编码通道对应的设备文件句柄。
- [RK_MPI_VENC_CloseFd](#): 关闭编码通道对应的设备文件句柄。
- [RK_MPI_VENC_SetSuperFrameStrategy](#): 设置编码超大帧配置。
- [RK_MPI_VENC_GetSuperFrameStrategy](#): 获取编码超大帧配置。
- [RK_MPI_VENC_SetFrameLostStrategy](#): 设置编码通道瞬时码率超过阈值时丢帧策略。
- [RK_MPI_VENC_GetFrameLostStrategy](#): 获取编码通道瞬时码率超过阈值时丢帧策略。
- [RK_MPI_VENC_SetIntraRefresh](#): 设置 P 帧刷 l slice 的参数。
- [RK_MPI_VENC_GetIntraRefresh](#): 获取 P 帧刷 l slice 的参数。
- [RK_MPI_VENC_SetHierarchicalQp](#): 设置分层 qp 参数。
- [RK_MPI_VENC_GetHierarchicalQp](#): 获取分层 qp 参数。
- [RK_MPI_VENC_SetChnRotation](#): 设置通道旋转角度。
- [RK_MPI_VENC_GetChnRotation](#): 获取通道旋转角度。
- [RK_MPI_VENC_AttachMbPool](#): 将编码通道绑定到某个视频缓存 MB 池中。
- [RK_MPI_VENC_DetachMbPool](#): 将编码通道从某个视频缓存 MB 池中解绑定。
- [RK_MPI_VENC_SetH264IntraPred](#): 设置H.264协议编码通道的帧内预测属性。
- [RK_MPI_VENC_GetH264IntraPred](#): 获取H.264协议编码通道的帧内预测属性。
- [RK_MPI_VENC_SetH264Trans](#): 设置H.264协议编码通道的变换、量化的属性。
- [RK_MPI_VENC_GetH264Trans](#): 获取H.264协议编码通道的变换、量化的属性。
- [RK_MPI_VENC_SetH264Entropy](#): 设置H.264协议编码通道的熵编码模式。
- [RK_MPI_VENC_GetH264Entropy](#): 获取H.264协议编码通道的熵编码模式。
- [RK_MPI_VENC_SetH264Dbblk](#): 设置H.264协议编码通道的Deblocking 类型。
- [RK_MPI_VENC_GetH264Dbblk](#): 获取H.264协议编码通道的Deblocking 类型。
- [RK_MPI_VENC_SetH264Vui](#): 设置H.264协议编码通道的Vui参数。
- [RK_MPI_VENC_GetH264Vui](#): 获取H.264协议编码通道的Vui参数。
- [RK_MPI_VENC_SetH265Trans](#): 设置H.265协议编码通道的变换、量化的属性。
- [RK_MPI_VENC_GetH265Trans](#): 获取H.265协议编码通道的变换、量化的属性。
- [RK_MPI_VENC_SetH265Entropy](#): 设置H.265协议编码通道的熵编码模式。
- [RK_MPI_VENC_GetH265Entropy](#): 获取H.265协议编码通道的熵编码模式。
- [RK_MPI_VENC_SetH265Dbblk](#): 设置H.265协议编码通道的Deblocking 类型。
- [RK_MPI_VENC_GetH265Dbblk](#): 获取H.265协议编码通道的Deblocking 类型。
- [RK_MPI_VENC_SetH265Sao](#): 设置H.265协议编码通道的Sao属性。
- [RK_MPI_VENC_GetH265Sao](#): 获取H.265协议编码通道的Sao属性。
- [RK_MPI_VENC_SetH265PredUnit](#): 设置H.265协议编码通道的PU属性。
- [RK_MPI_VENC_GetH265PredUnit](#): 获取H.265协议编码通道的PU属性。
- [RK_MPI_VENC_SetH265Vui](#): 设置H.265协议编码通道的Vui参数。

- [RK_MPI_VENC_GetH265Vui](#): 获取H.265协议编码通道的Vui参数。
- [RK_MPI_VENC_SetRefParam](#): 设置H.264/H.265编码通道高级跳帧参考参数。
- [RK_MPI_VENC_GetRefParam](#): 获取H.264/H.265编码通道高级跳帧参考参数。
- [RK_MPI_VENC_SetMjpegParam](#): 设置 MJPEG 协议编码通道的高级参数。
- [RK_MPI_VENC_GetMjpegParam](#): 获取 MJPEG 协议编码通道的高级参数配置。
- [RK_MPI_VENC_SetQpmap](#): 设置Qpmap MB_BLK。
- [RK_MPI_VENC_GetQpmap](#): 获取Qpmap MB_BLK。
- [RK_MPI_VENC_SetDeBreathEffect](#): 设置去除呼吸效应参数。
- [RK_MPI_VENC_GetDeBreathEffect](#): 获取去除呼吸效应参数。
- [RK_MPI_VENC_SetChnRefBufShareAttr](#): 设置编码通道参考帧共享属性。
- [RK_MPI_VENC_GetChnRefBufShareAttr](#): 获取编码通道参考帧共享属性。
- [RK_MPI_VENC_SetComboAttr](#): 设置编码通道Combo属性。
- [RK_MPI_VENC_GetComboAttr](#): 获取编码通道Combo属性。
- [RK_MPI_VENC_SetChnBufWrapAttr](#): 设置编码通道Buf卷绕属性。
- [RK_MPI_VENC_GetChnBufWrapAttr](#): 获取编码通道Buf卷绕属性。
- [RK_MPI_VENC_SetSliceSplit](#): 设置编码通道的slice分割属性。
- [RK_MPI_VENC_GetSliceSplit](#): 获取编码通道的slice分割属性。

4.1 RK_MPI_VENC_SetModParam

【描述】

设置编码相关的模块参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetModParam(const VENC\_PARAM\_MOD\_S *pstModParam);
```

【参数】

参数名	描述	输入/输出
pstModParam	编码模块参数指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 此接口必须在所有编码通道创建前调用。
- 如果pstModParam为空, 则返回失败。
- 如果没有调用此接口, 对应参数默认值查看[VENC_PARAM_MOD_S](#)结构体说明。

4.2 RK_MPI_VENC_GetModParam

【描述】

获取编码相关的模块参数。

【语法】

```
RK_S32 RK_MPI_VENC_GetModParam(VENC\_PARAM\_MOD\_S(#VENC\_PARAM\_MOD\_S) *pstModParam);
```

【参数】

参数名	描述	输入/输出
pstModParam	编码模块参数指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果pstModParam为空, 则返回失败。

4.3 RK_MPI_VENC_CreateChn

【描述】

创建编码通道。

【语法】

```
RK_S32 RK_MPI_VENC_CreateChn(VENC_CHN VeChn, const VENC\_CHN\_ATTR\_S *pstAttr);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstAttr	编码通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

无

4.4 RK_MPI_VENC_DestroyChn

【描述】

销毁编码通道。

【语法】

```
RK_S32 RK_MPI_VENC_DestroyChn(VENC_CHN VeChn);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

无

4.5 RK_MPI_VENC_ResetChn

【描述】

复位通道。

【语法】

```
RK_S32 RK_MPI_VENC_ResetChn(VENC_CHN VeChn);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

无

4.6 RK_MPI_VENC_StartRecvFrame

【描述】

开始编码通道接收输入图像。

【语法】

```
RK_S32 RK_MPI_VENC_StartRecvFrame(VENC_CHN VeChn, const VENC\_RECV\_PIC\_PARAM\_S *pstRecvParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstRecvParam	接收图像参数结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

无

4.7 RK_MPI_VENC_StopRecvFrame

【描述】

停止编码通道接收输入图像。

【语法】

```
RK_S32 RK_MPI_VENC_StopRecvFrame(VENC_CHN VeChn);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

无

4.8 RK_MPI_VENC_QueryStatus

【描述】

查询编码通道状态。

【语法】

```
RK_S32 RK_MPI_VENC_QueryStatus(VENC_CHN VeChn, VENC\_CHN\_STATUS\_S *pstStatus);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstStatus	编码通道的状态指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

无

4.9 RK_MPI_VENC_SetChnAttr

【描述】

设置编码通道属性。

【语法】

```
RK_S32 RK_MPI_VENC_SetChnAttr(VENC_CHN VeChn, const VENC\_CHN\_ATTR\_S *pstChnAttr);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstChnAttr	编码通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

无

4.10 RK_MPI_VENC_GetChnAttr

【描述】

获取编码通道属性。

【语法】

```
RK_S32 RK_MPI_VENC_GetChnAttr(VENC_CHN VeChn, VENC\_CHN\_ATTR\_S *pstChnAttr);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstChnAttr	编码通道属性指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

无

4.11 RK_MPI_VENC_SendFrame

【描述】

用户发送原始图像进行编码。

【语法】

```
RK_S32 RK_MPI_VENC_SendFrame(VENC_CHN VeChn, const VIDEO\_FRAME\_INFO\_S *pstFrame, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/ 输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstFrame	原始图像信息结构指针。	输入
s32MilliSec	超时参数 s32MilliSec 设为-1 时，为阻塞接口；0 时为非阻塞接口；大于 0 时为超时等待时间，超时时间的单位为毫秒（ms）。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 输入数据宽高对齐有特别要求，具体说明详见[VIDEO_FRAME_INFO_S](#)。

4.12 RK_MPI_VENC_SendFrameEx

【描述】

支持用户发送原始图像及该图的QpMap表信息进行编码。

【语法】

```
RK_S32 RK_MPI_VENC_SendFrameEx(VENC_CHN VeChn, const USER\_FRAME\_INFO\_S *pstFrame, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/ 输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstFrame	原始图像信息结构指针。	输入
s32MilliSec	超时参数 s32MilliSec 设为-1 时，为阻塞接口；0 时为非阻塞接口；大于 0 时为超时等待时间，超时时间的单位为毫秒（ms）。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- QpMap定义参考[RK_MPI_VENC_SetQpmap](#)。
- 仅编码通道协议为 H.264 或者 H.265 支持此接口。
- Rc 为 fixqp 模式下不支持该接口。

4.13 RK_MPI_VENC_GetStream

【描述】

获取编码的码流。

【语法】

```
RK_S32 RK_MPI_VENC_GetStream(VENC_CHN VeChn, VENC_STREAM_S *pstStream, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstStream	码流结构体指针。	输出
s32MilliSec	超时参数 s32MilliSec 设为-1 时，为阻塞接口；0 时为非阻塞接口；大于 0 时为超时等待时间，超时时间的单位为毫秒 (ms) 。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 支持单包、多包模式方式获取码流：
 - 单包模式（默认为此模式）则每次获取的为单个VENC_PACK_S信息，通过pMbBlk及u32Len获取数据（代码见举例的单包模式），VENC_PACK_INFO_S为当前帧的NAL信息包（H264包含SPS/PPS/SEI包类型信息，H265包含VPS/SPS/PPS/SEI包类型信息）。
 - 多包模式（目前仅H264/H265支持此模式）则每次获取的为多个VENC_PACK_S（目前最大设置支持16个），每个VENC_PACK_S包含单个NAL包或SLICE包（split slice模式开启时），通过pMbBlk、u32Len及u32Offset获取单个NAL/SLICE包数据（代码见举例的多包模式）。
- 码流结构体VENC_STREAM_S：
 - 码流包信息指针 pstPack
指向一组 VENC_PACK_S 的内存空间，该空间由调用者分配。如果是单包模式获取，则此空间不小于 sizeof (VENC_PACK_S) 的大小；如果是多包模式获取，则此空间不小于 N × sizeof (VENC_PACK_S) 的大小，其中 N 代表当前帧之中的包的个数（目前需要按照最大包个数16个设置）。
 - 码流包个数 u32PackCount
在输入时，此值指定 pstPack 中 VENC_PACK_S的个数。按单包模式获取时，u32PackCount 必须不小于 1；按多包模式获取时，u32PackCount 必须不小于当前帧的包个数（目前需要按照最大包个数16个设置，配置过小可能会存在丢数据包问题）。在函数调用成功后，u32PackCount 返回实际填充 pstPack 的包的个数。

【举例】

- 单包模式（对应模块参数的u32OneStreamBuffer为1，默认为此模式）

```
stFrame.pstPack = reinterpret_cast<VENC_PACK_S *>(malloc(sizeof(VENC_PACK_S)));
while (!pstCtx->threadExit) {
    stFrame.u32PackCount = 1;
    s32Ret = RK_MPI_VENC_GetStream(u32Ch, &stFrame, -1);
    if (s32Ret >= 0) {
        s32StreamCnt++;
        for (RK_U32 i = 0; i < stFrame.pstPack->u32DataNum; i++) {
            RK_LOGD("get chn %d stream %d index %d type %d offset %d lenght %d", u32Ch, s32StreamCnt, i,
                    stFrame.pstPack->stPackInfo[i].u32PackType,
                    stFrame.pstPack->stPackInfo[i].u32PackOffset,
                    stFrame.pstPack->stPackInfo[i].u32PackLength);
        }
        if (pstCtx->dstFilePath != RK_NULL) {
            pData = (char *)RK_MPI_MB_Handle2VirAddr(stFrame.pstPack->pMbBlk);
            fwrite(pData, 1, stFrame.pstPack->u32Len, fp);
            fflush(fp);
        }
        RK_MPI_VENC_ReleaseStream(u32Ch, &stFrame);
        if (stFrame.pstPack->bStreamEnd == RK_TRUE) {
```

```

        RK_LOGI("chn %d reach EOS stream", u32Ch);
        break;
    }
} else {
    if (pstCtx->threadExit) {
        break;
    }
    usleep(1000llu);
}
}

```

- 多包模式（对应模块参数的u32OneStreamBuffer为0，目前仅H264/H265支持此模式）

```

#define MAX_PACKET_NUM      16

stFrame.pstPack = reinterpret_cast<VENC_PACK_S *>(malloc(MAX_PACKET_NUM * sizeof(VENC_PACK_S)));
while (!pstCtx->threadExit) {
    stFrame.u32PackCount = MAX_PACKET_NUM; // 输入参数需要按照16个配置
    s32Ret = RK_MPI_VENC_GetStream(u32Ch, &stFrame, -1);
    if (s32Ret >= 0) {
        s32StreamCnt++;
        for (RK_U32 i = 0; i < stFrame.u32PackCount; i++) {
            RK_LOGD("get chn(%d) stream(%d) packet(%d) eoi(%d) type(%d) offset(%d) lenth(%d)",
                    u32Ch, s32StreamCnt, i,
                    stFrame.pstPack[i].bFrameEnd,
                    stFrame.pstPack[i].DataTp,
                    stFrame.pstPack[i].u32Offset,
                    stFrame.pstPack[i].u32Len);
            if (pstCtx->dstFilePath != RK_NULL) {
                pData = (char *)RK_MPI_MB_Handle2VirAddr(stFrame.pstPack[i].pMbBlk);
                fwrite(pData + stFrame.pstPack[i].u32Offset, 1, stFrame.pstPack[i].u32Len, fp);
                fflush(fp);
            }
        }
        RK_MPI_VENC_ReleaseStream(u32Ch, &stFrame);
        if (stFrame.pstPack->bStreamEnd == RK_TRUE) {
            RK_LOGI("chn %d reach EOS stream", u32Ch);
            break;
        }
    } else {
        if (pstCtx->threadExit) {
            break;
        }
        usleep(1000llu);
    }
}

```

4.14 RK_MPI_VENC_ReleaseStream

【描述】

释放码流缓存。

【语法】

```
RK_S32 RK_MPI_VENC_ReleaseStream(VENC_CHN VeChn, VENC_STREAM_S *pstStream);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstStream	码流结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

无

4.15 RK_MPI_VENC_SetJpegParam

【描述】

设置 JPEG 协议编码通道的高级参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetJpegParam(VENC_CHN VeChn, const VENC\_JPEG\_PARAM\_S *pstJpegParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstJpegParam	JPEG 协议编码通道的高级参数集合。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

无

4.16 RK_MPI_VENC_GetJpegParam

【描述】

获取 JPEG 协议编码通道的高级参数。

【语法】

```
RK_S32 RK_MPI_VENC_GetJpegParam(VENC_CHN VeChn, VENC\_JPEG\_PARAM\_S *pstJpegParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstJpegParam	JPEG 协议编码通道的高级参数集合。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

无

4.17 RK_MPI_VENC_GetRcParam

【描述】

获取通道码率控制高级参数。

【语法】

```
RK_S32 RK_MPI_VENC_GetRcParam(VENC_CHN VeChn, VENC\_RC\_PARAM\_S* pstRcParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstRcParam	通道码率控制参数指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

无

4.18 RK_MPI_VENC_SetRcParam

【描述】

设置编码通道码率控制器的高级参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetRcParam(VENC_CHN VeChn, const VENC\_RC\_PARAM\_S*pstRcParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstRcParam	编码通道码率控制器的高级参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

无

4.19 RK_MPI_VENC_GetRcParam2

【描述】

获取通道码率控制高级参数2。

【语法】

```
RK_S32 RK_MPI_VENC_GetRcParam2(VENC_CHN VeChn, VENC\_RC\_PARAM2\_S *pstRcParam2);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstRcParam2	通道码率控制参数指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

无

4.20 RK_MPI_VENC_SetRcParam2

【描述】

设置编码通道码率控制器的高级参数2。

【语法】

```
RK_S32 RK_MPI_VENC_SetRcParam2(VENC_CHN VeChn, const VENC\_RC\_PARAM2\_S *pstRcParam2);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstRcParam2	编码通道码率控制器的高级参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

无

4.21 RK_MPI_VENC_RequestIDR

【描述】

请求 IDR 帧。

【语法】

```
RK_S32 RK_MPI_VENC_RequestIDR(VENC_CHN VeChn, RK_BOOL bInstant);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
bInstant	是否使能立即编码 IDR 帧。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- bInstant暂时只支持RK_FALSE, 设置RK_TRUE会返回不支持[VENC错误码](#)RK_ERR_VENC_NOT_SUPPORT。

4.22 RK_MPI_VENC_GetRoiAttr

【描述】

获取 H.264/H.265 通道的 ROI 配置高级属性。

【语法】

```
RK_S32 RK_MPI_VENC_GetRoiAttr(VENC_CHN VeChn, VENC\_ROI\_ATTR\_S *pstRoiAttr, RK_S32 roi_index);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstRoiAttr	对应 ROI 区域的配置。	输出
u32Index	H.264/H.265 协议编码通道 ROI 区域索引。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

无

4.23 RK_MPI_VENC_SetRoiAttr

【描述】

设置 H.264/H.265 通道的 ROI 配置高级属性。

【语法】

```
RK_S32 RK_MPI_VENC_SetRoiAttr(VENC_CHN VeChn, const VENC\_ROI\_ATTR\_S *pstRoiAttr, RK_S32 roi_index);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstRoiAttr	ROI 区域参数。	输入
u32Index	H.264/H.265 协议编码通道 ROI 区域索引。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

无

4.24 RK_MPI_VENC_GetFd

【描述】

获取编码通道对应的设备文件句柄。通过此接口返回值可以使用select/poll查询对应通道的数据状态。

【语法】

```
RK_S32 RK_MPI_VENC_GetFd(VENC_CHN VeChn);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功，编码通道的设备文件句柄。
非0	失败，请参见 VENC错误码 。

【注意】

- 关闭文件句柄需要调用[RK_MPI_VENC_CloseFd](#)。
- 建议在创建编码器后，使能编码器[RK_MPI_VENC_StartRecvFrame](#)前进行调用。

4.25 RK_MPI_VENC_CloseFd

【描述】

关闭编码通道对应的设备文件句柄。

【语法】

```
RK_S32 RK_MPI_VENC_CloseFd(VENC_CHN VeChn);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功
负数	失败，请参见 VENC错误码 。

【注意】

无

4.26 RK_MPI_VENC_SetChnParam

【描述】

设置通道参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetChnParam(VENC_CHN VeChn, const VENC\_CHN\_PARAM\_S *pstChnParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstChnParam	Venc 的通道参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建，则返回失败。

4.27 RK_MPI_VENC_GetChnParam

【描述】

获取通道参数。

【语法】

```
RK_S32 RK_MPI_VENC_GetChnParam(VENC_CHN VeChn, VENC_CHN_PARAM_S *pstChnParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstChnParam	Venc 的通道参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建，则返回失败。

4.28 RK_MPI_VENC_InsertUserData

【描述】

插入用户数据。

【语法】

```
RK_S32 RK_S32 RK_MPI_VENC_InsertUserData(VENC_CHN VeChn, RK_U8 *pu8Data, RK_U32 u32Len);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pu8Data	用户数据指针。	输入
u32Len	用户数据长度。 取值范围：(0, 1024]，以 byte 为单位。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建，则返回失败。
- 如果 pu8Data 为空，则返回失败。
- 插入用户数据，只支持 H.264/H.265 和 MJPEG/JPEG 编码协议。
- H.264/H.265 协议通道最多同时分配 4 块内存空间用于缓存用户数据，且每段用户数据大小不超过 1k byte。如果用户插入的数据多于 4 块，或插入的一段用户数据大于 1k byte 时，此接口会返回错误。每段用户数据以 SEI 包的形式被插入到最新的图像码流包之前。在某段用户数据包被编码发送之后，H.264/H.265 通道内缓存这段用户数据的内存空间被清零，用于存放新的用户数据。
- JPEG/MJPEG 协议通道最多同时分配 4 块内存空间用于缓存用户数据，且每段用户数据大小不超过 1k byte。如果用户插入的数据多于 4 块，或插入的一段用户数据大于 1k byte 时，此接口会返回错误。用户数据以 APPsegment (0xFFE7) 形式添加到图像码流中。在用户数据被编码发送之后，JPEG/MJPEG 通道内缓存这段用户数据的内存空间被清零，用于存放新的用户数据。

4.29 RK_MPI_VENC_SetRcAdvParam

【描述】

设置RC模块的高级参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetRcAdvParam(VENC_CHN VeChn, const VENC\_RC\_ADVPARAM\_S *pstRcAdvParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstRcAdvParam	RC高级参数，此接口会包含一些与码率控制算法相关性较小的高级参数，并且未来可能会扩展。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建，则返回失败。
- 本接口用于设置 RC 模块的高级参数。
- u32ClearStatAfterSetAttr：设置新的通道码率后，是否清除码率控制的统计信息，默认为 1。目前只支持清除码率控制统计信息。

4.30 RK_MPI_VENC_GetRcAdvParam

【描述】

获取RC模块的高级参数。

【语法】

```
RK_S32 RK_MPI_VENC_GetRcAdvParam(VENC_CHN VeChn, VENC\_RC\_ADVPARAM\_S *pstRcAdvParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstRcAdvParam	RC模块的高级参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建，则返回失败。

4.31 RK_MPI_VENC_SetSuperFrameStrategy

【描述】

设置编码超大帧配置。

【语法】

```
RK_S32 RK_MPI_VENC_SetSuperFrameStrategy(VENC_CHN VeChn, const VENC\_SUPERFRAME\_CFG\_S*pstSuperFrmParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstSuperFrmParam	编码超大帧配置参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建，则返回失败。
- 本接口属于高级接口，用户可以选择性调用，系统默认关闭此功能。

4.32 RK_MPI_VENC_GetSuperFrameStrategy

【描述】

获取通道参数。

【语法】

```
RK_S32 RK_MPI_VENC_GetSuperFrameStrategy(VENC_CHN VeChn, VENC\_SUPERFRAME\_CFG\_S*pstSuperFrmParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstSuperFrmParam	编码超大帧配置参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstSuperFrmParam为空，则返回失败。

4.33 RK_MPI_VENC_SetFrameLostStrategy

【描述】

设置编码通道瞬时码率超过阈值时丢帧策略。

【语法】

```
RK_S32 RK_MPI_VENC_SetFrameLostStrategy(VENC_CHN VeChn, const VENC\_FRAMELOST\_S *pstFrmLostParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstFrmLostParam	编码通道丢帧策略的参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建，则返回失败。
- pstFrmLostParam 主要由四个参数决定
 - bFrmLostOpen：丢帧开关
 - u32FrmLostBpsThr：丢帧阈值百分比
 - enFrmLostMode：丢帧策略模式
 - u32EncFrmGaps：最大允许连续丢帧帧数
- 本接口属于高级接口，用户可以选择性调用，系统有默认值，默认在瞬时码率超出阈值时为丢帧。
- 本接口提供瞬时码率超过阈值时两种处理方式：丢帧和编码pskip帧，Mjpeg只支持丢帧处理方式。

4.34 RK_MPI_VENC_GetFrameLostStrategy

【描述】

获取编码通道瞬时码率超过阈值时丢帧策略。

【语法】

```
RK_S32 RK_MPI_VENC_GetFrameLostStrategy(VENC_CHN VeChn, VENC\_FRAMELOST\_S *pstFrmLostParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstFrmLostParam	编码通道丢帧策略的参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstFrmLostParam为空，则返回失败。

4.35 RK_MPI_VENC_SetIntraRefresh

【描述】

设置 P 帧刷 lslice 的参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetIntraRefresh(VENC_CHN VeChn, const VENC_INTRA_REFRESH_S *pstIntraRefresh);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstIntraRefresh	设置 P 帧刷 lslice 的参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建，则返回失败。
- 本接口属于高级接口，用户可以选择性调用，系统默认关闭此功能。

4.36 RK_MPI_VENC_GetIntraRefresh

【描述】

获取 P 帧刷 lslice 的设置参数。

【语法】

```
RK_S32 RK_MPI_VENC_GetIntraRefresh(VENC_CHN VeChn, VENC_INTRA_REFRESH_S *pstIntraRefresh);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstIntraRefresh	刷 lslice 的参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstIntraRefresh为空，则返回失败。

4.37 RK_MPI_VENC_SetHierarchicalQp

【描述】

设置分层 qp 参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetHierarchicalQp(VENC_CHN VeChn, const VENC\_HIERARCHICAL\_QP\_S *pstHierarchicalQp);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstHierarchicalQp	分层 qp 参数。	输入

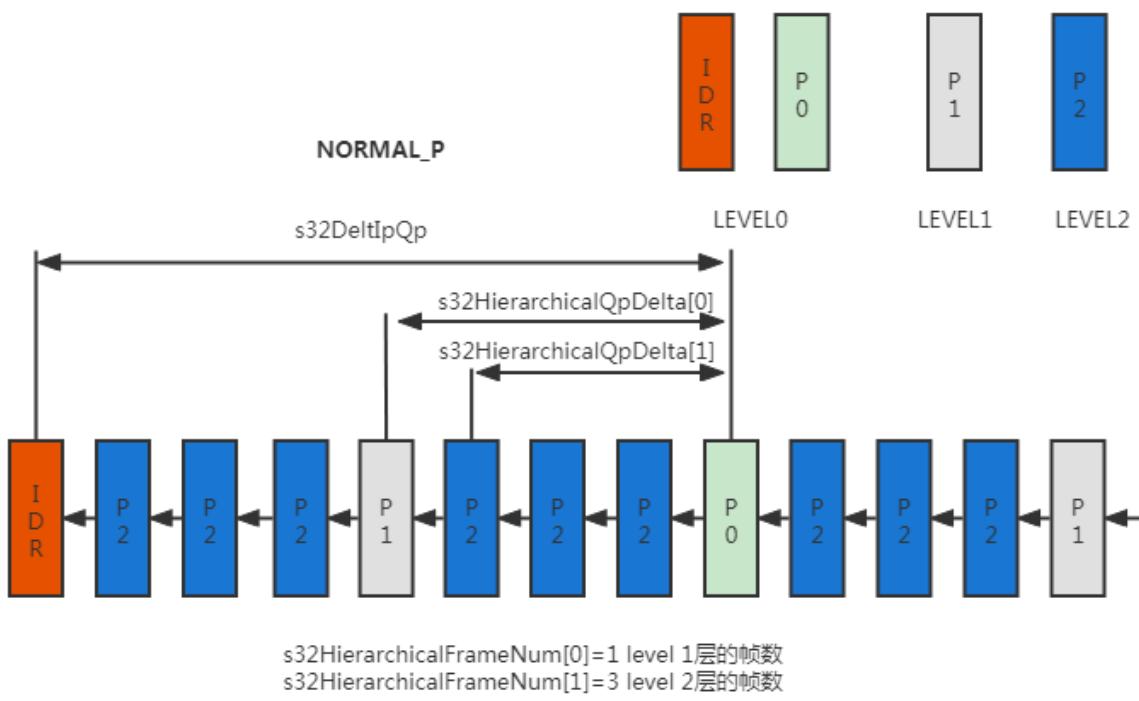
【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

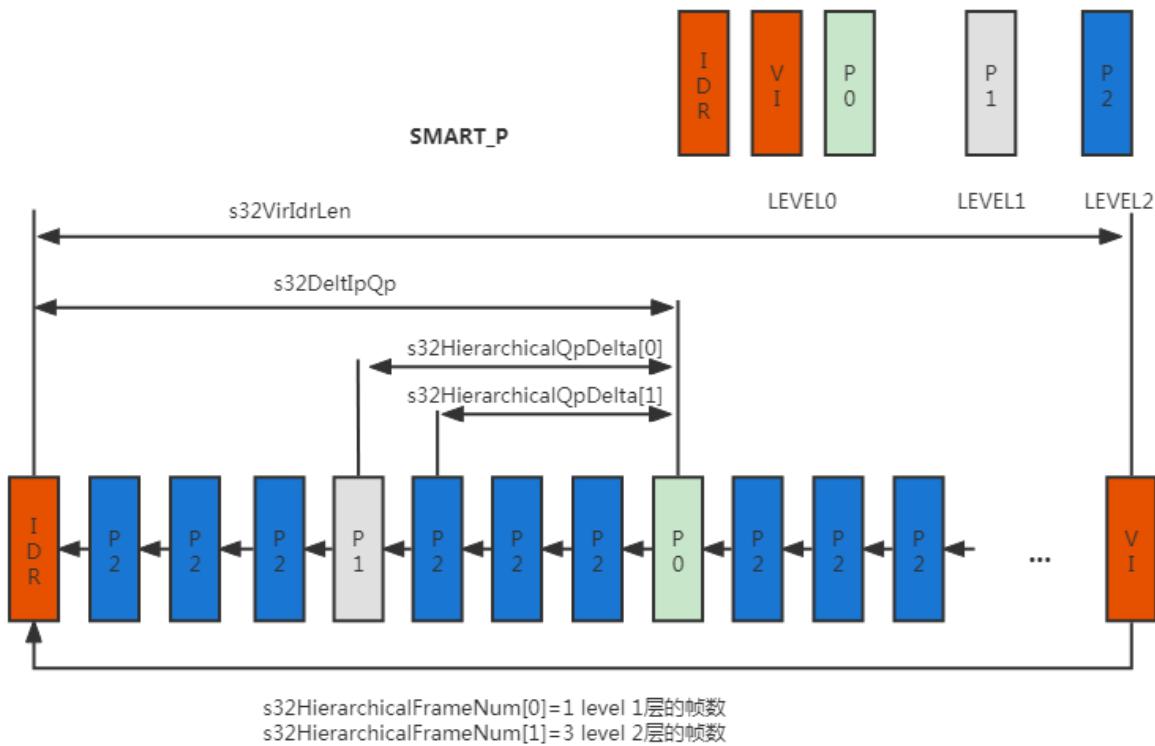
【注意】

- 如果通道未创建, 则返回失败。
- 本接口属于高级接口, 用户可以选择性调用, 系统默认关闭此功能。
- 仅 H.264/H.265 编码支持分层qp参数设置。
- 建议用户在调用此接口之前, 先调用RK_MPI_VENC_GetHierarchicalQp接口, 然后再进行设置。
- 该功能主要由如下两个参数决定。
 - bHierarchicalQpEn: 分层qp功能使能配置。
 - s32HierarchicalQpDelta: 设置第0层P帧的相对于每一层的Qp差值QpDelta。由于0层p帧的qp不需要进行调整, 故 s32HierarchicalQpDelta [0]为第1层, s32HierarchicalQpDelta [1]为第2层, 以此类推。
 - s32HierarchicalFrameNum: 设置每一层帧的数目, 其中s32HierarchicalFrameNum [0]为第1层, s32HierarchicalFrameNum [1]为第2层, 以此类推。
- 分层QP分为Normal P模式下分层QP与 Smart P模式下分层QP两种, 两者主要区别在于Smart P多出一个虚拟I帧, 其中虚拟I帧比其他P帧的优先级高, P0、P1、P2分别表示第0层P帧, 第1层P帧, 第2层P帧。分层QP示意图如下:

Normal P模式下分层QP示意图



Smart P模式下分层QP示意图



4.38 RK_MPI_VENC_GetHierarchicalQp

【描述】

获取分层 qp 参数。

【语法】

RK_S32 RK_MPI_VENC_GetHierarchicalQp(VENC_CHN VeChn, [VENC_HIERARCHICAL_QP_S](#) *pstHierarchicalQp);

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstHierarchicalQp	分层 qp 参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstHierarchicalQp为空, 则返回失败。
- 仅 H.264/H.265 用于获取分层qp参数。

4.39 RK_MPI_VENC_SetChnRotation

【描述】

设置通道旋转角度。

【语法】

```
RK_S32 RK_MPI_VENC_SetChnRotation(VENC_CHN VeChn, ROTATION\_E enRotation);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
enRotation	编码旋转角度。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建, 则返回失败。
- 编码为jpeg/mjpeg并且旋转为90/270的时候输入图像的宽高需要16对齐。

4.40 RK_MPI_VENC_GetChnRotation

【描述】

获取通道旋转角度。

【语法】

```
RK_S32 RK_MPI_VENC_GetChnRotation(VENC_CHN VeChn, ROTATION\_E *enRotation);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
enRotation	编码旋转角度。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者enRotation为空，则返回失败。

4.41 RK_MPI_VENC_AttachMbPool

【描述】

将编码通道绑定到某个视频缓存 MB 池中。

【语法】

RK_S32 RK_MPI_VENC_AttachMbPool(VENC_CHN VeChn, MB_POOL hMbPool)

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
hMbPool	视频缓存 MB 池信息。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 通道必须已创建，否则会返回通道不存在的错误RK_ERR_VENC_UNEXIST。
- 参数 hMbPool必须是已创建且有效的MB内存池，否则返回错误RK_ERR_VENC_ILLEGAL_PARAM。
- 用户通过调用接口RK_MPI_MB_CreatePool创建一个视频缓存MB池，再通过调用接口RK_MPI_VENC_AttachMbPool 将创建MB内存池绑定到该编码通道中。
- 不允许把同一个编码通道绑定到多个MB池中。
- 不允许多个编码通路绑定到同一个MB池中。
- 使用此接口可以优化启动编码通道的耗时。
- 接口需要在RK_MPI_VENC_StartRecvFrame函数前调用，编码过程中不支持动态调用。

4.42 RK_MPI_VENC_DetachMbPool

【描述】

将编码通道从某个视频缓存 MB 池中解绑定。

【语法】

RK_S32 RK_MPI_VENC_DetachMbPool(VENC_CHN VeChn)

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 通道必须已创建，否则会返回通道不存在的错误RK_ERR_VENC_UNEXIST。
- 需要在RK_MPI_VENC_StopRecvFrame函数后使用，编码过程中不支持动态调用。

4.43 RK_MPI_VENC_SetFilter

【描述】

设置输入源滤波参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetFilter(VENC_CHN VeChn, const VENC\_FILTER\_S *pstFilter);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstFilter	输入源滤波参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- ISP输入给编码器的yuv，清晰度调试参数是一致的，不同路的编码要求不一定相同，可以加上滤波自主适配；
- 在极低码率需求下，特别是小分辨率的主观效果问题更为明显，适当滤波减轻编码压力可以缓解效果问题；
- 运动场景比较复杂时，可以在运动时间段加上滤波以缓解编码压力。

4.44 RK_MPI_VENC_GetFilter

【描述】

获取输入源滤波参数。

【语法】

```
RK_S32 RK_MPI_VENC_GetFilter(VENC_CHN VeChn, VENC\_FILTER\_S *pstFilter);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstFilter	输入源滤波参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 无

4.45 RK_MPI_VENC_SetH264IntraPred

【描述】

设置H.264协议编码通道的帧内预测属性。

【语法】

```
RK_S32 RK_MPI_VENC_SetH264IntraPred(VENC_CHN VeChn, const VENC\_H264\_INTRA\_PRED\_S *pstH264IntraPred);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstH264IntraPred	H.264协议编码通道的帧内预测配置。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH264IntraPred为空, 则返回失败。
- 帧间预测的属性仅支持 constrained_intra_pred_flag, 具体的含义, 请参见 H.264 协议。
- 此接口属于高级接口, 用户可以选择性调用, 不建议调用, 系统会有默认值。系统默认 constrained_intra_pred_flag 为 0。
- 本接口在编码通道创建之后, 编码通道销毁之前设置。本接口在编码过程中被调用时, 会插入一帧I帧并生效。
- 建议用户在创建通道之后, 启动编码之前调用此接口, 减少在编码过程中调用的次数。
- 建议用户在调用此接口之前, 先调用RK_MPI_VENC_GetH264IntraPred接口, 获取当前编码通道的IntraPred 配置, 然后再进行设置。

4.46 RK_MPI_VENC_GetH264IntraPred

【描述】

获取H.264协议编码通道的帧内预测属性。

【语法】

```
RK_S32 RK_MPI_VENC_GetH264IntraPred(VENC_CHN VeChn, VENC\_H264\_INTRA\_PRED\_S *pstH264IntraPred);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstH264IntraPred	H.264协议编码通道的帧内预测配置。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH264IntraPred为空，则返回失败。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

4.47 RK_MPI_VENC_SetH264Trans

【描述】

设置H.264协议编码通道的变换、量化属性。

【语法】

```
RK_S32 RK_MPI_VENC_SetH264Trans(VENC_CHN VeChn, const VENC\_H264\_TRANS\_S *pstH264Trans);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstH264Trans	H.264协议编码通道的变换、量化属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH264Trans为空，则返回失败。
- 变换、量化属性主要由五个参数组成：
 - u32TransMode：帧内/帧间预测宏块的变换属性。u32TransMode= 0表示支持对帧内/帧间预测宏块支持 4x4 变换和 8x8 变换；u32TransMode= 1表示只支持对帧内/帧间预测宏块支持 4x4 变换。
 - bScalingListValid：表示 InterScalingList8x8、IntraScalingList8x8 量化表是否有效。不支持设置 bScalingListValid 为 true。保留，暂时没有使用。
 - InterScalingList8x8：对帧间预测宏块进行 8x8 变换时，可由用户通过此数组提供量化表，保留，暂时没有使用。
 - IntraScalingList8x8：对帧内预测宏块进行 8x8 变换时，可由用户通过此数组提供量化表，保留，暂时没有使用。
 - chroma_qp_index_offset：默认值-6，具体含义请参见 H.264 协议。
- 本接口在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，会插入一帧并生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用RK_MPI_VENC_GetH264Trans接口，获取当前编码通道的trans配置，然后再进行设置。

4.48 RK_MPI_VENC_GetH264Trans

【描述】

获取H.264协议编码通道的变换、量化属性。

【语法】

```
RK_S32 RK_MPI_VENC_GetH264Trans(VENC_CHN VeChn, VENC\_H264\_TRANS\_S *pstH264Trans);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstH264Trans	H.264协议编码通道的变换、量化属性。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH264Trans为空，则返回失败。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

4.49 RK_MPI_VENC_SetH264Entropy

【描述】

设置H.264协议编码通道的熵编码模式。

【语法】

```
RK_S32 RK_MPI_VENC_SetH264Entropy(VENC_CHN VeChn, const VENC\_H264\_ENTROPY\_S *pstH264EntropyEnc);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstH264EntropyEnc	H.264协议编码通道的熵编码模式。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建pstH264EntropyEnc为空，则返回失败。
- 变换、量化属性主要由两个参数组成：
 - u32EntropyEncMode：熵编码方式，u32EntropyEncMode=0表示用 cavlc 编码，u32EntropyEncMode=1表示使用 cabac 编码方式。默认为1，即cabac编码方式。
 - Cabac_init_idc：cabac 初始化索引，系统默认为 0。具体含义请参见 H.264 协议。

- 本接口在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，会插入一帧I帧并生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用RK_MPI_VENC_GetH264Entropy接口，获取当前编码通道的entropy配置，然后再进行设置。

4.50 RK_MPI_VENC_GetH264Entropy

【描述】

获取H.264协议编码通道的熵编码模式。

【语法】

```
RK_S32 RK_MPI_VENC_GetH264Entropy(VENC_CHN VeChn, VENC\_H264\_ENTROPY\_S *pstH264EntropyEnc);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstH264EntropyEnc	H.264协议编码通道的熵编码模式。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH264EntropyEnc为空，则返回失败。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

4.51 RK_MPI_VENC_SetH264Dbblk

【描述】

设置H.264协议编码通道的 Deblocking 类型。

【语法】

```
RK_S32 RK_MPI_VENC_SetH264Dbblk(VENC_CHN VeChn, const VENC\_H264\_DBLK\_S *pstH264Dbblk);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstH264Dbblk	H.264协议编码通道的Deblocking类型。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建pstH264Dblk为空，则返回失败。
- 变换、量化属性主要由三个参数组成：
 - disable_deblocking_filter_idc：具体含义请参见 H.264 协议。
 - slice_alpha_c0_offset_div2：具体含义请参见 H.264 协议。
 - slice_beta_offset_div2：具体含义请参见 H.264 协议。
- 系统默认打开 deblocking 功能，默认 disable_deblocking_filter_idc = 0, slice_alpha_c0_offset_div2 = 0, slice_beta_offset_div2 = 0。
- 如果用户想关闭 deblocking 功能，可以将 disable_deblocking_filter_idc 置为 1。
- 本接口在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，会插入一帧I帧并生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用RK_MPI_VENC_GetH264Dblk接口，获取当前编码通道的dblk配置，然后再进行设置。

4.52 RK_MPI_VENC_GetH264Dblk

【描述】

获取H.264协议编码通道的 Deblocking 类型。

【语法】

```
RK_S32 RK_MPI_VENC_GetH264Dblk(VENC_CHN VeChn, VENC\_H264\_DBLK\_S*pstH264Dblk);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstH264Dblk	H.264协议编码通道的Deblocking类型。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH264Dblk为空，则返回失败。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

4.53 RK_MPI_VENC_SetH264Qbias

【描述】

设置H.264协议编码通道的量化偏移值。

【语法】

```
RK_S32 RK_MPI_VENC_SetH264Qbias(VENC_CHN VeChn, const VENC\_H264\_QBIAS\_S*pstQbias);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstQbias	H.264协议编码通道的量化偏移值。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 无

4.54 RK_MPI_VENC_GetH264Qbias

【描述】

获取H.264协议编码通道的量化偏移值。

【语法】

```
RK_S32 RK_MPI_VENC_GetH264Qbias(VENC_CHN VeChn, VENC\_H264\_QBIAS\_S *pstQbias);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstQbias	H.264协议编码通道的量化偏移值。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 无

4.55 RK_MPI_VENC_SetH264Vui

【描述】

设置H.264协议编码通道的 Vui 参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetH264Vui(VENC_CHN VeChn, const VENC\_H264\_VUI\_S*pstH264Vui);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstH264Vui	H.264协议编码通道的Vui参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH264Vui为空，则返回失败。
- 本接口在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，会插入一帧I帧并生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

4.56 RK_MPI_VENC_GetH264Vui

【描述】

获取H.264协议编码通道的Vui参数。

【语法】

```
RK_S32 RK_MPI_VENC_GetH264Vui(VENC_CHN VeChn, VENC\_H264\_VUI\_S *pstH264Vui);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstH264Vui	H.264协议编码通道的Vui参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH264Vui为空，则返回失败。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

4.57 RK_MPI_VENC_SetH265Trans

【描述】

设置H.265通道的变换量化属性。

【语法】

```
RK_S32 RK_MPI_VENC_SetH265Trans(VENC_CHN VeChn, const VENC\_H265\_TRANS\_S *pstH265Trans);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstH265Trans	H.265通道的变换量化属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH265Trans为空, 则返回失败。
- 变换、量化属性主要由以下参数组成:
 - cb_qp_offset: 默认值-6, 具体的含义, 请参见 H.265 协议关于 pps_cb_qp_offset 的解释。
 - cr_qp_offset: 默认值-6, 具体的含义, 请参见 H.265 协议关于 pps_cr_qp_offset 的解释。
 - bScalingListEnabled: 表示量化表是否有效, 无量化表时置为 0。保留, 暂时没有使用。
 - bScalingListTu4Valid: 表示 InterScalingList4X4、IntraScalingList4X4 量化表是否有效, 使用协议默认量化表时置为 0。保留, 暂时没有使用。
 - InterScalingList4X4[2][16] : InterScalingList4X4[0][16] 表示帧间亮度量化表, InterScalingList4X4[1][16] 表示帧间色度量化表, 可由用户通过此数组提供量化表。保留, 暂时没有使用。
 - IntraScalingList4X4[2][16] : IntraScalingList4X4[0][16] 表示帧内亮度量化表, IntraScalingList4X4[1][16] 表示帧内色度量化表, 可由用户通过此数组提供量化表。
 - IbScalingListTu8Valid: 表示 InterScalingList8X8、IntraScalingList8X8 量化表是否有效, 使用协议默认量化表时置为 0。保留, 暂时没有使用。
 - InterScalingList8X8[2][64] : InterScalingList8X8[0][64] 表示帧间亮度量化表, InterScalingList8X8[1][64] 表示帧间色度量化表, 可由用户通过此数组提供量化表。保留, 暂时没有使用。
 - IntraScalingList8X8[2][64] : IntraScalingList8X8[0][64] 表示帧内亮度量化表, IntraScalingList8X8[1][64] 表示帧内色度量化表, 可由用户通过此数组提供量化表。保留, 暂时没有使用。
 - bScalingListTu16Valid: 表示 InterScalingList16X16、IntraScalingList16X16 量化表是否有效, 使用协议默认量化表时置为 0。保留, 暂时没有使用。保留, 暂时没有使用。
 - InterScalingList16X16[2][64] : InterScalingList16X16[0][64] 表示帧间亮度量化表, InterScalingList16X16[1][64] 表示帧间色度量化表, 可由用户通过此数组提供量化表。保留, 暂时没有使用。
 - IntraScalingList16X16[2][64] : IntraScalingList16X16[0][64] 表示帧内亮度量化表, IntraScalingList16X16[1][64] 表示帧内色度量化表, 可由用户通过此数组提供量化表。保留, 暂时没有使用。
 - bScalingListTu32Valid: 表示 InterScalingList32X32、IntraScalingList32X32 量化表是否有效, 使用协议默认量化表时置为 0。保留, 暂时没有使用。
 - InterScalingList32X32 [64]: InterScalingList32X32 [64] 表示帧间亮度量化表, 可由用户通过此数组提供量化表。保留, 暂时没有使用。
 - IntraScalingList32X32 [64]: IntraScalingList32X32 [64] 表示帧内亮度量化表, 可由用户通过此数组提供量化表。保留, 暂时没有使用。
- 本接口在编码通道创建之后, 编码通道销毁之前设置。本接口在编码过程中被调用时, 会插入一帧I帧并生效。
- 建议用户在创建通道之后, 启动编码之前调用此接口, 减少在编码过程中调用的次数。
- 建议用户在调用此接口之前, 先调用RK_MPI_VENC_GetH265Trans接口, 获取当前编码通道的trans配置, 然后再进行设置。

4.58 RK_MPI_VENC_GetH265Trans

【描述】

获取H.265通道的变换量化属性。

【语法】

```
RK_S32 RK_MPI_VENC_GetH265Trans(VENC_CHN VeChn, VENC_H265_TRANS_S *pstH265Trans);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstH265Trans	H.265通道的变换量化属性。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH265Trans为空, 则返回失败。
- 本接口在编码通道创建之后, 编码通道销毁之前调用。
- 建议用户在创建通道之后, 启动编码之前调用此接口, 减少在编码过程中调用的次数。

4.59 RK_MPI_VENC_SetH265Entropy

【描述】

设置H.265通道的熵编码属性。

【语法】

```
RK_S32 RK_MPI_VENC_SetH265Entropy(VENC_CHN VeChn, const VENC\_H265\_ENTROPY\_S *pstH265Entropy);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstH265Entropy	H.265通道的熵编码属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH265Entropy为空, 则返回失败。
- 熵编码属性主要由一个参数决定
cabac_init_flag: 具体的含义, 请参见 H.265 协议。保留, 暂时没有使用。
- 本接口在编码通道创建之后, 编码通道销毁之前设置。本接口在编码过程中被调用时, 会插入一帧I帧并生效。
- 建议用户在创建通道之后, 启动编码之前调用此接口, 减少在编码过程中调用的次数。
- 建议用户在调用此接口之前, 先调用RK_MPI_VENC_GetH265Entropy接口, 获取当前编码通道的entropy配置, 然后再进行设置。

4.60 RK_MPI_VENC_GetH265Entropy

【描述】

获取H.265通道的熵编码属性。

【语法】

```
RK_S32 RK_MPI_VENC_GetH265Entropy(VENC_CHN VeChn, VENC\_H265\_ENTROPY\_S *pstH265Entropy);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstH265Entropy	H.265通道的熵编码属性。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH265Entropy为空，则返回失败。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

4.61 RK_MPI_VENC_SetH265Dbblk

【描述】

设置H.265通道的Deblocking属性。

【语法】

```
RK_S32 RK_MPI_VENC_SetH265Dbblk(VENC_CHN VeChn, const VENC\_H265\_DBLK\_S *pstH265Dbblk);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstH265Dbblk	H.265通道的Deblocking属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH265Dbblk为空，则返回失败。
- Deblocking 属性主要由三个参数决定
 - slice_deblocking_filter_disabled_flag：具体的含义，请参见 H.265 协议。
 - slice_beta_offset_div2：具体的含义，请参见 H.265 协议。
 - slice_tc_offset_div2：具体的含义，请参见 H.265 协议。
- 本接口属于高级接口，用户可以选择性调用，建议不调用，系统默认打开deblocking 功能， 默认 slice_deblocking_filter_disabled_flag=0, slice_tc_offset_div2=0, slice_beta_offset_div2=0。
- 如果用户想关闭 deblocking 功能，可以将 slice_deblocking_filter_disabled_flag 置为1。
- 本接口在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，会插入一帧I帧并生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用RK_MPI_VENC_GetH265Dbblk接口，获取当前编码通道的dblk配置，然后再进行设置。

4.62 RK_MPI_VENC_GetH265Dbblk

【描述】

获取H.265通道的Deblocking属性。

【语法】

```
RK_S32 RK_MPI_VENC_GetH265Dbblk(VENC_CHN VeChn, VENC\_H265\_DBLK\_S*pstH265Dbblk);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstH265Dbblk	H.265通道的Deblocking属性。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH265Dbblk为空，则返回失败。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

4.63 RK_MPI_VENC_SetH265Sao

【描述】

设置H.265通道的Sao属性。

【语法】

```
RK_S32 RK_MPI_VENC_SetH265Sao(VENC_CHN VeChn, const VENC\_H265\_SAO\_S*pstH265Sao);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstH265Sao	H.265协议编码通道的Sao配置。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH265Sao为空，则返回失败。
- Sao 属性主要由两个参数决定
 - slice_sao_luma_flag：当前 slice 亮度分量是否作 Sao 滤波。
 - slice_sao_chroma_flag：当前 slice 色度分量是否作 Sao 滤波。

- 本接口属于高级接口，用户可以选择性调用，建议不调用，系统默认打开 sao 功能，默认 slice_sao_luma_flag = 1, slice_sao_chroma_flag = 1。
- 本接口在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，会插入一帧I帧并生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用RK_MPI_VENC_GetH265Sao接口，获取当前编码通道的sao配置，然后再进行设置。

4.64 RK_MPI_VENC_GetH265Sao

【描述】

获取H.265通道的Sao属性。

【语法】

```
RK_S32 RK_MPI_VENC_GetH265Sao(VENC_CHN VeChn, VENC\_H265\_SAO\_S *pstH265Sao);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstH265Sao	H.265协议编码通道的Sao配置。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH265Sao为空，则返回失败。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

4.65 RK_MPI_VENC_SetH265PredUnit

【描述】

设置H.265通道的PU属性。

【语法】

```
RK_S32 RK_MPI_VENC_SetH265PredUnit(VENC_CHN VeChn, const VENC\_H265\_PU\_S *pstPredUnit);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstPredUnit	H.265通道的PU配置。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstPredUnit为空, 则返回失败。
- PU属性主要由两个参数决定
 - constrained_intra_pred_flag: 具体的含义, 请参见 H.265 协议。保留, 暂时没有使用。
 - strong_intra_smoothing_enabled_flag: 具体的含义, 请参见 H.265 协议。
- 本接口属于高级接口, 用户可以选择性调用, 建议不调用, 默认 constrained_intra_pred_flag=0, strong_intra_smoothing_enabled_flag=1。
- 本接口在编码通道创建之后, 编码通道销毁之前设置。本接口在编码过程中被调用时, 会插入一帧I帧并生效。
- 建议用户在创建通道之后, 启动编码之前调用此接口, 减少在编码过程中调用的次数。
- 建议用户在调用此接口之前, 先调用RK_MPI_VENC_GetH265PredUnit接口, 获取当前编码通道的pu配置, 然后再进行设置。

4.66 RK_MPI_VENC_GetH265PredUnit

【描述】

获取H.265通道的PU属性。

【语法】

```
RK_S32 RK_MPI_VENC_GetH265PredUnit(VENC_CHN VeChn, VENC\_H265\_PU\_S *pstPredUnit);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstPredUnit	H.265通道的PU配置。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstPredUnit为空, 则返回失败。
- 本接口在编码通道创建之后, 编码通道销毁之前调用。
- 建议用户在创建通道之后, 启动编码之前调用此接口, 减少在编码过程中调用的次数。

4.67 RK_MPI_VENC_SetH265Vui

【描述】

设置H.265协议编码通道的 Vui 参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetH265Vui(VENC_CHN VeChn, const VENC\_H265\_VUI\_S *pstH265Vui);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstH265Vui	H.265协议编码通道的Vui参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH265Vui为空，则返回失败。
- 本接口在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，会插入一帧I帧并生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

4.68 RK_MPI_VENC_GetH265Vui

【描述】

获取H.265协议编码通道的Vui参数。

【语法】

```
RK_S32 RK_MPI_VENC_GetH265Vui(VENC_CHN VeChn, VENC\_H265\_VUI\_S*pstH265Vui);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstH265Vui	H.265协议编码通道的Vui参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstH265Vui为空，则返回失败。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

4.69 RK_MPI_VENC_SetRefParam

【描述】

设置H.264/H.265编码通道高级跳帧参考参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetRefParam(VENC_CHN VeChn, const VENC\_REF\_PARAM\_S*pstRefParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstRefParam	H.264/H.265编码通道高级跳帧参考参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstRefParam为空，则返回失败。
- 本接口在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，会插入一帧I帧并生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

4.70 RK_MPI_VENC_GetRefParam

【描述】

获取H.264/H.265编码通道高级跳帧参考参数。

【语法】

```
RK_S32 RK_MPI_VENC_GetRefParam(VENC_CHN VeChn, VENC\_REF\_PARAM\_S*pstRefParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstRefParam	H.264/H.265编码通道高级跳帧参考参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstRefParam为空，则返回失败。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

4.71 RK_MPI_VENC_SetH265Qbias

【描述】

设置H.265协议编码通道的量化偏移值。

【语法】

```
RK_S32 RK_MPI_VENC_SetH265Qbias(VENC_CHN VeChn, const VENC\_H265\_QBIAS\_S*pstQbias);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstQbias	H.265协议编码通道的量化偏移值。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 无

4.72 RK_MPI_VENC_GetH265Qbias

【描述】

获取H.265协议编码通道的量化偏移值。

【语法】

```
RK_S32 RK_MPI_VENC_GetH265Qbias(VENC_CHN VeChn, VENC\_H265\_QBIAS\_S *pstQbias);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstQbias	H.265协议编码通道的量化偏移值。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 无

4.73 RK_MPI_VENC_SetMjpegParam

【描述】

设置 MJPEG 协议编码通道的高级参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetMjpegParam(VENC_CHN VeChn, VENC\_MJPEG\_PARAM\_S *pstMjpegParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstMjpegParam	MJPEG 协议编码通道的高级参数集合。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstMjpegParam为空, 则返回失败。
- 本接口在编码通道创建之后, 编码通道销毁之前调用。
- 建议用户在创建通道之后, 启动编码之前调用此接口, 减少在编码过程中调用的次数。
- 此参数优先于q_factor设置, 设置此接口后q_factor参数会失效。
- 高级参数主要由四个参数组成
 - U8YQt[64], u8CbQt[64], u8CrQt[64]: 对应三个量化表空间, 用户可以通过这三个参数设置用户的量化表。
 - u32MCUPerECS: 每个Ecs中包含多少Mcu。系统模式 u32MCUPerECS = 0, 表示当前帧的所有MCU被编码为一个ECS。u32MCUPerECS的最小值为0, 最大值不超过(picwidth+15)>>4 (picheight+15)>>4 (yuv420sp)、(picwidth+15)>>4 (picheight+15)>>4 2(yuv422sp)、(picwidth+15)>>4 (picheight+15)>>4 * 4 (yuv444sp)。

4.74 RK_MPI_VENC_GetMjpegParam

【描述】

获取 MJPEG 协议编码通道的高级参数配置。

【语法】

```
RK_S32 RK_MPI_VENC_GetMjpegParam(VENC_CHN VeChn, VENC\_MJPEG\_PARAM\_S *pstMjpegParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstMjpegParam	MJPEG 协议编码通道的高级参数配置。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstMjpegParam为空, 则返回失败。
- 本接口在编码通道创建之后, 编码通道销毁之前调用。
- 建议用户在创建通道之后, 启动编码之前调用此接口, 减少在编码过程中调用的次数。

4.75 RK_MPI_VENC_SetQpmap

【描述】

设置Qpmap MB_BLK。

【语法】

```
RK_S32 RK_MPI_VENC_SetQpmap(VENC_CHN VeChn, const MB_BLK blk);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
blk	Qpmap的MB_BLK。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- QPMAP优先级高于ROI，同时配置QPMAP和ROI，只有QPMAP生效。
- RK_MPI_VENC_SetQpmap优先级低于RK_MPI_VENC_SendFrameEx。
- QPMAP模式下允许用户自由决定码控的策略。用户需要申请内存，内存会映射到编码的图像上，通过内存的赋值完成编码图像的每一块qp的配置。Qp的配置是以16x16块为单位，每一个16x16块的Qp值，采用用户设定的相应块的Qp值。所有这些块的Qp值组成Qp表，该表中的Qp值的组织方式如下所示：
- H.264 QPMAP MB排放位置

H.264 QPMAP MB(16 * 16) 排放位置：

举例：图像宽：916像素，图像高：416像素

0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26
27	28	29	30	31	32	33	34	35

H.264 每一个MB QPMAP值：每一个MB为16bit，其中：

- [15] : 绝对qp 标志位， 1: 绝对qp； 0: 相对qp；
 [14:8] : qp值， 绝对qp: [0,51]； 相对qp: [-32,31]；
 [7:0] : 必须配置为0x80。

上表MB QPMAP值在内存的排放顺序(一个值占2byte，小端模式)为：

0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26
27	28	29	30	31	32	33	34	35

上表QPMAP值需要的内存为：942 = 72 byte。

- H.265 QPMAP MB排放位置(RV1109,RV1126,RK3568,RK3588)

H.265 QPMAP MB(16x16) LCU (64x64) 排放位置：

举例：图像宽：916像素，图像高：516像素

0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26
27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44

H.265 每一个MB QPMAP值：每一个MB为16bit，其中：

[15] : 绝对qp 标志位， 1: 绝对qp; 0: 相对qp;

[14:8] : qp值， 绝对qp: [0,51]; 相对qp: [-32,31];

[7:0] : 必须配置为0x80。

上表MB QPMAP值在内存的排放顺序(一个值占2byte, 小端模式)为：

0	1	2	3	9	10	11	12	18	19	20	21	27	28	29	30
4	5	6	7	13	14	15	16	22	23	24	25	31	32	33	34
8	x	x	x	17	x	x	x	26	x	x	x	35	x	x	x
36	37	38	39	x	x	x	x	x	x	x	x	x	x	x	x
40	41	42	43	x	x	x	x	x	x	x	x	x	x	x	x
44	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

QPMAP值需要4对齐，x代表任意值。上表QPMAP值需要的内存为： $16 \times 2 = 192$ byte。

- H.265 QPMAP MB排放位置(RV1106)

H.265 QPMAP MB(16 16) LCU (32 32) 排放位置：

举例： 图像宽： 9 16像素， 图像高： 5 16像素

0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26
27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44

H.265 每一个MB QPMAP值：每一个MB为16bit，其中：

[15] : 绝对qp 标志位， 1: 绝对qp; 0: 相对qp;

[14:8] : qp值， 绝对qp: [0,51]; 相对qp: [-32,31];

[7:0] : 必须配置为0x80。

上表MB QPMAP值在内存的排放顺序(一个值占2byte, 小端模式)为：

0	1	9	10	2	3	11	12	4	5	13	14
6	7	15	16	8	x	17	x	18	19	27	28
20	21	29	30	22	23	31	32	24	25	33	34
26	x	35	x	36	37	x	x	38	39	x	x
40	41	x	x	42	43	x	x	44	x	x	x

QPMAP值需要2对齐，x代表任意值。上表QPMAP值需要的内存为： $12 \times 2 = 120$ byte。

4.76 RK_MPI_VENC_GetQpmap

【描述】

获取Qpmap MB_BLK。

【语法】

```
RK_S32 RK_MPI_VENC_GetQpmap(VENC_CHN VeChn, MB_BLK *pBlk);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pBlk	Qpmap的MB_BLK。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 无

4.77 RK_MPI_VENC_SetDeBreathEffect

【描述】

设置去除呼吸效应参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetDeBreathEffect(VENC_CHN VeChn, const VENC\_DEBREATHFFECT\_S *pstDeBreathEffect)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstDeBreathEffect	去除呼吸效应参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 目前仅RK3588, RV1106, RV1103 VI输入普通模式支持此功能。VI输入在线模式不支持此功能。使能参考帧共享属性不支持此功能。
- 仅H264/H265支持去呼吸效应功能。
- 必须保证通道已创建，否则会返回通道未创建的VENC错误码。
- 本接口在编码通道创建之后，编码通道销毁之前设置。此接口在编码过程中被调用时，等到下一个帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

- 当GOP等于1时，不允许使能去除呼吸效应功能。
- 如果开启P帧刷Islice，需关闭P帧刷Islice功能，再使能去除呼吸效应，否则返回不允许。
- 建议用户在调用此接口之前，先调用[RK_MPI_VENC_GetDeBreathEffect](#)接口，获取去除呼吸效应参数，然后再进行设置。
- 该功能主要由如下三个参数决定。
 - bEnable：去除呼吸效应使能控制。
 - s32Strength0为设置去呼吸效应强度调节参数0，未使用。
 - s32Strength1为设置去呼吸效应强度调节参数1，默认值为16。其值越大，I帧会越大，其值越小，I帧会越小。如果I帧变大对客户应用有影响，客户可以尝试把s32Strength1调小一点，会降低I帧码流的大小，但是可能会减弱呼吸效应的改善程度。

4.78 RK_MPI_VENC_GetDeBreathEffect

【描述】

获取去除呼吸效应参数。

【语法】

```
RK_S32 RK_MPI_VENC_GetDeBreathEffect(VENC_CHN VeChn, VENC\_DEBREATHFFECT\_S*pstDeBreathEffect)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstDeBreathEffect	去除呼吸效应参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 目前仅RK3588，RV1106，RV1103 VI输入普通模式支持此功能。VI输入在线模式不支持此功能。
- 仅H264/H265支持去呼吸效应功能。
- 本接口在编码通道创建之后，编码通道销毁之前调用。

4.79 RK_MPI_VENC_SetChnRefBufShareAttr

【描述】

设置编码通道参考帧共享属性。

【语法】

```
RK_S32 RK_MPI_VENC_SetChnRefBufShareAttr(VENC_CHN VeChn, const VENC\_CHN\_REF\_BUF\_SHARE\_S*pstVencChnRefBufShare)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM]。	输入
pstVencChnRefBufShare	参考帧共享属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 仅RV1106, RV1103支持。
- 仅H264, H265支持。
- 使能参考帧共享属性可以节省内存, 但是无法支持超大帧重编和去呼吸效应。

4.80 RK_MPI_VENC_GetChnRefBufShareAttr

【描述】

获取编码通道参考帧共享属性。

【语法】

```
RK_S32 RK_MPI_VENC_GetChnRefBufShareAttr(VENC_CHN VeChn, VENC\_CHN\_REF\_BUF\_SHARE\_S*pstVencChnRefBufShare);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstVencChnRefBufShare	参考帧共享属性指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 仅RV1106, RV1103支持。
- 仅H264, H265支持。

4.81 RK_MPI_VENC_SetComboAttr

【描述】

设置编码通道Combo属性。

【语法】

```
RK_S32 RK_MPI_VENC_SetComboAttr(VENC_CHN VeChn, VENC\_COMBO\_ATTR\_S*pstComboAttr)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstComboAttr	Combo属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 仅RV1106, RV1103支持。
- 仅MJPEG, JPEG支持。
- 编码通道设置Combo属性后, 不支持bind, 编码通道的输入源与Combo通道的输入源一致, 支持与H264、H265编码通道Combo, 两个Combo通道的分辨率一样。
- 编码通道设置Combo属性后, H264、H265编码通道切换分辨率后, 对应的MJPEG, JPEG通道分辨率也需要切换。
- 编码通道设置Combo属性后, H264、H265编码通道设置卷绕属性后, 对应的MJPEG, JPEG通道也需要设置卷绕属性。

4.82 RK_MPI_VENC_GetComboAttr

【描述】

获取编码通道Combo属性。

【语法】

```
RK_S32 RK_MPI_VENC_GetComboAttr(VENC_CHN VeChn, VENC\_COMBO\_ATTR\_S *pstComboAttr)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstComboAttr	Combo属性指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 仅RV1106, RV1103支持。
- 仅MJPEG, JPEG支持。

4.83 RK_MPI_VENC_SetChnBufWrapAttr

【描述】

设置编码通道Buf卷绕属性。

【语法】

```
RK_S32 RK_MPI_VENC_SetChnBufWrapAttr(VENC_CHN VeChn, const VENC\_CHN\_BUF\_WRAP\_S *pstVencChnBufWrap)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstVencChnBufWrap	Buf卷绕属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 仅RV1106, RV1103支持。
- 设置卷绕属性满足VI、VENC同时设置卷绕, VI需要bind VENC。
- 支持一路、两路卷绕。
- Buf卷绕行数、Buf卷绕大小配置需要和VI一致。
- 设置卷绕后不支持VENC旋转。
- 设置卷绕后不支持去呼吸效应。
- 设置卷绕后支持超大帧丢帧, 不支持超大帧重编。

4.84 RK_MPI_VENC_GetChnBufWrapAttr

【描述】

获取编码通道Buf卷绕属性。

【语法】

```
RK_S32 RK_MPI_VENC_GetChnBufWrapAttr(VENC_CHN VeChn, VENC\_CHN\_BUF\_WRAP\_S*pstVencChnBufWrap)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstVencChnBufWrap	Buf卷绕属性指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 仅RV1106, RV1103支持。

4.85 RK_MPI_VENC_EnableSvc

【描述】

使能智能编码。 Svc: Smart video coding。

【语法】

```
RK_S32 RK_MPI_VENC_EnableSvc(VENC_CHN VeChn, RK_BOOL bEnable)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
bEnable	是否使能	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 仅RV1106, RV1103支持。
- u32MinQp、u32MinIqp建议配置10, u32MaxQp、u32MaxIqp建议配置51, u32FrmMinQp、u32FrmMinIqp建议配置为28。
- u32Gop建议配置为2s~8s时长对应的帧数。配置为2s以上时可以配置gop mode为smartP, 配置虚拟I帧。
- enRcMode需要配置为VBR, u32MinBitRate和u32MaxBitRate根据具体产品不同分辨率进行配置, 典型的配置如下:

	u32MinBitRate	u32MaxBitRate
1080P	100Kbits	1500Kbits
1440P	200Kbits	3000Kbits

4.86 RK_MPI_VENC_EnableMotionDeblur

【描述】

使能运动去模糊。

【语法】

```
RK_S32 RK_MPI_VENC_EnableMotionDeblur(VENC_CHN VeChn, RK_BOOL bEnable)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
bEnable	是否使能	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 仅RV1106, RV1103支持。
- 开启智能编码时默认使能, 无需额外配置。

4.87 RK_MPI_VENC_EnableThumbnail

【描述】

使能缩略图。

【语法】

```
RK_S32 RK_MPI_VENC_EnableThumbnail(VENC_CHN VeChn)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 仅RV1106, RV1103支持。
- 仅JPEG通道支持，一个JPEG通道使能缩略图后，仅支持用户请求缩略图后输出，不支持JPEG拍照等功能，也就是说拍照通道和缩略图通道不能使用同一个通道，缩略图通道为了拍照通道或者H264/H265 I帧缩略图服务。

4.88 RK_MPI_VENC_ThumbnailBind

【描述】

缩略图绑定。

【语法】

```
RK_S32 RK_MPI_VENC_ThumbnailBind(VENC_CHN VeChn, VENC_CHN VeChnTb)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
VeChnTb	缩略图编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 仅RV1106, RV1103支持。
- VeChn可以为拍照通道或者H264/H265录像通道。
- VeChnTb需为RK_MPI_VENC_EnableThumbnail对应的通道号。

4.89 RK_MPI_VENC_ThumbnailRequest

【描述】

缩略图请求。

【语法】

```
RK_S32 RK_MPI_VENC_ThumbnailRequest(VENC_CHN VeChn)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 仅RV1106, RV1103支持。
- VeChn可以为拍照通道或者H264/H265录像通道。

4.90 RK_MPI_VENC_SetSceneMode

【描述】

设置SceneMode。

【语法】

```
RK_S32 RK_MPI_VENC_SetSceneMode(VENC_CHN VeChn, const VENC\_SCENE\_MODE\_E enSceneMode)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
enSceneMode	SceneMode。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 VENC错误码 。

【注意】

- 无

4.91 RK_MPI_VENC_GetSceneMode

【描述】

获取SceneMode。

【语法】

```
RK_S32 RK_MPI_VENC_GetSceneMode(VENC_CHN VeChn, VENC\_SCENE\_MODE\_E*penSceneMode)
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
penSceneMode	SceneMode。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 无

4.92 RK_MPI_VENC_SetSliceSplit

【描述】

设置编码通道slice分割属性。

【语法】

```
RK_S32 RK_MPI_VENC_SetSliceSplit(VENC_CHN VeChn, const VENC\_SLICE\_SPLIT\_S *pstSliceSplit);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstSliceSplit	编码通道slice分割属性参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstSliceSplit为空, 则返回失败。
- 本接口在编码通道创建之后, 编码通道销毁之前设置。
- 建议用户在创建通道之后, 启动编码之前调用此接口, 减少在编码过程中调用的次数。

4.93 RK_MPI_VENC_GetSliceSplit

【描述】

获取编码通道slice分割属性。

【语法】

```
RK_S32 RK_MPI_VENC_GetSliceSplit(VENC_CHN VeChn, VENC\_SLICE\_SPLIT\_S *pstRefParam);
```

【参数】

参数名	描述	输入/输出
VeChn	编码通道号。取值范围: [0, VENC_MAX_CHN_NUM)。	输入
pstSliceSplit	编码通道slice分割属性参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 VENC错误码 。

【注意】

- 如果通道未创建或者pstSliceSplit为空, 则返回失败。
- 本接口在编码通道创建之后, 编码通道销毁之前调用。

5. 数据类型

5.1 VENC_MAX_CHN_NUM

【说明】

定义编码通道最大个数。

【定义】

```
#define VENC_MAX_CHN_NUM      16
```

【注意事项】

无

5.2 VENC_CHN_ATTR_S

【说明】

定义编码通道属性结构体。

【定义】

```
typedef struct rkVENC_CHN_ATTR_S {
    VENC_ATTR_S stVencAttr;
    VENC_RC_ATTR_S stRcAttr;
    VENC_GOP_ATTR_S stGopAttr;
} VENC_CHN_ATTR_S;
```

【成员】

成员名称	描述
stVencAttr	编码器属性。
stRcAttr	码率控制器属性。
stGopAttr	Gop Mode 类型的结构体。

【注意事项】

无

5.3 VENC_ATTR_S

【说明】

定义编码器属性结构体。

【定义】

```
typedef struct rkVENC_ATTR_S {
    RK_CODEC_ID_E enType;
    PIXEL_FORMAT_E enPixelFormat;
    MIRROR_E enMirror;
    RK_U32 u32BufSize;
    RK_U32 u32Profile;
    RK_BOOL bByFrame;
    RK_U32 u32PicWidth;
    RK_U32 u32PicHeight;
    RK_U32 u32VirWidth;
    RK_U32 u32VirHeight;
    RK_U32 u32StreamBufCnt;
    union {
        VENC_ATTR_H264_S stAttrH264e;
        VENC_ATTR_H265_S stAttrH265e;
        VENC_ATTR_MJPEG_S stAttrMjpeg;
        VENC_ATTR_JPEG_S stAttrJjpeg;
    };
} VENC_ATTR_S;
```

【成员】

成员名称	描述
enType	编码协议类型。H264、H265等。
enPixelFormat	编码输入图像像素格式。 详情参看rk_comm_video.h中PIXEL_FORMAT_E定义。
enMirror	编码输出图像镜像。
u32BufSize	码流 buffer 大小。 推荐值： 对于 H264/H265： 一幅图像大小的 1/2。 对于 Jpeg/Mjpeg： 一幅图像宽高的乘积。
u32Profile	编码的等级。H.264 取值。 66: Baseline。 77: Main Profile。 100: High Profile。 H.265 取值 0: Main Profile。 1: Main 10 Profile。 Jpeg/Mjpeg 取值 0: Baseline
bByFrame	帧/包模式获取码流。取值范围：{RK_TRUE, RK_FALSE}。 RK_TRUE：按帧获取。 RK_FALSE：按包获取。 暂未使用。 目前默认输出按帧获取，264 I帧包含sps/pps/idr 3个NALU包信息； 265 I帧包含vps/sps/pps/idr 4个NALU包信息。
u32PicWidth	编码图像宽度，以像素为单位。
u32PicHeight	编码图像高度，以像素为单位。
u32VirWidth	硬件编码缓存宽度。以像素单位。
u32VirHeight	硬件编码缓存高度。以像素单位。
u32StreamBufCnt	编码输出的最大缓存个数。
stAttrH264e/stAttrMjpege/stAttrJpege/stAttrH265e	码率控制器属性。

【注意事项】

- RK_MPI_VENC_StartRecvFrame之前所有成员可以修改，RK_MPI_VENC_StartRecvFrame之后所有成员不可以修改。

5.4 VENC_RC_ATTR_S

【说明】

定义编码通道码率控制器属性。

【定义】

```
typedef struct rkVENC_RC_ATTR_S {
    VENC_RC_MODE_E enRcMode;
    union {
        VENC_H264_CBR_S stH264Cbr;
        VENC_H264_VBR_S stH264Vbr;
        VENC_H264_AVBR_S stH264Avbr;
        VENC_H264_FIXQP_S stH264FixQP;
    }
}
```

```

VENC_MJPEG_CBR_S stMjpegCbr;
VENC_MJPEG_VBR_S stMjpegVbr;
VENC_MJPEG_FIXQP_S stMjpegFixQp;

VENC_H265_CBR_S stH265Cbr;
VENC_H265_VBR_S stH265Vbr;
VENC_H265_AVBR_S stH265Avbr;
VENC_H265_FIXQP_S stH265FixQp;
};

} VENC_RC_ATTR_S;

```

【成员】

成员名称	描述
enRcMode	RC 模式。
stH264Cbr	H.264 协议编码通道 Cbr 模式属性。
stH264Vbr	H.264 协议编码通道 Vbr 模式属性。
stH264Avbr	H.264 协议编码通道 AVbr 模式属性。
stH264FixQp	H.264 协议编码通道 FixQp 模式属性。
stMjpegCbr	Mjpeg 协议编码通道 cbr 模式属性。
stMjpegVbr	Mjpeg 协议编码通道 Vbr 模式属性。
stMjpegFixQp	Mjpeg 协议编码通道 FixQp 模式属性。
stH265Cbr	H.265 协议编码通道 Cbr 模式属性。
stH265Vbr	H.265 协议编码通道 Vbr 模式属性。
stH265Avbr	H.265 协议编码通道 AVbr 模式属性。
stH265FixQp	H.265 协议编码通道 FixQp 模式属性。

【注意事项】

- 所有成员均为动态属性。
- 码率设置要求：u32MinBitRate <= u32BitRate <= u32MaxBitRate。
- 如果u32BitRate设置超过u32MaxBitRate， u32MaxBitRate将以u32BitRate为准。即实际生效为：u32MinBitRate<=u32BitRate=u32BitRate。
- 如果u32BitRate小于u32MinBitRate， u32MinBitRate将以u32BitRate为准。即实际生效为：u32BitRate=u32BitRate<=u32MaxBitRate。

5.5 VENC_RC_PARAM_S

【说明】

定义编码通道的码率控制高级参数。

【定义】

```

typedef struct rkVENC_RC_PARAM_S {
    RK_U32 s32FirstFrameStartQp;
    union {
        VENC_PARAM_H264_S stParamH264;
        VENC_PARAM_H265_S stParamH265;
        VENC_PARAM_MJPEG_S stParamMjpeg;
    };
} VENC_RC_PARAM_S;

```

【成员】

成员名称	描述
s32FirstFrameStartQp	第一帧的qp值，目前仅针对H264/H265编码有效，其他编码可不设置。 取值范围：[u32MinIQP, u32MaxIQP]。默认26。
stParamH264	h264 通道qp参数控制。
stParamH265	h265 通道qp参数控制。
stParamMjpeg	mjpeg 通道qp参数控制。

【注意事项】

- H264/H265编码时，s32FirstFrameStartQp的取值受对应I帧qp参数的限制，取值范围应在[u32MinIQP, u32MaxIQP]，否则会导致所有码率控制高级参数设置失败。
- 所有成员均为动态属性。

5.6 VENC_RC_PARAM2_S

【说明】

定义编码通道的码率控制高级参数2。

【定义】

```
typedef struct rkVENC_RC_PARAM2_S {
    RK_U32 u32ThrdI[16];
    RK_U32 u32ThrdP[16];
    RK_S32 s32AqStepI[17];
    RK_S32 s32AqStepP[17];
} VENC_RC_PARAM2_S;
```

【成员】

成员名称	描述
u32ThrdI	I帧对应的块级纹理复杂度阈值，数组内每个值取值范围:[0, 255]。 默认取值:[0,0,0,0,3,3,5,5,8,8,8,15,15,20,25,25]。
u32ThrdP	P帧对应的块级纹理复杂度阈值，数组内每个值取值范围:[0, 255]。 默认取值:[0,0,0,0,3,3,5,5,8,8,8,15,15,20,25,25]。
s32AqStepI	I帧的u32ThrdI[16]共16个阈值划分得到17个区间，每个区间对应s32AqStepI[17]的17个delta qp值，数组内每个值取值范围[-51,51]。 默认取值:[-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,7,8]
s32AqStepP	P帧的u32ThrdI[16]共16个阈值划分得到17个区间，每个区间对应s32AqStepI[17]的17个delta qp值，数组内每个值取值范围[-51,51]。 默认取值:[-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,7,8]
u32RowQpDelta	P帧行级码控调节幅度，取值范围[0, 10]，常用范围[0, 4]，264/265默认值为2，Smart video coding强制0
u32RowIQpDelta	I帧行级码控调节幅度，取值范围[0, 10]，265默认值为2，264默认值为1，Smart video coding强制为0

【注意事项】

- u32ThrdI、u32ThrdP要求从小到大依次排列，一般不会频繁的调整该阈值；
- s32AqStepI、s32AqStepP在RK3588、RV1106、RV1103的数组有效长度为16，数组最后一个值用于后向兼容；
- s32AqStepI、s32AqStepP要求从小到大依次排列，负值表示块级qp减小对应负值的绝对值，正值表示块级qp累加上该值；
- 例如，u32ThrdI[10]=8,u32ThrdI[11]=15,若s32AqStepI[11]=3，则表示madi在(8,15]的值对应的delta qp为3；若s32AqStepI[11]=8，则表示madi在(8,15]的值对应的delta qp为8，8相对于3增大了5，后者把madi在(8,15]区间内的qp值相较于前者增大了5，自然后者会让madi在(8,15]区间范围内的画质编码得更差，但相应的码率会更加节省。

5.7 VENC_PARAM_H264_S

【说明】

定义 H264 协议编码通道qp参数配置。

【定义】

```
typedef struct rkVENC_PARAM_H264_S {  
    RK_U32 u32StepQp;  
    RK_U32 u32MaxQp;  
    RK_U32 u32MinQp;  
    RK_U32 u32MaxIQP;  
    RK_U32 u32MinIQP;  
    RK_S32 s32DeltaIpQP;  
    RK_S32 s32MaxReEncodeTimes;  
    RK_U32 u32FrmMaxQP;  
    RK_U32 u32FrmMinQP;  
    RK_U32 u32FrmMaxIQP;  
    RK_U32 u32FrmMinIQP;  
    RK_U32 u32MotionStaticSwitchFrmQP;  
} VENC_PARAM_H264_S;
```

【成员】

成员名称	描述
u32StepQp	qp最大步进参数设置。
u32MaxQp	P帧最大qp设置。 取值范围：[1, 51]。默认51。
u32MinQp	P帧最小qp设置。 取值范围：[1, u32MaxQp]。默认10。
u32MaxIQP	I帧最大qp设置。 取值范围：[1, 51]。默认46。
u32MinIQP	I帧最小qp设置。 取值范围：[1, u32MaxIQP]。默认24。
s32DeltaIpQP	I帧前几帧P帧平均QP与I帧的差值，即s32DeltaIpQP=average(PPPP)-I； 该值越大，I帧质量相对越好，呼吸效应越明显； 该值越小，I帧质量相对越差，呼吸效应相对较弱，但是也会影响后续的P帧质量。 取值范围：[-7, 7]。默认2。
s32MaxReEncodeTimes	最大重编次数：[0, 3]。默认1。
u32FrmMaxQP	P帧最大帧级qp设置。 取值范围：[1, 51]。
u32FrmMinQP	P帧最小帧级qp设置。 取值范围：[1, 51]。
u32FrmMaxIQP	I帧最大帧级qp设置。 取值范围：[1, 51]。
u32FrmMinIQP	I帧最小帧级qp设置。 取值范围：[1, 51]。
u32MotionStaticSwitchFrmQP	动静切换帧级qp设置。 取值范围：[1, 51]。

【注意事项】

- 相关参数需要设置在范围内，并且最小qp不能超过最大qp，否则会导致所有qp参数设置失败。
- 所有成员均为动态属性。

5.8 VENC_PARAM_H265_S

【说明】

定义 H265 协议编码通道qp参数配置。

【定义】

```
typedef struct rkVENC_PARAM_H265_S {  
    RK_U32 u32StepQp;  
    RK_U32 u32MaxQp;  
    RK_U32 u32MinQp;  
    RK_U32 u32MaxIQP;  
    RK_U32 u32MinIQP;  
    RK_S32 s32DeltaIpQP;  
    RK_S32 s32MaxReEncodeTimes;  
    RK_U32 u32FrmMaxQP;  
    RK_U32 u32FrmMinQP;  
    RK_U32 u32FrmMaxIQP;  
    RK_U32 u32FrmMinIQP;  
    RK_U32 u32MotionStaticSwitchFrmQP;  
} VENC_PARAM_H265_S;
```

【成员】

成员名称	描述
u32StepQp	qp最大步进参数设置。
u32MaxQp	P帧最大qp设置。 取值范围：[1, 51]。默认51。
u32MinQp	P帧最小qp设置。 取值范围：[1, u32MaxQp]。默认10。
u32MaxIQP	I帧最大qp设置。 取值范围：[1, 51]。默认46。
u32MinIQP	I帧最小qp设置。 取值范围：[1, u32MaxIQP]。默认24。
s32DeltaIpQP	I帧前几帧P帧平均QP与I帧的差值，即s32DeltaIpQP=average(PPPP)-I； 该值越大，I帧质量相对越好，呼吸效应越明显； 该值越小，I帧质量相对越差，呼吸效应相对较弱，但是也会影响后续的P帧质量。 取值范围：[-7, 7]。默认2。
s32MaxReEncodeTimes	最大重编次数：[0, 3]。默认1。
u32FrmMaxQP	P帧最大帧级qp设置。 取值范围：[1, 51]。
u32FrmMinQP	P帧最小帧级qp设置。 取值范围：[1, 51]。
u32FrmMaxIQP	I帧最大帧级qp设置。 取值范围：[1, 51]。
u32FrmMinIQP	I帧最小帧级qp设置。 取值范围：[1, 51]。
u32MotionStaticSwitchFrmQP	动静切换帧级qp设置。 取值范围：[1, 51]。

【注意事项】

- 相关参数需要设置在范围内，并且最小qp不能超过最大qp，否则会导致所有qp参数设置失败。
- 所有成员均为动态属性。

5.9 VENC_PARAM_MJPEG_S

【说明】

定义 MJPEG 协议编码通道qp参数配置。

【定义】

```
typedef struct rkVENC_PARAM_MJPEG_S {
    RK_U32 u32Qfactor;
    RK_U32 u32MaxQfactor;
    RK_U32 u32MinQfactor;
} VENC_PARAM_MJPEG_S;
```

【成员】

成员名称	描述
u32Qfactor	MJPEG 编码的 Qfactor。 取值范围： [1, 99]。默认值： 70。
u32MaxQfactor	MJPEG 编码的最大Qfactor。 取值范围： [u32Qfactor, 99]。默认值： 99。
u32MinQfactor	MJPEG 编码的最小Qfactor。 取值范围： [1, u32Qfactor]。默认值： 30。

【注意事项】

- u32Qfactor同[VENC_MJPEG_FIXQP_S](#)中的u32Qfactor一致，都可控制mpeg的Qfactor； mjpeg fixqp时建议通过[VENC_MJPEG_FIXQP_S](#)一起设置即可。
- MJPEG设置为CBR或VBR模式时可通过设置qfactor范围进行qp参数配置。
- 所有成员均为动态属性。

5.10 VIDEO_FRAME_INFO_S

【说明】

定义视频图像帧信息结构体。

【定义】

```
typedef struct rkVIDEO_FRAME_S {
    MB_BLK      pMbBlk;
    RK_U32      u32Width;
    RK_U32      u32Height;
    RK_U32      u32VirWidth;
    RK_U32      u32VirHeight;
    VIDEO_FIELD_E   enField;
    PIXEL_FORMAT_E  enPixelFormat;
    VIDEO_FORMAT_E  enVideoFormat;
    COMPRESS_MODE_E enCompressMode;
    DYNAMIC_RANGE_E enDynamicRange;
    COLOR_GAMUT_E   enColorGamut;
    RK_VOID      *pVirAddr[RK_MAX_COLOR_COMPONENT];
    RK_U32      u32TimeRef;
    RK_U64      u64PTS;
    RK_U64      u64PrivateData;
    RK_U32      u32FrameFlag;
} VIDEO_FRAME_S;

typedef struct rkVIDEO_FRAME_INFO_S {
    VIDEO_FRAME_S stVFrame;
} VIDEO_FRAME_INFO_S;
```

【成员】

成员名称	描述
pMbBlk	用户数据。
u32Width	图像宽度。
u32Height	图像高度。
u32VirWidth	图像内存宽度。 非压缩数据：JPEG/MJPEG需要16对齐，H264需要16对齐，H265需要64对齐。 压缩数据：H264/H265（目前仅H264/H265支持压缩数据）需要16对齐。
u32VirHeight	图像内存高度。 非压缩数据：JPEG/MJPEG需要16对齐，H264/H265需要2对齐。 压缩数据：H264/H265（目前仅H264/H265支持压缩数据）需要4对齐。
enField	图像帧场模式。 暂未使用，可不设置。
enVideoFormat	图像视频格式。 暂未使用，可不设置。
enPixelFormat	图像像素格式。详情参看rk_comm_video.h中PIXEL_FORMAT_E定义。
enCompressMode	图像压缩模式。 COMPRESS_MODE_NONE：无压缩； COMPRESS_AFBC_16x16：AFBC压缩。
enDynamicRange	图像动态范围。 暂未使用，可不设置。
enColorGamut	图像色域范围。 暂未使用，可不设置。
pVirAddr	图像数据虚拟地址。 暂未使用，可不设置。
u32TimeRef	图像帧序列号。 暂未使用，可不设置。
u64PTS	图像时间戳，单位us。建议通过clock_gettime(CLOCK_MONOTONIC, &time)设置时间戳。
u64PrivateData	私有数据。 暂未使用，可不设置。
u32FrameFlag	当前帧的标记，使用RTMBFlags里面的值标记，可以按位或操作。无特殊操作，建议设置为0即可。

【注意事项】

- 用户获取图像数据，可以通过RK_MPI_MB_Handle2VirAddr转换pMbBlk成虚拟地址来使用。

```
data = RK_MPI_MB_Handle2VirAddr(sFrame.stVFrame.pMbBlk);
fwrite(data, 1, sFrame.stVFrame.u32Width * sFrame.stVFrame.u32Height * 3 / 2, fp);
fflush(fp);
```

5.11 VENC_H264_CBR_S

【说明】

定义 H.264 编码通道 CBR 属性结构。

【定义】

```

typedef struct rkVENC_H264_CBR_S {
    RK_U32 u32Gop;
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_U32 fr32DstFrameRateNum;
    RK_U32 fr32DstFrameRateDen;
    RK_U32 u32BitRate;
    RK_U32 u32StatTime;
} VENC_H264_CBR_S;

```

【成员】

成员名称	描述
u32Gop	H.264 gop 值。 取值范围：[1, 65536]。
u32SrcFrameRateNum	输入帧率分子
u32SrcFrameRateDen	输出帧率分母
fr32DstFrameRateNum	输出帧率分子
fr32DstFrameRateDen	输出帧率分母
u32BitRate	平均 bitrate, 以 kbps 为单位。 取值范围：[3,200000], 默认值：u32VirWidth <i>u32VirHeight / 830</i> 。
u32StatTime	码率统计时间, 以秒为单位。 取值范围：[1, 60], 默认值3。

【注意事项】

- 假设输入帧率为30, u32SrcFrameRateNum 应设置为30, u32SrcFrameRateDen设置为1。
- 假设输出帧率为30, fr32DstFrameRateNum 应设置为30, fr32DstFrameRateDen设置为1。
- 假设输入帧率25, 输出帧率12, 则表示将从25帧输入图像中取12帧进行编码, 其余13帧将丢掉。
- 所有成员均为动态属性。

5.12 VENC_H264_VBR_S

【说明】

定义 H.264 编码通道 VBR 属性结构。

【定义】

```

typedef struct rkVENC_H264_VBR_S {
    RK_U32 u32Gop;
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_U32 fr32DstFrameRateNum;
    RK_U32 fr32DstFrameRateDen;
    RK_U32 u32BitRate;
    RK_U32 u32MaxBitRate;
    RK_U32 u32MinBitRate;
    RK_U32 u32StatTime;
} VENC_H264_VBR_S;

```

【成员】

成员名称	描述
u32Gop	H.264 gop 值。 取值范围：[1, 65536]。
u32SrcFrameRateNum	输入帧率分子
u32SrcFrameRateDen	输出帧率分母
fr32DstFrameRateNum	输出帧率分子
fr32DstFrameRateDen	输出帧率分母
u32BitRate	平均 bitrate，以 kbps 为单位。 取值范围：[3,200000]，默认值： $u32VirWidth \cdot u32VirHeight / 830$ 。
u32MaxBitRate	最高bitrate，以 kbps 为单位。 取值范围：[u32BitRate,200000]，默认值： $u32VirWidth \cdot u32VirHeight / 830 * 3 / 2$ 。
u32MinBitRate	最低bitrate，以 kbps 为单位。 取值范围：[3, u32BitRate]，默认值： $u32VirWidth \cdot u32VirHeight / 830 * 1 / 2$ 。
u32StatTime	码率统计时间，以秒为单位。 取值范围：[1, 60]，默认值3。

【注意事项】

- 请参见 VENC_H264_CBR_S注意事项。
- 所有成员均为动态属性。

5.13 VENC_H264_AVBR_S

【说明】

定义 H.264 编码通道 AVBR 属性结构。

【定义】

```
typedef struct rkVENC_H264_AVBR_S {
    RK_U32 u32Gop;
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_U32 fr32DstFrameRateNum;
    RK_U32 fr32DstFrameRateDen;
    RK_U32 u32BitRate;
    RK_U32 u32MaxBitRate;
    RK_U32 u32MinBitRate;
    RK_U32 u32StatTime;
} VENC_H264_AVBR_S;
```

【成员】

成员名称	描述
u32Gop	H.264 gop 值。 取值范围：[1, 65536]。
u32SrcFrameRateNum	输入帧率分子
u32SrcFrameRateDen	输出帧率分母
fr32DstFrameRateNum	输出帧率分子
fr32DstFrameRateDen	输出帧率分母
u32BitRate	平均 bitrate，以 kbps 为单位。 取值范围：[3,200000]，默认值： $u32VirWidth \cdot u32VirHeight / 830$ 。
u32MaxBitRate	最高bitrate，以 kbps 为单位。 取值范围：[u32BitRate,200000]，默认值： $u32VirWidth \cdot u32VirHeight / 830 * 3 / 2$ 。
u32MinBitRate	最低bitrate，以 kbps 为单位。 取值范围：[3, u32BitRate]，默认值： $u32VirWidth \cdot u32VirHeight / 830 * 1 / 2$ 。
u32StatTime	码率统计时间，以秒为单位。 取值范围：[1, 60]，默认值3。

【注意事项】

- 请参见 VENC_H264_CBR_S注意事项。
- 所有成员均为动态属性。

5.14 VENC_H264_FIXQP_S

【说明】

定义 H.264 编码通道 FIXQP 属性结构。

【定义】

```
typedef struct rkVENC_H264_FIXQP_S {
    RK_U32 u32Gop;
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_U32 fr32DstFrameRateNum;
    RK_U32 fr32DstFrameRateDen;
    RK_U32 u32Iqp;
    RK_U32 u32Pqp;
    RK_U32 u32Bqp;
} VENC_H264_FIXQP_S;
```

【成员】

成员名称	描述
u32Gop	H.264 gop 值。 取值范围：[1, 65536]。
u32SrcFrameRateNum	输入帧率分子
u32SrcFrameRateDen	输出帧率分母
fr32DstFrameRateNum	输出帧率分子
fr32DstFrameRateDen	输出帧率分母
u32IQp	I 帧所有宏块 Qp 值。 取值范围：[1, 51]。
u32PQp	P 帧所有宏块 Qp 值。 取值范围：[1, 51]。
u32BQp	B 帧所有宏块 Qp 值。 取值范围：[1, 51]。 暂未使用。

【注意事项】

- 请参见 VENC_H264_CBR_S 注意事项。
- 所有成员均为动态属性。

5.15 VENC_H265_CBR_S

【说明】

定义 H.265 编码通道 CBR 属性结构。

【定义】

```
typedef struct rkVENC_H265_CBR_S {
    RK_U32 u32Gop;
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_U32 fr32DstFrameRateNum;
    RK_U32 fr32DstFrameRateDen;
    RK_U32 u32BitRate;
    RK_U32 u32StatTime;
} VENC_H265_CBR_S;
```

【成员】

成员名称	描述
u32Gop	H.265 gop 值。 取值范围：[1, 65536]。
u32SrcFrameRateNum	输入帧率分子
u32SrcFrameRateDen	输出帧率分母
fr32DstFrameRateNum	输出帧率分子
fr32DstFrameRateDen	输出帧率分母
u32BitRate	平均 bitrate，以 kbps 为单位。 取值范围：[3,200000]，默认值：u32VirWidth * u32VirHeight / 830。
u32StatTime	码率统计时间，以秒为单位。 取值范围：[1, 60]，默认值3。

【注意事项】

- 请参见 VENC_H264_CBR_S注意事项。
- 所有成员均为动态属性。

5.16 VENC_H265_VBR_S

【说明】

定义 H.265 编码通道 VBR 属性结构。

【定义】

```
typedef struct rkVENC_H265_VBR_S {  
    RK_U32 u32Gop;  
    RK_U32 u32SrcFrameRateNum;  
    RK_U32 u32SrcFrameRateDen;  
    RK_U32 fr32DstFrameRateNum;  
    RK_U32 fr32DstFrameRateDen;  
    RK_U32 u32BitRate;  
    RK_U32 u32MaxBitRate;  
    RK_U32 u32MinBitRate;  
    RK_U32 u32StatTime;  
} VENC_H265_VBR_S;
```

【成员】

成员名称	描述
u32Gop	H.265 gop 值。 取值范围：[1, 65536]。
u32SrcFrameRateNum	输入帧率分子
u32SrcFrameRateDen	输出帧率分母
fr32DstFrameRateNum	输出帧率分子
fr32DstFrameRateDen	输出帧率分母
u32BitRate	平均 bitrate，以 kbps 为单位。 取值范围：[3,200000]，默认值： $u32VirWidth \cdot u32VirHeight / 830$ 。
u32MaxBitRate	最高bitrate，以 kbps 为单位。 取值范围：[u32BitRate,200000]，默认值： $u32VirWidth \cdot u32VirHeight / 830 * 3 / 2$ 。
u32MinBitRate	最低bitrate，以 kbps 为单位。 取值范围：[3, u32BitRate]，默认值： $u32VirWidth \cdot u32VirHeight / 830 * 1 / 2$ 。
u32StatTime	码率统计时间，以秒为单位。 取值范围：[1, 60]，默认值3。

【注意事项】

- 请参见 VENC_H264_CBR_S注意事项。
- 所有成员均为动态属性。

5.17 VENC_H265_AVBR_S

【说明】

定义 H.265 编码通道 AVBR 属性结构。

【定义】

```

typedef struct rkVENC_H265_AVBR_S {
    RK_U32 u32Gop;
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_U32 fr32DstFrameRateNum;
    RK_U32 fr32DstFrameRateDen;
    RK_U32 u32BitRate;
    RK_U32 u32MaxBitRate;
    RK_U32 u32MinBitRate;
    RK_U32 u32StatTime;
} VENC_H265_AVBR_S;

```

【成员】

成员名称	描述
u32Gop	H.265 gop 值。 取值范围：[1, 65536]。
u32SrcFrameRateNum	输入帧率分子
u32SrcFrameRateDen	输出帧率分母
fr32DstFrameRateNum	输出帧率分子
fr32DstFrameRateDen	输出帧率分母
u32BitRate	平均 bitrate，以 kbps 为单位。 取值范围：[3,200000]，默认值： $u32VirWidth \cdot u32VirHeight / 830$ 。
u32MaxBitRate	最高bitrate，以 kbps 为单位。 取值范围：[u32BitRate,200000]，默认值： $u32VirWidth \cdot u32VirHeight / 830 * 3 / 2$ 。
u32MinBitRate	最低bitrate，以 kbps 为单位。 取值范围：[3, u32BitRate]，默认值： $u32VirWidth \cdot u32VirHeight / 830 * 1 / 2$ 。
u32StatTime	码率统计时间，以秒为单位。 取值范围：[1, 60]，默认值3。

【注意事项】

- 请参见 VENC_H264_CBR_S注意事项。
- 所有成员均为动态属性。

5.18 VENC_H265_FIXQP_S

【说明】

定义 H.265 编码通道 FIXQP 属性结构。

【定义】

```

typedef struct rkVENC_H265_FIXQP_S {
    RK_U32 u32Gop;
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_U32 fr32DstFrameRateNum;
    RK_U32 fr32DstFrameRateDen;
    RK_U32 u32Iqp;
    RK_U32 u32Pqp;
    RK_U32 u32Bqp;
} VENC_H265_FIXQP_S;

```

【成员】

成员名称	描述
u32Gop	H.265 gop 值。 取值范围：[1, 65536]。
u32SrcFrameRateNum	输入帧率分子
u32SrcFrameRateDen	输出帧率分母
fr32DstFrameRateNum	输出帧率分子
fr32DstFrameRateDen	输出帧率分母
u32IQp	I 帧所有宏块 Qp 值。 取值范围：[1, 51]。
u32PQp	P 帧所有宏块 Qp 值。 取值范围：[1, 51]。
u32BQp	B 帧所有宏块 Qp 值。 取值范围：[1, 51]。 暂未使用。

【注意事项】

- 请参见 VENC_H264_CBR_S 注意事项。
- 所有成员均为动态属性。

5.19 VENC_MJPEG_CBR_S

【说明】

定义 mjpeg 编码通道 CBR 属性结构。

【定义】

```
typedef struct rkVENC_MJPEG_CBR_S {
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_U32 fr32DstFrameRateNum;
    RK_U32 fr32DstFrameRateDen;
    RK_U32 u32BitRate;
    RK_U32 u32StatTime;
} VENC_MJPEG_CBR_S;
```

【成员】

成员名称	描述
u32SrcFrameRateNum	输入帧率分子
u32SrcFrameRateDen	输出帧率分母
fr32DstFrameRateNum	输出帧率分子
fr32DstFrameRateDen	输出帧率分母
u32BitRate	平均 bitrate，以 kbps 为单位。 取值范围：[5,800000]，默认值： $u32VirWidth \cdot u32VirHeight / 830$ 。
u32StatTime	码率统计时间，以秒为单位。 取值范围：[1, 60]，默认值3。

【注意事项】

- 请参见 VENC_H264_CBR_S 注意事项。

- 所有成员均为动态属性。

5.20 VENC_MJPEG_VBR_S

【说明】

定义 jpeg 编码通道 VBR 属性结构。

【定义】

```
typedef struct rkVENC_MJPEG_VBR_S {
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_U32 fr32DstFrameRateNum;
    RK_U32 fr32DstFrameRateDen;
    RK_U32 u32BitRate;
    RK_U32 u32MaxBitRate;
    RK_U32 u32MinBitRate;
    RK_U32 u32StatTime;
} VENC_MJPEG_VBR_S;
```

【成员】

成员名称	描述
u32SrcFrameRateNum	输入帧率分子
u32SrcFrameRateDen	输出帧率分母
fr32DstFrameRateNum	输出帧率分子
fr32DstFrameRateDen	输出帧率分母
u32BitRate	平均 bitrate，以 kbps 为单位。 取值范围：[5,800000]，默认值： $u32VirWidth \cdot u32VirHeight / 830$ 。
u32MaxBitRate	最高bitrate，以 kbps 为单位。 取值范围：[u32BitRate,800000]，默认值： $u32VirWidth \cdot u32VirHeight / 830 * 3 / 2$ 。
u32MinBitRate	最低bitrate，以 kbps 为单位。 取值范围：[5, u32BitRate]，默认值： $u32VirWidth \cdot u32VirHeight / 830 * 1 / 2$ 。
u32StatTime	码率统计时间，以秒为单位。 取值范围：[1, 60]，默认值3。

【注意事项】

- 请参见 VENC_H264_CBR_S 注意事项。
- 所有成员均为动态属性。

5.21 VENC_MJPEG_FIXQP_S

【说明】

定义 jpeg 编码通道 FIXQP 属性结构。

【定义】

```

typedef struct rkVENC_MJPEG_FIXQP_S {
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_U32 fr32DstFrameRateNum;
    RK_U32 fr32DstFrameRateDen;
    RK_U32 u32Qfactor;
} VENC_MJPEG_FIXQP_S;

```

【成员】

成员名称	描述
u32SrcFrameRateNum	输入帧率分子
u32SrcFrameRateDen	输出帧率分母
fr32DstFrameRateNum	输出帧率分子
fr32DstFrameRateDen	输出帧率分母
u32Qfactor	MJPEG 编码的 Qfactor。 取值范围：[1, 99]。默认值：70。

【注意事项】

- 请参见 VENC_H264_CBR_S 注意事项。
- 所有成员均为动态属性。

5.22 VENC_GOP_ATTR_S

【说明】

定义编码器 GOP 属性结构体。

【定义】

```

typedef struct rkVENC_GOP_ATTR_S {
    VENC_GOP_MODE_E enGopMode;
    RK_S32 s32VirldrLen;
    RK_U32 u32MaxLtrCount;
    RK_U32 u32TsvcPreload;
} VENC_GOP_ATTR_S;

```

【成员】

成员名称	描述
enGopMode	编码 GOP 类型。
s32VirldrLen	虚拟I帧长度。默认为0，即不生效。
u32MaxLtrCount	最大长期参考帧个数。
u32TsvcPreload	Tsvc预加载。

【注意事项】

- s32VirldrLen仅针对enGopMode为VENC_GOPMODE_SMARTP时生效。建议设置为GOP长度的一半或者能被其整数倍整除。
- u32MaxLtrCount: normalp和其他gop mode需要动态切换时，设置为1；如果仅有normalp，设置为0。
- u32TsvcPreload: smartp和tsvc mode需要动态切换时，设置为1；如果仅有smartp，设置为0。
- 所有成员均为动态属性。

5.23 VENC_RECV_PIC_PARAM_S

【说明】

定义编码通道连续接收并编码的帧数结构体。

【定义】

```
typedef struct rkVENC_RECV_PIC_PARAM_S {
    RK_S32 s32RecvPicNum;
} VENC_RECV_PIC_PARAM_S;
```

【成员】

成员名称	描述
s32RecvPicNum	编码通道连续接收并编码的帧数。范围：[-1,0) ∪ (0,∞]。默认-1，即连续编码不自动停止。

【注意事项】

- 如s32RecvPicNum设置为正数时，为抓图模式，编码到对应帧数后不再输出编码数据。
- RK_MPI_VENC_StopRecvFrame之后，成员才可以重新配置。

5.24 VENC_CHN_STATUS_S

【说明】

定义编码通道的状态结构体。

【定义】

```
typedef struct rkVENC_CHN_STATUS_S {
    RK_U32 u32LeftPics;
    RK_U32 u32LeftStreamBytes;
    RK_U32 u32LeftStreamFrames;
    RK_U32 u32CurPacks;
    RK_U32 u32LeftRecvPics;
    RK_U32 u32LeftEncPics;
    RK_BOOL bJpegSnapEnd;
    VENC_STREAM_INFO_S stVencStrmInfo;
} VENC_CHN_STATUS_S;
```

【成员】

成员名称	描述
u32LeftPics	待编码的图像数。
u32LeftStreamBytes	码流 buffer 剩余的 byte 数。
u32LeftStreamFrames	码流 buffer 剩余的帧数。
u32CurPacks	当前帧的码流包个数。
u32LeftRecvPics	剩余待接收的帧数，在用户调用接口 RK_MPI_VENC_StartRecvFrame 设置接收帧数后有效。
u32LeftEncPics	剩余待编码的帧数，在用户调用接口 RK_MPI_VENC_StartRecvFrame 设置接收帧数后有效。
bJpegSnapEnd	Jpeg 抓拍模式下指示抓拍过程是否结束。 暂未使用。
stVencStrmInfo	编码器码流特征信息。

【注意事项】

无

5.25 VENC_STREAM_S

【说明】

定义帧码流类型结构体。

【定义】

```
typedef struct rkVENC_STREAM_S {
    VENC_PACK_S ATTRIBUTE* pstPack;
    RK_U32    ATTRIBUTE u32PackCount;
    RK_U32    u32Seq;

    union {
        VENC_STREAM_INFO_H264_S stH264Info;
        VENC_STREAM_INFO_JPEG_S stJpegInfo;
        VENC_STREAM_INFO_H265_S stH265Info;
        VENC_STREAM_INFO_PRORES_S stProresInfo;
    };

    union {
        VENC_STREAM_ADVANCE_INFO_H264_S stAdvanceH264Info;
        VENC_STREAM_ADVANCE_INFO_JPEG_S stAdvanceJpegInfo;
        VENC_STREAM_ADVANCE_INFO_H265_S stAdvanceH265Info;
        VENC_STREAM_ADVANCE_INFO_PRORES_S stAdvanceProresInfo;
    };
} VENC_STREAM_S;
```

【成员】

成员名称	描述
pstPack	帧码流包结构。
u32PackCount	一帧码流的所有包的个数。 单包输出： -- 作为输入参数时设置为1。 -- 在函数调用成功后，此值为实际填充 pack 的包个数，默认为1 (H264/H265 I帧包含多个NALU包)，默认此模式。 多包输出： -- 作为输入参数时，此值为pstPack包的个数，必须不小于当前帧的包个数，目前需要按照16设置，如果此数值太小，可能会导致编码码流包获取失败导致丢帧，并且返回 VENC错误码 RK_ERR_VENC_ILLEGAL_PARAM 。 -- 在函数调用成功后，此值为实际填充 pack 的包个数。
u32Seq	码流序列号。按帧获取帧序号；按包获取包序号。
stH264Info/stJpegInfo/ stH265Info/stProresInfo	码流特征信息。
stAdvanceH264Info/ stAdvanceJpegInfo/ stAdvanceH265Info/stAdvanceProresInfo	码流高级特征信息。

【注意事项】

无

5.26 VENC_PACK_S

【说明】

定义帧码流包结构体。

【定义】

```
typedef struct rkVENC_PACK_S {  
    MB_BLK      pMbBlk;  
    RK_U32      u32Len;  
    RK_U64      u64PTS;  
    RK_BOOL     bFrameEnd;  
    RK_BOOL     bStreamEnd;  
    VENC_DATA_TYPE_U  DataType;  
    RK_U32      u32Offset;  
    RK_U32      u32DataNum;  
    VENC_PACK_INFO_S  stPackInfo[8];  
} VENC_PACK_S;
```

【成员】

成员名称	描述
pMbBlk	码流包缓存块句柄。
u32Len	码流包长度。 (H264/H265 I帧包含多个NALU包总长度)。
u64PTS	时间戳。单位：us。
bFrameEnd	帧结束标识。默认为RK_TRUE。 取值范围： RK_TRUE：该码流包是该帧的最后一个包。 RK_FALSE：该码流包不是该帧的最后一个包。
bStreamEnd	流结束标识。 取值范围： RK_TRUE：该码流包是数据流的最后一个包，编码器将输出编码图像最后一帧码流。 RK_FALSE：该码流包不是数据流的最后一个包。
DataType	码流类型，支持 H.264/JPEG/ H.265 协议类型的数据包。
u32Offset	码流包中有效数据与码流包缓存块的偏移。当前一个packet只有一帧数据，默认为0。
u32DataNum	当前码流包（当前包的类型由 DataType 指定）数据中包含其他类型码流包的个数（即除Islice/Pslice 外的NALU包个数）。目前仅H264/H265有效。JPEG/MJPEG时默认为1。 目前仅数组0的u32PackType数据有效，切等同于DataType。如需读取码流类型可直接读取DataType 中对应的值即可。
stPackInfo	当前码流包数据中包含其他类型码流包数据信息。目前仅H264/H265有效。

【注意事项】

无

5.27 VENC_DATA_TYPE_U

【说明】

定义码流结果类型。

【定义】

```

typedef union rkVENC_DATA_TYPE_U {
    H264E_NALU_TYPE_E enH264EType;
    JPEGE_PACK_TYPE_E enJPEGEType;
    H265E_NALU_TYPE_E enH265EType;
    PRORES_PACK_TYPE_E enPRORESType;
} VENC_DATA_TYPE_U;

```

【成员】

成员名称	描述
enH264EType	H.264 码流包类型。
enJPEGEType	JPEG 码流包类型。
enH265EType	H.265 码流包类型。
enPRORESType	PRORES 码流包类型。

【注意事项】

无

5.28 VENC_PACK_INFO_S

【说明】

定义当前码流包数据中包含的其他类型码流包数据的结构体。

【定义】

```

typedef struct rkVENC_PACK_INFO_S {
    VENC_DATA_TYPE_U u32PackType;
    RK_U32 u32PackOffset;
    RK_U32 u32PackLength;
} VENC_PACK_INFO_S;

```

【成员】

成员名称	描述
u32PackType	当前码流包数据包含其他码流包的类型。
u32PackOffset	当前码流包数据包含其他码流包数据的偏移。 暂未使用。
u32PackLength	当前码流包数据包含其他码流包数据的大小。 暂未使用。

【注意事项】

无

5.29 H264E_NALU_TYPE_E

【说明】

定义 H.264 码流 NALU 类型。

【定义】

```

typedef enum rkH264E_NALU_TYPE_E {
    H264E_NALU_BSslice = 0,
    H264E_NALU_PSLICE = 1,
    H264E_NALU_ISLICE = 2,
    H264E_NALU_IDRSLICE = 5,
    H264E_NALU_SEI = 6,
    H264E_NALU_SPS = 7,
    H264E_NALU_PPS = 8,
    H264E_NALU_BUTT
} H264E_NALU_TYPE_E;

```

【成员】

成员名称	描述
H264E_NALU_BSslice	BSslice 类型。
H264E_NALU_PSLICE	PSLICE 类型。
H264E_NALU_ISLICE	ISLICE 类型，帧类型为 P 帧。
H264E_NALU_IDRSLICE	ISLICE 类型，帧类型为 IDR 帧。
H264E_NALU_SEI	SEI 类型。
H264E_NALU_SPS	SPS 类型。
H264E_NALU_PPS	PPS 类型。
H264E_NALU_BUTT	最大值。

【注意事项】

无

5.30 H265E_NALU_TYPE_E

【说明】

定义 H.265 码流 NALU 类型。

【定义】

```

typedef enum rkH265E_NALU_TYPE_E {
    H265E_NALU_BSslice = 0,
    H265E_NALU_PSLICE = 1,
    H265E_NALU_ISLICE = 2,
    H265E_NALU_IDRSLICE = 19,
    H265E_NALU_VPS = 32,
    H265E_NALU_SPS = 33,
    H265E_NALU_PPS = 34,
    H265E_NALU_SEI = 39,
    H265E_NALU_BUTT
} H265E_NALU_TYPE_E;

```

【成员】

成员名称	描述
H265E_NALU_BSLICE	BSLICE 类型。
H265E_NALU_PSLICE	PSLICE 类型。
H265E_NALU_ISLICE	ISLICE 类型，帧类型为 P 帧。
H265E_NALU_IDRSlice	ISLICE 类型，帧类型为 IDR 帧。
H265E_NALU_VPS	VPS 类型。
H265E_NALU_SPS	SPS 类型。
H265E_NALU_PPS	PPS 类型。
H265E_NALU_SEI	SEI 类型。
H265E_NALU_BUTT	最大值。

【注意事项】

无

5.31 H264E_REF_TYPE_E

【说明】

定义 H.264 跳帧参考码流的帧类型以及参考属性。

【定义】

```
typedef enum rkH264E_REF_TYPE_E {
    BASE_IDRSlice = 0,
    BASE_PSLICE_REFTOIDR,
    BASE_PSLICE_REFBYBASE,
    BASE_PSLICE_REFBYENHANCE,
    ENHANCE_PSLICE_REFBYENHANCE,
    ENHANCE_PSLICE_NOTFORREF,
    ENHANCE_PSLICE_BUTT
} H264E_REF_TYPE_E;
```

【成员】

成员名称	描述
BASE_IDRSlice	base 层中的 IDR 帧。
BASE_PSLICE_REFTOIDR	base 层中的 P 帧，用于 base 层中其他帧的 参考且只参考 IDR 帧。
BASE_PSLICE_REFBYBASE	base 层中的 P 帧，用于 base 层中其他帧的 参考。
BASE_PSLICE_REFBYENHANCE	base 层中的 P 帧，用于 enhance 层中的帧 的参考。
ENHANCE_PSLICE_REFBYENHANCE	enhance 层中的 P 帧，用于 enhance 层中其他帧的参考。
ENHANCE_PSLICE_NOTFORREF	enhance 层中的 P 帧，不用于参考。
ENHANCE_PSLICE_BUTT	最大值。

【注意事项】

无

5.32 H265E_REF_TYPE_E

【说明】

定义 H.265 跳帧参考码流的帧类型以及参考属性。

【定义】

```
typedef enum rkH264E_REF_TYPE_E H265E_REF_TYPE_E;
```

【成员】

请参考 [H264E_REF_TYPE_E](#)。

【注意事项】

无

5.33 JPEGE_PACK_TYPE_E

【说明】

定义 JPEG 码流的 PACK 类型。

【定义】

```
typedef enum rkJPEGE_PACK_TYPE_E {
    JPEGE_PACK_ECS = 5,
    JPEGE_PACK_APP = 6,
    JPEGE_PACK_VDO = 7,
    JPEGE_PACK_PIC = 8,
    JPEGE_PACK_DCF = 9,
    JPEGE_PACK_DCF_PIC = 10,
    JPEGE_PACK_BUTT
} JPEGE_PACK_TYPE_E;
```

【成员】

成员名称	描述
JPEGE_PACK_ECS	ECS 类型。
JPEGE_PACK_APP	APP 类型。
JPEGE_PACK_VDO	VDO 类型。
JPEGE_PACK_PIC	PIC 类型。
JPEGE_PACK_DCF	DCF ECS类型。
JPEGE_PACK_DCF_PIC	ECS类型。
JPEGE_PACK_BUTT	最大值。

【注意事项】

- 以上参数均暂未使用。

5.34 PRORES_PACK_TYPE_E

【说明】

定义 PRORES 码流的 PACK 类型。

【定义】

```
typedef enum rkPRORES_PACK_TYPE_E {
    PRORES_PACK_PIC = 1,
    PRORES_PACK_BUTT
} PRORES_PACK_TYPE_E;
```

【成员】

成员名称	描述
PRORES_PACK_PIC	PRORES 数据包。
PRORES_PACK_BUTT	最大值。

【注意事项】

- 以上参数均暂未使用。

5.35 VENC_STREAM_INFO_H264_S

【说明】

定义 H.264 协议码流特征信息。

【定义】

```
typedef struct rkVENC_PACK_S {
    RK_U32          u32PicBytesNum;
    RK_U32          u32Inter16x16MbNum;
    RK_U32          u32Inter8x8MbNum;
    RK_U32          u32Intra16MbNum;
    RK_U32          u32Intra8MbNum;
    RK_U32          u32Intra4MbNum;
    H264E_REF_TYPE_E enRefType;
    RK_U32          u32UpdateAttrCnt;
    RK_U32          u32StartQp;
    RK_U32          u32MeanQp;
    RK_BOOL         bPSkip;
} VENC_PACK_S;
```

【成员】

成员名称	描述
u32PicBytesNum	编码当前帧的字节 (BYTE) 数。 暂未使用。
u32Inter16x16MbNum	编码当前帧中采用 Inter16x16 预测模式的宏块数。 暂未使用。
u32Inter8x8MbNum	编码当前帧中采用 Inter8x8 预测模式的宏块数。 暂未使用。
u32Intra16MbNum	编码当前帧中采用 Intra16 预测模式的宏块数。 暂未使用。
u32Intra8MbNum	编码当前帧中采用 Intra8 预测模式的宏块数。 暂未使用。
u32Intra4MbNum	编码当前帧中采用 Intra4 预测模式的宏块数。 暂未使用。
enRefType	高级跳帧参考下的编码帧类型。
u32UpdateAttrCnt	通道属性或参数(包含 RC 参数)被设置的次数。 暂未使用。
u32StartQp	编码当前帧的 startqp 值。 暂未使用。
u32MeanQp	编码当前帧的平均 QP 值。 暂未使用。
bPSkip	标识当前帧是否为 pskip 帧。 暂未使用。

【注意事项】

无

5.36 VENC_STREAM_INFO_H265_S

【说明】

定义 H.265 协议码流特征信息。

【定义】

```
typedef struct rkVENC_PACK_S {
    RK_U32      u32PicBytesNum;
    RK_U32      u32Inter64x64CuNum;
    RK_U32      u32Inter32x32CuNum;
    RK_U32      u32Inter16x16CuNum;
    RK_U32      u32Inter8x8CuNum;
    RK_U32      u32Intra32x32CuNum;
    RK_U32      u32Intra16x16CuNum;
    RK_U32      u32Intra8x8CuNum;
    RK_U32      u32Intra4x4CuNum;
    H265E_REF_TYPE_E enRefType;
    RK_U32      u32UpdateAttrCnt;
    RK_U32      u32StartQp;
    RK_U32      u32MeanQp;
    RK_BOOL     bPSkip;
} VENC_PACK_S;
```

【成员】

成员名称	描述
u32PicBytesNum	编码当前帧的字节（BYTE）数。 暂未使用。
u32Inter64x64CuNum	编码当前帧中采用 Inter64x64 预测模式的 CU 块数。 暂未使用。
u32Inter32x32CuNum	编码当前帧中采用 Inter32x32 预测模式的 CU 块数 暂未使用。
u32Inter16x16CuNum	编码当前帧中采用 Inter16x16 预测模式的 CU 块数 暂未使用。
u32Inter8x8CuNum	编码当前帧中采用 Inter8x8 预测模式的 CU 块数 暂未使用。
u32Intra32x32CuNum	编码当前帧中采用 Intra32x32 预测模式的 CU 块数 暂未使用。
u32Intra16x16CuNum	编码当前帧中采用 Intra16x16 预测模式的 CU 块数 暂未使用。
u32Intra8x8CuNum	编码当前帧中采用 Intra8x8 预测模式的 CU 块数 暂未使用。
u32Intra4x4CuNum	编码当前帧中采用 Intra4x4 预测模式的 CU 块数 暂未使用。
enRefType	高级跳帧参考下的编码帧类型。
u32UpdateAttrCnt	通道属性或参数(包含 RC 参数)被设置的次数。 暂未使用。
u32StartQp	编码当前帧的 startqp 值。 暂未使用。
u32MeanQp	编码当前帧的平均 QP 值。 暂未使用。
bPSkip	标识当前帧是否为 pskip 帧。 暂未使用。

【注意事项】

无

5.37 VENC_STREAM_INFO_JPEG_S

【说明】

定义 JPEG/MJPEG 协议码流特征信息。

【定义】

```
typedef struct rkVENC_STREAM_INFO_JPEG_S {
    RK_U32 u32PicBytesNum;
    RK_U32 u32UpdateAttrCnt;
    RK_U32 u32Qfactor;
} VENC_STREAM_INFO_JPEG_S;
```

【成员】

成员名称	描述
u32PicBytesNum	一帧 JPEG 码流大小，以字节 (byte) 为单位。 暂未使用。
u32UpdateAttrCnt	通道属性或参数(包含 RC 参数)被设置的次数。 暂未使用。
u32Qfactor	编码当前帧的Qfactor。 暂未使用。

【注意事项】

无

5.38 VENC_STREAM_INFO_PRORES_S

【说明】

定义 PRORES 协议码流特征信息。

【定义】

```
typedef struct rkVENC_STREAM_INFO_PRORES_S {
    RK_U32 u32PicBytesNum;
    RK_U32 u32UpdateAttrCnt;
} VENC_STREAM_INFO_PRORES_S;
```

【成员】

成员名称	描述
u32PicBytesNum	编码当前帧的字节 (Byte) 数。
u32UpdateAttrCnt	通道属性或参数被设置的次数。

【注意事项】

无

5.39 VENC_STREAM_ADVANCE_INFO_H264_S

【说明】

定义 H.264 协议码流高级特征信息。

【定义】

```
typedef struct rkVENC_STREAM_ADVANCE_INFO_H264_S {
    RK_U32      u32ResidualBitNum;
    RK_U32      u32HeadBitNum;
    RK_U32      u32MadiVal;
    RK_U32      u32MadpVal;
    RK_DOUBLE   dPSNRVal;
    RK_U32      u32MseLcuCnt;
    RK_U32      u32MseSum;
    VENC_SSE_INFO_S stSSEInfo[8];
    RK_U32      u32QpHstgrm[VENC_QP_SGRM_NUM];
    RK_U32      u32MoveScene16x16Num;
    RK_U32      u32MoveSceneBits;
} VENC_STREAM_ADVANCE_INFO_H264_S;
```

【成员】

成员名称	描述
u32ResidualBitNum	编码当前帧残差 (bit) 数。 暂未使用。
u32HeadBitNum	编码当前帧头信息的 (bit) 数。 暂未使用。
u32MadiVal	编码当前帧空域纹理复杂度 Madi 值。 暂未使用。
u32MadpVal	编码当前帧时域运动复杂度 Madp 值。 暂未使用。
dPSNRVal	编码当前帧的 PSNR (峰值信噪比) 值。 暂未使用。
u32MseLcuCnt	编码当前帧中 LCU 个数。 暂未使用。
u32MseSum	编码当前帧中 MSE (均方差) 值。 暂未使用。
stSSEInfo[8]	编码当前帧中 8 个区域的 SSE (和方差) 值。 暂未使用。
u32QpHstgrm	编码当前帧 Qp 直方图。 暂未使用。
u32MoveScene16x16Num	判断为图像前景的 16x16 块的数目，需要开启前景宏块级码控制。 暂未使用。
u32MoveSceneBits	判断为图像前景区域编码 bit 数，需要开启前景宏块级码控制。 暂未使用。

【注意事项】

无

5.40 VENC_STREAM_ADVANCE_INFO_H265_S

【说明】

定义 H.265 协议码流高级特征信息。

【定义】

```
typedef struct rkVENC_STREAM_ADVANCE_INFO_H265_S {
    RK_U32      u32ResidualBitNum;
    RK_U32      u32HeadBitNum;
    RK_U32      u32MadiVal;
    RK_U32      u32MadpVal;
    RK_DOUBLE   dPSNRVal;
    RK_U32      u32MseLcuCnt;
    RK_U32      u32MseSum;
    VENC_SSE_INFO_S stSSEInfo[8];
    RK_U32      u32QpHstgrm[VENC_QP_SGRM_NUM];
    RK_U32      u32MoveScene32x32Num;
    RK_U32      u32MoveSceneBits;
} VENC_STREAM_ADVANCE_INFO_H265_S;
```

【成员】

成员名称	描述
u32ResidualBitNum	编码当前帧残差 (bit) 数。 暂未使用。
u32HeadBitNum	编码当前帧头信息的 (bit) 数。 暂未使用。
u32MadiVal	编码当前帧空域纹理复杂度 Madi 值。 暂未使用。
u32MadpVal	编码当前帧时域运动复杂度 Madp 值。 暂未使用。
dPSNRVal	编码当前帧的 PSNR (峰值信噪比) 值。 暂未使用。
u32MseLcuCnt	编码当前帧中 LCU 个数。 暂未使用。
u32MseSum	编码当前帧中 MSE (均方差) 值。 暂未使用。
stSSEInfo[8]	编码当前帧中 8 个区域的 SSE (和方差) 值。 暂未使用。
u32QpHstgrm	编码当前帧 Qp 直方图。 暂未使用。
u32MoveScene32x32Num	判断为图像前景的 32x32 块的数目，需要开启前景宏块级码控制。 暂未使用。
u32MoveSceneBits	判断为图像前景区域编码 bit 数，需要开启前景宏块级码控制。 暂未使用。

【注意事项】

无

5.41 VENC_STREAM_ADVANCE_INFO_JPEG_S

【说明】

定义 JPEG/MJPEG 协议码流高级特征信息。

【定义】

```
typedef struct rkVENC_STREAM_ADVANCE_INFO_H264_S {
    // RK_U32 u32Reserved;
} VENC_STREAM_ADVANCE_INFO_H264_S;
```

【成员】

无

【注意事项】

无

5.42 VENC_STREAM_ADVANCE_INFO_PRORES_S

【说明】

定义 PRORES 协议码流高级特征信息。

【定义】

```
typedef struct rkVENC_STREAM_ADVANCE_INFO_PRORES_S {
    // RK_U32 u32Reserved;
} VENC_STREAM_ADVANCE_INFO_PRORES_S;
```

【成员】

无

【注意事项】

无

5.43 VENC_SSE_INFO_S

【说明】

定义 H264/H265 协议 SSE 信息。

【定义】

```
typedef struct rkVENC_SSE_INFO_S {
    RK_BOOL bSSEEn;
    RK_U32 u32SSEVal;
} VENC_SSE_INFO_S;
```

【成员】

成员名称	描述
bSSEEn	区域 SSE 使能。 暂未使用。
u32SSEVal	区域 SSE 值。 暂未使用。

【注意事项】

- 所有成员均为动态属性。

5.44 VENC_JPEG_PARAM_S

【说明】

定义 JPEG 协议编码通道高级参数结构体。

【定义】

```
typedef struct rkVENC_JPEG_PARAM_S {
    RK_U32 u32Qfactor;
    RK_U8 u8YQt[64];
    RK_U8 u8CbQt[64];
    RK_U8 u8CrQt[64];
    RK_U32 u32MCUPerECS;
} VENC_JPEG_PARAM_S;
```

【成员】

成员名称	描述
u32Qfactor	量化表因子，具体含义请参见 RFC2435 协议，系统默认为 70。取值范围：[0, 99]。
u8YQt	Y 量化表。取值范围：[1, 255]，系统默认为128。
u8CbQt	Cb 量化表。取值范围：[1, 255]，系统默认为128。
u8CrQt	Cr 量化表。取值范围：[1, 255]，系统默认为128。
u32MCUPerECS	每个 ECS 中包含多少个 MCU，系统默认为 0，表示不划分 Ecs。

【注意事项】

- 所有成员均为动态属性。
- u32Qfactor为0时Y/Cb/Cr分量使用自定义量化表u8YQt/u8CbQt/u8CrQt。
- u32Qfactor为正数时量化参数只参考u32Qfactor，此时u8YQt/u8CbQt/u8CrQt自定义量化表无效。

5.45 VENC_ROI_ATTR_S

【说明】

定义编码感兴趣区域信息。

【定义】

```
typedef struct rkVENC_ROI_ATTR_S {
    RK_U32 u32Index;
    RK_BOOL bEnable;
    RK_BOOL bAbsQp;
    RK_S32 s32Qp;
    RK_BOOL bIntra;
    RECT_S stRect;
} VENC_ROI_ATTR_S;
```

【成员】

成员名称	描述
u32Index	ROI 区域的索引，系统支持的索引范围为[0,7]，不支持超出这个范围的索引。
bEnable	是否使能这个 ROI 区域。
bAbsQp	ROI 区域 QP 模式。 RK_FALSE：相对QP。 RK_TRUE：绝对QP。
s32Qp	QP 值。 当bAbsQp模式为相对QP，s32Qp 为 QP 相对于该帧qp值的偏移，s32Qp 范围[-51,51]； 当bAbsQp模式为绝对QP，s32Qp 为宏块 QP 值，s32Qp 范围[1,51]。
bIntra	ROI区域是否强制编码为I块。 RK_FALSE：不开启。 RK_TRUE：强制编码为I块。
RECT_S	ROI 区域。 s32X、s32Y、u32Width、u32Height 必须是 16 对齐。

【注意事项】

- 仅针对H264/H265有效。
- s32Qp最终设置的值需要遵循[VENC_RC_PARAM_S](#)中qp的设置范围，即如果是I帧，此区域最终的qp值会被限制在 [u32MinIqp, u32MaxIqp]，如果是P帧，此区域最终的qp值会被限制在[u32MinPqp, u32MaxPqp]。
- 所有成员均为动态属性。

5.46 VENC_CHN_PARAM_S

【说明】

定义 Venc 通道参数结构体。

【定义】

```
typedef struct rkVENC_CHN_PARAM_S {
    RK_BOOL bColor2Grey;
    RK_U32 u32Priority;
    RK_U32 u32MaxStrmCnt;
    RK_U32 u32PollWakeUpFrmCnt;
    VENC_CROP_INFO_S stCropCfg;
    VENC_FRAME_RATE_S stFrameRate;
} VENC_CHN_PARAM_S;
```

【成员】

成员名称	描述
bColor2Grey	开启或关闭一个通道的彩转灰功能。 (暂未使用，可不设置)
u32Priority	编码通道优先级参数。 (暂未使用，可不设置)
u32MaxStrmCnt	最大码流缓存帧数。 (暂未使用，可不设置)
u32PollWakeUpFrmCnt	当通道使用超时或阻塞获取码流，编码指定的帧 u32PollWakeUpFrmCnt 之后唤醒阻塞接口。 (暂未使用，可不设置)
stCropCfg	通道截取 (Crop) 参数，包含缩放功能。使用说明详见： VENC_CROP_INFO_S 结构体定义。
stFrameRate	通道帧率控制参数。

【注意事项】

- 所有成员均为动态属性。

5.47 VENC_FRAME_RATE_S

【说明】

定义通道帧率控制参数。

【定义】

```
typedef struct rkVENC_FRAME_RATE_S {
    RK_BOOL bEnable;
    RK_S32 s32SrcFrmRateNum;
    RK_S32 s32SrcFrmRateDen;
    RK_S32 s32DstFrmRateNum;
    RK_S32 s32DstFrmRateDen;
} VENC_FRAME_RATE_S;
```

【成员】

成员名称	描述
bEnable	是否使能帧率设置。 RK_TRUE:使能帧率设置 RK_FALSE:关闭帧率设置
s32SrcFrmRateNum	输入帧率分子
s32SrcFrmRateDen	输入帧率分母
s32DstFrmRateNum	输出帧率分子
s32DstFrmRateDen	输出帧率分母

【注意事项】

- 所有成员均为动态属性。

5.48 VENC_CROP_INFO_S

【说明】

定义通道截取 (Crop) 参数 (包含缩放 (scale) 功能)。

【定义】

```
typedef struct rkVENC_CROP_INFO_S {
    VENC_CROP_TYPE_E enCropType;
    RECT_S stCropRect;
    VENC_SCALE_RECT_S stScaleRect;
} VENC_CROP_INFO_S;
```

【成员】

成员名称	描述
enCropType	VENC_CROP_NONE: 不开启 VENC_CROP_ONLY: 开启裁剪功能, 由stCropRect参数控制 VENC_CROP_SCALE: 开启裁剪缩放功能, 由stScaleRect参数控制 VENC_CROP_BUTT
stCropRect	裁剪的区域。 stCropRect.s32X: 裁剪起点X, 必须2像素对齐。 stCropRect.s32Y: 裁剪起点Y。 stCropRect.u32Width: 裁剪图像宽, 必须2像素对齐。 stCropRect.u32Height: 裁剪图像高, 必须2像素对齐。
stScaleRect	裁剪缩放控制。 RECT_S stSrc: 裁剪缩放源区域 stSrc.s32X: 裁剪缩放源起点X, 必须2像素对齐。 stSrc.s32Y: 裁剪缩放源起点Y, 必须2像素对齐。 stSrc.u32Width: 裁剪缩放源图像宽, 必须2像素对齐。 stSrc.u32Height: 裁剪缩放源图像高, 必须2像素对齐。 RECT_S stDst: 裁剪缩放目标区域。 stDst.s32X: 裁剪缩放目标起点X, 必须2像素对齐。 stDst.s32Y: 裁剪缩放目标起点Y, 必须2像素对齐。 stDst.u32Width: 裁剪缩放目标图像宽, 必须2像素对齐。 stDst.u32Height: 裁剪缩放目标图像高, 必须2像素对齐。 如需要将1920x1080的图像在10, 20的像素点开始裁剪100,200大小的图然后缩放到1280x720。 则设置参数: stSrc.s32X=10, stSrc.s32Y=20, stSrc.u32Width=100, stSrc.u32Height=200, stDst.s32X=0, stDst.s32Y=0, stDst.u32Width=1280, stDst.u32Height=720。 如需要将1920x1080的图像缩放到1280x720。 则设置参数: stSrc.s32X=0, stSrc.s32Y=0, stSrc.u32Width=1920, stSrc.u32Height=1080, stDst.s32X=0, stDst.s32Y=0, stDst.u32Width=1280, stDst.u32Height=720。

【注意事项】

- 源图像大小由送入编码通道的视频帧决定。
- 裁剪功能与裁剪缩放功能分别由参数stCropRect、stScaleRect决定。
- RK3588缩放倍数不能超过8倍, RV1109/RV1126/RK356X缩放倍数不能超过16倍。
- 建议输出分辨率都采用16对齐。
- 裁剪缩放与旋转或者镜像同时存在时, 处理顺序为裁剪缩放>旋转>镜像。
- 开启裁剪缩放功能, 只需缩放时, 将stSrc.u32Width、stSrc.u32Height设置为0, 此时送入编码通道的图像会被缩放到stDst设置的分辨率。
- 所有成员均为动态属性。

【举例】

- 裁剪功能: 将送入编码通道的图像从坐标为X=10 Y=20的像素点位置裁剪出100x200的图像输出。

```
VENC_CHN_PARAM_S stParam;
RK_MPI_VENC_GetChnParam(u32Ch, &stParam);
stParam.stCropCfg.enCropType = VENC_CROP_ONLY;
stParam.stCropCfg.stCropRect.s32X = 10;
stParam.stCropCfg.stCropRect.s32Y = 20;
stParam.stCropCfg.stCropRect.u32Height = 100;
stParam.stCropCfg.stCropRect.u32Width = 200;
RK_MPI_VENC_SetChnParam(u32Ch, &stParam);
```

- 缩放功能: 将送入编码通道的图像缩放为640x360的图像输出, 将源区域信息设置为0。

```
VENC_CHN_PARAM_S stParam;
RK_MPI_VENC_GetChnParam(u32Ch, &stParam);
stParam.stCropCfg.enCropType = VENC_CROP_SCALE;
stParam.stCropCfg.stScaleRect.stSrc.s32X = 0;
stParam.stCropCfg.stScaleRect.stSrc.s32Y = 0;
stParam.stCropCfg.stScaleRect.stSrc.u32Width = 0;
stParam.stCropCfg.stScaleRect.stSrc.u32Height = 0;
stParam.stCropCfg.stScaleRect.stDst.s32X = 0;
stParam.stCropCfg.stScaleRect.stDst.s32Y = 0;
stParam.stCropCfg.stScaleRect.stDst.u32Width = 640;
stParam.stCropCfg.stScaleRect.stDst.u32Height = 360;
```

- 裁剪缩放功能：将送入编码通道的图像从坐标为X=10 Y=20的像素点位置裁剪出1280x720的图像，然后缩放为640x360的图像输出。

```
VENC_CHN_PARAM_S stParam;
RK_MPI_VENC_GetChnParam(u32Ch, &stParam);
stParam.stCropCfg.enCropType = VENC_CROP_SCALE;
stParam.stCropCfg.stScaleRect.stSrc.s32X = 10;
stParam.stCropCfg.stScaleRect.stSrc.s32Y = 20;
stParam.stCropCfg.stScaleRect.stSrc.u32Width = 1280;
stParam.stCropCfg.stScaleRect.stSrc.u32Height = 720;
stParam.stCropCfg.stScaleRect.stDst.s32X = 0;
stParam.stCropCfg.stScaleRect.stDst.s32Y = 0;
stParam.stCropCfg.stScaleRect.stDst.u32Width = 640;
stParam.stCropCfg.stScaleRect.stDst.u32Height = 360;
```

5.49 VENC_GOP_MODE_E

【说明】

定义 H.264/H.265 GOP 类型。

【定义】

```
typedef enum rkVENC_GOP_MODE_E {
    VENC_GOPMODE_INIT = 0,
    VENC_GOPMODE_NORMALP,
    VENC_GOPMODE_TSVC2,
    VENC_GOPMODE_TSVC3,
    VENC_GOPMODE_TSVC4,
    VENC_GOPMODE_SMARTP,
    VENC_GOPMODE_BUTT
} VENC_GOP_MODE_E;
```

【成员】

成员名称	描述
VENC_GOPMODE_INIT	无设置。系统默认NORMALP。
VENC_GOPMODE_NORMALP	编码单参考帧 P 帧 .
VENC_GOPMODE_TSVC2	TSVC2。 TSVC-2提供两层编码，帧率可以在1/2和全帧率之间变化
VENC_GOPMODE_TSVC3	TSVC3。 TSVC-3提供三层编码，帧率可以在1/4, 1/2, 3/4和全帧率之间变化。
VENC_GOPMODE_TSVC4	TSVC4。 TSVC-4提供四层编码，帧率可以在1/8,1/4,3/8,1/2,5/8,3/4,7/8和全帧率之间变化。
VENC_GOPMODE_SMARTP	编码智能 P 帧。
VENC_GOPMODE_BUTT	最大值。

【注意事项】

无

5.50 VENC_RC_ADVPARAM_S

【说明】

定义RC模块的高级参数，此接口会包含与码流控制算法无关的功能，并且未来版本还有可能扩展。

【定义】

```
typedef struct rkVENC_RC_ADVPARAM_S {
    RK_U32 u32ClearStatAfterSetAttr;
} VENC_RC_ADVPARAM_S;
```

【成员】

成员名称	描述
u32ClearStatAfterSetAttr	设置新的通道码率后，是否清除码率控制的统计信息，默认值为 1。 0：设置通道码率后不清除 RC 的帧率和码率统计信息； 1：设置通道码率后清除 RC 的帧率和码率统计信息； 注意：默认为1，当前只支持1。

【注意事项】

- 所有成员均为动态属性。

5.51 VENC_SUPERFRAME_CFG_S

【说明】

超大帧处理策略参数。

【定义】

```
typedef struct rkVENC_SUPERFRAME_CFG_S {
    VENC_SUPERFRM_MODE_E enSuperFrmMode;
    RK_U32 u32SuperIFrmBitsThr;
    RK_U32 u32SuperPFrmBitsThr;
    RK_U32 u32SuperBfrmBitsThr;
    VENC_RC_PRIORITY_E enRcPriority;
} VENC_SUPERFRAME_CFG_S;
```

【成员】

成员名称	描述
enSuperFrmMode	超大帧处理模式，默认为 SUPERFRM_NONE。
u32SuperIFrmBitsThr	I帧超大阈值，单位bit。取值范围：大于等于 0。
u32SuperPfrmBitsThr	P帧超大阈值，单位bit。取值范围：大于等于 0。
u32SuperBFrmBitsThr	B帧超大阈值，单位bit。取值范围：大于等于 0。
enRcPriority	码率控制优先级，默认为 VENC_RC_PRIORITY_BITRATE_FIRST。

【注意事项】

- 所有成员均为动态属性。

5.52 VENC_SUPERFRM_MODE_E

【说明】

定义码率控制中超大帧处理模式。

【定义】

```
typedef enum rkRC_SUPERFRM_MODE_E {
    SUPERFRM_NONE = 0,
    SUPERFRM_DISCARD,
    SUPERFRM_REENCODE,
    SUPERFRM_BUTT
} VENC_SUPERFRM_MODE_E;
```

【成员】

成员名称	描述
SUPERFRM_NONE	无特殊策略。
SUPERFRM_DISCARD	丢弃超大帧。
SUPERFRM_REENCODE	重编超大帧。
SUPERFRM_BUTT	最大值。

【注意事项】

无

5.53 VENC_RC_PRIORITY_E

【说明】

定义超大帧重编优先级枚举。

【定义】

```
typedef enum rkVENC_RC_PRIORITY_E {
    VENC_RC_PRIORITY_BITRATE_FIRST = 1,
    VENC_RC_PRIORITY_FRAMEBITS_FIRST,
    VENC_RC_PRIORITY_BUTT,
} VENC_RC_PRIORITY_E;
```

【成员】

成员名称	描述
VENC_RC_PRIORITY_BITRATE_FIRST	目标码率优先。
VENC_RC_PRIORITY_FRAMEBITS_FIRST	超大帧阈值优先。
VENC_RC_PRIORITY_BUTT	最大值。

【注意事项】

此优先级只在超大帧重编时有效。

5.54 VENC_FRAMELOST_S

【说明】

瞬时码率超过阈值时的丢帧策略参数。

【定义】

```
typedef struct rkVENC_FRAMELOST_S {
    RK_BOOL bFrmLostOpen;
    RK_U32 u32FrmLostBpsThr;
    VENC_FRAMELOST_MODE_E enFrmLostMode;
    RK_U32 u32EncFrmGaps;
} VENC_FRAMELOST_S;
```

【成员】

成员名称	描述
bFrmLostOpen	丢帧开关。 RK_TRUE: 开; RK_FALSE: 关。 默认关闭。
u32FrmLostBpsThr	丢帧阈值百分比。 丢帧码率值=(bps_max * (100 + u32FrmLostBpsThr) / (float)100)。 取值范围: >=0。
enFrmLostMode	丢帧策略模式。 FRMLOST_NORMAL: 瞬时码率超过阈值时正常丢帧; FRMLOST_PSKIP: 瞬时码率超过阈值时编码 pskip 帧。
u32EncFrmGaps	最大允许连续丢帧帧数。0表示全丢。建议设置为1或2。

【注意事项】

- 所有成员均为动态属性。

5.55 VENC_INTRA_REFRESH_S

【说明】

P 帧刷 Islice 控制参数。

【定义】

```

typedef struct rkVENC_INTRA_REFRESH_S {
    RK_BOOL      bRefreshEnable;
    VENC_INTRA_REFRESH_MODE_E enIntraRefreshMode;
    RK_U32       u32RefreshNum;
    RK_U32       u32ReqIQP;
} VENC_INTRA_REFRESH_S;

```

【成员】

成员名称	描述
bRefreshEnable	是否使能刷slice功能。 RK_TRUE: 使能； RK_FALSE: 不使能。 默认为不使能。
enIntraRefreshMode	I宏块刷新模式，分为按行刷新和按列刷新。 INTRA_REFRESH_ROW: 按行刷新； INTRA_REFRESH_COLUMN: 按列刷新。
u32RefreshNum	每次I宏块刷新行数或者列数，可以通过这个变量控制刷新的速度及码流的平稳程度。刷新行数或者列数越多，刷新的速度越快，但是码流平稳度越差；刷新的行数或者列数越少，刷新的速度越慢，但是码流平稳度越好。
u32ReqIQP	I帧QP值，在帧内刷新模式，可能方案还需要插入IDR帧，设置I帧QP用于控制插入的IDR帧的质量，质量越好IDR帧大小越大；质量越差IDR帧大小越小。 取值范围：[1, 51]。 暂不支持该参数设置。

【注意事项】

- 所有成员均为动态属性。

5.56 VENC_HIERARCHICAL_QP_S

【说明】

QP分层参数结构体。

【定义】

```

typedef struct rkVENC_HIERARCHICAL_QP_S {
    RK_BOOL  bHierarchicalQPEn;
    RK_S32   s32HierarchicalQPDelta[4];
    RK_S32   s32HierarchicalFrameNum[4];
} VENC_HIERARCHICAL_QP_S;

```

【成员】

成员名称	描述
bHierarchicalQpEn	QP分层是否使能。 RK_TRUE：使能； RK_FALSE：不使能。 默认为不使能。
s32HierarchicalQpDelta	第0层P帧的相对于每一层的QP差值QpDelta，即QpDelta=P0的QP-当前层的QP，正数表示当前层的QP比第0层P帧的QP小，输出的层的P帧质量越好，数据越大。 取值范围：[-10, 10]。 默认值：0。
s32HierarchicalFrameNum	每一层对应P帧的数目。 取值范围：[0, 5]。 默认值：0。

【注意事项】

- 所有成员均为动态属性。

5.57 VENC_FILTER_S

【说明】

定义输入源滤波结构体。

【定义】

```
typedef struct rkVENC_FILTER_S {
    RK_U32 u32StrengthI;
    RK_U32 u32StrengthP;
} VENC_FILTER_S;
```

【成员】

成员名称	描述
u32StrengthI	默认为 0。 取值范围：0-3。
u32StrengthP	默认为 0。 取值范围：0-3。

【注意事项】

- 0表示无滤波；滤波强度越大图像越模糊，码率越省。
- 所有成员均为动态属性。

5.58 VENC_H264_INTRA_PRED_S

【说明】

定义H.264协议编码通道帧内预测结构体。

【定义】

```
typedef struct rkVENC_H264_INTRA_PRED_S {
    RK_U32 constrained_intra_pred_flag;
} VENC_H264_INTRA_PRED_S
```

【成员】

成员名称	描述
constrained_intra_pred_flag	默认为 0。 取值范围： 0 或 1。

【注意事项】

- 以上参数具体含义请参见H.264协议。
- 所有成员均为动态属性。

5.59 VENC_H264_TRANS_S

【说明】

定义H.264协议编码通道变换、量化结构体。

【定义】

```
typedef struct rkVENC_H264_TRANS_S {
    RK_U32    u32TransMode;
    RK_BOOL   bScalingListValid;
    RK_U8     InterScalingList8X8[64];
    RK_U8     IntraScalingList8X8[64];
    RK_S32   chroma_qp_index_offset;
} VENC_H264_TRANS_S;
```

【成员】

成员名称	描述
u32TransMode	帧内、帧间预测的变换模式： 0：支持 4x4, 8x8 变换，high profile,svc-t 支持。 1：4x4 变换。 系统默认值为 0。 动态属性。
bScalingListValid	InterScalingList8x8、IntraScalingList8x8是否有效标识，只在high profile,svc-t下才有意义。 取值范围：0 或 1。 0：无效； 1：有效。只支持配置 0。
InterScalingList8X8[64]	帧间预测8x8的量化表，在high profile,svc-t下，用户可以使用自己的量化表，保留，暂不使用。 取值范围：[1, 255]。
IntraScalingList8X8[64]	帧内预测 8x8 的量化表，在 high profile,svc-t 下，用户可以使用自己的量化表，保留，暂不使用。 取值范围：[1, 255]。
chroma_qp_index_offset	具体含义请参见 H.264 协议。 系统默认值为 -6。 取值范围：[-12, 12]。 动态属性。

【注意事项】

- 以上参数具体含义请参见 H.264 协议。
- 不支持量化表，因此不支持设置 bScalingListValid、InterScalingList8X8[64]、IntraScalingList8X8[64]。

5.60 VENC_H264_ENTROPY_S

【说明】

定义H.264协议编码通道熵编码结构体。

【定义】

```
typedef struct rkVENC_H264_ENTROPY_S {  
    RK_U32 u32EntropyEncMode;  
    RK_U32 cabac_init_idc;  
} VENC_H264_ENTROPY_S;
```

【成员】

成员名称	描述
u32EntropyEncMode	熵编码模式。0: cavlc 1: cabac。系统默认值为1。
cabac_init_idc	取值范围: [0, 2], 默认值 0, 具体含义请参见H.264协议。

【注意事项】

- 所有成员均为动态属性。

5.61 VENC_H264_DBLK_S

【说明】

定义H.264协议编码通道 Dblk 结构体。

【定义】

```
typedef struct rkVENC_H264_DBLK_S {  
    RK_U32 disable_deblocking_filter_idc;  
    RK_S32 slice_alpha_c0_offset_div2;  
    RK_S32 slice_beta_offset_div2;  
} VENC_H264_DBLK_S;
```

【成员】

成员名称	描述
disable_deblocking_filter_idc	取值范围: [0, 2], 默认值 0, 具体含义请参见 H.264 协议。
slice_alpha_c0_offset_div2	取值范围: [-6, 6], 默认值 0, 具体含义请参见 H.264 协议。
slice_beta_offset_div2	取值范围: [-6, 6], 默认值 0, 具体含义请参见 H.264 协议。

【注意事项】

- 所有成员均为动态属性。

5.62 VENC_H264_VUI_S

【说明】

定义H.264协议编码通道 Vui 结构体。

【定义】

```
typedef struct rkVENC_H264_VUI_S {  
    VENC_VUI_ASPECT_RATIO_S      stVuiAspectRatio;  
    VENC_VUI_H264_TIME_INFO_S    stVuiTimeInfo;  
    VENC_VUI_VIDEO_SIGNAL_S     stVuiVideoSignal;  
    VENC_VUI_BITSTREAM_RESTRIC_S stVuiBitstreamRestric;  
} VENC_H264_VUI_S;
```

【成员】

成员名称	描述
VENC_VUI_ASPECT_RATIO_S	具体含义请参见 H.264 协议。暂不支持该参数设置。
VENC_VUI_H264_TIME_INFO_S	具体含义请参见 H.264 协议。暂不支持该参数设置。
VENC_VUI_VIDEO_SIGNAL_S	具体含义请参见 H.264 协议。部分参数支持。
VENC_VUI_BITSTREAM_RESTRICT_S	具体含义请参见 H.264 协议。暂不支持该参数设置。

【注意事项】

- 所有成员均为动态属性。

5.63 VENC_VUI_ASPECT_RATIO_S

【说明】

定义 H.264/H.265 协议编码通道 Vui 中 AspectRatio 信息的结构体。

【定义】

```
typedef struct rkVENC_VUI_ASPECT_RATIO_S {
    RK_U8 aspect_ratio_info_present_flag;
    RK_U8 aspect_ratio_idc;
    RK_U8 overscan_info_present_flag;
    RK_U8 overscan_appropriate_flag;
    RK_U16 sar_width;
    RK_U16 sar_height;
} VENC_VUI_ASPECT_RATIO_S;
```

【成员】

成员名称	描述
aspect_ratio_info_present_flag	具体含义请参见 H.264/H.265 协议，系统默认为 0。取值范围：0 或 1。 暂不支持该参数设置。
aspect_ratio_idc	具体含义请参见 H.264/H.265 协议，系统默认为 0。取值范围：[0,255],17~254 保留。 暂不支持该参数设置。
overscan_info_present_flag	具体含义请参见 H.264/H.265 协议，系统默认为 0。取值范围：0 或 1。 暂不支持该参数设置。
overscan_appropriate_flag	具体含义请参见 H.264/H.265 协议，系统默认为 0。取值范围：0 或 1。 暂不支持该参数设置。
sar_width	具体含义请参见 H.264/H.265 协议，系统默认为 0。取值范围：(0, 65535]，并且与 sar_height 互质。 暂不支持该参数设置。
sar_height	具体含义请参见 H.264/H.265 协议，系统默认为 0。取值范围：(0, 65535]，并且与 sar_width 互质。 暂不支持该参数设置。

【注意事项】

- 所有成员均为动态属性。

5.64 VENC_VUI_H264_TIME_INFO_S

【说明】

定义 H.264 协议编码通道 Vui 中 Time_Info 信息的结构体。

【定义】

```
typedef struct rkVENC_H264_VUI_TIME_INFO_S {  
    RK_U8 timing_info_present_flag;  
    RK_U8 fixed_frame_rate_flag;  
    RK_U32 num_units_in_tick;  
    RK_U32 time_scale;  
} VENC_VUI_H264_TIME_INFO_S;
```

【成员】

成员名称	描述
timing_info_present_flag	具体含义请参见 H.264 协议，系统默认为 1。取值范围：0 或 1。 暂不支持该参数设置。
fixed_frame_rate_flag	具体含义请参见 H.264 协议，系统默认为 1。取值范围：0 或 1。 暂不支持该参数设置。
num_units_in_tick	具体含义请参见 H.264 协议，系统默认为 1。取值范围：大于 0。 暂不支持该参数设置。
time_scale	具体含义请参见 H.264 协议，系统默认为 60。取值范围：大于 0。 暂不支持该参数设置。

【注意事项】

- 所有成员均为动态属性。

5.65 VENC_VUI_VIDEO_SIGNAL_S

【说明】

定义 H.264/H.265 协议编码通道 Vui 中信息的 Video_Signal 结构体。

【定义】

```
typedef struct rkVENC_VUI_VIDEO_SIGNAL_S {  
    RK_U8 video_signal_type_present_flag;  
    RK_U8 video_format;  
    RK_U8 video_full_range_flag;  
    RK_U8 colour_description_present_flag;  
    RK_U8 colour_primaries;  
    RK_U8 transfer_characteristics;  
    RK_U8 matrix_coefficients;  
} VENC_VUI_VIDEO_SIGNAL_S;
```

【成员】

成员名称	描述
video_signal_type_present_flag	具体含义请参见 H.264/H.265 协议，系统默认为 1。取值范围：0 或 1。 暂不支持该参数设置。
video_format	具体含义请参见 H.264/H.265 协议，系统默认为 5。取值范围：H.264:[0, 7], H.265: [0, 5]。 暂不支持该参数设置。
video_full_range_flag	0: limit color range; 1: full color range。 系统默认为 1。取值范围：0 或 1。
colour_description_present_flag	具体含义请参见 H.264/H.265 协议，系统默认为 0。该值为只读信息。取值范围：0 或 1。 暂不支持该参数设置。
colour_primaries	具体含义请参见 H.264/H.265 协议，系统默认为 0。该值为只读信息，内部根据当前图像实际信息填写。取值范围：[0, 255]。 暂不支持该参数设置。
transfer_characteristics	具体含义请参见 H.264/H.265 协议，系统默认为 0。该值为只读信息，内部根据当前图像实际信息填写。取值范围：[0, 255]。 暂不支持该参数设置。
matrix_coefficients	具体含义请参见 H.264/H.265 协议，系统默认为 0。该值为只读信息，内部根据当前图像实际信息填写。取值范围：[0, 255]。 暂不支持该参数设置。

【注意事项】

- 所有成员均为动态属性。

5.66 VENC_VUI_BITSTREAM_RESTRIC_S

【说明】

定义 H.264/H.265 协议编码通道 Vui 中信息的 Bitstream_Restriction 结构体。

【定义】

```
typedef struct rkVENC_VUI_BITSTREAM_RESTRIC_S {
    RK_U8 bitstream_restriction_flag;
} VENC_VUI_BITSTREAM_RESTRIC_S;
```

【成员】

成员名称	描述
bitstream_restriction_flag	具体含义请参见 H.264/H.265 协议，系统默认为 1。取值范围：0 或 1。 暂不支持该参数设置。

【注意事项】

- 所有成员均为动态属性。

5.67 VENC_H264_QBIAS_S

【说明】

定义 H.264 协议编码通道量化偏移值。

【定义】

```
typedef struct rkVENC_H264_QBIAS_S {  
    RK_BOOL bEnable;  
    RK_U32 u32QbiasI;  
    RK_U32 u32QbiasP;  
} VENC_H264_QBIAS_S;
```

【成员】

成员名称	描述
bEnable	取值范围：[0, 1], 默认值 0。
u32QbiasI	取值范围：[0,1023]。建议最小值、常用值、最大值：常规码控[341:683:800], 智能编码[480:500:683]
u32QbiasP	取值范围：[0,1023]。建议最小值、常用值、最大值：常规码控[200:341:450], 智能编码[240:300:341]

【注意事项】

- 所有成员均为动态属性。
- Qbias过小会造成模糊，继而引发较重的呼吸效应；过大使得图像过于清晰而码率过曝或者保留下来的细节有主观异常。

5.68 VENC_H265_TRANS_S

【说明】

定义H.265协议编码通道变换量化的结构体。

【定义】

```
typedef struct rkVENC_H265_TRANS_S {  
    RK_S32 cb_qp_offset;  
    RK_S32 cr_qp_offset;  
    RK_BOOL bScalingListEnabled;  
    RK_BOOL bScalingListTu4Valid;  
    RK_U8 InterScalingList4X4[2][16];  
    RK_U8 IntraScalingList4X4[2][16];  
    RK_BOOL bScalingListTu8Valid;  
    RK_U8 InterScalingList8X8[2][64];  
    RK_U8 IntraScalingList8X8[2][64];  
    RK_BOOL bScalingListTu16Valid;  
    RK_U8 InterScalingList16X16[2][64];  
    RK_U8 IntraScalingList16X16[2][64];  
    RK_BOOL bScalingListTu32Valid;  
    RK_U8 InterScalingList32X32[64];  
    RK_U8 IntraScalingList32X32[64];  
} VENC_H265_TRANS_S;
```

【成员】

成员名称	描述
cb_qp_offset	默认为 -6， 取值范围：[-12, 12]。
cr_qp_offset	默认为 -6， 取值范围：[-12, 12]。
bScalingListEnabled	默认为 0。保留，暂不使用。
bScalingListTu4Valid	默认为 0。保留，暂不使用。
InterScalingList4X4[2][16]	默认为 0。保留，暂不使用。
IntraScalingList4X4[2][16]	默认为 0。保留，暂不使用。
bScalingListTu8Valid	默认为 0。保留，暂不使用。
InterScalingList8X8[2]	默认为 0。保留，暂不使用。
IntraScalingList8X8[2][64]	默认为 0。保留，暂不使用。
bScalingListTu16Valid	默认为 0。保留，暂不使用。
InterScalingList16X16[2][64]	默认为 0。保留，暂不使用。
IntraScalingList16X16[2][64]	默认为 0。保留，暂不使用。
bScalingListTu32Valid	默认为 0。保留，暂不使用。
InterScalingList32X32[64]	默认为 0。保留，暂不使用。
IntraScalingList32X32[64]	默认为 0。保留，暂不使用。

【注意事项】

- 以上参数具体含义请参见H.265协议。
- 所有成员均为动态属性。

5.69 VENC_H265_ENTROPY_S

【说明】

定义H.265协议编码通道熵编码的结构体。

【定义】

```
typedef struct rkVENC_H265_ENTROPY_S {
    RK_U32 cabac_init_flag;
} VENC_H265_ENTROPY_S;
```

【成员】

成员名称	描述
cabac_init_flag	默认为0，取值范围：0或1。保留，暂不使用。

【注意事项】

- 所有成员均为动态属性。

5.70 VENC_H265_DBLOCK_S

【说明】

定义H.265协议编码通道Deblocking的结构体。

【定义】

```
typedef struct rkVENC_H265_DBLK_S {
    RK_U32 slice_deblocking_filter_disabled_flag;
    RK_S32 slice_beta_offset_div2;
    RK_S32 slice_tc_offset_div2;
} VENC_H265_DBLK_S;
```

【成员】

成员名称	描述
slice_deblocking_filter_disabled_flag	默认为 0。取值范围：0 或 1。
slice_beta_offset_div2	默认为 0。取值范围：[-6, 6]。
slice_tc_offset_div2	默认为 0。取值范围：[-6, 6]。

【注意事项】

- 以上参数具体含义请参见H.265协议。
- 所有成员均为动态属性。

5.71 VENC_H265_SAO_S

【说明】

定义H.265协议编码通道Sao的结构体。

【定义】

```
typedef struct rkVENC_H265_SAO_S {
    RK_U32 slice_sao_luma_flag;
    RK_U32 slice_sao_chroma_flag;
} VENC_H265_SAO_S;
```

【成员】

成员名称	描述
slice_sao_luma_flag	默认为 1。取值范围：0 或 1。
slice_sao_chroma_flag	默认为 1。取值范围：0 或 1。

【注意事项】

- 以上参数具体含义请参见H.265协议。
- 所有成员均为动态属性。

5.72 VENC_H265_PU_S

【说明】

定义H.265协议编码通道PU的结构体。

【定义】

```
typedef struct rkVENC_H265_PU_S {
    RK_U32 constrained_intra_pred_flag;
    RK_U32 strong_intra_smoothing_enabled_flag;
} VENC_H265_PU_S;
```

【成员】

成员名称	描述
constrained_intra_pred_flag	默认为 0。取值范围：0 或 1。保留，暂不使用。
strong_intra_smoothing_enabled_flag	默认为 1。取值范围：0 或 1。

【注意事项】

- 以上参数具体含义请参见H.265协议。
- 所有成员均为动态属性。

5.73 VENC_H265_VUI_S

【说明】

定义H.265协议编码通道 Vui 结构体。

【定义】

```
typedef struct rkVENC_H265_VUI_S {
    VENC_VUI_ASPECT_RATIO_S      stVuiAspectRatio;
    VENC_VUI_H265_TIME_INFO_S   stVuiTimeInfo;
    VENC_VUI_VIDEO_SIGNAL_S     stVuiVideoSignal;
    VENC_VUI_BITSTREAM_RESTRIC_S stVuiBitstreamRestrict;
} VENC_H265_VUI_S;
```

【成员】

成员名称	描述
VENC_VUI_ASPECT_RATIO_S	具体含义请参见 H.265 协议。暂不支持该参数设置。
VENC_VUI_H265_TIME_INFO_S	具体含义请参见 H.265 协议。暂不支持该参数设置。
VENC_VUI_VIDEO_SIGNAL_S	具体含义请参见 H.265 协议。部分参数支持。
VENC_VUI_BITSTREAM_RESTRIC_S	具体含义请参见 H.265 协议。暂不支持该参数设置。

【注意事项】

- 所有成员均为动态属性。

5.74 VENC_VUI_H265_TIME_INFO_S

【说明】

定义 H.265 协议编码通道 Vui 中 Time_Info 信息的结构体。

【定义】

```
typedef struct rkVENC_VUI_H265_TIME_INFO_S {
    RK_U32 timing_info_present_flag;
    RK_U32 num_units_in_tick;
    RK_U32 time_scale;
    RK_U32 num_ticks_poc_diff_one_minus1;
} VENC_VUI_H265_TIME_INFO_S;
```

【成员】

成员名称	描述
timing_info_present_flag	具体含义请参见 H.265 协议，系统默认为 1。取值范围：0 或 1。 暂不支持该参数设置。
fixed_frame_rate_flag	具体含义请参见 H.265 协议，系统默认为 1。取值范围：0 或 1。 暂不支持该参数设置。
num_units_in_tick	具体含义请参见 H.265 协议，系统默认为 1。取值范围：大于 0。 暂不支持该参数设置。
time_scale	具体含义请参见 H.265 协议，系统默认为 60。取值范围：大于 0。 暂不支持该参数设置。

【注意事项】

- 所有成员均为动态属性。

5.75 VENC_H265_QBIAS_S

【说明】

定义H.265协议编码通道量化偏移值。

【定义】

```
typedef struct rkVENC_H265_QBIAS_S {
    RK_BOOL bEnable;
    RK_U32 u32QbiasI;
    RK_U32 u32QbiasP;
} VENC_H265_QBIAS_S;
```

【成员】

成员名称	描述
bEnable	取值范围：[0, 1]，默认值 0。
u32QbiasI	取值范围：[0,511]。建议最小值、常用值、最大值：常规码控[90:171:341]，智能编码[128:150:171]
u32QbiasP	取值范围：[0,511]。建议最小值、常用值、最大值：常规码控[40:85:120]，智能编码[40:70:85]

【注意事项】

- 所有成员均为动态属性。
- Qbias过小会造成模糊，继而引发较重的呼吸效应；过大使得图像过于清晰而码率过曝或者保留下来的细节有主观异常。

5.76 VENC_MJPEG_PARAM_S

【说明】

定义 MJPEG 协议编码通道高级参数结构体。

【定义】

```
typedef struct rkVENC_MJPEG_PARAM_S {
    RK_U8 u8YQt[64];
    RK_U8 u8CbQt[64];
    RK_U8 u8CrQt[64];
    RK_U32 u32MCUPerECS;
} VENC_MJPEG_PARAM_S;
```

【成员】

成员名称	描述
u8YQt	Y量化表。 取值范围：[1, 255]，系统默认为 128。
u8CbQt	Cb量化表。 取值范围：[1, 255]，系统默认为 128。
u8CrQt	Cr量化表。 取值范围：[1, 255]，系统默认为 128。
u32MCUPerECS	每个 ECS 中包含多少个 MCU，系统默认为 0，表示不划分 Ecs。 最小值为0，最大值与输入格式相关，具体如下： 输入为yuv420sp: $(\text{picwidth}+15) \gg 4$ $(\text{picheight}+15) \gg 4$ 输入为yuv422sp: $(\text{picwidth}+15) \gg 4$ $(\text{picheight}+15) \gg 4 * 2$ 输入为yuv444sp: $(\text{picwidth}+15) \gg 4$ $(\text{picheight}+15) \gg 4 * 4$

【注意事项】

- 所有成员均为动态属性。

5.77 VENC_REF_PARAM_S

【说明】

定义H.264/H.265编码的高级跳帧参考参数。

【定义】

```
typedef struct rkVENC_REF_PARAM_S {
    RK_U32    u32Base;
    RK_U32    u32Enhance;
    RK_BOOL   bEnablePred;
} VENC_REF_PARAM_S;
```

【成员】

成员名称	描述
u32Base	base层的周期。取值范围：(0, +∞)。
u32Enhance	enhance层的周期。取值范围：[0, 255]。
bEnablePred	代表base层的帧是否被base层其他帧用作参考。当bEnablePred设置为RK_FALSE时，等同于u32Base设置为无限大，base层的所有帧都参考IDR帧。

【注意事项】

- 所有成员均为动态属性。

5.78 ROTATION_E

【说明】

定义旋转角度枚举。

【定义】

```

typedef enum rkROTATION_E {
    ROTATION_0    = 0,
    ROTATION_90   = 1,
    ROTATION_180  = 2,
    ROTATION_270  = 3,
    ROTATION_BUTT
} ROTATION_E;

```

【成员】

成员名称	描述
ROTATION_0	不旋转。
ROTATION_90	旋转90度。
ROTATION_180	旋转180度。
ROTATION_270	旋转270度。
ROTATION_BUTT	最大值。

【注意事项】

- 旋转与镜像同时存在时，先做旋转处理。

5.79 MIRROR_E

【说明】

定义镜像枚举。

【定义】

```

typedef enum rkROTATION_E {
    MIRROR_NONE,
    MIRROR_HORIZONTAL,
    MIRROR_VERTICAL,
    MIRROR_BOTH,
    MIRROR_BUTT
} ROTATION_E;

```

【成员】

成员名称	描述
MIRROR_NONE	不镜像。
MIRROR_HORIZONTAL	水平镜像。
MIRROR_VERTICAL	垂直镜像。
MIRROR_BOTH	水平+垂直镜像（等同旋转180）。
MIRROR_BUTT	最大值。

【注意事项】

- 旋转与镜像同时存在时，先做旋转处理。

5.80 USER_FRAME_INFO_S

【说明】

用户发送图像信息结构体。

【定义】

```
typedef struct rkUSER_FRAME_INFO_S {  
    VIDEO_FRAME_INFO_S stUserFrame;  
    USER_RC_INFO_S   stUserRcInfo;  
} USER_FRAME_INFO_S;
```

【成员】

成员名称	描述
stUserFrame	用户发送图像帧信息结构体。
stUserRcInfo	用户码控信息结构体。

【注意事项】

无。

5.81 USER_RC_INFO_S

【说明】

用户码控信息结构体。

【定义】

```
typedef struct rkUSER_RC_INFO_S {  
    RK_BOOL bQpMapValid;  
    RK_BOOL bSkipWeightValid;  
    RK_U32 u32BlkStartQp;  
    MB_BLK pMbBlkQpMap;  
    MB_BLK pMbBlkSkipWeight;  
    VENC_FRAME_TYPE_E enFrameType;  
} USER_RC_INFO_S;
```

【成员】

成员名称	描述
bQpMapValid	QPMAP模式中， Qp表是否生效。
bSkipWeightValid	QPMAP模式中， SkipWeight表是否生效。 (暂未使用)
u32BlkStartQp	QPMAP模式中， 第一个16*16块的Qp值。 (暂未使用) 取值范围： [0, 51]。
pMbBlkQpMap	QPMAP模式中， Qp表的MB_BLOCK。
pMbBlkSkipWeight	QPMAP模式中， SkipWeight表的MB_BLOCK。 (暂未使用)
enFrameType	指定当前帧编码帧类型。 (暂未使用)

【注意事项】

无。

5.82 VENC_SCENE_MODE_E

【说明】

定义SceneMode枚举。

【定义】

```
typedef enum rkVENC_SCENE_MODE_E {  
    SCENE_0 = 0,  
    SCENE_1 = 1,  
    SCENE_2 = 2,  
    SCENE_3 = 3,  
    SCENE_BUTT  
} VENC_SCENE_MODE_E;
```

【成员】

成员名称	描述
SCENE_0	典型场景静置ipc。
SCENE_1	典型场景sport dv。
SCENE_2	典型场景cvr。
SCENE_3	典型场景带ptz功能ipc。
SCENE_BUTT	最大值。

【注意事项】

- 无

5.83 VENC_DEBREATHFFECT_S

【说明】

去除呼吸效应参数结构体。

【定义】

```
typedef struct rkVENC_DEBREATHFFECT_S {  
    RK_BOOL bEnable;  
    RK_S32 s32Strength0;  
    RK_S32 s32Strength1;  
} VENC_DEBREATHFFECT_S;
```

【成员】

成员名称	描述
bEnable	去除呼吸效应是否使能。 RK_TRUE: 使能； RK_FALSE: 不使能。
s32Strength0	去除呼吸效应强度调节参数0。未使用
s32Strength1	去除呼吸效应强度调节参数1。其值越大，I帧会越大，其值越小，I帧会越小。 取值范围：[0, 35] 默认值：16

5.84 VENC_CHN_REF_BUF_SHARE_S

【说明】

参考帧共享参数结构体。

【定义】

```
typedef struct rkVENC_CHN_REF_BUF_SHARE_S {  
    RK_BOOL bEnable;  
} VENC_CHN_REF_BUF_SHARE_S;
```

【成员】

成员名称	描述
bEnable	参考帧共享是否使能。 RK_TRUE: 使能； RK_FALSE: 不使能。

5.85 VENC_COMBO_ATTR_S

【说明】

Combo属性参数结构体。

【定义】

```
typedef struct rkVENC_COMBO_ATTR_S {  
    RK_BOOL bEnable;  
    RK_S32 s32ChnId;  
} VENC_COMBO_ATTR_S;
```

【成员】

成员名称	描述
bEnable	Combo是否使能。 RK_TRUE: 使能； RK_FALSE: 不使能。
s32ChnId	Combo的数据源通道

5.86 VENC_CHN_BUF_WRAP_S

【说明】

Buf卷绕参数结构体。

【定义】

```
typedef struct rkVENC_CHN_BUF_WRAP_S {  
    RK_BOOL bEnable;  
    RK_U32 u32BufLine;  
    RK_U32 u32WrapBufferSize;  
} VENC_CHN_BUF_WRAP_S;
```

【成员】

成员名称	描述
bEnable	Buf卷绕是否使能。 RK_TRUE: 使能； RK_FALSE: 不使能。
u32BufLine	Buf卷绕行数
u32WrapBufferSize	Buf卷绕大小

5.87 VENC_SLICE_SPLIT_S

【说明】

编码通道SLICE分割结构体。

【定义】

```
typedef struct rkVENC_SLICE_SPLIT_S {
    RK_BOOL bSplitEnable;
    RK_U32 u32SplitMode;
    RK_U32 u32SplitSize;
} VENC_SLICE_SPLIT_S;
```

【成员】

成员名称	描述
bSplitEnable	slice分割是否使能。 RK_TRUE: 使能； RK_FALSE: 不使能。 默认不使能。
u32SplitMode	slice分割模式。 0: 按byte数分割，最大支持501个slice包分割，所有slice输出到一帧，仅支持单包模式获取（无slice单包信息）。 1: 按MB或者LCU分割，最大支持501个slice包分割，所有slice输出到一帧，仅支持单包模式获取（无slice单包信息）。 2: 按MB或者LCU分割，最大支持8个slice包分割，所有slice输出到一帧，支持单包模式及多包模式获取，多包模式下支持每个slice包信息的获取。 3: 按MB或者LCU分割，最大支持8个slice包分割，每次获取帧数据时，获取到一个slice包，仅支持单包模式获取。（此模式为低延时拓展预留，暂未使用）。
u32SplitSize	slice分割大小。 split_mode=0, 表示每个slice的byte数； split_mode=1/2/3, 表示每个slice的MB或LCU数。

【注意事项】

- slice分割模式为0或者1时，H264最大支持slice分割个数为500，H265最大支持slice分割个数为500个（单tile）。超过最大个数后的剩余区域会编码成一个slice。
- slice分割模式为2或者3时，H264/H265最大支持slice分割个数为8个。
- JPEG/MJPEG只支持按照宏块分割。宏块大小跟yuv格式相关：420:16x16, 422:16x8, 444:8x8。
- H264编码MB大小为16x16大小。
- H265编码LCU大小，目前除rv1106/rv1103为32x32，其他芯片均为64x64。
- 如果要获取每个slice包信息（数据长度及偏移），需要设置slice分割模式为2，并且对应模块的u32OneStreamBuffer设置为0（多包模式）。

5.88 VENC_PARAM_MOD_S

【说明】

编码相关模块参数结构体。

【定义】

```
typedef struct rkVENC_MODPARAM_S {
    VENC_MODTYPE_E enVencModType;
    VENC_MOD_VENC_S stVencModParam;
    VENC_MOD_H264E_S stH264eModParam;
    VENC_MOD_H265E_S stH265eModParam;
    VENC_MOD_JPEGE_S stJpegeModParam;
    VENC_MOD_RC_S stRcModParam;
} VENC_PARAM_MOD_S;
```

【成员】

成员名称	描述
enVencModType	设置或者获取模块参数的类型。
stVencModParam/ stH264eModParam/ stH265eModParam/ stJpegeModParam/stRcModParam	编码各个模块参数结构。

【注意事项】

- 无

5.89 VENC_MODTYPE_E

【说明】

编码相关模块参数类型。

【定义】

```
typedef enum rkVENC_MODTYPE_E {
    MODTYPE_VENC = 1,
    MODTYPE_H264E,
    MODTYPE_H265E,
    MODTYPE_JPEGE,
    MODTYPE_RC,
    MODTYPE_BUTT
} VENC_MODTYPE_E;
```

【成员】

成员名称	描述
MODTYPE_VENC	VENC模块参数类型。
MODTYPE_H264E	H264模块参数类型。
MODTYPE_H265E	H265模块参数类型。
MODTYPE_JPEGE	JPEG/MJPEG模块参数类型。
MODTYPE_RC	RC模块参数类型。
MODTYPE_BUTT	最大值。

【注意事项】

- 无

5.90 VENC_MOD_VENC_S

【说明】

VENC模块参数结构体。

【定义】

```
typedef struct rkVENC_MOD_VENC_S {  
    RK_U32 u32VencBufferCache;  
    RK_U32 u32FrameBufRecycle;  
} VENC_MOD_VENC_S;
```

【成员】

成员名称	描述
u32VencBufferCache	码流获取是否支持 cache 方式。 0: 关闭码流 Buffer Cache。 1: 打开码流 Buffer Cache。 默认值: 0。 (暂未使用)
u32FrameBufRecycle	帧存是否回收。 0: 关闭编码帧存回收。 1: 打开编码帧存回收。 默认值: 0。 (暂未使用)

【注意事项】

- 无

5.91 VENC_MOD_H264E_S

【说明】

H264模块参数结构体。

【定义】

```
typedef struct rkVENC_MOD_H264E_S {  
    RK_U32     u32OneStreamBuffer;  
    RK_U32     u32H264eMiniBufMode;  
    RK_U32     u32H264ePowerSaveEn;  
    MB_SOURCE_E enH264eMBSource;  
    RK_BOOL    bQpHstgrmEn;  
} VENC_MOD_H264E_S;
```

【成员】

成员名称	描述
u32OneStreamBuffer	编码码流帧配置模式。 0：多包模式。 1：单包模式。 默认值：1
u32H264eMiniBufMode	编码码流 buffer 配置模式。 0：码流 buffer 根据分辨率分配。 1：码流 buffer 下限为 32k，用户保证合理。 默认值：0。 (暂未使用)
u32H264ePowerSaveEn	低功耗模式。 0：关闭低功耗模式。 1：使能低功耗模式。 2：使能极低功耗模式。 默认值：0。 (暂未使用)
enH264eMBSource	参考帧和重构帧的帧存分配方式。 默认值：0。 (暂未使用)
bQpHstgrmEn	Qp 直方图输出控制模式。 默认值：0。 (暂未使用)

【注意事项】

- 无

5.92 VENC_MOD_H265E_S

【说明】

H265模块参数结构体。

【定义】

```
typedef struct rkVENC_MOD_H265E_S {
    RK_U32      u32OneStreamBuffer;
    RK_U32      u32H265eMiniBufMode;
    RK_U32      u32H265ePowerSaveEn;
    MB_SOURCE_E enH265eMBSource;
    RK_BOOL     bQpHstgrmEn;
} VENC_MOD_H265E_S;
```

【成员】

成员名称	描述
u32OneStreamBuffer	编码码流帧配置模式。 0：多包模式。 1：单包模式。 默认值：1
u32H265eMiniBufMode	编码码流 buffer 配置模式。 0：码流 buffer 根据分辨率分配。 1：码流 buffer 下限为 32k，用户保证合理。 默认值：0。 (暂未使用)
u32H265ePowerSaveEn	低功耗模式。 0：关闭低功耗模式。 1：使能低功耗模式。 2：使能极低功耗模式。 默认值：0。 (暂未使用)
enH265eMBSource	参考帧和重构帧的帧存分配方式。 默认值：0。 (暂未使用)
bQpHstgrmEn	Qp 直方图输出控制模式。 默认值：0。 (暂未使用)

【注意事项】

- 无

5.93 VENC_MOD_JPEGE_S

【说明】

JPEG/MJPEG模块参数结构体。

【定义】

```
typedef struct rkVENC_MOD_JPEGE_S {
    RK_U32 u32OneStreamBuffer;
    RK_U32 u32JpegeMiniBufMode;
    RK_U32 u32JpegClearStreamBuf;
} VENC_MOD_JPEGE_S;
```

【成员】

成员名称	描述
u32OneStreamBuffer	编码码流帧配置模式。 0：多包模式。 1：单包模式。 默认值：1。JPEG/MJPEG目前仅支持单包模式。
u32JpegeMiniBufMode	编码码流 buffer 配置模式。 0：码流 buffer 根据分辨率分配。 1：码流 buffer 下限为 32k，用户保证合理。 默认值：0。 (暂未使用)
u32JpegClearStreamBuf	JPEG 编码通道设置属性时是否清空码流 buffer。 默认值：0。 (暂未使用)

【注意事项】

- 无

5.94 VENC_MOD_RC_S

【说明】

RC模块参数结构体。

【定义】

```
typedef struct rkVENC_MOD_RC_S {  
    RK_U32 u32ClrStatAfterSetBr;  
} VENC_MOD_RC_S;
```

【成员】

成员名称	描述
u32ClrStatAfterSetBr	设置通道码率时是否清除 RC 相关统计。 默认值：0。 (暂未使用)

【注意事项】

- 无

6. VENC错误码

视频编码 API VENC错误码如下所示：

错误代码	宏定义	描述
0xA0048002	RK_ERR_VENC_INVALID_CHNID	通道 ID 超出合法范围
0xA0048003	RK_ERR_VENC_ILLEGAL_PARAM	参数超出合法范围
0xA0048004	RK_ERR_VENC_EXIST	试图申请或者创建已经存在的设备、通道或者资源
0xA0048005	RK_ERR_VENC_UNEXIST	试图使用或者销毁不存在的设备、通道或者资源
0xA0048006	RK_ERR_VENC_NULL_PTR	函数参数中有空指针
0xA0048007	RK_ERR_VENC_NOT_CONFIG	使用前未配置
0xA0048008	RK_ERR_VENC_NOT_SUPPORT	不支持的参数或者功能
0xA0048009	RK_ERR_VENC_NOT_PERM	该操作不允许，如试图修改静态配置参数
0xA004800C	RK_ERR_VENC_NOMEM	分配内存失败，如系统内存不足
0xA004800D	RK_ERR_VENC_NOBUF	分配缓存失败，如申请的数据缓冲区太大
0xA004800E	RK_ERR_VENC_BUF_EMPTY	缓冲区中无数据
0xA004800F	RK_ERR_VENC_BUF_FULL	缓冲区中数据满
0xA0048010	RK_ERR_VENC_SYS_NOTREADY	系统没有初始化或没有加载 相应模块
0xA0048012	RK_ERR_VENC_BUSY	VENC 系统忙

视频输出

[概述](#)

[VOP系统架构](#)

[模块功能](#)

[VO资源数目表](#)

[API参考](#)

[数据类型](#)

[VO错误码](#)

[UI绘制说明](#)

[虚拟图层](#)

1. 概述

视频输出（VO）模块是视频输出的封装。

2. VOP系统架构

表1：VOP输出性能

芯片	显示输出设备ID	最大分辨率
RK3568	0	4096x2160
	1	2048x1536
	2	1920x1080
RK3588	0	4096x2160(*)
	1	4096x2160(*)
	2	4096x2160
RK3576	3	2048x1536
	0	4096x2160
	1	2560x1600
	2	1920x1080

备注：

(*) 3588 输出8K时，需要输出0 和 输出1合并成一个输出设备0

图1 RK3568 VO 输出接口与VoDev连接关系

VOP Path Map

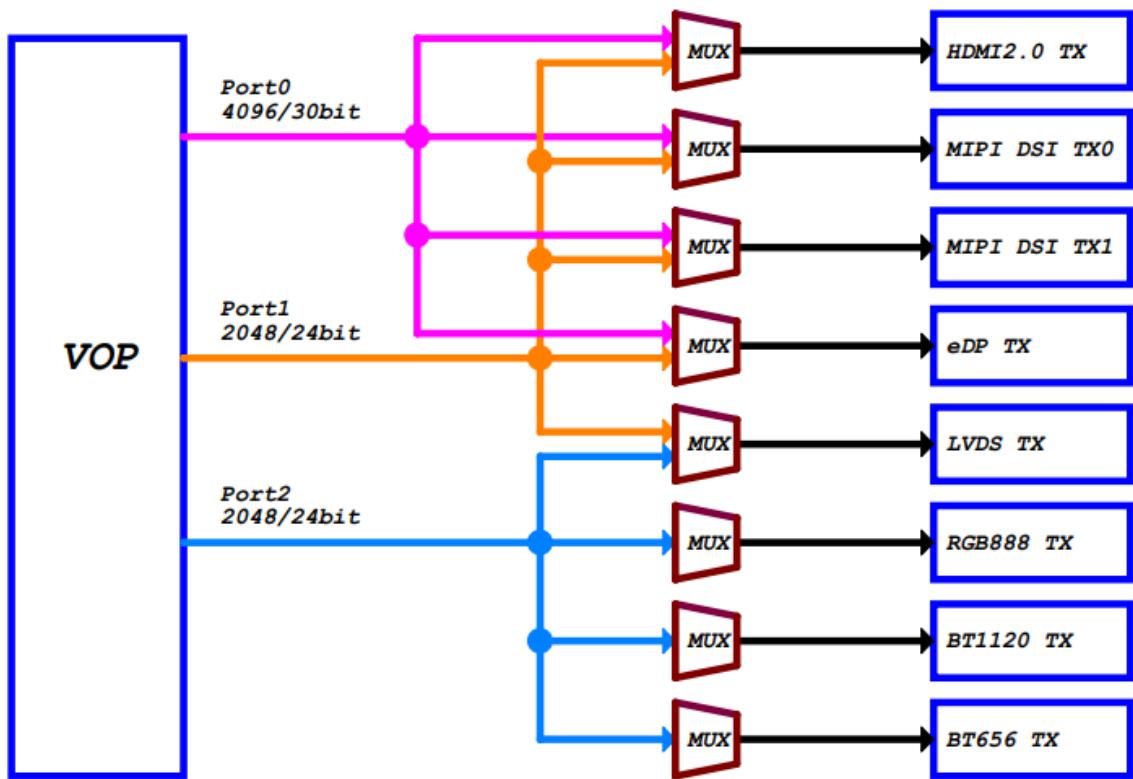


图2 RK3588 VO 输出接口与VoDev连接关系

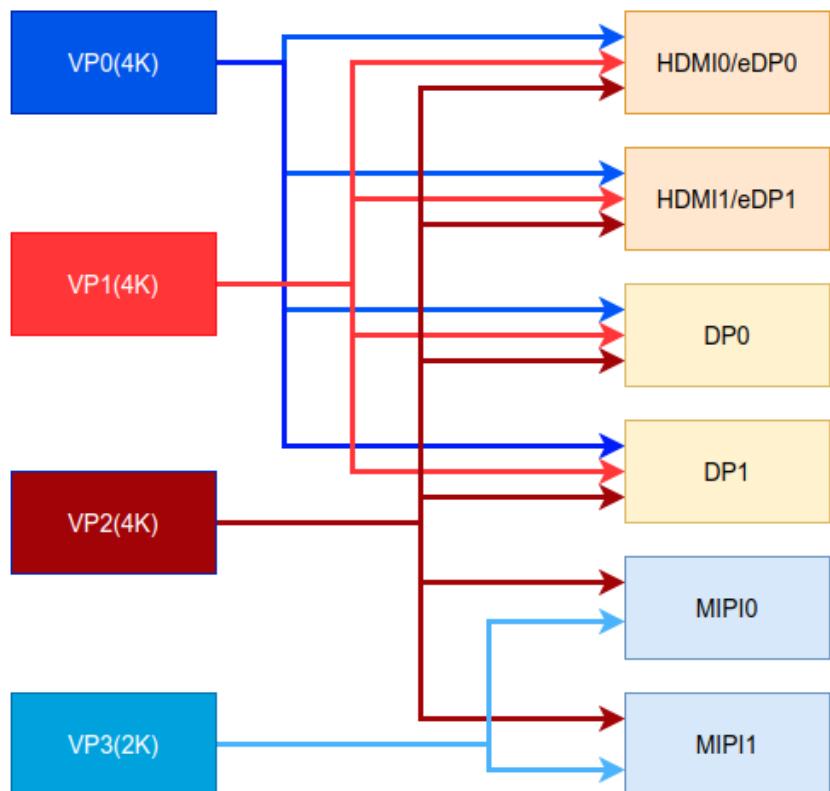
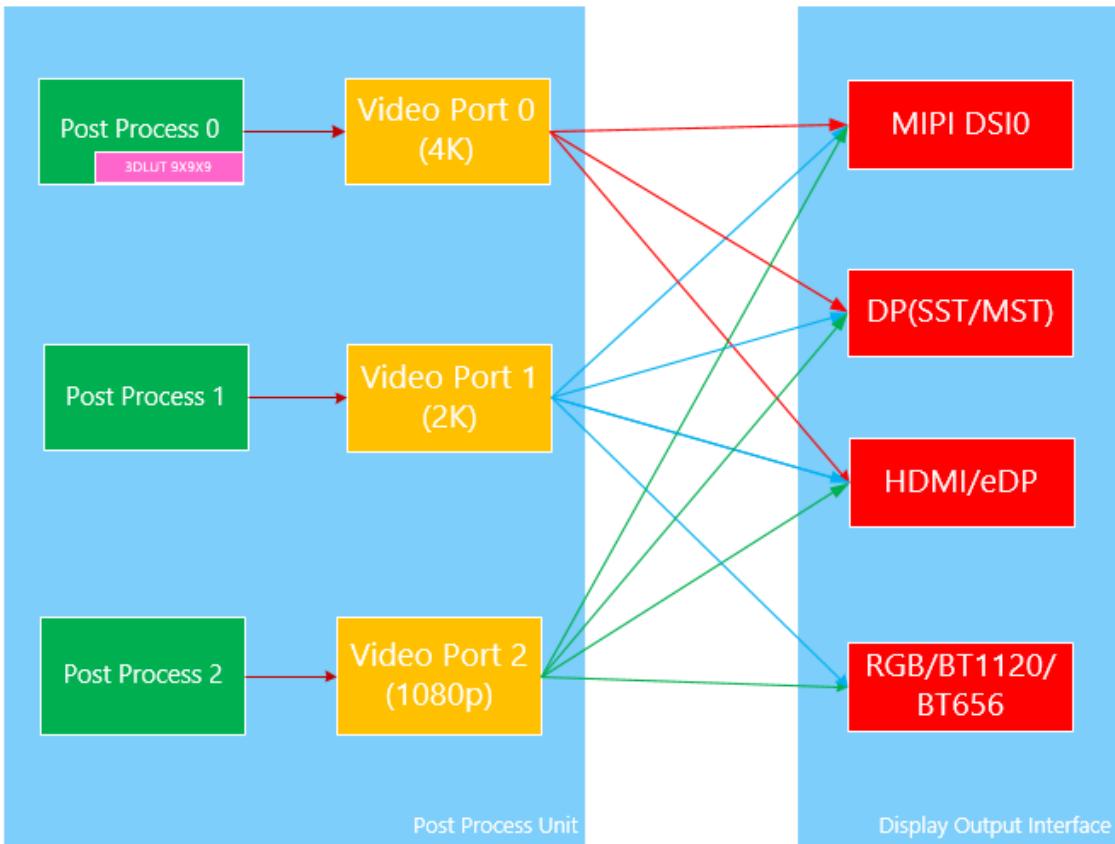


图3 RK3576 VO 输出接口与VoDev连接关系



【备注】

RK3588 绑定多个接口在同一个VoDev时，输出颜色格式必须相同。例如HDMI0和DP0同时输出RGB888。

RK3576 BT1120只支持YUV422的颜色格式。

表2：图层性能

芯片	图层名称	图层ID	缩放	压缩格式	YUV格式	RGB格式
RK3568	Cluster0	0	支持	支持(*)	支持	支持
	Cluster1	2	支持	支持(*)	支持	支持
	Esmart0	4	支持	不支持	支持	支持
	Esmart1	5	支持	不支持	支持	支持
	Smart0	6	不支持	不支持	不支持	支持
	Smart1	7	不支持	不支持	不支持	支持
RK3588	Cluster0	0	支持	支持	支持(*)	支持
	Cluster1	1	支持	支持	支持(*)	支持
	Cluster2	2	支持	支持	支持(*)	支持
	Cluster3	3	支持	支持	支持(*)	支持
	Esmart0	4	支持	不支持	支持	支持
	Esmart1	5	支持	不支持	支持	支持
RK3576	Cluster0	0	支持	支持(**)	支持	支持
	Cluster1	1	支持	支持(**)	支持	支持
	Esmart0	4	支持	不支持	支持	支持
	Esmart1	5	支持	不支持	支持	支持
	Esmart2	6	支持	不支持	支持	支持
	Esmart3	7	支持	不支持	支持	支持

【备注】

- (*)只支持压缩数据，不支持非压缩数据。
- (**)RGB支持AFBC和RKFBC压缩数据，YUV只支持RKFBC压缩数据。
- 图层ID可以参考[VO_LAYER_NAME_RK356X_E](#)、[VO_LAYER_NAME_E](#)和[VO_VIR_LAYER_NAME_E](#)。

3. 模块功能

视频输出（VO）模块用于视频输出管理，支持多VOP以及多图层显示。实现启用视频输出设备或通道、发送视频数据或者UI数据到输出通道等功能。

4. VO资源数目表

模块名称	数量	说明
VoChn	VO_MAX_CHN_NUM	视频输出通道号。
VoDev	VO_MAX_DEV_NUM	显示输出设备号，取值范围[0, VO_MAX_DEV_NUM - 1]。
VoLayer	VO_MAX_LAYER_NUM	RK356x的VOP有2个cluster、2个emart、2个smart图层。
VoWbc	VO_MAX_WBC_NUM	回写设备，芯片只支持一个回写通路，所以 VoWbc 只有一个值：0。

5. API参考

5.1 设备操作

5.1.1 RK_MPI_VO_SetPubAttr

【描述】

设置视频输出设备的公共属性。

【语法】

```
RK_S32 RK_MPI_VO_SetPubAttr(VO_DEV VoDev, const VO_PUB_ATTR_S *pstPubAttr)
```

【参数】

参数名称	描述	输入/输出
VoDev	显示输出设备号。	输入
pstPubAttr	视频输出设备公共属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_DEVID	VoDev不正确。
RK_ERR_VO_NULL_PTR	pstPubAttr为空指针。
RK_ERR_VO_NOT_SUPPORT	输入enIntfSync不在支持列表内。
RK_ERR_VO_BUSY	设置分辨率失败。

【举例】

请参见[RK_MPI_VO_Enable](#)的举例。

【相关主题】

[RK_MPI_VO_GetPubAttr](#)

5.1.2 RK_MPI_VO_GetPubAttr

【描述】

获取视频输出设备的公共属性。

【语法】

```
RK_S32 RK_MPI_VO_GetPubAttr(VO_DEV VoDev, VO_PUB_ATTR_S *pstPubAttr)
```

【参数】

参数名称	描述	输入/输出
VoDev	显示输出设备号。	输入
pstPubAttr	视频输出设备公共属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_DEVID	VoDev不正确。
RK_ERR_VO_NULL_PTR	pstPubAttr为空指针。
RK_ERR_VO_SYS_NOTREADY	VO相关服务创建失败。

【举例】

请参见[RK_MPI_VO_Enable](#)的举例。

【相关主题】

[RK_MPI_VO_SetPubAttr](#)

5.1.3 RK_MPI_VO_Enable

【描述】

启用视频输出设备。

【语法】

```
RK_S32 RK_MPI_VO_Enable(VO_DEV VoDev)
```

【参数】

参数名称	描述	输入/输出
VoDev	显示输出设备号。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_DEVID	VoDev不正确。
RK_ERR_VO_SYS_NOTREADY	VO相关服务创建失败。
RK_ERR_VO_NOT_SUPPORT	显示输出接口类型不支持。
RK_ERR_VO_BUSY	创建、使能显示输出设备失败。

【举例】

```
VO_PUB_ATTR_S VoPubAttr;
VO_VIDEO_LAYER_ATTR_S stLayerAttr;
RK_U32 s32Ret;

VO_LAYER VoLayer = RK356X_VOP_LAYER_CLUSTER_0;
VO_DEV VoDev = RK356X_VO_DEV_HD0;

memset(&VoPubAttr, 0, sizeof(VO_PUB_ATTR_S));
s32Ret = RK_MPI_VO_GetPubAttr(VoDev, &VoPubAttr);
```

```

if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}

VoPubAttr.enIntfType = VO_INTF_HDMI;
VoPubAttr.enIntfSync = VO_OUTPUT_1080P60;

s32Ret = RK_MPI_VO_SetPubAttr(VoDev, &VoPubAttr);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}
s32Ret = RK_MPI_VO_Enable(VoDev);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}

s32Ret = RK_MPI_VO_Disable(VoDev);
if (s32Ret != RK_SUCCESS)
    return s32Ret;

s32Ret = RK_MPI_VO_CloseFd();
if (s32Ret != RK_SUCCESS)
    return s32Ret;

```

【相关主题】

[RK_MPI_VO_Disable](#)

5.1.4 RK_MPI_VO_Disable

【描述】

禁用视频输出设备。

【语法】

```
RK_S32 RK_MPI_VO_Disable(VO_DEV VoDev)
```

【参数】

参数名称	描述	输入/输出
VoDev	显示输出设备号。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_DEVID	VoDev不正确。

【举例】

```

RK_S32 s32Ret = RK_SUCCESS;
VO_DEV VoDev = RK356X_VO_DEV_HD0;

s32Ret = RK_MPI_VO_Disable(VoDev);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;

return s32Ret;

```

5.1.5 RK_MPI_VO_GetPostProcessParam

【描述】

获取设备输出图像效果。

【语法】

```
RK_S32 RK_MPI_VO_GetPostProcessParam(VO_DEV VoDev, VO_CSC_S *pstParam)
```

【参数】

参数名称	描述	输入/输出
VoDev	显示输出设备号。	输入
pstParam	图像输出效果结构体指针。	输出

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_DEVID	VoDev不正确。
RK_ERR_VO_NULL_PTR	pstParam为空指针。
RK_ERR_VO_SYS_NOTREADY	VO相关服务创建失败。

5.1.6 RK_MPI_VO_SetPostProcessParam

【描述】

设置设备输出图像效果。

【语法】

```
RK_S32 RK_MPI_VO_SetPostProcessParam(VO_DEV VoDev, VO_CSC_S *pstParam)
```

【参数】

参数名称	描述	输入/输出
VoDev	显示输出设备号。	输入
pstParam	图像输出效果结构体指针。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_DEVID	VoDev不正确。
RK_ERR_VO_NULL_PTR	pstParam为空指针。
RK_ERR_VO_SYS_NOTREADY	VO相关服务创建失败。
RK_ERR_VO_BUSY	设置失败。

5.1.7 RK_S32 RK_MPI_VO_Get_Edid

【描述】

获取显示设备EDID数据

【语法】

```
RK_S32 RK_MPI_VO_Get_Edid(RK_U32 enIntfType, RK_U32 u32Id, VO_EDID_S *pstEdidData)
```

【参数】

参数名称	描述	输入/输出
enIntfType	显示输出接口类型	输入
u32Id	同类型显示输出接口设备号，比如HDMI0、HDMI1	输入
pstEdidData	EDID数据	输出

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_ILLEGAL_PARAM	enIntfType不正确。
RK_ERR_VO_NULL_PTR	pstEdidData为空指针。
RK_ERR_VO_DEV_NOT_ENABLE	VoDev对应的显示设备未使能。

【备注】

u32Id为芯片相同接口类型的ID号，比如有两个芯片内置HDMI，u32Id可以取值0和1。

5.1.8 RK_S32 RK_MPI_VO_GetSinkCapability

【描述】

获取显示设备连接状态和相关能力。

【语法】

```
RK_S32 RK_MPI_VO_GetSinkCapability(RK_U32 enIntfType, RK_U32 u32Id, VO_SINK_CAPABILITY_S *pstSinkCap)
```

【参数】

参数名称	描述	输入/输出
enIntfType	显示输出接口类型	输入
u32Id	同类型显示输出接口设备号，比如HDMI0、HDMI1	输入
pstSinkCap	显示设备状态。	输出

[返回值]

返回值	描述
0	成功。
RK_ERR_VO_ILLEGAL_PARAM	enIntfType不正确。
RK_ERR_VO_NULL_PTR	pstSinkCap为空指针。
RK_ERR_VO_DEV_NOT_ENABLE	VoDev对应的显示设备未使能。

【备注】

- u32Id为芯片相同接口类型的ID号，比如有两个芯片内置HDMI，u32Id可以取值0和1。

5.1.9 RK_MPI_VO_RegCallbackFunc

【描述】

注册显示设备热插拔回调函数

【语法】

```
RK_S32 RK_MPI_VO_RegCallbackFunc(RK_U32 enIntfType, RK_U32 u32Id, RK_VO_CALLBACK_FUNC_S *pstCallbackFunc)
```

【参数】

参数名称	描述	输入/输出
enIntfType	显示输出接口类型	输入
u32Id	同类型显示输出接口设备号，比如HDMI0、HDMI1	输入
pstCallbackFunc	热插拔回调函数	输入

[返回值]

返回值	描述
0	成功。
RK_ERR_VO_ILLEGAL_PARAM	enIntfType不正确。
RK_ERR_VO_NULL_PTR	pstCallbackFunc 为空指针。
RK_ERR_VO_DEV_NOT_ENABLE	VoDev对应的显示设备未使能。

【备注】

- 仅支持有热插拔能力的显示接口，比如HDMI和EDP。
- 需要显示设备处于使能状态。
- u32Id为芯片相同接口类型的ID号，比如有两个芯片内置HDMI，u32Id可以取值0和1。

5.1.10 RK_MPI_VO_UnRegCallbackFunc

【描述】

注册显示设备热插拔回调函数

【语法】

```
RK_S32 RK_MPI_VO_UnRegCallbackFunc(RK_U32 enIntfType, RK_U32 u32Id, RK_VO_CALLBACK_FUNC_S *pstCallbackFunc)
```

【参数】

参数名称	描述	输入/输出
enIntfType	显示输出接口类型	输入
u32Id	同类型显示输出接口设备号，比如HDMI0、HDMI1	输入
pstCallbackFunc	热插拔回调函数	输入

[返回值]

返回值	描述
0	成功。
RK_ERR_VO_ILLEGAL_PARAM	enIntfType不正确。
RK_ERR_VO_NULL_PTR	pstCallbackFunc 为空指针。
RK_ERR_VO_DEV_NOT_ENABLE	VoDev对应的显示设备未使能。
RK_ERR_VO_ILLEGAL_PARAM	pstCallbackFunc 不是已注册函数。

【备注】

- 仅支持有热插拔能力的显示接口，比如HDMI和EDP。
- 需要显示设备处于使能状态。
- u32Id为芯片相同接口类型的ID号，比如有两个芯片内置HDMI，u32Id可以取值0和1。

5.1.11 RK_MPI_VO_RegVsyncCallbackFunc

【描述】

注册显示设备帧中断回调函数

【语法】

```
RK_S32 RK_MPI_VO_RegVsyncCallbackFunc(VO_DEV VoDev, RK_VO_VSYNC_CALLBACK_FUNC_S *pstCallbackFunc)
```

【参数】

参数名称	描述	输入/输出
VoDev	显示输出接口设备号	输入
pstCallbackFunc	帧中断回调函数	输入

[返回值]

返回值	描述
0	成功。
RK_ERR_VO_INVALID_DEVID	VoDev不正确。
RK_ERR_VO_NULL_PTR	pstCallbackFunc 为空指针。
RK_ERR_VO_DEV_NOT_ENABLE	VoDev对应的显示设备未使能。
RK_ERR_VO_SYS_NOTREADY	VO相关服务创建失败
RK_ERR_VO_ILLEGAL_PARAM	回调函数无效

【备注】

- 需要显示设备处于使能状态。

5.1.12 RK_MPI_VO_UnRegVsyncCallbackFunc

【描述】

注销显示设备帧中断回调函数

【语法】

```
RK_S32 RK_MPI_VO_UnRegCallbackFunc(VO_DEV VoDev, RK_VO_VSYNC_CALLBACK_FUNC_S *pstCallbackFunc)
```

【参数】

参数名称	描述	输入/输出
VoDev	显示输出接口设备号	输入
pstCallbackFunc	帧中断回调函数	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_DEVID	VoDev不正确。
RK_ERR_VO_NULL_PTR	pstCallbackFunc 为空指针。
RK_ERR_VO_DEV_NOT_ENABLE	VoDev对应的显示设备未使能。
RK_ERR_VO_SYS_NOTREADY	VO相关服务创建失败
RK_ERR_VO_ILLEGAL_PARAM	回调函数无效

【备注】

- 需要显示设备处于使能状态。

5.1.13 RK_MPI_VO_GetHdmiParam

【描述】

获取显示接口的HDMI属性参数

【语法】

```
RK_S32 RK_MPI_VO_GetHdmiParam(RK_U32 enIntfType, RK_U32 u32Id, VO_HDMI_PARAM_S *pstHDMIParam)
```

【参数】

参数名称	描述	输入/输出
enIntfType	显示输出接口类型	输入
u32Id	同类型显示输出接口设备号，比如HDMI0、HDMI1	输入
pstHDMIParam	HDMI属性结构体指针	输出

【返回值】

返回值	描述
0	成功
RK_ERR_VO_ILLEGAL_PARAM	enIntfType不正确
RK_ERR_VO_NULL_PTR	pstHDMIParam为空指针
RK_ERR_VO_SYS_NOTREADY	系统为准备好

【备注】

- u32Id为芯片相同接口类型的ID号，比如有两个芯片内置HDMI，u32Id可以取值0和1。

5.1.14 RK_MPI_VO_SetHdmiParam

【描述】

设置显示接口的HDMI属性参数

【语法】

```
RK_S32 RK_MPI_VO_SetHdmiParam(RK_U32 enIntfType, RK_U32 u32Id, const VO_HDMI_PARAM_S *pstHDMIParam)
```

【参数】

参数名称	描述	输入/输出
enIntfType	显示输出接口类型	输入
u32Id	同类型显示输出接口设备号，比如HDMI0、HDMI1	输入
pstHDMIParam	HDMI属性结构体指针	输入

【返回值】

返回值	描述
0	成功
RK_ERR_VO_ILLEGAL_PARAM	enIntfType不正确
RK_ERR_VO_NULL_PTR	pstHDMIParam为空指针
RK_ERR_VO_SYS_NOTREADY	系统为准备好

【备注】

- u32Id为芯片相同接口类型的ID号，比如有两个芯片内置HDMI，u32Id可以取值0和1。

【举例】

```
VO_HDMI_PARAM_S stHDMIParam;
RK_MPI_VO_GetHdmiParam(VO_INTF_HDMI, 0, &stHDMIParam);
stHDMIParam.enColorFmt = VO_HDMI_COLOR_FORMAT_RGB;
stHDMIParam.enQuantRange = VO_HDMI_QUANT_RANGE_FULL;
stHDMIParam.enHdmiMode = VO_HDMI_MODE_HDMI;
RK_MPI_VO_SetHdmiParam(VO_INTF_HDMI, 0, &stHDMIParam);
```

5.1.15 RK_MPI_VO_SyncDevs

【描述】

同步不同的VoDev帧中断。

【语法】

```
RK_MPI_VO_SyncDevs(RK_U32 u32Vodevs)
```

【参数】

参数名称	描述	输入/输出
u32Vodevs	每个比特表示需要同步的显示输出设备ID。	输入

【返回值】

返回值	描述
0	成功
RK_ERR_VO_INVALID_DEVID	VoDev不正确。
RK_ERR_VO_BUSY	创建、使能显示输出设备失败。

【备注】

- 必须在所有需要同步的VoDev使能RK_MPI_Enable之后调用。

【举例】

```
....  
RK_MPI_Enable(0)  
....  
RK_MPI_Enable(2);  
....  
// 同步VoDev0 和 VoDev2  
u32TimeOut = 10;  
u32Devs = BIT(0) | BIT(2);  
while (timeout--) {  
    u32Ret = RK_MPI_VO_SyncDevs(u32Devs);  
    if (u32Ret == RK_ERR_VO_NOT_SUPPORT)  
        return;  
    else if (u32Ret == 0)  
        break;  
    else  
        usleep(100000);  
}
```

5.1.16 RK_MPI_VO_CloseFd

【描述】

关闭视频输出模块所有占用的Fd。

【语法】

```
RK_S32 RK_MPI_VO_CloseFd(RK_VOID)
```

【参数】

无

【返回值】

返回值	描述
0	成功。

5.2 Framebuffer设置

5.2.1 RK_MPI_VO_CreateGraphicsFrameBuffer

【描述】

创建图形层Framebuffer。

【语法】

```
RK_S32 RK_MPI_VO_CreateGraphicsFrameBuffer(int Width, int Height, RK_U32 Formant, RK_VOID **fd)
```

【参数】

参数名称	描述	输入/输出
Width	图形宽度。	输入。
Height	图形高度。	输入。
Formant	像素格式类型。	输入。
fd	Void *类型指针。	输入。

【返回值】

返回值	描述
0	申请buffer失败。
非0	申请的buffer长度。

【举例】

```
RK_U32 u32ImageWidth;
RK_U32 u32ImageHeight;
RK_VOID *pMblk = RK_NULL;
VO_FRAME_INFO_S stFrameInfo;
RK_U32 u32BuffSize;

u32ImageWidth = 1920;
u32ImageHeight = 1080;
u32BuffSize = RK_MPI_VO_CreateGraphicsFrameBuffer(u32ImageWidth, u32ImageHeight,
    RK_FMT_YUV420SP, &pMblk);
if (u32BuffSize == 0) {
    return RK_FAILURE;
}
/*do something to fill Graphic FrameBuffer */
RK_MPI_VO_GetFrameInfo(*pMblk, &stFrameInfo);
...
RK_MPI_VO_DestroyGraphicsFrameBuffer(pMblk);
```

【相关主题】

[RK_MPI_VO_DestroyGraphicsFrameBuffer](#)

5.2.2 RK_MPI_VO_DestroyGraphicsFrameBuffer

【描述】

销毁图形层Framebuffer。

【语法】

```
RK_S32 RK_MPI_VO_DestroyGraphicsFrameBuffer(RK_VOID* fd)
```

【参数】

参数名称	描述	输入/输出
fd	void类型指针。	输入。

【返回值】

返回值	描述
0	成功。
非0	失败。

【相关主题】

[RK_MPI_VO_CreateGraphicsFrameBuffer](#)

5.2.3 RK_MPI_VO_GetGraphicsFrameBuffer

【描述】

获取创建的Framebuffer虚拟地址。

【语法】

```
RK_VOID* RK_MPI_VO_GetGraphicsFrameBuffer(RK_VOID* fd)
```

【参数】

参数名称	描述	输入/输出。
fd	void类型指针。	输入。

【返回值】

返回值	描述
NULL	获取地址失败。
非零	获取的Frame Buffer虚拟地址。

5.2.4 RK_MPI_VO_SetGFXMode

【描述】

设置图形buffer的创建模式。

【语法】

```
RK_S32 RK_MPI_VO_SetGFXMode(VO_GFX_MODE_E u32Mode)
```

【参数】

参数名称	描述	输入/输出
u32Mode	图形buffer创建模式。	输入。

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_ILLEGAL_PARAM	Mode参数错误。
【备注】	

- 该函数需要在与VO相关的任何操作函数之前调用。

【举例】

请参见[RK_MPI_VO_GetGfxFrameBuffer](#)的举例。

【相关主题】

[RK_MPI_VO_GetGfxFrameBuffer](#)

5.2.5 RK_MPI_VO_GetGfxFrameBuffer

【描述】

获取指定图形通道预创建的buffer。该API创建的是单个图形buffer，适合于OSD，画字，画边框等不需要多个UI buffer的画图场景。

【语法】

```
RK_S32 RK_MPI_VO_GetGfxFrameBuffer(VO_LAYER VoLayer, VO_CHN VoChn, VO_FRAME_INFO_S *pstFrame)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	图形通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstFrame	VO_FRAME_INFO_S指针。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	输入VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	输入VoChn不正确。
RK_ERR_VO_NULL_PTR	pstFrame为空指针。
RK_ERR_VO_NOT_PERMIT	VoLayer没有设置预创建模式。
RK_ERR_VO_NOT_SUPPORT	不支持pstFrame的enPixelFormat。
RK_ERR_VO_BUSY	VoChn对应的图层通道已经申请过buffer。

【备注】

- 预创建图形buffer需要在对VO进行相关操作之前，适合于Video层和UI层融合场景。
- 图形通道数据格式为BGRA5551/RGBA5551时，需要填写u32FgAlpha和u32BgAlpha。
- 在视频层上使用预创建图形buffer的方式，不用再使能UI图形层，也即是不需要使用UI layer层。
- 通过RK_MPI_VO_GetGfxFrameBuffer设置ui单buffer的方式相对于使用Video层+UI层的方式，能很好降低VOP的带宽。

【举例】

```
VO_PUB_ATTR_S VoPubAttr;
VO_VIDEO_LAYER_ATTR_S stLayerAttr;
RK_U32 s32Ret;
RK_U32 u32DispWidth = 1920;
RK_U32 u32DispHeight = 1080;
RK_U32 u32ImageWidth = 1920;
RK_U32 u32ImageHeight = 1080;

memset(&VoPubAttr, 0, sizeof(VO_PUB_ATTR_S));
memset(&stLayerAttr, 0, sizeof(VO_VIDEO_LAYER_ATTR_S));

stLayerAttr.enPixFormat      = RK_FMT_YUV420SP;
stLayerAttr.stDispRect.s32X   = 0;
stLayerAttr.stDispRect.s32Y   = 0;
stLayerAttr.stDispRect.u32Width = u32DispWidth;
stLayerAttr.stDispRect.u32Height = u32DispHeight;
stLayerAttr.stImageSize.u32Width = u32ImageWidth;
stLayerAttr.stImageSize.u32Height = u32ImageHeight;

VO_LAYER VoLayer = RK356X_VOP_LAYER_CLUSTER0;
VO_DEV VoDev = RK356X_VO_DEV_HD0;
VoPubAttr.enIntfType = VO_INTF_HDMI;
VoPubAttr.enIntfSync = VO_OUTPUT_1080P60;

VO_FRAME_INFO_S stVFrame;
RK_MPI_VO_SetGfxMode(VO_MODE_GFX_PRE_CREATED);
/* Get 1st GFX buffer */
stVFrame.enPixelFormat = RK_FMT_BGRA5551;
stVFrame.u32FgAlpha = 128;
stVFrame.u32BgAlpha = 0;
stVFrame.u32Width = 1920;
stVFrame.u32Height = 1080;
RK_MPI_VO_GetGfxFrameBuffer(VoLayer, 127, &stVFrame);
/* Draw 1st GFX */
memset(stVFrame.pData, 0xff, stVFrame.u32Size);

/* Get 2nd GFX buffer */
stVFrame.u32FgAlpha = 128;
stVFrame.u32BgAlpha = 0;
RK_MPI_VO_GetGfxFrameBuffer(VoLayer, 126, &stVFrame);

/* Draw 2nd GFX */
memset(stVFrame.pData, 0, stVFrame.u32Size);

s32Ret = RK_MPI_VO_SetPubAttr(VoDev, &VoPubAttr);
if (s32Ret != RK_SUCCESS) {
    return RK_FAILURE;
}
s32Ret = RK_MPI_VO_Enable(VoDev);
if (s32Ret != RK_SUCCESS) {
    return RK_FAILURE;
}
s32Ret = RK_MPI_VO_SetLayerAttr(VoLayer, &stLayerAttr);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;
s32Ret = RK_MPI_VO_EnableLayer(VoLayer);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;
s32Ret = RK_MPI_VO_DisableLayer(VoLayer);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;
```

【相关主题】

[RK_MPI_VO_ReleaseGfxFrameBuffer](#)

5.2.6 RK_MPI_VO_ReleaseGfxFrameBuffer

【描述】

释放指定图形通道预创建的buffer。

【语法】

```
RK_S32 RK_MPI_VO_ReleaseGfxFrameBuffer(VO_LAYER VoLayer, VO_CHN VoChn)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	图形通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	输入VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	输入VoDev不正确。
RK_ERR_VO_NULL_PTR	pstFrame为空指针。
RK_ERR_VO_NOT_PERMIT	VoLayer没有设置预创建模式。

5.3 图层操作

每个VoDev支持的图层分为视频层、图形成、鼠标层、虚拟图层四种，视频层和图形层支持多通道操作，每个图层可以灵活绑定到指定显示输出设备上。

RK3568默认绑定关系如下表。

显示输出设备	视频层	图形层	鼠标层	虚拟层
Dev0	Cluster0	Esmart0	Smart0	无
Dev1	Custer1	Esmart1	Smart1	无
Dev2	无	无	无	无

RK3588默认绑定关系如下表。

显示输出设备	视频层	图形层	鼠标层	虚拟层
Dev0	Cluster0	Esmart0	Esmart3	无
Dev1	Custer1	Esmart1	无	无
Dev2	Custer2	Esmart2	无	无
Dev3	Custer3	无	无	无

5.3.1 RK_MPI_VO_ClearLayersBinding

【描述】

解除所有图层的绑定关系。

【语法】

```
RK_S32 RK_MPI_VO_ClearLayersBinding(RK_VOID)
```

【参数】

无

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_NOT_SUPPORT	设备不支持

【备注】

调用时，所有图层和显示设备需要处于关闭状态。

5.3.2 RK_MPI_VO_BindLayer

【描述】

设置图层和显示设备绑定关系，绑定图层到指定VOP设备。

【语法】

```
RK_S32 RK_MPI_VO_BindLayer(VO_LAYER VoLayer, VO_DEV VoDev, VO_LAYER_MODE_E Mode)
```

【参数】

参数名称	描述	输入/输出
VoLayer	需要绑定的图层号。	输入
VoDev	需要绑定的VOP设备号。	输入
Mode	图层类型。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_DEVID	输入VoDev不正确。
RK_ERR_VO_INVALID_LAYERID	输入VoLayer不正确。
RK_ERR_VO_SYS_NOTREADY	VO相关服务创建失败。
RK_ERR_VO_DEV_HAS_BINDED	VoLayer已经绑定过其他显示输出设备。
RK_ERR_VO_ILLEGAL_PARAM	Mode参数错误。
【备注】	

- 图层若已绑定过显示输出设备，再次绑定前需先解除图层和原显示输出设备的绑定关系。
- 一个VoDe只支持一个鼠标层 + 一个视频层 + 一个图形层。
- 绑定的显示设备要处于关闭状态。
- 虚拟图层绑定的图层类型必须为VO_LAYER_MODE_VIRTUAL。

- 多个虚拟图层可以绑定到同一个VoDev。

【举例】

请参见[RK_MPI_VO_EnableLayer](#)

【相关主题】

[RK_MPI_VO_UnBindLayer](#)

5.3.3 RK_MPI_VO_UnBindLayer

【描述】

解除图层和显示输出设备的绑定关系。

【语法】

```
RK_S32 RK_MPI_VO_UnBindLayer(VO_LAYER VoLayer, VO_DEV VoDev)
```

【参数】

参数名称	描述	输入/输出
VoLayer	需要解除绑定的图层号。	输入
VoDev	需要解除绑定的VOP设备号。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_DEVID	输入VoDev不正确。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_SYS_NOTREADY	VO相关服务创建失败。

【备注】

- 绑定的显示输出设备（VoDev）要处于关闭状态。

【举例】

请参见[RK_MPI_VO_EnableLayer](#)

【相关主题】

[RK_MPI_VO_BindLayer](#)

5.3.4 RK_MPI_VO_SetLayerAttr

【描述】

设置图层属性。

【语法】

```
RK_S32 RK_MPI_VO_SetLayerAttr(VO_LAYER VoLayer, const VO_VIDEO_LAYER_ATTR_S *pstLayerVideoAttr)
```

【参数】

参数名称	描述	输入/输出
VoLayer	图层号。	输入
pstLayerVideoAttr	视频层属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_NULL_PTR	pstLayerVideoAttr为空指针。
RK_ERR_VO_ILLEGAL_PARAM	不支持pstLayerVideoAttr说明的颜色格式。

【备注】

- 图层属性的显示帧率不能超过图层绑定显示输出设备的刷新率。
- 出于性能考虑，建议设置图层属性的显示帧率不超过30帧。
- 视频层为Cluster图层时，建议配置图层属性enCompressMode为COMPRESS_AFBC_16x16，可以节省系统带宽。
- VO_VIDEO_LAYER_ATTR_S的bBypassFrame属性在GPU/RGA拼接模式，且只有一个视频通道使能时有效。

【举例】

请参见[RK_MPI_VO_EnableLayer](#)的举例。

5.3.5 RK_MPI_VO_GetLayerAttr

【描述】

获取图层属性。

【语法】

```
RK_S32 RK_MPI_VO_GetLayerAttr(VO_LAYER VoLayer, VO_VIDEO_LAYER_ATTR_S *pstLayerAttr)
```

【参数】

参数名称	描述	输入/输出
VoLaye	图层号。	输入
pstLayerAttr	视频层属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_NULL_PTR	pstLayerAttr为空指针。

5.3.6 RK_MPI_VO_SetLayerSpliceMode

【描述】

设置图层合成方式。

【语法】

```
RK_S32 RK_MPI_VO_SetLayerSpliceMode(VO_LAYER VoLayer, VO SPLICE_MODE_E enSpliceMode);
```

【参数】

参数名称	描述	输入/输出
VoLayer	图层号。	输入。
enSpliceMode	图层合成方式。	输入。

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_ILLEGAL_PARAM	enSpliceMode参数不正确。

【备注】

- 默认图层使用GPU合成方式。
- 需要在RK_MPI_VO_Enable图层使能前设置图层合成方式。
- 在RGA合成模式下，图层需使用支持非afbc格式的图层。（不能使用Cluster图层）。

5.3.7 RK_MPI_VO_GetLayerSpliceMode

【描述】

获取图层合成方式。

【语法】

```
RK_S32 RK_MPI_VO_GetLayerSpliceMode(VO_LAYER VoLayer, VO SPLICE_MODE_E *enSpliceMode)
```

【参数】

参数名称	描述	输入/输出
VoLayer	图层号。	输入。
enSpliceMode	图层合成方式。	输出。

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_NULL_PTR	enSpliceMode为空指针。

5.3.8 RK_MPI_VO_EnableLayer

【描述】

使能图层。

【语法】

```
RK_S32 RK_MPI_VO_EnableLayer(VO_LAYER VoLayer)
```

【参数】

参数名称	描述	输入/输出
VoLayer	图层号。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_BUSY	VO相关服务创建失败。
RK_ERR_VO_LAYER_NOT_BINDED	VoLayer未绑定VoDev。
RK_ERR_VO_DEV_NOT_ENABLE	VoLayer绑定的VoDev未使能。
RK_ERR_VO_NO_MEM	系统内存不足。
RK_ERR_VO_SYS_NOTREADY	图层相关服务创建失败。

【备注】

- 图层使能前需要先调用RK_MPI_VO_BindLayer。
- 图层绑定的显示输出设备要先调用RK_MPI_VO_Enable使能。

【举例】

```

VO_PUB_ATTR_S VoPubAttr;
VO_VIDEO_LAYER_ATTR_S stLayerAttr;
RK_U32 s32Ret;
RK_U32 VoDev, VoLayer;

memset(&VoPubAttr, 0, sizeof(VO_PUB_ATTR_S));
memset(&stLayerAttr, 0, sizeof(VO_VIDEO_LAYER_ATTR_S));

VoPubAttr.enIntfType = VO_INTF_HDMI;
VoPubAttr.enIntfSync = VO_OUTPUT_1080P50;
VoDev = RK356X_VO_DEV_HD0;
VoLayer = RK356X_VOP_LAYER_CLUSTER0;

RK_MPI_VO_BindLayer(VoLayer, VoDev, VO_LAYER_MODE_VIDEO);
s32Ret = RK_MPI_VO_SetPubAttr(VoDev, &VoPubAttr);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}
s32Ret = RK_MPI_VO_Enable(VoDev);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}
stLayerAttr.stDispRect.s32X = 0;
stLayerAttr.stDispRect.s32Y = 0;
stLayerAttr.stDispRect.u32Width = 1920;
stLayerAttr.stDispRect.u32Height = 1080;
stLayerAttr.stImageSize.u32Width = 1920;
stLayerAttr.stImageSize.u32Height = 1080;
stLayerAttr.u32DispFrmRt = 25;
stLayerAttr.enPixelFormat = RK_FMT_YUV420SP;
stLayerAttr.enCompressMode = COMPRESS_AFBC_16x16;
s32Ret = RK_MPI_VO_SetLayerAttr(VoLayer, &stLayerAttr);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}
s32Ret = RK_MPI_VO_EnableLayer(VoLayer);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}
s32Ret = RK_MPI_VO_DisableLayer(VoLayer);
if (s32Ret != RK_SUCCESS)
    return s32Ret;

```

5.3.9 RK_MPI_VO_DisableLayer

【描述】

禁止图层。

【语法】

```
RK_S32 RK_MPI_VO_DisableLayer(VO_LAYER VoLayer)
```

【参数】

参数名称	描述	输入/输出
VoLayer	图层号。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_WBC_NOT_DISABLE	图层的回写功能未关闭。

5.3.10 RK_MPI_VO_SetLayerDispBufLen

【描述】

设置图层上的显示缓存长度。

【语法】

```
RK_S32 RK_MPI_VO_SetLayerDispBufLen(VO_LAYER VoLayer, RK_U32 u32BufLen)
```

【参数】

参数名称	描述	输入/输出
VoLayer	图层号。	输入
u32BufLen	显示缓冲的长度。【3, 15】。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_DEV_HAS_ENABLED	图层已经使能。
RK_ERR_VO_ILLEGAL_PARAM	u32BufLen 值不正确。

5.3.11 RK_MPI_VO_GetLayerDispBufLen

【描述】

获取图层上的显示缓存长度。

【语法】

```
RK_S32 RK_MPI_VO_GetLayerDispBufLen(VO_LAYER VoLayer, RK_U32 *pu32BufLen)
```

【参数】

参数名称	描述	输入/输出
VoDev	图层号。	输入
pu32BufLen	u32类型指针。	输出

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_NULL_PTR	pu32BufLen为空指针。

5.3.12 RK_MPI_VO_GetLayerFrame

【描述】

获取图层上的输出图像帧。

【语法】

```
RK_S32 RK_MPI_VO_GetLayerFrame(VO_LAYER VoLayer, VIDEO_FRAME_INFO_S *pstVFrame, RK_S32 s32MilliSec)
```

【参数】

参数名称	描述	输入/输出
VoLayer	图层号	输入
pstVFrame	获取的输出屏幕图像数据信息结构体指针。	输出
s32MilliSec	超时参数，目前默认设置为0。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_NULL_PTR	pstVFrame为空指针。
RK_ERR_VO_LAYER_NOT_ENABLE	图层未使能。

【备注】

只支持获取视频层和虚拟层数据。

5.3.13 RK_MPI_VO_ReleaseLayerFrame

【描述】

释放图层上的输出图像帧。

【语法】

```
RK_S32 RK_MPI_VO_ReleaseLayerFrame(VO_LAYER VoLayer, VIDEO_FRAME_INFO_S *pstVFrame)
```

【参数】

参数名称	描述	输入/输出
VoLayer	图层号。	输入
pstVFrame	释放的输出屏幕图像数据信息结构体指针。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_NULL_PTR	pstVFrame为空指针。

5.3.14 RK_MPI_VO_SendLayerFrame

【描述】

将图像送入指定图形层输出通道显示。

【语法】

```
RK_S32 RK_MPI_VO_SendLayerFrame(VO_LAYER VoLayer, VIDEO_FRAME_INFO_S *pstVFrame)
```

【参数】

参数名称	描述	输入/输出
VoLayer	图形层号。	输入
pstVFrame	发送的输出屏幕图像数据信息结构体指针。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_NULL_PTR	pstVFrame为空指针。
RK_ERR_VO_LAYER_NOT_ENABLE	图层未使能。
RK_ERR_VO_ILLEGAL_PARAM	pstVFrame不符合要求。

【备注】

- 该函数直接将图像数据送给图层显示，图像数据格式由图层性能决定。

5.3.15 RK_MPI_VO_SetCursorPostion

【描述】

更新鼠标层的坐标。

【语法】

```
RK_S32 RK_MPI_VO_SetCursorPostion(VO_LAYER VoLayer, const RK_U32 x, const RK_U32 y)
```

【参数】

参数名称	描述	输入/输出
VoLayer	鼠标图层号。	输入
x	x轴坐标	输入
y	y轴坐标	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确，或者图层不是鼠标层。
RK_ERR_VO_NULL_PTR	图层数据为空。
RK_ERR_VO_LAYER_NOT_ENABLE	图层未使能。

【备注】

- 首先需要调用RK_MPI_VO_SendLayerFrame将第一帧送显，第二帧开始再用该函数更新坐标

5.3.16 RK_MPI_VO_SetCursorLogicalRect

【描述】

设置鼠标移动区域。

【语法】

```
RK_S32 RK_MPI_VO_SetCursorLogicalRect(VO_LAYER VoLayer, const RK_U32 maxWidth, const RK_U32 maxHeight)
```

【参数】

参数名称	描述	输入/输出
VoLayer	鼠标图层号。	输入
maxWidth	鼠标右边界	输入
maxHeight	鼠标下边界	输入

【备注】

- 当应用层鼠标移动范围与实际显示范围不一致时候使用，例如显示器4k分辨率，UI和鼠标都在1080P范围内时候调用该接口，当maxWidth/maxHeight设为0时，恢复1: 1比例。

【举例】

```

VO_PUB_ATTR_S      stVoPubAttr;
VO_VIDEO_LAYER_ATTR_S stLayerAttr;
VO_CHN_ATTR_S      stChnAttr;
RK_U32              x=100,y=100;
RK_U32              u32BuffSize;
RK_S32              s32Ret = RK_SUCCESS;
RK_S32              VoDev = RK356X_VO_DEV_HD1;
VO_LAYER            VoLayer = RK356X_VOP_LAYER_SMART_0;
VIDEO_FRAME_INFO_S  stVFrame;
void* pmlk;

RK_MPI_VO_BindLayer(VoLayer, VoDev, VO_LAYER_MODE_CURSOR);

/* Enable VO Device , set 4k resolution*/
stVoPubAttr.enIntfType = VO_INTF_HDMI;
stVoPubAttr.enIntfSync = VO_OUTPUT_3840x2160_30;

```

```

RK_MPI_VO_SetPubAttr(VoDev, &stVoPubAttr);

u32BuffSize = RK_MPI_VO_CreateGraphicsFrameBuffer(64, 64, RK_FMT_BGRA8888, &pMblk);

memset(&stLayerAttr, 0, sizeof(VO_VIDEO_LAYER_ATTR_S));
stLayerAttr.enPixFormat = RK_FMT_BGRA8888;
stLayerAttr.stDispRect.s32X = 0;
stLayerAttr.stDispRect.s32Y = 0;
stLayerAttr.stDispRect.u32Width = 64;
stLayerAttr.stDispRect.u32Height = 64;
stLayerAttr.stImageSize.u32Width = 64;
stLayerAttr.stImageSize.u32Height = 64;
s32Ret = RK_MPI_VO_SetLayerAttr(VoLayer, &stLayerAttr);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;

/*set cursor range*/
RK_MPI_VO_SetCursorLogicalRect(VoLyaer, 1920, 1080);

s32Ret = RK_MPI_VO_EnableLayer(VoLayer);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;

stVFrame.stVFrame.pMbBlk = pmlk;

// 配置鼠标buffer
RK_MPI_VO_SendLayerFrame(VoLayer, &stVFrame);
// 更新鼠标坐标
for (int i=0;i<10;j++) {
    RK_MPI_VO_SetCursorPostion(VoLayer, x, y);
    x+=10;
    y+=10;
}

```

5.4 通道操作

5.4.1 RK_MPI_VO_EnableChn

【描述】

启用指定的视频输出通道。

【语法】

```
RK_S32 RK_MPI_VO_EnableChn (VO_LAYER VoLayer, VO_CHN VoChn)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NO_MEM	系统内存不足。

【举例】

```
VO_VIDEO_LAYER_ATTR_S stLayerAttr;
VO_CHN_ATTR_S stChnAttr;
RK_U32 i;
RK_S32 s32Ret = RK_SUCCESS;
RK_U32 u32Layers = 2;
VO_LAYER VoLayer = 4;

memset(&stLayerAttr, 0, sizeof(VO_VIDEO_LAYER_ATTR_S));
stLayerAttr.enPixFormat = RK_FMT_BGRA8888;
stLayerAttr.stDispRect.s32X = 0;
stLayerAttr.stDispRect.s32Y = 0;
stLayerAttr.stImageSize.u32Width = 1920;
stLayerAttr.stImageSize.u32Height = 1080;
stLayerAttr.u32DispFrmRt = 25;
s32Ret = RK_MPI_VO_SetLayerAttr(VoLayer, &stLayerAttr);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;

s32Ret = RK_MPI_VO_EnableLayer(VoLayer);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;

s32Ret = RK_MPI_VO_GetLayerAttr(VoLayer, &stLayerAttr);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("[%s] Get Layer Attr failed\n", __func__);
    return RK_FAILURE;
}

for (i = 0; i < u32Layers; i++) {
    stChnAttr.stRect.s32X = 0;
    stChnAttr.stRect.s32Y = 0;
    stChnAttr.stRect.u32Width = stLayerAttr.stImageSize.u32Width;
    stChnAttr.stRect.u32Height = stLayerAttr.stImageSize.u32Height;
    stChnAttr.u32Priority = i;
    if (i == 0) {
        stChnAttr.u32FgAlpha = 128;
        stChnAttr.u32BgAlpha = 0;
    } else {
        stChnAttr.u32FgAlpha = 0;
        stChnAttr.u32BgAlpha = 128;
    }
}

s32Ret = RK_MPI_VO_SetChnAttr(VoLayer, i, &stChnAttr);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("[%s] set chn Attr failed\n", __func__);
    return RK_FAILURE;
}

s32Ret = RK_MPI_VO_EnableChn(VoLayer, i);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("[%s] Enable chn failed\n", __func__);
    return RK_FAILURE;
}
```

【相关主题】

[RK_MPI_DisableChn](#)

5.4.2 RK_MPI_VO_DisableChn

【描述】

禁用指定的视频输出通道。

【语法】

```
RK_S32 RK_MPI_DisableChn(VO_LAYER VoLayer, VO_CHN VoChn)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。

【举例】

```
RK_U32 i, u32Windows;
VO_LAYER VoLayer = 0;
RK_S32 s32Ret = RK_SUCCESS;
RK_S32 u32Windows = 4;

for (i = 0; i < u32Windows; i++) {
    RK_MPI_VO_ClearChnBuffer(VoLayer, i, RK_TRUE);
    s32Ret = RK_MPI_VO_DisableChn(VoLayer, i);
    if (s32Ret != RK_SUCCESS)
        return RK_FAILURE;
}
```

【相关主题】

[RK_MPI_VO_EnableChn](#)

5.4.3 RK_MPI_VO_SetChnAttr

【描述】

配置指定视频输出通道的属性。

【语法】

```
RK_S32 RK_MPI_VO_SetChnAttr(VO_LAYER VoLayer, VO_CHN VoChn, const VO_CHN_ATTR_S *pstChnAttr)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstAttr	视频通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pstChnAttr为空指针。
RK_ERR_VO_ILLEGAL_PARAM	pstChnAttr里的参数不正确。

【举例】

请参见[RK_MPI_VO_EnableChn](#)的举例。

【相关主题】

[RK_MPI_VO_SetChnAttr](#)

5.4.4 RK_MPI_VO_GetChnAttr

【描述】

获取指定视频输出通道的属性。

【语法】

```
RK_S32 RK_MPI_VO_GetChnAttr(VO_LAYER VoLayer, VO_CHN VoChn, VO_CHN_ATTR_S *pstAttr)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstAttr	视频通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pstAttr为空指针。

【举例】

请参见[RK_MPI_VO_EnableChn](#)的举例。

【相关主题】

[RK_MPI_VO_SetChnAttr](#)

5.4.5 RK_MPI_VO_SetChnParam

【描述】

配置指定视频输出通道的参数。

【语法】

```
RK_S32 RK_MPI_VO_SetChnParam(VO_LAYER VoLayer, VO_CHN VoChn, const VO_CHN_PARAM_S *pstChnParam)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstChnParam	视频通道参数指针。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pstChnParam为空指针。

【举例】

```

VO_CHN_PARAM_S     stChnParam;
VO_VIDEO_LAYER_ATTR_S stLayerAttr;

VO_LAYER  VoLayer = RK356X_VOP_LAYER_CLUSTER_0;
VO_CHN    VoChn = 0;
RK_S32   s32Ret = RK_SUCCESS;;

s32Ret = RK_MPI_VO_GetLayerAttr(VoLayer, &stLayerAttr);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;

s32Ret = RK_MPI_VO_GetChnParam(VoLayer, VoChn, &stChnParam);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;

stChnParam.stAspectRatio.enMode = ASPECT_RATIO_MANUAL;
u32Width = stLayerAttr.stDispRect.u32Width;
u32Height = stLayerAttr.stDispRect.u32Height;
stChnParam.stAspectRatio.stVideoRect.s32X = 0;
stChnParam.stAspectRatio.stVideoRect.s32Y = 0;
stChnParam.stAspectRatio.stVideoRect.u32Width = u32Width;
stChnParam.stAspectRatio.stVideoRect.u32Height = u32Height;
s32Ret = RK_MPI_VO_SetChnParam(VoLayer, VoChn, &stChnParam);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;

```

5.4.6 RK_MPI_VO_GetChnParam

【描述】

获取指定视频输出通道的参数。

【语法】

```
RK_S32 RK_MPI_VO_GetChnParam(VO_LAYER VoLayer, VO_CHN VoChn, VO_CHN_ATTR_S *pstChnParam)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstChnParam	视频通道参数指针。	输出

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pstChnParam为空指针。

【举例】

请参见[RK_MPI_VO_SetChnParam](#)的举例。

【相关主题】

[RK_MPI_VO_SetChnParam](#)

5.4.7 RK_MPI_SetChnDispPos

【描述】

设置指定视频输出通道的显示位置。

【语法】

```
RK_S32 RK_MPI_VO_SetChnDispPos(VO_LAYER VoLayer, VO_CHN VoChn, const POINT_S *pstDispPos)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstDispPos	通道显示位置。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pstDispPos为空指针。

5.4.8 RK_MPI_VO_GetChnDispPos

【描述】

获取指定视频输出通道的显示位置。

【语法】

```
RK_S32 RK_MPI_VO_GetChnDispPos(VO_LAYER VoLayer, VO_CHN VoChn, POINT_S *pstDispPos)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstDispPos	视频通道显示位置属性指针。	输出

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pstDispPos为空指针。

5.4.9 RK_MPI_VO_GetChnFrame

【描述】

获取通道帧。

【语法】

```
RK_S32 RK_MPI_VO_GetChnFrame(VO_LAYER VoLayer, VO_CHN VoChn, VIDEO_FRAME_INFO_S *pstFrame, RK_S32 s32MilliSec)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstVFrame	视频数据信息指针。	输入
s32MilliSec	超时时间的单位为毫秒（ms）。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pstFrame为空指针。

5.4.10 RK_MPI_VO_ReleaseChnFrame

【描述】

释放输出通道图像数据。

【语法】

```
RK_S32 RK_MPI_VO_ReleaseChnFrame(VO_LAYER VoLayer, VO_CHN VoChn, const VIDEO_FRAME_INFO_S *pstFrame)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstVFrame	释放的输出通道图像数据信息结构体指针。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pstFrame为空指针。

5.4.11 RK_MPI_VO_SendFrame

【描述】

将视频图像送入指定输出通道显示。

【语法】

```
RK_S32 RK_MPI_VO_SendFrame(VO_LAYER VoLayer, VO_CHN VoChn, VIDEO_FRAME_INFO_S *pstVFrame, RK_S32 s32MilliSec)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstVFrame	视频数据信息指针。	输入
s32MilliSec	超时时间的单位为毫秒（ms）。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pstFrame为空指针。
RK_ERR_VO_ILLEGAL_PARAM	pstFrame不正确。

【举例】

```
VIDEO_FRAME_INFO_S *pstVFrame;
RK_VOID *pMblk;
VO_LAYER VoVideoLayer;
VO_CHN VoChn;
RK_S32 ret = RT_OK;

pstVFrame = (VIDEO_FRAME_INFO_S *)(malloc(sizeof(VIDEO_FRAME_INFO_S)));
```

```

VoVideoLayer = RK356X_VOP_LAYER_CLUSTER_0;
VoChn = 0;
/*fill pMbBlk*/
pstVFrame->stVFrame.pMbBlk = pMbBlk;
do {
    ret = RK_MPI_VO_SendFrame(VoVideoLayer, VoChn, pstVFrame, -1);
    if (ret == RT_OK) {
        break;
    } else {
        RK_MPI_VO_DestroyGraphicsFrameBuffer(pMbBlk);
        free(pstVFrame);
        break;
    }
} while (1);

```

5.4.12 RK_MPI_VO_SetChnFrameRate

【描述】

设置指定视频输出通道的显示帧率。

【语法】

```
RK_S32 RK_MPI_VO_SetChnFrameRate(VO_LAYER VoLayer, VO_CHN VoChn, RK_S32 s32ChnFrmRate)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
s32VoFramerate	视频通道显示帧率。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
【举例】	

```

VO_LAYER  VoLayer = RK356X_VOP_LAYER_CLUSTER_0;
VO_CHN   VoChn = 0;
RK_S32   s32ChnFrmRate = 30;

s32Ret = RK_MPI_VO_SetChnFrameRate(VoLayer, VoChn, s32ChnFrmRate);
if (s32Ret != RK_VO_OK) {
    printf("Set channel %d frame rate failed with errno %#x!\n", VoChn,s32Ret);
    return RK_FAILURE;
}

```

【相关主题】

[RK_MPI_VO_GetChnFrameRate](#)

5.4.13 RK_MPI_VO_GetChnFrameRate

【描述】

获取指定视频输出通道的显示帧率。

【语法】

```
RK_S32 RK_MPI_VO_GetChnFrameRate(VO_LAYER VoLayer, VO_CHN VoChn, RK_S32 *ps32ChnFrmRate)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
ps32VoFramerate	视频通道显示帧率。	输出

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	ps32ChnFrmRate为空指针。

【相关主题】

[RK_MPI_VO_SetChnFrameRate](#)

5.4.14 RK_MPI_VO_PauseChn

【描述】

暂停指定的视频输出通道。

【语法】

```
RK_S32 RK_MPI_VO_PauseChn (VO_LAYER VoLayer, VO_CHN VoChn)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。

【举例】

```

RK_S32 i;
VO_LAYER VoLayer = 0;
RK_U32 u32Screen0Chn = 16;
while (1) {
    for (i = 0; i < u32Screen0Chn; i++) {
        RK_MPI_VO_PauseChn(VoLayer, i);
    }
    usleep(1000llu * 2000);
    for (i = 0; i < u32Screen0Chn; i++) {
        RK_MPI_VO_ResumeChn(VoLayer, i);
    }
}

```

【相关主题】

[RK_MPI_VO_PauseChn](#)

5.4.15 RK_MPI_VO_ResumeChn

【描述】

恢复指定的视频输出通道。

【语法】

```
RK_S32 RK_MPI_VO_ResumeChn(VO_LAYER VoLayer, VO_CHN VoChn)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。

【举例】

请参见[RK_MPI_VO_PauseChn](#)的举例。

5.4.16 RK_MPI_VO_StepChn

【描述】

单帧播放指定的视频输出通道。

【语法】

```
RK_S32 RK_MPI_VO_StepChn(VO_LAYER VoLayer, VO_CHN VoChn)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。

【举例】

```
RK_S32 i;
VO_LAYER VoLayer = 0;
RK_U32 u32ScreenChn = 16;
RK_S32 step_frame_num = 50;

while (1) {
    for (i = 0; i < ctx->u32ScreenChn; i++) {
        for (RK_S32 j = 0; j < step_frame_num; j++) {
            RK_MPI_VO_StepChn(VoLayer, i);
        }
    }
    usleep(1000llu * 2000);
    for (i = 0; i < ctx->u32Screen0Chn; i++) {
        RK_MPI_VO_ResumeChn(VoLayer, i);
    }
}
```

【相关主题】

[RK_MPI_VO_ResumeChn](#)

5.4.17 RK_MPI_VO_RefreshChn

【描述】

刷新指定的视频输出通道。

【语法】

```
RK_S32 RK_MPI_VO_RefreshChn(VO_LAYER VoLayer, VO_CHN VoChn)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。

【备注】

- 该函数暂未支持。

5.4.18 RK_MPI_VO_ShowChn

【描述】

显示指定通道。

【语法】

```
RK_S32 RK_MPI_VO_ShowChn (VO_LAYER VoLayer, VO_CHN VoChn)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。

【举例】

请参见[RK_MPI_VO_HideChn](#)的举例。

【相关主题】

[RK_MPI_VO_HideChn](#)

5.4.19 RK_MPI_VO_HideChn

【描述】

隐藏指定通道。

【语法】

```
RK_S32 RK_MPI_VO_HideChn(VO_LAYER VoLayer, VO_CHN VoChn)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确
RK_ERR_VO_INVALID_CHNID	VoChn不正确

【举例】

```

/* hide/show Chn 4 */
VO_LAYER VoLayer;
VoLayer = 0;
RK_MPI_VO_HideChn(VoLayer, 4);
usleep(1000llu * 5000);
RK_MPI_VO_ShowChn(VoLayer, 4);

```

【相关主题】

[RK_MPI_VO_ShowChn](#)

5.4.20 RK_MPI_VO_GetChnPts

【描述】

获取指定视频输出通道正在显示图像的时间戳。

【语法】

```
RK_S32 RK_MPI_VO_GetChnPts(VO_LAYER VoLayer, VO_CHN VoChn, RK_U64 *pu64ChnPts)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pu64VoChnPts	通道时间戳指针。	输出

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pu64ChnPts为空指针。
RK_ERR_VO_LAYER_NOT_ENABLE	VoLayer未使能。
RK_ERR_VO_CHN_NOT_ENABLE	VoChn未使能。

5.4.21 RK_MPI_VO_QueryChnStat

【描述】

查询视频输出通道状态。

【语法】

```
RK_S32 RK_MPI_VO_QueryChnStat(VO_LAYER VoLayer, VO_CHN VoChn, VO_QUERY_STATUS_S *pstStatus)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstStatus	通道状态结构体指针。	输出

【返回值】

返回值	描述
0	成功
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pstStatus为空指针。
RK_ERR_VO_LAYER_NOT_ENABLE	VoLayer未使能。
RK_ERR_VO_CHN_NOT_ENABLE	VoChn未使能。

5.4.22 RK_MPI_VO_ClearChnBuffer

【描述】

清空指定输出通道的缓存 buffer 数据。

【语法】

```
RK_S32 RK_MPI_VO_ClearChnBuffer(VO_LAYER VoLayer, VO_CHN VoChn, RK_BOOL bClrAll)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
bClrAll	是否将通道 buffer 中的数据全部清空。	输入

【返回值】

返回值	描述
0	成功
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确
RK_ERR_VO_INVALID_CHNID	VoChn不正确

【举例】

请参见[RK_MPI_VO_DisableChn](#)的举例。

5.4.23 RK_MPI_VO_SetChnBorder

【描述】

设置通道的边框属性。

【语法】

```
RK_S32 RK_MPI_VO_SetChnBorder(VO_LAYER VoLayer, VO_CHN VoChn, const VO_BORDER_S *pstBorder)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstBorder	通道边框属性指针。	输入

【返回值】

返回值	描述
0	成功
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pstBorder为空指针。

【举例】

```

VO_BORDER_S border;
RK_U32 s32Ret;
VO_LAYER VoLayer = RK356X_VOP_LAYER_ESMART_0;
VO_CHN VoChn = 0;

memset(&border, 0, sizeof(VO_BORDER_S));
/* set border */
border.bBorderEn = RK_TRUE;
/* Navy blue #000080 */
/* Midnight Blue #191970 */
border.stBorder.u32Color = 0x191970;
border.stBorder.u32LeftWidth = 2;
border.stBorder.u32RightWidth = 2;
border.stBorder.u32TopWidth = 2;
border.stBorder.u32BottomWidth = 2;
s32Ret = RK_MPI_VO_SetChnBorder(VoLayer, VoChn, &border);
if (s32Ret != RK_SUCCESS) {
    printf("Set channel %d Border failed with errno %#x!\n", VoChn, s32Ret);
}
}

```

5.4.24 RK_MPI_VO_GetChnBorder

【描述】

获取通道的边框属性

【语法】

```
RK_S32 RK_MPI_VO_GetChnBorder(VO_LAYER VoLayer, VO_CHN VoChn, VO_BORDER_S *pstBorder)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstBorder	通道边框属性指针。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pstBorder为空指针。

【相关主题】

[RK_MPI_VO_SetChnBorder](#)

5.4.25 RK_MPI_VO_SetChnRecvThreshold

【描述】

设置视频输出通道的显示门限值。主要应用场景：PCIV 级联，从片传输图像到主片进行预览显示时，由于传输到主片的图像的时间间隔是不均匀的，可能造成通道待显示图像超过默认的门限值，引起丢帧卡顿现象，需要调大門限值。如果需要显示稳定低延迟，则需要调小門限值。

【语法】

```
RK_S32 RK_MPI_VO_SetChnRecvThreshold(VO_LAYER VoLayer, VO_CHN VoChn, RK_U32 u32Threshold)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
u32Threshold	视频输出通道的显示门限值。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_ILLEGAL_PARAM	u32Threshold为非法值。

【注意】

- 显示門限取值范围为[1, 4]，默认門限为 3，表示该通道图像总的待显示帧数为 3，该通道缓冲队列最多可接收 2 帧图像，再加上正在（或即将）显示的一帧图像，即总的待显示帧数为 3。
- 門限值越小，预览延迟越小，反之越大，因此，正常情况下，不建议用户调大门限。
- 可动态调整門限值。如需恢复默认值，需由用户调用本接口将門限设回默认值。

【相关主题】

[RK_MPI_VO_GetChnRecvThreshold](#)

5.4.26 RK_MPI_VO_GetChnRecvThreshold

【描述】

获取视频输出通道的显示門限值。

【语法】

```
RK_S32 RK_MPI_VO_GetChnRecvThreshold(VO_LAYER VoLayer, VO_CHN VoChn, RK_U32 *pu32Threshold)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
VoChn	视频输出通道号，取值范围：[0, VO_MAX_CHN_NUM)。	输入
pu32Threshold	视频输出通道的显示门限值。	输出

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_LAYERID	VoLayer不正确。
RK_ERR_VO_INVALID_CHNID	VoChn不正确。
RK_ERR_VO_NULL_PTR	pu32Threshold为空指针。

【相关主题】

[RK_MPI_VO_SetChnRecvThreshold](#)

5.5 WBC回写操作

5.5.1 RK_MPI_VO_SetWbcSource

【描述】

设置回写通路的回写数据源。

【语法】

```
RK_S32 RK_MPI_VO_SetWbcSource(VO_WBC VoWbc, const VO_WBC_SOURCE_S *pstWbcSource)
```

【参数】

参数名称	描述	输入/输出
VoWbc	回写通路号。	输入
pstWbcSource	视频回写源结构体。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_WBCID	VoWbc不正确。
RK_ERR_VO_NULL_PTR	pstWbcSource为空指针。
RK_ERR_VO_NOT_SUPPORT	不支持pstWbcSource的enSourceType。

【举例】

请参见[RK_MPI_VO_EnableWbc](#)的举例。

【相关主题】

[RK_MPI_VO_GetWbcSource](#)

5.5.2 RK_MPI_VO_GetWbcSource

【描述】

获取回写通路的回写源。

【语法】

```
RK_S32 RK_MPI_VO_GetWbcSource(VO_WBC VoWbc, VO_WBC_SOURCE_S *pstWbcSources)
```

【参数】

参数名称	描述	输入/输出
VoWbc	回写通路号。	输入
pstWbcSource	视频回写源结构体。	输出

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_WBCID	VoWbc不正确。
RK_ERR_VO_NULL_PTR	pstWbcSources为空指针。

【举例】

请参见[RK_MPI_VO_EnableWbc](#)的举例。

【相关主题】

[RK_MPI_VO_SetWbcSource](#)

5.5.3 RK_MPI_VO_EnableWbc

【描述】

使能回写。

【语法】

```
RK_S32 RK_MPI_VO_EnableWbc(VO_WBC VoWbc)
```

【参数】

参数名称	描述	输入/输出
VoWbc	回写通路号。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_WBCID	VoWbc不正确。
RK_ERR_VO_NO_MEM	系统内存不足。
RK_ERR_VO_SYS_NOTREADY	回写相关服务创建失败。
RK_ERR_VO_NOT_SUPPORT	系统不支持。

【备注】

- 必须先设置视频回写属性，再使能视频回写功能。

- 重复使能视频回写返回成功。
- 视频回写要将回写源enSourceType设置为VO_WBC_SOURCE_VIDEO。
- 设备回写要将回写源enSourceType设置为VO_WBC_SOURCE_DEV。
- 视频回写功能支持Auto-bind模式和手动模式，Auto-bind模式需要绑定WBC到其他模块，比如VO模块或者VENC模块，但是不能绑定到VPSS模块。

【举例】

```

VO_WBC_SOURCE_S stWbcSource;
VO_WBC_ATTR_S stWbcAttr;
VO_WBC_MODE_E enWbcMode;
RK_U32    VoWbc;
RK_S32    s32Ret = RK_SUCCESS;
MPP_CHN_S stSrcChn, stDestChn;

VoWbc = 0;
stWbcSource.enSourceType = VO_WBC_SOURCE_VIDEO
stWbcSource.u32SourceId = RK356X_VO_DEV_HD0;
RK_MPI_VO_SetWbcSource(VoWbc, &stWbcSource);

stWbcAttr.stTargetSize.u32Width = 1920;
stWbcAttr.stTargetSize.u32Height = 1080;
stWbcAttr.enPixelFormat = RK_FMT_RGB888;
stWbcAttr.u32FrameRate = 25;
stWbcAttr.enCompressMode = COMPRESS_MODE_NONE;

s32Ret = RK_MPI_VO_SetWbcAttr(VoWbc, &stWbcAttr);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}

s32Ret = RK_MPI_VO_EnableWbc(VoWbc);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}

/* bind wbc to VO, i.e. */
stSrcChn.enModId = RK_ID_WBC;
stSrcChn.s32DevId = VoWbc;
stSrcChn.s32ChnId = 0;

stDestChn.enModId = RK_ID_VO;
stDestChn.s32ChnId = 0;
stDestChn.s32DevId = RK356X_VOP_LAYER_CLUSTER_1;

RK_MPI_SYS_Bind(&stSrcChn, &stDestChn);

/* disable wbc */
s32Ret = RK_MPI_VO_DisableWbc(VoWbc);
if (s32Ret != RK_SUCCESS) {
    return s32Ret;
}

```

【相关主题】

[RK_MPI_VO_DisableWbc](#)

5.5.4 RK_MPI_VO_DisableWbc

【描述】

禁用回写。

【语法】

```
RK_S32 RK_MPI_VO_DisableWbc(VO_WBC VoWbc)
```

【参数】

参数名称	描述	输入/输出
VoWbc	回写通路号。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_WBCID	VoWbc不正确。

5.5.5 RK_MPI_VO_SetWbcAttr

【描述】

设置视频回写属性。

【语法】

```
RK_S32 RK_MPI_VO_SetWbcAttr(VO_WBC VoWbc, const VO_WBC_ATTR_S *pstWbcAttr)
```

【参数】

参数名称	描述	输入/输出
VoWbc	回写通路号。	输入
pstWbcAttr	视频回写属性指针。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_WBCID	VoWbc不正确。
RK_ERR_VO_NULL_PTR	pstWbcAttr为空指针。
RK_ERR_VO_NOT_SUPPORT	不支持pstWbcAttr中enPixelFormat定义的颜色格式。

【备注】

- 视频回写最大输出分辨率1920x1080p。
- 支持放大，比如将720p源放大到1080p输出。
- 视频回写属性不支持NV12格式。

【举例】

请参见[RK_MPI_VO_EnableWbc](#)的举例。

【相关主题】

[RK_MPI_VO_GetWbcAttr](#)

5.5.6 RK_MPI_VO_GetWbcAttr

【描述】

获取视频回写属性。

【语法】

```
RK_S32 RK_MPI_VO_GetWbcAttr(VO_WBC VoWbc, VO_WBC_ATTR_S *pstWbcAttr)
```

【参数】

参数名称	描述	输入/输出
VoWbc	回写通路号。	输入
pstWbcAttr	视频回写属性指针。	输出

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_WBCID	VoWbc不正确。
RK_ERR_VO_NULL_PTR	pstWbcAttr为空指针。

【举例】

请参见[RK_MPI_VO_EnableWbc](#)的举例。

【相关主题】

[RK_MPI_VO_SetWbcAttr](#)

5.5.7 RK_MPI_VO_GetWbcFrame

【描述】

获取回写图像。

【语法】

```
RK_S32 RK_MPI_VO_GetWbcFrame(VO_WBC VoWbc, VIDEO_FRAME_INFO_S *pstVFrame, RK_S32 s32MilliSec)
```

【参数】

参数名称	描述	输入/输出
VoWbc	回写通路号。	输入
pstVFrame	获取的视频回写图像数据信息结构体指针。	输入
s32MilliSec	超时参数，目前设置为0。	输入

【返回值】

返回值	描述
0	成功。
RK_ERR_VO_INVALID_WBCID	VoWbc不正确。
RK_ERR_VO_NULL_PTR	pstVFrame为空指针。
RK_ERR_VO_WBC_NOT_CONFIG	VoWbc对应的回写失败未配置或使能。

5.5.8 RK_MPI_VO_ReleaseWbcFrame

【描述】

释放回写图像。

【语法】

```
RK_S32 RK_MPI_VO_ReleaseWbcFrame(VO_WBC VoWbc, VIDEO_FRAME_INFO_S *pstVFrame)
```

【参数】

参数名称	描述	输入/输出
VoWbc	回写通路号。	输入
pstVFrame	释放的输出通道图像数据信息结构体指针。	输入

【返回值】

返回值	描述
0	成功
RK_ERR_VO_INVALID_WBCID	VoWbc不正确
RK_ERR_VO_NULL_PTR	pstVFrame为空指针

5.5.9 RK_MPI_VO_SetWbcDepth

【描述】

设置回写缓存个数

【语法】

```
RK_S32 RK_MPI_VO_SetWbcDepth(VO_WBC VoWbc, RK_S32 u32Depth)
```

【参数】

参数名称	描述	输入/输出
VoWbc	回写通路号。	输入
u32Depth	缓存个数	输入

6. 数据类型

视频输出相关数据类型、数据结构定义如下：

- VO_DEV：定义设备号。
- VO_LAYER：定义视频层号。
- VO_CHN：定义视频输出通道号。
- VO_WBC：定义回写通路号。
- VO_MAX_DEV_NUM：定义视频输出设备最大个数。
- VO_MAX_LAYER_NUM：定义图层最大个数。
- VO_MAX_CHN_NUM：定义通道最大个数。
- VO_MAX_WBC_NUM：定义回写设备最大个数。
- VO_LAYER_MODE_E：定义图层类型。
- VO_PUB_ATTR_S：定义视频输出设备属性结构体。
- VO_VIDEO_LAYER_ATTR_S：定义视频输出公共属性结构体。
- VO_WBC_ATTR_S：定义视频回写属性结构体。
- VO_WBC_SOURCE_S：定义视频回写源。

6.1 VO_DEV

【说明】

定义显示输出设备号。

【定义】

```
typedef RK_S32 VO_DEV;
```

6.2 VO_LAYER

【说明】

定义图层号。

【定义】

```
typedef RK_S32 VO_LAYER;
```

【备注】

取值参考[VO_LAYER_NAME_RK356X_E](#)、[VO_LAYER_NAME_E](#)和[VO_VIR_LAYER_NAME_E](#)。

6.3 VO_CHN

【说明】

定义视频输出通道号。

【定义】

```
typedef RK_S32 VO_CHN;
```

6.4 VO_WBC

【说明】

定义回写通路号。

【定义】

```
typedef RK_S32 VO_WBC;
```

6.5 VO_MAX_DEV_NUM

【说明】

定义视频输出设备最大个数。

【定义】

```
#define VO_MAX_DEV_NUM      3
```

6.6 VO_MAX_LAYER_NUM

【说明】

定义图层最大个数。

【定义】

```
#define VO_MAX_LAYER_NUM     8
```

6.7 VO_MAX_CHN_NUM

【说明】

定义通道最大个数。

【定义】

```
#define VO_MAX_CHN_NUM      128
```

6.8 VO_MAX_WBC_NUM

【说明】

定义回写设备最大个数。

【定义】

```
#define VO_MAX_WBC_NUM      1
```

6.9 VO_GFX_MODE_E

【说明】

定义图层buffer创建模式。

【定义】

```
typedef enum rkVO_GFX_MODE_E {  
    VO_MODE_NORMAL,  
    VO_MODE_GFX_PRE_CREATED  
} VO_GFX_MODE_E;
```

【成员】

成员名称	描述
VO_MODE_NORMAL	正常模式。
VO_MODE_GFX_PRE_CREATED	预创建模式。

6.10 VO_HDMI_MODE_E

【说明】

定义HDMI输出模式

【定义】

```
typedef enum rkVO_HDMI_MODE_E {  
    VO_HDMI_MODE_AUTO = 0,  
    VO_HDMI_MODE_HDMI,  
    VO_HDMI_MODE_DVI  
} VO_HDMI_MODE_E;
```

【成员】

成员名称	描述
VO_HDMI_MODE_AUTO	根据EDID自动选择
VO_HDMI_MODE_HDMI	强制HDMI模式
VO_HDMI_MODE_DVI	强制DVI模式

6.11 VO_HDMI_COLOR_FMT_E

【说明】

定义HDMI输出颜色格式

【定义】

```
typedef enum rkVO_HDMI_COLOR_FMT_E {
    VO_HDMI_COLOR_FORMAT_RGB = 0,
    VO_HDMI_COLOR_FORMAT_YCBCR444,
    VO_HDMI_COLOR_FORMAT_YCBCR422,
    VO_HDMI_COLOR_FORMAT_YCBCR420,
    VO_HDMI_COLOR_FORMAT_AUTO,
    VO_HDMI_COLOR_FORMAT_BUTT
} VO_HDMI_COLOR_FMT_E;
```

【成员】

成员名称	描述
VO_HDMI_COLOR_FORMAT_RGB	RGB
VO_HDMI_COLOR_FORMAT_YCBCR444	YCbCr444
VO_HDMI_COLOR_FORMAT_YCBCR422	YCbCr422
VO_HDMI_COLOR_FORMAT_YCBCR420	YCbCr420
VO_HDMI_COLOR_FORMAT_AUTO	根据EDID按如下顺序选择， YCbCr444->YCbCr422->YCbCr420->RGB

6.12 VO_HDMI_QUANT_RANGE_E

【说明】

定义HDMI输出量化范围

【定义】

```
typedef enum rkVO_HDMI_QUANT_RANGE_E {
    VO_HDMI_QUANT_RANGE_AUTO = 0,
    VO_HDMI_QUANT_RANGE_LIMITED,
    VO_HDMI_QUANT_RANGE_FULL,
    VO_HDMI_QUANT_RANGE_BUTT
} VO_HDMI_QUANT_RANGE_E;
```

【成员】

成员变量	描述
VO_HDMI_QUANT_RANGE_AUTO	自动
VO_HDMI_QUANT_RANGE_LIMITED	量化范围【16-235】
VO_HDMI_QUANT_RANGE_FULL	量化范围【0-255】

【备注】

- 仅在输出RGB颜色格式时有效

6.13 VO_PUB_ATTR_S

【说明】

视频输出公共属性结构体。

【定义】

```
typedef struct rkVO_PUB_ATTR_S {  
    RK_U32 u32BgColor;  
    VO_INTF_TYPE_E enIntfType;  
    VO_INTF_SYNC_E enIntfSync;  
    VO_SYNC_INFO_S stSyncInfo;  
} VO_PUB_ATTR_S;
```

【成员】

成员名称	描述
u32BgColor	未使用。
enIntfType	显示接口类型，可以选择以下值或组合： VO_INTF_CVBS VO_INTF_YPBPR VO_INTF_VGA VO_INTF_BT656 VO_INTF_BT1120 VO_INTF_LCD VO_INTF_LVDS VO_INTF_MIPI VO_INTF_MIPI1 VO_INTF_EDP VO_INTF_EDP1 VO_INTF_HDMI VO_INTF_HDMI1 VO_INTF_DP VO_INTF_DP1
enIntfSync	显示接口时序。
stSyncInfo	接口时序信息。 intf_sync 配置用户时序 RK_VO_OUT_USER 时，该结构体生效。

【备注】

- 当系统带有两个同类型输出接口时，VO_INTF_MIPI1、VO_INTF_EDP1、VO_INTF_HDMI1、VO_INTF_DP1用于表示第二个输出接口。
- RK3588 绑定多个接口到同一个VoDev时，接口输出的颜色格式必须相同。比如DP和HDMI同时输出RGB888。

6.14 VO_EDID_S

【说明】

EDID数据结构体

【定义】

```
typedef struct rk_VO_EDID_S {
    RK_BOOL      bEdidValid;
    RK_U32       u32Edidlenth;
    RK_U8        u8Edid[512];
} VO_EDID_S;
```

【成员】

成员名称	描述
bEdidValid	EDID合法标志。
u32Edidlenth	EDID数据长度。
u8Edid	EDID数据。

6.15 VO_SINK_CAPABILITY_S

【说明】

显示设备状态属性结构

【定义】

```
typedef struct rk_VO_SINK_CAPABILITY_S {
    RK_BOOL      bConnected;
    RK_BOOL      bSupportYCbCr;
    RK_BOOL      bSupportHDMI;
} VO_SINK_CAPABILITY_S;
```

【成员】

成员名称	描述
bConnected	显示设备连接状态, RK_TRUE: 连接; RK_FALSE: 断开
bSupportYCbCr	显示设备是否支持YCbCr数据, RK_TRUE: 支持; RK_FALSE: 不支持
bSupportHDMI	显示设备是否支持HDMI, RK_TRUE: 支持; RK_FALSE: 不支持

6.16 RK_VO_CallBack

【说明】

显示设备热插拔事件回调函数

【定义】

```
typedef void (*RK_VO_CallBack)(RK_VOID *pPrivateData);
```

6.17 RK_VO_CALLBACK_FUNC_S

【说明】

显示设备热插拔事件回调函数结构体

【定义】

```
typedef struct rk_VO_CALLBACK_FUNC_S {
    RK_VO_CallBack  pfnEventCallback;
    RK_VOID        *pPrivateData;
} RK_VO_CALLBACK_FUNC_S;
```

【成员】

成员名称	描述
pfnEventCallback	回调函数。
pPrivateData	回调函数私有变量。

6.18 RK_VO_VsyncCallBack

【说明】

显示设备帧中断事件回调函数

【定义】

```
typedef void (*RK_VO_CallBack)(RK_VOID *pPrivateData, VO_CB_INFO_S* info);
```

6.19 RK_VO_CALLBACK_FUNC_S

【说明】

显示设备帧中断回调函数结构体

【定义】

```
typedef struct rk_VO_CALLBACK_FUNC_S {
    RK_VO_VsyncCallBack pfnEventCallback;
    RK_VOID        *pPrivateData;
} RK_VO_CALLBACK_FUNC_S;
```

【成员】

成员名称	描述
pfnEventCallback	回调函数。
pPrivateData	回调函数私有变量。

6.20 VO_CB_INFO_S

【说明】

显示设备帧中断事件信息

【定义】

```
typedef struct rk_VO_CALLBACK_FUNC_S {
    RK_U32 u32Id;
    RK_U32 u32Sec;
    RK_U32 u32Usec;
} RK_VO_CALLBACK_FUNC_S;
```

【成员】

成员名称	描述
u32Id	显示设备号。
u32Sec	当前中断时间 (s)。
u32Usec	当前中断时间 (us)。

6.21 VO_HDMI_PARAM_S

【说明】

HDMI属性结构体

【定义】

```
typedef struct rk_VO_HDMI_PARAM_S {  
    VO_HDMI_MODE_E enHdmiMode;  
    VO_HDMI_COLOR_FMT_E enColorFmt;  
    VO_HDMI_QUANT_RANGE_E enQuantRange;  
} VO_HDMI_PARAM_S;
```

【成员】

成员名称	描述
enHdmiMode	输出模式
enColorFmt	输出颜色格式
enQuantRange	输出量化范围

【备注】

- enQuantRange仅在enColorFmt为RGB格式时生效。

6.22 VO_FRAME_INFO_S

【说明】

定义图形帧信息结构体。

【定义】

```
typedef struct rkVO_GFX_FRAME_INFO_S {  
    RK_U32      u32Width;  
    RK_U32      u32Height;  
    RK_U32      u32VirWidth;  
    RK_U32      u32VirHeight;  
    PIXEL_FORMAT_E   enPixelFormat;  
    RK_U32      u32FgAlpha;  
    RK_U32      u32BgAlpha;  
    RK_VOID     *pData;  
    RK_U32      u32Size;  
} VO_FRAME_INFO_S;
```

【成员】

成员名称	描述
u32Width	图像宽度，以像素为单位。
u32Height	图像高度，以像素为单位。
u32VirWidth	图像虚宽，以像素为单位。
u32VirHeight	图像虚高，以像素为单位。
enPixelFormat	像素格式。
u32FgAlpha	前景Alpha值。
u32BgAlpha	背景Alpha值。
pData	内存地址。
u32Size	内存大小。
【备注】	

- 图形通道数据格式为BGRA5551/RGBA5551时，需要填写u32FgAlpha和u32BgAlpha。

6.23 VO_LAYER_NAME_RK356X_E

【说明】

定义RK356X图层号。

【定义】

```
typedef enum rk356X_VO_LAYER_NAME_E {
    RK356X_VOP_LAYER_CLUSTER0 = 0,
    RK356X_VOP_LAYER_CLUSTER1 = 2,
    RK356X_VOP_LAYER_ESMART0,
    RK356X_VOP_LAYER_ESMART1,
    RK356X_VOP_LAYER_SMART0,
    RK356X_VOP_LAYER_SMART1,
} VO_LAYER_NAME_RK356X_E;
```

【成员】

成员名称	描述
RK356X_VOP_LAYER_CLUSTER0	CLUSTER图层0。
RK356X_VOP_LAYER_CLUSTER1	CLUSTER图层1。
RK356X_VOP_LAYER_ESMART0	ESMART图层0。
RK356X_VOP_LAYER_ESMART1	ESMART图层1。
RK356X_VOP_LAYER_SMART0	SMART图层0。
RK356X_VOP_LAYER_SMART1	SMART图层1。

【备注】

适用于RK356X系列。

6.24 VO_LAYER_NAME_E

【说明】

定义图层号。

【定义】

```

typedef enum rkVOP2_LAYER_NAME_E {
    VO_LAYER_CLUSTER0 = 0,
    VO_LAYER_CLUSTER1,
    VO_LAYER_CLUSTER2,
    VO_LAYER_CLUSTER3,
    VO_LAYER_ESMART0,
    VO_LAYER_ESMART1,
    VO_LAYER_ESMART2,
    VO_LAYER_ESMART3,
    VO_LAYER_BUTT
} VO_LAYER_NAME_E;

```

【成员】

成员名称	描述
VO_LAYER_CLUSTER0	CLUSTER图层0。
VO_LAYER_CLUSTER1	CLUSTER图层1。
VO_LAYER_CLUSTER2	CLUSTER图层2。
VO_LAYER_CLUSTER3	CLUSTER图层3。
VO_LAYER_ESMART0	ESMART图层0。
VO_LAYER_ESMART1	ESMART图层1。
VO_LAYER_ESMART2	ESMART图层2。
VO_LAYER_ESMART3	ESMART图层3。

6.25 VO_VIR_LAYER_NAME_E

【说明】

定义虚拟图层号。

【定义】

```

typedef enum rkVO_VIR_LAYER_NAME_E {
    VO_LAYER_VIRTUAL0 = VO_LAYER_BUTT,
    VO_LAYER_VIRTUAL1,
    VO_LAYER_VIRTUAL2,
    VO_LAYER_VIRTUAL3,
} VO_VIR_LAYER_NAME_E;

```

【成员】

成员名称	描述
VO_LAYER_VIRTUAL0	虚拟图层0。
VO_LAYER_VIRTUAL1	虚拟图层1。
VO_LAYER_VIRTUAL2	虚拟图层2。
VO_LAYER_VIRTUAL3	虚拟图层3。

6.26 VO_LAYER_MODE_E

【说明】

定义图层类型。

【定义】

```
typedef enum rkVO_LAYER_MODE_E {
    VO_LAYER_MODE_CURSOR = 0,
    VO_LAYER_MODE_GRAPHIC,
    VO_LAYER_MODE_VIDEO,
    VO_LAYER_MODE_VIRTUAL,
    VO_LAYER_MODE_BUTT
} VO_LAYER_MODE_E;
```

【成员】

成员名称	描述
VO_LAYER_MODE_CURSOR	鼠标图层。
VO_LAYER_MODE_GRAPHIC	图形图层。
VO_LAYER_MODE_VIDEO	视频图层。
VO_LAYER_MODE_VIRTUAL	虚拟图层。

6.27 VO_SPLICE_MODE_E

【说明】

定义图层合成类型。

【定义】

```
typedef enum rkVO_SPLICE_MODE_E {
    VO_SPLICE_MODE_GPU = 0,
    VO_SPLICE_MODE_RGA
} VO_SPLICE_MODE_E;
```

【成员】

成员名称	描述
VO_SPLICE_MODE_GPU	GPU合成模式。
VO_SPLICE_MODE_RGA	RGA合成模式。

6.28 VO_VIDEO_LAYER_ATTR_S

【说明】 定义视频层属性。

【定义】

```
typedef struct rkVO_VIDEO_LAYER_ATTR_S {
    RECT_S stDispRect;
    SIZE_S stImageSize;
    RK_U32 u32DispFrmRt;
    PIXEL_FORMAT_E enPixFormat;
    RK_BOOL bBypassFrame;
    RK_BOOL bLowDelay;
    COMPRESS_MODE_E enCompressMode;
    DYNAMIC_RANGE_E enDstDynamicRange;
} VO_VIDEO_LAYER_ATTR_S;
```

【成员】

成员名称	描述
stDispRect	显示分辨率大小。
stImageSize	视频层画布大小。
u32DispFrmRt	显示帧率。
enPixelFormat	视频层使用的像素格式。
bBypassFrame	视频帧不经过拼接，直送给视频图层。
bLowDelay	使用通道0的送帧事件，触发拼接，拼接后立即送显
enCompressMode	视频层帧缓存压缩模式。
enDstDynamicRange	未使用。

6.29 VO_CHN_ATTR_S

【说明】

通道属性结构体。

【定义】

```
typedef struct rkVO_CHN_ATTR_S {
    RK_U32 u32Priority; /* Video out overlay pri sd */
    RECT_S stRect; /* Rectangle of video output channel */
    RK_BOOL bDeflicker; /* Deflicker or not sd */
    RK_U32 u32FgAlpha; /* Alpha of A = 1 in pixel format BGRA5551/RGBA5551 */
    RK_U32 u32BgAlpha; /* Alpha of A = 0 in pixel format BGRA5551/RGBA5551 */
    RK_BOOL bEnKeyColor; /* Enable key color or not when pixel format BGRA5551/RGBA5551 */
    RK_U32 u32KeyColor; /* Key color value of pixel format BGRA5551/RGBA5551, B[0:4] G[5:9] R[10:14] */
    MIRROR_E enMirror; /* RW, Mirror */
    ROTATION_E enRotation; /* RW, rotation. */
    RK_U32 u32MaxChnQueue; /* vo channel max queue length */
} VO_CHN_ATTR_S;
```

【成员】

成员名称	描述
u32Priority	通道优先级，通道显示位置有交叠时，高优先级通道覆盖低优先级通道。
stRect	通道在图层画布上的显示区域。
bDeflicker	不支持。
u32FgAlpha	BGRA5551/RGBA5551格式是，A=1对应的Alpha值。
u32BgAlpha	BGRA5551/RGBA5551格式是，A=0对应的Alpha值。
bEnKeyColor	当像素格式为BGRA5551/RGBA5551时，使能/禁用关键色功能
u32KeyColor	当像素格式为BGRA5551/RGBA5551时关键色
enMirror	使能mirror
enRotation	使能rotation
u32MaxChnQueue	vo通道queue最大长度，未设置时，默认最大为10
【备注】	

- 通道数据格式为BGRA5551/RGBA5551时，需要填写u32FgAlpha和u32BgAlpha。
- u32Priority 动态属性，下一个合成周期生效。
- stRect 动态属性，下一个合成周期生效。

- u32FgAlpha动态属性，下一个合成周期生效，RGA合成模式下此参数无效。
- u32BgAlpha动态属性，下一个合成周期生效，RGA合成模式下此参数无效。
- bEnKeyColor动态属性，下一个合成周期生效，RGA合成模式下此参数无效。
- u32KeyColor动态属性，下一个合成周期生效，RGA合成模式下此参数无效。
- enMirror动态属性，下一个合成周期生效。
- enRotation动态属性，下一个合成周期生效。

6.30 VO_CHN_PARAM_S

【说明】

通道参数结构体。

【定义】

```
typedef struct rkVO_CHN_PARAM_S {
    ASPECT_RATIO_S stAspectRatio; /* RW; aspect ratio */
} VO_CHN_PARAM_S;
```

【成员】

成员名称	描述
stAspectRatio	画面宽高比属性。
【备注】	

- stAspectRatio动态属性，下一个合成周期生效，RGA合成模式下此参数无效。

6.31 VO_BORDER_S

【说明】

边框属性结构体。

【定义】

```
typedef struct rkVO_BORDER_S {
    RK_BOOL bBorderEn; /* RW; Do frame or not */
    BORDER_S stBorder; /* RW; frame's top, bottom, left, right width and color */
} VO_BORDER_S;
```

【成员】

成员名称	描述
bBorderEn	使能边框。
stBorder	边框属性。
【备注】	

- bBorderEn动态属性，下一个合成周期生效，RGA合成模式下此参数无效。
- stBorder动态属性，下一个合成周期生效，RGA合成模式下此参数无效。

6.32 VO_WBC_ATTR_S

【说明】

定义视频回写属性。

【定义】

```

typedef struct rkVO_WBC_ATTR_S {
    SIZE_S stTargetSize;
    PIXEL_FORMAT_E enPixelFormat;
    RK_U32 u32FrameRate;
    DYNAMIC_RANGE_E enDynamicRange;
    COMPRESS_MODE_E enCompressMode;
} VO_WBC_ATTR_S;

```

【成员】

成员名称	描述
stTargetSize	视频回写图像大小。
enPixelFormat	视频回写数据格式。
u32FrameRate	未使用。
enDynamicRange	未使用。
enCompressMode	视频回写压缩模式。

6.33 VO_WBC_SOURCE_TYPE_E

【说明】

定义视频回写类型。

【定义】

```

typedef enum rkVO_WBC_SOURCE_TYPE_E {
    VO_WBC_SOURCE_DEV = 0x0, /* WBC source is device */
    VO_WBC_SOURCE_VIDEO = 0x1, /* WBC source is video layer */
    VO_WBC_SOURCE_GRAPHIC = 0x2, /* WBC source is graphic layer. Warning: not supported */
    VO_WBC_SOURCE_VIRTUAL = 0x3, /* WBC source is virtual layer */
    VO_WBC_SOURCE_BUTT
} VO_WBC_SOURCE_TYPE_E;

```

【成员】

成员名称	描述
VO_WBC_SOURCE_DEV	回写源为设备。
VO_WBC_SOURCE_VIDEO	回写源为视频层。
VO_WBC_SOURCE_GRAPHIC	回写源为图形层，暂不支持。
VO_WBC_SOURCE_VIRTUAL	回写源为虚拟图层。

6.34 VO_WBC_SOURCE_S

【说明】

定义视频回写源。

【定义】

```

typedef struct rkVO_WBC_SOURCE_S {
    VO_WBC_SOURCE_TYPE_E enSourceType;
    RK_U32 u32SourceId;
} VO_WBC_SOURCE_S;

```

【成员】

成员名称	描述
enSourceType	视频回写源类型枚举。
u32SourceId	视频回写源的编号。
【备注】	

- enSourceType为VO_WBC_SOURCE_VIRTUAL时，u32SourceId为虚拟图层ID。
- enSourceType为VO_WBC_SOURCE_DEV/VO_WBC_SOURCE_VIDEO/VO_WBC_SOURCE_GRAPHIC时，u32SourceId为VoDev ID。

7. VO错误码

视频输出 API VO错误码如下所示：

错误代码	宏定义	描述
0xA0098012	RK_ERR_VO_BUSY	VO相关服务创建、使能输出、设置分辨率等操作失败
0xA009800C	RK_ERR_VO_NO_MEM	系统内存不足
0xA0098006	RK_ERR_VO_NULL_PTR	函数参数中有空指针
0xA0098010	RK_ERR_VO_SYS_NOTREADY	VO相关服务创建失败
0xA0098001	RK_ERR_VO_INVALID_DEVID	设备 ID 超出合法范围
0xA0098002	RK_ERR_VO_INVALID_CHNID	通道 ID 超出合法范围
0xA0098003	RK_ERR_VO_ILLEGAL_PARAM	参数超出合法范围
0xA0098008	RK_ERR_VO_NOT_SUPPORT	VO不支持输入的参数
0xA0098009	RK_ERR_VO_NOT_PERMIT	操作不允许
0xA0098058	RK_ERR_VO_INVALID_WBCID	WBC 号超出范围
0xa0098059	RK_ERR_VO_INVALID_LAYERID	视频层号超出范围
0xA0098041	RK_ERR_VO_DEV_NOT_ENABLE	对应的设备未使能
0xA0098042	RK_ERR_VO_DEV_HAS_ENABLED	对应的设备已使能
0xA0098043	RK_ERR_VO_DEV_HAS_BINDED	设备已经绑定过其他显示输出设备
0xA0098044	RK_ERR_VO_DEV_NOT_BINDED	显示输出设备未被绑定
0xA0098045	RK_ERR_VO_LAYER_NOT_ENABLE	图层未使能
0xA0098047	RK_ERR_VO_LAYER_NOT_CONFIG	图层未配置
0xA009805b	RK_ERR_VO_LAYER_NOT_BINDED	图层未绑定
0xA0098055	RK_ERR_VO_WBC_NOT_DISABLE	WBC 未禁止
0xA0098056	RK_ERR_VO_WBC_NOT_CONFIG	WBC 属性未设置
0xA0098049	RK_ERR_VO_CHN_NOT_ENABLE	通道未使能

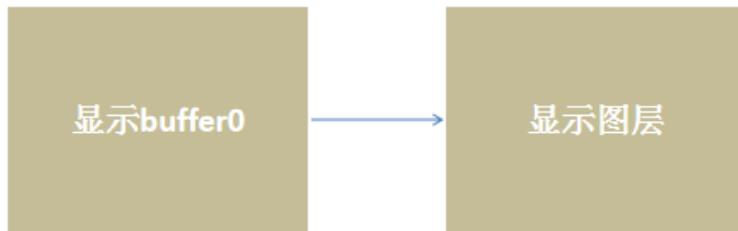
8. UI绘制说明

目前提供的API可以支持如下几种场景的UI绘制。

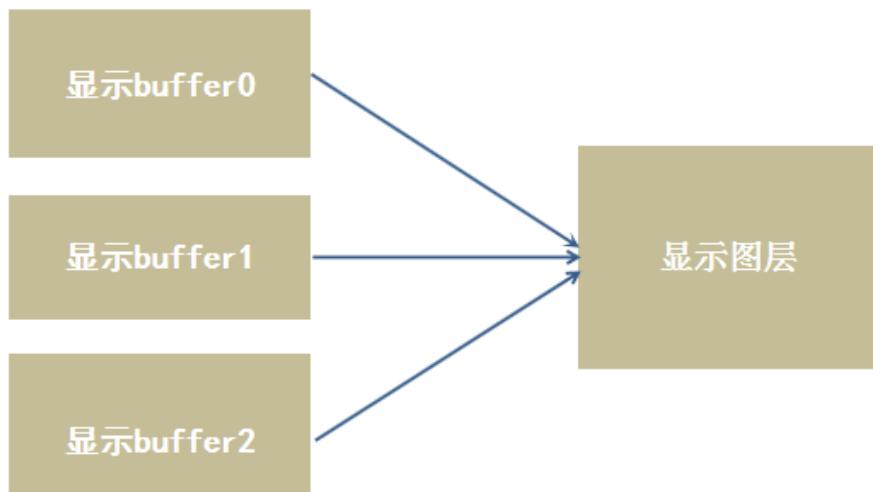
UI绘制方式	VOP图层	显示buffer模式	申请buffer方法	GPU参与合成	送VO方法	支持格式
1	UI图层	单buffer	RK_MPI_VO_CreateGraphicsFrameBuffer	否	RK_MPI_VO_SendLayerFrame	ARGB8888
2	UI图层	多buffer	RK_MPI_VO_CreateGraphicsFrameBuffer	否	RK_MPI_VO_SendLayerFrame	ARGB8888
3	UI图层	单buffer	RK_MPI_VO_GetGfxFrameBuffer	是	无需送帧函数	ARGB1555/ARGB8888
4	UI图层	多buffer	RK_MPI_VO_CreateGraphicsFrameBuffer	是	RK_MPI_VO_SendFrame	ARGB1555/ARGB8888
5	视频图层	单buffer	RK_MPI_VO_GetGfxFrameBuffer	是	无需送帧函数	ARGB1555/ARGB8888
6	视频图层	多buffer	RK_MPI_VO_CreateGraphicsFrameBuffer	是	RK_MPI_VO_SendFrame	ARGB1555/ARGB8888

【备注】

- 单buffer示意图如下图所示。



- 多buffer示意图如下图所示。



8.1 【方式1举例】

```

VO_PUB_ATTR_S      stVoPubAttr;
VO_VIDEO_LAYER_ATTR_S stLayerAttr;
VIDEO_FRAME_INFO_S  stVFrame;
VO_LAYER VoLayer;
VO_DEV VoDev;
RK_VOID *pMbblk = RK_NULL;
RK_U32 s32Ret = RK_SUCCESS;
RK_U32 u32BuffSize;

/* alloc single buffer, and fill pMbblk for UI data */
u32BuffSize = RK_MPI_VO_CreateGraphicsFrameBuffer(1920, 1080, RK_FMT_ARGB8888, &pMbblk);
/* store data to stVFrame.stVFrame.pMbblk */

...
/* Bind Layer */
  
```

```

VoLayer = RK356X_VOP_LAYER_ESMART_0;
VoDev == RK356X_VO_DEV_HD0;

RK_MPI_VO_BindLayer(VoLayer, RK356X_VO_DEV_HD0, VO_LAYER_MODE_GRAPHIC);

stVoPubAttr.enIntfType = VO_INTF_HDMI;
stVoPubAttr.enIntfSync = VO_OUTPUT_1080P60;

/* Enable VO Device */
RK_MPI_VO_SetPubAttr(VoDev, &stVoPubAttr);
RK_MPI_VO_Enable(VoDev);
/* Enable Layer */
stLayerAttr.enPixelFormat = RK_FMT_ARGB8888;
stLayerAttr.stDispRect.s32X = 0;
stLayerAttr.stDispRect.s32Y = 0;
tLayerAttr.stDispRect.u32Width = 1920;
stLayerAttr.stDispRect.u32Height = 1080;
stLayerAttr.stImageSize.u32Width = 1920;
stLayerAttr.stImageSize.u32Height = 1080;
/* Set Layer */
RK_MPI_VO_SetLayerAttr(VoLayer, &stLayerAttr);
RK_MPI_VO_EnableLayer(VoLayer);

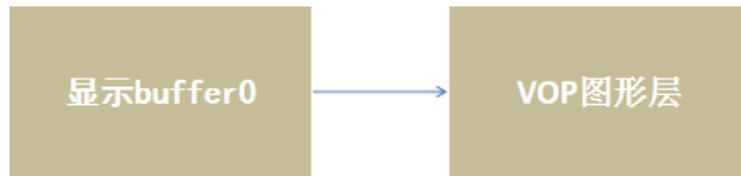
stVFrame.stVFrame.pMbBlk = pMbBlk;
RK_MPI_VO_SendLayerFrame(VoLayer, &stVFrame);

```

【备注】

- 方式1绘制UI的方式是通过RK_MPI_VO_CreateGraphicsFrameBuffer外部申请到单buffer后，将buffer送UI图层显示，如下图所示。

CreateGraphicsFrameBuffer



8.2 【方式2举例】

```

VO_PUB_ATTR_S     stVoPubAttr;
VO_VIDEO_LAYER_ATTR_S stLayerAttr;
VIDEO_FRAME_INFO_S stVFrame;
VO_LAYER VoLayer;
VO_DEV VoDev;
RK_VOID *pMbBlk = RK_NULL;
RK_VOID *pMbBlk_480P = RK_NULL;
RK_U32 s32Ret = RK_SUCCESS;
RK_U32 u32BuffSize;
RK_U32 count;

/* alloc muti buffer, and fill pMbBlk for UI data */
u32BuffSize = RK_MPI_VO_CreateGraphicsFrameBuffer(1920, 1080, RK_FMT_ARGB8888, &pMbBlk);
u32BuffSize = RK_MPI_VO_CreateGraphicsFrameBuffer(1920, 1080, RK_FMT_ARGB8888, &pMbBlk_480P);
/* store data to stVFrame.stVFrame.pMbBlk */
...

/* Bind Layer */
VoLayer = RK356X_VOP_LAYER_ESMART_0;
VoDev == RK356X_VO_DEV_HD0;

```

```

RK_MPI_VO_BindLayer(VoLayer, RK356X_VO_DEV_HD0, VO_LAYER_MODE_GRAPHIC);

stVoPubAttr.enIntfType = VO_INTF_HDMI;
stVoPubAttr.enIntfSync = VO_OUTPUT_1080P60;

/* Enable VO Device */
RK_MPI_VO_SetPubAttr(VoDev, &stVoPubAttr);
RK_MPI_VO_Enable(VoDev);
/* Enable Layer */
stLayerAttr.enPixFormat      = RK_FMT_ARGB8888;
stLayerAttr.stDispRect.s32X   = 0;
stLayerAttr.stDispRect.s32Y   = 0;
tLayerAttr.stDispRect.u32Width = 1920;
stLayerAttr.stDispRect.u32Height = 1080;
stLayerAttr.stImageSize.u32Width = 1920;
stLayerAttr.stImageSize.u32Height = 1080;
/* Set Layer */
RK_MPI_VO_SetLayerAttr(VoLayer, &stLayerAttr);
RK_MPI_VO_EnableLayer(VoLayer);

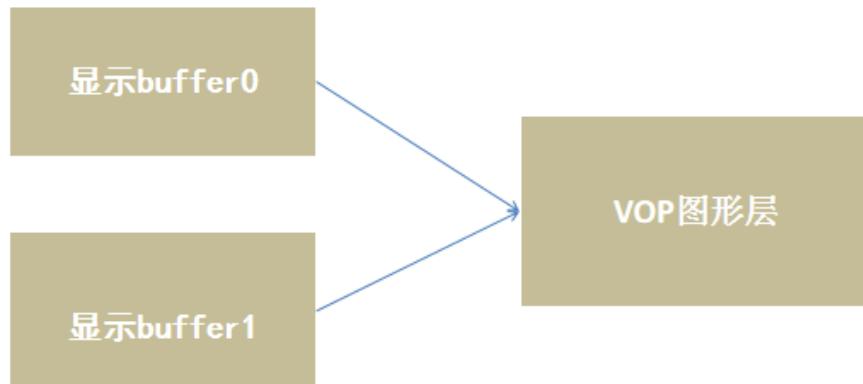
while (1) {
    count++;
    if (count % 2) {
        stVFrame.stVFrame.pMbBlk = pMbBlk;
        RK_MPI_VO_SendLayerFrame(VoLayer, &stVFrame);
    } else {
        stVFrame.stVFrame.pMbBlk = pMbBlk_480P;
        RK_MPI_VO_SendLayerFrame(VoLayer, &stVFrame);
    }
}

```

【备注】

- 该示例绘制UI的方式是通过RK_MPI_VO_CreateGraphicsFrameBuffer外部申请到两个buffer后，将buffer直接送UI图形层显示，如下图所示。

CreateGraphicsFrameBuffer



8.3 【方式3举例】

```

VO_PUB_ATTR_S     stVoPubAttr;
VO_FRAME_INFO_S   stVFrame;
VO_LAYER VoLayer;
VO_VIDEO_LAYER_ATTR_S stLayerAttr;
RK_U32 s32Ret = RK_SUCCESS;

stLayerAttr.enPixFormat      = RK_FMT_ARGB8888;
stLayerAttr.stDispRect.s32X   = 0;
stLayerAttr.stDispRect.s32Y   = 0;

```

```

stLayerAttr.stDispRect.u32Width    = 1920;
stLayerAttr.stDispRect.u32Height   = 1080t;
stLayerAttr.stImageSize.u32Width   = 1920;
stLayerAttr.stImageSize.u32Height  = 1080;
VoDev = RK356X_VO_DEV_HD0;
VoLayer = RK356X_VOP_LAYER_ESMART_0; /* UI layer esmart0 */
RK_MPI_VO_SetGfxMode(VO_MODE_GFX_PRE_CREATED);
stVFrame.enPixelFormat = RK_FMT_ARGB8888;
stVFrame.u32Width = 1920;
stVFrame.u32Height = 1080;
RK_MPI_VO_GetGfxFrameBuffer(VoLayer, 0, &stVFrame);
/* fill stVFrame */
memset(stVFrame.pData, 0xff, stVFrame.u32Size);
stVoPubAttr.enIntfSync = VO_OUTPUT_1080P60;
stVoPubAttr.enIntfType = VO_INTF_HDMI;
RK_MPI_VO_SetPubAttr(VoDev, &pstPubAttr);
RK_MPI_VO_Enable(VoDev);

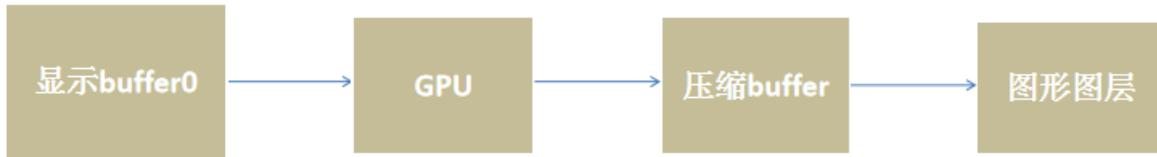
s32Ret = RK_MPI_VO_SetLayerAttr(VoLayer, &stLayerAttr);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;
s32Ret = RK_MPI_VO_EnableLayer(VoLayer);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;

```

【备注】

- 示例申请到单buffer后，完成数据填充，enable vo后，UI图像送往UI图层。
- 方式3采用单buffer方式，通过GPU后，由内部合成buffer后，直接送往图层显示（无需送帧函数）。如下图所示。

RK_MPI_VO_GetGfxFrameBuffer



【方式4举例】

```

MPI_CTX_S *ctx = (MPI_CTX_S *) (pArgs);
RK_S32 s32Ret = 0;

VIDEO_FRAME_INFO_S *pstVFrame;
RK_VOID *pMblk_G0, *pMblk_G1;
RK_U32 count;
ctx->stVoCfg.u32UIVoLayer = RK356X_VOP_LAYER_ESMART_0;
/* CreateGraphicsFrameBuffer and fill pMblk_G0/pMblk_G1 */
Sample_VO_CreateGFXData(1920, 1080, RK_FMT_BGRA5551, 0x1F, &pMblk_G0);
Sample_VO_CreateGFXData(1920, 1080, RK_FMT_BGRA5551, 0x1F << 10, &pMblk_G1);

pstVFrame = (VIDEO_FRAME_INFO_S *) (malloc(sizeof(VIDEO_FRAME_INFO_S)));

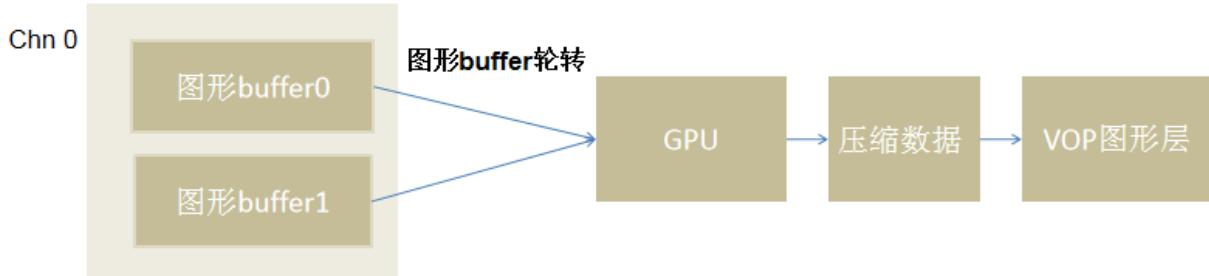
while (1) {
    for (RK_U32 i = 0; i < ctx->stVoCfg.u32UIChn; i++) {
        if (i == 0)
            pMblk = pMblk_G0;
        else if (i == 1)
            pMblk = pMblk_G1;
        else
            continue;
        pstVFrame->stVFrame.pMbBlk = pMblk;
        RK_MPI_VO_SendFrame(ctx->stVoCfg.u32UIVoLayer, i, pstVFrame, 0);
        usleep(33000llu);
    }
}

```

【备注】

- 该实例中UI的绘制使用独立的UI图层，申请了两个显示buffer，调用RK_MPI_VO_SendFrame函数，对图形buffer进行轮转，分别送图形层显示，如下图所示。
- 绘制UI时候，如果据格式为BGRA5551/RGBA5551时，需要填写u32FgAlpha和u32BgAlpha。

CreateGraphicsFrameBuffer



8.4 【方式5举例】

```
VO_PUB_ATTR_S VoPubAttr;
VO_VIDEO_LAYER_ATTR_S stLayerAttr;
RK_U32 s32Ret;
RK_U32 u32DispWidth = 1920;
RK_U32 u32DispHeight = 1080;
RK_U32 u32ImageWidth = 1920;
RK_U32 u32ImageHeight = 1080;

memset(&VoPubAttr, 0, sizeof(VO_PUB_ATTR_S));
memset(&stLayerAttr, 0, sizeof(VO_VIDEO_LAYER_ATTR_S));

stLayerAttr.enPixFormat      = RK_FMT_ARGB8888;
stLayerAttr.stDispRect.s32X   = 0;
stLayerAttr.stDispRect.s32Y   = 0;
stLayerAttr.stDispRect.u32Width = u32DispWidth;
stLayerAttr.stDispRect.u32Height = u32DispHeight;
stLayerAttr.stImageSize.u32Width = u32ImageWidth;
stLayerAttr.stImageSize.u32Height = u32ImageHeight;

VO_LAYER VoLayer = RK356X_VOP_LAYER_CLUSTER0;
VO_DEV VoDev = RK356X_VO_DEV_HD0;
VoPubAttr.enIntfType = VO_INTF_HDMI;
VoPubAttr.enIntfSync = VO_OUTPUT_1080P60;

VO_FRAME_INFO_S stVFrame;
RK_MPI_VO_SetGfxMode(VO_MODE_GFX_PRE_CREATED);
stVFrame.enPixelFormat = RK_FMT_BGRA5551;
stVFrame.u32FgAlpha = 128;
stVFrame.u32BgAlpha = 0;
stVFrame.u32Width = 1920;
stVFrame.u32Height = 1080;
/* Get single GFX buffer */
RK_MPI_VO_GetGfxFrameBuffer(VoLayer, 126, &stVFrame);

/* Set Chn 126 to 4K */
VO_CHN_ATTR_S stChnAttr;
RK_MPI_VO_GetChnAttr(VoLayer, 126, &stChnAttr);
stChnAttr.stRect.u32Width = 3840;
stChnAttr.stRect.u32Height = 2160;
stChnAttr.bEnKeyColor = RK_TRUE; // Enable Key color fuction
stChnAttr.u32KeyColor = 0x8001; // Set Key color value
/* Zoom to 4K */
RK_MPI_VO_SetChnAttr(VoLayer, 126, &stChnAttr);
```

```

memset(stVFrame.pData, 0, stVFrame.u32Size);

s32Ret = RK_MPI_VO_SetPubAttr(VoDev, &VoPubAttr);
if (s32Ret != RK_SUCCESS) {
    return RK_FAILURE;
}
s32Ret = RK_MPI_VO_Enable(VoDev);
if (s32Ret != RK_SUCCESS) {
    return RK_FAILURE;
}
s32Ret = RK_MPI_VO_SetLayerAttr(VoLayer, &stLayerAttr);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;
s32Ret = RK_MPI_VO_EnableLayer(VoLayer);
if (s32Ret != RK_SUCCESS)
    return RK_FAILURE;

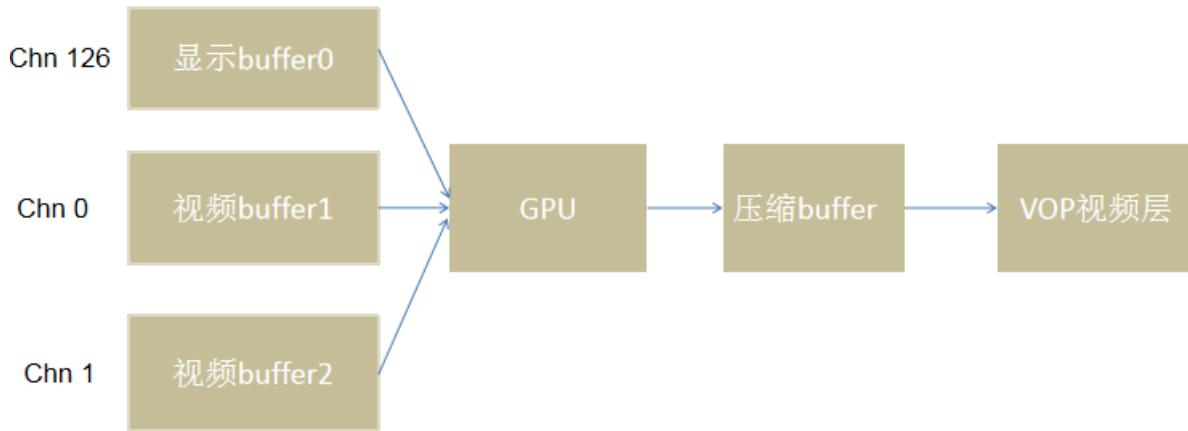
```

【相关主题】

[RK_MPI_VO_GetGfxFrameBuffer](#)

【备注】

- 方式5是在视频层进行UI绘制，chanel 126通道申请了一个单显示buffer，将图像数据和视频数据合成显示在视频图层，没有单独的UI图形层，示意图如下所示，将合成后的buffer送给VOP视频层。



【注意事项】

- 使用预创建模式申请UI显示buffer，建议使用高位的通道号，比如VO_MAX_CHN_NUM-1, VO_MAX_CHN_NUM-2。
- UI图形的分辨率和显示分辨率是1: 1，比如在OSD使场景下，OSD源数据是1080P,最终显示也是1080P,将VO_FRAME_INFO_S结构体宽高和设置为1080P即可。
- UI图形的分辨率和显示分辨率不是1: 1，比如在OSD场景下，OSD源数据是1080P,但是需要4K显示输出，则需要调用getGfxFrameBuffer接口后，再调用RK_MPI_VO_SetChnAttr设置输出的宽高为4K，设置后VOP会将OSD放大到4K。

8.5 【方式6举例】

```

MPI_CTX_S *ctx = (MPI_CTX_S *) (pArgs);
RK_S32 s32Ret = 0;

VIDEO_FRAME_INFO_S *pstVFrame;
RK_VOID *pMblk_G0, *pMblk_G1;
RK_U32 count;
ctx->stVoCfg.u32VideoVoLayer = RK356X_VOP_LAYER_CLUSTER_0;
/* CreateGraphicsFrameBuffer and fill pMblk_G0/pMblk_G1 */
Sample_VO_CreateGFXData(1920, 1080, RK_FMT_BGRA5551, 0x1F, &pMblk_G0);
Sample_VO_CreateGFXData(1920, 1080, RK_FMT_BGRA5551, 0x1F << 10, &pMblk_G1);

pstVFrame = (VIDEO_FRAME_INFO_S *) (malloc(sizeof(VIDEO_FRAME_INFO_S)));

```

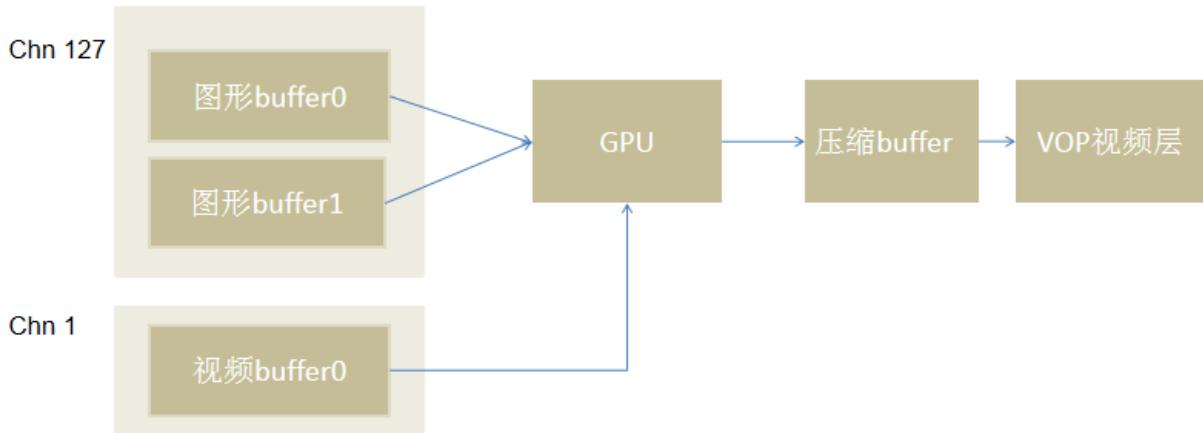
```

while (1) {
    for (RK_U32 i = 0; i < ctx->stVoCfg.u32UIChn; i++) {
        if (i == 0)
            pMblk = pMblk_G0;
        else if (i == 1)
            pMblk = pMblk_G1;
        else
            continue;
        pstVFrame->stVFrame.pMbBlk = pMblk;
        RK_MPI_VO_SendFrame(ctx->stVoCfg.u32VideoVoLayer, i, pstVFrame, 0);
        usleep(33000llu);
    }
}

```

【备注】

- 示例将申请的两个显示buffer，填充后通过RK_MPI_VO_SendFrame接口送往VOP视频层显示。



【注意事项】

- 方式5和方式6主要区别在于内存的使用，单buffer更省内存。RK3568平台，对于有UI+视频的使用场景中，推荐使用方式5，能很好降低VOP的使用带宽。

9. 虚拟图层

虚拟图层用于不需要屏幕显示的图像拼接。

- VO支持4个虚拟图层，定义见 [VO_VIR_LAYER_NAME_E](#)。
- 使能虚拟图层需要将图层绑定到VoDev。
- 可以将多个虚拟图层绑定到同一个VoDev。
- 可以通过RK_MPI_VO_GetLayerFrame 获取虚拟图层拼接的图像帧，获取后需调用RK_MPI_VO_ReleaseLayerFrame释放图像帧。
- 回写的数据源可选择虚拟图层，此时VO_WBC_SOURCE_S成员需配置成虚拟图层ID。

9.1 参考代码

```

stVoPubAttr.enIntfType = VO_INTF_HDMI;
stVoPubAttr.enIntfSync = VO_OUTPUT_1080P50;
s32Ret = Sample_VO_StartDev(VoDev, &stVoPubAttr);
if (s32Ret) {
    Sample_Print("Sample_VO_StartDev s32Ret %x\n", s32Ret);
    return s32Ret;
}
VoLayer = VO_LAYER_VIRTUAL0;
s32Ret = RK_MPI_VO_BindLayer(VoLayer, VoDev, VO_LAYER_MODE_VIRTUAL);
if (s32Ret) {

```

```

Sample_Print("RK_MPI_VO_BindLayer s32Ret %x\n", s32Ret);
    return s32Ret;
}

/* Enable Layer */
stLayerAttr.enPixFormat = RK_FMT_BGR888;
stLayerAttr.stDispRect.s32X = 0;
stLayerAttr.stDispRect.s32Y = 0;
stLayerAttr.stDispRect.u32Width = u32DispWidth;
stLayerAttr.stDispRect.u32Height = u32DispHeight;
stLayerAttr.stlImageSize.u32Width = u32DispWidth;
stLayerAttr.stlImageSize.u32Height = u32DispHeight;
stLayerAttr.u32DispFrmRt = 25;
Sample_VO_StartLayer(VoLayer, &stLayerAttr);

/* Get/Release Virtual Layer Frame */
s32Ret = RK_MPI_VO_GetLayerFrame(VoLayer, &stVideoFrame, 100);
if (s32Ret == 0) {
    RK_MPI_VO_ReleaseLayerFrame(VoLayer, &stVideoFrame);
} else {
    Sample_Print("GetLayer Frame Faild %x\n", s32Ret);
}

/* Enable Wbc */
stWbcSource.enSourceType = VO_WBC_SOURCE_VIRTUAL;
stWbcSource.u32SourceId = VO_LAYER_VIRTUAL0;
RK_MPI_VO_SetWbcSource(VoWbc, &stWbcSource);

stWbcAttr.enPixelFormat = RK_FMT_RGB888;
stWbcAttr.stTargetSize.u32Width = 1280;
stWbcAttr.stTargetSize.u32Height = 720;
stWbcAttr.u32FrameRate = u32Fps;
stWbcAttr.enCompressMode = COMPRESS_MODE_NONE;
RK_MPI_VO_SetWbcAttr(VoWbc, &stWbcAttr);

RK_MPI_VO_SetWbcDepth(VoWbc, 8);

s32Ret = RK_MPI_VO_EnableWbc(VoWbc);
if (s32Ret) {
    Sample_Print("RK_MPI_VO_EnableWbc s32Ret %x\n", s32Ret);
    return s32Ret;
}

```

视频图形子系统

[基本概念](#)
[功能描述](#)
[硬件规格](#)
[API 参考](#)
[数据类型](#)
[VGS 错误码](#)

1. 基本概念

1.1 task

对一幅图像完成具体的一个或多个操作，比如打OSD、缩放或旋转等。VGS默认的最大的task数为200。

1.2 job

VGS管理task的结构，一个job里可以包含多个task，VGS保证task按照添加到job的顺序一次性提交硬件执行。
VGS默认的最大的job数为100。

1.3 HANDLE

任务句柄，标识一个job。

2. 功能描述

VGS的功能有缩放、抠图、旋转、打osd、打cover、画线、打mosaic。

2.1 缩放

VGS 支持对一幅图像进行缩放，最大支持图像宽高放大和缩小32倍。支持单分量Y进行缩放。

2.2 旋转

VGS 支持对一幅图像进行0、90、180、270角度的旋转。

2.3 打OSD

VGS支持在一幅图像上面叠加一张位图、支持批量打osd。

打osd操作不支持对输入的位图进行反色操作。

打osd时输入和输出图像使用的是同一块内存。

2.4 打COVER

VGS支持对一幅图像进行遮挡操作，遮挡的形状为矩形或者任意的四边形。

当cover为矩形的时候，cover只能是实心，cover为任意四边形的时候，cover可以是实心或者空心的。

打cover操作时，输入和输出图像使用的是同一块内存。

2.5 画线

VGS支持对一幅图像进行画线操作， 支持批量画线操作。

画线操作时，输入和输出图像使用的是同一块内存。

2.6 打mosaic

VGS支持对一幅图像进行打mosaic操作， 支持不同的blksize 输入， 支持批量进行打mosaic 操作。

打mosaic操作时，输入和输出图像使用的是同一块内存。

3. 硬件规格

3.1 RK356X VGS 硬件规格

数据格式	宽度对齐像素	高度对齐像素	像素比特数
RK_FMT_YUV420SP RK_FMT_YUV420SP_VU	16	1	8
RK_FMT_YUV420P	64	1	8
RK_FMT_YUV422SP RK_FMT_YUV422_YUYV	32	1	16
RK_FMT_YUV400SP	64	1	8
RK_FMT_YUV420SP_10BIT	32	1	10
RK_FMT_RGB565 RK_FMT_BGR565	32	1	16
RK_FMT_RGBA5551 RK_FMT_BGRA5551	32	1	16
RK_FMT_RGB888 RK_FMT_BGR888	64	1	24
RK_FMT_BGRA8888 RK_FMT_RGBA8888	16	1	32

【注意】

- VGS支持图像大小在64x64到8192x8192范围内的缩放。

4. API 参考

该功能模块为用户提供以下 MPI:

- [RK_MPI_VGS_BeginJob](#): 启动一个 job。
- [RK_MPI_VGS_AddScaleTask](#): 往一个已经启动的 job 添加缩放 task。
- [RK_MPI_VGS_AddCropTask](#): 往一个已经启动的 job 添加裁剪 task。
- [RK_MPI_VGS_AddDrawLineTask](#): 往一个已经启动的 job 添加画线 task。
- [RK_MPI_VGS_AddCoverTask](#): 往一个已经启动的 job 添加打 COVER task。
- [RK_MPI_VGS_AddOsdTask](#): 往一个已经启动的 job 添加打 OSD task。
- [RK_MPI_VGS_AddRotationTask](#): 往一个已经启动的 job 里添加旋转任务。

- [RK_MPI_VGS_AddMosaicTask](#): 往一个已经启动的 job 里添加打马赛克任务。
- [RK_MPI_VGS_AddDrawLineTaskArray](#): 往一个已经启动的 job 里添加批量画线的任务。
- [RK_MPI_VGS_AddCoverTaskArray](#): 往一个已经启动的 job 里添加批量打COVER的任务。
- [RK_MPI_VGS_AddOsdTaskArray](#): 往一个已经启动的 job 里添加批量打OSD的任务。
- [RK_MPI_VGS_AddMosaicTaskArray](#): 往一个已经启动的 job 里添加批量打马赛克的任务。

4.1 RK_MPI_VGS_BeginJob

【描述】

启动一个 job。

【语法】

```
RK_S32 RK_MPI_VGS_BeginJob(VGS\_HANDLE*phHandle);
```

【参数】

参数名称	描述	输入\输出
phHandle	返回的job handle	输出

【返回值】

返回值	描述
0	成功
非0	失败，见VGS错误码

【注意】

- 可一次启动多个 job，但必须判断 [RK_MPI_VGS_BeginJob](#) 函数返回成功后才能使用phHandle 返回的 HANLDE。
- phHandle 不能为空指针或非法指针。

【举例】

```
RK_S32 s32Ret = RK_SUCCESS;
VGS_HANDLE hHandle;
VGS_TASK_ATTR_S stTask;
s32Ret = RK_MPI_VGS_BeginJob(&hHandle);
if (s32Ret != RK_SUCCESS)
{
    VGS_ERROR_PROCESS(s32Ret);
}
s32Ret = RK_MPI_VGS_AddScaleTask(hHandle, &stTask, VGS_SCLCOEF_NORMAL);
if (s32Ret != RK_SUCCESS)
{
    RK_MPI_VGS_CancelJob(hHandle);
    VGS_ERROR_PROCESS(s32Ret);
}
s32Ret = RK_MPI_VGS_EndJob(hHandle);
if (s32Ret != RK_SUCCESS)
{
    RK_MPI_VGS_CancelJob(hHandle);
    VGS_ERROR_PROCESS(s32Ret);
}
```

4.2 RK_MPI_VGS_EndJob

【描述】

提交一个 job。

【语法】

```
RK_S32 RK_MPI_VGS_EndJob(VGS HANDLE hHandle);
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE。	输入

【返回值】

返回值	描述
0	成功
非0	失败, 见 VGS错误码

【注意】

- 如果此接口返回失败, 必须调用[RK_MPI_VGS_CancelJob](#)接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job。

4.3 RK_MPI_VGS_CancelJob

【描述】

取消一个 job。

【语法】

```
RK_S32 RK_MPI_VGS_CancelJob(VGS HANDLE hHandle);
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE。	输入

【返回值】

返回值	描述
0	成功
非0	失败, 见 VGS错误码

【注意】

- hHandle 标识的 job 必须是已经启动的 job。

4.4 RK_MPI_VGS_AddScaleTask

【描述】

往一个已经启动的job里添加缩放task。

【语法】

```
RK_S32 RK_MPI_VGS_AddScaleTask(VGS HANDLE hHandle, const VGS\_TASK\_ATTR\_S *pstTask,  
VGS\_SCLCOEF\_MODE\_E enScaleCoefMode);
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE。	输入
pstTask	VGS task 属性指针	输入
enScaleCoefMode	缩放系数模式，暂不支持	输入

【返回值】

返回值	描述
0	成功
非0	失败，见 VGS错误码

【注意】

- 如果此接口返回失败，必须调用[RK_MPI_VGS_CancelJob](#)接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job。
- 输入图像的宽高，对齐方式 参考硬件规格中的图像格式对齐说明
- 缩放时输入或者输出的宽高不符合VGS模块要求的对齐时，缩放的job都会失败。
- 缩放任务时， 输入和输出的可以是不同的两块内存。

【举例】

- 参考[RK_MPI_VGS_BeginJob](#)的举例

4.5 RK_MPI_VGS_AddCropTask

【描述】

往一个已经启动的job里添加裁剪task。

【语法】

```
RK_S32 RK_MPI_VGS_AddCropTask(VGS\_HANDLE hHandle, const VGS\_TASK\_ATTR\_S *pstTask, const VGS\_CROP\_INFO\_S*pstVgsCrop)
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE。	输入
pstTask	VGS task 属性指针	输入
pstVgsCrop	需要裁剪的区域	输入

【返回值】

返回值	描述
0	成功
非0	失败，见VGS错误码

【注意】

- 如果此接口返回失败，必须调用[RK_MPI_VGS_CancelJob](#)接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job。
- 输入图像的宽高，对齐方式 参考硬件规格中的图像格式对齐说明。

【举例】

- 参考[RK_MPI_VGS_BeginJob](#)的举例

4.6 RK_MPI_VGS_AddDrawLineTask

【描述】

往一个已经启动的job里添加画线task。

【语法】

```
RK_S32 RK_MPI_VGS_AddDrawLineTask(VGS\_HANDLE hHandle, const VGS\_TASK\_ATTR\_S *pstTask, const VGS\_DRAW\_LINE\_S *pstVgsDrawLine);
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE。	输入
pstTask	VGS task 属性指针	输入
pstVgsDrawLine	VGS 画线属性配置指针	输入

【返回值】

返回值	描述
0	成功
非0	失败, 见 VGS错误码

【注意】

- 如果此接口返回失败，必须调用[RK_MPI_VGS_CancelJob](#)接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job。
- 做画线任务时，输入和输出使用的是同一块buffer。

【举例】

- 参考[RK_MPI_VGS_BeginJob](#)的举例

4.7 RK_MPI_VGS_AddCoverTask

【描述】

往一个已经启动的job里添加打COVER task。

【语法】

```
RK_S32 RK_MPI_VGS_AddCoverTask(VGS\_HANDLE hHandle, const VGS\_TASK\_ATTR\_S *pstTask, const VGS\_ADD\_COVER\_S *pstVgsAddCover)
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE。	输入
pstTask	VGS task 属性指针	输入
pstVgsAddCover	VGS 打 COVER 属性配置指针	输入

【返回值】

返回值	描述
0	成功
非0	失败, 见 VGS错误码

【注意】

- 如果此接口返回失败, 必须调用 [RK_MPI_VGS_CancelJob](#)接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job。
- 做COVER任务时, 输入和输出使用的是同一块buffer。

【举例】

- 参考[RK_MPI_VGS_BeginJob](#)的举例

4.8 RK_MPI_VGS_AddOsdTask

【描述】

往一个已经启动的job里添加打OSD task。

【语法】

```
RK_S32 RK_MPI_VGS_AddOsdTask(VGS\_HANDLE hHandle, const VGS\_TASK\_ATTR\_S *pstTask, const  
VGS\_ADD OSD\_S *pstVgsAddOsd)
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE。	输入
pstTask	VGS task 属性指针	输入
pstVgsAddOsd	VGS 打 OSD 属性配置指针	输入

【返回值】

返回值	描述
0	成功
非0	失败, 见 VGS错误码

【注意】

- 如果此接口返回失败, 必须调用 [RK_MPI_VGS_CancelJob](#)接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job。
- 做OSD任务时, 输入和输出使用的是同一块buffer。

【举例】

- 参考[RK_MPI_VGS_BeginJob](#)的举例

4.9 RK_MPI_VGS_AddRotationTask

【描述】

往一个已经启动的job里添加旋转 task。

【语法】

```
RK_S32 RK_MPI_VGS_AddRotationTask(VGS\_HANDLE hHandle, const VGS\_TASK\_ATTR\_S *pstTask, ROTATION\_E enRotationAngle)
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE。	输入
pstTask	VGS task 属性指针	输入
enRotationAngle	旋转角度	输入

【返回值】

返回值	描述
0	成功
非0	失败, 见 VGS错误码

【注意】

- 如果此接口返回失败，必须调用[RK_MPI_VGS_CancelJob](#)接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job。

【举例】

- 参考[RK_MPI_VGS_BeginJob](#)的举例

4.10 RK_MPI_VGS_AddMosaicTask

【描述】

往一个已经启动的job里添加打马赛克 task。

【语法】

```
RK_S32 RK_MPI_VGS_AddMosaicTask(VGS\_HANDLE hHandle, const VGS\_TASK\_ATTR\_S *pstTask, const VGS\_MOSAIC\_S *pstVgsMosaic)
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE。	输入
pstTask	VGS task 属性指针	输入
pstVgsMosaic	Mosaic属性配置结构体	输入

【返回值】

返回值	描述
0	成功
非0	失败, 见 VGS错误码

【注意】

- 如果此接口返回失败，必须调用[RK_MPI_VGS_CancelJob](#)接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job。
- 做mosaic任务的时候，输入和输出的图像为同一块buffer。

【举例】

- 参考[RK_MPI_VGS_BeginJob](#)的举例

4.11 RK_MPI_VGS_AddDrawLineTaskArray

【描述】

往一个已经启动的job里添加批量画线task。

【语法】

```
RK_S32 RK_MPI_VGS_AddDrawLineTaskArray(VGS\_HANDLE hHandle, const VGS\_TASK\_ATTR\_S *pstTask, const VGS\_DRAW\_LINE\_S astVgsDrawLine[], RK_U32 u32ArraySize);
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE。	输入
pstTask	VGS task 属性指针	输入
astVgsDrawLine	VGS 画线属性配置结构体数组	输入
u32ArraySize	VGS 画线数目，范围[1,100]	输入

【返回值】

返回值	描述
0	成功
非0	失败，见 VGS错误码

【注意】

- 如果此接口返回失败，必须调用[RK_MPI_VGS_CancelJob](#)接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job。
- 做批量画线任务时，输入和输出的图像为同一块buffer。

【举例】

- 参考[RK_MPI_VGS_BeginJob](#)的举例

4.12 RK_MPI_VGS_AddCoverTaskArray

【描述】

往一个已经启动的job里添加批量打COVER task。

【语法】

```
RK_S32 RK_MPI_VGS_AddCoverTaskArray(VGS\_HANDLE hHandle, const VGS\_TASK\_ATTR\_S *pstTask, const VGS\_ADD\_COVER\_S astVgsAddCover[], RK_U32 u32ArraySize)
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE。	输入
pstTask	VGS task 属性指针	输入
astVgsAddCover	VGS 打 COVER 属性配置结构体数组	输入
u32ArraySize	VGS Cover数目，范围[1,100]	输入

【返回值】

返回值	描述
0	成功
非0	失败，见 VGS错误码

【注意】

- 如果此接口返回失败，必须调用[RK_MPI_VGS_CancelJob](#)接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job。
- 做批量COVER任务时，输入和输出的图像为同一块buffer。

【举例】

- 参考[RK_MPI_VGS_BeginJob](#)的举例

4.13 RK_MPI_VGS_AddOsdTaskArray

【描述】

往一个已经启动的job里添加批量打OSD task。

【语法】

```
RK_S32 RK_MPI_VGS_AddOsdTaskArray(VGS\_HANDLE hHandle, const VGS\_TASK\_ATTR\_S *pstTask, const VGS\_ADD OSD S astVgsAddOsd[], RK_U32 u32ArraySize)
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE。	输入
pstTask	VGS task 属性指针	输入
astVgsAddOsd	VGS 打 OSD 属性配置结构体数组	输入
u32ArraySize	VGS OSD数目，范围[1,100]	输入

【返回值】

返回值	描述
0	成功
非0	失败，见 VGS错误码

【注意】

- 如果此接口返回失败，必须调用[RK_MPI_VGS_CancelJob](#)接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job。
- 做批量叠加OSD任务时，输入和输出的图像为同一块buffer。

【举例】

- 参考[RK_MPI_VGS_BeginJob](#)的举例

4.14 RK_MPI_VGS_AddMosaicTaskArray

【描述】

往一个已经启动的job里添加批量打mosaic task。

【语法】

```
RK_S32 RK_MPI_VGS_AddMosaicTaskArray(VGS_HANDLE hHandle, const VGS_TASK_ATTR_S *pstTask, const  
VGS_MOSAIC_S astVgsMosaic[], RK_U32 u32ArraySize)
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE。	输入
pstTask	VGS task 属性指针	输入
astVgsMosaic	VGS 打 mosaic 属性配置结构体数组	输入
u32ArraySize	mosaic 数目, 范围[1,100]	输入

【返回值】

返回值	描述
0	成功
非0	失败, 见 VGS错误码

【注意】

- 如果此接口返回失败，必须调用[RK_MPI_VGS_CancelJob](#)接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job。
- 做批量mosaic任务时，输入和输出的图像为同一块buffer。

【举例】

- 参考[RK_MPI_VGS_BeginJob](#)的举例

5. 数据类型

VGS 模块相关数据类型定义如下：

- VGS_HANDLE**：定义 VGS job 的句柄。
- VGS_TASK_ATTR_S**: 定义 VGS task 的属性。
- VGS_CROP_COORDINATE_E**: 定义 VGS 裁剪起始坐标的模式。
- VGS_CROP_INFO_S**: 定义 VGS 裁剪所需要的相关配置。
- VGS_DRAW_LINE_S**: 定义 VGS 画线操作的相关配置。
- VGS_COVER_TYPE_E**: 定义 VGS 上的 COVER 类型。
- VGS_ADD_COVER_S**: 定义 VGS 上 COVER 的配置。
- VGS_MOSAIC_S**: 定义 VGS 上 MOSAIC 的配置
- VGS_COLOR_REVERT_MODE_E**: 定义 VGS 上 OSD 的反色模式
- VGS OSD REVERT S**: 定义 VGS 上 OSD 反色的配置。
- VGS_ADD OSD S**: 定义 VGS 上 OSD 的配置。
- VGS_SCLCOEF_MODE_E**: 定义 VGS 缩放系数模式的配置。
- VGS_MOSAIC_BLK_SIZE_E**: 定义 VGS mosaic 块大小的配置。

5.1 VGS_HANDLE

【说明】

定义 VGS job 的句柄。

【定义】

```
typedef RK_S32 VGS_HANDLE
```

【注意事项】

无。

【相关数据类型及接口】

无。

5.2 VGS_TASK_ATTR_S

【说明】

定义 VGS task 的属性。

【定义】

```
typedef struct rkVGS_TASK_ATTR_S
{
    VIDEO_FRAME_INFO_S stImgIn;
    VIDEO_FRAME_INFO_S stImgOut;
    RK_U64 au64privateData[4];
    RK_U32 reserved;
}VGS_TASK_ATTR_S;
```

【成员】

参数名称	描述
stImgIn	输入图像属性
stImgOut	输出图像属性
au64privateData	与 task 相关的私有数据，VGS 不会使用和修改其中的数据。
reserved	保留项

5.3 VGS_CROP_COORDINATE_E

【说明】

定义 VGS 裁剪起始坐标的模式。

【定义】

```
typedef enum rkVGS_CROP_COORDINATE_E {
    VGS_CROP_RATIO_COOR = 0, /* Ratio coordinate. */
    VGS_CROP_ABS_COOR     /* Absolute coordinate. */
} VGS_CROP_COORDINATE_E;
```

【成员】

参数名称	描述
VGS_CROP_RATIO_COOR	相对坐标
VGS_CROP_ABS_COOR	绝对坐标

【注意事项】

相对坐标，即起始点的坐标值是以与当前图像宽高的比率来表示，使用时需做转换，具体请参见 VGS_CROP_INFO_S。

5.4 VGS_CROP_INFO_S

【说明】

定义 VGS 裁剪所需要的相关配置。

【定义】

```
typedef struct rkVGS_CROP_INFO_S {
    VGS_CROP_COORDINATE_E enCropCoordinate;
    RECT_S stCropRect;
} VGS_CROP_INFO_S;
```

【成员】

参数名称	描述
enCropCoordinate	CROP 起始点坐标模式
stCropRect	CROP 的矩形区域

【注意事项】

- 若 enCropCoordinate 为 VPSS_CROP_RATIO_COOR（相对坐标模式），使用 stCropRect 的成员时应做转换，计算公式为：
 $s32X = \text{起始点坐标 } x \times \text{原始图像宽度}/1000$ ，合法取值范围：[0, 999]，计算完成后会进行取整操作和对齐操作。公式同样适用于纵坐标计算。
 $u32Width = \text{区域宽度} \times \text{实际图像宽度}/1000$ ，区域宽度取值范围：[1, 1000]。计算完成后会进行取整操作和对齐操作。公式同样适用于区域高度计算。

5.5 VGS_DRAW_LINE_S

【说明】

定义 VGS 画线操作的相关配置。

【定义】

```
typedef struct rkVGS_DRAW_LINE_S {
    POINT_S stStartPoint;
    POINT_S stEndPoint;
    RK_U32 u32Thick;
    RK_U32 u32Color;
} VGS_DRAW_LINE_S;
```

【成员】

参数名称	描述
stStartPoint	线的起始点坐标。
stEndPoint	线的结束点坐标。
u32Thick	线的宽度。
u32Color	线的颜色，RGBA8888 格式，取值范围[0x0, 0xFFFFFFFF]。

5.6 VGS_COVER_TYPE_E

【说明】

定义 VGS 上的 COVER 类型。

【定义】

```
typedef enum rkVGS_COVER_TYPE_E {
    COVER_RECT = 0,
    COVER_QUAD_RANGLE,
    COVER_BUTT
} VGS_COVER_TYPE_E;
```

【成员】

参数名称	描述
COVER_RECT	矩形 COVER。
COVER_QUAD_RANGLE	任意四边形 COVER。

5.7 VGS_ADD_COVER_S

【说明】

定义 VGS 上 COVER 的配置。

【定义】

```
typedef struct rkVGS_ADD_COVER_S {
    VGS_COVER_TYPE_E          enCoverType;
    union {
        RECT_S                stDstRect;
        VGS_QUADRANGLE_COVER_S stQuadRangle;
    };
    RK_U32                  u32Color;
} VGS_ADD_COVER_S;
```

【成员】

参数名称	描述
enCoverType	COVER 类型。
stDstRect	矩形 COVER 的位置和宽高。
stQuadRangle	任意四边形 COVER 的相关配置。四点坐标值和边框厚度。
u32Color	COVER 的颜色， RGBA8888 格式， 取值范围[0x0, 0xFFFFFFFF]。

5.8 VGS_COLOR_REVERT_MODE_E

【说明】

定义 VGS 上 OSD 反色模式的配置。

【定义】

```

typedef enum rkVGS_COLOR_REVERT_MODE_E {
    VGS_COLOR_REVERT_NONE = 0,
    VGS_COLOR_REVERT_RGB,
    VGS_COLOR_REVERT_ALPHA,
    VGS_COLOR_REVERT_BOTH,
    VGS_COLOR_REVERT_BUTT
} VGS_COLOR_REVERT_MODE_E;

```

【成员】

参数名称	描述
VGS_COLOR_REVERT_NONE	不反色。
VGS_COLOR_REVERT_RGB	仅对 RGB 反色。
VGS_COLOR_REVERT_ALPHA	仅对 alpha 反色。
VGS_COLOR_REVERT_BOTH	对 RGB 和 alpha 反色。

5.9 VGS OSD REVERT_S

【说明】

定义 VGS 上 OSD 反色的配置。

【定义】

```

typedef struct rkVGS OSD REVERT_S {
    RECT_S          stSrcRect;
    VGS_COLOR_REVERT_MODE_E enColorRevertMode;
} VGS OSD REVERT_S;

```

【成员】

参数名称	描述
stSrcRect	OSD 反色的起始坐标及宽高。位置和宽高值均要求 2 对齐。
enColorRevertMode	OSD 反色模式。

5.10 VGS_ADD OSD S

【说明】

定义 VGS 上 OSD 的配置。

【定义】

```

typedef struct rkVGS_ADD OSD_S {
    MB_BLK          pMbBlk;
    RECT_S          stRect;
    PIXEL_FORMAT_E   enPixelFormat;
    RK_U32           u32FgAlpha;
    RK_U32           u32BgAlpha;
} VGS_ADD OSD_S;

```

【成员】

参数名称	描述
stRect	OSD 的起始坐标及宽高。
enPixelFmt	OSD 的像素格式。
pMbBlk	OSD 图像的物理地址。
u32FgAlpha	像素格式为RGBA551或BGRA5551时，OSD 的前景 alpha 值。
u32BgAlpha	像素格式为RGBA551或BGRA5551时，OSD 的背景 alpha 值。

5.11 VGS_MOSAIC_BLK_SIZE_E

【说明】

mosaic 块大小枚举。

【定义】

```
typedef enum rkVGS_MOSAIC_BLK_SIZE_E {
    RK_MOSAIC_BLK_SIZE_8 = 8,
    RK_MOSAIC_BLK_SIZE_16 = 16,
    RK_MOSAIC_BLK_SIZE_32 = 32,
    RK_MOSAIC_BLK_SIZE_64 = 64,
    RK_MOSAIC_BLK_SIZE_BUT
} VGS_MOSAIC_BLK_SIZE_E;
```

【成员】

参数名称	描述
RK_MOSAIC_BLK_SIZE_8	8x8 大小的Mosaic块。
RK_MOSAIC_BLK_SIZE_16	16x16 大小的Mosaic块。
RK_MOSAIC_BLK_SIZE_32	32x32 大小的Mosaic块。
RK_MOSAIC_BLK_SIZE_64	64x64 大小的Mosaic块。

5.12 VGS_MOSAIC_S

【说明】

定义 VGS 上 mosaic 的配置。

【定义】

```
typedef struct rkVGS_MOSAIC_S {
    VGS_MOSAIC_BLK_SIZE_E enBlkSize;
    RECT_S stDstRect;
} VGS_MOSAIC_S;
```

【成员】

参数名称	描述
enBlkSize	mosaic 块大小。
stDstRect	矩形坐标。

6. VGS错误码

VGS API VGS错误码如下

错误代码	宏定义	描述
0xA007800E	RK_ERR_VGS_BUF_EMPTY	VGS的job,task 或node 节点已经使用完毕
0xA0078003	RK_ERR_VGS_ILLEGAL_PARAM	VGS 参数设置无效
0xA0078006	RK_ERR_VGS_NULL_PTR	输入参数空指针错误
0xA0078008	RK_ERR_VGS_NOT_SUPPORT	操作不支持
0xA0078009	RK_ERR_VGS_NOT_PERMITTED	操作不允许
0xA007800D	RK_ERR_VGS_NOBUF	分配内存失败
0xA0078006	RK_ERR_VGS_NULL_PTR	输入参数空指针错误
0xA0078010	RK_ERR_VGS_SYS_NOTREADY	系统未初始化
0xA007800F	RK_ERR_VGS_BUF_FULL	没有剩余 BUF

图形处理

[基本概念](#)

[硬件规格](#)

[API 参考](#)

[数据类型](#)

[TDE错误码](#)

1. 基本概念

TDE (Two Dimensional Engine) 利用硬件RGA提供快速的图形处理功能，主要有快速位图搬移、快速色彩填充、快速位图旋转、快速位图缩放、位图格式转换、位图alpha叠加、ColorKey 操作。

1.1 举例

```
RK_S32 s32Ret = RK_SUCCESS;
FILE *file = RK_NULL;
RK_S32 u32JobTestTime = 1;
RK_S32 u32TaskTestTime = 2;
void *pSrcData = RK_NULL;
RK_U32 u32taskCount = 0;
MB_BLK srcBlk = RK_NULL;
RK_U32 u32ImgSize = COLOR_WIDTH * COLOR_HEIGHT * 3 / 2;
TDE_HANDLE hHandle = 0;
TDE_SURFACE_S pstDst[TDE_MAX_TASK_NUM];
TDE_RECT_S pstDstRect[TDE_MAX_TASK_NUM];

RK_TDE_Open();
hHandle = RK_TDE_BeginJob();
if (RK_ERR_TDE_INVALID_HANDLE == hHandle) {
    RK_LOGE("start job fail");
    return RK_FAILURE;
}
TDE_SURFACE_S pstSrc;
TDE_RECT_S pstSrcRect;
test_tde_quick_resize_task(&pstSrc,
    &pstSrcRect,
    &pstDst[u32TaskIndex],
    &pstDstRect[u32TaskIndex],
    srcBlk, file, u32ImgSize);
s32Ret = RK_TDE_QuickResize(hHandle, &pstSrc, &pstSrcRect,
    &pstDst[u32TaskIndex], &pstDstRect[u32TaskIndex]);
if (s32Ret != RK_SUCCESS) {
    RK_TDE_CancelJob(hHandle);
    return RK_FAILURE;
}
s32Ret = RK_TDE_EndJob(hHandle, RK_FALSE, RK_TRUE, 10);
if (s32Ret != RK_SUCCESS) {
    RK_TDE_CancelJob(hHandle);
    return RK_FAILURE;
}

RK_TDE_WaitForDone(hHandle);
RK_TDE_CancelJob(hHandle);
RK_TDE_Close();
RK_MPI_SYS_Free(srcBlk);
```

2. 硬件规格

2.1 RK356X TDE 硬件规格

数据格式	宽度对齐像素	高度对齐像素	单像素比特数
RK_FMT_YUV420SP RK_FMT_YUV420SP_VU	4	2	8
RK_FMT_YUV420p	4	2	8
RK_FMT_YUV422SP RK_FMT_YUV422_YUYV	4	2	8
RK_FMT_YUV400SP	4	2	8
RK_FMT_YUV420SP_10BIT RK_FMT_YUV422SP_10BIT	16	2	10
RK_FMT_RGB565 RK_FMT_BGR565	2	1	16
RK_FMT_RGBA5551 RK_FMT_BGRA5551	32	1	16
RK_FMT_RGB888 RK_FMT_BGR888	4	1	24
RK_FMT_BGRA8888 RK_FMT_RGBA8888	1	1	32

【注意】

- TDE操作区域的对齐跟格式有关系，如果是rgb格式的宽高和偏移不需要对齐，yuv格式宽高和偏移均需2像素对齐。
- 调用TDE操作时，请确保[RK_TDE_Open\(\)](#)被成功调用，否则硬件无法被使用。
- 除了快速拷贝的操作外，TDE暂不支持输入输出压缩格式，请保证图像输入为非压缩（非AFBC等任意压缩格式）。
- TDE支持图像大小在64x64到8192x8192范围内的缩放。

2.2 RK3588 TDE 硬件规格

数据格式	宽度对齐像素	高度对齐像素	单像素比特数	压缩格式
RK_FMT_YUV420SP RK_FMT_YUV420SP_VU	16	2	8	支持
RK_FMT_YUV420P	16	2	8	不支持
RK_FMT_YUV422SP	16	2	8	支持
RK_FMT_YUV422_YUYV	16	2	8	不支持
RK_FMT_YUV400SP	16	2	8	不支持
RK_FMT_YUV420SP_10BIT RK_FMT_YUV422SP_10BIT	64	2	10	支持
RK_FMT_RGB565	8	1	16	支持
RK_FMT_BGR565	8	1	16	不支持
RK_FMT_RGB888	16	1	24	支持
RK_FMT_BGR888	16	1	24	不支持
RK_FMT_RGBA8888	4	1	32	不支持
RK_FMT_BGRA8888	4	1	32	支持
RK_FMT_RGBA5551 RK_FMT_BGRA5551	32	1	16	支持

【注意】

- TDE操作区域的对齐和rk3568保持一致。
- 压缩格式作为输入和输出时，只支持最大8倍的缩放。
- 压缩格式作为输入和输出时，支持的分辨率范围128x128 到8176x8176。

2.3 RV1106/RV1103 TDE 硬件规格

数据格式	宽度对齐像素	高度对齐像素	单像素比特数	压缩格式
RK_FMT_YUV420SP RK_FMT_YUV420SP_VU RK_FMT_YUV420P	4	2	8	不支持
RK_FMT_YUV422SP RK_FMT_YUV422P	4	2	8	不支持
RK_FMT_YUV400SP	4	2	8	不支持
RK_FMT_YUV422_YUYV	4	2	8	不支持
RK_FMT_RGB888 RK_FMT_BGR888	4	1	24	不支持
RK_FMT_RGB565 RK_FMT_BGR565	2	1	16	不支持
RK_FMT_BGRA8888 RK_FMT_RGBA8888	1	1	32	不支持
RK_FMT_BGRA4444	1	1	16	不支持

【注意】

- 调用TDE操作时，请确保[RK_TDE_Open\(\)](#)被成功调用，否则硬件无法被使用。
- 除了快速拷贝的操作外，TDE暂不支持输入输出压缩格式，请保证图像输入为非压缩（非AFBC等任意压缩格式）。

- 输入和输出，只支持最大16倍的缩放。
- TDE支持图像大小在64x64到8192x8192范围内的缩放。

3. API 参考

该功能模块为用户提供以下API:

- [RK_TDE_Open](#): 打开TDE设备。
- [RK_TDE_Close](#): 关闭TDE设备。
- [RK_TDE_BeginJob](#): 创建1个TDE任务。
- [RK_TDE_EndJob](#): 提交已创建的TDE任务。
- [RK_TDE_CancelJob](#): 取消TDE任务。
- [RK_TDE_WaitForDone](#): 待指定的任务完成。
- [RK_TDE_WaitAllDone](#): 等待TDE的所有任务完成。
- [RK_TDE_QuickCopy](#): 向指定任务中添加快速拷贝操作。
- [RK_TDE_QuickResize](#): 向任务中添加光栅位图缩放操作。
- [RK_TDE_Bitblit](#): 向任务中添加对光栅位图进行有附加功能的搬移操作。
- [RK_TDE_QuickFill](#): 向任务中添加快速填充操作。
- [RK_TDE_Rotate](#): 向任务中添加光栅位图旋转操作。
- [RK_TDE_BitmapMaskBlend](#): 向任务中添加对光栅位图进行Mask Blend搬移操作。根据Mask位图实现前景位图和背景位图带Mask位图的叠加效果。

3.1 RK_TDE_Open

【描述】

打开TDE设备。

【语法】

```
RK_S32 RK_TDE_Open(RK_VOID);
```

【参数】

无

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 TDE错误码 。

【注意】

- 使用TDE设备前需要调用Open接口完成TDE设备初始化。
- 不支持重复调用，重复调用返回失败。

3.2 RK_TDE_Close

【描述】

关闭TDE设备。

【语法】

```
RK_VOID RK_TDE_Close(RK_VOID);
```

【参数】

无

【返回值】

无

【注意】

- RK_TDE_Open需要被成功调用。

3.3 RK_TDE_BeginJob

【描述】

创建 1 个 TDE 任务。

【语法】

`TDE_HANDLE RK_TDE_BeginJob(RK_VOID);`

【参数】

无

【返回值】

返回值	描述
句柄	成功。
非0	失败，其值为TDE错误码。

【注意】

- [RK_TDE_Open](#)需要被成功调用。
- 应该判断返回值，确保获得1个正确的任务句柄。
- TDE最多支持128个任务。

3.4 RK_TDE_EndJob

【描述】

提交已创建的 TDE 任务。可以指定为阻塞还是非阻塞，阻塞的可以设置超时时间。

• 阻塞

函数调用不会立即返回，只有在TDE Job 中的任务都执行完成或者超时时间到达的情况下才会返回。

• 非阻塞

函数调用会立即返回，不关心TDE中的job是不是执行完成。

【语法】

`RK_S32 RK_TDE_EndJob(TDE_HANDLE s32Handle, RK_BOOL bSync, RK_BOOL bBlock, RK_U32 u32TimeOut);`

【参数】

参数名	描述	输入/输出
s32Handle	TDE任务句柄。	输入
bSync	暂时不使用。	输入
bBlock	阻塞标志。 RK_TRUE：阻塞。 RK_FALSE：非阻塞。	输入
u32TimeOut	超时时间，单位：ms。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为TDE错误码。

【注意】

- [RK_TDE_Open](#)需要被成功调用。
- 在调用此接口前应保证调用[RK_TDE_BeginJob](#)获得了有效的任务句柄。
- 此接口若为阻塞接口，到达超时时间调用函数会返回，但是操作然会继续完成。

3.5 RK_TDE_CancelJob

【描述】

取消 TDE 任务。

【语法】

```
RK_S32 RK_TDE_CancelJob(TDE\_HANDLE s32Handle);
```

【参数】

参数名	描述	输入/输出
s32Handle	TDE任务句柄。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 TDE错误码 。

【注意】

- [RK_TDE_Open](#)需要被成功调用。
- 保证调用[RK_TDE_BeginJob](#)获得了有效的任务句柄，否则返回值无效。
- 已经提交的任务不能够再取消。
- 取消后的任务不再有效，故不能再向其添加操作，也不能提交该任务。

3.6 RK_TDE_WaitForDone

【描述】

等待指定的任务完成。

【语法】

```
RK_S32 RK_TDE_WaitForDone(TDE\_HANDLE s32Handle);
```

【参数】

参数名	描述	输入/输出
s32Handle	TDE任务句柄。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 TDE错误码 。

【注意】

- [RK_TDE_Open](#)需要被成功调用。
- [RK_TDE_EndJob](#)接口采用非阻塞方式或者有超时时间的，都需要调用这个接口等待任务完成。此接口为阻塞接口，会阻塞等待指定的任务完成。

3.7 RK_TDE_WaitAllDone

【描述】

等待 TDE 的所有任务完成。

【语法】

```
RK_S32 RK_TDE_WaitAllDone(RK\_VOID);
```

【参数】

无

【返回值】

返回值	描述
0	成功。
非0	失败，其值为TDE错误码。

【注意】

- [RK_TDE_Open](#)需要被成功调用。
- 此接口为阻塞接口，会阻塞等待所有的 TDE 任务完成。

3.8 RK_TDE_QuickCopy

【描述】

向指定任务中添加快速拷贝操作。

【语法】

```
RK_S32 RK_TDE_QuickCopy(TDE\_HANDLE s32Handle,  
                      const TDE\_SURFACE\_S *pstSrc,  
                      const TDE\_RECT\_S *pstSrcRect,  
                      const TDE\_SURFACE\_S *pstDst,  
                      const TDE\_RECT\_S *pstDstRect);
```

【参数】

参数名	描述	输入/输出
s32Handle	TDE任务句柄。	输入
pstSrc	源位图。	输入
pstSrcRect	源位图操作区域。	输入
pstDst	目标位图。	输入
pstDstRect	目标位图操作区域。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为TDE错误码。

【注意】

- [RK_TDE_Open](#)需要被成功调用。
- 将基地址为 pstSrc 的位图的指定区域 pstSrcRect 拷贝到以 pstDst 为目的地址、pstDstRect 为输出区域的内存中。
- 位图信息由[TDE_SURFACE_S](#) 表示，它描述位图的基本信息，包括：位图的像素宽度、像素高度、颜色格式等。
- 操作区域由[TDE_RECT_S](#)表示，它描述位图中参与本次操作的矩形范围，包括：起始位置和尺寸信息。
- 输入和输出的图像宽高需要按照不同格式的对齐要求进行对齐。

3.9 RK_TDE_QuickResize

【描述】

向任务中添加光栅位图缩放操作。

【语法】

```
RK_S32 RK_TDE_QuickResize(TDE\_HANDLE s32Handle,  
                           const TDE\_SURFACE\_S *pstSrc,
```

```

const TDE\_RECT\_S*pstSrcRect,
      const TDE\_SURFACE\_S *pstDst,
const TDE\_RECT\_S*pstDstRect);

```

【参数】

参数名	描述	输入/输出
s32Handle	TDE任务句柄。	输入
pstSrc	源位图。	输入
pstSrcRect	源位图操作区域。	输入
pstDst	目标位图。	输入
pstDstRect	目标位图操作区域。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 TDE错误码 。

【注意】

- [RK_TDE_Open](#)需要被成功调用。
- 将基地址为 pstSrc 的位图以区域 pstSrcRect 指定的尺寸缩放至 pstDstRect 的尺寸，将结果拷贝到以 pstDst 为目的地址、pstDstRect 为输出区域的内存中。
- 缩放大小为，pstSrcRect的尺寸和pstDstRect的尺寸设置比例。
- 缩小和放大的倍数目前都没有限制

3.10 RK_TDE_Bitblit

【描述】

向任务中添加对光栅位图进行有附加功能的搬移操作。

【语法】

```

RK_S32 RK_TDE_Bitblit(TDE\_HANDLE s32Handle,
                      const TDE\_SURFACE\_S *pstBackGround,
const TDE\_RECT\_S*pstBackGroundRect,
                      const TDE\_SURFACE\_S *pstForeGround,
const TDE\_RECT\_S*pstForeGroundRect,
                      const TDE\_SURFACE\_S *pstDst,
const TDE\_RECT\_S*pstDstRect,
                      const TDE\_OPT\_S *pstOpt);

```

【参数】

参数名	描述	输入/输出
s32Handle	TDE任务句柄。	输入
pstBackGround	背景位图。	输入
pstBackGroundRect	背景位图操作区域。	输入
pstForeGround	前景位图。	输入
pstForeGroundRect	前景位图操作区域。	输入
pstDst	目标位图。	输入
pstDstRect	目标位图操作区域。	输入
pstOpt	运算参数设置结构。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为TDE错误码。

【注意】

- [RK_TDE_Open](#)需要被成功调用。
- 将前景位图（pstForeGround）与背景位图（pstBackGround）的指定区域（pstForeGroundRect、pstBackGroundRect）进行运算，将运算后的位图拷贝到目标位图（pstDst）的指定区域（pstDstRect）中。其中背景位图（pstBackGround）的指定区域（pstBackGroundRect）和目标位图（pstDst）的指定区域（pstDstRect）必须一致。
- 支持指定区域旋转、裁剪、缩放、colorkey和叠加操作。暂时不支持ROP操作。
- Alpha 混合操作
有两种方式 一种是将前景位图和背景位图根据配置的混合模型进行Alpha叠加计算，然后输出到背景位图上面。另一种是将背景位图和前景位图进行Alpha叠加后输出到目标位图。
- ColorKey 操作
ColorKey技术是对源图像进行预处理，将符合色键过滤条件的像素的alpha分量置零，其中所述色键过滤条件为透明的颜色值，并将预处理后的源图像与目标图像进行alpha混合模式。RK356x仅支持对前景进行ColorKey的操作。

3.11 RK_TDE_QuickFill

【描述】

向任务中添加快速填充操作。

【语法】

```
RK_S32 RK_TDE_QuickFill(TDE HANDLE s32Handle,
TDE SURFACE S pstDst,
TDE RECT S pstDstRect,
RK_U32 u32FillData);
```

【参数】

参数名	描述	输入/输出
s32Handle	TDE任务句柄。	输入
pstDst	目标位图。	输入
pstDstRect	目标位图操作区域。	输入
u32FillData	填充值。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 TDE错误码 。

【注意】

- [RK_TDE_Open](#)需要被成功调用。
- 该操作直接将u32FillData 填充在位图的指定区域内。
- 目标位图为ARGB1555 / BGRA5551时，u32FillData参数由高到低位分别是A(1bit)、R(5bit)、G(5bit)、B(5bit)。例如，红色：u32FillData = 0x00007C00。
- 其他位图格式，u32FillData参数由高到低位分别是A(8bit)、R(8bit)、G(8bit)、B(8bit)。例如，红色：u32FillData = 0x00ff0000。
- 不支持处理输入的数据为压缩格式。

3.12 RK_TDE_Rotate

【描述】

向任务中添加光栅位图旋转操作。

【语法】

```
RK_S32 RK_TDE_Rotate(TDE\_HANDLE s32Handle,  
                      TDE\_SURFACE\_S pstSrc,  
                      TDE\_RECT\_S pstSrcRect,  
                      TDE\_SURFACE\_S pstDst,  
                      TDE\_RECT\_S pstDstRect,  
                      ROTATION_E enRotateAngle);
```

【参数】

参数名	描述	输入/输出
s32Handle	TDE任务句柄。	输入
pstSrc	源位图。	输入
pstSrcRect	源位图操作区域。	输入
pstDst	目标位图。	输入
pstDstRect	目标位图操作区域。	输入
enRotateAngle	旋转的角度。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 TDE错误码 。

【注意】

- RK_TDE_Open需要被成功调用。
- 将基地址为pstSrc的位图以区域pstSrcRect指定的尺寸旋转至pstDstRect 的尺寸，将结果拷贝到以pstDst为目的地址、pstDstRect为输出区域的内存中，可以做90度，180度和270 度顺时针转。

4. 数据类型

4.1 TDE_MAX_JOB_NUM

【说明】

定义最大的job个数

【定义】

```
#define TDE_MAX_JOB_NUM      128
```

【注意】

- 无

4.2 TDE_MAX_TASK_NUM

【说明】

定义最大的task个数

【定义】

```
#define TDE_MAX_TASK_NUM      200
```

【注意】

- 无

4.3 TDE_HANDLE

【说明】

定义job的句柄。

【定义】

```
typedef RK_S32 TDE_HANDLE;
```

【注意】

- 无

4.4 TDE_SURFACE_S

【说明】

定义job的句柄。

【定义】

```
typedef struct rkTDE_SURFACE_S {
    MB_BLK pMbBlk;
    PIXEL_FORMAT_E enColorFmt;
    RK_U32 u32Height;
    RK_U32 u32Width;
    COMPRESS_MODE_E enComprocessMode;
} TDE_SURFACE_S;
```

【成员】

成员名称	描述
pMbBlk	缓存块句柄。
enColorFmt	图像格式类型。
u32Height	图像高度。
u32Width	图像宽度。
enComprocessMode	图像的压缩类型
bAlphaExt1555	是否使能1555的Alpha扩展。位图格式为ARGB1555时，该项有效。
u8Alpha0	Alpha0值。取值范围：[0, 255]。当位图格式为ARGB1555且bAlphaExt1555为TRUE时，该项有效。在ARGB1555格式下，当像素的最高位为0时，选择该值作为alpha叠加的alpha值。
u8Alpha1	Alpha0值。取值范围：[0, 255]。当位图格式为ARGB1555且bAlphaExt1555为TRUE时，该项有效。在ARGB1555格式下，当像素的最高位为1时，选择该值作为alpha叠加的alpha值

【注意】

- TDE输入输出图像均存储在pMbBlk中，输入输出均需要外部申请合理合法的缓存块。
- RK3588支持输入输出图像的压缩格式设置。
- RK356X不支持压缩图像，需设置为COMPRESS_MODE_NONE。

4.5 TDE_RECT_S

【说明】

TDE 操作区域属性。

【定义】

```
typedef struct rkTDE_RECT_S {
    RK_S32 s32Xpos;
    RK_S32 s32Ypos;
    RK_U32 u32Width;
    RK_U32 u32Height;
} TDE_RECT_S;
```

【成员】

成员名称	描述
s32Xpos	操作区域的起始横坐标，以像素数为单位。有效范围：[0, 位图宽度）。
s32Ypos	操作区域的起始纵坐标，以像素数为单位。有效范围：[0, 位图高度）。
u32Width	操作区域的宽度，以像素数为单位。有效范围：(0, 8192]。
u32Height	操作区域的高度，以像素数为单位。有效范围：(0, 8192]。

【注意】

- TDE输入输出图像均存储在pMbBlk中，输入输出均需要外部申请合理合法的缓存块。
- 操作区域不可超出位图区域，若超出，则返回相应[TDE错误码](#)。

4.6 TDE_COLORKEY_MODE_E

【说明】

TDE colorkey 模式属性。

【定义】

```

typedef enum rkTDE_COLORKEY_MODE_E {
    TDE_COLORKEY_MODE_NONE = 0,
    TDE_COLORKEY_MODE_FOREGROUND,
    TDE_COLORKEY_MODE_BACKGROUND,
    TDE_COLORKEY_MODE_BUTT
} TDE_COLORKEY_MODE_E;

```

【成员】

成员名称	描述
TDE_COLORKEY_MODE_NONE	不做 colorkey 操作。
TDE_COLORKEY_MODE_FOREGROUND	对前景位图进行 colorkey 操作。
TDE_COLORKEY_MODE_BACKGROUND	对背景位图进行 colorkey 操作。
TDE_COLORKEY_MODE_BUTT	无效的 colorkey 模式。

【注意】

- 无

4.7 TDE_BLENDCMD_E

【说明】

TDE blend 模式属性, Alpha 混合命令, 用于计算进行 alpha 混合以后的像素值

【定义】

```

typedef enum rkTDE_BLENDCMD_E {
    TDE_BLENDCMD_NONE = 0x0,
    TDE_BLENDCMD_CLEAR,
    TDE_BLENDCMD_SRC,
    TDE_BLENDCMD_SRCOVER,
    TDE_BLENDCMD_DSTOVER,
    TDE_BLENDCMD_SRCIN,
    TDE_BLENDCMD_DSTIN,
    TDE_BLENDCMD_SRCOUT,
    TDE_BLENDCMD_DSTOUT,
    TDE_BLENDCMD_SRCATOP,
    TDE_BLENDCMD_DSTATOP,
    TDE_BLENDCMD_ADD,
    TDE_BLENDCMD_XOR,
    TDE_BLENDCMD_DST,
    TDE_BLENDCMD_CONFIG,
    TDE_BLENDCMD_BUTT,
} TDE_BLENDCMD_E;

```

【成员】

- pixel = (foreground x fs + background x fd), 其中:
- fs: foreground blend coefficient;
- fd: destination blend coefficient;
- pixel: 运算以后的像素值;
- foreground: 前景位图的像素值;
- background: 背景位图的像素值;
- sa: foreground alpha;
- da: background alpha.

成员名称	描述
TDE_BLENDCMD_NONE	fs 取 sa, fd 取 1.0 - sa
TDE_BLENDCMD_CLEAR	fs 取 0.0, fd 取 0.0
TDE_BLENDCMD_SRC	fs 取 1.0, fd 取 0.0
TDE_BLENDCMD_SRCOVER	fs 取 1.0, fd 取 1.0 - sa
TDE_BLENDCMD_DSTOVER	fs 取 1.0 - da, fd 取 1.0
TDE_BLENDCMD_SRCIN	fs 取 da, fd 取 0.0
TDE_BLENDCMD_DSTIN	fs 取 0.0, fd 取 sa
TDE_BLENDCMD_SRCOUT	fs 取 1.0 - da, fd 取 0.0
TDE_BLENDCMD_DSTOUT	fs 取 0.0, fd 取 1.0 - sa
TDE_BLENDCMD_SRCATOP	fs 取 da, fd 取 1.0 - sa
TDE_BLENDCMD_DSTATOP	fs 取 1.0 - da, fd 取 sa
TDE_BLENDCMD_ADD	fs 取 1.0, fd 取 1.0
TDE_BLENDCMD_XOR	fs 取 1.0 - da, fd 取 1.0 - sa
TDE_BLENDCMD_DST	fs 取 0.0, fd 取 1.0。
TDE_BLENDCMD_CONFIG	用户自己配置参数
TDE_BLENDCMD_BUTT	无效的 alpha 混合命令。

【注意】

- alpha混合目前只支持将前景位图和背景位图根据配置的混合模型进行Alpha叠加计算，然后输出到背景位图上面；
- 目前只支持 TDE_BLENDCMD_SRC、TDE_BLENDCMD_SRCOVER、TDE_BLENDCMD_DSTOVER 和 TDE_BLENDCMD_DST；
- dst 格式不支持 YUV。

4.8 TDE_OPT_S

【说明】

TDE 操作属性结构体。

【定义】

```
typedef struct rkTDE_OPT_S {
    TDE_COLORKEY_MODE_E enColorKeyMode;
    RK_U32        unColorKeyValue;
    MIRROR_E      enMirror;
    TDE_RECT_S    stClipRect;
    RK_U32        u32GlobalAlpha;
    TDE_BLENDCMD_E eBlendCmd;
} TDE_OPT_S;
```

【成员】

成员名称	描述
enColorKeyMode	colorkey 方式。
unColorKeyValue	colorkey 设置值。
enMirror	镜像类型。
stClipRect	clip 区域定义。
u32GlobalAlpha	全局 alpha 值。取值范围：[0, 255]
eBlendCmd	blend 方式

【注意】

- 无

5. TDE错误码

TDE API TDE 错误码如下所示。

错误代码	宏定义	描述
0xA00E8005	RK_ERR_TDE_DEV_NOT_OPEN	TDE设备未打开, API调用失败
0xA00E8012	RK_ERR_TDE_DEV_OPEN_FAILED	开启TDE设备失败
0xA00E8006	RK_ERR_TDE_NULL_PTR	参数中有空指针错误
0xA00E800C	RK_ERR_TDE_NO_MEM	内存不足，无法添加操作
0xA00E8001	RK_ERR_TDE_INVALID_HANDLE	非法的TDE任务句柄
0xA00E8003	RK_ERR_TDE_INVALID_PARA	无效的参数设置
0xA00E803E	RK_ERR_TDE_NOT_ALIGNED	Clut 表的起始地址没有按照4byte对齐
0xA00E803F	RK_ERR_TDE_MINIFICATION	缩小倍数过大
0xA00E8040	RK_ERR_TDE_CLIP_AREA	操作区域与clip区域没有交集，显示不会有更新
0xA00E8041	RK_ERR_TDE_JOB_TIMEOUT	等待超时
0xA00E8042	RK_ERR_TDE_UNSUPPORTED_OP_ERATION	不支持的操作
0xA00E8043	RK_ERR_TDE_QUERY_TIMEOUT	指定的任务超时未完成
0xA00E8044	RK_ERR_TDE_INTERRUPT	等待任务完成被中断

音频子系统

[概述](#)

[功能描述](#)

[API参考](#)

[数据类型](#)

[AUDIO错误码](#)

[附录A：注册解码/编码器API说明](#)

[附录B：静态编译声音质量增强和事件检测功能时编译选项说明](#)

1. 概述

AUDIO 模块包括音频输入、音频输出、音频编码、音频解码四个子模块。音频输入和输出模块通过对RK芯片音频接口的控制实现音频输入输出功能。音频编码和解码模块内部提供g711a、g711u、g722、g726等格式的音频编解码功能，并支持外部注册编解码器。

注意：客户如果需要使用AAC格式的专利，必须从版权权利人处获取授权，并缴纳Licensing Fee。

2. 功能描述

2.1 音频输入和输出

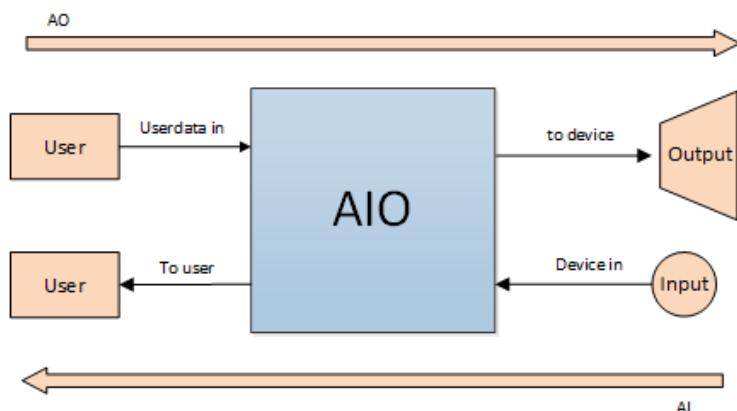
2.1.1 音频输入输出接口

音频输入输出接口简称为AIO（Audio Input/Output）接口，用于对接音频框架Alsa（Linux平台）、Tinyalsa和AudioTrack/AudioRecord（Android平台），完成声音的录制和播放。

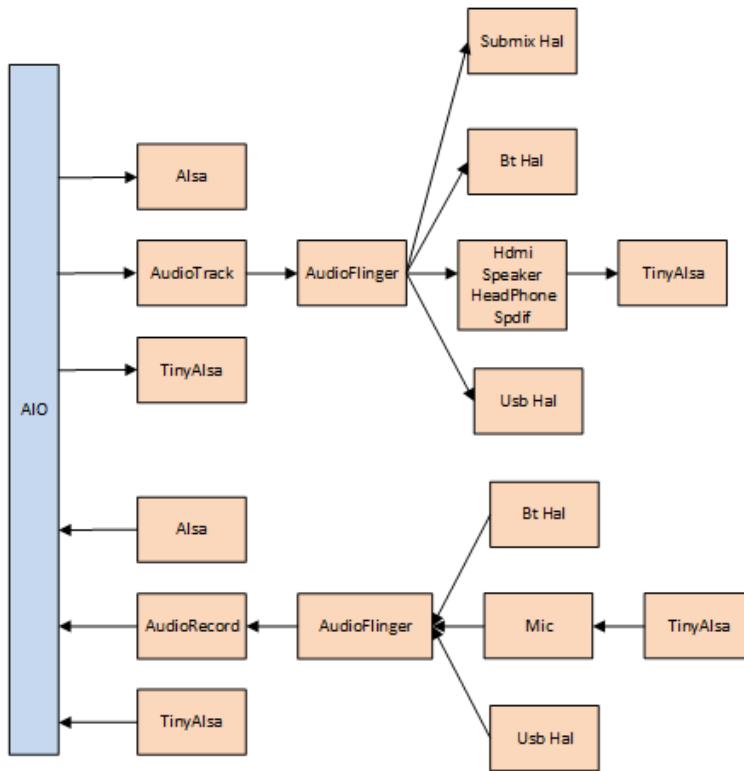
AIO 接口分为两种类型：只支持输入或只支持输出。

- 当为输入类型时，又称为 AI(Audio Input)
- 当为输出类型时，又称为 AO(Audio Output)

AIO接口如下图所示，箭头方向为数据流动方向。



AIO支持芯片平台Linux、Android(仅对接Android 10及以上)的音频框架接口对接，完成声音的录制和播放。



2.1.2 AIO设备

软件中负责抽象音频接口输入功能的，称之为AI设备；负责抽象音频接口输出功能的单元，称之为AO设备。对每个输入输出接口，软件根据该接口支持的功能，分别与AI设备和AO设备建立映射关系。例如：AI0只支持音频输入，则AI0映射为AiDev0；AO0只支持音频输出，则AO0映射为AoDev0。

AIO设备号是抽象的软件层次的概念，设备号一般从0开始，比如AoDev0, AoDev1....., AiDev0, AiDev1....，通常意义上，一个AIO设备号对应一个Audio驱动的声卡，以如下挂载的声卡为例，实际客户可能挂载的声卡会不同。

```
rk356x_box:/ $ 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
```

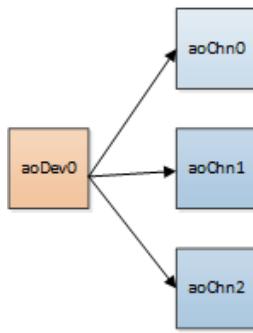
如上截图中，定义了3个声卡。序号为0的声卡为HDMI声卡，其只有放音功能；序号为1的声卡为RK809 Codec对应的声卡，其有放音和录音功能；序号为2的声卡为SPDIF声卡，其只有放音功能。

因此对于有如上配置的产品，我们可以定义3个AO设备，一个AI设备，例如定义AoDev0, AoDev1, AoDev2分别对应HDMI, RK809以及SPDIF的放音通路，定义AiDev0对应RK809录音通路。需要强调的是，AIO设备号，并不是跟声卡(Audio Codec)的序号一一对应的，即设备号为0(即AoDev0)并不一定去输出声卡0，也可以去输出声卡1或者声卡2，为了好理解，建议一一对应。AIO设备与其实用的声卡的对应关系，请见FAQ文档AUDIO章节介绍。

2.1.3 AIO通道

AIO通道，这里指的是AIO设备上输入/出的音频流，一个音频流对应一个通道。

- 对AO，一个设备可以创建多个通道(音频流)，多个通道的数据经混音成一路音频流后，输出声卡上播放出来。目前AO通路上只支持Alsa(Linux)和AudioFlinger(Android)的混音，TinyAlsa不支持多通道混音。

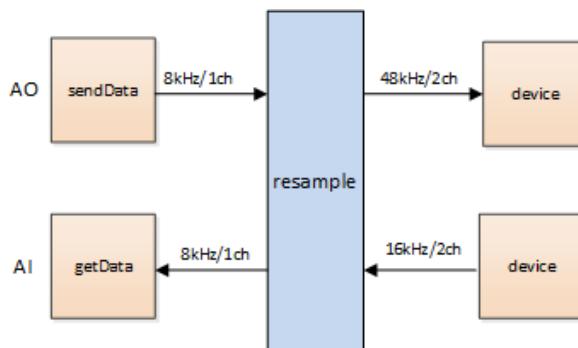


- 对AI，一个设备只支持一个通道。

2.1.4 重采样

重采样是指对音频数据的采样率(samplerate)，声道数(channels)和格式(format)进行转换。AIO模块支持对音频数据实施重采样。如果启用AI重采样功能，AI内部按照重采样设置，进行重采样处理，应用可通过调用RK_MPI_AI_GetFrame接口获取处理的数据。同理，如果启用了AO重采样功能，则用户通过RK_MPI_AO_SendFrame将音频数据送给AO，AO内部会对数据进行重采样处理，并将处理后的数据通过声卡播放出来。

- AO重采样主要是将输入数据重采成声卡支持的数据。如下图，AO接收到的数据为(8kHz, 1ch)，而声卡只支持为(48kHz, 2ch)数据，此时可开启AO重采样，将(8kHz, 1ch)的数据转换成声卡支持的(48kHz, 2ch)数据。
- AI重采样主要是将声卡采集到的数据转换成应用所需要获取数据。如下图，录音声卡只支持(16kHz, 2ch)的参数，而应用需要(8kHz, 1ch)的数据，此时可开启AI重采样，会重采样成用户需要的数据。
- AIO模块中，对输入/出的每帧数据的采样率，声道数，格式等都有判断，对于经过AIO重采样处理的数据，如果进入数据的参数和重采样设置的参数相同，那么将不会进入重采样处理，直接输出给后级处理。以AO为例，AO接收到的(48kHz, 2ch)数据，AO重采样开启，且设置重采样参数为(48kHz, 2ch)，那么当(48kHz, 2ch)的数据流程重采样处理时，数据不会进行重采处理，而是直接送入重采后的下一级处理，因此即使开启了AO的重采处理，在条件满足时(即重采后的是参数与送入数据参数相等时)，也不会因为重采开启而浪费cpu资源，因此建议客户始终开启AIO的重采样功能。



2.2 音频编码和解码

2.2.1 音频编解码流程

rockit支持的编解码类型为g711a、g711u、g722、g726。音频的编解码都是通过CPU软解。支持SYS模块的绑定接口，将一个AI通道绑定到AENC通道，实现录音编码功能；也可以将一个ADEC通道绑定到AO通道，实现解码播放功能。

2.2.2 音频编解码协议

rockit内部支持的音频编解码协议如下表所示。

协议	采样率	声道数	码率(kbps)	码字(bits)	压缩比	描述
g711a	8000 16000	1 2	64	8	16:8	优点：语音质量最好；CPU消耗小；支持广泛。缺点：压缩效率低。欧洲和其他地区大都采用A律编码。
g711u	8000 16000	1 2	64	8	16:8	优点：语音质量最好；CPU消耗小；支持广泛。缺点：压缩效率低。北美与日本通常采用μ律编码。
g726	8000	1	40 32 24 16	5 4 3 2	16:2 16:3 16:4 16:5	g726编解码类型需要按照码率设置对应的码字(codeword)，g726编解码器把128kbit/s线性数据(64kbit/s PCM数据)压缩为16kbit/s、24kbit/s、32kbit/s、40kbit/s，数据压缩比分别为8:1、16:3、4:1和16:5，码字分别为2、3、4和5 bits。采用越高压缩比，码率越小，质量越差。最常用的是32kbit/s，即设置码字为4就好。
g722	16000	1	64	8	16:8	g722的优点是延时和传输位误差率非常低。

2.3 音频输入输出设备和内置codec

不同客户需要的产品形态不同，需要的声卡驱动不同，需要实际调整，配置主从模式和声卡配置需要修改内核dts。实际开发的时候联系驱动音频工程师进行指导。

3. API参考

3.1 音频输入

该功能模块为用户提供以下API:

- [RK_MPI_AI_SetPubAttr](#): 设置AI设备属性。
- [RK_MPI_AI_GetPubAttr](#): 获取AI设备属性。
- [RK_MPI_AI_Enable](#): 启用AI设备。
- [RK_MPI_AI_Disable](#): 禁用AI设备。
- [RK_MPI_AI_EnableChn](#): 启用AI通道。
- [RK_MPI_AI_DisableChn](#): 禁用AI通道。
- [RK_MPI_AI_EnableDataRead](#): 启用AI读数据模式。
- [RK_MPI_AI_DisableDataRead](#): 禁用AI读数据模式。
- [RK_MPI_AI_SendFrame](#): 发送音频帧。
- [RK_MPI_AI_GetFrame](#): 获取音频帧。
- [RK_MPI_AI_EnableReSmp](#): 启用AI重采样。
- [RK_MPI_AI_DisableReSmp](#): 禁用AI重采样。
- [RK_MPI_AI_ReleaseFrame](#): 释放音频帧。
- [RK_MPI_AI_SetChnParamt](#): 获取AI通道参数。
- [RK_MPI_AI_GetChnParam](#): 获取AI通道参数。
- [RK_MPI_AI_SetTrackMode](#): 设置声道模式。
- [RK_MPI_AI_GetTrackMode](#): 获取声道模式。
- [RK_MPI_AI_QueryFileStatus](#): 查询音频输出通道是否处于存文件的状态。
- [RK_MPI_AI_SaveFile](#): 开启音频输出保存文件功能。
- [RK_MPI_AI_ClrPubAttr](#): 清除AI设备属性。
- [RK_MPI_AI_EnableVqe](#): 启用AI声音质量增强相关属性。
- [RK_MPI_AI_DisableVqe](#): 禁用AI声音质量增强相关属性。
- [RK_MPI_AI_SetVqeAttr](#): 设置AI声音质量增强功能相关属性。
- [RK_MPI_AI_GetVqeAttr](#): 获取AI声音质量增强功能相关属性。
- [RK_MPI_AI_SetVqeModuleEnable](#): 启用AI声音质量增强使能模块。
- [RK_MPI_AI_EnableAed](#): 启用AI声音事件检测相关属性。

- [RK_MPI_AI_DisableAed](#): 禁用AI声音事件检测相关属性。
- [RK_MPI_AI_SetAedAttr](#): 设置AI声音事件检测功能相关属性。
- [RK_MPI_AI_GetAedAttr](#): 获取AI声音事件检测功能相关属性。
- [RK_MPI_AI_GetAedResult](#): 获取AI声音事件检测结果。
- [RK_MPI_AI_EnableBcd](#): 启用AI婴儿哭声检测相关属性。
- [RK_MPI_AI_DisableBcd](#): 禁用AI婴儿哭声检测相关属性。
- [RK_MPI_AI_SetBcdAttr](#): 设置AI婴儿哭声检测功能相关属性。
- [RK_MPI_AI_GetBcdAttr](#): 获取AI婴儿哭声检测功能相关属性。
- [RK_MPI_AI_GetBcdResult](#): 获取AI婴儿哭声事件检测结果。
- [RK_MPI_AI_EnableBuz](#): 启用AI蜂鸣器报警声检测相关属性。
- [RK_MPI_AI_DisableBuz](#): 禁用AI蜂鸣器报警声检测相关属性。
- [RK_MPI_AI_SetBuzAttr](#): 设置AI蜂鸣器报警声检测功能相关属性。
- [RK_MPI_AI_GetBuzAttr](#): 获取AI蜂鸣器报警声检测功能相关属性。
- [RK_MPI_AI_GetBuzResult](#): 获取AI蜂鸣器报警声事件检测结果。
- [RK_MPI_AI_EnableGbs](#): 启用AI玻璃破碎声检测相关属性。
- [RK_MPI_AI_DisableGbs](#): 禁用AI玻璃破碎声检测相关属性。
- [RK_MPI_AI_SetGbsAttr](#): 设置AI玻璃破碎声检测功能相关属性。
- [RK_MPI_AI_GetGbsAttr](#): 获取AI玻璃破碎声检测功能相关属性。
- [RK_MPI_AI_GetGbsResult](#): 获取AI玻璃破碎声事件检测结果。

3.2 RK_MPI_AI_SetPubAttr

【描述】

设置 AI 设备属性。

【语法】

```
RK_S32 RK_MPI_AI_SetPubAttr(AUDIO\_DEV AiDevId, const AIO\_ATTR\_S *pstAttr);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
pstAttr	AI设备属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AUDIO错误码 。

【注意】

音频输入的属性包括输出数据的属性和输入设备(声卡)属性: 采样率、声道数、采样精度、buffer大小、每帧的采样点数等。

- 在设置声卡设备属性之前需要保证AiDevId对应的AI处于禁用状态, 如果处于启用状态则需要首先禁用AI声卡设备。
- 采样率

采样率是指一秒钟的采样率点数。设置声卡设备的采样率, 要确认声卡设备是否支持所需要设定的采样率。

- 通道数目

指当前输入设备的 AI 功能的通道数目。需要与对接的Audio Codec的配置保持一致。如rk809-codec仅支持2路录制。

3.3 RK_MPI_AI_GetPubAttr

【描述】

获取 AI 设备属性。

【语法】

```
RK_S32 RK_MPI_AI_GetPubAttr(AUDIO\_DEV AiDevId, AIO\_ATTR\_S *pstAttr);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围：[0, AI_DEV_MAX_NUM)。	输入
pstAttr	AI设备属性指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 获取的属性为前一次配置的属性。
- 如果从未配置过属性，则返回属性未配置的错误。

3.4 RK_MPI_AI_Enable

【描述】

启用AI设备。

【语法】

```
RK_S32 RK_MPI_AI_Enable(AUDIO\_DEV AiDevId);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围：[0, AI_DEV_MAX_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 要求在启用前配置 AI 设备属性，否则会返回属性未配置的错误。
- 如果 AI 设备已经启用，重复启用，则返回正在使用中的错误。

3.5 RK_MPI_AI_Disable

【描述】

禁用AI设备。

【语法】

```
RK_S32 RK_MPI_AI_Disable(AUDIO\_DEV AiDevId);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 如果 AI 设备未启用，则返回未启用的AUDIO错误码。
- 禁用 AI 设备前必须先禁用设备下所有 AI 通道。
- 声卡设备使用完成后，必须关闭声卡设备。

3.6 RK_MPI_AI_EnableChn

【描述】

启用AI通道。

【语法】

```
RK_S32 RK_MPI_AI_EnableChn(AUDIO\_DEV AiDevId, AI\_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 启用AI通道前，必须先启用其所属的AI设备，否则返回设备未启动的AUDIO错误码。

3.7 RK_MPI_AI_DisableChn

【描述】

禁用 AI 通道。

【语法】

```
RK_S32 RK_MPI_AI_DisableChn(AUDIO\_DEV AiDevId, AI\_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AUDIO错误码 。

【注意】

- 通道使用完成后, 必须调用关闭AI通道。

3.8 RK_MPI_AI_EnableDataRead

【描述】

启用AI读数据模式。

【语法】

```
RK_S32 RK_MPI_AI_EnableDataRead(AUDIO\_DEV AiDevId, AI\_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AUDIO错误码 。

【注意】

- 启用AI读数据模式前, 必须先启用其所属的AI设备, 否则返回设备未启动的AUDIO错误码。

3.9 RK_MPI_AI_DisableDataReader

【描述】

禁用AI读数据模式。

【语法】

```
RK_S32 RK_MPI_AI_DisableDataReader(AUDIO\_DEV AiDevId, AI\_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围：[0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围：[0, AI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 读数据模式使用完成后，必须调用关闭AI读数据模式。

3.10 RK_MPI_AI_SendFrame

【描述】

发送音频帧。

【语法】

```
RK_S32 RK_MPI_AI_SendFrame(AUDIO\_DEV AiDevId, AI\_CHN AiChn, AUDIO\_FRAME\_S *pstFrm, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围：[0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围：[0, AI_MAX_CHN_NUM)。	输入
pstFrm	音频帧结构体指针。	输出
s32MilliSec	获取数据的超时时间： -1表示阻塞模式； >=0表示非阻塞模式，设置读取数据的超时时间s32MilliSec(毫秒)	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- s32MilliSec 等于-1 时，表示采用阻塞模式发送数据，该接口会被阻塞直到数据被成功送入；大于等于 0 时，表示非阻塞模式发送数据的超时时间(毫秒)。如在超时时间内完成数据的发送，则返回成功，否则返回超时并报错。
- 发送音频帧数据前，必须先开启对应的AI 通道和声卡设备。

3.11 RK_MPI_AI_GetFrame

【描述】

获取音频帧。

【语法】

```
RK_S32 RK_MPI_AI_GetFrame(AUDIO\_DEV AiDevId, AI\_CHN AiChn, AUDIO\_FRAME\_S pstFrm, AEC\_FRAME\_S pstAecFrm, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围：[0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围：[0, AI_MAX_CHN_NUM)。	输入
pstFrm	音频帧结构体指针。	输出
pstAecFrm	回声抵消参考帧结构体指针。	输出
s32MilliSec	获取数据的超时时间： -1表示阻塞模式； >=0表示非阻塞模式，设置读取数据的超时时间s32MilliSec(毫秒)	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 目前pstAecFrm未使用，置为空。
- s32MilliSec 等于-1 表示采用阻塞模式读取数据，该接口会一直阻塞直到成功读取到数据或当前通道退出/禁用为止；大于等于 0 ，表示以非阻塞模式读取数据的超时时间(毫秒)，如在s32MilliSec 毫秒内没有读取到数据，则返回超时并报错。
- 获取音频帧数据前，必须先开启对应的AI 通道和声卡设备。

3.12 RK_MPI_AI_ReleaseFrame

【描述】

释放音频帧。

【语法】

```
RK_S32 RK_MPI_AI_ReleaseFrame(AUDIO\_DEV AiDevId, AI\_CHN AiChn, const AUDIO\_FRAME\_S pstFrm, const AEC\_FRAME\_S pstAecFrm);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围：[0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围：[0, AI_MAX_CHN_NUM)。	输入
pstFrm	音频帧结构体指针。	输入
pstAecFrm	回声抵消参考帧结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 目前不需要释放回声抵消参考帧，pstAecFrm置为NULL即可。

3.13 RK_MPI_AI_SetChnParam

【描述】

设置 AI 通道参数。

【语法】

```
RK_S32 RK_MPI_AI_SetChnParam(AUDIO\_DEV AiDevId, AI\_CHN AiChn, const AI\_CHN\_PARAM\_S *pstChnParam);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围：[0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围：[0, AI_MAX_CHN_NUM)。	输入
pstChnParam	音频通道参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 通道参数目前只有一个成员变量，用于设置用户获取音频帧的缓存个数。默认为4个。

3.14 RK_MPI_AI_GetChnParam

【描述】

获取 AI 通道参数。

【语法】

```
RK_S32 RK_MPI_AI_GetChnParam(AUDIO\_DEV AiDevId, AI\_CHN AiChn, AI\_CHN\_PARAM\_S*pstChnParam);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstChnParam	音频通道参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AUDIO错误码 。

【注意】

- 无。

3.15 RK_MPI_AI_EnableReSmp

【描述】

启用AI重采样。

【语法】

```
RK_S32 RK_MPI_AI_EnableReSmp(AUDIO\_DEV AiDevId, AI\_CHN AiChn, AUDIO\_SAMPLE\_RATE\_E enOutSampleRate);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
enOutSampleRate	音频重采样的输出采样率。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AUDIO错误码 。

【注意】

- 在启用AI通道之后, 调用此接口启用重采样功能。
- 允许重复启用重采样功能, 但必须保证后配置的重采样输入采样率与之前配置的重采样输入采样率不一样。
- 在禁用AI通道后, 如果重新启用AI通道, 并使用重采样功能, 需调用此接口重新启用重采样。
- AI 重采样的输入采样率为读取数据的采样率。
- 强烈建议启用此接口, 内部插件会判断会有参数判断, 参数一致时数据会直接bypass。

【举例】

以 AI 从录制 32K 到 8K 的重采样为例, 配置如下:

```

/* dev attr of ai */
aiAttr.soundCard.channels = 2;
aiAttr.soundCard.sampleRate = 32000;
aiAttr.soundCard.bitWidth = AUDIO_BIT_WIDTH_16;

aiAttr.enBitwidth = AUDIO_BIT_WIDTH_16;
aiAttr.enSamplerate = AUDIO_SAMPLE_RATE_8000;
aiAttr.enSoundmode = AUDIO_SOUND_MODE_STEREO;
aiAttr.u32FrmNum = 4;
aiAttr.u32PtNumPerFrm = 1024;
RK_MPI_AI_EnableReSmp(AiDev, AiChn, AUDIO_SAMPLE_RATE_8000);

```

3.16 RK_MPI_AI_DisableReSmp

【描述】

禁用 AI 重采样。

【语法】

```
RK_S32 RK_MPI_AI_DisableReSmp(AUDIO\_DEV AiDevId, AI\_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输出通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AUDIO错误码 。

【注意】

- 不再使用 AI 重采样功能的话, 应该调用此接口将其禁用。

3.17 RK_MPI_AI_SetTrackMode

【描述】

设置 AI 通道模式。

【语法】

```
RK_S32 RK_MPI_AI_SetTrackMode(AUDIO\_DEV AiDevId, AUDIO\_TRACK\_MODE\_E enTrackMode);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
enTrackMode	音频声道模式。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 在 AI 设备成功启用后再调用此接口。

3.18 RK_MPI_AI_GetTrackMode

【描述】

获取 AI 声道模式。

【语法】

```
RK_S32 RK_MPI_AI_GetTrackMode(AUDIO\_DEV AiDevId, AUDIO\_TRACK\_MODE\_E *penTrackMode);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入
penTrackMode	音频输入声道模式指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 在 AI 设备成功启用后再调用此接口。

3.19 RK_MPI_AI_ClrPubAttr

【描述】

清空Pub属性。

【语法】

```
RK_S32 RK_MPI_AI_ClrPubAttr(AUDIO\_DEV AiDevId);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围：[0, AI_DEV_MAX_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 清除设备属性前，需要先停止设备。

3.20 RK_MPI_AI_SaveFile

【描述】

开启音频输入保存文件功能。

【语法】

```
RK_S32 RK_MPI_AI_SaveFile(AUDIO\_DEV AiDevId, AI\_CHN AiChn, const AUDIO\_SAVE\_FILE\_INFO\_S*pstSaveFileInfo);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围：[0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输出通道号。 取值范围：[0, AI_MAX_CHN_NUM)。	输入
pstSaveFileInfo	音频保存文件属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 此接口仅用于AI非绑定模式下保存AI从声卡录制的PCM文件。

3.21 RK_MPI_AI_QueryFileStatus

【描述】

查询音频输入通道是否处于存文件的状态。

【语法】

```
RK_S32 RK_MPI_AI_QueryFileStatus(AUDIO\_DEV AiDevId, AI\_CHN AiChn, AUDIO\_FILE\_STATUS\_S* pstFileStatus);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围：[0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围：[0, AI_MAX_CHN_NUM)。	输入
pstFileStatus	状态属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 此接口用于查询音频输入通道是否处于存文件的状态，当用户调用[RK_MPI_AI_SaveFile](#)存储文件后，可调用此接口查询存储的文件是否达到了指定的大小，如果 `pstFileStatus` 的 `bSaving` 为 `RK_TRUE`，说明还没有达到指定大小，为 `RK_FALSE` 则已经达到指定大小。

3.22 RK_MPI_AI_EnableVqe

【描述】

启用AI声音质量增强相关属性。

【语法】

```
RK_S32 RK_MPI_AI_EnableVqe(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围：[0, AI_DEV_MAX_NUM]。	输入
AiChn	音频输入通道号。 取值范围：[0, AI_MAX_CHN_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 AUDIO错误码 。

【注意】

- 声音质量增强功能启用时，输出的音频声道数都将变为1

3.23 RK_MPI_AI_DisableVqe

【描述】

禁用AI声音质量增强相关属性。

【语法】

```
RK_S32 RK_MPI_AI_DisableVqe(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围：[0, AI_DEV_MAX_NUM]。	输入
AiChn	音频输入通道号。 取值范围：[0, AI_MAX_CHN_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.24 RK_MPI_AI_SetVqeAttr

【描述】

设置AI声音质量增强相关属性。

【语法】

```
RK_S32 RK_MPI_AI_SetVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_DEV AoDevId, AO_CHN AoChn, const
AI_VQE_CONFIG_S *pstVqeConfig);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
AoDevId	用于回声抵消的AO设备号。 取值范围: [0, AO_DEV_MAX_NUM)。	输入
AoChn	用于回声抵消的AO通道号。 取值范围: [0, AO_MAX_CHN_NUM)。	输入
pstVqeConfig	VQE属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 需要给pstVqeConfig结构体成员参数aCfgFile指定AI VQE对应的json配置文件路径。

3.25 RK_MPI_AI_GetVqeAttr

【描述】

获取AI声音质量增强相关属性。

【语法】

```
RK_S32 RK_MPI_AI_GetVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AI\_VQE\_CONFIG\_S *pstVqeConfig);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstVqeConfig	VQE属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.26 RK_MPI_AI_SetVqeModuleEnable

【描述】

启用AI声音质量增强使能模块。

【语法】

```
RK_S32 RK_MPI_AI_SetVqeModuleEnable(AUDIO_DEV AiDevId, AI_CHN AiChn,
AI\_VQE\_MOD\_ENABLE\_S *pstVqeModuleEnable);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstVqeModuleEnable	VQE模块使能结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.27 RK_MPI_AI_EnableAed

【描述】

启用AI声音质量增强相关属性。

【语法】

```
RK_S32 RK_MPI_AI_EnableAed(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.28 RK_MPI_AI_DisableAed

【描述】

禁用AI声音质量增强相关属性。

【语法】

```
RK_S32 RK_MPI_AI_DisableAed(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.29 RK_MPI_AI_SetAedAttr

【描述】

设置AI声音质量增强相关属性。

【语法】

```
RK_S32 RK_MPI_AI_SetAedAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, const AI\_AED\_CONFIG\_S *pstAedConfig);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstAedConfig	AED属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.30 RK_MPI_AI_GetAedAttr

【描述】

获取AI声音质量增强相关属性。

【语法】

```
RK_S32 RK_MPI_AI_GetAedAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AI\_AED\_CONFIG\_S *pstAedConfig);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstAedConfig	AED属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.31 RK_MPI_AI_GetAedResult

【描述】

获取AI声音质量增强相关属性。

【语法】

```
RK_S32 RK_MPI_AI_GetAedResult(AUDIO_DEV AiDevId, AI_CHN AiChn, AI\_AED\_RESULT\_S *pstAedResult);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstAedResult	AED事件检测结果结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.32 RK_MPI_AI_EnableBcd

【描述】

启用AI婴儿哭声检测相关属性。

【语法】

```
RK_S32 RK_MPI_AI_EnableBcd(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.33 RK_MPI_AI_DisableBcd

【描述】

禁用AI婴儿哭声检测相关属性。

【语法】

```
RK_S32 RK_MPI_AI_DisableBcd(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.34 RK_MPI_AI_SetBcdAttr

【描述】

设置AI婴儿哭声检测相关属性。

【语法】

```
RK_S32 RK_MPI_AI_SetBcdAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, const AI_BCD_CONFIG_S *pstBcdConfig);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstBcdConfig	BCD属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.35 RK_MPI_AI_GetBcdAttr

【描述】

获取AI婴儿哭声检测相关属性。

【语法】

```
RK_S32 RK_MPI_AI_GetBcdAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AI_BCD_CONFIG_S *pstBcdConfig);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstBcdConfig	BCD属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.36 RK_MPI_AI_GetBcdResult

【描述】

获取AI婴儿哭声事件检测结果。

【语法】

```
RK_S32 RK_MPI_AI_GetBcdResult(AUDIO_DEV AiDevId, AI_CHN AiChn, AI\_BCD\_RESULT\_S*pstBcdResult);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstBcdResult	BCD事件检测结果结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.37 RK_MPI_AI_EnableBuz

【描述】

启用AI蜂鸣器报警声相关属性。

【语法】

```
RK_S32 RK_MPI_AI_EnableBuz(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.38 RK_MPI_AI_DisableBuz

【描述】

禁用AI蜂鸣器报警声相关属性。

【语法】

```
RK_S32 RK_MPI_AI_DisableBuz(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.39 RK_MPI_AI_SetBuzAttr

【描述】

设置AI蜂鸣器报警声相关属性。

【语法】

```
RK_S32 RK_MPI_AI_SetBuzAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, const AI\_BUZ\_CONFIG\_S *pstBcdConfig);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstBuzConfig	BUZ属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.40 RK_MPI_AI_GetBuzAttr

【描述】

获取AI蜂鸣器报警声相关属性。

【语法】

```
RK_S32 RK_MPI_AI_GetBuzAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AI\_BUZ\_CONFIG\_S *pstBuzConfig);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstBuzConfig	BUZ属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.41 RK_MPI_AI_GetBuzResult

【描述】

获取AI蜂鸣器报警声事件检测结果。

【语法】

```
RK_S32 RK_MPI_AI_GetBuzResult(AUDIO_DEV AiDevId, AI_CHN AiChn, AI\_BUZ\_RESULT\_S*pstBuzResult);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstBuzResult	BUZ事件检测结果结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.42 RK_MPI_AI_EnableGbs

【描述】

启用AI玻璃破碎声相关属性。

【语法】

```
RK_S32 RK_MPI_AI_EnableBcd(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.43 RK_MPI_AI_DisableGbs

【描述】

禁用AI玻璃破碎声相关属性。

【语法】

```
RK_S32 RK_MPI_AI_DisableGbs(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.44 RK_MPI_AI_SetGbsAttr

【描述】

设置AI玻璃破碎声相关属性。

【语法】

```
RK_S32 RK_MPI_AI_SetGbsAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, const AI\_GBS\_CONFIG\_S *pstGbsConfig);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstGbsConfig	GBS属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.45 RK_MPI_AI_GetGbsAttr

【描述】

获取AI玻璃破碎声相关属性。

【语法】

```
RK_S32 RK_MPI_AI_GetGbsAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AI\_GBS\_CONFIG\_S*pstGbsConfig);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstGbsConfig	GBS属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.46 RK_MPI_AI_GetGbsResult

【描述】

获取AI玻璃破碎声事件检测结果。

【语法】

```
RK_S32 RK_MPI_AI_GetGbsResult(AUDIO_DEV AiDevId, AI_CHN AiChn, AI\_GBS\_RESULT\_S*pstGbsResult);
```

【参数】

参数名	描述	输入/输出
AiDevId	音频设备号。 取值范围: [0, AI_DEV_MAX_NUM)。	输入
AiChn	音频输入通道号。 取值范围: [0, AI_MAX_CHN_NUM)。	输入
pstGbsResult	GBS事件检测结果结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

3.47 音频输出

该功能模块为用户提供以下API:

- [RK_MPI_AO_SetPubAttr](#): 设置 AO 设备属性。
- [RK_MPI_AO_GetPubAttr](#): 获取 AO 设备属性。
- [RK_MPI_AO_Enable](#): 启用 AO 设备。
- [RK_MPI_AO_Disable](#): 禁用 AO 设备。
- [RK_MPI_AO_EnableChn](#): 启用 AO 通道。
- [RK_MPI_AO_DisableChn](#): 禁用 AO 通道。
- [RK_MPI_AO_SendFrame](#): 发送 AO 音频帧。
- [RK_MPI_AO_EnableReSmp](#): 启用 AO 重采样。
- [RK_MPI_AO_DisableReSmp](#): 禁用 AO 重采样。
- [RK_MPI_AO_PauseChn](#): 暂停 AO 通道。
- [RK_MPI_AO_ResumeChn](#): 恢复 AO 通道。
- [RK_MPI_AO_ClearChnBuf](#): 清除 AO 通道中当前的音频数据缓存。
- [RK_MPI_AO_QueryChnStat](#): 查询 AO 通道中当前的音频数据缓存状态。
- [RK_MPI_AO_SetTrackMode](#): 置 AO 设备声道模式。
- [RK_MPI_AO_GetTrackMode](#): 获取 AO 设备声道模式。
- [RK_MPI_AO_SetVolume](#): 设置 AO 设备音量大小。
- [RK_MPI_AO_GetVolume](#): 获取 AO 设备音量大小。
- [RK_MPI_AO_SetChnParams](#): 设置 AO 通道参数。
- [RK_MPI_AO_GetChnParams](#): 获取 AO 通道参数。
- [RK_MPI_AO_SetMute](#): 设置 AO 设备静音状态。
- [RK_MPI_AO_GetMute](#): 获取 AO 设备静音状态。
- [RK_MPI_AO_SaveFile](#): 开启音频输出保存文件功能。
- [RK_MPI_AO_QueryFileStatus](#): 查询音频输出通道是否处于存文件的状态。
- [RK_MPI_AO_ClrPubAttr](#): 清除 AO 设备属性。
- [RK_MPI_AO_WaitEos](#): 等待指定设备和通道播放完成。

3.48 RK_MPI_AO_SetPubAttr

【描述】

设置AO设备（声卡）参数。

【语法】

```
RK_S32 RK_MPI_AO_SetPubAttr(AUDIO\_DEV AoDevId, const AIO\_ATTR\_S *pstAttr);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围: [0, AO_DEV_MAX_NUM)。	输入
pstAttr	音频输出设备属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AUDIO错误码 。

【注意】

- 在设置AO属性之前，需要确保AO处于禁用状态，如果处于启用状态则需要首先禁用AO声卡设备。
- 必须设置声卡设备的支持的采样率，声道数。其他参数可选配置(没有设置则采用默认值)，参见[AIO_ATTR_S](#)结构体。
- linux平台声卡驱动配置参见/etc/asound.conf文件，linux rk356x声卡默认配置使用pcm.card0。

3.49 RK_MPI_AO_GetPubAttr

【描述】

获取AO设备属性。

【语法】

```
RK_S32 RK_MPI_AO_GetPubAttr(AUDIO\_DEV AoDevId, AIO\_ATTR\_S *pstAttr);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入
pstAttr	音频输出设备属性。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 获取设置的AO属性。
- 如果从未配置过属性，则返回属性未配置的错误。

3.50 RK_MPI_AO_Enable

【描述】

启用AO设备。

【语法】

```
RK_S32 RK_MPI_AO_Enable(AUDIO\_DEV AoDevId);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 要求在启用前配置AO设备属性，否则会返回属性未配置的错误。

- 如果 AO 设备已经启用，重复启用，则返回正在使用中的错误。

3.51 RK_MPI_AO_Disable

【描述】

禁用AO设备。

【语法】

```
RK_S32 RK_MPI_AO_Disable(AUDIO_DEV AoDevId);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 如果 AO 设备未启用，则返回未启用的AUDIO错误码。
- 禁用 AO 设备前必须先禁用设备下所有 AO 通道。

3.52 RK_MPI_AO_EnableChn

【描述】

启用AO通道。

【语法】

```
RK_S32 RK_MPI_AO_EnableChn(AUDIO\_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入
AoChn	音频输出通道号。 取值范围：[0, AO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 启用AO通道前，必须先启用其所属的AO设备，否则返回设备未启动的AUDIO错误码。

3.53 RK_MPI_AO_DisableChn

【描述】

禁用 AO 通道。

【语法】

```
RK_S32 RK_MPI_AO_DisableChn(AUDIO\_DEV AoDevId, AO\_CHN AoChn);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入
AoChn	音频输出通道号。 取值范围：[0, AO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 通道使用完成后，必须调用此函数关闭AO通道。

3.54 RK_MPI_AO_SendFrame

【描述】

发送AO音频帧。

【语法】

```
RK_S32 RK_MPI_AO_SendFrame(AUDIO\_DEV AoDevId, AO\_CHN AoChn, const AUDIO\_FRAME\_S *pstData, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入
AoChn	音频输出通道号。 取值范围：[0, AO_MAX_CHN_NUM)。	输入
pstData	音频帧结构体指针。	输入
s32MilliSec	发送数据的超时时间： -1表示阻塞模式； >=0表示设置非阻塞模式的超时时间。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 该接口用于用户发送音频帧至 AO 输出，如果 AO 通道已经通过系统绑定（RK_MPI_SYS_Bind）接口与 AI 或 ADEC 绑定，不需要也不建议调此接口。
- s32MilliSec 等于-1 时，表示采用阻塞模式发送数据，该接口会被阻塞直到数据被成功送入；大于等于 0 时，表示非阻塞模式发送数据的超时时间(毫秒)。如在超时时间内完成数据的发送，则返回成功，否则返回超时并报错。
- 调用该接口发送音频帧到 AO 输出时，必须先开启对应的 AO 通道和声卡设备。
- 音频申请内存的方式建议尽量使用 malloc 方式申请。

3.55 RK_MPI_AO_EnableReSmp

【描述】

启用AO重采样。

【语法】

```
RK_S32 RK_MPI_AO_EnableReSmp(AUDIO\_DEV AoDevId, AO\_CHN AoChn, AUDIO\_SAMPLE\_RATE\_E enInSampleRate);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM]。	输入
AoChn	音频输出通道号。 取值范围：[0, AO_MAX_CHN_NUM]。	输入
enInSampleRate	音频重采样的输入采样率。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO 错误码。

【注意】

- 应该在启用AO通道之后，绑定AO通道之前，调用此接口启用重采样功能。
- 允许重复启用重采样功能，但必须保证后配置的重采样输入采样率与之前配置的重采样输入采样率不一样。
- 在禁用AO通道后，如果重新启用AO通道，并使用重采样功能，需调用此接口重新启用重采样。
- AO重采样接口设置的采样率为发送数据的采样率。
- 就算不做重采样，也建议启用此接口，内部插件会判断会有参数判断，参数一致时数据会直接bypass。

【举例】

以 ADEC 到 AO 的解码回放 8K 到 32K 重采样为例，配置如下：

```
/* dev attr of ao */
AIO_ATTR_S aoAttr;
/* dev attr of ai */
aoAttr.soundCard.channels = 2;
aoAttr.soundCard.sampleRate = 32000;
aoAttr.soundCard.bitWidth = AUDIO_BIT_WIDTH_16;

aoAttr.enBitwidth = AUDIO_BIT_WIDTH_16;
aoAttr.enSamplerate = AUDIO_SAMPLE_RATE_8000;
aoAttr.enSoundmode = AUDIO_SOUND_MODE_MONO;

RK_MPI_AO_SetPubAttr(aoDevId, &aoAttr);
RK_MPI_AO_EnableReSmp(AoDev, AoChn, AUDIO_SAMPLE_RATE_8000);
```

3.56 RK_MPI_AO_DisableReSmp

【描述】

禁用 AO 重采样。

【语法】

```
RK_S32 RK_MPI_AO_DisableReSmp(AUDIO\_DEV AoDevId, AO\_CHN AoChn);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围: [0, AO_DEV_MAX_NUM)。	输入
AoChn	音频输出通道号。 取值范围: [0, AO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AUDIO错误码 。

【注意】

- 不再使用 AO 重采样功能的话, 应该调用此接口将其禁用。

3.57 RK_MPI_AO_PauseChn

【描述】

暂停 AO 通道。

【语法】

```
RK_S32 RK_MPI_AO_PauseChn(AUDIO\_DEV AoDevId, AO\_CHN AoChn);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围: [0, AO_DEV_MAX_NUM)。	输入
AoChn	音频输出通道号。 取值范围: [0, AO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AUDIO错误码 。

【注意】

- AO通道暂停后, 如果绑定的ADEC通道继续向此通道发送音频帧数据, 发送的音频帧数据将会被阻塞。而如果绑定的 AI 通道继续向此通道发送音频帧数据, 在通道缓冲未满的情况下则将音频帧放入缓冲区, 在满的情况下则将音频帧丢弃。
- AO通道为禁用状态时, 不允许调用此接口暂停AO通道。

3.58 RK_MPI_AO_ResumeChn

【描述】

启用 AO 重采样。

【语法】

```
RK_S32 RK_MPI_AO_ResumeChn(AUDIO\_DEV AoDevId, AO\_CHN AoChn);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入
AoChn	音频输出通道号。 取值范围：[0, AO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- AO 通道暂停后可以通过调用此接口重新恢复。
- AO 通道为暂停状态或使能状态下，调用此接口返回成功；否则调用将返回错误。

3.59 RK_MPI_AO_ClearChnBuf

【描述】

清除 AO 通道中当前的音频数据缓存。

【语法】

```
RK_S32 RK_MPI_AO_ClearChnBuf(AUDIO\_DEV AoDevId, AO\_CHN AoChn);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入
AoChn	音频输出通道号。 取值范围：[0, AO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 在 AO 通道成功启用后再调用此接口。
- 为完全清除解码回放通路上所有缓存数据，此接口还应该与[RK_MPI_ADEC_ClearChnBuf](#) 接口配合使用。

3.60 RK_MPI_AO_QueryChnStat

【描述】

查询 AO 通道中当前的音频数据缓存状态。

【语法】

```
RK_S32 RK_MPI_AO_QueryChnStat(AUDIO\_DEV AoDevId, AO\_CHN AoChn, AO\_CHN\_STATE\_S*pstStatus);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围: [0, AO_DEV_MAX_NUM)。	输入
AoChn	音频输出通道号。 取值范围: [0, AO_MAX_CHN_NUM)。	输入
pstStatus	缓存状态结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 在 AO 通道成功启用后再调用此接口。

3.61 RK_MPI_AO_SetTrackMode

【描述】

设置 AO 设备声道模式。

【语法】

```
RK_S32 RK_MPI_AO_SetTrackMode(AUDIO\_DEV AoDevId, AUDIO\_TRACK\_MODE\_E enTrackMode);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围: [0, AO_DEV_MAX_NUM)。	输入
enTrackMode	音频设备声道模式。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- [AUDIO_TRACK_MODE_E](#) 定义参见 rk_comm_aio.h。
- 在 AO 通道成功启用后再调用此接口。

3.62 RK_MPI_AO_GetTrackMode

【描述】

获取 AO 设备声道模式。

【语法】

```
RK_S32 RK_MPI_AO_GetTrackMode(AUDIO\_DEV AoDevId, AUDIO\_TRACK\_MODE\_E* penTrackMode);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围: [0, AO_DEV_MAX_NUM)。	输入
penTrackMode	音频设备声道模式指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AUDIO错误码 。

【注意】

- 在 AO 通道成功启用后再调用此接口。

3.63 RK_MPI_AO_SetVolume

【描述】

设置 AO 设备音量大小。

【语法】

```
RK_S32 RK_MPI_AO_SetVolume(AUDIO\_DEV AoDevId, RK_S32 s32VolumeDb);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围: [0, AO_DEV_MAX_NUM)。	输入
s32VolumeDb	音频设备音量大小。取值范围: [0,100]	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AUDIO错误码 。

【注意】

- 在 AO 通道成功启用后再调用此接口。

3.64 RK_MPI_AO_GetVolume

【描述】

获取 AO 设备音量大小。

【语法】

```
RK_S32 RK_MPI_AO_GetVolume([AUDIO_DEV](#AUDIO_DEV) AoDevId, RK_S32 *ps32VolumeDb);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围: [0, AO_DEV_MAX_NUM)。	输入
ps32VolumeDb	音频设备音量大小指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 在 AO 通道成功启用后再调用此接口。

3.65 RK_MPI_AO_SetChnParams

【描述】

设置 AO 通道参数。

【语法】

```
RK_S32 RK_MPI_AO_SetChnParams(AUDIO\_DEV AoDevId, AO\_CHN AoChn, const AO\_CHN\_PARAM\_S *pstParams);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围: [0, AO_DEV_MAX_NUM)。	输入
AoChn	音频输出通道号。 取值范围: [0, AO_MAX_CHN_NUM)。	输入
pstParams	通道参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 必须在 AO 通道启用前调用此接口。

【举例】

配置AO通道0数据输出到左声道，配置如下：

```
AUDIO_DEV aoDevId = 0;
AO_CHN aoChn0 = 0;
AIO_ATTR_S aoAttr;
memset(&aoAttr, 0, sizeof(AIO_ATTR_S));

aoAttr.soundCard.channels = 2;
aoAttr.soundCard.sampleRate = 48000;
aoAttr.soundCard.bitWidth = AUDIO_BIT_WIDTH_16;
aoAttr.enBitwidth = AUDIO_BIT_WIDTH_16;
aoAttr.enSamplerate = AUDIO_SAMPLE_RATE_8000;
aoAttr.enSoundmode = AUDIO_SOUND_MODE_MONO;
aoAttr.u32FrmNum = 4;
aoAttr.u32PtNumPerFrm = 1024;
aoAttr.u32EXFlag = 0;
aoAttr.u32ChnCnt = 2;

RK_MPI_AO_SetPubAttr(aoDevId, &aoAttr);
RK_MPI_AO_Enable(aoDevId);

AO_CHN_PARAM_S pstParams;
memset(&pstParams, 0, sizeof(AO_CHN_PARAM_S));
pstParams.enMode = AUDIO_CHN_MODE_LEFT;
RK_MPI_AO_SetChnParams(aoDevId, aoChn0, &pstParams);

RK_MPI_AO_EnableChn(aoDevId, aoChn0);
RK_MPI_AO_EnableReSmp(aoDevId, aoChn0, AUDIO_SAMPLE_RATE_8000);
```

3.66 RK_MPI_AO_GetChnParams

【描述】

获取 AO 通道参数。

【语法】

```
RK_S32 RK_MPI_AO_GetChnParams(AUDIO\_DEV AoDevId, AO\_CHN AoChn, AO\_CHN\_PARAM\_S *pstParams);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入
AoChn	音频输出通道号。 取值范围：[0, AO_MAX_CHN_NUM)。	输入
pstParams	通道参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

3.67 RK_MPI_AO_SetMute

【描述】

设置 AO 设备静音状态。

【语法】

```
RK_S32 RK_MPI_AO_SetMute(AUDIO\_DEV AoDevId, RK_BOOL bEnable, const AUDIO\_FADE\_S *pstFade);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入
bEnable	音频设备是否启用静音。RK_TRUE：启用静音功能；RK_FALSE：关闭静音功能。	输入
pstFade	淡入淡出结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 在 AO 通道成功启用后再调用此接口。
- 调用此接口时，用户可以选择是否使用淡入淡出功能，如果不使用淡入淡出则将结构体指针赋为空即可。

3.68 RK_MPI_AO_GetMute

【描述】

获取 AO 设备音量大小。

【语法】

```
RK_S32 RK_MPI_AO_GetVolume(AUDIO\_DEV AoDevId, RK_S32 *ps32VolumeDb);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入
pbEnable	音频设备静音状态指针。	输出
pstFade	淡入淡出结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 在 AO 通道成功启用后再调用此接口。

3.69 RK_MPI_AO_SaveFile

【描述】

开启音频输出保存文件功能。

【语法】

```
RK_S32 RK_MPI_AO_SaveFile(AUDIO\_DEV AoDevId, AO\_CHN AoChn, AUDIO\_SAVE\_FILE\_INFO\_S* pstSaveFileInfo);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围: [0, AO_DEV_MAX_NUM)。	输入
AoChn	音频输出通道号。 取值范围: [0, AO_MAX_CHN_NUM)。	输入
pstSaveFileInfo	音频保存文件属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 在 AO 通道成功启用后再调用此接口。

3.70 RK_MPI_AO_QueryFileStatus

【描述】

查询音频输出通道是否处于存文件的状态。

【语法】

```
RK_S32 RK_MPI_AO_QueryFileStatus(AUDIO\_DEV AoDevId, AO\_CHN AoChn, AUDIO\_FILE\_STATUS\_S* pstFileStatus);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。取值范围: [0, AO_DEV_MAX_NUM)。	输入
AoChn	音频输出通道号。 取值范围: [0, AO_MAX_CHN_NUM)。	输入
pstFileStatus	状态属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

此接口用于查询音频输出通道是否处于存文件的状态。

3.71 RK_MPI_AO_ClrPubAttr

【描述】

清除 AO 设备属性。

【语法】

```
RK_S32 RK_MPI_AO_ClrPubAttr(AUDIO\_DEV AoDevId);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 清除设备属性前，需要先停止设备。

3.72 RK_MPI_AO_WaitEos

【描述】

等待指定设备和通道播放完成。

【语法】

```
RK_S32 RK_MPI_AO_WaitEos(AUDIO\_DEV AoDevId, AO\_CHN AoChn, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
AoDevId	音频设备号。 取值范围：[0, AO_DEV_MAX_NUM)。	输入
AoChn	音频输出通道号。 取值范围：[0, AO_MAX_CHN_NUM)。	输入
s32MilliSec	等待的超时时间，-1表示阻塞模式；>=0表示非阻塞模式的超时时间(毫秒)	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AUDIO错误码 。

【注意】

- 在 AO 通道成功启用后再调用此接口。
- 当在ADEC绑定AO之后，建议调用此接口使用阻塞模式，等待播放完成。
- s32MilliSec 的值等于-1时表示阻塞模式等待播放完成；大于等于 0，表示等待s32MilliSec 毫秒后，查询播放结束的结果，如果超过等待时间内没有等到播放完毕，则返回超时并报错。

3.73 音频解码

该功能模块为用户提供以下API:

- [RK_MPI_ADEC_CreateChn](#): 创建音频解码通道。
- [RK_MPI_ADEC_DestroyChn](#): 销毁音频解码通道。
- [RK_MPI_ADEC_SendStream](#): 发送音频码流到音频解码通道。
- [RK_MPI_ADEC_ClearChnBuf](#): 清除ADEC通道中当前的音频数据缓存。
- [RK_MPI_ADEC_GetFrame](#): 获取音频解码帧数据。
- [RK_MPI_ADEC_ReleaseFrame](#): 释放音频解码帧数据。
- [RK_MPI_ADEC_SendEndOfStream](#): 向解码器发送码流结束标识符。
- [RK_MPI_ADEC_QueryChnStat](#): 查询ADEC通道中当前的音频数据缓存状态。
- [RK_MPI_ADEC_RegisterDecoder](#): 注册解码器
- [RK_MPI_ADEC_UnRegisterDecoder](#): 注销解码器。

3.74 RK_MPI_ADEC_CreateChn

【描述】

创建音频解码通道。

【语法】

```
RK_S32 RK_MPI_ADEC_CreateChn (ADEC\_CHN AdChn, const ADEC\_CHN\_ATTR\_S *pstAttr);
```

【参数】

参数名	描述	输入/输出
AdChn	通道号。 取值范围: [0, ADEC_MAX_CHN_NUM)。	输入
pstAttr	通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 音频解码支持的解码协议 : RK_AUDIO_ID_PCM_ALAW、RK_AUDIO_ID_PCM_MULAW、RK_AUDIO_ID_ADPCM_G722、RK_AUDIO_ID_ADPCM_G726等, 参见rk_common.h中RK_CODEC_ID_E枚举音频定义。
- g726音频格式需要设置码流码字 (codecwords) , 支持的协议说明见表[音频编解码协议](#)。
- 音频解码的初始化属性必须设置码流的采样率 (u32SampleRate) 、声道数 (u32Channels) 、codec id (enType) 三个参数。解码模式默认为PACK模式, 支持STREAM模式, 但是建议使用PACK模式。
- 在通道闲置时才能使用此接口, 如果通道已经被创建, 则返回通道已经创建的错误。

3.75 RK_MPI_ADEC_DestroyChn

【描述】

销毁音频解码通道。

【语法】

```
RK_S32 RK_MPI_ADEC_DestroyChn(ADEC\_CHN AdChn);
```

【参数】

参数名	描述	输入/输出
AdChn	通道号。 取值范围: [0, ADEC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 通道未创建的情况下调用此接口会返回RK_ERR_ADEC_UNEXIST。
- 建议通道使用完成后调用此接口。

3.76 RK_MPI_ADEC_SendStream

【描述】

向音频解码通道发送码流。

【语法】

```
RK_S32 RK_MPI_ADEC_SendStream (ADEC\_CHN AdChn, const AUDIO\_STREAM\_S *pstStream, RK_BOOL bBlock);
```

【参数】

参数名	描述	输入/输出
AdChn	通道号。 取值范围: [0, ADEC_MAX_CHN_NUM)。	输入
pstStream	音频码流。	输入
bBlock	阻塞标识。RK_TRUE : 阻塞。 RK_FALSE : 非阻塞。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 创建解码通道时可以指定解码方式为pack方式或者stream方式:
 - pack方式用于确定码流包为一帧的情况下, 比如从网络直接获取的码流, 从文件读取确切知道一帧边界的码流, 效率较高。
 - stream方式用于不确定码流包为一帧的情况下, 效率较低, 且可能会有延迟。
- 发送数据时必须保证通道已经被创建, 否则直接返回失败。
- 支持阻塞或者非阻塞方式发送码流。
- 当阻塞方式发送码流时, 如果用于缓存解码后的音频帧的buffer满, 则此接口调用会被阻塞, 直到解码后的音频帧数据被取走, 或ADEC通道被销毁, 建议选择阻塞方式发送码流。
- 确保发送给ADEC通道的码流数据的正确性, 否则可能引起解码器异常退出。
- 音频申请内存的方式建议尽量使用malloc方式申请。

3.77 RK_MPI_ADEC_GetFrame

【描述】

获取音频解码帧数据。

【语法】

```
RK_S32 RK_MPI_ADEC_GetFrame (ADEC\_CHN AdChn, AUDIO\_FRAME\_INFO\_S *pstFrmInfo, RK_BOOL bBlock);
```

【参数】

参数名	描述	输入/输出
AdChn	通道号。 取值范围: [0, ADEC_MAX_CHN_NUM)。	输入
pstFrmInfo	音频帧数据结构体。	输出
bBlock	是否以阻塞方式获取	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 必须在ADEC通道创建之后调用。
- 使用本接口获取音频帧数据时, 建议发送码流时按pack方式(帧模式)发送。
- 使用本接口获取音频帧数据时, 如果发送码流按stream发送, 请务必保证获取解码帧数据的及时性; 如果SendStream和GetFrame接口都是阻塞的, 需要各自在不同的线程, 不然会阻塞住。
- GetFrame成功后, 需要调用ReleaseFrame。
- 使用本接口获取音频数据时, ADEC BIND AO的方式不需要用此接口, 请先解除ADEC与AO的绑定关系, 否则获取到的帧是不连续的。

3.78 RK_MPI_ADEC_ReleaseFrame

【描述】

释放获取到的音频解码帧数据。

【语法】

```
RK_S32 RK_MPI_ADEC_ReleaseFrame (ADEC\_CHN AdChn, AUDIO\_FRAME\_INFO\_S *pstFrmInfo);
```

【参数】

参数名	描述	输入/输出
AdChn	通道号。 取值范围: [0, ADEC_MAX_CHN_NUM)。	输入
pstFrmInfo	音频帧数据结构。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 必须在ADEC通道创建之后调用。
- 本接口必须与接口RK_MPI_ADEC_GetFrame配合使用。

3.79 RK_MPI_ADEC_SendEndOfStream

【描述】

向解码器发送码流结束标识符。

【语法】

```
RK_S32 RK_MPI_ADEC_SendEndOfStream (ADEC\_CHN AdChn, RK_BOOL bInstant);
```

【参数】

参数名	描述	输入/输出
AdChn	通道号。 取值范围: [0, ADEC_MAX_CHN_NUM)。	输入
bInstant	是否立刻清除解码器内部的缓存数据。 取值范围: RK_FALSE: 不清除缓存数据, 解码继续进行, 直到解码完音频码 RK_TRUE: 立刻清除解码器内部的缓存数据。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 建议bInstant使用RK_FALSE。

3.80 RK_MPI_ADEC_QueryChnStat

【描述】

查询ADEC通道中当前的音频数据缓存状态。

【语法】

```
RK_S32 RK_MPI_ADEC_QueryChnStat (ADEC\_CHN AdChn, ADEC\_CHN\_STATE\_S *pstBufferStatus);
```

【参数】

参数名	描述	输入/输出
AdChn	通道号。 取值范围: [0, ADEC_MAX_CHN_NUM)。	输入
pstBufferStatus	缓存状态结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 在ADEC通道成功启用后再调用此接口。

3.81 RK_MPI_ADEC_RegisterDecoder

【描述】

注册音频解码器。

【语法】

```
RK_S32 RK_MPI_ADEC_RegisterDecoder(RK_S32 ps32Handle, const ADEC\_DECODER\_S*pstDecoder);
```

【参数】

参数名	描述	输入/输出
ps32Handle	注册句柄。	输出
pstDecoder	解码器属性结构体。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 客户使用该接口注册并使用解码器时，必须先从版权权利人处获取注册解码器授权，并缴纳Licensing Fee。
- 在ADEC通道创建前注册音频解码器。
- 用户通过传入解码器属性结构体，向ADEC模块注册一个解码器，并返回注册句柄，用户通过注册句柄来注销该解码器。
- ADEC模块外部最大可注册20个解码器，且自身已注册LPCM、G711a、G711u、G726、ADPCM五个解码器。
- 不允许外部重复注册同一种解码器，例如假如外部已注册G711a解码器，不允许另外再注册一个G711a解码器。
- 允许外部注册ADEC内部已注册的解码器，例如ADEC模块自身包含了G711a解码器，但允许外部再次注册G711a解码器，且优先使用外部注册的解码器。
- 示例请见[附录A：注册解码/编码器API说明](#)

3.82 RK_MPI_ADEC_UnRegisterDecoder

【描述】

注销解码器。

【语法】

```
RK_S32 RK_MPI_ADEC_UnRegisterDecoder(RK_S32 s32Handle);
```

【参数】

参数名	描述	输入/输出
s32Handle	注册句柄（注册解码器时获得的句柄）。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 当多个ADEC通道使用同一注册的解码器，只有使用该解码器的通道全部销毁时，才能注销解码器。

3.83 音频编码

该功能模块为用户提供以下API：

- [RK_MPI_AENC_CreateChn](#)：创建音频编码通道。
- [RK_MPI_AENC_DestroyChn](#)：销毁音频编码通道。
- [RK_MPI_AENC_SendFrame](#)：发送音频编码音频帧。
- [RK_MPI_AENC_GetStream](#)：获取音频编码码流。
- [RK_MPI_AENC_ReleaseStream](#)：释放音频编码码流。
- [RK_MPI_AENC_SaveFile](#)：开启音频编码之前通道存文件功能。
- [RK_MPI_AENC_QueryFileStatus](#)：查询音频编码通道是否处于存文件的状态。
- [RK_MPI_AENC_RegisterEncoder](#)：注册编码器
- [RK_MPI_AENC_UnRegisterEncoder](#)：注销编码器

3.84 RK_MPI_AENC_CreateChn

【描述】

创建音频解码通道。

【语法】

```
RK_S32 RK_MPI_AENC_CreateChn(AENC\_CHN AeChn, const AENC\_CHN\_ATTR\_S *pstAttr);
```

【参数】

参数名	描述	输入/输出
AeChn	通道号。 取值范围：[0, AENC_MAX_CHN_NUM]。	输入
pstAttr	通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 AUDIO错误码 。

【注意】

- 音频编码支持的解码协议：RK_AUDIO_ID_ADPCM_G722、RK_AUDIO_ID_ADPCM_G726、RK_AUDIO_ID_PCM_MULAW、RK_AUDIO_ID_PCM_ALAW等，参见rk_common.h中RK_CODEC_ID_E枚举音频定义。
- 音频编码的初始化属性必须设置码流的采样率（u32SampleRate）、声道数（u32Channels）、采样精度（enBitwidth）、codec id（enType）参数。
- 在通道闲置时才能使用此接口，如果通道已经被创建，则返回通道已经创建的错误。

3.85 RK_MPI_AENC_DestroyChn

【描述】

销毁音频编码通道。

【语法】

```
RK_S32 RK_MPI_AENC_DestroyChn(AENC\_CHN AeChn);
```

【参数】

参数名	描述	输入/输出
AeChn	通道号。 取值范围: [0, AENC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 通道未创建的情况下调用此接口会返回RK_ERR_AENC_UNEXIST。
- 建议通道使用完成后调用此接口。

3.86 RK_MPI_AENC_SendFrame

【描述】

发送音频编码音频帧。

【语法】

```
RK_S32 RK_MPI_AENC_SendFrame(AENC\_CHN AeChn, const AUDIO\_FRAME\_S *pstFrm, const AEC\_FRAME\_S *pstAecFrm,
RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
AeChn	通道号。 取值范围: [0, AENC_MAX_CHN_NUM)。	输入
pstFrm	音频帧结构体指针。	输入
pstAecFrm	回声抵消参考帧结构体指针。	输入
s32MilliSec	发送数据的超时时间: -1表示阻塞模式; >=0表示非阻塞模式的超时时间(毫秒)	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 目前pstAecFrm未使用, 置为NULL。
- s32MilliSec等于-1时, 表示采用阻塞模式发送数据, 该接口会一直等待直到成功发送数据; 大于等于0时表示非阻塞模式发送数据。s32MilliSec表示超时时间, 该接口在超时时间s32MilliSec(毫秒)内, 成功发送数据则返回成功, 如超过设定的时间, 没有成功发送数据则报错。
- 该接口用于用户主动发送音频帧进行编码, 如果 AENC 通道已经通过系统绑定(RK_MPI_SYS_Bind)接口与 AI 绑定, 不需要也不建议调此接口。
- 调用该接口发送音频编码音频帧时, 必须先创建对应的编码通道。

3.87 RK_MPI_AENC_GetStream

【描述】

获取编码后码流。

【语法】

```
RK_S32 RK_MPI_AENC_GetStream(AENC\_CHN AeChn, AUDIO\_STREAM\_S *pstStream, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
AeChn	通道号。 取值范围：[0, AENC_MAX_CHN_NUM)。	输入
pstStream	音频帧数据结构体。	输出
s32MilliSec	获取数据的超时时间： -1表示阻塞模式； >=0表示非阻塞模式超时时间	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 AUDIO错误码 。

【注意】

- 必须创建通道后才能通过该接口获取码流。
- s32MilliSec等于-1时，表示采用阻塞模式获取数据，该接口会一直等待直到成功获取到数据；大于等于0时表示非阻塞模式获取数据。该接口在超时时间s32MilliSec(毫秒)内，如获取到数据则返回成功，如超过设定的时间，没有获取到数据则报错。

3.88 RK_MPI_AENC_ReleaseStream

【描述】

释放从音频编码通道获取的码流。

【语法】

```
RK_S32 RK_MPI_AENC_ReleaseStream(AENC\_CHN AeChn, const AUDIO\_STREAM\_S *pstStream);
```

【参数】

参数名	描述	输入/输出
AeChn	通道号。 取值范围：[0, AENC_MAX_CHN_NUM)。	输入
pstStream	获取的码流指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 AUDIO错误码 。

【注意】

- 码流最好能够在使用完之后立即释放，如果不及时释放，会导致编码过程阻塞。
- 释放的码流必须是从该通道获取的码流，不得对码流信息结构体进行任何修改，否则会导致码流不能释放，使此码流 buffer 丢失，甚至导致程序异常。
- 释放码流时必须保证通道已经被创建，否则直接返回失败，如果在释放码流过程中销毁通道则会立刻返回失败。

3.89 RK_MPI_AENC_SaveFile

【描述】

开启音频编码之前通道存文件功能。

【语法】

```
RK_S32 RK_MPI_AENC_SaveFile(AENC\_CHN AeChn, const AUDIO\_SAVE\_FILE\_INFO\_S* pstSaveFileInfo);
```

【参数】

参数名	描述	输入/输出
AeChn	通道号。 取值范围：[0, AENC_MAX_CHN_NUM)。	输入
pstSaveFileInfo	音频保存文件属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 AUDIO错误码 。

【注意】

- 在 AENC通道成功启用后再调用此接口。

3.90 RK_MPI_AENC_QueryFileStatus

【描述】

查询频编码通道是否处于存文件的状态。

【语法】

```
RK_S32 RK_MPI_AENC_QueryFileStatus(AENC\_CHN AeChn, AUDIO\_FILE\_STATUS\_S* pstFileStatus);
```

【参数】

参数名	描述	输入/输出
AeChn	通道号。 取值范围：[0, AENC_MAX_CHN_NUM)。	输入
pstFileStatus	状态属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，请参见 AUDIO错误码 。

【注意】

- 此接口用于查询音频输出通道是否处于存文件的状态。

3.91 RK_MPI_AENC_RegisterEncoder

【描述】

注册编码器。

【语法】

```
RK_S32 RK_MPI_AENC_RegisterEncoder(RK_S32 ps32Handle, const AENC\_ENCODER\_S*pstEncoder);
```

【参数】

参数名	描述	输入/输出
ps32Handle	注册句柄。	输出
pstEncoder	编码器属性结构体。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 客户使用该接口注册并使用编码器时，必须先从版权权利人处获取注册编码器授权，并缴纳Licensing Fee。
- 在AENC通道创建前注册音频编码器。
- 用户通过传入编码器属性结构体，向AENC模块注册一个编码器，并返回注册句柄，用户通过注册句柄来注销该编码器。
- AENC模块外部最大可注册20个编码器，且自身已注册G711a、G711u、G722、G7264个编码器。
- 不允许外部重复注册同一种编码器，例如假如外部已注册G711a编码器，不允许另外再注册一个G711a编码器。
- 允许外部注册AENC内部已注册的编码器，例如AENC模块自身包含了G711a编码器，但允许外部再次注册G711a编码器，且优先使用外部注册的编码器。
- 示例请见[附录A：注册解码/编码器API说明](#)

3.92 RK_MPI_AENC_UnRegisterEncoder

【描述】

注销编码器。

【语法】

```
RK_S32 RK_MPI_AENC_UnRegisterEncoder(RK_S32 s32Handle);
```

【参数】

参数名	描述	输入/输出
ps32Handle	注册句柄（注册编码器时获得的句柄）。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 请参见 AUDIO错误码 。

【注意】

- 当多个AENC通道使用同一注册的编码器，只有使用该编码器的通道全部销毁时，才能注销编码器。

4. 数据类型

4.1 音频输入输出

音频输入输出相关数据类型、数据结构定义如下：

- **AUDIO_DEV**: 定义 AO 设备句柄。
- **AO_CHN**: 定义 AO 通道。
- **AO_MAX_CHN_NUM**: 定义音频输出通道的最大个数。
- **AO_DEV_MAX_NUM**: 定义音频输出设备的最大个数。
- **AO_CHN_STATE_S**: 音频输出通道的数据缓存状态结构体。
- **AI_CHN**: 定义 AI 通道。
- **AI_MAX_CHN_NUM**: 定义音频输入通道的最大个数。
- **AI_DEV_MAX_NUM**: 定义音频输入设备的最大个数。
- **AI_CHN_PARAM_S**: 定义通道参数结构体。
- **MAX_AUDIO_FILE_PATH_LEN**: 音频保存文件的路径的最大长度限制。
- **MAX_AUDIO_FILE_NAME_LEN**: 音频保存文件的名称的最大长度限制。
- **AIO_ATTR_S**: 定义音频输入输出设备属性结构体。
- **AUDIO_FRAME_S**: 定义音频帧数据结构体。
- **AUDIO_SOUND_MODE_E**: 定义音频声道模式。
- **AUDIO_BIT_WIDTH_E**: 定义音频采样精度。
- **AIO_SOUND_CARD**: 定义打开声卡参数。
- **AUDIO_STREAM_S**: 定义音频码流结构体。
- **AUDIO_SAMPLE_RATE_E**: 定义音频采样率。
- **AUDIO_TRACK_MODE_E**: 音频设备声道模式类型。
- **AUDIO_FADE_RATE_E**: 定义音频输出设备淡入淡出速度类型。
- **AUDIO_FADE_S**: 音频输出设备淡入淡出配置结构体。
- **AUDIO_CHN_MODE_E**: 音频通道模式类型。
- **AO_CHN_PARAM_S**: 音频输出通道参数结构体。
- **AUDIO_SAVE_FILE_INFO_S**: 定义音频保存文件功能配置信息结构体。
- **AUDIO_FILE_STATUS_S**: 定义音频文件保存状态结构体。
- **AIO_VQE_CONFIG_METHOD**: 定义VQE模块配置方法结构体。
- **AI_VQE_CONFIG_S**: 定义VQE模块配置/结构体。
- **AI_VQE_MOD_ENABLE_S**: 定义VQE模块使能结构体。
- **AI_AED_CONFIG_S**: 定义AED属性结构体。
- **AI_AED_RESULT_S**: 定义AED事件检测结果结构体。
- **AI_BCD_CONFIG_S**: 定义BCD属性结构体。
- **AI_BCD_RESULT_S**: 定义BCD事件检测结果结构体。
- **AI_BUZ_CONFIG_S**: 定义BUZ属性结构体。
- **AI_BUZ_RESULT_S**: 定义BUZ事件检测结果结构体。
- **AI_GBS_CONFIG_S**: 定义GBS属性结构体。
- **AI_GBS_RESULT_S**: 定义GBS事件检测结果结构体。

4.2 AUDIO_DEV

【说明】

定义 AO 设备句柄。

【定义】

```
typedef RK_S32 AUDIO_DEV;
```

4.3 AO_CHN

【说明】

定义 AO 通道。

【定义】

```
typedef RK_S32 AO_CHN;
```

4.4 AO_MAX_CHN_NUM

【说明】

定义音频输出通道的最大个数。

【定义】

```
#define AO_MAX_CHN_NUM 3
```

4.5 AO_DEV_MAX_NUM

【说明】

定义音频输出设备的最大个数。

【定义】

```
#define AO_DEV_MAX_NUM 2
```

4.6 AO_CHN_STATE_S

【说明】

音频输出通道的数据缓存状态结构体。

【定义】

```
typedef struct rkAO_CHN_STATE_S {
    RK_U32 u32ChnTotalNum; /* total number of channel buffer */
    RK_U32 u32ChnFreeNum; /* free number of channel buffer */
    RK_U32 u32ChnBusyNum; /* busy number of channel buffer */
} AO_CHN_STATE_S;
```

【成员】

成员名称	描述
u32ChnTotalNum	输出通道总的缓存块数。
u32ChnFreeNum	可用的空闲缓存块数。
u32ChnBusyNum	被占用缓存块数。

4.7 AI_CHN

【说明】

定义 AI 通道。

【定义】

```
typedef RK_S32 AI_CHN;
```

4.8 AI_MAX_CHN_NUM

【说明】

定义音频输入通道的最大个数。

【定义】

```
#define AI_MAX_CHN_NUM 1
```

4.9 AI_DEV_MAX_NUM

【说明】

定义音频输入设备的最大个数。

【定义】

```
#define AI_DEV_MAX_NUM 1
```

4.10 AI_CHN_PARAM_S

【说明】

定义通道参数结构体。

【定义】

```
typedef struct rkAI_CHN_PARAM_S {
    RK_S32 s32UsrFrmDepth;
    RK_S32 s32UsrFrmCount;
    AUDIO_LOOPBACK_MODE_E enLoopbackMode;
    RK_U32 u32MapPtNumPerFrm;
    AUDIO_SAMPLE_RATE_E enSamplerate;
    RK_S32 s32SedQueLen;
} AI_CHN_PARAM_S;
```

【成员】

成员名称	描述
s32UsrFrmDepth	音频帧缓存深度。
s32UsrFrmCount	未使用参数，设置无效。
enLoopbackMode	未使用参数，设置无效。
u32MapPtNumPerFrm	未使用参数，设置无效。
enSamplerate	未使用参数，设置无效。
s32SedQueLen	SED模块线程队列的大小，默认为3。

4.11 MAX_AUDIO_FILE_PATH_LEN

【说明】

音频保存文件的路径的最大长度限制。

【定义】

```
#define MAX_AUDIO_FILE_PATH_LEN 256
```

4.12 MAX_AUDIO_FILE_NAME_LEN

【说明】

音频保存文件的名称的最大长度限制。

【定义】

```
#define MAX_AUDIO_FILE_NAME_LEN 256
```

4.13 AIO_ATTR_S

【说明】

定义音频输入输出设备属性结构体。

【定义】

```
typedef struct rkAIO_ATTR_S {
    // params of sound card
    AIO_SOUND_CARD    soundCard;
    // sample rate
    AUDIO_SAMPLE_RATE_E enSamplerate;
    // bitwidth
    AUDIO_BIT_WIDTH_E  enBitwidth;
    // momo or steror
    AUDIO_SOUND_MODE_E enSoundmode;
    /* expand 8bit to 16bit,use AI_EXPAND(only valid for AI 8bit),
     * use AI_CUT(only valid for extern Codec for 24bit)
     */
    RK_U32      u32EXFlag;
    /* frame num in buf[2,MAX_AUDIO_FRAME_NUM] */
    RK_U32      u32FrmNum;
    /*
     * point num per frame (80/160/240/320/480/1024/2048)
     * (ADPCM IMA should add 1 point, AMR only support 160)
     */
    RK_U32      u32PtNumPerFrm;
    RK_U32      u32ChnCnt; /* channle number on FS, valid value:1/2/4/8 */
    /*
     * name of sound card, if it is setted, we will
     * using it to open sound card, otherwise, use
     * the index of device to open sound card
     */
    RK_U8       u8CardName[64];
    RK_U8       u8MapOutChns[AI_MAX_CHN_NUM];
    RK_U8       u8MapChns[AI_MAX_CHN_NUM][MAX_SOUND_CARD_CHANNEL];
    RK_BOOL     bMultichnFlag;
    RK_S32     s32DevQueLen;
} AIO_ATTR_S;
```

【成员】

成员名称	描述
soundCard	打开声卡时配置的参数。
enSamplerate	音频采样率。
enBitwidth	音频采样精度。
enSoundmode	音频声道模式。
u32EXFlag	设置默认为0。
u32FrmNum	缓冲个数。
u32PtNumPerFrm	ai: 表示应用取帧的长度 (byte) 。 ao: 表示应用送帧的长度 (byte) 。
u32ChnCnt	支持的声音数目。
u8CardName[64]	声卡名字。
u8MapOutChns[AI_MAX_CHN_NUM]	每个音频通道中声卡输出声道个数。
u8MapChns[AI_MAX_CHN_NUM] [MAX_SOUND_CARD_CHANNEL]	每个音频通道中声卡输出声道对应的每个声道映射到实际声卡的对应声道的编号。
bMultichnFlag	音频通道不为1时的标志位。
s32DevQueLen	设备驱动线程的队列大小， 默认为4。

【注意事项】

- soundCard: 配置打开声卡的参数。需要设置声卡驱动支持的采样率，声道数和采样精度。必须配置。
- enSamplerate: 对于AI，表示用户获取数据的采样率，比如打开声卡16K，需要读取8K的数据，此时配置enSamplerate为8K，需要调用RK_MPI_AI_EnableReSmp使能重采样，把16k数据转换成8k读取。对于AO，表示发送数据的采样率，比如用户发送16k采样率的数据，设置enSamplerate为16k，此时声卡只支持打开48k，需要调用RK_MPI_AI_EnableReSmp使能重采样，把16k数据转换成48k送去播放。必须配置。
- enBitwidth: 原理同enSamplerate，设置数据的采样精度。必须配置。
- enSoundmode: 原理同enSamplerate，设置数据的声道数。必须配置。
- u32FrmNum: 每次DMA运输处理音频数据的帧数。如果周期大小设定得较大，则单次处理的数据较多，这意味着单位时间间隔内硬件中断的次数较少，CPU也就有更多时间处理其他任务，功耗也更低，但这样也带来一个显著的弊端——数据处理的时延会增大。
- u8CardName: 外部可配需要打开的声卡，代码内部有默认配置。
- u8MapOutChns: 使用时，需要将bMultichnFlag置为RK_TRUE。
- u8MapChns: 使用时，需要将bMultichnFlag置为RK_TRUE。使用举例：u8MapOutChns[3]为8，则表示第四个通道输出的声道数为8，那么u8MapChns[3][0]-u8MapChns[3][7]分别代表8个声道对应的实际声卡声道映射，u8MapChns[3][0] = 0表示第四个通道输出的第一个声道为实际声卡的第一个声道，而u8MapChns[3][1] = 3表示第四个通道输出的第二个声道为实际声卡的第四个声道，以此类推。

4.14 AUDIO_FRAME_S

【说明】

定义音频帧数据结构体。

【定义】

```

typedef struct rkAUDIO_FRAME_S {
    MB_BLK      pMbBlk;
    AUDIO_BIT_WIDTH_E enBitWidth; /*audio frame bitwidth*/
    AUDIO_SOUND_MODE_E enSoundMode; /*audio frame momo or stereo mode*/
    RK_U64      u64TimeStamp; /*audio frame timestamp*/
    RK_U32      u32Seq;     /*audio frame seq*/
    RK_U32      u32Len;
    /*data length per channel in frame, u32Len <= 0 mean eos */
    RK_BOOL      bBypassMbBlk;
    /* FALSE: copy, TRUE: MbBlock owned by internal */
    RK_S32      s32SampleRate; /*audio frame sampleRate*/
} AUDIO_FRAME_S;

```

【成员】

成员名称	描述
pMbBlk	缓存块句柄。
enBitWidth	音频采样精度。
enSoundMode	音频声道模式。
u64TimeStamp	音频帧时间戳。以 μs 为单位。
u32Seq	音频帧序号。
u32Len	音频帧长度。以 byte 为单位。
bBypassMbBlk	否需要拷贝外部定义的MB_BLK到内部。FALSE: 需要拷贝，TRUE: 不需要拷贝
s32SampleRate	音频采样率

【注意事项】

- AUDIO_FRAME_S的数据存储在pMbBlk中，输入输出均需要外部申请合理合法的缓存块。

4.15 AUDIO_SOUND_MODE_E

【说明】

定义音频声道模式。

【定义】

```

typedef enum rkAIO_SOUND_MODE_E {
    AUDIO_SOUND_MODE_MONO = 0,/*mono*/
    AUDIO_SOUND_MODE_STEREO = 1,/*stereo*/
    AUDIO_SOUND_MODE_BUTT
} AUDIO_SOUND_MODE_E;

```

【成员】

成员名称	描述
AUDIO_SOUND_MODE_MONO	单声道。
AUDIO_SOUND_MODE_STEREO	双声道。

4.16 AUDIO_BIT_WIDTH_E

【说明】

定义音频采样精度。

【定义】

```
typedef enum rkAUDIO_BIT_WIDTH_E {
    AUDIO_BIT_WIDTH_8 = 0, /* 8bit width */
    AUDIO_BIT_WIDTH_16 = 1, /* 16bit width*/
    AUDIO_BIT_WIDTH_24 = 2, /* 24bit width*/
    AUDIO_BIT_WIDTH_BUTT,
} AUDIO_BIT_WIDTH_E;
```

【成员】

成员名称	描述
AUDIO_BIT_WIDTH_8	采样精度为8bit位宽。
AUDIO_BIT_WIDTH_16	采样精度为16bit位宽。
AUDIO_BIT_WIDTH_24	采样精度为24bit位宽。

4.17 AIO_SOUND_CARD

【说明】

定义音频采样精度。

【定义】

```
typedef struct rkAIO_SOUND_CARD {
    RK_U32 channels;
    RK_U32 sampleRate;
    AUDIO_BIT_WIDTH_E bitWidth;
} AIO_SOUND_CARD;
```

【成员】

成员名称	描述
channels	打开声卡声道数。
sampleRate	打开声卡采样率。
bitWidth	打开声卡采样位深。

4.18 AUDIO_STREAM_S

【说明】

定义音频码流结构体。

【定义】

```
typedef struct rkAUDIO_STREAM_S {
    MB_BLK pMbBlk;
    RK_U32 u32Len; /* stream length, by bytes */
    RK_U64 u64TimeStamp; /* frame time stamp */
    RK_U32 u32Seq; /* frame seq, if stream is not a valid frame, u32Seq is 0 */
    RK_BOOL bBypassMbBlk; /* FALSE: copy, TRUE: MbBlock owned by internal */
} AUDIO_STREAM_S;
```

【成员】

成员名称	描述
pMbBlk	缓存块句柄。
u32Len	音频码流长度。
u32Seq	音频码流序号。
u64TimeStamp	音频码流时间戳。
bBypassMbBlk	是否需要拷贝外部定义的MB_BLK到内部。FALSE: 需要拷贝, TRUE: 不需要拷贝

4.19 AUDIO_SAMPLE_RATE_E

【说明】

定义音频采样率。

【定义】

```
typedef enum rkAUDIO_SAMPLE_RATE_E {
    AUDIO_SAMPLE_RATE_DISABLE = 0,
    AUDIO_SAMPLE_RATE_8000 = 8000, /* 8K samplerate*/
    AUDIO_SAMPLE_RATE_12000 = 12000, /* 12K samplerate*/
    AUDIO_SAMPLE_RATE_11025 = 11025, /* 11.025K samplerate*/
    AUDIO_SAMPLE_RATE_16000 = 16000, /* 16K samplerate*/
    AUDIO_SAMPLE_RATE_22050 = 22050, /* 22.050K samplerate*/
    AUDIO_SAMPLE_RATE_24000 = 24000, /* 24K samplerate*/
    AUDIO_SAMPLE_RATE_32000 = 32000, /* 32K samplerate*/
    AUDIO_SAMPLE_RATE_44100 = 44100, /* 44.1K samplerate*/
    AUDIO_SAMPLE_RATE_48000 = 48000, /* 48K samplerate*/
    AUDIO_SAMPLE_RATE_64000 = 64000, /* 64K samplerate*/
    AUDIO_SAMPLE_RATE_96000 = 96000, /* 96K samplerate*/
    AUDIO_SAMPLE_RATE_BUTT,
} AUDIO_SAMPLE_RATE_E;
```

4.20 AUDIO_TRACK_MODE_E

【说明】

音频设备声道模式类型。

【定义】

```
typedef enum rkAUDIO_TRACK_MODE_E {
    AUDIO_TRACK_NORMAL = 0,
    AUDIO_TRACK_BOTH_LEFT = 1,
    AUDIO_TRACK_BOTH_RIGHT = 2,
    AUDIO_TRACK_EXCHANGE = 3,
    AUDIO_TRACK_MIX = 4,
    AUDIO_TRACK_LEFT_MUTE = 5,
    AUDIO_TRACK_RIGHT_MUTE = 6,
    AUDIO_TRACK_BOTH_MUTE = 7,
    AUDIO_TRACK_BUTT
} AUDIO_TRACK_MODE_E;
```

【成员】

成员名称	描述
AUDIO_TRACK_NORMAL	正常模式(默认), 不做处理。
AUDIO_TRACK_BOTH_LEFT	两个声道全部为左声道声音。
AUDIO_TRACK_BOTH_RIGHT	两个声道全部为右声道声音。
AUDIO_TRACK_EXCHANGE	左右声道数据互换, 左声道为右声道声音, 右声道为左声道声音。
AUDIO_TRACK_MIX	左右两个声道输出为左右声道相加 (混音)。
AUDIO_TRACK_LEFT_MUTE	左声道静音, 右声道播放原右声道声音。
AUDIO_TRACK_RIGHT_MUTE	右声道静音, 左声道播放原左声道声音。
AUDIO_TRACK_BOTH_MUTE	左右声道均静音。

4.21 AUDIO_FADE_RATE_E

【说明】

定义音频输出设备淡入淡出速度类型。

【定义】

```
typedef enum rkAUDIO_FADE_RATE_E {
    AUDIO_FADE_RATE_1 = 0,
    AUDIO_FADE_RATE_2 = 1,
    AUDIO_FADE_RATE_4 = 2,
    AUDIO_FADE_RATE_8 = 3,
    AUDIO_FADE_RATE_16 = 4,
    AUDIO_FADE_RATE_32 = 5,
    AUDIO_FADE_RATE_64 = 6,
    AUDIO_FADE_RATE_128 = 7,
    AUDIO_FADE_RATE_BUTT
} AUDIO_FADE_RATE_E;
```

【成员】

成员名称	描述
AUDIO_FADE_RATE_1	1个采样点改变一次。
AUDIO_FADE_RATE_2	2个采样点改变一次。
AUDIO_FADE_RATE_4	4个采样点改变一次。
AUDIO_FADE_RATE_8	8个采样点改变一次。
AUDIO_FADE_RATE_16	16个采样点改变一次。
AUDIO_FADE_RATE_32	32个采样点改变一次。
AUDIO_FADE_RATE_64	64个采样点改变一次。
AUDIO_FADE_RATE_128	128个采样点改变一次。

4.22 AUDIO_FADE_S

【说明】

音频输出设备淡入淡出配置结构体。

【定义】

```
typedef struct rkAUDIO_FADE_S {  
    RK_BOOL    bFade;  
    AUDIO_FADE_RATE_E enFadeInRate;  
    AUDIO_FADE_RATE_E enFadeOutRate;  
} AUDIO_FADE_S;
```

【成员】

成员名称	描述
bFade	是否开启淡入淡出功能。RK_TRUE：开启淡入淡出功能。RK_FALSE：关闭淡入淡出功能。
enFadeInRate	音频输出设备音量淡入速度。
enFadeOutRate	音频输出设备音量淡出速度。

4.23 AUDIO_CHN_MODE_E

【说明】

音频通道模式类型。

【定义】

```
typedef enum rkAUDIO_CHN_MODE_E {  
    AUDIO_CHN_MODE_LEFT = 10,  
    AUDIO_CHN_MODE_RIGHT = 11,  
    AUDIO_CHN_MODE_BUTT  
} AUDIO_CHN_MODE_E;
```

【成员】

成员名称	描述
AUDIO_CHN_MODE_LEFT	通道独立输出到左声道。
AUDIO_CHN_MODE_RIGHT	通道独立输出到右声道。

4.24 AO_CHN_PARAM_S

【说明】

音频输出通道参数结构体。

【定义】

```
typedef struct rkAO_CHN_PARAM_S {  
    AUDIO_CHN_MODE_E enMode;  
} AO_CHN_PARAM_S;
```

【成员】

成员名称	描述
enMode	AO通道独立输出模式。

4.25 AUDIO_SAVE_FILE_INFO_S

【说明】

定义音频保存文件功能配置信息结构体。

【定义】

```
typedef struct rkAUDIO_SAVE_FILE_INFO_S {
    RK_BOOL    bCfg;
    RK_CHAR    aFilePath[MAX_AUDIO_FILE_PATH_LEN];
    RK_CHAR    aFileName[MAX_AUDIO_FILE_NAME_LEN];
    RK_U32    u32FileSize; /*in KB*/
} AUDIO_SAVE_FILE_INFO_S;
```

【成员】

成员名称	描述
bCfg	配置使能开关。
aFilePath	音频文件保存路径。
aFileName	音频文件保存名称。
u32FileSize	文件大小，单位byte。

4.26 AUDIO_FILE_STATUS_S

【说明】

定义音频文件保存状态结构体。

【定义】

```
typedef struct rkAUDIO_FILE_STATUS_S {
    RK_BOOL    bSaving;
} AUDIO_FILE_STATUS_S;
```

【成员】

成员名称	描述
bSaving	文件是否处于保存状态。

4.27 AIO_VQE_CONFIG_METHOD

【说明】

定义VQE模块配置方法结构体。

【定义】

```
typedef enum rkAIO_VQE_CONFIG_METHOD {
    AIO_VQE_CONFIG_NONE,
    AIO_VQE_CONFIG_USER, // user set all configs by parameters
    AIO_VQE_CONFIG_LOAD_FILE, // read and parse config file
} AIO_VQE_CONFIG_METHOD;
```

【成员】

成员名称	描述
AIO_VQE_CONFIG_NONE	无配置。
AIO_VQE_CONFIG_USER	用户输入配置。（目前未使用）
AIO_VQE_CONFIG_LOAD_FILE	读文件配置。

4.28 AI_VQE_CONFIG_S

【说明】

定义VQE模块配置结构体。

【定义】

```
typedef struct rkAI_VQE_CONFIG_S {
    AIO_VQE_CONFIG_METHOD enCfgMode; /* see AIO_VQE_CONFIG_METHOD */
    RK_S32      s32WorkSampleRate;   /* Sample Rate: 8KHz/16KHz/48KHz. default: 16KHz */
    RK_S32      s32FrameSample;     /* VQE frame length, default: 256 frames(16ms,16KHz) */
    RK_S64      s64RefChannelType;  /* Ref channel layout type */
    RK_S64      s64RecChannelType;  /* Rec channel layout type */
    RK_S64      s64ChannelLayoutType; /* Channel layout type */

    union {
        /* config file if enCfgMode = AIO_VQE_CONFIG_LOAD_FILE */
        RK_CHAR    aCfgFile[MAX_AUDIO_FILE_PATH_LEN];
    };
} AI_VQE_CONFIG_S;
```

【成员】

成员名称	描述
enCfgMode	VQE模块配置方法结构体。
s32WorkSampleRate	输入音频采样率（目前未使用）。
s32FrameSample	算法处理音频每帧采样点（目前未使用）。
s64RefChannelType	回采数据的通道类型。该值为算法的私有参数，用于标示出回采数据所在的通道。示例中，其值为"63"，63的二进制可写成0b'111111，标示8通道的音频数据中，通道0~通道5为回采数据。
s64RecChannelType	录音数据的通道类型。该值为算法的私有参数，用于标示出录音数据所在的通道。示例中，其值为"192"，192的二进制可写成0b'11000000，标示8通道的音频数据中，通道6，通道7为录音数据。
s64ChannelLayoutType	输入音频通道类型。该值的每一个bit位表示一个声道，比如对于8声道的数据，其layout的二进制为: 0b' 11111111，对应的十进制为255，因此可将其设置为"255"。
aCfgFile[MAX_AUDIO_FILE_PATH_LEN]	读文件方法的配置文件路径。

4.29 AI_VQE_MOD_ENABLE_S

【说明】

定义VQE模块使能结构体。

【定义】

```

typedef struct rkAI_VQE_MOD_ENABLE_S {
    RK_BOOL bAec;
    RK_BOOL bBf;
    RK_BOOL bFastAec;
    RK_BOOL bAes;
    RK_BOOL bGsc;
    RK_BOOL bAgc;
    RK_BOOL bAnr;
    RK_BOOL bNlp;
    RK_BOOL bDereverb;
    RK_BOOL bCng;
    RK_BOOL bDtd;
} AI_VQE_MOD_ENABLE_S;

```

【成员】

成员名称	描述
bAec	回声消除模块使能。
bBf	波束形成模块使能。
bFastAec	线性残留回声消除模块使能。
bAes	非线性残留回声消除模块使能。
bGsc	开波束后滤波模块使能。
bAgc	自动增益控制模块使能。
bAnr	噪声抑制模块使能。
bNlp	非线性残留回声消除模块使能。
bDereverb	去混响模块使能。
bCng	舒适度噪声模块使能。
bDtd	双讲检测模块使能。

4.30 AI_AED_CONFIG_S

【说明】

定义AED属性结构体。

【定义】

```

typedef struct rkAI_AED_CONFIG_S {
    RK_FLOAT fSnrDB;
    RK_FLOAT fLsdDB;
    RK_S32 s32Policy;
} AI_AED_CONFIG_S;

```

【成员】

成员名称	描述
fSnrDB	语音信噪比阈值，大于则输出1。
fLsdDB	超大声阈值，大于则输出1。最大为0dB。
s32Policy	信噪比检测算法灵敏度，取值范围为[0, 2]，值越大越灵敏，越容易满足检测阈值。默认取1。

4.31 AI_AED_RESULT_S

【说明】

定义AED事件检测结果结构体。

【定义】

```
typedef struct rkAI_BCD_RESULT_S {  
    RK_BOOL bAcousticEventDetected;  
    RK_BOOL bLoudSoundDetected;  
} AI_AED_RESULT_S;
```

【成员】

成员名称	描述
bAcousticEventDetected	语音信噪比事件检测结果，1为满足信噪比阈值，0不满足。
bLoudSoundDetected	超大声事件检测结果，1为满足超大声阈值，0不满足。
lsdResult	超大声检测得到的音量大小，以db为单位。

4.32 AI_BCD_CONFIG_S

【说明】

定义BCD属性结构体。

【定义】

```
typedef struct rkAI_BCD_CONFIG_S {  
    int mFrameLen;  
} AI_BCD_CONFIG_S;
```

【成员】

成员名称	描述
mFrameLen	哭声检测确认帧长，数值应小于125，建议长度100，越长检测延迟越高，越容易漏检测；越短越容易误检测。

4.33 AI_BCD_RESULT_S

【说明】

定义BCD事件检测结果结构体。

【定义】

```
typedef struct rkAI_BCD_RESULT_S {  
    RK_BOOL bBabyCry;  
} AI_BCD_RESULT_S;
```

【成员】

成员名称	描述
bBabyCry	婴儿哭声事件检测结果，1为检测到哭声，0没检测到。

4.34 AI_BUZ_CONFIG_S

【说明】

定义BUZ属性结构体。

【定义】

```
typedef struct rkAI_BUZ_CONFIG_S {  
    int mFrameLen;  
} AI_BUZ_CONFIG_S;
```

【成员】

成员名称	描述
mFrameLen	蜂鸣器报警声检测确认帧长，数值应小于125，建议长度100，越长检测延迟越高，越容易漏检测；越短越容易误检测。

4.35 AI_BUZ_RESULT_S

【说明】

定义BUZ事件检测结果结构体。

【定义】

```
typedef struct rkAI_BUZ_RESULT_S {  
    RK_BOOL bBuzz;  
} AI_BUZ_RESULT_S;
```

【成员】

成员名称	描述
bBuzz	蜂鸣器报警声事件检测结果，1为检测到警报声，0没检测到。

4.36 AI_GBS_CONFIG_S

【说明】

定义GBS属性结构体。

【定义】

```
typedef struct rkAI_GBS_CONFIG_S {  
    int mFrameLen;  
} AI_GBS_CONFIG_S;
```

【成员】

成员名称	描述
mFrameLen	玻璃破碎声检测确认帧长，数值应小于125，大于20，建议长度25-50，越长检测延迟越高，越容易漏检测；越短越容易误检测。

4.37 AI_GBS_RESULT_S

【说明】

定义GBS事件检测结果结构体。

【定义】

```
typedef struct rkAI_GBS_RESULT_S {  
    RK_BOOL bGbs;  
} AI_GBS_RESULT_S;
```

【成员】

成员名称	描述
bGbs	玻璃破碎声事件检测结果，1为检测到玻璃破碎声，0没检测到。

4.38 音频编码和解码

音频编解码相关数据类型、数据结构定义如下：

- [ADEC_CHN](#): 定义 ADEC 通道。
- [ADEC_MAX_CHN_NUM](#): 定义音频解码通道的最大个数。
- [ADEC_MODE_E](#): 定义解码方式。
- [ADEC_ATTR_CODEC_S](#): 定义音频解码器属性结构体。
- [ADEC_CHN_ATTR_S](#): 定义解码通道属性结构体。
- [ADEC_DECODER_S](#): 定义解码器属性结构体。
- [AUDIO_FRAME_INFO_S](#): 定义解码后的音频帧信息结构体。
- [ADEC_CHN_STATE_S](#): 定义音频解码通道的数据缓存状态结构体。
- [AENC_CHN](#): 定义 AENC 通道。
- [AENC_MAX_CHN_NUM](#): 定义编码解码通道的最大个数。
- [AENC_ATTR_CODEC_S](#): 定义音频编码器属性结构体。
- [AENC_CHN_ATTR_S](#): 定义编码通道属性结构体。
- [AENC_ENCODER_S](#): 定义编码器属性结构体。
- [AUDIO_STREAM_S](#): 定义音频码流结构体。
- [AUDIO_ADEC_PARAM_S](#): 定义音频编/解码函数数据交互结构体

4.39 ADEC_CHN

【说明】

定义 ADEC 通道。

【定义】

```
typedef RK_S32 ADEC_CHN;
```

4.40 ADEC_MAX_CHN_NUM

【说明】

定义音频解码通道的最大个数。

【定义】

```
#define ADEC_MAX_CHN_NUM 32
```

4.41 ADEC_MODE_E

【说明】

定义解码方式。

【定义】

```
typedef enum rkADEC_MODE_E {  
    /*  
     * require input is valid dec pack(a complete frame encode result),  
     * e.g.the stream get from AENC is a valid dec pack, the stream know  
     * actually pack len from file is also a dec pack.  
     * this mode is high-performative*/  
    ADEC_MODE_PACK = 0,  
    /*  
     * input is stream,low-performative, if you couldn't find out whether  
     * a stream is valid dec pack,you could use this mode  
     */  
    ADEC_MODE_STREAM,  
    ADEC_MODE_BUTT  
} ADEC_MODE_E;
```

【成员】

成员名称	描述
ADEC_MODE_PACK	pack方式解码。
ADEC_MODE_STREAM	stream方式解码。

【注意事项】

- pack方式用于用户确认当前码流包为一帧数据编码结果的情况下，解码器会直接进行对其解码，如果不是一帧，解码器会出错。这种模式的效率比较高，在使用AENC 模块编码的码流包如果没有破坏，均可以使用此方式解码。
- stream方式用于用户不能确认当前码流包是不是一帧数据的情况下，解码器需要对码流进行判断并缓存，此工作方式的效率低下，一般用于读文件码流送解码或者不确定码流包边界的情况。当然由于语音编码码流长度固定，很容易确定在码流中的帧边界，推荐使用pack方式解码。

4.42 AUDIO_G726_BPS

【说明】

定义G726比特率。

```
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;
```

【成员】

成员名称	描述
G726_BPS_16K	输入/出数据比特率16000。对应G726编码/解码器码字2bit。
G726_BPS_24K	输入/出数据比特率24000。对应G726编码/解码器码字3bit。
G726_BPS_32K	输入/出数据比特率32000。对应G726编码/解码器码字4bit。
G726_BPS_40K	输入/出数据比特率40000。对应G726编码/解码器码字5bit。

4.43 ADEC_ATTR_CODEC_S

【说明】

定义音频解码器属性结构体

【定义】

```
typedef struct rkADEC_ATTR_CODEC_S {
    RK_CODEC_ID_E enType;
    RK_U32 u32Channels;
    RK_U32 u32SampleRate;
    RK_U32 u32Bitrate;

    RK_VOID *pExtraData;
    RK_U32 u32ExtraDataSize;

    RK_U32 u32Resv[4];
    RK_VOID *pstResv;
} ADEC_ATTR_CODEC_S;
```

【成员】

成员名称	描述
enType	解码协议(同ADEC_CHN_ATTR_S的enType)。
u32Channels	码流声道数。
u32SampleRate	码流采样率。
u32Bitrate	音频数据比特率
pExtraData	解码外部数据
u32ExtraDataSize	解码外部数据长度
u32Resv[4]	保留字节，用于解码器参数扩展
pstResv	保留结构体指针，用于解码器参数扩展

【注意事项】

- 当结构体变量不足以传递解码器参数时，用户可通过保留字节u32Resv和保留结构体指针pstResv来扩展参数。
- 特定解码器需要设置当前音频协议支持的音频数据比特率，比如G726解码器。G726解码器在初始化时，需要根据比特率设置正确的码字(codeword)，见[AUDIO_G726_BPS](#)。

4.44 ADEC_CHN_ATTR_S

【说明】

定义解码通道属性结构体。

【定义】

```
typedef struct rkADEC_CH_ATTR_S {
    RK_CODEC_ID_E enType;
    ADEC_MODE_E enMode;
    RK_U32 u32BufCount;
    RK_U32 u32BufSize;

    ADEC_ATTR_CODEC_S stCodecAttr;
} ADEC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enType	解码协议。
enMode	解码方式, 见 ADEC_MODE_E 。
u32BufCount	解码缓存数量。
u32BufSize	解码缓存大小(单位字节)。
stCodecAttr	解码器属性结构体。

【注意事项】

- u32BufSize定义解码器buffer大小。当使用注册解码器解码时, 函数pfnDecodeFrm的pu8Outbuf大小即为u32BufSize定义的大小。当该值定义过小时, 会造成一次解码的数据, 要多次读取的情况。当该值为定义时(即0), ADEC默认解码buffer大小为4096字节。

【相关数据类型及接口】

[RK_MPI_ADEC_CreateChn](#)

4.45 ADEC_DECODER_S

【说明】

定义解码器结构体。

【定义】

```
typedef struct rkADEC_DECODER_S {
    RK_CODEC_ID_E enType;
    RK_UCHAR aszName[17];
    RK_S32 (*pfnOpenDecoder)(RK_VOID *pDecoderAttr, RK_VOID **ppDecoder);
    RK_S32 (*pfnDecodeFrm)(RK_VOID *pDecoder, RK_VOID *pDecParam);
    RK_S32 (*pfnGetFrmInfo)(RK_VOID *pDecoder, RK_VOID *pInfo);
    RK_S32 (*pfnCloseDecoder)(RK_VOID *pDecoder);
    RK_S32 (*pfnResetDecoder)(RK_VOID *pDecoder);
} ADEC_DECODER_S;
```

【成员】

成员名称	描述
enType	解码协议类型。
aszName	解码器名称。
pfnOpenDecoder	打开解码器的函数指针。
pfnDecodeFrm	解码的函数指针。
pfnGetFrmInfo	获取音频帧信息的函数指针。
pfnCloseDecoder	关闭解码器的函数指针。
pfnResetDecoder	清空缓存 buffer, 复位解码器。

【注意事项】

- pfnOpenDecoder函数第一个参数pDecoderAttr为[ADEC_ATTR_CODEC_S](#)类型, 用户可在对应的注册函数中强行类型转换, 可访问到[ADEC_ATTR_CODEC_S](#)中的相关变量。
- pfnDecodeFrm函数的第二个参数pDecParam为[AUDIO_ADENC_PARAM_S](#)类型, 用户可在对应的注册函数中强行类型转换, 可访问到[AUDIO_ADENC_PARAM_S](#)中的相关变量。

【相关数据类型及接口】

[RK_MPI_ADEC_RegisterDecoder](#)

4.46 ADEC_DECODER_RESULT

【说明】

定义注册解码器解码函数返回值

【定义】

```
typedef enum rkADEC_DECODER_RESULT {
    ADEC_DECODER_OK = RK_SUCCESS,
    ADEC_DECODER_TRY AGAIN,
    ADEC_DECODER_ERROR,
    ADEC_DECODER_EOS,
} ADEC_DECODER_RESULT;
```

【成员】

成员名称	描述
ADEC_DECODER_OK	解码成功(送解码数据成功且拿到了解码数据)。
ADEC_DECODER_TRY AGAIN	解码重试。
ADEC_DECODER_ERROR	解码错误。
ADEC_DECODER_EOS	解码最后一帧数据。

【注意事项】

- ADEC_DECODER_TRY AGAIN: 该值用于在表示当前解码数据送入失败(比如当前解码内部缓冲满, 无法送入新的数据)或者送入码流数据后, 没有拿到解码数据的情形。ADEC模块在获取该返回值后, 会再次送入当前码流(如果[AUDIO_ADECN_PARAM_S](#)的u32InLen输出值不为0)。
- ADEC_DECODER_ERROR: 解码错误。ADEC获取到该值后, 会丢弃当前帧。
- ADEC_DECODER_EOS: 该值用于通知ADEC模块, 解码器已送出最后一帧数据(标记当前解码结束)。ADEC模块获取到该值后, 不再调用解码器相关函数进行解码, 并标记当前音频帧为EOS帧。

4.47 ADEC_FRAME_INFO_S

【说明】

定义注册解码器音频帧信息

【定义】

```
typedef struct rkADEC_FRAME_INFO_S {
    RK_U32    u32SampleRate;
    RK_U32    u32Channels;
    RK_U64    u64ChnLayout;
    AUDIO_BIT_WIDTH_E enBitWidth;
    RK_U32    resv[2];
} ADEC_FRAME_INFO_S;
```

【成员】

成员名称	描述
u32SampleRate	解码数据的采样率。
u32Channels	解码数据的声道数。
u64ChnLayout	解码数据的声道布局。
enBitWidth	解码数据的采样精度。
resv[2]	保留位, 用于扩展。

4.48 AUDIO_FRAME_INFO_S

【说明】

定义解码后的音频帧信息结构体。

【定义】

```
typedef struct rkAUDIO_FRAME_INFO_S {  
    AUDIO_FRAME_S *pstFrame; /*frame ptr*/  
    RK_U32     u32Id; /*frame id*/  
} AUDIO_FRAME_INFO_S;
```

【成员】

成员名称	描述
pstFrame	音频帧指针。
u32Id	音频帧的索引。

4.49 ADEC_CHN_STATE_S

【说明】

定义音频解码通道的数据缓存状态结构体。

【定义】

```
typedef struct rkADEC_CH_STATE_S {  
    RK_BOOL bEndOfStream; /* EOS flag */  
    RK_U32 u32BufferFrmNum; /* total number of channel buffer */  
    RK_U32 u32BufferFreeNum; /* free number of channel buffer */  
    RK_U32 u32BufferBusyNum; /* busy number of channel buffer */  
} ADEC_CHN_STATE_S;
```

【成员】

成员名称	描述
bEndOfStream	解码码流结束状态。
u32BufferFrmNum	解码通道总的缓存块数。
u32BufferFreeNum	可用的空闲缓存块数。
u32BufferBusyNum	被占用缓存块数。

4.50 AENC_CHN

【说明】

定义 AENC 通道。

【定义】

```
typedef RK_S32 AENC_CHN;
```

4.51 AENC_MAX_CHN_NUM

【说明】

定义音频解码通道的最大个数。

【定义】

```
#define AENC_MAX_CHN_NUM 32
```

4.52 AENC_ATTR_CODEC_S

【说明】

定义音频编码器属性结构体

【定义】

```
typedef struct rkAENC_ATTR_CODEC_S {  
    RK_CODEC_ID_E    enType;  
    AUDIO_BIT_WIDTH_E enBitwidth;  
    RK_U32          u32Channels;  
    RK_U32          u32SampleRate;  
    RK_U32          u32Bitrate;  
    RK_U32          u32Resv[4];  
    RK_VOID         *pstResv;  
} AENC_ATTR_CODEC_S;
```

【成员】

成员名称	描述
enType	编码协议类型(同AENC_CHN_ATTR_S的enType)。
enBitwidth	音频采样精度。
u32Channels	音频声道数。
u32SampleRate	音频采样率。
u32Bitrate;	音频流输出比特率
u32Resv[4]	保留字节，用于编码器参数扩展。
pstResv	保留结构体指针，用于编码器参数扩展。

【注意事项】

- 部分编码器(如G726，见[AUDIO_G726_BPS](#))需要设置当前音频协议支持的比特率，用于编码码流的比特率控制。

4.53 AENC_CHN_ATTR_S

【说明】

定义编码通道属性结构体。

【定义】

```
typedef struct rkAENC_CHN_ATTR_S {  
    RK_CODEC_ID_E    enType;  
    RK_U32          u32BufCount;  
  
    AENC_ATTR_CODEC_S stCodecAttr;  
} AENC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enType	编码协议类型。
u32BufCount	音频编码缓存个数。
stCodecAttr	编码器属性结构体。

4.54 AENC_ENCODER_S

【说明】

定义解码器结构体。

【定义】

```
typedef struct rkAENC_ENCODER_S {
    RK_CODEC_ID_E enType;
    RK_U32 u32MaxFrmLen;
    RK_CHAR aszName[17];
    RK_S32 (*pfnOpenEncoder)(RK_VOID *pEncoderAttr, RK_VOID **ppEncoder);
    RK_S32 (*pfnEncodeFrm)(RK_VOID *pEncoder, RK_VOID *pParam);
    RK_S32 (*pfnCloseEncoder)(RK_VOID *pEncoder);
} AENC_ENCODER_S;
```

【成员】

成员名称	描述
enType	编码协议类型。
u32MaxFrmLen	最大码流长度。
aszName	编码器名称。
pfnOpenEncoder	打开编码器的函数指针。
pfnEncodeFrm	进行编码的函数指针。
pfnCloseEncoder	关闭编码器的函数指针。

【注意事项】

- pfnOpenEncoder函数第一个参数pEncoderAttr为[AENC_ATTR_CODEC_S](#)类型，用户可在对应的注册函数中强行类型转换，并访问到[AENC_ATTR_CODEC_S](#)中的相关变量。
- pfnEncodeFrm函数的第二个参数pParam为[AUDIO_ADENC_PARAM_S](#)类型，，用户可在对应的注册函数中强行类型转换，并访问到[AUDIO_ADENC_PARAM_S](#)中的相关变量。

【相关数据类型及接口】

[RK_MPI_AENC_RegisterEncoder](#)

4.55 AUDIO_ADENC_PARAM_S

【说明】

定义注册解码器/编码器 解码/编码输入输出结构体。

【定义】

```

typedef struct rkAUDIO_ADENC_PARAM_S {
    RK_U8      *pu8InBuf;
    RK_U32     u32InLen;
    RK_U64     u64InTimeStamp;

    RK_U8      *pu8OutBuf;
    RK_U32     u32OutLen;
    RK_U64     u64OutTimeStamp;
} AUDIO_ADENC_PARAM_S;

```

【成员】

成员名称	描述	输入/输出
pu8InBuf	输入数据buffer指针。	输入
u32InLen	输入： 输入数据长度(即pu8InBuf中输入有效数据的长度)。 输出： 输入buffer中(即pu8InBuf)剩余数据的长度	输入/出
u64InTimeStamp	输入数据的时间戳。	输入
pu8OutBuf	输出数据buffer指针。	输入
u32OutLen	输入： 输出buffer的最大容量。 输出： 输出buffer中有效数据的长度。	输入/出
u64OutTimeStamp	输出数据的时间戳。	输出

4.56 AENC_ENCODER_RESULT

【说明】

定义注册编码器编码返回值。

【定义】

```

typedef enum rkENC_ENCODER_RESULT {
    AENC_ENCODER_OK = RK_SUCCESS,
    AENC_ENCODER_TRY AGAIN,
    AENC_ENCODER_ERROR,
    AENC_ENCODER_EOS,
} AENC_ENCODER_RESULT;

```

【成员】

成员名称	描述
AENC_ENCODER_OK	编码成功(送编码数据成功且拿到了编码后的码流数据)。
AENC_ENCODER_TRY AGAIN	编码重试。
AENC_ENCODER_ERROR	编码错误。
AENC_ENCODER_EOS	编码最后一帧数据。

【注意事项】

- AENC_ENCODER_TRY AGAIN：该值用于在表示当前帧送入失败或者送入当前帧后，没有获取码流。AENC模块在获取该返回值后，会再次送入当前码流(如果[AUDIO_ADENC_PARAM_S](#)的u32InLen输出值不为0时)。
- AENC_ENCODER_ERROR：AENC模块获取该返回值后，会丢弃当前输入/输出数据。
- AENC_ENCODER_EOS：该值用于通知AENC模块，编码器已送出最后一帧数据(标记当前解码结束)。AENC模块获取到该值后，不再调用编码器相关函数进行编码，并标记当前音频帧为EOS帧。

5. AUDIO错误码

5.1 音频输入AUDIO错误码

音频输入APIAUDIO错误码如下所示。

错误代码	宏定义	描述
0xA00A8001	RK_ERR_AI_INVALID_DEVID	音频输入设备号无效
0xA00A8002	RK_ERR_AI_INVALID_CHNID	音频输入通道号无效
0xA00A8003	RK_ERR_AI_ILLEGAL_PARAM	音频输入参数设置无效
0xA00A8005	RK_ERR_AI_NOT_ENABLED	音频输入设备或通道没使能
0xA00A8006	RK_ERR_AI_NULL_PTR	输入空指针错误
0xA00A8007	RK_ERR_AI_NOT_CONFIG	音频输入设备属性未设置
0xA00A8008	RK_ERR_AI_NOT_SUPPORT	操作不被支持
0xA00A8009	RK_ERR_AI_NOT_PERM	操作不允许
0xA00A800C	RK_ERR_AI_NOMEM	分配内存失败
0xA00A800D	RK_ERR_AI_NOBUF	音频输入缓存不足
0xA00A800E	RK_ERR_AI_BUF_EMPTY	音频输入缓存为空
0xA00A800F	RK_ERR_AI_BUF_FULL	音频输入缓存为满
0xA00A8010	RK_ERR_AI_SYS_NOTREADY	音频输入系统未初始化
0xA00A8012	RK_ERR_AI_BUSY	音频输入系统忙
0xA00A8041	RK_ERR_AI_VQE_ERR	AO VQE 处理错误

5.2 音频输出AUDIO错误码

音频输出APIAUDIO错误码如下所示。

错误代码	宏定义	描述
0xA00B8001	RK_ERR_AO_INVALID_DEVID	音频输出设备号无效
0xA00B8002	RK_ERR_AO_INVALID_CHNID	音频输出通道号无效
0xA00B8003	RK_ERR_AO_ILLEGAL_PARAM	音频输出参数设置无效
0xA00B8005	RK_ERR_AO_NOT_ENABLED	音频输出设备或通道没使能
0xA00B8006	RK_ERR_AO_NULL_PTR	输出空指针错误
0xA00B8007	RK_ERR_AO_NOT_CONFIG	音频输出设备属性未设置
0xA00B8008	RK_ERR_AO_NOT_SUPPORT	操作不被支持
0xA00B8009	RK_ERR_AO_NOT_PERM	操作不允许
0xA00B800C	RK_ERR_AO_NOMEM	系统内存不足
0xA00B800D	RK_ERR_AO_NOBUF	音频输出缓存不足
0xA00B800E	RK_ERR_AO_BUF_EMPTY	音频输出缓存为空
0xA00B800F	RK_ERR_AO_BUF_FULL	音频输出缓存为满
0xA00B8010	RK_ERR_AO_SYS_NOTREADY	音频输出系统未初始化
0xA00B8012	RK_ERR_AO_BUSY	音频输出系统忙
0xA00B8041	RK_ERR_AO_VQE_ERR	AO VQE 处理错误

5.3 音频解码AUDIO错误码

音频解码APIAUDIO错误码如下所示。

错误代码	宏定义	描述
0xA00D8001	RK_ERR_ADEC_INVALID_DEVID	音频解码设备号无效
0xA00D8002	RK_ERR_ADEC_INVALID_CHNID	音频解码通道号无效
0xA00D8003	RK_ERR_ADEC_ILLEGAL_PARAM	音频解码参数设置无效
0xA00D8004	RK_ERR_ADEC_EXIST	音频解码通道已被使用
0xA00D8005	RK_ERR_ADEC_UNEXIST	音频解码通道未被创建
0xA00D8006	RK_ERR_ADEC_NULL_PTR	输入参数空指针错误
0xA00D8007	RK_ERR_ADEC_NOT_CONFIG	音频解码属性未设置
0xA00D8008	RK_ERR_ADEC_NOT_SUPPORT	操作不被支持
0xA00D8009	RK_ERR_ADEC_NOT_PERM	操作不允许
0xA00D800C	RK_ERR_ADEC_NOMEM	解码分配内存失败
0xA00D800D	RK_ERR_ADEC_NOBUF	解码通道缓存分配失败
0xA00D800E	RK_ERR_ADEC_BUF_EMPTY	音频解码缓存buffer为空
0xA00D800F	RK_ERR_ADEC_BUF_FULL	音频解码缓存为满
0xA00D8010	RK_ERR_ADEC_SYS_NOTREADY	音频解码系统未初始化
0xA00D8040	RK_ERR_ADEC_DECODER_ERR	音频解码数据错误
0xA00D8041	RK_ERR_ADEC_BUF_LACK	音频解码输入缓存空间不够
0xA00D8042	RK_ERR_ADEC_REGISTER_ERR	音频解码器注册/注销失败

5.4 音频编码AUDIO错误码

音频解码APIAUDIO错误码如下所示。

错误代码	宏定义	描述
0xA00C8001	RK_ERR_AENC_INVALID_DEVID	音频设备号无效
0xA00C8002	RK_ERR_AENC_INVALID_CHNID	音频编码通道号无效
0xA00C8003	RK_ERR_AENC_ILLEGAL_PARAM	音频编码参数设置无效
0xA00C8004	RK_ERR_AENC_EXIST	音频编码通道已经创建
0xA00C8005	RK_ERR_AENC_UNEXIST	音频编码通道未创建
0xA00C8006	RK_ERR_AENC_NULL_PTR	输入参数空指针错误
0xA00C8007	RK_ERR_AENC_NOT_CONFIG	编码通道未配置
0xA00C8008	RK_ERR_AENC_NOT_SUPPORT	操作不被支持
0xA00C8009	RK_ERR_AENC_NOT_PERM	操作不允许
0xA00C800C	RK_ERR_AENC_NOMEM	系统内存不足
0xA00C800D	RK_ERR_AENC_NOBUF	编码通道缓存分配失败
0xA00C800E	RK_ERR_AENC_BUF_EMPTY	编码通道缓存空
0xA00C800F	RK_ERR_AENC_BUF_FULL	音频输出缓存为满
0xA00C8010	RK_ERR_AENC_SYS_NOTREADY	系统没有初始化
0xA00C8040	RK_ERR_AENC_ENCODER_ERR	音频编码数据错误
0xA00C8041	RK_ERR_AENC_VQE_ERR	AENC VQE 处理错误
0xA00C8042	RK_ERR_AENC_REGISTER_ERR	音频编码器注册/注销失败

6. 附录A：注册解码/编码器API说明

ADEC和AENC通过开放注册接口，以便于用户集成第三方提供的编/解码协议。编/解码使用示例代码在rockit相关目录。

SDK发布包中的以下API用于注册和注销编码器和解码器。

6.1 注册解码器

[RK_MPI_ADEC_RegisterDecoder](#): 注册解码器

[RK_MPI_ADEC_UnRegisterDecoder](#): 注销解码器

[ADEC_DECODER_S](#): 定义解码器属性结构体

解码器属性包括解码器类型、解码器名称、打开解码器的函数指针、进行解码的函数指针、获取音频帧信息的函数指针、关闭解码器的函数指针。

- 解码器类型：SDK以枚举标识解码协议，注册时应选择相关协议的解码器类型。
- 解码器名称：用字符串表示，用于在信息中显示。
- 打开解码器的函数指针：

`RK_S32 (pfnOpenDecoder)(RK_VOID*pDecoderAttr, RK_VOID **ppDecoder);`

该函数用于打开解码器。其中第一个参数是解码器属性[ADEC_ATTR_CODEC_S](#)。不同的解码器实现，可能需要不同的设置参数，当[ADEC_ATTR_CODEC_S](#)中变量不足以描述相关设置时，客户可扩展预留的u32Resv[4]或者pstResv指针来完成对应参数的传递。不同的解码器在初始化时，需要部分参数来完成解码器的初始化，因此用户可根据解码的需要以及实际码流，按需填写ADEC_ATTR_CODEC_S部分或者全部参数；第二个参数是解码器句柄，用于返回可用于操作解码器的句柄。该解码器句柄还将用于解码和关闭解码器。该函数返回值0表示打开解码器成功，非0值表示打开解码器失败。

返回值：0表示成功，非0表示失败。

- 进行解码的函数指针

RK_S32 (*pfnDecodeFrm*)(RK_VOID *pEncoder, RK_VOID *pParam);

该函数用于解码数据。第一个参数是pfnOpenDecoder打开解码器时返回的解码器句柄；第二个参数是[AUDIO_ADENC_PARAM_S](#)。如下对[AUDIO_ADENC_PARAM_S](#)结构体进行简单说明：

pu8InBuf：待解码的码流起始地址。

u32InLen：输入：待解码码流的有效长度。

输出：剩余待解码码流的长度，当输入数据全部用完时，设置该值为0。

u64InTimeStamp：待解码码流的时间戳。

pu8OutBuf：存放解码输出帧的起始地址。

u32OutLen：输入：存放解码数据的缓冲的容量；该值大小由[ADEC_CHN_ATTR_S](#)的u32BufSize设置，当没有设置该值时，默认大小

为4096bytes。由于该buffer用于保存解码后的数据，当该buffer大小不足以保存一次解码的数据时，会造成一次解码

的数据需要多长拷贝的情况，进而影响解码效率，因此在内存允许时，可设置[ADEC_CHN_ATTR_S](#)的u32BufSize为一个

较大的值。

输出：解码数据的有效长度；

u64OutTimeStamp：解码数据的时间戳。

需要强调的是，ADEC模块只支持非planar的数据，因此如果解码数据格式为planar格式时，在数据送回ADEC模块时，需要对数据格式进行格式转换。

返回值：见[ADEC_DECODER_RESULT](#)。

- 获取音频帧信息的函数指针

RK_S32 (*pfnGetFrmInfo*)(RK_VOID *pDecoder, RK_VOID *pInfo);

该函数用于获取音频帧参数。第一个参数是pfnOpenDecoder打开解码器时返回的解码器句柄；第二个参数是[ADEC_FRAME_INFO_S](#)，该参数用于ADEC模块获取pfnEncodeFrm解码后输出数据的参数。

返回值：0 表示成功，非0表示失败。

- 关闭解码器的函数指针

RK_S32 (*pfnCloseDecoder*)(RK_VOID *pDecoder);

该函数用于关闭解码器。输入参数是pfnOpenDecoder打开解码器时返回的解码器句柄。

返回值：0 表示成功，非0表示失败。

- 复位解码器的函数指针

RK_S32 (*pfnResetDecoder*)(RK_VOID *pDecoder);

该函数用于复位解码器。输入参数是pfnOpenDecoder打开解码器时返回的解码器句柄。如果用户解码器不需要此函数，可以为该函数原型封装一个空函数。

返回值：0 表示成功，非0表示失败。

【举例】

下面的代码为MP3解码器的注册：

```
RK_S32 register_ext_adec(TEST_ADEC_CTX_S *params) {
    ADEC_DECODER_S adecCtx;
    memset(&adecCtx, 0, sizeof(ADEC_DECODER_S));
    // set handle to default value
    params->s32ExtCodecHandle = -1;
    adecCtx.enType = RK_AUDIO_ID_MP3;
    snprintf((RK_CHAR*)(adecCtx.aszName), sizeof(adecCtx.aszName), "rkaudio");
    adecCtx.pfnOpenDecoder = RKAdudioMp3DecoderOpen;
    adecCtx.pfnDecodeFrm = RKAdudioMp3DecoderDecode;
    adecCtx.pfnGetFrmInfo = RKAdudioMp3DecoderGetFrameInfo;
    adecCtx.pfnCloseDecoder = RKAdudioMp3DecoderClose;
    adecCtx.pfnResetDecoder = RK_NULL;

    RK_LOGD("register_ext_adec, name = %s", adecCtx.aszName);
    RK_S32 ret = RK_MPI_ADEC_RegisterDecoder(&params->s32ExtCodecHandle, &adecCtx);
```

```

if (ret != RK_SUCCESS) {
    RK_LOGE("adec register decoder fail, ret = 0x%x", ret);
    return RK_FAILURE;
}

return RK_SUCCESS;
}

```

MP3解码器的注销。

```

RK_S32 unregister_ext_adec(TEST_ADEC_CTX_S *params) {
    if (params == NULL || params->s32ExtCodecHandle == -1) {
        return RK_SUCCESS;
    }

    RK_S32 ret = RK_MPI_ADEC_UnRegisterDecoder(params->s32ExtCodecHandle);
    if (ret != RK_SUCCESS) {
        RK_LOGE("adec unregister decoder fail, ret = 0x%x", ret);
        return RK_FAILURE;
    }

    params->s32ExtCodecHandle = -1;
    return RK_SUCCESS;
}

```

ADEC chn0使用MP3解码器解码MP3 8K 1channel码流的示例：

```

RK_S32 s32ret = 0;
RK_S32 adecChn = 0;
ADEC_CHN_ATTR_S stChnAttr;
memset(&stChnAttr, 0, sizeof(ADEC_CHN_ATTR_S));
// init ADEC_CHN_ATTR_S
stChnAttr.enType = RK_AUDIO_ID_PCM_ALAW; // MP3
stChnAttr.enMode = ADEC_MODE_PACK;
stChnAttr.u32BufCount = 4;
stChnAttr.u32BufSize = 10*1024; // 10K bytes
// init ADEC_ATTR_CODEC_S
stChnAttr.stCodecAttr.enType = stChnAttr.enType;
stChnAttr.stCodecAttr.u32SampleRate = 8000;
stChnAttr.stCodecAttr.u32Channels = 1;

s32ret = RK_MPI_ADEC_CreateChn(adecChn, &stChnAttr);
if (s32ret != 0) {
    RK_LOGE("RK_MPI_ADEC_CreateChn(chn = %d) fail", adecChn);
}

return s32ret;
}

```

【注意事项】

- 不同解码器实现对解码器属性结构体[ADEC_ATTR_CODEC_S](#)需要的参数可能不一样，当现有变量无法满足解码器设置时，可通过扩展保留位u32Resv和保留结构体指针pstResv来完成相应设置。
- [ADEC CHN ATTR S](#)变量u32BufSize定义ADEC模块与pfnDecodeFrm函数之间解码输出缓冲的大小(单位bytes)，默认值为4096字节。当该缓冲比一次解码数据小时，导致pfnDecodeFrm只能一次返回部分解码数据，因此为了防止数据丢失，需要在pfnDecodeFrm函数进入下一次解码前，先将剩余部分解码数据输出。如果条件允许，建议u32BufSize定义成一个较大的值。

6.2 注册编码器

[RK MPI AENC RegisterEncoder](#): 注册编码器

[RK MPI AENC UnRegisterEncoder](#): 注销编码器

[AENC_ENCODER_S](#): 编码器属性结构体

编码器属性包括编码器类型、最大码流长度、编码器名称、打开编码器的函数指针、进行编码的函数指针、关闭编码器的函数指针。

- 编码器类型

SDK 以枚举标识编码协议，注册时应选择相关协议的编码器类型。

- 最大码流长度

每帧编码后码流的最大长度，AENC 模块将根据注册的最大码流长度分配内存大小。当用户未设置时，该值默认为 4096bytes。

- 编码器名称：用字符串表示，用于在信息中显示。

- 打开编码器的函数指针

RK_S32 (*pfnOpenEncoder*)(RK_VOID*pEncoderAttr, RK_VOID **ppEncoder);

其中第一个参数是编码器属性[AENC_ATTR_CODEC_S](#)，用于传入不同类型的编码器的特定属性。用户可根据需求，自行扩展u32Resv和pstResv；第二个参数是编码器句柄，用于返回可用于操作编码器的句柄。

不同的编码协议以及不同的编码器实现，对编码数据要求不一样。比如某特定协议，可能只支持16K, 1channel, 16bit的数据进行编码；某些特性协议，可能只支持特定长度(特定帧数)的数据进行编码。因此，强烈推荐一次送入AENC模块的待编码数据为满足编码器要求的(采样率，声道数，帧数)的数据；当送入数据不能满足编码器要求时，需要对数据进行对应处理(比如重采样)后，再送入AENC模块。

返回值：0 表示成功，非0表示失败。

- 进行编码的函数指针

RK_S32 (*pfnEncodeFrm*)(RK_VOID*pEncoder, RK_VOID *pParam);

第一个参数是pfnOpenEncoder函数打开编码器时返回的编码器句柄；第二个参数是[AUDIO_ADENC_PARAM_S](#)用于传入/出音频帧数据；

如下对[AUDIO_ADENC_PARAM_S](#)结构体进行简单说明：

pu8InBuf：待编码数据起始地址。

u32InLen：输入：待编码数据的有效长度。

输出：剩余待编码数据的长度，当输入数据全部用完时，设置该值为0。

u64InTimeStamp：待编码数据的时间戳。

pu8OutBuf：存放编码成功后码流数据的起始地址。

u32OutLen：输入：存放码流数据缓冲的容量；该值大小由最大码流长度设置。

输出：编码码流的有效长度；

u64OutTimeStamp：编码码流时间戳。

返回值：见[AENC_ENCODER_RESULT](#)。

- 关闭编码器的函数指针

RK_S32 (*pfnCloseEncoder*)(RK_VOID*pEncoder);

输入参数是pfnOpenEncoder函数打开编码器时返回的编码器句柄。

返回值：0 表示成功，非0表示失败。

【举例】

下面的代码为 MP3编码器的注册：

```
RK_S32 register_ext_aenc(TEST_AENC_CTX_S *params) {
    AENC_ENCODER_S aencCtx;
    memset(&aencCtx, 0, sizeof(AENC_ENCODER_S));
    params->s32ExtCodecHandle = -1;

    aencCtx.enType = RK_AUDIO_ID_MP3;
    snprintf((RK_CHAR*)(aencCtx.aszName),
              sizeof(aencCtx.aszName), "rkaudio");
    aencCtx.u32MaxFrmLen = 2048;
    aencCtx.pfnOpenEncoder = RKAdudioMp3EncoderOpen;
    aencCtx.pfnEncodeFrm = RKAdudioMp3EncoderEncode;
    aencCtx.pfnCloseEncoder = RKAdudioMp3EncoderClose;

    RK_LOGD("register_ext_aenc %s", aencCtx.aszName);
```

```

RK_S32 ret = RK_MPI_AENC_RegisterEncoder(&params->s32ExtCodecHandle, &aencCtx);
if (ret != RK_SUCCESS) {
    RK_LOGE("aenc %s register decoder fail", aencCtx.aszName, ret);
    return RK_FAILURE;
}

return RK_SUCCESS;
}

```

MP3编码器的注销。

```

RK_S32 unregister_ext_aenc(TEST_AENC_CTX_S *params) {
    if (params == NULL || params->s32ExtCodecHandle == -1) {
        return RK_SUCCESS;
    }
    RK_LOGD("unregister_ext_aenc");
    RK_S32 ret = RK_MPI_AENC_UnRegisterEncoder(params->s32ExtCodecHandle);
    if (ret != RK_SUCCESS) {
        RK_LOGE("aenc unregister decoder fail", ret);
        return RK_FAILURE;
    }

    params->s32ExtCodecHandle = -1;
    return RK_SUCCESS;
}

```

AENC chn0使用MP3编码器编码MP3(输入数据参数:8K 1channel 16bit)码流的示例：

```

RK_S32 s32ret = 0;
RK_S32 aencChn = 0;
AENC_CHN_ATTR_S stChnAttr;
memset(&stChnAttr, 0, sizeof(AENC_CHN_ATTR_S));
// init AENC_CHN_ATTR_S
stChnAttr.enType = RK_AUDIO_ID_PCM_ALAW; // MP3
stChnAttr.u32BufCount = 4;

// init AENC_ATTR_CODEC_S
stChnAttr.stCodecAttr.enType = stChnAttr.enType;
stChnAttr.stCodecAttr.enBitwidth = AUDIO_BIT_WIDTH_16;
stChnAttr.stCodecAttr.u32SampleRate = 8000;
stChnAttr.stCodecAttr.u32Channels = 1;
s32ret = RK_MPI_AENC_CreateChn(aencChn, &stChnAttr);
if (s32ret) {
    RK_LOGE("create aenc chn %d err:0x%x\n", aencChn, s32ret);
}

return s32ret;

```

7. 附录B：静态编译声音质量增强和事件检测功能时编译选项说明

7.1 静态编译声音质量增强功能

-Wl,--whole-archive libaec_bf_process.a -Wl,--no-whole-archive librkaudio_common.a

7.2 静态编译事件检测功能（AED，BCD，BUZ，GBS）

-Wl,--whole-archive librkaudio_detect.a -Wl,--no-whole-archive librkaudio_common.a

区域管理

[重要概念](#)

[使用步骤介绍](#)

[举例](#)

[API 参考](#)

[数据类型](#)

[RGN错误码](#)

1. 重要概念

- 区域类型
 - OVERLAY：视频叠加区域，其中区域支持位图的加载、背景色更新等功能。

▪ 图1-1 RGN OVERLAY效果图



- COVER：视频遮挡区域，其中区域支持纯色块遮挡。

▪ 图1-2 RGN COVER效果图



- MOSAIC：马赛克遮挡区域，支持精度调节。

▪ 图1-3 RGN 马赛克效果图



- LINE：拓展的线条视频叠加区域，支持设置颜色、线宽等属性。

▪ 图1-4 RGN LINE效果图



- 区域层次

区域层次表示区域的叠加级别，层次值越大，表示区域的显示级别越高。当发生重叠时，层次值大的将会覆盖层次值小的。目前仅在VO中支持区域层次管理。

- 位图填充(针对 OVERLAY 有效)

位图填充是指将位图的内存值填充到区域内存空间中，位图将会从区域的左上角开始填充。当位图小于区域时，只能填充一部分内存，剩余部分保持原有值；位图大小等于区域时，将刚好全部填充；当位图大于区域时，位图只能将自身和区域一样大小的内存信息填充到区域中。

- 区域公共属性 ([RGN_ATTR_S](#))

用户创建一个区域时，需要设置该属性信息，它包含公共的资源信息。例如，OVERLAY 包含像素格式，大小。

- 通道显示属性 ([RGN_CHN_ATTR_S](#))

通道显示属性表明区域在某通道的显示特征。例如，OVERLAY 的通道显示属性包含显示位置，层次，前景 Alpha，背景 Alpha 等。当通道显示属性中的区域是否显示(is_show)为 TRUE 时，表示显示在该通道中；反之，表示在该通道中存在，但处于隐藏状态。

- 区域 QP 保护

当区域叠加到视频上进行压缩编码时，为了保证叠加区域的清晰度不因为数据压缩而变模糊，可以单独设定叠加区域部分的压缩特性，即设定 qp 保护功能参数。qp 保护功能是 OVERLAY 特有的功能，且仅针对 H.264/H.265 类型编码通道有效，对其它类型无效。

1.1 表1-1 RK356X region 支持的模块信息

类型	支持模块	设备号取值范围	通道号取值范围
OVERLAY	VENC	0	[0, VENC_MAX_CHN_NUM - 1]
	VO	[0, VO_MAX_LAYER_NUM - 1]	[0, VO_MAX_CHN_NUM - 1]
	VPSS	[0, VPSS_MAX_GRP_NUM - 1]	[0, VPSS_MAX_CHN_NUM - 1]
COVER	VENC	0	[0, VENC_MAX_CHN_NUM - 1]
	VO	[0, VO_MAX_LAYER_NUM - 1]	[0, VO_MAX_CHN_NUM - 1]
	VPSS	[0, VPSS_MAX_GRP_NUM - 1]	[0, VPSS_MAX_CHN_NUM - 1]
MOSAIC	VPSS	[0, VPSS_MAX_GRP_NUM - 1]	[0, VPSS_MAX_CHN_NUM - 1]
	VO	[0, VO_MAX_LAYER_NUM - 1]	[0, VO_MAX_CHN_NUM - 1]
LINE	VPSS	[0, VPSS_MAX_GRP_NUM - 1]	[0, VPSS_MAX_CHN_NUM - 1]

【注意事项】

- 对于VENC模块，COVER本质是纯色的OVERLAY，因此OVERLAY和COVER加起来的数量最大不能超过8个。
- 区域支持的功能

目前各种类型的区域支持的功能如[表1-2](#) 所示。

1.2 表1-2 RK356X region 支持的功能

支持的功能	OVERLAY			COVER			MOSAIC		LINE
	支持的模块	VO	VENC	VPSS	VO	VENC	VPSS	VO	VPSS
像素格式	BGRA8888、 BGRA5551	BGRA8888、 BGRA5551、 BGRA4444	BGRA8888、 BGRA5551、 NV12	RGB888	RGB888	RGB888	N/A	N/A	RGB888
叠加层次	支持	支持	支持	支持	支持	支持	支持	支持	N/A
位图填充	支持	支持	支持	N/A	N/A	N/A	N/A	N/A	N/A
叠加透明度	支持	不支持	支持	N/A	N/A	N/A	N/A	N/A	N/A
前景alpha范围	0~255	不支持	0~255	N/A	N/A	N/A	N/A	N/A	N/A
背景alpha范围	0~255	不支持	0~255	N/A	N/A	N/A	N/A	N/A	N/A
反色	不支持	不支持	不支持	N/A	N/A	N/A	N/A	N/A	N/A
QP保护	N/A	支持	N/A	N/A	N/A	N/A	N/A	N/A	N/A

1.3 表1-3 RV1106/RV1103 region 支持的模块信息

类型	支持模块	设备号取值范围	通道号取值范围	RGN 支持的最大数量
OVERLAY	VENC	0	[0, RK_VENC_MAX_CHN_NUM - 1] 8	
OVERLAY_EX	VPSS	0	[0, VPSS_MAX_GRP_NUM - 1] [0, RK_VENC_MAX_CHN_NUM]	8
OVERLAY_EX	VEMC	0	[0, RK_VENC_MAX_CHN_NUM - 1] 8	
COVER	VI	0	[0, RK_VI_MAX_CHN_NUM - 1] 12	
MOSAIC	VI	0	[0, RK_VI_MAX_CHN_NUM - 1] 12	

【注意事项】

- 对于VI模块，COVER和MOSAIC加起来的数量最大不能超过12个。
- 对于VPSS模块，通道为RK_VENC_MAX_CHN_NUM表示叠加在GRP通道上。
- 区域支持的功能

目前各种类型的区域支持的功能如[表1-4](#) 所示。

1.4 表1-4 RV1106/RV1103 region 支持的功能

支持的功能	OVERLAY	COVER	MOSAIC	OVERLAY_EX
支持的模块	VENC	VI	VI	VPSS, VENC
像素格式	BGRA8888、BGRA5551、BGRA4444、2BPP、1BPP	RGB888	N/A	BGRA8888
叠加层次	支持	支持	支持	支持
位图填充	支持	N/A	N/A	支持
叠加透明度	N/A	N/A	N/A	N/A
前景alpha范围	N/A	N/A	N/A	N/A
背景alpha范围	N/A	N/A	N/A	N/A
反色	支持	N/A	N/A	N/A
QP保护	支持	N/A	N/A	N/A

- 2BPP颜色配置说明

1.5 表1-5 2BPP格式颜色配置说明

2BPP(1B A + 1B RGB)	ALPHA 配置	RGB颜色配置
1 0	u32FgAlpha 取值范围[0 - 255]	u32ColorLUT[0] 格式RGB
1 1	u32FgAlpha 取值范围[0 - 255]	u32ColorLUT[1] 格式RGB
0 0	u32BgAlpha 取值范围[0 - 255]	u32ColorLUT[0] 格式RGB
0 1	u32BgAlpha 取值范围[0 - 255]	u32ColorLUT[1] 格式RGB

【注意】

- 2BPP格式有四种组合，通过使用 OVERLAY_CHN_ATTR_S 的成员 u32FgAlpha, u32BgAlpha , A (2BPP) = 0 使用 u32FgAlpha 的值，A(2BPP)=1 使用u32BgAlpha 值。
- 通过使用 u32ColorLUT[RGN_COLOR_LUT_NUM] 颜色表配置rgb不同值的颜色，RGB(2BPP) = 0， 使用 u32ColorLUT[0]，RGB(2BPP) = 1 使用u32ColorLUT[1]。
- u32ColorLUT 值: 0xffff00 高位配置 R, 0x00ff00 中位配置 G, 0x0000ff 低位配置 B。

1.6 表1-6 1BPP格式颜色配置说明

- 1BPP颜色配置说明

1BPP(1BIT RGB)	ALPHA 配置	RGB颜色配置
0	u32BgAlpha	u32ColorLUT[0] 格式RGB
1	u32FgAlpha	u32ColorLUT[1] 格式RGB

【注意】

- u32ColorLUT 值: 0xff0000 高位配置 R, 0x00ff00 中位配置 G, 0x0000ff 低位配置 B。

2. 使用步骤介绍

2.1 OVERLAY/OVERLAY_EX使用步骤

使用过程包含以下步骤：

- 步骤1：调用[RK_MPI_RGN_Create](#) 填充区域属性并创建区域。
- 步骤2：调用[RK_MPI_RGN_AttachToChn](#) 将画布绑定到通道特定区域上。
- 步骤3：调用[RK_MPI_RGN_GetCanvasInfo](#) 获取画布信息。
- 步骤4：将位图数据写入画布信息中。
- 步骤5：调用[RK_MPI_RGN_UpdateCanvas](#) 更新画布。
- 更新画布信息时重复步骤3~步骤5。
- 步骤6：不用时调用[RK_MPI_RGN_DetachFromChn](#) 将画布从绑定通道中解绑。
- 步骤7：调用[RK_MPI_RGN_Destroy](#) 销毁区域。

2.2 COVER使用步骤

使用过程包含以下步骤：

- 步骤1：调用[RK_MPI_RGN_Create](#) 填充遮挡区域相关属性并创建区域。
- 步骤2：调用[RK_MPI_RGN_AttachToChn](#) 将画布绑定到通道特定区域上。
- 步骤3：调用[RK_MPI_RGN_GetDisplayAttr](#) 获取遮挡区域信息。
- 步骤4：修改需要改变的参数（例如位置信息），调用[RK_MPI_RGN_SetDisplayAttr](#)更新信息。
- 更新遮挡区域信息时重复步骤3~步骤4。
- 步骤5：不用时调用[RK_MPI_RGN_DetachFromChn](#) 将画布从绑定通道中解绑。
- 步骤6：调用[RK_MPI_RGN_Destroy](#) 销毁区域。

3. 举例

```
coverHandle = 0;
stCoverAttr.enType = COVER_RGN;
s32Ret = RK_MPI_RGN_Create(coverHandle, &stCoverAttr);
if (RK_SUCCESS != s32Ret) {
    RK_LOGE("failed with %#x!", s32Ret);
    return RK_FAILURE;
}
stCoverChn.enModId = RK_ID_VENC;
stCoverChn.s32ChnId = 0;
stCoverChn.s32DevId = vencChn;
stCoverChnAttr.bShow = RK_TRUE;
```

```

stCoverChnAttr.enType = COVER_RGN;
stCoverChnAttr.unChnAttr.stCoverChn.enCoverType = AREA_RECT;
stCoverChnAttr.unChnAttr.stCoverChn.stRect.s32X = pstRgnCtx->stRegion.s32X;
stCoverChnAttr.unChnAttr.stCoverChn.stRect.s32Y = pstRgnCtx->stRegion.s32Y;
stCoverChnAttr.unChnAttr.stCoverChn.stRect.u32Width = pstRgnCtx->stRegion.u32Width;
stCoverChnAttr.unChnAttr.stCoverChn.stRect.u32Height = pstRgnCtx->stRegion.u32Height;
stCoverChnAttr.unChnAttr.stCoverChn.u32Color = 0xffffffff;
stCoverChnAttr.unChnAttr.stCoverChn.u32Layer = 0;
s32Ret = RK_MPI_RGN_AttachToChn(coverHandle, &stCoverChn, &stCoverChnAttr);
if (RK_SUCCESS != s32Ret) {
    RK_LOGE("failed with %#x!", s32Ret);
    goto AttachCover_failed;
}
stCoverChnAttr.unChnAttr.stCoverChn.stRect.s32X = 64;
stCoverChnAttr.unChnAttr.stCoverChn.stRect.s32Y = 64;
stCoverChnAttr.unChnAttr.stCoverChn.stRect.u32Width = 256;
stCoverChnAttr.unChnAttr.stCoverChn.stRect.u32Height = 256;
stCoverChnAttr.unChnAttr.stCoverChn.u32Color = 0x00f800;
stCoverChnAttr.unChnAttr.stCoverChn.u32Layer = 1;
s32Ret = RK_MPI_RGN_SetDisplayAttr(coverHandle, &stCoverChn, &stCoverChnAttr);
if (RK_SUCCESS != s32Ret) {
    RK_LOGE("failed with %#x!", s32Ret);
    goto exit;
}

RK_MPI_RGN_DetachFromChn(coverHandle, &stCoverChn);
RK_MPI_RGN_Destroy(coverHandle);

```

【注意事项】

- 输入区域均需要16位对齐。
- 起始位置X、Y坐标需要2位对齐。

4. API 参考

区域管理模块主要提供区域资源的控制管理功能，包括区域的创建、销毁，获取与设置区域属性，获取与设置区域的通道显示属性等。

该功能模块为用户提供以下API:

- [RK_MPI_RGN_Create](#): 创建区域。
- [RK_MPI_RGN_Destroy](#): 销毁区域。
- [RK_MPI_RGN_GetAttr](#): 获取区域属性。
- [RK_MPI_RGN_SetAttr](#): 设置区域属性。
- [RK_MPI_RGN_SetBitMap](#): 设置区域位图。
- [RK_MPI_RGN_AttachToChn](#): 将区域叠加到通道上。
- [RK_MPI_RGN_DetachFromChn](#): 将区域从通道中撤出。
- [RK_MPI_RGN_SetDisplayAttr](#): 设置区域的通道显示属性。
- [RK_MPI_RGN_GetDisplayAttr](#): 获取区域的通道显示属性。
- [RK_MPI_RGN_GetCanvasInfo](#): 获取区域的显示画布信息。
- [RK_MPI_RGN_UpdateCanvas](#): 更新区域的显示画布信息。

4.1 RK_MPI_RGN_Create

【描述】

创建区域。

【语法】

```
RK_S32 RK_MPI_RGN_Create(RGN\_HANDLE Handle, const RGN\_ATTR\_S *pstRegion);
```

【参数】

参数名称	描述	输入/输出
Handle	区域句柄号。 必须是未使用的 handle 号。 取值范围: [0, RGN_MAX_HANDLE_NUM)。	输入
pstRegion	区域属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 参见 RGN错误码 。

【需求】

- 头文件: rk_comm_rgn.h、rk_mpi_rgn.h
- 库文件: librockit.so

【注意】

- 该句柄由用户指定, 意义等同于 ID 号。
- 不支持重复创建。
- 区域属性必须合法, 具体约束参见[RGN_ATTR_S](#)。
- 区域属性指针不能为空。
- 创建 COVER、MOSAIC、LINE 时, 只需指定区域类型即可。其它的属性, 如区域位置, 层次等信息在调用 [RK_MPI_RGN_AttachToChn](#) 接口时指定。
- 创建区域时, 本接口只进行基本的参数的检查, 譬如: 最小宽高, 最大宽高等; 当区域 attach 到通道上时, 根据各通道模块支持类型的约束条件进行更加有针对性的参数检查, 譬如支持的像素格式等。
- 创建 OVERLAY 时, 支持配置画布数量, [OVERLAY_ATTR_S](#) 中成员 u32CanvasNum 代表画布数量, 当 u32CanvasNum = 0 时, 默认使用两块 buffer, 在 rv1106/1103 平台, 支持配置一块 buffer, 当 u32CanvasNum = 1 时, 仅支持 attach 到一个通道上。
- 在 rv1106/1103 平台上, OVERLAY 类型, 建议按照最大分辨率创建申请内存, 避免重复创建销毁导致内存碎片; 调用 [RK_MPI_RGN_GetAttr](#) 和 [RK_MPI_RGN_SetAttr](#) 动态调整区域宽高信息。
- 在 rv1106/1103 平台上, OVERLAY_EX 类型表示使用硬件 RGA 去叠加 osd, 使用方法和 OVERLAY 一致, 格式支持 ARGB8888、ABGR8888、RGBA8888、BGRA8888。

【相关主题】

- [RK_MPI_RGN_Destroy](#)
- [RK_MPI_RGN_GetAttr](#)
- [RK_MPI_RGN_SetAttr](#)

4.2 RK_MPI_RGN_Destroy

【描述】

销毁区域。

【语法】

```
RK_S32 RK_MPI_RGN_Destroy(RGN\_HANDLE Handle);
```

【参数】

参数名称	描述	输入/输出
Handle	区域句柄号。 取值范围: [0, RGN_MAX_HANDLE_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，参见 RGN错误码 。

【需求】

- 头文件：rk_comm_rgn.h、rk_mpi_rgn.h
- 库文件：librockit.so

【注意】

- 区域必须已创建。
- 调用该接口的过程中，不允许同时调用[RK_MPI_RGN_SetAttr](#)/[RK_MPI_RGN_SetBitMap](#)。

【相关主题】

- 无

4.3 RK_MPI_RGN_GetAttr

【描述】

获取区域属性。

【语法】

```
RK_S32 RK_MPI_RGN_GetAttr(RGN\_HANDLE Handle, RGN\_ATTR\_S *pstRegion);
```

【参数】

参数名称	描述	输入/输出
Handle	区域句柄号。 取值范围：[0, RGN_MAX_HANDLE_NUM)。	输入
pstRegion	区域属性指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，参见 RGN错误码 。

【需求】

- 头文件：rk_comm_rgn.h、rk_mpi_rgn.h
- 库文件：librockit.so

【注意】

- 区域必须已创建。
- 区域属性指针不能为空。

【相关主题】

- 无

4.4 RK_MPI_RGN_SetAttr

【描述】

设置区域属性。

【语法】

```
RK_S32 RK_MPI_RGN_SetAttr(RGN\_HANDLE Handle, const RGN\_ATTR\_S *pstRegion);
```

【参数】

参数名称	描述	输入/输出
Handle	区域句柄号。 取值范围: [0, RGN_MAX_HANDLE_NUM)。	输入
pstRegion	区域属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 参见 RGN错误码 。

【需求】

- 头文件: rk_comm_rgn.h、rk_mpi_rgn.h
- 库文件: librockit.so

【注意】

- 区域必须已创建。
- 区域属性指针不能为空。
- 调用该接口的过程中, 不允许同时调用 [RK_MPI_RGN_Destroy](#)。
- COVER、MOSAIC、LINE不支持此接口。
- 当调用[RK_MPI_RGN_SetBitMap](#)后调用该接口, 如果设置新的区域小于原有区域, 原有区域将会被销毁, 需要重新调用 [RK_MPI_RGN_SetBitMap](#)设置位图。

【相关主题】

- 无

4.5 RK_MPI_RGN_SetBitMap

【描述】

设置区域位图, 即对区域进行位图填充。

【语法】

```
RK_S32 RK_MPI_RGN_SetBitMap(RGN\_HANDLE Handle, const BITMAP_S *pstBitmap);
```

【参数】

参数名称	描述	输入/输出
Handle	区域句柄号。 取值范围: [0, RGN_MAX_HANDLE_NUM)。	输入
pstBitmap	位图属性指针。详细参见“系统控制”章节。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 参见 RGN错误码 。

【需求】

- 头文件: rk_comm_rgn.h、rk_mpi_rgn.h
- 库文件: librockit.so

【注意】

- 区域必须已创建。
- 支持位图的大小和区域的大小不一致。
- 位图从区域的(0,0)点开始加载。当位图比区域大时，将会自动将图像剪裁成区域大小。
- 位图的像素格式必须和区域的像素格式一致。
- 位图属性指针不能为空。
- 支持多次调用。
- 此接口只对 OVERLAY有效。
- 调用该接口的过程中，不允许同时调用 [RK_MPI_RGN_Destroy](#)。

【相关主题】

- 无

4.6 RK_MPI_RGN_AttachToChn

【描述】

将区域叠加到通道上。

【语法】

```
RK_S32 RK_MPI_RGN_AttachToChn(RGN\_HANDLE Handle, const MPP_CHN_S *pstChn, const RGN\_CHN\_ATTR\_S*pstChnAttr);
```

【参数】

参数名称	描述	输入/输出
Handle	区域句柄号。 取值范围：[0, RGN_MAX_HANDLE_NUM)。	输入
pstChn	通道结构体指针。具体描述请参见系统控制章节。	输入
pstChnAttr	区域通道显示属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，参见 RGN错误码 。

【需求】

- 头文件：rk_comm_rgn.h、rk_mpi_rgn.h
- 库文件：librockit.so

【注意】

- 区域必须已创建。
- 通道结构体指针不能为空。
- 区域通道显示属性指针不能为空。
- 不支持多次叠加。
- RK3588平台上，mainpath、selfpath两路通道在设置遮挡(cover)、马赛克(mosaic)时需要注意，二者是所有通道共享的功能，不可分别独立配置。设置其中一个会影响其他通道。最大支持8个窗口设置。
- RV1106/RV1103平台上，遮挡(cover)、马赛克(mosaic)支持打在每个通道上，也支持打在PIPE上，可根据需求设置。若作用在PIPE上，所有通道有效，窗口的位置与大小是根据sensor原始输出分辨率的大小为参考。若打在某个通道上，仅当前通道有效，其他通道无效，窗口位置与大小，以当前通道输出分辨率的大小为参考。最大支持12个窗口设置。

【相关主题】

- 无

4.7 RK_MPI_RGN_DetachFromChn

【描述】

将区域从通道中撤出。

【语法】

```
RK_S32 RK_MPI_RGN_DetachFromChn(RGN\_HANDLE Handle, const MPP_CHN_S *pstChn);
```

【参数】

参数名称	描述	输入/输出
Handle	区域句柄号。 取值范围: [0, RGN_MAX_HANDLE_NUM)。	输入
pstChn	通道结构体指针。具体描述请参见系统控制章节。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 参见 RGN错误码 。

【需求】

- 头文件: rk_comm_rgn.h、rk_mpi_rgn.h
- 库文件: librockit.so

【注意】

- 区域必须已创建。
- 通道结构体指针不能为空。
- 不支持多次调用。

【相关主题】

- 无

4.8 RK_MPI_RGN_SetDisplayAttr

【描述】

设置区域的通道显示属性。

【语法】

```
RK_S32 RK_MPI_RGN_SetDisplayAttr(RGN\_HANDLE Handle, const MPP_CHN_S *pstChn, const RGN\_CHN\_ATTR\_S *pstChnAttr);
```

【参数】

参数名称	描述	输入/输出
Handle	区域句柄号。 取值范围: [0, RGN_MAX_HANDLE_NUM)。	输入
pstChn	通道结构体指针。具体描述请参见系统控制章节。	输入
pstChnAttr	区域通道显示属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，参见 RGN错误码 。

【需求】

- 头文件：rk_comm_rgn.h、rk_mpi_rgn.h
- 库文件：librockit.so

【注意】

- 区域必须已创建。
- 建议先获取属性，再设置。
- 通道结构体指针不能为空。
- 区域通道显示属性指针不能为空。
- 区域必须先叠加到通道上。

【相关主题】

- 无

4.9 RK_MPI_RGN_GetDisplayAttr

【描述】

获取区域的通道显示属性。

【语法】

```
RK_S32 RK_MPI_RGN_GetDisplayAttr(RGN\_HANDLE Handle, const MPP_CHN_S *pstChn, RGN\_CHN\_ATTR\_S *pstChnAttr);
```

【参数】

参数名称	描述	输入/输出
Handle	区域句柄号。 取值范围：[0, RGN_MAX_HANDLE_NUM)。	输入
pstChn	通道结构体指针。具体描述请参见系统控制章节。	输入
pstChnAttr	区域通道显示属性指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，参见 RGN错误码 。

【需求】

- 头文件：rk_comm_rgn.h、rk_mpi_rgn.h
- 库文件：librockit.so

【注意】

- 区域必须已创建。
- 通道结构体指针不能为空。
- 区域通道显示属性指针不能为空。

【相关主题】

- 无

4.10 RK_MPI_RGN_GetCanvasInfo

【描述】

获取区域的显示画布信息。

【语法】

```
RK_S32 RK_MPI_RGN_GetCanvasInfo(RGN\_HANDLE Handle, RGN\_CANVAS\_INFO\_S *pstCanvasInfo);
```

【参数】

参数名称	描述	输入/输出
Handle	区域句柄号。 取值范围: [0, RGN_MAX_HANDLE_NUM)。	输入
RGN_CANVAS_INFO_S	区域显示画布信息。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 参见 RGN错误码 。

【需求】

- 头文件: rk_comm_rgn.h、rk_mpi_rgn.h
- 库文件: librockit.so

【注意】

- 区域必须已创建。
- 本接口与[RK_MPI_RGN_SetBitMap](#)功能类似, 主要用于实现overlay的位图数据导入, 相对于[RK_MPI_RGN_SetBitMap](#)接口, 该接口直接写入更新画布数据, 节省内存拷贝, 故更推荐使用这种方式。
- 本接口与[RK_MPI_RGN_UpdateCanvas](#)接口配套使用, 写入画布数据后, 调用[RK_MPI_RGN_UpdateCanvas](#)接口将画布真正更新到Attach的模块中去。
- 接口可以与[RK_MPI_RGN_SetBitMap](#)混用, 但不推荐混用。

【相关主题】

- 无

4.11 RK_MPI_RGN_UpdateCanvas

【描述】

更新区域的显示画布数据。

【语法】

```
RK_S32 RK_MPI_RGN_UpdateCanvas(RGN\_HANDLE Handle);
```

【参数】

参数名称	描述	输入/输出
Handle	区域句柄号。 取值范围: [0, RGN_MAX_HANDLE_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，参见 RGN错误码 。

【需求】

- 头文件：rk_comm_rgn.h、rk_mpi_rgn.h
- 库文件：librockit.so

【注意】

- 区域必须已创建。
- 本接口配合[RK_MPI_RGN_GetCanvasInfo](#)使用。主要用于画布内存数据更新之后，进行画布切换显示。
- 每次调用本接口前，都必须先获取画布信息，然后在调用本接口进行更新，否则不生效。
- 本接口仅支持OVERLAY类型的区域。
- 和[RK_MPI_RGN_GetCanvasInfo](#)配合使用时，需要注意更新画布需要线程同步，既调用[RK_MPI_RGN_GetCanvasInfo](#)后调用该接口，严格按照这一顺序，否则更新画布不成功。

【相关主题】

- 无

5. 数据类型

5.1 RGN_MIN_WIDTH

【说明】

定义区域最小宽度。

【定义】

RK356X：

```
#define RGN_MIN_WIDTH      16
```

【注意】

- 无

5.2 RGN_MIN_HEIGHT

【说明】

定义区域最小高度。

【定义】

RK356X：

```
#define RGN_MIN_HEIGHT     16
```

【注意】

- 无

5.3 RGN_COVER_MIN_X

【说明】

定义遮挡区域最小水平X。

【定义】

RK356X:

```
#define RGN_COVER_MIN_X      0
```

【注意】

- 无

5.4 RGN_COVER_MIN_Y

【说明】

定义遮挡区域最小垂直Y。

【定义】

RK356X:

```
#define RGN_COVER_MIN_Y      0
```

【注意】

- 无

5.5 RGN_COVER_MAX_X

【说明】

定义遮挡区域最大水平X。

【定义】

RK356X:

```
#define RGN_COVER_MAX_X     8192
```

【注意】

- 无

5.6 RGN_COVER_MAX_Y

【说明】

定义遮挡区域最小垂直Y。

【定义】

RK356X:

```
#define RGN_COVER_MAX_Y     8192
```

【注意】

- 无

5.7 RGN_COVER_MAX_WIDTH

【说明】

定义遮挡区域最大宽度。

【定义】

RK356X:

```
#define RGN_COVER_MAX_WIDTH    8192
```

【注意】

- 无

5.8 RGN_COVER_MAX_HEIGHT

【说明】

定义遮挡区域最大高度。

【定义】

RK356X:

```
#define RGN_COVER_MAX_HEIGHT   8192
```

【注意】

- 无

5.9 RGN_OVERLAY_MIN_X

【说明】

定义OVERLAY区域起始位置X坐标最小值。

【定义】

RK356X:

```
#define RGN_OVERLAY_MIN_X     0
```

【注意】

- 无

5.10 RGN_OVERLAY_MIN_Y

【说明】

定义OVERLAY区域起始位置Y坐标最小值。

【定义】

RK356X:

```
#define RGN_OVERLAY_MIN_Y     0
```

【注意】

- 无

5.11 RGN_OVERLAY_MAX_X

【说明】

定义OVERLAY区域起始位置X坐标最大值。

【定义】

RK356X:

```
#define RGN_OVERLAY_MAX_X      8192
```

【注意】

- 无

5.12 RGN_OVERLAY_MAX_Y

【说明】

定义OVERLAY区域起始位置Y坐标最大值。

【定义】

RK356X:

```
#define RGN_OVERLAY_MAX_Y      8192
```

【注意】

- 无

5.13 RGN_OVERLAY_MAX_WIDTH

【说明】

定义OVERLAY区域最大宽度。

【定义】

RK356X:

```
#define RGN_OVERLAY_MAX_WIDTH    8192
```

【注意】

- 无

5.14 RGN_OVERLAY_MAX_HEIGHT

【说明】

定义OVERLAY区域最大高度。

【定义】

RK356X:

```
#define RGN_OVERLAY_MAX_HEIGHT   8192
```

【注意】

- 无

5.15 RGN_MOSAIC_MIN_X

【说明】

定义马赛克区域起始位置X坐标最小值。

【定义】

RK356X:

```
#define RGN_MOSAIC_MIN_X      0
```

【注意】

- 无

5.16 RGN_MOSAIC_MIN_Y

【说明】

定义马赛克区域起始位置Y坐标最小值。

【定义】

RK356X:

```
#define RGN_MOSAIC_MIN_Y      0
```

【注意】

- 无

5.17 RGN_MOSAIC_MAX_X

【说明】

定义马赛克区域起始位置X坐标最大值。

【定义】

RK356X:

```
#define RGN_MOSAIC_MAX_X     8192
```

【注意】

- 无

5.18 RGN_MOSAIC_MAX_Y

【说明】

定义马赛克区域起始位置Y坐标最大值。

【定义】

RK356X:

```
#define RGN_MOSAIC_MAX_Y     8192
```

【注意】

- 无

5.19 RGN_MOSAIC_MIN_WIDTH

【说明】

定义马赛克区域最小宽度。

【定义】

RK356X:

```
#define RGN_MOSAIC_MIN_WIDTH    32
```

【注意】

- 无

5.20 RGN_MOSAIC_MIN_HEIGHT

【说明】

定义马赛克区域最小高度。

【定义】

RK356X:

```
#define RGN_MOSAIC_MIN_HEIGHT    32
```

【注意】

- 无

5.21 RGN_MOSAIC_MAX_WIDTH

【说明】

定义马赛克区域最大宽度。

【定义】

RK356X:

```
#define RGN_MOSAIC_MAX_WIDTH    8192
```

【注意】

- 无

5.22 RGN_MOSAIC_MAX_HEIGHT

【说明】

定义马赛克区域最大高度。

【定义】

RK356X:

```
#define RGN_MOSAIC_MAX_HEIGHT    8192
```

【注意】

- 无

5.23 RGN_LINE_MIN_X

【说明】

定义画线的最小水平坐标。

【定义】

RK356X:

```
#define RGN_LINE_MIN_X      0
```

【注意】

- 无

5.24 RGN_LINE_MIN_Y

【说明】

定义画线的最小垂直坐标。

【定义】

RK356X:

```
#define RGN_LINE_MIN_Y      0
```

【注意】

- 无

5.25 RGN_LINE_MAX_X

【说明】

定义画线的最大水平坐标。

【定义】

RK356X:

```
#define RGN_LINE_MAX_X     8192
```

【注意】

- 无

5.26 RGN_LINE_MAX_Y

【说明】

定义画线的最大垂直坐标。

【定义】

RK356X:

```
#define RGN_LINE_MAX_Y     8192
```

【注意】

- 无

5.27 RGN_LINE_MIN_THICK

【说明】

定义画线的最小线宽。

【定义】

RK356X:

```
#define RGN_LINE_MIN_THICK     1
```

【注意】

- 无

5.28 RGN_LINE_MAX_THICK

【说明】

定义画线的最大线宽。

【定义】

RK356X:

```
#define RGN_LINE_MAX_THICK    32
```

【注意】

- 无

5.29 RGN_ALIGN

【说明】

定义区域对齐方式。

【定义】

RK356X:

```
#define RGN_ALIGN      16
```

【注意】

- 无

5.30 RGN_MAX_HANDLE_NUM

【说明】

定义最大的RGN句柄个数。

【定义】

```
#define RGN_MAX_HANDLE_NUM   128
```

【注意】

- 无

5.31 RGN_MAX_BUF_NUM

【说明】

定义最大的RGN画布数量。

【定义】

```
#define RGN_MAX_BUF_NUM      2
```

【注意】

- 无

5.32 RGN_COLOR_LUT_NUM

【说明】

定义RGB颜色查表个数。

【定义】

```
#define RGN_COLOR_LUT_NUM      2
```

【注意】

- 无

5.33 RGN_HANDLE

【说明】

定义区域句柄。

【定义】

```
typedef RK_U32 RGN_HANDLE;
```

【注意】

- 无

5.34 RGN_TYPE_E

【说明】

定义区域类型。

【定义】

```
typedef enum rkRGN_TYPE_E {
    OVERLAY_RGN = 0,
    OVERLAY_EX_RGN,
    COVER_RGN,
    MOSAIC_RGN,
    LINE_RGN,
    RGN_BUTT
} RGN_TYPE_E;
```

【成员】

成员名称	描述
OVERLAY_RGN	视频叠加区域。
OVERLAY_EX_RGN	视频叠加区域。
COVER_RGN	视频遮挡区域。
MOSAIC_RGN	MOSAIC 视频区域。
LINE_RGN	视频画线区域。

【注意】

- 当区域模块为VENC时，同一个通道多个区域建议只使用OVERLAY_RGN或者COVER_RGN的一种类型（VENC同一个通道使用同一个调色板，当区域类型不一致时，可能会导致颜色出现变化）。

5.35 RGN_COORDINATE_E

【说明】

定义坐标类型。

【定义】

```
typedef enum rkRGN_COORDINATE_E {
    RGN_ABS_COOR = 0,
    RGN_RATIO_COOR
} RGN_COORDINATE_E;
```

【成员】

成员名称	描述
RGN_ABS_COOR	坐标类型为绝对坐标。
RGN_RATIO_COOR	坐标类型为相对坐标。

【注意】

- 默认类型为绝对坐标。
- 目前相对坐标，只支持区域类型为 COVER 且叠加在 VO 上的相对坐标配置。

5.36 OVERLAY_ATTR_S

【说明】

定义通道叠加区域属性结构体。

【定义】

```
typedef struct rkOVERLAY_ATTR_S {
    PIXEL_FORMAT_E enPixelFmt;
    SIZE_S stSize;
    RK_U32 u32CanvasNum;
    RK_U32 u32ClutNum;
    RK_U32 u32Clut[RGN_CLUT_NUM];
    VIDEO_PROC_DEV_TYPE_E enVProcDev;
} OVERLAY_ATTR_S;
```

【成员】

成员名称	描述
enPixelFmt	像素格式。具体描述请参见“系统控制”章节。 取值范围： 参照 表1-2 。
stSize	区域的高宽。 取值范围： 当 overlay 绑定到 VENC 通道时，支持： 宽度：[RGN_MIN_WIDTH , RGN_OVERLAY_MAX_WIDTH]，要求以 16 位对齐。 高度：[RGN_MIN_HEIGHT , RGN_OVERLAY_MAX_HEIGHT]，要求以 16 位对齐。
u32CanvasNum	区域的可用画布数量 取值范围： [1, RGN_MAX_BUF_NUM] 静态属性。
u32ClutNum	定义 RGB 颜色值画板颜色个数，目前只针对BGRA8888格式有效。 取值范围：[0,255]。 0：使用系统默认画板； 其他：使用u32Clut数组定义的前u32ClutNum个颜色值作为画板。
u32Clut	定义 RGB 颜色值画板，目前只针对BGRA8888格式有效。 当用户设置使用该画板后，venc overlay会使用该画板的前u32ClutNum个颜色进行查询颜色。 取值范围：[0x0~0xFFFFFFFF]。32位从高位到低位分别为透明度（alpha/24-31bit）、红色（red/16-23bit）、绿色（green/8-15bit）、蓝色（blue/0-7bit）。
enVProcDev	设备类型，支持的硬件设备分别是 GPU 和 RGA，目前只支持 attach 到 VENC 模块做 OVERLAY_EX 类型 OSD

【注意】

- u32CanvasNum如果设置小于1，将会强制被设置为2。
- u32CanvasNum如果大于[RGN_MAX_BUF_NUM](#)，将会被设置为[RGN_MAX_BUF_NUM](#)。
- u32CanvasNum推荐为2，为1时，在更新画布的同时可能画布被执行贴图，会造成数据不同步。
- u32CanvasNum在第一次设置属性时被确定，之后不可更改。
- 定义u32ClutNum不为0时，需要定义数组u32Clut前u32ClutNum个数的颜色值作为画板查询。
- 当区域模块为VENC并且类型为overlay时，建议配置u32ClutNum及u32Clut参数并且选择BGRA8888格式进行调色板的设置。多个区域的调色板需要使用同样的调色板，所有区域的调色板以最后一个区域的设置值为准。
- 当区域模块为VENC并且类型为overlay_ex时，需要设置设备类型，不同芯片种类有区分，RGA 只支持 ARGB8888、ABGR8888、RGBA8888、BGRA8888 格式，不支持动态调整 OSD 的位置。
- 当区域模块为VENC时，多个区域的像素格式需要使用同一个格式。

芯片名称	硬件设备类型
RV1109/RV1126	RGA
RK356X	GPU、RGA
RK3588	GPU、RGA

5.37 OVERLAY_CHN_ATTR_S

【说明】

定义通道叠加区域的通道显示属性。

【定义】

```

typedef struct rkOVERLAY_CHN_ATTR_S {
    POINT_S stPoint;
    RK_U32 u32FgAlpha;
    RK_U32 u32BgAlpha;
    RK_U32 u32Layer;
    OVERLAY_QP_INFO_S stQpInfo;
    RK_U32 u32ColorLUT[RGN_COLOR_LUT_NUM];
} OVERLAY_CHN_ATTR_S;

```

【成员】

成员名称	描述
stPoint	区域位置。 取值范围： 水平坐标X: [RGN_OVERLAY_MIN_X, RGN_OVERLAY_MAX_X]。 垂直坐标Y: [RGN_OVERLAY_MIN_Y, RGN_OVERLAY_MAX_Y]。
u32FgAlpha	Alpha 位为 1 的像素点的透明度。也称前景 Alpha。 取值范围: [0, 255]。 取值越小，越透明。图像高度。
u32BgAlpha	Alpha 位为 0 的像素点的透明度。也称背景 Alpha。 取值范围: [0, 255]。 取值越小，越透明。图像宽度。
u32Layer	区域层次。取值范围: [0, 7]。值越大，层次越高。
stQpInfo	此区域编码时使用的 qp 值，仅支持 VENC。 动态属性。
u32ColorLUT[RGN_COLOR_LUT_NUM]	定义RGB颜色值。 个数: RGN_COLOR_LUT_NUM 范围: [0x0, 0xFFFF]. 动态属性。

【注意】

- 区域内存信息为 RRK_FMT_BGRA5551 格式时，将会扩展 Alpha 值。当 Alpha 位为 1 时，使用 u32FgAlpha 进行透明度叠加；当 Alpha 位为 0 时，使用 u32BgAlpha 进行透明度叠加。当区域模块为 VENC 时，单个通道的 OVERLAY 与 COVER 共用透明度数值，改变通道的某一个区域的透明度，会使同一个通道不同区域（包括 OVERLAY 与 COVER）透明度同步改变。
- 0 表示全透明；255 表示不透明。
- 当区域模块为 VENC 时，OVERLAY 与 COVER 的 u32Layer 信息共用一个，即 OVERLAY 与 COVER 加起来不能超过 8 个 u32Layer。并且 u32Layer 互斥，例如当 OVERLAY 使用 u32Layer 为 0 时，COVER 不能再用 u32Layer 为 0 的区域。
- 当区域模块为 VENC 时，u32Layer 每个层次只能有一个 OVERLAY 区域信息，同一个 u32Layer 后设置的会覆盖之前设置的。
- u32ColorLUT[RGN_COLOR_LUT_NUM]，RGB 颜色查表属性，用于在 VENC 上叠加 OVERLAY 时设置 RGN 颜色值。主要用二值化 buff 操作。

5.38 RGN_AREA_TYPE_E

【说明】

定义 cover 类型和 mosaic 类型的区域类型。

【定义】

```

typedef enum rkRGN_AREA_TYPE_E {
    AREA_RECT = 0,
    AREA_QUAD_RANGLE,
    AREA_BUTT
} RGN_AREA_TYPE_E;

```

【成员】

成员名称	描述
AREA_RECT	矩形区域。
AREA_QUAD_RANGLE	四边形区域。

5.39 COVER_CHN_ATTR_S

【说明】

定义遮挡区域的通道显示属性。

【定义】

```
typedef struct rkCOVER_CHN_ATTR_S {
    RECT_S stRect;
    RK_U32 u32Color;
    RK_U32 u32Layer;
    RGN_COORDINATE_E enCoordinate;
} COVER_CHN_ATTR_S;
```

【成员】

成员名称	描述
stRect	区域位置，宽高。 位置取值范围： 水平位置X： [RGN_COVER_MIN_X, RGN_COVER_MAX_X]，要求以16位对齐。 垂直位置Y： [RGN_COVER_MIN_Y, RGN_COVER_MAX_Y]，要求以16位对齐。 宽高取值范围： 宽度：[RGN_MIN_WIDTH, RGN_COVER_MAX_WIDTH]，要求以16位对齐。 高度：[RGN_MIN_HEIGHT, RGN_COVER_MAX_HEIGHT]，要求以16位对齐。 动态属性。
u32Color	区域颜色。以ARGB888定义颜色值。
u32Layer	区域层次。取值范围：[0, 7]。 动态属性。
enCoordinate	区域坐标类型。

【注意】

- 当区域模块为VENC时，OVERLAY与COVER的u32Layer信息共用一个，即OVERLAY与COVER加起来不能超过8个u32Layer。并且u32Layer互斥，例如当OVERLAY使用u32Layer为0时，COVER不能再用u32Layer为0的区域。
- 当区域模块为VENC时，u32Layer每个层次只能有一个COVER区域信息，同一个u32Layer后设置的会覆盖之前设置的。
- 当区域模块为VPSS，并且硬件设备类型为硬件VPSS时，u32Color才支持alpha叠加，取值范围0~15，0为全透。
- RK3588平台上，X与Y要求以2位对齐，宽、高要求以8位对齐。
- RV1106平台上，X与Y要求以2位对齐，宽、高要求以2位对齐。

5.40 MOSAIC_BLK_SIZE_E

【说明】

定义 mosaic 类型的块大小。

【定义】

```

typedef enum rkMOSAIC_BLK_SIZE_E {
    MOSAIC_BLK_SIZE_8 = 0,
    MOSAIC_BLK_SIZE_16,
    MOSAIC_BLK_SIZE_32,
    MOSAIC_BLK_SIZE_64,
    MOSAIC_BLK_SIZE_BUTT
} MOSAIC_BLK_SIZE_E;

```

【成员】

成员名称	描述
MOSAIC_BLK_SIZE_8	8*8 大小。
MOSAIC_BLK_SIZE_16	16*16 大小。
MOSAIC_BLK_SIZE_32	32*32 大小。
MOSAIC_BLK_SIZE_64	64*64 大小。

【注意】

- 无

5.41 MOSAIC_CHN_ATTR_S

【说明】

定义马赛克区域的通道显示属性。

【定义】

```

typedef struct rkMOSAIC_CHN_ATTR_S {
    RGN_AREA_TYPE_E enMosaicType;
    union {
        RECT_S stRect;
        RGN_QUADRANGLE_S stQuadRangle;
    };
    MOSAIC_BLK_SIZE_E enBlkSize;
    RK_U32 u32Layer;
} MOSAIC_CHN_ATTR_S;

```

【成员】

成员名称	描述
enMosaicType	马赛克区域类型。
stRect	马赛克矩形区域位置。
stQuadRangle	马赛克四边形区域位置。
enBlkSize	马赛克显示类型。
u32Layer	区域层次。

【注意】

- enMosaicType 马赛克区域类型为AREA_RECT 时，配置stRect 马赛克矩形区域位置；为AREA_QUAD_RANGLE 时，配置stQuadRangle 马赛克四边形区域位置。
- stQuadRangle 马赛克四边形区域位置顺序如下：
point0 ----- point1
| |
| |
point2 ----- point3

- 各模块支持情况：

芯片类型	模块	是否支持	支持条件
RK356x	VENC	否	
	VPSS	是	硬件设备类型为 GPU
RK3588	VENC	否	
	VPSS	是	硬件设备类型为 GPU
RV1126/ RV1109	VPSS	否	
	VENC	否	

5.42 LINE_CHN_ATTR_S

【说明】

定义画线区域的通道显示属性。

【定义】

```
typedef struct rkLINE_CHN_ATTR_S {
    RK_U32 u32Thick;
    RK_U32 u32Color;
    POINT_S stStartPoint;
    POINT_S stEndPoint;
} LINE_CHN_ATTR_S;
```

【成员】

成员名称	描述
u32Thick	线宽。 取值范围: [RGN_LINE_MIN_THICK, RGN_LINE_MAX_THICK]
u32Color	画线的颜色, 以RGB值定义。
stStartPoint	画线的起始点。
stEndPoint	画线的终止点。

【注意】

- 无

5.43 RGN_ATTR_U

【说明】

定义区域属性联合体。

【定义】

```
typedef union rkRGN_ATTR_U {
    OVERLAY_ATTR_S    stOverlay;
} RGN_ATTR_U;
```

【成员】

成员名称	描述
stOverlay	通道叠加区域属性。

【注意】

- 无

5.44 RGN_CHN_ATTR_U

【说明】

定义区域通道显示属性联合体。

【定义】

```
typedef union rkRGN_CHN_ATTR_U {
    OVERLAY_CHN_ATTR_S    stOverlayChn;
    COVER_CHN_ATTR_S     stCoverChn;
    MOSAIC_CHN_ATTR_S   stMosaicChn;
    LINE_CHN_ATTR_S     stLineChn;
} RGN_CHN_ATTR_U;
```

【成员】

成员名称	描述
stOverlayChn	通道叠加区域通道显示属性。
stCoverChn	遮挡区域通道显示属性。
stMosaicChn	马赛克显示属性。
stLineChn	画线区域的显示属性。

【注意】

- 无

5.45 RGN_ATTR_S

【说明】

定义区域属性结构体。

【定义】

```
typedef struct rkRGN_ATTR_S {
    RGN_TYPE_E enType;
    RGN_ATTR_U unAttr;
} RGN_ATTR_S;
```

【成员】

成员名称	描述
enType	区域类型。
unAttr	区域属性。

【注意】

- 无

5.46 RGN_CHN_ATTR_S

【说明】

定义马赛克区域的通道显示属性。

【定义】

```
typedef struct rkRGN_CHN_ATTR_S {  
    RK_BOOL      bShow;  
    RGN_TYPE_E   enType;  
    RGN_CHN_ATTR_U unChnAttr;  
} RGN_CHN_ATTR_S;
```

【成员】

成员名称	描述
bShow	区域是否显示。 取值范围：RK_TRUE 或者 RK_FALSE。 动态属性。
enType	区域类型。 静态属性。
unChnAttr	区域通道显示属性。

【注意】

- 无

5.47 OVERLAY_QP_INFO_S

【说明】

定义overlay区域的QP保护信息。

【定义】

```
typedef struct rkOVERLAY_QP_INFO {  
    RK_BOOL bEnable;  
    RK_BOOL bAbsQp;  
    RK_BOOL bForceIntra;  
    RK_S32 s32Qp;  
} OVERLAY_QP_INFO_S;
```

【成员】

成员名称	描述
bEnable	区域是否开启QP保护。 取值范围：RK_TRUE 或者 RK_FALSE。 动态属性。
bAbsQp	区域QP保护类型。 RK_TRUE： 绝对QP。 RK_FALSE: 相对QP。 (相对未ROI区域的QP变化值) 动态属性。
bForceIntra	区域是否强制编码为Intra宏块。 取值范围：RK_TRUE 或者 RK_FALSE。 动态属性。
s32Qp	区域的QP值。 取值范围：绝对QP开启时：[0, 51]。 相对QP时：[-51, 51]。 动态属性。

【注意】

- 此结构体仅在贴图到VENC时方可生效，其他模块时不生效。

5.48 RGN_CANVAS_INFO_S

【说明】

定义画布信息。

【定义】

```
typedef struct rkRGN_CANVAS_INFO_S {
    MB_BLK canvasBlk;
    RK_U64 u64VirAddr;
    SIZE_S stSize;
    RK_U32 u32VirWidth;
    RK_U32 u32VirHeight;
    PIXEL_FORMAT_E enPixelFmt;
} RGN_CANVAS_INFO_S;
```

【成员】

成员名称	描述
canvasBlk	画布数据的MB内存。
u64VirAddr	画布数据的虚地址。
stSize	画布的大小。
u32VirWidth	画布的虚宽。
u32VirHeight	画布的虚高。
enPixelFmt	画布的像素格式。

【注意】

- 无

6. RGN错误码

区域管理 API RGN 错误码如表 1-3 所示。

6.1 表1-3 区域管理 API RGN 错误码

错误代码	宏定义	描述
0xA0038001	RK_ERR_RGN_INVALID_DEVID	设备 ID 超出合法范围
0xA0038002	RK_ERR_RGN_INVALID_CHNID	通道组号错误或无效区域句柄
0xA0038003	RK_ERR_RGN_ILLEGAL_PARAM	参数超出合法范围
0xA0038004	RK_ERR_RGN_EXIST	重复创建已存在的设备、通道或资源
0xA0038005	RK_ERR_RGN_UNEXIST	试图使用或者销毁不存在的设备、通道或者资源
0xA0038006	RK_ERR_RGN_NULL_PTR	函数参数中有空指针
0xA0038007	RK_ERR_RGN_NOT_CONFIG	模块没有配置
0xA0038008	RK_ERR_RGN_NOT_SUPPORT	不支持的参数或者功能
0xA0038009	RK_ERR_RGN_NOT_PERM	该操作不允许，如试图修改静态配置参数
0xA003800C	RK_ERR_RGN_NOMEM	分配内存失败，如系统内存不足
0xA003800D	RK_ERR_RGN_NOBUF	分配缓存失败，如申请的数据缓冲区太大
0xA003800E	RK_ERR_RGN_BUF_EMPTY	缓冲区中无数据
0xA003800F	RK_ERR_RGN_BUF_FULL	缓冲区中数据满
0xA0038010	RK_ERR_RGN_NOTREADY	系统没有初始化或没有加载相应模块
0xA0038011	RK_ERR_RGN_BADADDR	地址非法
0xA0038012	RK_ERR_RGN_BUSY	系统忙

几何畸变矫正子系统

[基本概念](#)
[功能描述](#)
[硬件规格](#)
[API 参考](#)
[数据类型](#)
[GDC 错误码](#)

1. 基本概念

1.1 task

对一幅图像完成具体的一个或多个操作，比如全景矫正、获取点链信息等。GDC默认的最大的task数为200。

1.2 job

GDC管理task的结构，一个job里可以包含多个task，GDC保证task按照添加到job的顺序一次性提交硬件执行。GDC默认的最大的job数为128。

1.3 HANDLE

任务句柄，标识一个 job。

2. 功能描述

GDC的功能有鱼眼矫正、获取点链信息。

2.1 鱼眼矫正

GDC 支持对一幅图像进行鱼眼矫正处理。目前仅支持从VO模块输出鱼眼矫正数据进行显示，不支持GDC直接输出处理数据。

2.2 获取点链信息

GDC 支持从矫正图像坐标点获取原始图像坐标点（支持单点及多点）、从矫正图像坐标点获取处理后的全景图像坐标点（支持单点及多点）。

3. 硬件规格

3.1 GDC 硬件规格

3.1.1 鱼眼矫正输入输出图像规格

芯片	图像格式支持	图像限制
RK3588	RK_FMT_YUV420SP RK_FMT_YUV420SP_VU RK_FMT_RGB888 RK_FMT_BGR888 RK_FMT_BGRA8888 RK_FMT_RGBA8888	输入图像最大8192x8192，输入最小1920x1080； 输出图像最大8192x8192，输出图像最小 640x360。

4. API 参考

该功能模块为用户提供以下 MPI:

- [RK_MPI_GDC_BeginJob](#): 启动一个 job。
- [RK_MPI_GDC_EndJob](#): 提交一个 job, GDC 开始对这个 job 进行处理。
- [RK_MPI_GDC_CancelJob](#): 取消一个 job。
- [RK_MPI_GDC_StopJob](#): 退出一个 job, 关闭 job 所有 task 资源。
- [RK_MPI_GDC_SetConfig](#): 设置鱼眼镜头的配置。
- [RK_MPI_GDC_AddCorrectionTask](#): 往一个已经启动的 job 添加鱼眼矫正 task。
- [RK_MPI_GDC_AddCorrectionExTask](#): 往一个已经开启的 job 里添加鱼眼矫正 task (XY 模式)。
- [RK_MPI_GDC_FisheyePosQueryDst2Src](#): 根据鱼眼矫正输出图像坐标点查找源图像坐标点。
- [RK_MPI_GDC_FisheyePosQueryDst2SrcArray](#): 根据鱼眼矫正输出图像坐标点数组查找源图像坐标点数组。
- [RK_MPI_GDC_FisheyePosQueryDst2Pano](#): 根据鱼眼矫正输出图像坐标点查找全景图像坐标点。
- [RK_MPI_GDC_FisheyePosQueryDst2PanoArray](#): 根据鱼眼矫正输出图像坐标点数组查找全景图像坐标点数组。

4.1 RK_MPI_GDC_BeginJob

【描述】

启动一个 job。

【语法】

```
RK_S32 RK_MPI_GDC_BeginJob(GDC\_HANDLE *phHandle);
```

【参数】

参数名称	描述	输入\输出
phHandle	返回的 job handle	输出

【返回值】

返回值	描述
0	成功
非0	失败, 见 GDC 错误码

【注意】

- 可一次启动多个 job, 但必须判断 [RK_MPI_GDC_BeginJob](#) 函数返回成功后才能使用 phHandle 返回的 HANLDE。
- phHandle 不能为空指针或非法指针。

【举例】

```
RK_S32 s32Ret = RK_SUCCESS;
GDC_HANDLE hHandle;
GDC_TASK_ATTR_S stTask;
s32Ret = RK_MPI_GDC_BeginJob(&hHandle);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_GDC_BeginJob err:0x%x!", s32Ret);
    goto __FAILED;
}
s32Ret = RK_MPI_GDC_AddCorrectionTask(hHandle, &ctx->stTask, &ctx->stFisheyeAttr);
if (s32Ret != RK_SUCCESS) {
    RK_MPI_GDC_CancelJob(hHandle);
    goto __FAILED;
}
s32Ret = RK_MPI_GDC_EndJob(hHandle);
if (s32Ret != RK_SUCCESS) {
    RK_MPI_GDC_CancelJob(hHandle);
    RK_LOGE("RK_MPI_GDC_EndJob err:0x%x!", s32Ret);
```

```

    goto __FAILED;
}
RK_MPI_GDC_StopJob(hHandle);

```

4.2 RK_MPI_GDC_EndJob

【描述】

提交一个 job。

【语法】

```
RK_S32 RK_MPI_GDC_EndJob(GDC\_HANDLE hHandle);
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE	输入

【返回值】

返回值	描述
0	成功
非0	失败, 见 GDC错误码

【注意】

- 如果此接口返回失败, 必须调用 [RK_MPI_GDC_CancelJob](#) 接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job

4.3 RK_MPI_GDC_CancelJob

【描述】

取消一个 job。

【语法】

```
RK_S32 RK_MPI_GDC_CancelJob(GDC\_HANDLE hHandle);
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE	输入

【返回值】

返回值	描述
0	成功
非0	失败, 见 GDC错误码

【注意】

- hHandle 标识的 job 必须是已经启动的 job。

4.4 RK_MPI_GDC_StopJob

【描述】

退出一个job，关闭job所有task资源。

【语法】

```
RK_S32 RK_MPI_GDC_StopJob(GDC\_HANDLE hHandle);
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE	输入

【返回值】

返回值	描述
0	成功
非0	失败，见 GDC错误码

【注意】

- hHandle 标识的 job 必须是已经启动的 job。
- 在不使用job时，需要调用此函数进行job相关资源的销毁。
- 这个接口调用后，对应job开启的task资源及对应的dumpsys gcd的task参数信息会被清空。

4.5 RK_MPI_GDC_SetConfig

【描述】

设置鱼眼镜头的配置。

【语法】

```
RK_S32 RK_MPI_GDC_SetConfig(GDC\_HANDLE hHandle, const FISHEYE_JOB_CONFIG_S *pstJobConfig);
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE	输入
pstJobConfig	鱼眼镜头的配置	

【返回值】

返回值	描述
0	成功
非0	失败，见 GDC错误码

【注意】

- 此接口功能暂未实现。

4.6 RK_MPI_GDC_AddCorrectionTask

【描述】

往一个已经启动的 job 添加鱼眼矫正 task。

【语法】

```
RK_S32 RK_MPI_GDC_AddCorrectionTask(GDC_HANDLE hHandle, const GDC_TASK_ATTR_S *pstTask,  
FISHEYE_ATTR_S *pstFisheyeAttr);
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE	输入
pstTask	GDC task 属性指针	输入
pstFisheyeAttr	FISHEYE 纠正区域属性配置指针	输入

【返回值】

返回值	描述
0	成功
非0	失败, 见 GDC错误码

【注意】

- 如果此接口返回失败，必须调用[RK_MPI_GDC_CancelJob](#)接口取消掉 hHandle 标识的 job。否则会导致 hHandle 标识的 job 不能再被循环利用。
- hHandle 标识的 job 必须是已经启动的 job。
- 区域矫正模式FISHEYE_VIEW_MODE_E设置为FISHEYE_NO_TRANSFORMATION时，对应区域会根据输入图像及输出图像区域配置做缩放处理。需要显示原图缩放效果的可以将对应区域参数进行设置，不需要再通过VGS等模块处理原图缩放。
- 鱼眼矫正支持三种矫正模式，360 全景模式，180 全景模式，normal 模式。360 全景模式支持地装和顶装；180 全景模式支持壁装；normal 模式支持地装、顶装、壁装。
- 目前支持从VO输出GDC结果（直接送VO显示输出）或者从GDC获取畸变矫正数据（保存文件、送VENC编码等场景）
 - 直接从VO输出GDC结果配置：

输入数据及输出数据VIDEO_FRAME_INFO_S的stVFrame.pMbBlk赋值为同一个，并且最终赋值给VO进行显示使用，具体使用见举例--GDC结果直接送VO显示。

◦ 从GDC获取畸变矫正数据配置：

输入数据及输出数据VIDEO_FRAME_INFO_S的stVFrame.pMbBlk赋值为不同的pMbBlk，具体使用见举例--从GDC获取畸变矫正数据。

【举例】

GDC结果直接送VO显示（输入输出pMbBlk为同一个）：

```
// imgin  
ctx->stTask.stImgIn.stVFrame.pMbBlk = RK_MPI_MB_GetMB(ctx->inPool, ctx->u32SrcSize, RK_TRUE);  
// imgout  
ctx->stTask.stImgOut.stVFrame.pMbBlk = ctx->stTask.stImgIn.stVFrame.pMbBlk;  
// read image  
pSrcData = RK_MPI_MB_Handle2VirAddr(ctx->stTask.stImgIn.stVFrame.pMbBlk);  
s32Ret = read_image(reinterpret_cast<RK_U8*>(pSrcData), ctx->u32SrcWidth, ctx->u32SrcHeight, ctx->u32SrcVirWidth, ctx->u32SrcVirHeight, ctx->s32SrcPixelFormat, pFile);  
s32Ret = RK_MPI_GDC_BeginJob(&hHandle);  
if (s32Ret != RK_SUCCESS) {  
    RK_LOGE("RK_MPI_GDC_BeginJob err:0x%x!", s32Ret);  
    goto __FAILED;
```

```

}

s32Ret = RK_MPI_GDC_AddCorrectionTask(hHandle, &ctx->stTask, &ctx->stFisheyeAttr);
if (s32Ret != RK_SUCCESS) {
    RK_MPI_GDC_CancelJob(hHandle);
    RK_LOGE("RK_MPI_GDC_AddCorrectionTask err:0x%x!", s32Ret);
    goto __FAILED;
}
s32Ret = RK_MPI_GDC_EndJob(hHandle);
if (s32Ret != RK_SUCCESS) {
    RK_MPI_GDC_CancelJob(hHandle);
    RK_LOGE("RK_MPI_GDC_EndJob err:0x%x!", s32Ret);
    goto __FAILED;
}
// gcd out show to vo
stFrame.stVFrame = ctx->stTask.stImgOut.stVFrame;
s32Ret = RK_MPI_VO_SendFrame(ctx->s32VoLayer, ctx->s32VoChn, &stFrame, -1);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_VO_SendFrame failed ret:0x%x", s32Ret);
    RK_MPI_VO_DestroyGraphicsFrameBuffer(ctx->stTask.stImgIn.stVFrame.pMbBlk);
}
RK_MPI_MB_ReleaseMB(ctx->stTask.stImgIn.stVFrame.pMbBlk);

```

从GDC获取畸变矫正数据（输入输出pMbBlk不同）：

```

// imgin
ctx->stTask.stImgIn.stVFrame.pMbBlk = RK_MPI_MB_GetMB(ctx->inPool, ctx->u32SrcSize, RK_TRUE);
// imgout
ctx->stTask.stImgOut.stVFrame.pMbBlk = RK_MPI_MB_GetMB(ctx->outPool, ctx->u32DstSize, RK_TRUE);
// read image
pSrcData = RK_MPI_MB_Handle2VirAddr(ctx->stTask.stImgIn.stVFrame.pMbBlk);
s32Ret = read_image(reinterpret_cast<RK_U8*>(pSrcData), ctx->u32SrcWidth, ctx->u32SrcHeight, ctx->u32SrcVirWidth, ctx-
>u32SrcVirHeight, ctx->s32SrcPixelFormat, pFile);
s32Ret = RK_MPI_GDC_BeginJob(&hHandle);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_GDC_BeginJob err:0x%x!", s32Ret);
    goto __FAILED;
}
s32Ret = RK_MPI_GDC_AddCorrectionTask(hHandle, &ctx->stTask, &ctx->stFisheyeAttr);
if (s32Ret != RK_SUCCESS) {
    RK_MPI_GDC_CancelJob(hHandle);
    RK_LOGE("RK_MPI_GDC_AddCorrectionTask err:0x%x!", s32Ret);
    goto __FAILED;
}
s32Ret = RK_MPI_GDC_EndJob(hHandle);
if (s32Ret != RK_SUCCESS) {
    RK_MPI_GDC_CancelJob(hHandle);
    RK_LOGE("RK_MPI_GDC_EndJob err:0x%x!", s32Ret);
    goto __FAILED;
}
RK_MPI_MB_ReleaseMB(ctx->stTask.stImgIn.stVFrame.pMbBlk);
// get gcd output data
RK_MPI_SYS_MmzFlushCache(ctx->stTask.stImgOut.stVFrame.pMbBlk, RK_TRUE);
char *pData = (char *)RK_MPI_MB_Handle2VirAddr(ctx->stTask.stImgOut.stVFrame.pMbBlk);
fwrite(pData, 1, ctx->u32DstSize, pDstFile);
fflush(pDstFile);
RK_MPI_MB_ReleaseMB(ctx->stTask.stImgOut.stVFrame.pMbBlk);

```

4.7 RK_MPI_GDC_AddCorrectionExTask

【描述】

往一个已经开启的 job 里添加鱼眼矫正 task (XY 模式)。

【语法】

```
RK_S32 RK_MPI_GDC_AddCorrectionExTask(GDC_HANDLE hHandle, const FISHEYE_ATTR_EX_S *pstTask,  
FISHEYE_ATTR_S *pstFishEyeAttrrEx, RK_BOOL bCheckMode);
```

【参数】

参数名称	描述	输入\输出
hHandle	表示一个已启动 job 的 HANDLE	输入
pstTask	GDC task 属性指针	输入
pstFishEyeAttrrEx	FISHEYE 纠正区域属性配置指针(XY 模式)	输入
bCheckMode	检查模式 取值为 RK_TRUE 时，为纯黑边检查模式。暂不支持此功能 取值为 RK_FALSE 时，为矫正模式。目前仅支持此模式	

【返回值】

返回值	描述
0	成功
非0	失败，见 GDC错误码

【注意】

- 此接口功能暂未实现。

4.8 RK_MPI_GDC_FisheyePosQueryDst2Src

【描述】

根据鱼眼矫正输出图像坐标点查找源图像坐标点。

【语法】

```
RK_S32 RK_MPI_GDC_FisheyePosQueryDst2Src(const GDC_FISHEYE_POINT_QUERY_ATTR_S  
pstFisheyePointQueryAttr, const VIDEO_FRAME_INFO_S *pstVideoInfo, const POINT_S *pstDstPoint, POINT_S  
*pstSrcPoint);
```

【参数】

参数名称	描述	输入\输出
pstFisheyePointQueryAttr	鱼眼矫正坐标点反向查找属性参数	输入
pstVideoInfo	鱼眼输入图像帧信息 VIDEO_FRAME_INFO_S	输入
pstDstPoint	鱼眼矫正图上需要查找映射关系的坐标点	输入
pstSrcPoint	鱼眼原图上查找到的坐标点	输出

【返回值】

返回值	描述
0	成功
非0	失败，见 GDC错误码

【注意】

- 多区域时，每个输出区域的坐标点按照每个输出区域单独计算，不能按照拼接之后的图像来计算。
- 返回的原图坐标基于原图大小返回。

【举例】

```

RK_S32 s32Ret;
for (RK_U32 region = 1; region < ctx->stFisheyeAttr.u32RegionNum; region++) {
    GDC_FISHEYE_POINT_QUERY_ATTR_S stQuery;
    POINT_S stDstPoint, stSrcPoint;
    memset(&stQuery, 0, sizeof(stQuery));
    stQuery.pstFishEyeAttr = &ctx->stFisheyeAttr;
    stQuery.u32RegionIndex = region;
    stDstPoint.s32X = 0;
    stDstPoint.s32Y = 540;
    s32Ret = RK_MPI_GDC_FisheyePosQueryDst2Src(&stQuery, &ctx->stTask.stImgIn, &stDstPoint, &stSrcPoint);
    if (s32Ret != RK_SUCCESS) {
        RK_LOGE("RK_MPI_GDC_FisheyePosQueryDst2Src err:0x%x!", s32Ret);
    }
    RK_LOGD("region:%d dst:%d,%d src:%d,%d", region, stDstPoint.s32X, stDstPoint.s32Y,
            stSrcPoint.s32X, stSrcPoint.s32Y);
}

```

4.9 RK_MPI_GDC_FisheyePosQueryDst2SrcArray

【描述】

根据鱼眼矫正输出图像坐标点数组查找源图像坐标点数组。

【语法】

```
RK_S32 RK_MPI_GDC_FisheyePosQueryDst2SrcArray(const GDC_FISHEYE_POINT_QUERY_ATTR_S
pstFisheyePointQueryAttr, const VIDEO_FRAME_INFO_S *pstVideoInfo, const RK_U32 u32PointNum, const
POINT_S *pastDstPoint, POINT_S *pastSrcPoint);
```

【参数】

参数名称	描述	输入\输出
pstFisheyePointQueryAttr	鱼眼矫正坐标点反向查找属性参数	输入
pstVideoInfo	鱼眼输入图像帧信息 VIDEO_FRAME_INFO_S	输入
u32PointNum	查找坐标点个数	输入
pastDstPoint	鱼眼矫正图上需要查找映射关系的坐标点数组	输入
pastSrcPoint	鱼眼原图上查找到的坐标点数组	输出

【返回值】

返回值	描述
0	成功
非0	失败，见 GDC错误码

【注意】

- 多区域时，每个输出区域的坐标点按照每个输出区域单独计算，不能按照拼接之后的图像来计算。
- 返回的原图坐标基于原图大小返回。

【举例】

```

RK_S32 s32Ret;
GDC_FISHEYE_POINT_QUERY_ATTR_S stQuery;
memset(&stQuery, 0, sizeof(stQuery));
stQuery.pstFishEyeAttr = &ctx->stFisheyeAttr;
stQuery.u32RegionIndex = region;
s32Ret = RK_MPI_GDC_FisheyePosQueryDst2SrcArray(&stQuery, &ctx->stTask.stImgIn, ctx->u32PointLen, ctx->pastDstPoint, ctx-
>pastSrcPoint);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_GDC_FisheyePosQueryDst2SrcArray err:0x%x!", s32Ret);
}

```

4.10 RK_MPI_GDC_FisheyePosQueryDst2Pano

【描述】

根据鱼眼矫正输出图像坐标点查找全景图像坐标点。

【语法】

```

RK_S32 RK_MPI_GDC_FisheyePosQueryDst2Pano(const GDC_FISHEYE_POINT_QUERY_ATTR_S
pstFisheyePointQueryAttr, const VIDEO_FRAME_INFO_S *pstVideoInfo, const RK_U32 u32PanoRegionIndex, const
POINT_S *pstDstPoint, POINT_S *pstSrcPoint);

```

【参数】

参数名称	描述	输入\输出
pstFisheyePointQueryAttr	鱼眼矫正坐标点反向查找属性参数	输入
pstVideoInfo	鱼眼输入图像帧信息 VIDEO_FRAME_INFO_S	输入
u32PanoRegionIndex	全景图序号值，第一个区域为0开始计算	输入
pstDstPoint	鱼眼矫正图上需要查找映射关系的坐标点	输入
pstSrcPoint	全景图像上查找到的坐标点	输出

【返回值】

返回值	描述
0	成功
非0	失败，见 GDC错误码

【注意】

- 多区域时，每个输出区域的坐标点按照每个输出区域单独计算，不能按照拼接之后的图像来计算。
- 返回的全景图坐标基于全景图输出大小，与原图大小无关。

【举例】

```

RK_S32 s32Ret;
GDC_FISHEYE_POINT_QUERY_ATTR_S stQuery;
POINT_S stDstPoint, stPanoPoint;
memset(&stQuery, 0, sizeof(stQuery));
stQuery.pstFishEyeAttr = &ctx->stFisheyeAttr;
stQuery.u32RegionIndex = region;
stDstPoint.s32X = 0;
stDstPoint.s32Y = 540;
s32Ret = RK_MPI_GDC_FisheyePosQueryDst2Pano(&stQuery, &ctx->stTask.stImgIn, 0,
&stDstPoint, &stPanoPoint);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_GDC_FisheyePosQueryDst2Pano err:0x%x!", s32Ret);
}

```

4.11 RK_MPI_GDC_FisheyePosQueryDst2PanoArray

【描述】

根据鱼眼矫正输出图像坐标点数组查找全景图像坐标点数组。

【语法】

```
RK_S32 RK_MPI_GDC_FisheyePosQueryDst2PanoArray(const GDC\_FISHEYE\_POINT\_QUERY\_ATTR\_S
pstFisheyePointQueryAttr, const VIDEO\_FRAME\_INFO\_S pstVideoInfo, const RK_U32 u32PanoRegionIndex, const
RK_U32 u32PointNum, const POINT\_S*pastDstPoint, POINT\_S*pastSrcPoint);
```

【参数】

参数名称	描述	输入\输出
pstFisheyePointQueryAttr	鱼眼矫正坐标点反向查找属性参数	输入
pstVideoInfo	鱼眼输入图像帧信息 VIDEO_FRAME_INFO_S	输入
u32PanoRegionIndex	全景图序号值，第一个区域为0开始计算	输入
u32PointNum	查找坐标点个数	输入
pastDstPoint	鱼眼矫正图上需要查找映射关系的坐标点数组	输入
pastSrcPoint	全景图像上查找到的坐标点数组	输出

【返回值】

返回值	描述
0	成功
非0	失败，见 GDC错误码

【注意】

- 多区域时，每个输出区域的坐标点按照每个输出区域单独计算，不能按照拼接之后的图像来计算。
- 返回的全景图坐标基于全景图输出大小，与原图大小无关。

【举例】

```
RK_S32 s32Ret;
GDC\_FISHEYE\_POINT\_QUERY\_ATTR\_S stQuery;
memset(&stQuery, 0, sizeof(GDC\_FISHEYE\_POINT\_QUERY\_ATTR\_S));
stQuery.pstFishEyeAttr = &ctx->stFisheyeAttr;
stQuery.u32RegionIndex = region;
s32Ret = RK_MPI_GDC_FisheyePosQueryDst2PanoArray(&stQuery, &ctx->stTask.stImgIn,
pano_region, ctx->u32PointLen,
ctx->pastDstPoint, ctx->pastSrcPoint);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_GDC_FisheyePosQueryDst2PanoArray err:0x%x!", s32Ret);
}
```

5. 数据类型

GDC 模块相关数据类型定义如下：

- [GDC_HANDLE](#): 定义 GDC job 的句柄。
- [GDC_TASK_ATTR_S](#): 定义 GDC task 的属性。
- [FISHEYE_LMFCOEF_NUM](#): 鱼眼镜头 LMF 参数个数。
- [FISHEYE_CONFIG_S](#): 定义 FISHEYE 的鱼眼镜头 LMF 参数配置。
- [FISHEYE_MOUNT_MODE_E](#): 定义 FISHEYE 属性中的安装模式。

- [FISHEYE_VIEW_MODE_S](#): 定义 FISHEYE 属性中的矫正模式。
- [RECT_S](#): 定义矩形区域的相关属性。
- [FISHEYE_REGION_ATTR_S](#): 定义 FISHEYE 每个矫正区域的属性配置。
- [FISHEYE_REGION_ATTR_EX_S](#): 定义 FISHEYE 每个矫正区域的属性配置(XY 模式)。
- [FISHEYE_MAX_REGION_NUM](#): 定义 FISHEYE 支持的最大矫正区域个数。
- [FISHEYE_ATTR_S](#): 定义 FISHEYE 属性的相关配置。
- [FISHEYE_ATTR_EX_S](#): 定义 FISHEYE 属性的相关配置(XY模式)。
- [GDC_FISHEYE_POINT_QUERY_ATTR_S](#): 鱼眼矫正坐标点反向查找属性参数。

5.1 GDC_HANDLE

【说明】

定义 GDC job 的句柄。

【定义】

```
typedef RK_S32 GDC_HANDLE
```

【注意事项】

无。

【相关数据类型及接口】

无。

5.2 GDC_TASK_ATTR_S

【说明】

定义 GDC task 的属性。

【定义】

```
typedef struct rkGDC_TASK_ATTR_S {
    VIDEO_FRAME_INFO_S stImgIn;
    VIDEO_FRAME_INFO_S stImgOut;
    RK_U64 au64privateData[4];
    RK_U64 u64TaskId;
} GDC_TASK_ATTR_S;
```

【成员】

参数名称	描述
stImgIn	输入图像属性
stImgOut	输出图像属性
au64privateData	调试功能参数 与 task 相关的私有数据，au64privateData[0]、au64privateData[1]分别为矫正图像的步进 stepX/stepY（取值范围为2的幂次方），可用于调试具体效果使用，内部会自适应配置相关参数。如果此值配置过低会导致系统负载加大，配置过大导致效果不理想。建议设置为0即可
u64TaskId	拓展功能预留参数 taskId设置，[0,GDC_MAX_TASK_NUM-1];非0值表示指定taskId，如果此taskId已经在使用过程中会返回错误RK_ERR_GDC_BUSY;0表示由系统默认使用非忙状态的task。建议设置为0即可

【注意】

- au64privateData及u64TaskId参数为调试或拓展使用，无特殊使用，建议用户均设置为0。

5.3 FISHEYE_LMFCOEF_NUM

【说明】

鱼眼镜头 LMF 参数个数。

【定义】

```
#define FISHEYE_LMFCOEF_NUM 128
```

【注意事项】

无

5.4 FISHEYE_CONFIG_S

【说明】

定义 FISHEYE 的鱼眼镜头 LMF 参数配置。

【定义】

```
typedef struct rkFISHEYE_CONFIG_S {  
    RK_U16          au16LMFCoeff[FISHEYE_LMFCOEF_NUM];  
} FISHEYE_CONFIG_S;
```

【成员】

参数名称	描述
au16LMFCoeff	鱼眼镜头 LMF 参数

【注意事项】

无

5.5 FISHEYE_JOB_CONFIG_S

【说明】

定义 FISHEYE 任务对应的鱼眼镜头 LMF 参数配置。

【定义】

```
typedef struct rkFISHEYE_JOB_CONFIG_S {  
    RK_U64          u64LenMapPhyAddr;  
} FISHEYE_JOB_CONFIG_S;
```

【成员】

参数名称	描述
u64LenMapPhyAddr	保存鱼眼 LMF 系数的物理地址

【注意事项】

无

5.6 FISHEYE_MOUNT_MODE_E

【说明】

定义 FISHEYE 属性中的安装模式。

【定义】

```
typedef enum rkFISHEYE_MOUNT_MODE_E {
    FISHEYE_DESKTOP_MOUNT = 0,
    FISHEYE_CEILING_MOUNT = 1,
    FISHEYE_WALL_MOUNT = 2,
    FISHEYE_MOUNT_MODE_BUTT
} FISHEYE_MOUNT_MODE_E;
```

【成员】

参数名称	描述
FISHEYE_DESKTOP_MOUNT	地装模式
FISHEYE_CEILING_MOUNT	顶装模式
FISHEYE_WALL_MOUNT	壁装模式
FISHEYE_MOUNT_MODE_BUTT	最大值

5.7 FISHEYE_VIEW_MODE_E

【说明】

定义 FISHEYE 属性中的矫正模式。

【定义】

```
typedef enum rkFISHEYE_VIEW_MODE_E {
    FISHEYE_VIEW_360_PANORAMA = 0,
    FISHEYE_VIEW_180_PANORAMA = 1,
    FISHEYE_VIEW_NORMAL = 2,
    FISHEYE_NO_TRANSFORMATION = 3,

    FISHEYE_VIEW_MODE_BUTT
} FISHEYE_VIEW_MODE_E;
```

【成员】

参数名称	描述
FISHEYE_VIEW_360_PANORAMA	360全景模式
FISHEYE_VIEW_180_PANORAMA	180全景模式
FISHEYE_VIEW_NORMAL	Normal矫正模式
FISHEYE_NO_TRANSFORMATION	不做矫正模式，对原始图缩放输出（根据对应区域设置的输出区域决定）
FISHEYE_VIEW_MODE_BUTT	最大值

【注意】

- 360 矫正模式下不支持壁装，180 矫正模式下不支持顶装和地装。

5.8 RECT_S

【说明】

定义矩形区域的相关属性。

【定义】

```
typedef struct rkRECT_S {  
    RK_S32 s32X;  
    RK_S32 s32Y;  
    RK_U32 u32Width;  
    RK_U32 u32Height;  
} RECT_S;
```

【成员】

参数名称	描述
s32X	该区域起始坐标x
s32Y	该区域起始坐标y
u32Width	该区域宽度
u32Height	该区域高度

5.9 FISHEYE_REGION_ATTR_S

【说明】

定义 FISHEYE 每个矫正区域的属性配置。

【定义】

```
typedef struct rkFISHEYE_REGION_ATTR_S {  
    FISHEYE_VIEW_MODE_E enViewMode;  
    RK_U32             u32InRadius;  
    RK_U32             u32OutRadius;  
    RK_U32             u32Pan;  
    RK_U32             u32Tilt;  
    RK_U32             u32HorZoom;  
    RK_U32             u32VerZoom;  
    RECT_S             stOutRect;  
} FISHEYE_REGION_ATTR_S;
```

【成员】

参数名称	描述
enViewMode	该矫正区域的矫正模式
u32InRadius	360 全景模式表示该矫正区域所对应原图的内半径，其他模式无效 取值范围：[0, u32OutRadius)
u32OutRadius	360 全景模式表示该矫正区域所对应原图的外半径，其他模式为矫正区域的可视半径 取值范围：[1, 3 x max(width of input picture/4, height of input picture/4)]
u32Pan	该矫正区域 PTZ 参数的 Pan 值 取值范围：[0, 360]
u32Tilt	该矫正区域 PTZ 参数的 Tilt 值 取值范围：[0, 360]
u32HorZoom	该矫正区域 PTZ 参数的水平 Zoom 值 取值范围：[1, 4095]
u32VerZoom	该矫正区域 PTZ 参数的垂直 Zoom 值 取值范围：[1, 4095]
stOutRect	该矫正区域的输出位置及宽高

【注意】

- 鱼眼支持对一幅图像的多个区域分别进行矫正，每个区域的属性配置是各自独立的。

5.10 FISHEYE_REGION_ATTR_EX_S

【说明】

定义 FISHEYE 每个矫正区域的属性配置(XY 模式)。

【定义】

```
typedef struct rkFISHEYE_REGION_ATTR_S {
    FISHEYE_VIEW_MODE_E enViewMode;
    RK_U32             u32InRadius;
    RK_U32             u32OutRadius;
    RK_U32             u32X;
    RK_U32             u32Y;
    RK_U32             u32HorZoom;
    RK_U32             u32VerZoom;
    RECT_S             stOutRect;
} FISHEYE_REGION_ATTR_S;
```

【成员】

参数名称	描述
enViewMode	该矫正区域的矫正模式
u32InRadius	360 全景模式表示该矫正区域所对应原图的内半径，其他模式无效 取值范围：[0, u32OutRadius)
u32OutRadius	360 全景模式表示该矫正区域所对应原图的外半径，其他模式为矫正区域的可视半径 取值范围：[1, 3 x max(width of input picture/4, height of input picture/4)]
u32X	选择区域的中心点 X 坐标 取值范围：[0, width of input picture-1]
u32Y	选择区域的中心点 Y 坐标 取值范围：[0, height of input picture-1]
u32HorZoom	该矫正区域 PTZ 参数的水平 Zoom 值 取值范围：[1, 4095]
u32VerZoom	该矫正区域 PTZ 参数的垂直 Zoom 值 取值范围：[1, 4095]
stOutRect	该矫正区域的输出位置及宽高

【注意】

- 鱼眼支持对一幅图像的多个区域分别进行矫正，每个区域的属性配置是各自独立的。

5.11 FISHEYE_MAX_REGION_NUM

【说明】

定义FISHEYE 支持的最大矫正区域个数。

【定义】

```
#define FISHEYE_MAX_REGION_NUM 9
```

5.12 FISHEYE_ATTR_S

【说明】

定义 FISHEYE 属性的相关配置。

【定义】

```
typedef struct rkFISHEYE_ATTR_S {
    RK_BOOL      bEnable;
    RK_BOOL      bLMF;
    RK_BOOL      bBgColor;
    RK_U32       u32BgColor;
    RK_S32       s32HorOffset;
    RK_S32       s32VerOffset;
    RK_U32       u32TrapezoidCoef;
    RK_S32       s32FanStrength;
    FISHEYE_MOUNT_MODE_E enMountMode;
    RK_U32       u32RegionNum;
    FISHEYE_REGION_ATTR_S astFishEyeRegionAttr[FISHEYE_MAX_REGION_NUM];
} FISHEYE_ATTR_S;
```

【成员】

参数名称	描述
bEnable	关闭或打开鱼眼矫正功能
bLMF	是否使用用户设置的鱼眼镜头 LMF 参数 该参数暂未使用
bBgColor	是否在输出图像打上背景色 该参数暂未使用
u32BgColor	背景色的颜色 RGB888 格式，取值范围[0, 0xFFFFFFF] 该参数暂未使用
s32HorOffset	镜头中心点相对于 SENSOR 中心点的水平偏移，取值范围[-511, 511]，单位为像素
s32VerOffset	镜头中心点相对于 SENSOR 中心点的垂直偏移，取值范围[- 511, 511]，单位为像素
u32TrapezoidCoef	梯形矫正强度系数。用于壁装时的俯仰角矫正，取值范围[0, 32]
s32FanStrength	扇矫正强度系数，仅在 180 模式时有效，取值范围[-760, 760]
enMountMode	鱼眼矫正安装模式
u32RegionNum	一幅图像的矫正区域数目，最多支持 9 个区域
astFishEyeRegionAttr	每个矫正区域各自的属性配置

【注意】

- FISHEYE 支持对一幅图像的多个区域分别进行矫正处理，每个矫正区域的矫正模式，PTZ 参数等互相独立。同时根据每个矫正区域设置的输出位置及宽高，对图像进行拼接后输出。

5.13 FISHEYE_ATTR_EX_S

【说明】

定义 FISHEYE 属性的相关配置(XY模式)。

【定义】

```
typedef struct rkFISHEYE_ATTR_EX_S {
    RK_BOOL      bEnable;
    RK_BOOL      bLMF;
    RK_BOOL      bBgColor;
    RK_U32       u32BgColor;
    RK_S32       s32HorOffset;
    RK_S32       s32VerOffset;
    RK_U32       u32TrapezoidCoef;
    RK_S32       s32FanStrength;
    FISHEYE_MOUNT_MODE_E   enMountMode;
    RK_U32       u32RegionNum;
    FISHEYE_REGION_ATTR_EX_S astFishEyeRegionAttr[FISHEYE_MAX_REGION_NUM];
} FISHEYE_ATTR_EX_S;
```

【成员】

参数名称	描述
bEnable	关闭或打开鱼眼矫正功能
bLMF	是否使用用户设置的鱼眼镜头 LMF 参数 该参数暂未使用
bBgColor	是否在输出图像打上背景色 该参数暂未使用
u32BgColor	背景色的颜色 RGB888 格式，取值范围[0, 0xFFFFFFF] 该参数暂未使用
s32HorOffset	镜头中心点相对于 SENSOR 中心点的水平偏移，取值范围[-511, 511]，单位为像素
s32VerOffset	镜头中心点相对于 SENSOR 中心点的垂直偏移，取值范围[- 511, 511]，单位为像素
u32TrapezoidCoef	梯形矫正强度系数。用于壁装时的俯仰角矫正，取值范围[0, 32]
s32FanStrength	扇矫正强度系数，仅在 180 模式时有效，取值范围[-760, 760]
enMountMode	鱼眼矫正安装模式
u32RegionNum	一幅图像的矫正区域数目，最多支持 9 个区域
astFishEyeRegionAttr	每个矫正区域各自的属性配置（XY 模式属性）

【注意】

- FISHEYE 支持对一幅图像的多个区域分别进行矫正处理，每个矫正区域的矫正模式，PTZ 参数等互相独立。同时根据每个矫正区域设置的输出位置及宽高，对图像进行拼接后输出。

5.14 GDC_FISHEYE_POINT_QUERY_ATTR_S

【说明】

鱼眼矫正坐标点反向查找属性参数。

【定义】

```
typedef struct rkGDC_FISHEYE_POINT_QUERY_ATTR_S {
    RK_U32    u32RegionIndex;
    FISHEYE_ATTR_S *pstFishEyeAttr;
    RK_U16    au16LMF[FISHEYE_LMFCOEF_NUM];
} GDC_FISHEYE_POINT_QUERY_ATTR_S;
```

【成员】

参数名称	描述
u32RegionIndex	鱼眼矫正区域号 取值范围：[0, FISHEYE_MAX_REGION_NUM-1]
pstFishEyeAttr	鱼眼属性参数
au16LMF	LMF 系数

6. GDC错误码

GDC API GDC错误码如下

错误代码	宏定义	描述
0xA0128002	RK_ERR_GDC_INVALID_CHNID	GDC的taskId无效
0xA0128003	RK_ERR_GDC_ILLEGAL_PARAM	GDC 参数设置无效
0xA0128005	RK_ERR_GDC_CHN_UNEXIST	GDC的task不存在
0xA0128006	RK_ERR_GDC_NULL_PTR	输入参数空指针错误
0xA0128008	RK_ERR_GDC_NOT_SUPPORT	操作不支持
0xA0128009	RK_ERR_GDC_NOT_PERMITTED	操作不允许
0xA012800D	RK_ERR_GDC_NOBUF	分配内存失败
0xA012800E	RK_ERR_GDC_BUF_EMPTY	GDC的job,task已经使用完毕
0xA012800F	RK_ERR_GDC_BUF_FULL	没有剩余 BUF
0xA0128010	RK_ERR_GDC_SYS_NOTREADY	系统未初始化
0xA0128012	RK_ERR_GDC_BUSY	GDC的job,task忙状态

全景拼接

[功能描述](#)

[API 参考](#)

[数据类型](#)

[AVS错误码](#)

[相关文档](#)

1. 功能描述

1.1 基本概念

1.1.1 GROUP

组（GROUP），以下均称为组，AVS 对用户提供组（GROUP）的概念。最大可用数为 [AVS_MAX_GRP_NUM](#) 个，各 GROUP 分时复用硬件设备。每个 AVS GROUP 包含多个 PIPE 和多个 CHANNEL。

1.1.2 PIPE

管道（PIPE），以下均称为管道，AVS 组的 PIPE。用于输入拼接源图像。PIPE 的数目即拼接路数。用户可以通过系统绑定和前端相连或者发送图像到 PIPE 中。

1.1.3 CHANNEL

AVS 组的通道。用于输出拼接的结果图像。

1.1.4 投影

将全景空间图像投影到平面坐标中用于编码和传输。AVS 支持的投影模式有 Equirectangular、Cylindrical、Rectilinear、CubeMap 和 Trans_Equirectangular 五种投影模式。

1.1.5 压缩和解压

1.1.5.1 压缩

支持通道输出图像进行压缩，支持AFBC压缩。

1.1.5.2 解压

支持对输入的压缩图像进行解压缩处理，支持AFBC解压。

1.1.5.3 各芯片平台压缩和解压支持列表

芯片名称	是否支持AFBC 压缩	是否支持AFBC 解压
RK3576	支持	支持
RK3588	支持	支持
RV1106	不支持	不支持
RV1109/RV1126	不支持	不支持

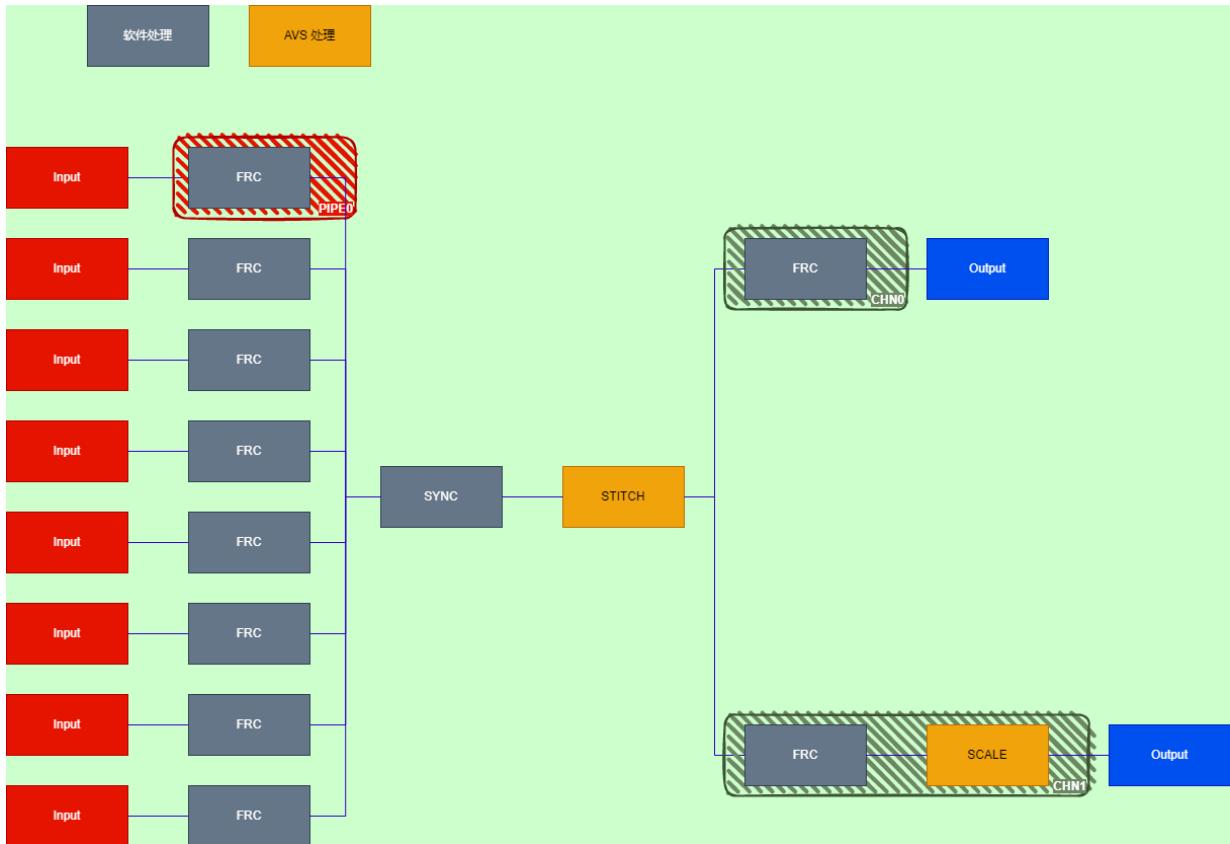
1.2 应用方式

通过调用 SYS 模块的绑定接口，可与 VI/VPSS 和 VO/VENC/VPSS 等模块进行绑定，其中前者为 AVS 的输入源，后者为 AVS 的接收者。用户可通过 AVS MPI 接口对 Group 进行管理，每个 Group 可与多个输入源绑定（通过绑定到 AVS 的 PIPE 上实现），可与多个接收者绑定（通过与 AVS 的 CHANNEL 绑定实现，AVS 的每个 CHANNEL 上只能与一个接收者绑定）。

1.2.1 AVS 数据流图

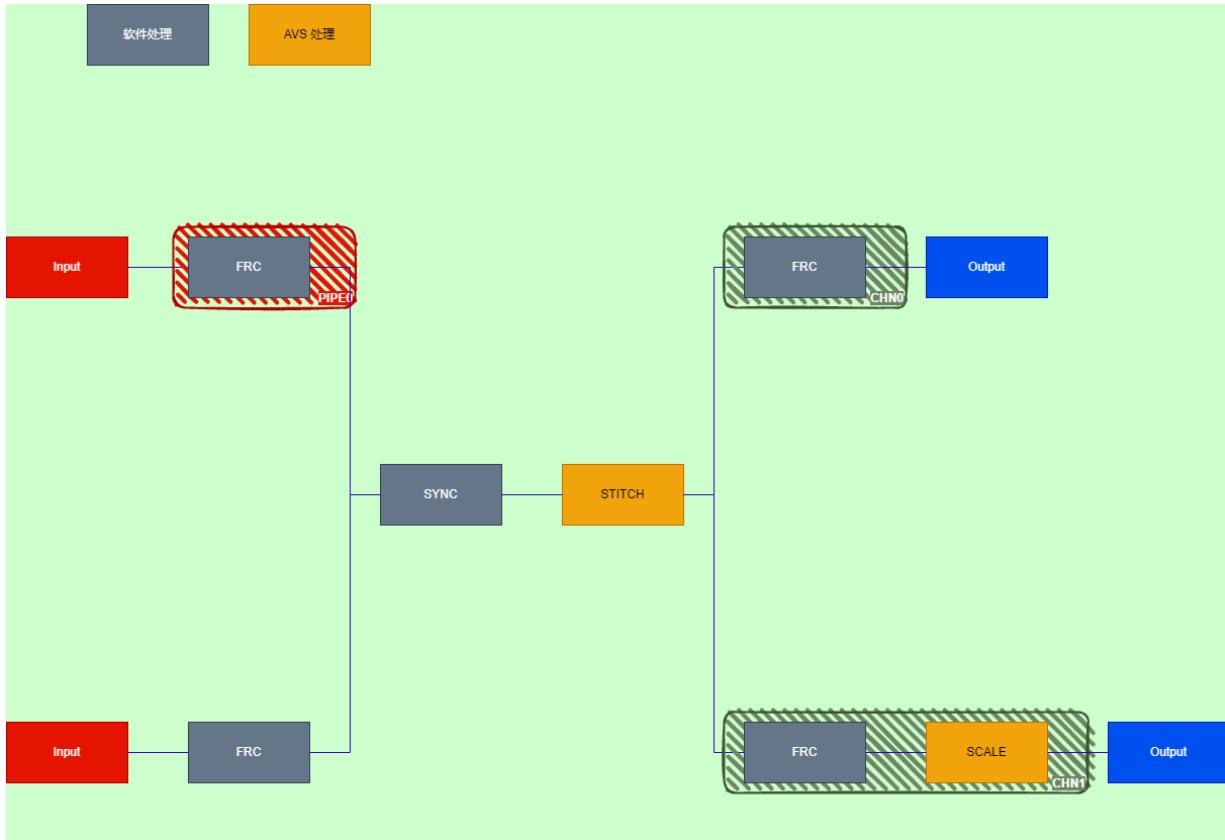
AVS 数据流图如下图所示，输入源的数据经各 PIPE 发送到 GROUP 处，GROUP 进行拼接处理并将处理完成的数据转发到 CHANNEL，等待用户获取或者送入接受者。

图 RK3588 平台的数据流图



RK3588 平台 AVS 处理由硬件 GPU 实现。

图 RV1106/RV1109/RV1126 平台的数据流图



RV1106/RV1109/RV1126 平台AVS 处理由CPU 和 硬件RGA 实现。

1.3 输入输出特性

1.3.1 RK3588/RK3576 平台输入输出图像支持列表

数据格式	宽度对齐	高度对齐	像素字节宽度 (bit)
RK_FMT_YUV420SP	16	4	8

1.3.2 RV1106 平台输入输出图像支持列表

数据格式	宽度对齐	高度对齐	像素字节宽度 (bit)
RK_FMT_YUV420SP	4	2	8

1.3.2.1 RV1126/RV1109 平台输入输出图像支持列表

数据格式	宽度对齐	高度对齐	像素字节宽度 (bit)
RK_FMT_YUV420SP	4	2	8

1.4 举例

详细测试DEMO，请参考发布文件: test_mpi_avs.cpp、test_comm_avs.cpp。

```
RK_S32 s32Ret = RK_FAILURE;
AVS_CFG_S *pstAvsCtx = reinterpret_cast<AVS_CFG_S *>(&(ctx->avContext));
```

```

pstAvsCtx->s32GrpId = 0;
pstAvsCtx->s32PipeId = 0;
pstAvsCtx->s32ChnId = 0;
pstAvsCtx->u32AvsPipeCnt = ctx->s32StitchNum;
pstAvsCtx->u32AvsChnCnt = 1;

pstAvsCtx->stAvsGrpAttr.enMode = ctx->enAvsWorkMode;
pstAvsCtx->u32OutW = 8192;
pstAvsCtx->u32OutH = 2700;

snprintf(pstAvsCtx->stAvsGrpAttr.stOutAttr.stCalib.pCalibFilePath, sizeof("/usr/share/avs_calib/calib_file_pos.pto"),
         "/usr/share/avs_calib/calib_file_pos.pto");
snprintf(pstAvsCtx->stAvsGrpAttr.stOutAttr.stCalib.pMeshAlphaPath, sizeof("/usr/share/avs_calib/"),
         "/usr/share/avs_calib/");

pstAvsCtx->stAvsGrpAttr.stLUT.enAccuracy      = AVS_LUT_ACCURACY_HIGH;

pstAvsCtx->stAvsGrpAttr.u32PipeNum      = pstAvsCtx->u32AvsPipeCnt;
pstAvsCtx->stAvsGrpAttr.stGainAttr.enMode    = AVS_GAIN_MODE_AUTO;

pstAvsCtx->stAvsGrpAttr.stOutAttr.enPrjMode    = AVS_PROJECTION_EQUIRECTANGULAR;
pstAvsCtx->stAvsGrpAttr.stOutAttr.stCenter.s32X  = 4196;
pstAvsCtx->stAvsGrpAttr.stOutAttr.stCenter.s32Y  = 2080;
pstAvsCtx->stAvsGrpAttr.stOutAttr.stFOV.u32FOVX = 28000;
pstAvsCtx->stAvsGrpAttr.stOutAttr.stFOV.u32FOVY = 9500;
pstAvsCtx->stAvsGrpAttr.stOutAttr.stORIRotation.s32Roll = 0;
pstAvsCtx->stAvsGrpAttr.stOutAttr.stORIRotation.s32Pitch = 0;
pstAvsCtx->stAvsGrpAttr.stOutAttr.stORIRotation.s32Yaw = 0;
pstAvsCtx->stAvsGrpAttr.stOutAttr.stRotation.s32Roll = 0;
pstAvsCtx->stAvsGrpAttr.stOutAttr.stRotation.s32Pitch = 0;
pstAvsCtx->stAvsGrpAttr.stOutAttr.stRotation.s32Yaw = 0;

pstAvsCtx->stAvsGrpAttr.bSyncPipe      = ctx->bFrameSync;
pstAvsCtx->stAvsGrpAttr.stFrameRate.s32SrcFrameRate = -1;
pstAvsCtx->stAvsGrpAttr.stFrameRate.s32DstFrameRate = -1;

pstAvsCtx->stAvsChnAttr[0].enCompressMode     = ctx->enLinkCompressMode;
pstAvsCtx->stAvsChnAttr[0].stFrameRate.s32SrcFrameRate = -1;
pstAvsCtx->stAvsChnAttr[0].stFrameRate.s32DstFrameRate = -1;
pstAvsCtx->stAvsChnAttr[0].u32Depth        = 0;
pstAvsCtx->stAvsChnAttr[0].u32Width        = pstAvsCtx->u32OutW;
pstAvsCtx->stAvsChnAttr[0].u32Height       = pstAvsCtx->u32OutH;
pstAvsCtx->stAvsChnAttr[0].enDynamicRange   = DYNAMIC_RANGE_SDR8;

s32Ret = RK_MPI_AVIS_SetModParam(&ctx->stAvsModParam);
if (RK_SUCCESS != s32Ret) {
    RK_LOGE("AVS set mod params failed with %#x!", s32Ret);
    goto __FAILED;
}

s32Ret = RK_MPI_AVIS_CreateGrp(ctx->s32GrpId, &ctx->stAvsGrpAttr);
if (RK_SUCCESS != s32Ret) {
    RK_LOGE("AVS creat grp failed with %#x!", s32Ret);
    goto __FAILED;
}

s32Ret = RK_MPI_AVIS_SetChnAttr(ctx->s32GrpId, ctx->s32ChnId, &ctx->stAvsChnAttr[0]);
if (RK_SUCCESS != s32Ret) {
    RK_LOGE("AVS set chn attr failed with %#x!", s32Ret);
    goto __FAILED;
}

s32Ret = RK_MPI_AVIS_EnableChn(ctx->s32GrpId, ctx->s32ChnId);
if (RK_SUCCESS != s32Ret) {
    RK_LOGE("AVS enable chn failed with %#x!", s32Ret);
    goto __FAILED;
}

```

```

s32Ret = RK_MPI_AVIS_StartGrp(ctx->s32GrpId);
if (RK_SUCCESS != s32Ret) {
    RK_LOGE("AVS start grp failed with %#x!", s32Ret);
    goto __FAILED;
}

__FAILED:

return s32Ret;

```

2. API 参考

该功能模块为用户提供以下API:

- [RK_MPI_AVIS_CreateGrp](#) : 创建一个 AVS 组。
- [RK_MPI_AVIS_DestroyGrp](#) : 销毁一个 AVS 组。
- [RK_MPI_AVIS_StartGrp](#) : 启动 AVS 组。
- [RK_MPI_AVIS_StopGrp](#) : 禁用 AVS 组。
- [RK_MPI_AVIS_ResetGrp](#) : 重置 AVS 组。
- [RK_MPI_AVIS_GetGrpAttr](#) : 获取 AVS 组属性。
- [RK_MPI_AVIS_SetGrpAttr](#) : 设置 AVS 组属性。
- [RK_MPI_AVIS_GetGrpRoi](#) : 获取 AVS 组感兴趣区域。
- [RK_MPI_AVIS_SetGrpRoi](#) : 设置 AVS 组感兴趣区域。
- [RK_MPI_AVIS_GetGrpRotation](#) : 获取 AVS 组旋转。
- [RK_MPI_AVIS_SetGrpRotation](#) : 设置 AVS 组旋转。
- [RK_MPI_AVIS_GetPipeAttr](#) : 获取 AVS 管道属性。
- [RK_MPI_AVIS_SetPipeAttr](#) : 设置 AVS 管道属性。
- [RK_MPI_AVIS_SendPipeFrame](#) : 用户向 AVS 组上的 PIPE 发送图像。
- [RK_MPI_AVIS_SetChnAttr](#) : 设置 AVS 通道属性。
- [RK_MPI_AVIS_GetChnAttr](#) : 获取 AVS 通道属性。
- [RK_MPI_AVIS_EnableChn](#) : 启用 AVS 通道。
- [RK_MPI_AVIS_DisableChn](#) : 禁用 AVS 通道。
- [RK_MPI_AVIS_GetChnFrame](#) : 用户获取一帧通道图像。
- [RK_MPI_AVIS_ReleaseChnFrame](#) : 用户释放一帧通道图像。
- [RK_MPI_AVIS_GetFinalLut](#) : 获取最终查找表。

2.1 RK_MPI_AVIS_CreateGrp

【描述】

创建一个 AVS 组。

【语法】

```
RK_S32 RK_MPI_AVIS_CreateGrp(AVS\_GRP AVSGrp, const AVS\_GRP\_ATTR\_S* pstGrpAttr);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围: [0, AVS_MAX_GRP_NUM)。	输入
pstGrpAttr	AVS 组属性	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AVS错误码 。

【注意】

- 组不支持重复创建。

【需求】

- 头文件：rk_comm_avs.h、rk_mpi_avs.h。
- 库文件：
 - RK3588 平台：libpanoStitchApp.so、librkgfx_avs.so。
 - RV1106/RV1109/RV1126 平台：librkAVS_genStitch.so、librkAVS_genLut.so。

2.2 RK_MPI_AVN_DestroyGrp

【描述】

销毁一个 AVS 组。

【语法】

```
RK_S32 RK_MPI_AVN_DestroyGrp(AVS\_GRP AVSGrp);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围：[0, AVS_MAX_GRP_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AVS错误码 。

【注意】

- 组必须已经创建。
- 调用此接口之前，必须先调用 RK_MPI_AVN_StopGrp 禁用此组。
- 调用此接口时，会一直等待此组当前任务处理结束才会真正销毁。

2.3 RK_MPI_AVN_StartGrp

【描述】

启动 AVS 组。

【语法】

```
RK_S32 RK_MPI_AVN_StartGrp(AVS\_GRP AVSGrp);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围：[0, AVS_MAX_GRP_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为AVS错误码。

【注意】

- 组必须已经创建。
- 重复调用该函数启用同一个组返回成功。

2.4 RK_MPI_AVIS_StopGrp

【描述】

禁用 AVS 组。

【语法】

```
RK_S32 RK_MPI_AVIS_StopGrp(AVS_GRP AVSGrp);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围：[0, AVS_MAX_GRP_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为AVS错误码。

【注意】

2.5 RK_MPI_AVIS_ResetGrp

【描述】

重置 AVS GROUP。

【语法】

```
RK_S32 RK_MPI_AVIS_ResetGrp(AVS_GRP AVSGrp);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围：[0, AVS_MAX_GRP_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为AVS错误码。

【注意】

- 组必须已经创建。

2.6 RK_MPI_AVG_GetGrpAttr

【描述】

获取 AVS 组属性。

【语法】

```
RK_S32 RK_MPI_AVG_GetGrpAttr(AVS\_GRP AVSGrp, const AVS\_GRP\_ATTR\_S* pstGrpAttr);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围: [0, AVS_MAX_GRP_NUM)。	输入
pstGrpAttr	AVS 组属性	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AVS错误码 。

【注意】

- 组必须已经创建。

2.7 RK_MPI_AVG_SetGrpAttr

【描述】

设置 AVS 组属性。

【语法】

```
RK_S32 RK_MPI_AVG_SetGrpAttr(AVS\_GRP AVSGrp, const AVS\_GRP\_ATTR\_S* pstGrpAttr);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围: [0, AVS_MAX_GRP_NUM)。	输入
pstGrpAttr	AVS 组属性	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AVS错误码 。

【注意】

- 组必须已经创建。
- 组属性必须合法, 其中部分静态属性不可动态设置, 具体详见 [AVS_GRP_ATTR_S](#)。

2.8 RK_MPI_AVG_GetGrpRotation

【描述】

获取 AVS 组旋转角度。

【语法】

```
RK_S32 RK_MPI_AVG_GetGrpRotation(AVS\_GRP AVSGrp, ROTATION\_E* penRotation);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围: [0, AVS_MAX_GRP_NUM)。	输入
penRotation	AVS 组旋转角度。	输出

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AVS错误码 。

【注意】

- 预留接口，暂未实现。
- GROUP 必须已创建。

2.9 RK_MPI_AVG_SetGrpRotation

【描述】

设置 AVS 组旋转角度。

【语法】

```
RK_S32 RK_MPI_AVG_SetGrpRotation(AVS\_GRP AVSGrp, ROTATION\_E enRotation);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围: [0, AVS_MAX_GRP_NUM)。	输入
enRotation	AVS 组旋转角度	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AVS错误码 。

【注意】

- 预留接口，暂未实现。
- 组必须已经创建。
- 仅在AVS_MODE_NOBLEND_VER、AVS_MODE_NOBLEND_HOR、AVS_MODE_NOBLEND_QR 模式下，调用该接口参数才会生效。

- 调用该接口实现Group 组中所有输入到PIPE 的图像旋转，仅支持 0 度、90 度、180 度、270 度的旋转，不支持任意角度旋转。

2.10 RK_MPI_AVG_GetGrpRoi

【描述】

获取 AVS 组感兴趣区域。

【语法】

```
RK_S32 RK_MPI_AVG_GetGrpRoi(AVS\_GRP AVSGrp, AVS\_ROI\_PARAM\_S* pstRoiParam);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围：[0, AVS_MAX_GRP_NUM)。	输入
pstRoiParam	AVS 组感兴趣区域。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AVS错误码 。

【注意】

- 组必须已经创建。
- 仅RK3588 平台支持该接口。
- 仅在AVS_MODE_BLEND 模式下，调用该接口参数才会生效。

【相关数据类型及接口】

- AVS 工作模式 [AVS_MODE_E](#)

2.11 RK_MPI_AVG_SetGrpRoi

【描述】

设置 AVS 组感兴趣区域。

【语法】

```
RK_S32 RK_MPI_AVG_SetGrpRoi(AVS\_GRP AVSGrp, const AVS\_GRP\_ATTR\_S* pstRoiParam);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围：[0, AVS_MAX_GRP_NUM)。	输入
pstRoiParam	AVS 组感兴趣区域	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AVS错误码 。

【注意】

- 组必须已经创建。
- 仅RK3588 平台支持该接口。
- 仅在 AVS_MODE_BLEND 模式下，调用该接口参数才会生效。
- pstRoiParam 除需要满足异常参数检查外，还需要满足如下条件：
 - 宽度：大于等于MESH 分辨率宽度的 1/2。
 - 高度：大于等于MESH 分辨率高度的 1/2。
- 该接口会在设置区域不可用，不能生成输出图像时，返回错误。

【相关数据类型及接口】

- [AVS_ROI_PARAM_S](#)
- [RK_MPI_AVG_GetGrpRoi](#)

2.12 RK_MPI_AVG_SetPipeAttr

【描述】

设置 AVS 通道属性。

【语法】

```
RK_S32 RK_MPI_AVG_SetPipeAttr(AVS\_GRP AVSGrp, const AVS\_PIPE AVSPipe,
AVS\_PIPE\_ATTR\_S* pstPipeAttr);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围：[0, AVS_MAX_GRP_NUM)。	输入
AVSPipe	AVS 管道号。 AVS_MODE_NOBLEND_OVL 模式下，取值范围[0, AVS_PIPE_NUM)。	输入
pstPipeAttr	AVS 管道属性	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AVS错误码 。

【注意】

- 组必须已经创建。
- 仅RV1106 平台支持该接口。
- 仅在AVS_MODE_NOBLEND_OVL 模式下，调用该接口参数才会生效。
- 管道属性必须合法，且不支持动态配置。

2.13 RK_MPI_AVG_GetPipeAttr

【描述】

获取 AVS 通道属性。

【语法】

```
RK_S32 RK_MPI_AVG_GetPipeAttr(AVS\_GRP AVSGrp, AVS\_PIPE AVSPipe,  
                                AVS\_PIPE\_ATTR\_S* pstPipeAttr);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围: [0, AVS_MAX_GRP_NUM)。	输入
AVSPipe	AVS 管道号。 取值范围[0, AVS_PIPE_NUM)。	输入
pstPipeAttr	AVS 管道属性	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AVS错误码 。

【注意】

- 组必须已经创建。
- 仅RV1106 平台支持该接口。

2.14 RK_MPI_AVG_SendPipeFrame

【描述】

用户向 AVS 组上的 PIPE 发送图像。

【语法】

```
RK_S32 RK_MPI_AVG_SendPipeFrame(AVS\_GRP AVSGrp, AVS\_PIPE AVSPipe, const  
                                VIDEO_FRAME_INFO_S* pstVideoFrame, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围: [0, AVS_MAX_GRP_NUM)。	输入
AVSPipe	用户送入图像的管道号。 取值范围: [0, AVS_PIPE_NUM)。	输入
pstVideoFrame	待发送的图像信息。具体描述请参见系统控制章节。	输入
s32MilliSec	超时参数 s32MilliSec 设为-1 时, 为阻塞接口; 0 时为非阻塞接口; 大于 0 时为超时等待时间, 超时时间的单位为毫秒 (ms)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AVS错误码 。

【注意】

- 建议整组输入BUF一齐送入AVS，整组送入均成功的情况下，如果不做其他使用，建议立即释放输入BUF；如果返回失败，必须释放整组BUF。

2.15 RK_MPI_AVSSetChnAttr

【描述】

设置AVS通道属性。

【语法】

```
RK_S32 RK_MPI_AVSSetChnAttr(AVS\_GRP AVSGrp, const AVS\_CHN AVSChn,
AVS\_CHN\_ATTR\_S* pstChnAttr);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS组号。 取值范围：[0, AVS_MAX_GRP_NUM)。	输入
AVSChn	AVS通道号。 AVS_MODE_BLEND模式下，取值范围[0, AVS_MAX_CHN_NUM) AVS_MODE_NOBLEND_VER、 AVS_MODE_NOBLEND_HOR和 AVS_MODE_NOBLEND_QR模式下只能取0。 在 AVS_PROJECTION_CUBE_MAP 投影模式下，只能取0。	输入
pstChnAttr	AVS通道属性	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AVS错误码 。

【注意】

- 组必须已经创建。
- 必须要先设置通道0的属性，才能设置其他通道的属性。
- 在[AVS_PROJECTION_CUBE_MAP](#)投影模式下，只能设置0通道的属性。

2.16 RK_MPI_AVSSetChnAttr

【描述】

获取AVS通道属性。

【语法】

```
RK_S32 RK_MPI_AVSSetChnAttr(AVS\_GRP AVSGrp, AVS\_CHN AVSChn,
AVS\_CHN\_ATTR\_S* pstChnAttr);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围: [0, AVS_MAX_GRP_NUM)。	输入
AVSChn	AVS 通道号。 取值范围[0, AVS_MAX_CHN_NUM)。	输入
pstChnAttr	AVS 通道属性	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AVS错误码 。

【注意】

- 组必须已经创建。

2.17 RK_MPI_AVIS_EnableChn

【描述】

启用 AVS 通道。

【语法】

```
RK_S32 RK_MPI_AVIS_EnableChn(AVS\_GRP AVSGrp, AVS\_CHN AVSChn);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围: [0, AVS_MAX_GRP_NUM)。	输入
AVSChn	AVS 通道号。 取值范围[0, AVS_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AVS错误码 。

【注意】

- GROUP 必须已创建。
- 必须先使用 [RK_MPI_AVIS_SetChnAttr](#) 接口设置通道属性才能启用通道。
- 通道 0 启用后才能启用其他的通道。
- 在 [AVS_PROJECTION_CUBE_MAP](#) 投影模式下，只能使能 0 通道。
- 多次启用通道返回成功。

2.18 RK_MPI_AVIS_DisableChn

【描述】

禁用 AVS 通道。

【语法】

```
RK_S32 RK_MPI_AVN_DisableChn(AVS\_GRP AVSGrp, AVS\_CHN AVSChn);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围: [0, AVS_MAX_GRP_NUM)。	输入
AVSChn	AVS 通道号。 取值范围[0, AVS_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AVS错误码 。

【注意】

- 组必须已经创建。
- 通道 0 要在其他的通道都禁用之后才能禁用。
- 重复禁用通道返回成功。

2.19 RK_MPI_AVN_GetChnFrame

【描述】

用户获取一帧通道图像。

【语法】

```
RK_S32 RK_MPI_AVN_GetChnFrame(AVS\_GRP AVSGrp, AVS\_CHN AVSChn,  
VIDEO_FRAME_INFO_S* pstVideoFrame, RK_S32 s32MilliSec);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围: [0, AVS_MAX_GRP_NUM)。	输入
AVSChn	AVS 通道号。 取值范围[0, AVS_MAX_CHN_NUM)。	输入
pstVideoFrame	获取的图像信息。具体描述请参见系统控制章节。	输入
s32MilliSec	超时参数 s32MilliSec 设为-1 时, 为阻塞接口; 0 时为非阻塞接口; 大于 0 时为超时等待时间, 超时时间的单位为毫秒 (ms) 。	输入

【返回值】

返回值	描述
0	成功。
非0	失败, 其值为 AVS错误码 。

【注意】

- 组必须已经创建。
- 只有设置通道队列深度不为 0，才能获取到图像。
- AVS 后级绑定且不需要调用该接口时，建议设置通道队列深度不为 0。
- 获取的图像要及时释放，否则将造成 AVS 内部处理 Buffer 不足的情况，建议与 [RK_MPI_AVN_ReleaseChnFrame](#) 接口配对使用。

2.20 RK_MPI_AVN_ReleaseChnFrame

【描述】

用户释放一帧通道图像。

【语法】

```
RK_S32 RK_MPI_AVN_ReleaseChnFrame(AVS\_GRP AVSGrp, AVS\_CHN AVSChn,
                                     const VIDEO_FRAME_INFO_S* pstVideoFrame);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围：[0, AVS_MAX_GRP_NUM)。	输入
AVSChn	AVS 通道号。 取值范围[0, AVS_MAX_CHN_NUM)。	输入
pstVideoFrame	图像信息。具体描述请参见系统控制章节。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为 AVS错误码 。

【注意】

- 实际上，此接口的 AVSGrp 和 AVSChn 参数并无实际用途，可在取值范围内任意设置。
- 此接口需与 [RK_MPI_AVN_GetChnFrame](#) 配对使用。

2.21 RK_MPI_AVN_GetFinalLut

【描述】

获取最终查找表。

【语法】

```
RK_S32 RK_MPI_AVN_GetFinalLut(AVS\_GRP AVSGrp, AVS_FINAL_LUT_S *pstFinalLut);
```

【参数】

参数名	描述	输入/输出
AVSGrp	AVS 组号。 取值范围：[0, AVS_MAX_GRP_NUM)。	输入
pstFinalLut	最终查找表。	输入

【注意】

- 组必须已经创建。
- 调用该接口前，pstFinalLut 中的 BLK 需要预先申请分配。

- RV1106/RV1109/RV11126 平台支持该接口，目前仅支持获取到LDCH 数据，仅能在AVS 非运行状态下调用。
- RK3588 平台不支持该接口。

3. 数据类型

AVS 模块相关数据类型定义如下：

- [AVS_MAX_GRP_NUM](#) : 定义 AVS 最大组个数。
- [AVS_PIPE_NUM](#) : 定义 AVS 最大管道个数。
- [AVS_MAX_CHN_NUM](#) : 定义 AVS 最大通道个数。
- [AVS_SPLIT_NUM](#) : 定义 AVS 在 7 路或者 8 路拼接的分块个数。
- [AVS_SPLIT_PIPE_NUM](#) : 定义 AVS 在 7 路或者 8 路拼接时一个分块中最多的拼接路数。
- [AVS_CUBE_MAP_SURFACE_NUM](#) : 定义 AVS 立方体投影的面的个数。
- [AVS_MAX_IN_WIDTH](#) : 定义 AVS 输入图像的最大宽度。
- [AVS_MAX_IN_HEIGHT](#) : 定义 AVS 输入图像的最大高度。
- [AVS_MIN_IN_WIDTH](#) : 定义 AVS 输入图像的最小宽度。
- [AVS_MIN_IN_HEIGHT](#) : 定义 AVS 输入图像的最小高度。
- [AVS_MAX_OUT_WIDTH](#) : 定义 AVS 输出图像的最大宽度。
- [AVS_MAX_OUT_HEIGHT](#) : 定义 AVS 输出图像的最大高度。
- [AVS_MIN_OUT_WIDTH](#) : 定义 AVS 输出图像的最小宽度。
- [AVS_MIN_OUT_HEIGHT](#) : 定义 AVS 输出图像的最小高度。
- [AVS_GRP](#) : 定义 AVS 组类型。
- [AVS_PIPE](#) : 定义 AVS 管道类型。
- [AVS_CHN](#) : 定义 AVS 通道类型。
- [AVS_MODE_E](#) : 定义 AVS 工作模式。
- [AVS_LUT_ACCURACY_E](#) : 定义查找表的精度。
- [AVS_FUSE_WIDTH_E](#) : 定义查找表的低频层融合带宽度。
- [AVS_LUT_STEP_E](#) : 定义查找表的采样步长。
- [AVS_STEP_ATTR_S](#) : 定义查找表的采样步长属性。
- [AVS_LUT_S](#) : 定义查找表属性。
- [AVS_CALIB_S](#) : 定义标定属性。
- [AVS_PROJECTION_MODE_E](#) : 定义拼接输出投影模式。
- [AVS_GAIN_MODE_E](#) : 定义亮度增益补偿模式。
- [AVS_GAIN_ATTR_S](#) : 定义亮度增益补偿属性。
- [AVS_ROTATION_S](#) : 定义 AVS 旋转属性。
- [AVS_FOV_S](#) : 定义 AVS 视场角。
- [AVS_SPLIT_ATTR_S](#) : 定义 7 路或者 8 路拼接时的分割属性。
- [AVS_CUBE_MAP_ATTR_S](#) : 定义立方体投影属性。
- [AVS_INPUT_ATTR_S](#) : 定义 AVS 拼接输入属性。
- [AVS_OUTPUT_ATTR_S](#) : 定义 AVS 拼接输出属性。
- [AVS_GRP_ATTR_S](#) : 定义 AVS 组属性。
- [AVS_PIPE_ATTR_S](#) : 定义 AVS 管道属性。
- [AVS_CHN_ATTR_S](#) : 定义 AVS 通道属性。
- [AVS_ROI_PARAM_S](#) : 定义 AVS 组感兴趣区域。
- [AVS_FINAL_LUT_S](#) : 定义最终查找表属性。

3.1 AVS_MAX_GRP_NUM

【说明】

定义 AVS 最大组个数。

【定义】

RK3588

```
#define AVS_MAX_GRP_NUM 32
```

RV1106

```
#define AVS_MAX_GRP_NUM 4
```

RV1109/RV1126

```
#define AVS_MAX_GRP_NUM 4
```

【注意事项】

无。

3.2 AVS_PIPE_NUM

【说明】

定义 AVS 最大管道个数。

【定义】

RK3588

```
#define AVS_PIPE_NUM 8
```

RK3576

```
#define AVS_PIPE_NUM 3
```

RV1106

```
#define AVS_PIPE_NUM 2
```

RV1109/RV1126

```
#define AVS_PIPE_NUM 2
```

【注意事项】

无。

3.3 AVS_MAX_CHN_NUM

【说明】

定义 AVS 最大通道个数。

【定义】

RK3588

```
#define AVS_MAX_CHN_NUM 2
```

RV1106

```
#define AVS_MAX_CHN_NUM 2
```

RV1109/RV1126

```
#define AVS_MAX_CHN_NUM 2
```

【注意事项】

无。

3.4 AVS_SPLIT_NUM

【说明】

定义 AVS 在 7 路或者 8 路拼接的分块个数。

【定义】

```
#define AVS_SPLIT_NUM 2
```

【注意事项】

无。

3.5 AVS_SPLIT_PIPE_NUM

【说明】

定义 AVS 在 7 路或者 8 路拼接时一个分块中最多的拼接路数。

【定义】

```
#define AVS_SPLIT_PIPE_NUM 6
```

【注意事项】

无。

3.6 AVS_CUBE_MAP_SURFACE_NUM

【说明】

定义 AVS 立方体投影的面的个数。

【定义】

```
#define AVS_CUBE_MAP_SURFACE_NUM 6
```

【注意事项】

无。

3.7 AVS_MAX_IN_WIDTH

【说明】

定义 AVS 输入图像的最大宽度。

【定义】

RK3588

```
#define AVS_MAX_IN_HEIGHT 4096
```

RV1106

```
#define AVS_MAX_IN_HEIGHT 4096
```

RV1109/RV1126

```
#define AVS_MAX_IN_HEIGHT 4096
```

【注意事项】

无。

3.8 AVS_MAX_IN_HEIGHT

【说明】

定义 AVS 输入图像的最大高度。

【定义】

RK3588

```
#define AVS_MAX_IN_HEIGHT 4096
```

RV1106

```
#define AVS_MAX_IN_HEIGHT 4096
```

RV1109/RV1126

```
#define AVS_MAX_IN_HEIGHT 4096
```

【注意事项】

无。

3.9 AVS_MIN_IN_WIDTH

【说明】

定义 AVS 输入图像的最小宽度。

【定义】

RK3588

```
#define AVS_MIN_IN_HEIGHT 1280
```

RV1106

```
#define AVS_MIN_IN_HEIGHT 720
```

RV1109/RV1126

```
#define AVS_MIN_IN_HEIGHT 1280
```

【注意事项】

无。

3.10 AVS_MIN_IN_HEIGHT

【说明】

定义 AVS 输入图像的最小高度。

【定义】

RK3588

```
#define AVS_MIN_IN_HEIGHT 720
```

RV1106

```
#define AVS_MIN_IN_HEIGHT 720
```

RV1109/RV1126

```
#define AVS_MIN_IN_HEIGHT 720
```

【注意事项】

无。

3.11 AVS_MAX_OUT_WIDTH

【说明】

定义 AVS 输出图像的最大宽度。

【定义】

RK3588

```
#define AVS_MAX_OUT_HEIGHT 16383
```

RV1106

```
#define AVS_MAX_OUT_HEIGHT 5120
```

RV1109/RV1126

```
#define AVS_MAX_OUT_HEIGHT 4096
```

【注意事项】

无。

3.12 AVS_MAX_OUT_HEIGHT

【说明】

定义 AVS 输出图像的最大高度。

【定义】

RK3588

```
#define AVS_MAX_OUT_HEIGHT 10000
```

RV1106

```
#define AVS_MAX_OUT_HEIGHT 5120
```

RV1109/RV1126

```
#define AVS_MAX_OUT_HEIGHT 4096
```

【注意事项】

无。

3.13 AVS_MIN_OUT_WIDTH

【说明】

定义 AVS 输出图像的最小宽度。

【定义】

RK3588

```
#define AVS_MIN_OUT_WIDTH 256
```

RV1106

```
#define AVS_MIN_OUT_WIDTH 256
```

RV1109/RV1126

```
#define AVS_MIN_OUT_WIDTH 256
```

【注意事项】

无。

3.14 AVS_MIN_OUT_HEIGHT

【说明】

定义 AVS 输出图像的最小高度。

【定义】

RK3588

```
#define AVS_MIN_OUT_HEIGHT 256
```

RV1106

```
#define AVS_MIN_OUT_HEIGHT 256
```

RV1109/RV1126

```
#define AVS_MIN_OUT_HEIGHT 256
```

【注意事项】

无。

3.15 AVS_GRP

【说明】

定义 AVS 组 类型。

【定义】

```
typedef RK_S32 AVS_GRP;
```

【注意事项】

无。

3.16 AVS_PIPE

【说明】

定义 AVS PIPE 类型。

【定义】

```
typedef RK_S32 AVS_PIPE;
```

【注意事项】

无。

3.17 AVS_CHN

【说明】

定义 AVS 通道类型。

【定义】

```
typedef RK_S32 AVS_CHN;
```

【注意事项】

无。

3.18 AVS_MODE_E

【说明】

定义 AVS 工作模式。

【定义】

```
typedef enum rkAVS_MODE_E {
    AVS_MODE_BLEND    = 0,
    AVS_MODE_NOBLEND_VER = 1,
    AVS_MODE_NOBLEND_HOR = 2,
    AVS_MODE_NOBLEND_QR = 3,
    AVS_MODE_NOBLEND_OVL = 4,
    AVS_MODE_BLEND_DYN = 5,
    AVS_MODE_BUTT
} AVS_MODE_E;
```

【成员】

成员名称	描述
AVS_MODE_BLEND	融合拼接模式。 根据 LUT 拼接，在拼接处有融合。
AVS_MODE_NOBLEND_VER	垂直非融合拼接模式。 将输入的各路输入垂直放在一起，拼接处没有融合。
AVS_MODE_NOBLEND_HOR	水平非融合拼接模式。 将输入的各路图像水平放在一起，拼接处没有融合。
AVS_MODE_NOBLEND_QR	田字形非融合拼接模式。 仅支持4路图像的拼接，两行摆放，每行两路放在一起，拼接处没有融合。
AVS_MODE_NOBLEND_OVL	叠加非融合拼接模式。 将输入的各路图像根据管道属性叠加放置，拼接处没有融合。
AVS_MODE_BLEND_DYN	动态融合拼接模式。 根据 LUT 拼接，在拼接处有融合且随着图像场景变化。

【注意事项】

- 所有模式下，输出图像的动态范围决定于输入图像决定。
- 所有模式下，输出图像的宽度、高度由通道属性决定。
- [AVS_MODE_NOBLEND_QR](#) 模式下，拼接路数仅支持 4 路。
- [AVS_MODE_NOBLEND_OVL](#) 模式，仅 RV1106 平台支持。
- [AVS_MODE_BLEND_DYN](#) 模式，仅 RK3576/RK3588 平台支持。
- 各模式下最大最小的输入分辨率如下表所示。

3.18.1 表2 RK3588 平台各模式下最大最小的输入分辨率

拼接模式	分辨率			
	最小宽度	最大宽度	最小高度	最大高度
AVS_MODE_BLEND	1280	AVS_MAX_IN_WIDTH	720	AVS_MAX_IN_HEIGHT
AVS_MODE_NOBLEND_VER	1280	16384	720	16384
AVS_MODE_NOBLEND_HOR	1280	16384	720	16384
AVS_MODE_NOBLEND_QR	1280	16384	720	16384

3.19 AVS_LUT_ACCURACY_E

【说明】

定义查找表的精度。

【定义】

```
typedef enum rkAVS_LUT_ACCURACY_E {
    AVS_LUT_ACCURACY_HIGH = 0,
    AVS_LUT_ACCURACY_LOW = 1,
    AVS_LUT_ACCURACY_BUTT
} AVS_LUT_ACCURACY_E;
```

【成员】

成员名称	描述
AVS_LUT_ACCURACY_HIGH	查找表高精度。
AVS_LUT_ACCURACY_LOW	查找表低精度。

【注意事项】

- 查找表的精度须与标定时的精度选择一致，否则 AVS 无法正常工作。
- 查找表精度高，拼接更准确，但拼接性能会比低精度查找表时有所下降。

3.20 AVS_PARAM_SOURCE_E

【说明】

定义融合模式的参数来源类型。

【定义】

```
typedef enum rkAVS_PARAM_SOURCE_E {
    AVS_PARAM_SOURCE_LUT = 0,
    AVS_PARAM_SOURCE_CALIB = 1,

    AVS_PARAM_SOURCE_MODE_BUT
} AVS_PARAM_SOURCE_E;
```

【成员】

成员名称	描述
AVS_PARAM_SOURCE_LUT	参数来源自查找表。
AVS_PARAM_SOURCE_CALIB	参数来源自标定文件。

【注意事项】

- RK3588 平台两种均支持，AVS_PARAM_SOURCE_CALIB 支持传入 pto、xml 文件。
- RV1106 平台仅支持 AVS_PARAM_SOURCE_CALIB。
- RV1109/RV1126 平台仅支持 AVS_PARAM_SOURCE_CALIB。

3.21 AVS_FUSE_WIDTH_E

【说明】

定义查找表的低频层融合带宽度。

【定义】

```
typedef enum rkAVS_FUSE_WIDTH_E {
    AVS_FUSE_WIDTH_HIGH = 0,
    AVS_FUSE_WIDTH_MEDIUM = 1,
    AVS_FUSE_WIDTH_LOW = 2,

    AVS_FUSE_WIDTH_BUTT
} AVS_FUSE_WIDTH_E;
```

【成员】

成员名称	描述
AVS_FUSE_WIDTH_HIGH	融合带宽度 128 像素。
AVS_FUSE_WIDTH_MEDIUM	融合带宽度 256 像素。
AVS_FUSE_WIDTH_LOW	融合带宽度 512 像素。

【注意事项】

3.22 AVS_LUT_STEP_E

【说明】

定义查找表的采样步长。

【定义】

```
typedef enum rkAVS_LUT_STEP_E {
    AVS_LUT_STEP_HIGH = 0,
    AVS_LUT_STEP_MEDIUM = 1,
    AVS_LUT_STEP_LOW = 2,
    AVS_LUT_STEP_BUTT
} AVS_LUT_STEP_E;
```

【成员】

成员名称	描述
AVS_LUT_STEP_HIGH	查找表采样步长 16 像素。
AVS_LUT_STEP_MEDIUM	查找表采样步长 32 像素。
AVS_LUT_STEP_LOW	查找表采样步长 64 像素。

【注意事项】

- 采样步长越小，查找表越大，需要的计算时间也越长。

3.23 AVS_STEP_ATTR_S

【说明】

定义查找表的采样步长属性。

【定义】

```
typedef struct rkAVS_LUT_STEP_ATTR_S {
    AVS_LUT_STEP_E enStepX;
    AVS_LUT_STEP_E enStepY;
} AVS_STEP_ATTR_S;
```

【成员】

成员名称	描述
enStepX	查找表在 X 方向上的采样步长。
enStepY	查找表在 Y 方向上的采样步长。

【注意事项】

3.24 AVS_LUT_S

【说明】

定义查找表属性。

【定义】

```
typedef struct rkAVS_LUT_S {  
    AVS_LUT_ACCURACY_E enAccuracy;  
    AVS_FUSE_WIDTH_E enFuseWidth;  
    AVS_STEP_ATTR_S stLutStep;  
    RK_VOID *pVirAddr[AVS_PIPE_NUM];  
} AVS_LUT_S;
```

【成员】

成员名称	描述
enAccuracy	查找表的精度，须和标定的配置一致。
enFuseWidth	查找表的低频层融合带宽度。
stLutStep	查找表的采样步长。
u64PhyAddr	各路查找表的虚拟地址，查找表数据来自标定。

【注意事项】

- 用户须保证查找表的虚拟地址和其中的内容正确有效，不然可能会发生卡死的情况。
- enAccuracy 须和标定时的值一致。

【相关数据类型及接口】

AVS 拼接输入属性[AVS_INPUT_ATTR_S](#)。

3.25 AVS_CALIB_S

【说明】

定义标定文件属性

【定义】

```
typedef struct rkAVS_CALIB_S {  
    const RK_CHAR *pCalibFilePath;  
    const RK_CHAR *pMeshAlphaPath;  
} AVS_CALIB_S;
```

【注意事项】

- 用户须保证标定文件的文件路径和其中的内容正确有效，不然可能会发生卡死的情况。

3.26 AVS_PROJECTION_MODE_E

【说明】

定义拼接输出投影模式。

【定义】

```

typedef enum rkAVS_PROJECTION_MODE_E {
    AVS_PROJECTION_EQUIRECTANGULAR = 0,
    AVS_PROJECTION_RECTILINEAR = 1,
    AVS_PROJECTION_CYLINDRICAL = 2,
    AVS_PROJECTION_CUBE_MAP = 3,
    AVS_PROJECTION_EQUIRECTANGULAR_TRANS = 4,
    AVS_PROJECTION_BUTT
} AVS_PROJECTION_MODE_E;

```

【成员】

成员名称	描述
AVS_PROJECTION_EQUIRECTANGULAR	等距柱面投影模式。
AVS_PROJECTION_RECTILINEAR	直线投影模式。
AVS_PROJECTION_CYLINDRICAL	柱面投影模式。
AVS_PROJECTION_CUBE_MAP	立方体投影模式。
AVS_PROJECTION_EQUIRECTANGULAR_TRANS	横向等距柱面投影模式。

【注意事项】

- 拼接路数超过 [AVS_SPLIT_PIPE_NUM](#) 时，不支持 直线投影模式[AVS_PROJECTION_RECTILINEAR](#) 和 立方体投影模式[AVS_PROJECTION_CUBE_MAP](#)。
- 暂未支持立方体投影模式[AVS_PROJECTION_CUBE_MAP](#)。
- RV1106 平台仅支持 [AVS_PROJECTION_EQUIRECTANGULAR](#) 等距柱面投影模式。
- RV1109/RV1126 平台仅支持 [AVS_PROJECTION_EQUIRECTANGULAR](#) 等距柱面投影模式。

【相关数据类型及接口】

无。

3.27 AVS_GAIN_MODE_E

【说明】

定义亮度增益补偿模式。

【定义】

```

typedef enum rkAVS_GAIN_MODE_E {
    AVS_GAIN_MODE_MANUAL = 0,
    AVS_GAIN_MODE_AUTO = 1,
    AVS_GAIN_MODE_BUTT
} AVS_GAIN_MODE_E;

```

【成员】

成员名称	描述
AVS_GAIN_MODE_MANUAL	手动模式，用户手动配置亮度增益补偿值。
AVS_GAIN_MODE_AUTO	自动模式，AVS 自动统计拼接处的亮度差异，自动进行亮度增益补偿。

【注意事项】

- 暂未使用，预留数据类型。

【相关数据类型及接口】

- AVS 亮度增益补偿属性[AVS_GAIN_ATTR_S](#)。

3.28 AVS_GAIN_ATTR_S

【说明】

定义亮度增益补偿属性。

【定义】

```
typedef struct rkAVS_GAIN_ATTR_S {  
    AVS_GAIN_MODE_E enMode;  
    RK_S32     s32Coef[AVS_PIPE_NUM];  
} AVS_GAIN_ATTR_S;
```

【成员】

成员名称	描述
enMode	亮度增益模式。
s32Coef[AVS_PIPE_NUM]	在手动模式下的各路亮度增益值。 取值范围: [0, 65535]。 配置为 16384 时亮度不校正, 比 16384 小时图像变暗, 比 16384 大时图像变亮。

【注意事项】

- 暂未使用, 预留数据类型。

3.29 AVS_ROTATION_S

【说明】

定义 AVS 旋转属性。

【定义】

```
typedef struct rkAVS_ROTATION_S {  
    RK_S32 s32Yaw;  
    RK_S32 s32Pitch;  
    RK_S32 s32Roll;  
} AVS_ROTATION_S;
```

【成员】

成员名称	描述
s32Yaw	偏航角。 取值范围: [-18000, 18000], 单位: 0.01°。
s32Pitch	俯仰角。 取值范围: [-1000, 1000], 单位: 0.01°。
s32Roll	翻滚角。 取值范围: [-1000, 1000], 单位: 0.01°。

【注意事项】

无。

3.30 AVS_FOV_S

【说明】

定义 AVS 视场角。

【定义】

```
typedef struct rkAVS_FOV_S {  
    RK_S32 u32FOVX;  
    RK_S32 u32FOVY;  
} AVS_FOV_S;
```

【成员】

成员名称	描述
u32FOVX	水平方向上的视场角，单位：0.01°。
u32FOVY	垂直方向上的视场角，单位：0.01°。

【注意事项】

- 当投影模式为 [AVS_PROJECTION_CUBE_MAP](#) 时，该参数无效。
- 各投影模式下，水平方向、垂直方向上的视场角具有不同的取值范围，具体参考下表。

视场角	投影模式	取值范围
水平方向	AVS_PROJECTION_EQUIRECTANGULAR	[1000, 36000]
	AVS_PROJECTION_RECTILINEAR	[1000, 11200]
	AVS_PROJECTION_CYLINDRICAL	[1000, 36000]
	AVS_PROJECTION_EQUIRECTANGULAR_TRANS	[1000, 36000]
垂直方向	AVS_PROJECTION_EQUIRECTANGULAR	[1000, 18000]
	AVS_PROJECTION_RECTILINEAR	[1000, 11200]
	AVS_PROJECTION_CYLINDRICAL	[1000, 11200]
	AVS_PROJECTION_EQUIRECTANGULAR_TRANS	[1000, 18000]

【相关数据类型及接口】

- 拼接输出投影模式 [AVS_PROJECTION_MODE_E](#)

3.31 AVS_SPLIT_ATTR_S

【说明】

定义 7 路或者 8 路拼接时的分割属性。

【定义】

```
typedef struct rkAVS_SPLIT_ATTR_S {  
    RK_U32 u32PipeNum;  
    AVS_PIPE AVSPipe[AVS_SPLIT_PIPE_NUM];  
} AVS_SPLIT_ATTR_S;
```

【注意事项】

- 用户需要保证分割属性的正确性，否则输出图可能会出现黑块。
- 只有拼接路数超过 [AVS_SPLIT_PIPE_NUM](#) 时，该参数才会生效。

3.32 AVS_CUBE_MAP_ATTR_S

【说明】

定义立方体投影属性。

【定义】

```
typedef struct rkAVS_CUBE_MAP_ATTR_S {
    RK_BOOL bBgColor;
    RK_U32 u32BgColor;
    RK_U32 u32SurfaceLength;
    POINT_S stStartPoint[AVS_CUBE_MAP_SURFACE_NUM];
} AVS_CUBE_MAP_ATTR_S;
```

【成员】

成员名称	描述
bBgColor	是否使能背景色。
u32BgColor	背景色属性，格式 RGB888。
u32SurfaceLength	立方体投影的单个面的边长，要求 4 对齐。 取值范围: [256, 4096]。
stStartPoint[AVS_CUBE_MAP_SURFACE_NUM]	立方体投影的每一个面在输出图像上的起点位置。不能超出图像， 要求 4 对齐。

【注意事项】

- 暂未支持，预留数据类型。
- 使能背景色，会增加 VGS 模块的性能和 MB 的消耗，如果展开的面填满了输出图像，建议关闭背景色。
- 展开面的大小不能超出输出图像的大小。
- 在 [AVS_PROJECTION_CUBE_MAP](#) 拼接模式下进行展开时，如果输出配置为压缩，则必须使能背景色。
- 在 [AVS_PROJECTION_CUBE_MAP](#) 拼接模式下进行展开时，如果输出配置为压缩，则起点的 X 坐标要求 256 对齐，
u32SurfaceLength 也要求 256 对齐。

【相关数据类型及接口】

拼接输出投影模式 [AVS_PROJECTION_MODE_E](#)

3.33 AVS_OVERLAY_ATTR_S

【说明】

定义 AVS 叠加属性。

【定义】

```
typedef struct rkAVS_OVERLAY_ATTR_S {
    RK_BOOL bBgColor;
    RK_U32 u32BgColor;
} AVS_OVERLAY_ATTR_S;
```

【成员】

成员名称	描述
bBgColor	是否使能背景色。
u32BgColor	背景色属性，格式 RGB888。

【注意事项】

- 预留接口，暂未实现。

【相关数据类型及接口】

3.34 AVS_INPUT_ATTR_S

【说明】

定义 AVS 拼接输入属性。

【定义】

```
typedef struct rkAVS_INPUT_ATTR_S {
    AVS_PARAM_SOURCE_E enParamSource;
    union {
        AVS_LUT_S     stLUT;
        AVS_CALIB_S   stCalib;
    };
    SIZE_S         stSize;
} AVS_INPUT_ATTR_S;
```

【成员】

成员名称	描述
enParamSource	拼接输入参数来源。
stLUT	查找表属性。
stCalib	标定属性。
stSize	输入图像分辨率。

【注意事项】

- enParamSource 拼接输入参数来源类型应与 stLut、stCalib 对应配置。
- 仅当组的工作模式为 融合模式[AVS_MODE_BLEND](#) 时，该参数生效。
- 各芯片平台参数限制参考 融合模式的参数来源类型 [AVS_PARAM_SOURCE_E](#)。
- 在RV1106\RV1109\RV1126 平台在融合模式必须配置合法的 stSize，参数具体范围详见 [AVS_MIN_IN_HEIGHT](#)、[AVS_MIN_IN_WIDTH](#)、[AVS_MAX_IN_HEIGHT](#)、[AVS_MAX_IN_WIDTH](#)。

【相关数据类型及接口】

- AVS 工作模式 [AVS_MODE_E](#)
- AVS 融合模式的参数来源类型 [AVS_PARAM_SOURCE_E](#)

3.35 AVS_OUTPUT_ATTR_S

【说明】

定义 AVS 拼接输出属性。

【定义】

```

typedef struct rkAVS_OUTPUT_ATTR_S {
    AVS_PROJECTION_MODE_E enPrjMode;           /* RW; Projection mode. */
    POINT_S stCenter;                         /* Center point. */
    AVS_FOV_S stFOV;                          /* Output FOV. */
    AVS_ROTATION_S stORIRotation;             /* Output original rotation. */
    AVS_ROTATION_S stRotation;                /* Output rotation. */
    AVS_SPLIT_ATTR_S stSplitAttr[AVS_SPLIT_NUM]; /* Split attribute for 7 or 8 inputs stitching. */
    AVS_CUBE_MAP_ATTR_S stCubeMapAttr;         /* Cube map attribute. */
    SIZE_S stSize;                           /* Mesh resolution in AVS_MODE_BLEND mode; Canvas resolution in
AVS_MODE_NOBLEND_OVERLAY mode. */
    RK_FLOAT fDistance;                      /* Optimum stitch distance. */
    AVS_OVERLAY_ATTR_S stOverlayAttr;          /* Overlay attribute. */
} AVS_OUTPUT_ATTR_S;

```

【成员】

成员名称	描述
enPrjMode	拼接输出的投影模式。
stCenter	投影中心在输出图中的位置。一般设置为输出图像的中心点，表示投影中心和输出图像中心点重合。
stFOV	拼接输出区域的视场角。
stORIRotation	拼接输出起始旋转角度属性。
stRotation	拼接输出旋转属性。
stSplitAttr[AVS_SPLIT_NUM]	7路或者8路拼接时的分割属性。无效参数，暂未支持。
stCubeMapAttr	立方体投影属性。无效参数，暂未支持。
stSize	AVS_MODE_BLEND 模式下的 MESH 分辨率。 AVS_MODE_NOBLEND_OVERLAY 模式下的画布分辨率。
fDistance	最佳拼接距离。静态参数。
stOverlayAttr	非融合拼接叠加属性。

【注意事项】

- stCenter 投影中心即为三维球面全景图转换到二维投影输出图的投影中心位置，当投影模式为 [AVS_PROJECTION_CUBE_MAP](#) 时，stCenter 不生效。
- 一般情况下，投影中心与输出图像中心重合，即投影输出图像的视场角 FOV 上下左右对称。如需要对输出图像进行裁剪，仅显示部分视野，则可以减小视场角 FOV，同时通过移动投影中心点，选择感兴趣的区域作为输出图。
- 投影中心点的 X 和 Y 的范围为(-16384, 16384)。
- 投影中心点和视场角 FOV 需要同时调节，否则输出图像会出现边缘拉伸、像素填充的情况。
- 投影中心点移动之后，超过最大视场角FOV 区域不显示图像，可以通过减小FOV 的配置来裁剪超出部分。
- fDistance 最佳拼接距离仅在 AVS_MODE_BLEND 模式下生效。
- fDistance 最佳拼接距离范围为 [0.5, 1000]。
- stOverlayAttr 叠加属性仅在 RV1106 平台下生效，且仅在 AVS_MODE_NOBLEND_OVERLAY 模式下生效。

【相关数据类型及接口】

- [AVS_PROJECTION_MODE_E](#)
- [AVS_OVERLAY_ATTR_S](#)
- [AVS_FOV_S](#)
- [AVS_ROTATION_S](#)

3.36 AVS_GRP_ATTR_S

【说明】

定义 AVS 组属性。

【定义】

```
typedef struct rkAVS_GRP_ATTR_S {
    AVS_MODE_E      enMode;
    RK_U32         u32PipeNum;
    RK_BOOL        bSyncPipe;
    AVS_INPUT_ATTR_S stInAttr;
    AVS_GAIN_ATTR_S stGainAttr;
    RK_U64          u64BBoxPhyAddr[AVS_PIPE_NUM];
    AVS_OUTPUT_ATTR_S stOutAttr;
    FRAME_RATE_CTRL_S stFrameRate;
} AVS_GRP_ATTR_S;
```

【成员】

成员名称	描述
enMode	拼接模式。静态属性，创建 GROUP 时设定，不可更改。
u32PipeNum	PIPE 数，即拼接路数。取值范围[1, AVS_PIPE_NUM]。静态属性，创建 GROUP 时设定，不可更改。
bSyncPipe	是否通过帧序 SeqID 进行各路图像的同步。静态属性，创建 GROUP 时设定，不可更改。
stInAttr	拼接输入属性。
stGainAttr	亮度增益补偿属性。暂未使用。
u64BBoxPhyAddr[AVS_PIPE_NUM]	各路图像的 Bounding Box 数据的物理地址。暂未使用。
stOutAttr	拼接输出属性。
stFrameRate	帧率属性。具体描述请参见系统控制章节。暂未使用。

【注意事项】

- bSyncPipe 如果打开，AVS 会根据各路输入图像帧序 SeqID 进行同步，找出 SeqID 一致的一组图像进行拼接，会导致 AVS 占用的前级 MB 数目增多，且同步过程中会进行前级丢帧；bSyncPipe 关闭时，AVS 不会进行前级丢帧。
- AVS 与前级模块通过RK_MPI_SYS_Bind 函数建立绑定关系传递 MB 时，必须开启帧序同步 SyncPipe；建议在调用函数 [RK_MPI_AVN_SendPipeFrame](#) 送入 MB 时，不开启 bSyncPipe，需要帧序对齐的话则由用户实现。
- 当u32PipeNum 大于6时，仅支持等距柱面投影模式[AVS_PROJECTION_EQUIRECTANGULAR](#)、柱面投影模式 [AVS_PROJECTION_CYLINDRICAL](#) 和 横向等距柱面投影模式[AVS_PROJECTION_EQUIRECTANGULAR_TRANS](#)。

【相关数据类型及接口】

3.37 AVS_CHN_ATTR_S

【说明】

定义 AVS 通道属性。

【定义】

```
typedef struct rkAVS_CHN_ATTR_S {
    RK_U32         u32Width;
    RK_U32         u32Height;
    COMPRESS_MODE_E enCompressMode;
    DYNAMIC_RANGE_E enDynamicRange
    RK_U32         u32Depth;
    FRAME_RATE_CTRL_S stFrameRate;
    RK_U32         u32FrameBufCnt;
} AVS_CHN_ATTR_S;
```

【成员】

成员名称	描述
u32Width	通道宽度，单位：像素，需要对齐，具体描述请参考 输入输出特性 。
u32Height	通道高度，单位：像素，需要对齐，具体描述请参考 输入输出特性 。
enCompressMode	通道压缩模式。具体描述请参见“系统控制”章节。
enDynamicRange	通道动态范围。具体描述请参见“系统控制”章节。
u32Depth	通道图像队列深度。取值范围[0, 8]。
stFrameRate	帧率属性。具体描述请参见“系统控制”章节。
u32FrameBufCnt	最大目标图像缓冲个数 仅在通道缓存模式为private模式时生效。

【注意事项】

- 通道帧率控制功能暂不支持，配置不会生效。
- 仅通道 0 支持 COMPRESS_AFBC_16x16 压缩，其余通道不支持压缩。
- 其余通道的高度不能小于通道 0 的高度 15 分之 1，不能大于通道 0 的高度。
- 其余通道的高度不能小于通道 0 的宽度 15 分之 1，不能大于通道 0 的宽度的 2 分之 1。
- 通道图像队列深度在获取通道图像的时候才有用，如没有必要，建议设置为 0，否则会导致 AVS 模块 MB 占用的增加。
- 通道 0 的动态范围设置无效，通道 0 输出的动态范围和输入图像相同。
- u32FrameBufCnt 根据平台内存大小有默认设置，
 - RK3588 平台默认值为 3，且 u32FrameBufCnt < 3 时，会强制设置为 3，以保证 AVS 输出满足最小可用缓存。
 - RV1109/RV1126 平台默认值为 1，且 u32FrameBufCnt < 1 时，会强制设置为 1，以保证 AVS 输出满足最小可用缓存。
 - RV1106 平台无以上处理。
- RV1106 平台，u32FrameBufCnt 和 u32Depth 设置为同一数值时，可能出现无可用输出缓存的情况。

【相关数据类型及接口】

- [RK_MPI_AVSS_GetChnAttr](#)
- [RK_MPI_AVSS_SetChnAttr](#)

3.38 AVS_PIPE_ATTR_S

【说明】

定义 AVS 管道属性。

【定义】

```
typedef struct rkAVS_PIPE_ATTR_S {
    RECT_S stSrcRect;
    RECT_S stDstRect;
    RK_U32 u32Priority; /* RW; Overlay priority */
} AVS_PIPE_ATTR_S;
```

【成员】

成员名称	描述
stSrcRect	管道输入图像尺寸。
stDstRect	管道输出图像在图层画布上的显示区域。
u32Priority	管道优先级，通道显示位置有交叠时，高优先级通道覆盖低优先级通道。

【注意事项】

- stSrcRect 必须与输入图像尺寸相符。
- stDstRect 必须位于图层画布大小范围内，图层画布大小由 [AVS_OUTPUT_ATTR_S](#) 结构体的 stSize 参数指定。

【相关数据类型及接口】

- [AVS_OUTPUT_ATTR_S](#)

3.39 AVS_ROI_PARAM_S

【说明】

定义 AVS 组感兴趣区域。

【定义】

```
typedef struct rkAVS_ROI_PARAM_S {
    RK_BOOL    bEnable;      /* RW; Range: [0, 1]; enable. */
    RECT_S    stRect;        /* Regin of Interest */
} AVS_ROI_PARAM_S;
```

【成员】

成员名称	描述
bEnable	ROI 使能开关。
stRect	感兴趣区域。

【注意事项】

- bEnable 仅在 AVS_MODE_BLEND 拼接下才会生效。
- stRect 仅在 bEnable 使能时做异常参数检查。
- stRect 取值范围：
 - 水平位置X: [0, [AVS_MAX_OUT_WIDTH](#)]
 - 垂直位置Y: [0, [AVS_MAX_OUT_HEIGHT](#)]
 - 宽度: [[AVS_MIN_OUT_WIDTH](#), [AVS_MAX_OUT_WIDTH](#)]
 - 高度: [[AVS_MIN_OUT_HEIGHT](#), [AVS_MAX_OUT_HEIGHT](#)]
 - stRect 必须在已经配置的MESH 分辨率大小内，否则输出图像会出现边缘拉伸、像素填充的情况。

【相关数据类型及接口】

- [AVS_OUTPUT_ATTR_S](#)

3.40 AVS_FINAL_LUT_S

【说明】

定义最终查找表属性。

【定义】

```
typedef struct rkAVS_FINAL_LUT_S {
    MB_BLK pMeshBlk[AVS_PIPE_NUM];
    MB_BLK pAlphaBlk[AVS_PIPE_NUM];
    MB_BLK pLdchBlk[AVS_PIPE_NUM];
    MB_BLK pParamBlk[AVS_PIPE_NUM];
} AVS_FINAL_LUT_S;
```

【成员】

成员名称	描述
pMeshBlk	MESH 数据。
pAlphaBlk	ALPHA 数据。
pLdchBlk	LDCH 数据。
pParamBlk	参数信息。

【注意事项】

无。

4. AVS错误码

全景拼接 API AVS错误码如下所示：

错误代码	宏定义	描述
0xA0118001	RK_ERR_AVIS_INVALID_DEVID	AVS 组号无效
0xA0118002	RK_ERR_AVIS_INVALID_CHNID	AVS 通道号无效
0xA0118003	RK_ERR_AVIS_ILLEGAL_PARAM	AVS 参数设置无效
0xA0118004	RK_ERR_AVIS_EXIST	AVS 组已创建
0xA0118005	RK_ERR_AVIS_UNEXIST	AVS 组未创建
0xA0118006	RK_ERR_AVIS_NULL_PTR	输入参数空指针错误
0xA0118008	RK_ERR_AVIS_NOT_SUPPORT	操作不支持
0xA0118009	RK_ERR_AVIS_NOT_PERM	操作不允许
0xA011800A	RK_ERR_AVIS_INVALID_PIPEID	AVS 管道号无效
0xA011800C	RK_ERR_AVIS_NOMEM	分配内存失败
0xA011800D	RK_ERR_AVIS_NOBUF	分配 BUF 池失败
0xA011800E	RK_ERR_AVIS_BUF_EMPTY	图像队列为空
0xA011800F	RK_ERR_AVIS_BUF_FULL	图像队列已满
0xA0118010	RK_ERR_AVIS_NOTREADY	AVS 系统未初始化
0xA0118012	RK_ERR_AVIS_BUSY	AVS 系统忙

5. 相关文档

- RK3588 平台：
 - 《Rockchip AVS tool 使用教程》 Rockchip_Developer_Tutorial_AVIS_Tool_CN.pdf
 - 《Rockchip RK3588 AVS全景拼接调试指南》 Rockchip_Developer_Guide_RK3588_AVIS_CN.pdf
- RV1106\RV1106\RV1126 平台：
 - 《Rockchip 双目拼接产线标定指南》 Rockchip_Guide_AVIS_Calib_Product_CN.pdf
 - 《Rockchip 双目拼接模型标定指南》 Rockchip_Guide_AVIS_Calib_Model_CN.pdf

Dump调试信息说明

[概述](#)

[SYS](#)

[MB](#)

[VPSS](#)

[RGN](#)

[VGS](#)

[ADEC](#)

[AENC](#)

[AO](#)

[AI](#)

[VI](#)

[VO](#)

[TDE](#)

[VDEC](#)

[VENC](#)

[AVS](#)

[GDC](#)

[ALL](#)

1. 概述

调试信息可实时反映当前系统的运行状态，所记录的信息可供问题定位及分析时使用。

1.1 运行命令

dumpsys 模块名

1.2 模块清单

模块名称	描述
sys	记录当前SYS模块的使用情况
mb	记录当前MB模块的buffer 使用情况
vgs	视频图像处理子系统单元状态信息
vpss	记录当前 VPSS 属性配置以及状态信息
rgn	记录当前区域资源信息
adec	记录当前音频解码属性配置以及状态信息
aenc	记录当前AENC属性配置以及状态信息
ao	记录当前AO属性配置以及状态信息
ai	记录当前AO属性配置以及状态信息
tde	记录当前TDE模块的状态信息
vdec	记录当前视频解码属性配置以及状态信息
venc	记录当前视频编码属性配置以及状态信息
vi	记录当前VI属性配置及状态信息
vo	记录当前VO属性配置及状态信息
avs	记录当前AVS属性配置及状态信息
all	记录所有注册模块的配置和状态信息

【注意】

- 支持获取多个注册模块的属性配置和状态信息

命令格式：

dumpsys [mod1]-[mod2]-[mod3]-....

命令示例：

dumpsys vdec-vpss-vo

- 命令运行条件。
- 应用需调用RK_MPI_SYS_Init方法进行初始化后，才能执行dumpsys命令。
- 应用进程需处于运行状态，当应用退出后，将无法获得dumpsys命令结果。

2. SYS

【调试信息】

```
[root@RK356X:/userdata]# ./dumpsys sys
-----
DUMP OF SERVICE sys:
----- bind relation table -----
src_mod src_dev src_chn dst_mod dst_dev dst_chn src_recv_cnt dst_recv_cnt src_recv_rate dst_recv_rate
vdec 0 0 vo 0 0 29040 28968 25.17 25.19
vdec 0 1 vo 0 1 28981 28967 25.00 24.79
vdec 0 2 vo 0 2 28979 28968 25.18 24.99
vdec 0 3 vo 0 3 28976 28965 25.40 24.81
vdec 0 4 vo 0 4 28974 28962 24.95 25.02
vdec 0 5 vo 0 5 28974 28960 24.99 24.64
vdec 0 6 vo 0 6 28969 28958 24.83 25.18
vdec 0 7 vo 0 7 28966 28955 25.43 25.14
vdec 0 8 vo 0 8 28967 28953 24.58 24.84
vdec 0 9 vo 0 9 28964 28950 24.95 24.98
vdec 0 10 vo 0 10 28962 28948 24.79 25.20
vdec 0 11 vo 0 11 28960 28946 24.66 25.18
vdec 0 12 vo 0 12 28957 28943 24.96 24.98
vdec 0 13 vo 0 13 28956 28941 25.00 25.00
vdec 0 14 vo 0 14 28952 28938 25.00 24.64
vdec 0 15 vo 0 15 28951 28936 24.98 25.00
vdec 0 16 vo 2 0 28949 28935 25.21 25.19
vdec 0 17 vo 2 1 28946 28932 25.00 25.00
vdec 0 18 vo 2 2 28944 28930 24.99 24.82
vdec 0 19 vo 2 3 28941 28927 25.21 25.18
vdec 0 20 vpss 0 0 90396 90382 78.28 78.70
vpss 0 0 vo 2 4 90382 28921 78.70 25.18
vpss 0 1 vo 2 5 90382 28917 78.70 24.94
vdec 0 21 vpss 1 0 90380 90366 78.42 78.29
vpss 1 0 vo 2 6 90366 28916 78.29 25.04
vpss 1 1 vo 2 7 90366 28915 78.29 25.07
-----
END DUMP OF SERVICE sys:
```

【调试信息分析】

记录当前 SYS 模块的使用情况。

【参数说明】

参数模块	参数名	描述
bind relation table 模块通道间的绑定关系	src_mod	绑定关系中第一级的模块名，数据由第一级发送给第二级。
	src_dev	绑定关系中第一级的设备号，数据由第一级发送给第二级。
	src_chn	绑定关系中第一级的通道号，数据由第一级发送给第二级。
	dst_mod	绑定关系中第二级的模块名，数据由第一级发送给第二级。
	dst_dev	绑定关系中第二级的设备号，数据由第一级发送给第二级
	dst_chn	绑定关系中第二级的通道号，数据由第一级发送给第二级。
	src_recv_cnt	第一级接收到的数据计数（一般以帧为单位）。
	dst_recv_cnt	第一级向第二级的发送数据计数（一般以帧为单位）。
	src_recv_rate	第一级接收到的数据帧率。
	dst_recv_rate	第一级向第二级的发送数据帧率。

3. MB

【调试信息】

- 简要信息

```
[root@RK356X:/userdata]# dumpsys mb

-----  
DUMP OF SERVICE mb:  
-----  
----- mb total statistics -----  
mb_type total_size total_cnt  
DMA 51796992 18  
MALLOC 16 1  
  
----- mb module info -----  
mod chn_cnt total_size total_cnt  
cmpli 1 47004672 16  
vpss 1 4792320 2  
  
----- mb channel info -----  
owner mod dev_id chn_id total_size total_cnt unused_cnt  
private cmpli 0 0 47004672 16 0  
private vpss 0 0 4792320 2 2  
  
-----  
END DUMP OF SERVICE mb:
```

- 详细信息

```
[root@RK3588:/userdata]# dumpsys mb d

-----  
DUMP OF SERVICE mb:  
-----  
----- mb total statistics -----  
mb_type total_size total_cnt  
DMA 23596032 9  
MALLOC 16 1  
  
----- mb module info -----  
mod chn_cnt total_size total_cnt  
cmpli 1 18803712 7  
vpss 1 4792320 2  
  
----- mb channel info -----  
owner mod dev_id chn_id total_size total_cnt unused_cnt  
private cmpli 0 0 18803712 7 0  
private vpss 0 0 4792320 2 2  
  
----- mb detail info -----  
owner mod dev_id chn_id blk virt fd phy_addr size length refs  
other cmpli 0 0 0x27a4ffe0 0x7facb68000 13 0 3133440 3133440 1  
other cmpli 0 0 0x27a41b30 0x7fac56e000 16 0 3133440 3133440 1  
other cmpli 0 0 0x27a4f6e0 0x7fac271000 17 0 3133440 3133440 1  
other cmpli 0 0 0x27a9ab30 0x7f96500000 20 0 3133440 3133440 3  
other cmpli 0 0 0x27aa0320 0x7f96203000 21 0 3133440 3133440 3  
other cmpli 0 0 0x27aa06a0 0x7f95f06000 22 0 3133440 3133440 3  
other cmpli 0 0 0x27a9fc20 0x7fb2e6000 26 0 3072 3072 1  
other vpss 0 0 0x27a4c3c0 0x7face65000 14 0 1658880 1658880 0  
other vpss 0 0 0x27a44460 0x7fac86b000 15 0 3133440 3133440 0  
  
-----  
END DUMP OF SERVICE mb:
```

【调试信息分析】

记录当前MB模块的buffer 使用情况。

【参数说明】

参数模块	参数名	描述
mb total statistics mb总统计信息	mb_type	内存类型
	total_size	该类型总内存大小
	total_cnt	该类型总内存个数
mb module info 按模块划分统计信息	mod	模块类型
	chn_cnt	该模块通道数
	total_size	该模块总内存大小
	total_cnt	该模块总内存个数
mb channel info 按通道划分统计信息	owner	拥有者类型
	mod	模块类型。
	dev_id	设备ID。
	chn_id	通道ID
	total_size	该通道总内存大小
	total_cnt	该通道总内存个数
	unused_cnt	该通道总内存空闲个数
mb detail info mb详细统计信息	owner	拥有者类型
	mod	模块类型
	dev_id	设备ID
	chn_id	通道ID
	blk	BLK值
	virt	虚地址
	fd	句柄
	phy_addr	物理地址
	size	内存大小
	length	内存使用的长度
	refs	引用计数

【调试命令】

- **mb数据dump**

dump虚拟地址对应物理内存的数据

例：dump mb虚拟地址为0x7f8b41c000对应的物理地址的数据到tmp下，文件名为mb.bin。其中虚拟地址可以通过dumpsys mb d 的virt获取到。

```
./dumpsys mb viraddr 0x7f8b41c000 /tmp/mb.bin
```

4. VPSS

【调试信息】

```
[root@RK3588:/]# dumpsys vpss
-----
DUMP OF SERVICE vpss:
----- vpss mod param -----
mb_source
PRIVATE

----- vpss group attr -----
grp_id max_w max_h pixel_format dym_range src_rate dst_rate is_compress rotate vproc_dev max_queue
0 4096 4096 image:nv21 0 -1 -1 N 0 GPU 0

----- vpss channel attr1 -----
grp_id chn_id mode width height pixel_format is_compress src_rate dst_rate depth align mirror flip frm_cnt
0 0 USER 1920 1080 image:bgr565 Y -1 -1 1 16 N N 4
0 1 USER 1920 1080 image:bgr565 Y -1 -1 1 16 N N 4

----- vpss channel attr2 -----
grp_id chn_id rotate aspect video_x video_y video_w video_h bg_color
0 0 0 NONE 0 0 0 0 0x0
0 1 0 NONE 0 0 0 0 0x0

----- vpss group crop info -----
grp_id crop_en coor_type x y width height
0 N RATIO 0 0 0 0

----- vpss chn crop info -----
grp_id chn_id crop_en coor_type x y width height
0 0 N RATIO 0 0 0 0
0 1 N RATIO 0 0 0 0

----- vpss group pic queue -----
grp_id delay backup lft_in_cnt lft_out_cnt now_cst_us avg_cst_us max_cost_us run_cnt
0 0 N 0 0 0 0 0 0

----- vpss group pic info -----
grp_id width height vir_w vir_h pix_format dyn_range compress
0 1920 1080 1920 1080 image:nv12 4707540 Y

----- vpss chn work status -----
grp_id chn_id get_frm_cnt get_frm_rate rel_frm_cnt lft_in_cnt lft_out_cnt now_cst_us avg_cst_us max_cst_us run_cnt
0 0 234 0.00 234 0 0 816 902 5861 234
0 1 0 0.00 0 0 1 1640 1634 5729 167

-----
END DUMP OF SERVICE vpss:
```

【调试信息分析】

记录当前 VPSS 属性配置以及状态信息。

【参数说明】

参数模块	参数名	描述
vpss mod param VPSS模块属性	mb_source	VPSS通道内存缓冲来源。 COMMON：公共内存缓冲池。 PRIVATE：私有内存缓冲池。 USER：用户内存缓冲池。
vpss group attr VPSS模块GRP属性	grp_id	GRP ID 号。有效范围：[0,VPSS_MAX_GRP_NUM)。
	max_w	组输入图像最大宽度。
	max_h	组输入图像最大高度。
	pixel_format	组输入图像像素格式。
	dym_range	组输入图像的动态范围。
	src_rate	GRP源帧率。
	dst_rate	GRP目标帧率。
	is_compress	参考帧压缩模式。Y：压缩； N：非压缩。
	rotate	旋转的角度。有效值：0, 90, 180, 270。
	vproc_dev	当前所选设备名
vpss channel attr1 VPSS模块通道属性1	grp_id	GRP ID 号。有效范围：[0,VPSS_MAX_GRP_NUM)。
	chn_id	通道 ID 号。有效范围：[0,VPSS_MAX_CHN_NUM)。
	mode	通道模式。 USER: USER 模式； AUTO : AUTO模式。
	width	通道的目标输出宽度。
	height	通道的目标输出高度。
	pixel_format	通道的目标像素格式。
	is_compress	通道的目标图像的压缩模式。Y：压缩； N：非压缩。
	src_rate	通道帧率控制：源帧率。
	dst_rate	通道帧率控制：目标帧率。
	depth	用户获取通道图像的队列长度。
	align	通道输出 YUV 宽高对齐。
	mirror	是否使能 mirror 功能。 Y: 打开； N : 关闭。
	flip	是否使能 flip 功能。 Y: 打开； N : 关闭。
	frm_cnt	最大通道图像缓冲个数。
vpss channel attr2 VPSS模块通道属性2	grp_id	GRP ID 号。有效范围：[0,VPSS_MAX_GRP_NUM)。
	chn_id	通道 ID 号。有效范围：[0,VPSS_MAX_CHN_NUM)。
	rotate	旋转的角度。有效值：0, 90, 180, 270。

参数模块	参数名	描述
	aspect	幅型比模式。 NONE: 无幅型比 AUTO: 自动计算幅型比, 等比例放大至全屏 MANUAL: 手动计算幅型比, 根据设定值确定显示区域
	video_x	幅型比水平方向起始坐标。 仅在MUNUAL模式下生效。
	video_y	幅型比垂直方向起始坐标。 仅在MUNUAL模式下生效。
	video_w	幅型比宽度。 仅在MUNUAL模式下生效。
	video_h	幅型比高度。 仅在MUNUAL模式下生效。
	bg_color	非显示区域背景颜色。
vpss group crop info VPSS模块GRP CROP信息	grp_id	GRP ID 号。有效范围: [0,VPSS_MAX_GRP_NUM)。
	crop_en	是否使能 CROP 功能。 Y: 打开。 N : 关闭。
	coor_type	坐标类型。 RATIO: 相对坐标; ABS: 绝对坐标。
	x	水平方向起始坐标。 坐标类型为相对坐标时, 合法取值范围为[0, 999]; 坐标类型为绝对坐标时, 合法取值范围为[0, VPSS_MAX_IMAGE_WIDTH]。
	y	垂直方向起始坐标。 坐标类型为相对坐标时, 合法取值范围为[0, 999]; 坐标类型为绝对坐标时, 合法取值范围为[0, VPSS_MAX_IMAGE_WIDTH]。
	width	CROP RECT 的宽。 坐标类型为相对坐标时, 合法取值范围为(0, 1000]; 坐标类型为绝对坐标时, 合法取值范围为(0, VPSS_MAX_IMAGE_WIDTH]。
	height	CROP RECT 的高。 坐标类型为相对坐标时, 合法取值范围为(0, 1000]; 坐标类型为绝对坐标时, 合法取值范围为(0, VPSS_MAX_IMAGE_WIDTH]。
vpss chn crop info VPSS模块通道 CROP 信息	grp_id	GRP ID 号。有效范围: [0,VPSS_MAX_GRP_NUM)。
	crop_en	是否使能 CROP 功能。 Y: 打开。 N : 关闭。
	chn_id	通道 ID 号。有效范围: [0,VPSS_MAX_CHN_NUM)。
	coor_type	坐标类型。 RATIO: 相对坐标; ABS: 绝对坐标。

参数模块	参数名	描述
	x	水平方向起始坐标。 坐标类型为相对坐标时，合法取值范围为[0, 999]； 坐标类型为绝对坐标时，合法取值范围为[0, VPSS_MAX_IMAGE_WIDTH]。
	y	垂直方向起始坐标。 坐标类型为相对坐标时，合法取值范围为[0, 999]； 坐标类型为绝对坐标时，合法取值范围为[0, VPSS_MAX_IMAGE_WIDTH]。
	width	CROP RECT 的宽。 坐标类型为相对坐标时，合法取值范围为(0, 1000]； 坐标类型为绝对坐标时，合法取值范围为(0, VPSS_MAX_IMAGE_WIDTH]。
	height	CROP RECT 的高。 坐标类型为相对坐标时，合法取值范围为(0, 1000]； 坐标类型为绝对坐标时，合法取值范围为(0, VPSS_MAX_IMAGE_WIDTH]。
vpss group pic queue VPSS模块GRP缓存图像队列状态信息	grp_id	GRP ID 号。有效范围：[0,VPSS_MAX_GRP_NUM)。
	delay	Delay 队列长度。
	backup	backup 帧使能。 Y: 打开； N: 关闭。
	lft_in_cnt	待处理的组输入缓存帧数。
	lft_out_cnt	待取走的组输出缓存帧数。
	now_cst_us	最近一帧图像处理耗时。
	avg_cst_us	最近10帧的平均处理耗时。
	max_cst_us	最大图像处理耗时。
	run_cnt	图像处理次数。
vpss group pic info 当前处理输入图像信息。	grp_id	GRP ID 号。有效范围：[0,VPSS_MAX_GRP_NUM)。
	width	输入图像实际宽度。
	height	输入图像实际高度。
	vir_w	输入图像的虚宽。
	vir_h	输入图像的虚高。
	pix_format	输入图像实际像素格式。
	dyn_range	输入图像实际动态范围。
	compress	输入图像压缩模式： Y: 压缩； N: 非压缩。
vpss chn work status VPSS 通道工作状态	grp_id	GRP ID 号。有效范围：[0,VPSS_MAX_GRP_NUM)。
	chn_id	通道 ID 号。有效范围：[0,VPSS_MAX_CHN_NUM)。
	get_frm_cnt	用户通过RK_MPI_VPSS_GetChnFrame取走的帧数。

参数模块	参数名	描述
	get_frm_rate	用户通过RK_MPI_VPSS_GetChnFrame取走的帧率。
	rel_frm_cnt	用户通过RK_MPI_VPSS_ReleaseChnFrame还回的帧数。
	lft_in_cnt	待处理的通道输入缓存帧数。
	lft_out_cnt	待取走的通道输出缓存帧数。
	now_cst_us	通道最近一帧的处理耗时。
	avg_cst_us	通道最近10帧的平均处理耗时。
	max_cst_us	通道最大的图像处理耗时。
	run_cnt	通道处理次数。
vpss aiisp attr VPSS AI ISP 模块属性	grp_id	GRP ID 号。有效范围: [0, VPSS_MAX_GRP_NUM)。
	enable	是否使能 AI ISP 功能。 Y: 打开。 N : 关闭。
	callback	AI ISP 回调函数。
	data	AI ISP 回调函数私有变量。
	module_path	AI ISP 模型路径。
	frm_cnt	最大图像缓冲个数。
vpss aiisp status VPSS AI ISP 模块工作状态	grp_id	GRP ID 号。有效范围: [0, VPSS_MAX_GRP_NUM)。
	lft_in_cnt	待处理的输入缓存帧数。
	lft_out_cnt	待取走的输出缓存帧数。
	now_cst_us	当前图像的耗时, 单位us。
	avg_cst_us	图像处理的平均耗时, 单位us。
	max_cst_us	图像处理的最大耗时, 单位us。
	run_cnt	处理次数。
	now_itv_us	当前图像处理的间隔时间, 单位us。
	avg_itv_us	图像处理的平均间隔时间, 单位us。
	max_itv_us	图像处理的最大间隔时间, 单位us。

【注意】

- VPSS 通道旋转信息显示旋转信息, 需要满足:
 - 成功调用 RK_MPI_VPSS_SetChnRotation、RK_MPI_VPSS_SetChnRotationEx;
 - 配置非0角度旋转。
- VPSS AI ISP 模块工作状态将显示 AI ISP 内部处理流程中的耗时、次数以及间隔时间。

5. RGN

【调试信息】

```
[root@RK356X:/userdata/test]# ./dumpsys rgn
```

```

-----
DUMP OF SERVICE rgn:
----- region status of overlay -----
hdl type used pixel_format width height mb virt clut_num

----- region chn status of overlay -----
hdl type mod dev chn is_show x y fg_alpha bg_alpha layer

----- region status of cover -----
hdl type used
0 1 N
1 1 N
2 1 N
3 1 N

----- 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
1 1 venc 0 0 true 64 64 256 256 0xf800 1 ABS
2 1 venc 0 0 true 128 128 256 256 0xf800 1 ABS
3 1 venc 0 0 true 192 192 256 256 0xf800 1 ABS

----- region status of mosaic -----
hdl type used

----- region chn status of mosaic -----
hdl type mod dev chn is_show x y width height blk_size layer

point_0 point_1 point_2 point_3
0 2 vpss 0 0 true (256 ,128 )(384 ,256 )(128 ,256 )(256 ,384 )8 0
0 2 vpss 0 1 true (256 ,128 )(384 ,256 )(128 ,256 )(256 ,384 )8 0
0 2 vpss 0 2 true (256 ,128 )(384 ,256 )(128 ,256 )(256 ,384 )8 0
0 2 vpss 0 3 true (256 ,128 )(384 ,256 )(128 ,256 )(256 ,384 )8 0
----- END DUMP OF SERVICE rgn:

```

【调试信息分析】

记录当前区域资源信息。

【参数说明】

参数模块	参数名	描述
region status of overlay overlay的状态信息	hdl	RGN的 Handle 号。
	type	OVERLAY 类型, 值为 0。
	used	该资源是否被占用。 N: 未占用; Y: 占用。
	pixel_format	OVERLAY 像素格式, 参看PIXEL_FORMAT_E。
	width	OVERLAY 区域宽度。
	height	OVERLAY 区域高度。
	mb	OVERLAY画布的内存MB ID。
	virt	OVERLAY画布的内存虚拟地址。
	clut_num	OVERLAY画板颜色个数。合法取值范围为[0, 255]。 0: 使用内部默认画板; 其他: 使用用户自定义画板, 并且颜色个数为clut_num。
region chn status of overlay OVERLAY在RGN通道中的显示状态	hdl	RGN的 Handle 号。
	type	OVERLAY 类型, 值为 0。
	mod	Attach 的模块。
	dev	设备号。
	chn	通道号。
	is_show	是否在该通道显示。 N : 隐藏。 Y : 显示。
	x	在该通道显示的起始 X 坐标。
	y	在该通道显示的起始 Y 坐标。
	fg_alpha	在该通道显示的前景 alpha。
	bg_alpha	在该通道显示的背景 alpha。
	layer	在该通道显示的层次。
region status of cover cover的状态信息	hdl	RGN的 Handle 号。
	type	COVER 类型, 值为 1。
	used	该资源是否被占用。 N: 未占用; Y: 占用。
region chn status of cover COVER在RGN通道中的显示状态	hdl	RGN的 Handle 号。
	type	COVER 类型, 值为 1。
	mod	Attach 的模块。
	dev	设备号。
	chn	通道号。

参数模块	参数名	描述
	is_show	是否在该通道显示。 N : 隐藏。 Y : 显示。
	x	COVER区域水平方向起始坐标。 坐标类型为相对坐标时，合法取值范围为[0, 999]；
	y	COVER区域垂直方向起始坐标。 坐标类型为相对坐标时，合法取值范围为[0, 999]；
	width	COVER区域的宽度。 坐标类型为相对坐标时，合法取值范围为(0, 1000]；
	height	COVER区域的高度。 坐标类型为相对坐标时，合法取值范围为(0, 1000]；
	color	COVER 颜色。
	layer	在该通道显示的层次。
	coord_type	坐标类型。 RATIO：相对坐标； ABS：绝对坐标。
region status of mosaic mosaic的状态信息	hdl	MOSAIC的 Handle 号。
	type	MOSAIC 类型。值为2。
	used	该资源是否被占用。 N: 未占用； Y: 占用。
region chn status of mosaic MOSAIC在通道中的显示状态	hdl	MOSAIC的 Handle 号。
	type	MOSAIC 类型。值为2。
	mod	Attach 的模块。
	dev	设备号。
	chn	通道号。
	is_show	是否在该通道显示。 N : 隐藏。 Y : 显示。
	x	MOSAIC区域水平方向起始坐标。
	y	MOSAIC 区域垂直方向起始坐标。
	width	MOSAIC区域的高度。
	height	MOSAIC区域的虚宽。
	piont_0	MOSAIC区域顶点0 直角坐标系位置。
	piont_1	MOSAIC区域顶点1 直角坐标系位置。
	piont_2	MOSAIC区域顶点2 直角坐标系位置。
	piont_3	MOSAIC区域顶点3 直角坐标系位置。
	blk_size	MOSAIC 精度。取值范围{8, 16, 32}
	layer	在该通道显示的层次。

6. VGS

【调试信息】

```
dumpsys vgs
-----
DUMP OF SERVICE vgs:
----- module params -----
g_max_job_num   g_max_task_num
128           200
----- recent job info1 -----
seq_no  job_hdl state  task_num in_size out_size cost_time hw_time
0      proced  1     345600 353280 1553    0
----- recent job info2 -----
seq_no  scale  cover  mosaic osd   line   rotate  crop
0      1     0     0     0     0     0     0
----- recent job info3 -----
job_seq task_seq srcw  srch  src_vir_w src_vir_h srcformat  src_rect_x src_rect_x src_rect_w src_rect_h
0      0     720   480   720   480   image:nv12 0     0     720   480
job_seq task_seq dstw  dsth  dst_vir_w dst_vir_h dstformat  dst_rect_x dst_rect_x dst_rect_w dst_rect_h
0      0     720   480   736   480   image:bgra5551 0     0     720   480
----- max waste time recent job info1 -----
seq_no  job_hdl state  task_num in_size out_size cost_time hw_time
0      proced  1     345600 353280 1553    0
----- max waste time recent job info2 -----
seq_no  scale  cover  mosaic osd   line   rotate  crop
0      1     0     0     0     0     0     0
----- max waste time recent job info3 -----
job_seq task_seq srcw  srch  src_vir_w src_vir_h srcformat  src_rect_x src_rect_x src_rect_w src_rect_h
0      0     720   480   720   480   image:nv12 0     0     720   480
job_seq task_seq dstw  dsth  dst_vir_w dst_vir_h dstformat  dst_rect_x dst_rect_x dst_rect_w dst_rect_h
0      0     720   480   736   480   image:bgra5551 0     0     720   480
----- vgs job status -----
success fail  cancel all_job_num  free_num begin_num busy_num procing_num
1      0     0     128     128   0     0     0
----- vgs task status -----
success fail  cancel all_task_num  free_num busy_num
1      0     0     200     199   1
-----
END DUMP OF SERVICE vgs:
```

【调试信息分析】

记录VGS模块最近完成的若干任务、最近耗时最大的认证、历史累计信息等。

【参数说明】

参数		描述
module para	g_max_job_num	最大的job数
	g_max_task_num	最大的task数
recent job info1 最近完成的job的信息	seq_no	打印序号 取值范围:[0,7]
	job_hdl	该job的handle号
	task_num	该job包含的task数目
	state	该job的处理状态
	in_size	该job下各task的输入图像面积之和，单位像素
	out_size	该job下各task的输出图像面积之和，单位像素
	cost_time	该job从提交 (end_job) 开始到成功完成的耗时时长。单位us
	hw_time	该job在硬件中处理的耗时时长。单位us
recent job info2 最近完成的job的信息	crop	CROP使能 (0: 关闭, 1: 打开)
	cover	Cover使能 (0: 关闭, 1: 打开)
	mosaic	Mosaic使能 (0: 关闭, 1: 打开)
	osd	OSD使能 (0: 关闭, 1: 打开)
	line	Line使能 (0: 关闭, 1: 打开)
	rotate	旋转使能 (0: 关闭, 1: 打开)
recent job info3 最近完成的job 中添加的task输入和输出图像信息	job_seq	job打印序号 取值范围:[0,7]
	task_seq	该job的添加task的序号
	srcw	task输入图像的实宽
	srch	task输入图像的实高
	src_vir_w	task输入图像的虚宽
	src_vir_h	task输入图像的虚高
	srcformat	task输入图像的format
	src_rect_x	task输入图像操作区域起点的x坐标
	src_rect_y	task输入图像操作区域起点的y坐标
	src_rect_w	task输入图像操作区域的宽
	src_rect_h	task输入图像操作区域的高
	dstw	task输出图像的实宽
	dsth	task输出图像的实高
	dst_vir_w	task输出图像的虚宽
	dst_vir_h	task输出图像的虚高
	dstformat	task输出图像的format
	dst_rect_x	task输出图像操作区域起点的x坐标

参数		描述
	dst_rect_x	task输出图像操作区域起点的y坐标
	dst_rect_w	task输出图像操作区域的宽
	dst_rect_h	task输出图像操作区域的高
max waste time job info 最近耗时最大的job的信息	各个成员同recent job info1的成员	最近500个任务中耗时最长的job信息，当任务数超过500的时候，会重置最大值。
max waste time job info2 最近耗时最大的job的信息	各个成员同recent job info2的成员	最近500个任务中耗时最长的job信息，当任务数超过500的时候，会重置最大值。
max waste time job info3 最近耗时最大的job的信息	各个成员同recent job info3的成员	最近500个任务中耗时最长的job信息，当任务数超过500的时候，会重置最大值。
vgs job status VGS任务状态	success	累计成功处理的job数
	fail	累计处理失败的job数
	cancel	累计主动取消的job数
	all_job_num	VGS所有可用的任务数
	free_num	空闲的job数
	begin_num	用户已经创建但是未提交的job数量
	busy_num	用户已经提交但是未提交给硬件处理的任务数
	procing_num	正在进行硬件处理的任务数
vgs task status VGS task状态	success	累计成功处理的task数量，如果一个job处理成功，那么job中包含的task也全部处理成功。Task处理成功的数量累加上成功job中的task数。
	fail	累计处理失败的task数量，如果一个job处理失败，那么job中包含的task也全部处理失败，Task处理失败的数量累加上失败job的task数。
	cancel	累计处理cancel的task数量，如果一个jobcancel，那么job中包含的task也全部cancel，Task处理cancel的数量累加上canceljob的task数。
	all_task_num	VGS Task的总数，一般为200。
	free_num	空闲的task数
	busy_num	已添加到job下的task数

【调试命令】

- **数据录制**

录制vgs job中task处理的输入和输出数据

例：启动录制，生成的数据存放在tmp路径下，最多录制10帧

```
./dumpsys vgs record /tmp/ 10
```

7. ADEC

【调试信息】

```
dumps sys adep  
-----  
DUMP OF SERVICE adep:  
----- adep chn attr -----  
chn_id codec_id buf_cnt mode rate channel orig_send_cnt send_cnt get_cnt put_cnt  
0 mp3 4 packet 44100 2 3320 3316 3315 3315  
-----
```

【调试信息分析】

记录当前音频解码属性配置以及状态信息。

【参数说明】

参数模块	参数名	描述
音频解码通道属性及状态	chn_id	通道号
	codec_id	解码协议类型
	buf_cnt	帧缓存数目
	mode	按流解码还是按帧解码
	rate	码流采样率
	channel	码流声道数
	orig_send_cnt	前端发送到解码器进行解码的码流帧数目
	send_cnt	成功发送解码器进行解码的音频帧数目
	get_cnt	用户获取的音频帧数目
	put_cnt	用户释放的音频帧数目

8. AENC

【调试信息】

```
dumps sys aenc  
-----  
DUMP OF SERVICE aenc:  
----- aenc chn attr -----  
chn_id codec_id buf_cnt rate channel bit_with  
0 flac 4 44100 2 16bit  
----- aenc chn status -----  
chn_id recv_frame enc_ok frame_err get_stream release_stream  
0 1248 69 0 69 69  
-----  
END DUMP OF SERVICE aenc:
```

【调试信息分析】

记录当前音频编码属性配置以及状态信息。

【参数说明】

参数模块	参数名	描述
音频编码通道属性及状态	chn_id	通道号
	codec_id	编码协议类型
	buf_cnt	帧缓存数目
	rate	音频PCM帧数据采样率
	channel	音频PCM帧数据声道数
	bit_with	音频PCM帧数据采样精度
音频编码通道状态	chn_id	通道号
	recv_frame	编码协议类型
	enc_ok	成功编码的音频帧数目
	frame_err	编码失败的音频帧数
	get_stream	用户获取音频码流的次数
	release_stream	用户释放音频码流的次数

9. AO

【调试信息】

```
dumpsys ao

DUMP OF SERVICE ao:
----- ao dev attr -----
ao_dev snd_rate snd_channel snd_bit_Width data_rate data_channel data_bit_width chn_cnt expand_flag frm_num
frm_size
0 48000 2 16bit 44100 stereo 16bit 2 0 4 1024
----- ao dev extend status -----
ao_dev track_mode mute volume
0 0 N 100
----- ao chn attr -----
ao_dev ao_chn card_name snd_open state resample_open in_rate in_ch out_rate out_ch
0 0 hw:0,0 open start Y 44100 2 48000 2
----- ao chn status -----
ao_dev ao_chn frm_len frm_total_cnt frm_total_len
0 0 1024 511 523264
----- ao node status -----
ao_dev ao_chn node in_size in_cnt proc_cnt write_size
0 0 0 2048 2 0 0
0 0 1 0 0 0 0
0 0 2 0 0 0 0
0 0 3 4460 4 508 565012
END DUMP OF SERVICE ao:
```

【调试信息分析】

记录当前AO属性配置以及状态信息。

【参数说明】

参数模块	参数名	描述
音频输出通道属性	ao_dev	设备号
	snd_rate	打开声卡的采样率
	snd_channel	打开声卡的声道数
	snd_bit_Width	打开声卡的比特位
	data_rate	发送数据的采样率 范围: [8k,96k]
	data_channel	发送数据的声道数 mono: 单声道 stereo: 双声道
	data_bit_width	发送数据的采样精度 范围: [8bit,16bit]
	chn_cnt	暂未使用, 可设为默认值2
	expand_flag	暂未使用, 可设为默认值0
	frm_num	设置送帧最大缓冲buffer数
	frm_size	设置送帧buffer大小
音频输出设备扩展信息	ao_dev	设备号
	track_mode	声道模式 0: normal 1: both_left 2: both_right 3: exchange 4: mix 5: left_mute 6: right_mute 5: both_mute
	mute	静音功能是否开启 Y: 开启 N: 关闭
	volume	音量值(0-100)
音频输出通道属性	ao_dev	设备号
	ao_chn	通道号
	card_name	声卡名
	snd_open	声卡是否打开: close:关闭 open:打开
	state	通道状态 idle: 闲置状态 pause: 暂停状态 start: 工作状态
	resample_open	重采样是否开启 Y: 开启 N: 关闭
	in_rate	重采样的源采样率
	in_ch	重采样的源声道数

参数模块	参数名	描述
	out_rate	重采样的目标采样率
	out_ch	重采样的目标声道数
音频输出通道信息	ao_dev	设备号
	ao_chn	通道号
	frm_len	ao送帧当前帧的长度
	frm_total_cnt	ao送帧累计帧数
	frm_total_len	ao送帧累计长度
音频输出通道节点信息	ao_dev	设备号
	ao_chn	通道号
	node	节点号
	in_size	当前节点inputBuffer缓冲长度
	in_cnt	当前节点inputBuffer缓冲个数
	proc_cnt	process函数线程累计计数 (只统计plackback节点)
	write_size	写入声卡的buffer长度 (只统计plackback节点)

10. AI

【调试信息】

```
dumpsys ai

DUMP OF SERVICE ai:
----- ai dev attr -----
ai_dev snd_rate snd_channel snd_bit_Width data_rate data_channel data_bit_width chn_cnt expand_flag frm_num
frm_size
0 48000 2 16bit 16000 stereo 16bit 2 0 4 1024
----- ai dev extend status -----
ai_dev track_mode
0 0
----- ai chn attr -----
ai_dev ai_chn card_name snd_open state resample_open in_rate in_ch out_rate out_ch
0 0 hw:1,0 open start Y 48000 2 16000 2
----- ai chn status -----
ai_dev ai_chn frm_len frm_total_cnt frm_total_len
0 0 1364 682 133250
----- ai node status -----
ai_dev ai_chn node in_size in_cnt
0 0 0 0 0
0 0 1 4096 1
0 0 2 0 0
END DUMP OF SERVICE ai:
```

【调试信息分析】

记录当前AI属性配置以及状态信息。

【参数说明】

参数模块	参数名	描述
音频输入设备属性	ai_dev	设备号
	snd_rate	打开声卡的采样率
	snd_channel	打开声卡的声道数
	snd_bit_Width	打开声卡的比特位
	data_rate	获取数据的采样率
	data_channel	获取数据的声道数 mono: 单声道 stereo: 双声道
	data_bit_width	获取数据的采样精度 范围: [8bit,16bit]
	chn_cnt	暂未使用, 可设为默认值2
	expand_flag	暂未使用, 可设为默认值0
	frm_num	ai暂未使用, 可设为默认值4
	frm_size	应用设置取帧的长度 (byte) 。
音频输入设备扩展信息	ai_dev	设备号
	track_mode	声道模式 0: normal 1: both_left 2: both_right 3: exchange 4: mix 5: left_mute 6: right_mute 5: both_mute
音频输入通道属性	ai_dev	设备号
	ai_chn	通道号
	card_name	声卡名
	snd_open	声卡是否打开: close:关闭 open:打开
	state	通道状态 idle: 闲置状态 pause: 暂停状态 start: 工作状态
	resample_open	重采样是否开启 Y: 开启 N: 关闭
	in_rate	重采样的源采样率
	in_ch	重采样的源声道数
	out_rate	重采样的目标采样率
	out_ch	重采样的目标声道数
音频输入通道信息	ai_dev	设备号
	ai_chn	通道号

参数模块	参数名	描述
	frm_len	ai取帧当前帧的长度
	frm_total_cnt	ai取帧累计帧数
音频输入通道节点信息	ai_dev	设备号
	ai_chn	通道号
	node	节点号
	in_size	当前节点inputBuffer缓冲长度
	in_cnt	当前节点inputBuffer缓冲个数

11. VI

【调试信息】

```
[root@RK356X:/]# dumpsys vi
-----
DUMP OF SERVICE vi:
-----
vi module_param -----
vi_max_chn_num
4

----- vi chn attr -----
pipe chn width height max_width max_height compress_mode
0 0 1920 1080 0 0 0
memory_type buf_type pix_format buf_count buf_size entity_name
4 0 image:nv12 3 3133440 rkispp_scale0

----- vi chn query state -----
pipe chn width height enabled input_lost output_lost frame_id framerate vbfail freeze
0 0 1920 1080 1 0 1 233 25 0 0

----- vi chn get buf state -----
pipe chn seq pts delay_us poll_cnt poll_fail_cnt get_cnt release_cnt
0 0 233 4835459158 52942 233 0 233 233
qbuf_fail_cnt commit_cnt refs
0 3 11100000

----- vi chn user picture attr -----
pipe chn enable mode bg_color frame width height pix_format refs seq pts
0 0 Y bgc 0x80 0x7f4023d2b0 1920 1080 image:nv12 3 2378 225580325

----- vi chn connect info -----
pipe chn width height pix_format connect
0 0 1920 1080 image:nv12 unknown

----- vi chn edid info -----
pipe chn max_block block_id pad edid_crc
0 1 2 0 0 a3
0 1 2 1 0 b5

----- vi chn stream info -----
pipe chn codec_type
0 1 image:h265

----- vi chn dis config -----
pipe chn mode motion pdt_type buf_num crop_ratio frm_rate gyro_range gyro_bit_width cam_steady
0 1 0 1 0 10 80 25 0 0 N

----- vi chn dis attr -----
```

```
pipe chn dis_chn enable mov_sub roef timelag view_angle still_crop hor_limit ver_limit
0 1 1 Y 0 0 40000 1000 N 512 512
```

```
-----  
END DUMP OF SERVICE vi:
```

【调试信息分析】

记录当前VI属性配置及状态信息。

【参数说明】

参数模块	参数名	描述
视频输入设备模块参数	vi_max_chn_num	视频输入设备支持最大通道数
视频输入通道参数	pipe	管道号
	chn	通道号
	width	通道图像宽度
	height	通道图像高度
	max_width	通道图像最大宽度（暂时不用）
	max_height	通道图像最大高度（暂时不用）
	compress_mode	通道图像压缩模式:0-不压缩；1-afbc
	memory_type	通道图像内存类型：0-mmap;1-userptr;2-overlay;3-dma
	buf_type	通道图像内存分配：0-内部；1-外部
	pix_format	通道图像像素格式
	buf_count	通道图像内存分配个数
	buf_size	通道图像单个内存大小
	entity_name	通道设备名字
视频输入通道状态	pipe	管道号
	chn	通道号
	width	通道图像宽度
	height	通道图像高度
	enabled	通道使能状态
	input_lost	通道输入丢帧数
	output_lost	通道输出丢帧数
	frame_id	通道输出图像的帧号
	framerate	通道帧率
	vbfail	通道获取失败次数
	freeze	通道视频输出冻结使能：0-不使能；1-使能
视频输入buffer状态	pipe	管道号
	chn	通道号
	seq	通道当前buffer的序列号seq
	pts	通道当前buffer的pts
	delay_us	通道当前buffer的延时，单位us
	poll_cnt	通道查询buffer的次数
	poll_fail_cnt	通道查询buffer失败的次数，数值一直增加表示获取不到buffer，可能是驱动已经异常
	get_cnt	通道buffer获取次数
	release_cnt	通道buffer释放次数

参数模块	参数名	描述
	qbuf_fail_cnt	通道返还buffer失败次数
	commit_cnt	通道buffer申请个数
	refs	通道buffer对应引用次数， 默认最大支持显示8个buffer的引用次数
视频输入图片插入信息	pipe	管道号
	chn	通道号
	enable	图片插入功能使能：Y-使能；N-不使能
	mode	图片插入功能模式：none-关闭；pic：本地图片插入；bgc：背景色插入
	bg_color	背景色颜色值（背景色插入时有效）
	frame	图片插入帧指针地址
	width	图片插入帧宽度
	height	图片插入帧高度
	pix_format	图片插入帧格式
	refs	图片插入帧引用次数
	seq	图片插入帧序列号seq
	pts	图片插入帧pts
通道连接信息	pipe	管道号
	chn	通道号
	width	通道接入图像宽度
	height	通道接入图像高度
	pix_format	通道接入图像格式
	connect	通道接入状态（hdmi rx） unknown：未知； connect：连接； disconnect：未连接
通道edid信息	pipe	管道号
	chn	通道号
	max_block	最大block个数
	block_id	当前block id
	pad	pad值
	edid_crc	当前block edid crc值
通道codec信息	pipe	管道号
	chn	通道号
	codec_type	设置的codec类型
动态防抖配置信息	pipe	管道号。
	chn	通道号。
	mode	动态防抖中使用不同自由度防抖算法。
	motion	摄像头的运动级别。
	pdt_type	使用动态防抖的产品形态。

参数模块	参数名	描述
	buf_num	动态防抖用于缓存图像的buffer 数量。
	crop_ratio	动态防抖输出图像的裁剪比例。
	frm_rate	vi 输出的实时帧率。
	gyro_range	陀螺仪输出的角度范围。
	gyro_bit_width	陀螺仪数据的位宽。
	cam_steady	镜头是否固定静止的。
动态防抖属性信息	pipe	管道号。
	chn	通道号。
	dis_chn	绑定动态防抖的通道号。
	enable	动态防抖是否使能。
	mov_sub	用于判断物体运动的级别。
	roef	rolling shutter 现象的矫正参数。
	time_lag	帧起始时间和陀螺仪数据采集时间的时间差。
	view_angle	镜头水平视场角参数。无效参数，未使用。
	still_crop	是否关闭动态防抖效果，但图像依旧保持裁剪比例输出。
	hor_limit	水平偏移限制。
	ver_limit	垂直偏移限制。
	strength	陀螺仪防抖的强度，仅对GYRO 模式有效。

12. VO

【调试信息】

```
[root@RK356X:/]# dumpsys vo
-----
DUMP OF SERVICE vo:
-----DEV CONFIG-----
DevId DevEn InfType InfSync Vcnt
0 N Unkown Unkown 0
1 Y EDP 1024x768p60 0
2 N Unkown Unkown 0
-----LAYER BIND CONFIG-----
DevId Video Gfx Cursor
0 clu0 esm0 sm0
1 clu1 esm1 sm1
2 unkown unkown unkown
-----LAYER STATUS-----
LayId LayEn PixFmt ImgW ImgH DispX DispY DispW DispH FrmRt BufLens Comprs ByPass VoDev
0 N NV12 0 0 0 0 0 0 0 0 N N -1
1 N NV12 0 0 0 0 0 0 0 0 N N -1
2 Y BGR24 1024 768 0 0 1024 768 25 3 Y N -1
3 N NV12 0 0 0 0 0 0 0 0 N N -1
4 N NV12 0 0 0 0 0 0 0 0 N N -1
5 Y BGRA32 1024 768 0 0 1024 768 25 3 N N -1
6 N NV12 0 0 0 0 0 0 0 0 N N -1
7 N NV12 0 0 0 0 0 0 0 0 N N -1
8 Y BGR24 1920 1080 0 0 1920 1080 25 4 N N 0
```

```
9  Y  BGR24 1920 1080 0 0 1920 1080 25 4 N N 0
10 Y  BGR24 1920 1080 0 0 1920 1080 25 4 N N 0
11 Y  BGR24 1920 1080 0 0 1920 1080 25 4 N N 0
LAYER clu1 CHANNEL STATUS:
ChnId Prio X Y W H ChnFrt FgAlpha BgAlpha KeyEn Color Show GDC Pause Step Cache RevThr RevCnt
0 0 960 540 960 540 0 128 0 N 0 Y Y N N 4 2 34
127 127 0 0 960 540 0 255 0 N 0 Y N N N 1 2 1
LAYER esm1 CHANNEL STATUS:
ChnId Prio X Y W H ChnFrt FgAlpha BgAlpha Show Pause Step Cache RevThr RevCnt
0 0 0 0 1024 768 25 128 0 Y N N 1 3 83
1 1 0 0 1024 768 25 128 0 Y N N 1 3 83
WBC STATUS:
WbcId En Src W H Fmt Comprs Frt Depth SendCnt
0 N Dev0 0 0 NV12 N 0 0 0
-----
```

【调试信息分析】

记录当前VO属性配置及状态信息。

【参数说明】

参数模块	参数名	描述
显示输出设备状态	DevId	显示输出设备号
	DevEn	显示输出设备使能状态：Y-开启； N-关闭
	InfType	显示输出接口类型
	InfSync	显示输出接口时序
	Vcnt	显示输出设备行中断
图层绑定关系	DevId	显示输出设备号
	Video	视频层名称
	Gfx	图形层名称
	Cursor	鼠标层名称
图层状态	LayId	图层设备号
	LayEn	图层使能状态：Y-开启； N-关闭
	PixFmt	图层像素格式
	ImgW	图层画布宽度
	ImgH	图层画布高度
	DispX	图层显示区域左上角X坐标
	DispY	图形显示区域左上角Y坐标
	DispW	图层显示区域宽度
	DispH	图层显示区域高度
	FrmRt	图层刷新帧率
图层通道状态	BufLens	图层画布缓存个数
	Comprs	图层画布压缩状态：Y-压缩； N-非压缩
	ByPass	图层是否启用直通模式：Y-开启； N-关闭
	VoDev	图层绑定的显示输出设备ID： -1表示未绑定
	ChnId	通道设备号
图层通道状态	Prio	通道优先级
	X	通道显示区域左上角X坐标
	Y	通道显示区域左上角Y坐标
	W	通道显示区域宽度
	H	通道显示区域高度
	ChnFrt	通道显示帧率
	FgAlpha	通道数据为RGBA5551格式时有效，对应A=1 Alpha值
	BgAlpha	通道数据为RGBA5551格式时有效，对应A=0 Alpha值
	KeyEn	通道是否使能关键色：Y-使能； N-禁用
	Color	通道关键色值，16进制
图层通道状态	Show	通道是否可见：Y-可见； N-隐藏
	GDC	通道是否开启畸变矫正：Y-开启； N-关闭

参数模块	参数名	描述
	Pause	通道播放状态：Y-暂停； N-播放
	Step	通道是否单帧步进：Y-使能； N-关闭
	Cache	通道缓存帧数
	RevThr	通道视频输出的显示门限值
	RevCnt	通道接收过的帧数，禁用/使能通道会清零
回写状态	WbclId	回写设备号
	En	回写使能状态：Y-开启； N-关闭
	Src	回写数据源名称
	W	回写数据宽度
	H	回写数据高度
	Fmt	回写数据格式
	FrT	回写数据帧率
	Depth	回写数据缓存帧数
	SendCnt	回写发送帧数

13. TDE

【调试信息】

```
dumpsys tde
-----
DUMP OF SERVICE tde:
----- module params -----
g_max_job_num   g_max_task_num
128           200
----- recent job info1 -----
seq_no  job_hdl state  task_num in_size out_size cost_time hw_time
0      proced  1    76800  518400 1160     0
----- recent job info2 -----
seq_no  copy   fill   resize bitblit rotate
0      1      0      0      0      0
----- recent job info3 -----
job_seq task_seq srcw   srch   src_vir_w src_vir_h srcformat   src_rect_x src_rect_x src_rect_w src_rect_h
0      0      320    240    320    240  image:nv12 0      0      320    240
job_seq task_seq dstw   dsth   dst_vir_w dst_vir_h dstformat   dst_rect_x dst_rect_x dst_rect_w dst_rect_h
0      0      320    240    960    540  image:nv12 56     0      320    240
----- max waste time recent job info1 -----
seq_no  job_hdl state  task_num in_size out_size cost_time hw_time
0      proced  1    76800  518400 1160     0
----- max waste time recent job info2 -----
seq_no  copy   fill   resize bitblit rotate
0      1      0      0      0      0
----- max waste time recent job info3 -----
job_seq task_seq srcw   srch   src_vir_w src_vir_h srcformat   src_rect_x src_rect_x src_rect_w src_rect_h
0      0      320    240    320    240  image:nv12 0      0      320    240
job_seq task_seq dstw   dsth   dst_vir_w dst_vir_h dstformat   dst_rect_x dst_rect_x dst_rect_w dst_rect_h
0      0      320    240    960    540  image:nv12 56     0      320    240
----- tde job status -----
success fail  cancel all_job_num free_num begin_num busy_num procing_num
1      0      0      128     128     0      0      0

```

```
----- tde task status -----
success fail cancel all_task_num free_num busy_num
1     0     0     200     199     1
```

```
-----  
END DUMP OF SERVICE tde:
```

【调试信息分析】

记录TDE模块最近完成的若干任务、最近耗时最大的认证、历史累计信息等。

【参数说明】

参数		描述
module para	g_max_job_num	最大的job数
	g_max_task_num	最大的task数
recent job info1 最近完成的job的信息	seq_no	打印序号 取值范围:[0,7]
	job_hdl	该job的handle号
	task_num	该job包含的task数目
	state	该job的处理状态
	in_size	该job下各task的输入图像面积之和，单位像素
	out_size	该job下各task的输出图像面积之和，单位像素
	cost_time	该job从提交 (end_job) 开始到成功完成的耗时时长。单位us
	hw_time	该job在硬件中处理的耗时时长。单位us
recent job info2 最近完成的job的信息	copy	COPY使能 (0: 关闭, 1: 打开)
	fill	FILL使能 (0: 关闭, 1: 打开)
	resize	Resize使能 (0: 关闭, 1: 打开)
	bitblit	BITBLIT使能 (0: 关闭, 1: 打开)
	rotate	旋转使能 (0: 关闭, 1: 打开)
recent job info3 最近完成的job 中添加的task输入和输出图像信息	job_seq	job打印序号 取值范围:[0,7]
	task_seq	该job的添加task的序号
	srcw	task输入图像的实宽
	srch	task输入图像的实高
	src_vir_w	task输入图像的虚宽
	src_vir_h	task输入图像的虚高
	srcformat	task输入图像的format
	src_rect_x	task输入图像操作区域起点的x坐标
	src_rect_y	task输入图像操作区域起点的y坐标
	src_rect_w	task输入图像操作区域的宽
	src_rect_h	task输入图像操作区域的高
	dstw	task输出图像的实宽
	dsth	task输出图像的实高
	dst_vir_w	task输出图像的虚宽
	dst_vir_h	task输出图像的虚高
	dstformat	task输出图像的format
	dst_rect_x	task输出图像操作区域起点的x坐标
	dst_rect_y	task输出图像操作区域起点的y坐标

参数		描述
	dst_rect_w	task输出图像操作区域的宽
	dst_rect_h	task输出图像操作区域的高
max waste time job info1 最近耗时最大的job的信息	各个成员同recent job info1的成员	最近500个任务中耗时最长的job信息，当任务数超过500的时候，会重置最大值。
max waste time job info2 最近耗时最大的job的信息	各个成员同recent job info2的成员	最近500个任务中耗时最长的job信息，当任务数超过500的时候，会重置最大值。
max waste time job info3 最近耗时最大的job的信息	各个成员同recent job info3的成员	最近500个任务中耗时最长的job信息，当任务数超过500的时候，会重置最大值。
tde job status TDE任务状态	success	累计成功处理的job数
	fail	累计处理失败的job数
	cancel	累计主动取消的job数
	all_job_num	TDE所有可用的任务数
	free_num	空闲的job数
	begin_num	用户已经创建但是未提交的job数量
	busy_num	用户已经提交但是未提交给硬件处理的任务数
	procng_num	正在进行硬件处理的任务数
tde task status TDE task状态	success	累计成功处理的task数量，如果一个job处理成功，那么job中包含的task也全部处理成功。Task处理成功的数量累加上成功job中的task数。
	fail	累计处理失败的task数量，如果一个job处理失败，那么job中包含的task也全部处理失败，Task处理失败的数量累加上失败job的task数。
	cancel	累计处理cancel的task数量，如果一个jobcancel，那么job中包含的task也全部cancel，Task处理cancel的数量累加上canceljob的task数。
	all_task_num	TDE Task的总数，一般为200。
	free_num	空闲的task数
	busy_num	已添加到job下的task数

【调试命令】

- **数据录制**

录制tde job中task处理的输入和输出数据

例：启动录制，生成的数据存放在tmp路径下，最多录制10帧

```
./dumpsys tde record /tmp/ 10
```

14. VDEC

【调试信息】

```
[root@RK356X:/userdata]# ./dumpsys vdec
```

```
-----
```

```
DUMP OF SERVICE vdec:
```

```

----- MODULE PARAM -----
vdec_max_chn_num  mb_src
64      1

----- CHN COMM ATTR & PARAM -----
id   type     width    height    vir_width   vir_height  disp_mode  state
0    image:h265 640      360      640        384        1          start
1    image:h264 720      576      768        592        1          start
2    image:h264 720      576      768        592        1          start
id   send_mode  send_timeout set_user_pic en_user_pic attach_pool
0    frame      300        0        0          0
1    frame      300        0        0          0
2    frame      300        0        0          0

----- CHN VIDEO ATTR & PARAMS -----
id   compress  en_dei   en_mv    pixfmt
0    Y         N       Y        N/A
1    Y         N       Y        N/A
2    Y         N       Y        N/A

----- CHN STATE -----
id   send      send_ok   send_rate max_input_cnt left_input_cnt left_input_size max_output_cnt left_output_cnt
left_output_size err_status
0    8748     5094     25.00    10        8        37749     8        0        0        0
1    8658     5094     25.11    10        10       48005     8        0        0        0
2    8825     5095     25.00    10        10       46747     8        0        0        0

----- CHN DECODE BUFFER STATE -----
id   input_strm_cnt output_frm_cnt error_frm_cnt unused_buf_cnt put_buf_cnt
0    7134      6061      2        0        6057
1    7134      6061      2        0        6057
2    7134      6061      2        0        6057

-----
END DUMP OF SERVICE vdec:

```

【调试信息分析】

记录当前视频解码属性配置以及状态信息。

【参数说明】

参数		描述
module param	vdec_max_chn_num	VDEC 支持的最大解码通道数。
	mb_src	解码帧存分配方式。 1: 模块 MODULE; 2: 私有 PRIVATE; 3: 用户 USER。
chn comm attr & params	id	解码通道号。
	type	解码通道类型。 image:h264; image:h265; image/jpeg;
	width	解码图像宽度。
	height	解码图像高度。
	vir_width	解码图像虚宽。
	vir_height	解码图像虚高。
	dispMode	显示模式 (0: 实时模式, 1: 回放模式)
	state	播放状态 (start: 通道开启, stop: 通道停止)
	send_mode	解码通道码流发送模式。FRAME: 按帧发送, STREAM: 流式发送
	send_timeout	解码通道码流等待超时时间, 单位ms, -1 为永久阻塞
	set_user_pic	是否设置了用户图片。
	en_user_pic	是否使能用户图片。
	attach_pool	是否使用外部内存池
CHN VIDEO ATTR & PARAMS	id	解码通道号
	compress	压缩模式 (0: 非压缩模式, 1: 压缩模式)
	en_dei	是否使能deinterlace
	en_mv	是否使能colmv
	pixfmt	jpeg输出格式: N/A (非jpeg)
CHN STATE	id	解码通道号
	send	发送码流次数
	send_ok	发送码流成功次数
	send_rate	发送码流帧率
	max_input_buf_cnt	输入最大缓存个数
	left_input_buf_cnt	未使用的输入buffer个数
	left_input_size	未使用的输入buffer大小
	max_output_buf_cnt	输出最大缓存个数

参数		描述
	left_output_buf_cnt	剩余输出帧个数
	left_output_size	剩余输出帧大小
	err_status	通道错误码
CHN DECODE BUFFER STATE	id	解码通道号
	input_strm_cnt	解码器stream输入个数
	output_frm_cnt	解码器frame输出个数
	error_frm_cnt	解码器错帧个数
	unused_buf_cnt	解码器可用buf个数
	put_buf_cnt	归还解码器buf个数

15. VENC

【调试信息】

```
[root@RK356X:/userdata]# ./dumpsys venc
-----
DUMP OF SERVICE venc:
----- venc module_param -----
venc_max_chn_num buf_cache frm_buf_cyc 264_one_pkt 265_one_pkt jpeg_one_pkt
16      0      0      multi    one     one

----- venc chn attr -----
id width height vir_w vir_h codec_type pix_format buf_count buf_size rc_mode gop_mode gop
0 1920 1080 1920 1080 8      image:nv12 5      3110400 H264CBR NORMALP 60
vir_idr_len
0

----- venc chn query stat -----
id left_pics left_strm_bytes left_strm_frms cur_packs left_recv_pics
0   0        0        0        0        0

----- venc chn rc info1 -----
id start_qp step_qp max_qp min_qp max_i_qp min_i_qp delt_ip_qp qfactor qf_min qf_max
0 26   8   51   10   46   24   2   0   0   0

----- venc chn rc info2 -----
id stat_time set_min_bps set_avg_bps set_max_bps get_min_bps get_avg_bps get_max_bps
0 30   10000000 10240000 10240000 9085631 10546106 10761830

----- venc chn roi info -----
id index is_intra abs_qp qp x y width height

----- venc chn osd info -----
id region total_size enable inverse start_x start_y width height offset

----- venc chn dump status -----
id in_fps out_fps seq snap_set attach attach_cnt attach_size fail_strm_cnt
0 24.05 23.98 207 10000 N 0 0 0

----- venc chn param -----
id crop_mode src_x src_y src_w src_h dst_x dst_y dst_w dst_y fps_en src_fps_set dst_fps_set
0 none 0 0 0 0 0 0 0 0/0 0/0

----- venc chn frame dump info1 -----
```

```

id width height vir_w vir_h hor_w ver_h format afbc_mode eos pts delay_us lost
0 1920 1080 1920 1080 1920 1080 image:nv12 0x0 0 0 10980 0

----- venc chn frame dump info2 -----
id rect_x rect_y rect_w rect_h send_idx send_ok_idx get_idx get_ok_idx now_cst_us avg_cst_us max_cst_us drop_cnt
0 0 0 0 60314 60314 60314 60313 4151 4154 4375 0

----- venc chn dump base config -----
id rotation mirror
0 0 none

----- venc chn dump h264 config -----
id intra_pred tran_mode chroma_qp entropy cabac_init dblk_dis dblk_a dblk_b full_range
0 0 0 -6 CABAC 0 0 0 0 Y

----- venc chn dump h265 config -----
id cb_qp cr_qp scaling_list dblk_dis dblk_a dblk_b sao_luma sao_cr pu_sis_en full_range

----- venc chn dump rc adv param -----
id clear_stat
0 1

----- venc chn dump super frm -----
id super_frm_mode rc_priority Iframe_bits_thr Pframe_bits_thr
0 0 0 0

----- venc chn dump frm lost -----
id frm_lost_open frm_lost_bps_thr frm_lost_mode enc_frm_gaps
0 0 0 0

----- venc chn dump intra refresh -----
id refresh_en refresh_mode refresh_num req_i_qp
0 0 0 0

----- venc chn dump hierarchical qp -----
id hier_qp_en hier_qp_delta hier_frame_num
0 0 0 0 0 0 0 0 0 0

----- venc chn dump de breath effect -----
id de_breath_en strength0 strength1
0 0 0 0

----- venc chn dump ref param -----
id base enhance en_preb
0 0 0 0

----- venc chn dump slice split -----
id enable mode size
0 Y 1 16

-----
END DUMP OF SERVICE venc:

```

【调试信息分析】

记录当前视频编码属性配置以及状态信息。

【参数说明】

参数		描述
venc module param	venc_max_chn_num	VENC 支持的最大编码通道数。
	buf_cache	码流获取是否支持 cache 方式。 (暂未使用)
	frm_buf_cyc	帧存是否回收。 (暂未使用)
	264_one_pkt	H264码流帧配置模式。 (one: 单包; multi: 多包)
	265_one_pkt	H265码流帧配置模式。 (one: 单包; multi: 多包)
	jpeg_one_pkt	mjpeg/jpeg码流帧配置模式。 (one: 单包; multi: 多包)
venc chn attr	id	编码通道号。
	width	编码图像宽度。
	height	编码图像高度。
	vir_width	编码图像虚宽。
	vir_height	编码图像虚高。
	codec_type	编码视频格式类型。 (H264/H265/JPEG/MJPEG)
	pix_format	编码的图像格式。
	buf_count	编码时的最大输出包缓冲个数。
	buf_size	编码时的单个输出buf大小。
	rc_mode	编码rc模式。
	gop_mode	编码gop模式。
	gop	编码gop长度。
	vir_idr_len	编码虚拟I帧长度。
venc chn query stat	id	编码通道号。
	left_pics	待编码的图像数。
	left_strm_bytes	码流 buffer 剩余的 byte 数。
	left_strm_frms	码流 buffer 剩余的帧数。
	cur_packs	当前帧的码流包个数。
	left_recv_pics	剩余待接收的帧数
venc chn rc info1	id	编码通道号。
	start_qp	编码起始帧qp。
	step_qp	编码qp步进。
	max_qp	编码最大qp。
	min_qp	编码最小qp。
	max_i_qp	编码I帧最大qp。
	min_i_qp	编码I帧最小qp。
	delt_ip_qp	编码I帧前5帧p帧qp的平均值与这个I帧的qp差值。
	qfactor	编码MJPEG/JPEG的质量参数。
	qf_min	编码MJPEG的最小质量参数，CBR/VBR码控时生效。

参数		描述
	qf_max	编码MJPEG的最大质量参数，CBR/VBR码控时生效。
venc chn rc info2	id	编码通道号。
	stat_time	编码码率统计时间，单位秒。
	set_min_bps	编码码率设置最小值，单位kbps。
	set_avg_bps	编码码率设置平均值，单位kbps。
	set_max_bps	编码码率设置最大值，单位kbps。
	get_min_bps	获取编码码率最小值，单位kbps。十秒更新一次。
	get_avg_bps	获取编码码率平均值，单位kbps。十秒更新一次。
	get_max_bps	获取编码码率最大值，单位kbps。十秒更新一次。
venc chn roi info	id	编码通道号。
	index	ROI 区域的索引。
	is_intra	是否为I帧。
	abs_qp	ROI 区域 QP 模式。N：相对 QP。Y：绝对 QP。
	qp	QP 值。
	x	ROI 区域水平起始点。
	y	ROI 区域垂直起始点。
	width	ROI 区域宽度。
	height	ROI 区域高度。
venc chn osd info	id	编码通道号。
	region	OSD区域序号，范围[0,7]。
	total_size	OSD总buf大小。
	enable	该区域OSD是否使能，N：不使能。Y：使能。
	inverse	该区域OSD是否反色，N：不反色。Y：反色。
	start_x	该区域OSD起始偏移X。
	start_y	该区域OSD起始偏移Y。
	width	该区域OSD宽度。
	height	该区域OSD高度。
	offset	该区域OSD在总buf的偏移位置。
venc chn dump status	id	编码通道号。
	in_fps	编码输入帧率。
	out_fps	编码输出帧率。
	seq	编码输出序列号。
	snap_set	编码输出帧数设置。
	attach	输出buf是否为attach pool，Y：是；N：否。
	attach_cnt	attach pool buf个数。

参数		描述
	attach_size	attach pool buf大小。
	fail_strm_cnt	输出码流获取失败帧数（多包模式下，输出packet_cnt小于当前帧输出包个数时会导致获取失败，此时会进行丢帧处理）。
venc chn param	id	编码通道号。
	crop_mode	编码裁剪模式。 none：不开启 crop_only：只裁剪 crop_scale：裁剪并缩放。
	src_x	crop_only：裁剪起始位置横坐标x； crop_scale：裁剪缩放起始位置横坐标x。
	src_y	crop_only：裁剪起始位置纵坐标y； crop_scale：裁剪缩放起始位置纵坐标y。
	src_w	crop_only：裁剪图像宽度w； crop_scale：裁剪缩放图像宽度w。
	src_h	crop_only：裁剪图像高度h； crop_scale：裁剪缩放图像高度h。
	dst_x	crop_scale：裁剪缩放目标位置横坐标x。
	dst_y	crop_scale：裁剪缩放目标位置纵坐标y。
	dst_w	crop_scale：裁剪缩放目标图像宽度w。
	dst_y	crop_scale：裁剪缩放目标图像高度h。
	fps_en	编码帧率控制使能；0：不使能；1：使能。
	src_fps_set	编码帧率控制输入帧率设置值。
	dst_fps_set	编码帧率控制输出帧率设置值。
venc chn frame dump info1	id	编码通道号。
	width	编码输入帧实际宽度。
	height	编码输入帧实际高度。
	vir_w	编码输入帧实际内存宽度。
	vir_h	编码输入帧实际内存宽度。
	hor_w	编码输入帧根据格式转换后的内存宽度。
	ver_h	编码输入帧根据格式转换后的内存高度。
	format	编码输入帧实际格式。
	afbc_mode	编码输入压缩模式。 0x0：无压缩；0x100000：AFBC V1；0x200000：AFBC V2。
	eos	编码输入帧结束符标志。1：结束符；0：非结束符。
	pts	编码输入帧pts。
	delay_us	输入帧到编码输出后的延时（从该帧最初打上pts到编码结束的全部延时）。
	lost	编码输入帧丢帧数。
venc chn frame dump info2	id	编码通道号。
	rect_x	编码输入帧区域信息偏移x。

参数		描述
	rect_y	编码输入帧区域信息偏移y。
	rect_w	编码输入帧区域信息宽度w。
	rect_h	编码输入帧区域信息高度h。
	send_idx	编码送帧数。
	send_ok_idx	编码送帧成功数。
	get_idx	编码获取帧数。
	get_ok_idx	编码获取帧成功数。
	now_cst_us	最近一帧编码耗时，单位us。
	avg_cst_us	最近十帧的平均编码耗时，单位us。
	max_cst_us	最大编码耗时，单位us。
	drop_cnt	通道复位后丢弃的输出码流帧总数（通道复位后，未取走的编码数据将会丢弃处理）。
venc chn dump base config	id	编码通道号。
	rotation	编码通路旋转，0/90/180/270度或illegal：非法参数。
	mirror	编码通路镜像，horizontal：水平镜像；vertical：垂直镜像；both：水平+垂直镜像；none：无镜像；illegal：非法参数。
venc chn dump h264 config	id	h264编码通道号。
	intra_pred	编码帧内预测属性； 0：关闭，无限制预测 1：开启限制预测模式。
	tran_mode	编码通道的变换、量化的属性； 0：支持trans4x4、trans8x8 1：支持trans4x4。
	chroma_qp	编码色彩偏移值。
	entropy	编码熵编码属性；0：cavlc 1：cabac
	cabac_init	编码用于决定关联变量的初始化过程中使用的初始化表格的序号（0-2）。
	dblk_dis	编码去块效应滤波器的操作在经过条带的一些块边缘时是否会被废弃，并指定该滤波器针对哪个边缘被废弃。 0：使能 1：不使能 2：在片分界处不使能。
	dblk_a	编码访问 alhpa 和 tC0 去块效应滤波器表格来滤波条带中的宏块所控制的操作使用的偏移。
	dblk_b	编码访问 beta去块效应滤波器表格来滤波带中的宏块所控制的操作使用的偏移。
	full_range	H264编码是否为full color range，Y：是；N：否。
venc chn dump h265 config	id	h265编码通道号。
	cb_qp	编码色彩蓝色浓度偏移值。
	cr_qp	编码色彩红色浓度偏移值。
	scaling_list	编码通道变换量化表使能。0：不使能 1：使能。

参数		描述
	dblk_dis	编码去块效应滤波器的操作在经过条带的一些块边缘时是否会被废弃，并指定该滤波器针对哪个边缘被废弃。 0：使能 1：不使能 2：在片分界处不使能。
	dblk_a	编码访问 alhpa和 tC0 去块效应滤波器表格来滤波条带中的宏块所控制的操作使用的偏移。
	dblk_b	编码访问 beta去块效应滤波器表格来滤波带中的宏块所控制的操作使用的偏移。
	sao_luma	当前 slice 亮度分量是否作 Sao 滤波；0：不使能； 1：使能。
	sao_cr	当前 slice 色度分量是否作 Sao 滤波；0：不使能； 1：使能。
	pu_sis_en	CVS的内部预测滤波过程中有条件地使用双线性插值；0：不使用；1：使用。
	full_range	H265编码是否为full color range，Y：是；N：否。
venc chn dump rc adv param	id	编码通道号。
	clear_stat	编码设置新的通道码率后，是否清除码率控制的统计信息； 0：关闭清除统计信息 1：开启清除统计信息。
venc chn dump super frm	id	编码通道号。
	super_frm_mode	编码超大帧处理模式；0：不处理 1：丢帧 2：重编该帧。
	rc_priority	编码超大帧重编优先级设置； 0：super_frm_mode为0时该值为0，表示未开启超大帧处理； 1：目标码率优先； 2：超大帧阈值优先。
	Iframe_bits_thr	编码I帧超大阈值bit设置。
	Pframe_bits_thr	编码P帧超大阈值bit设置。
venc chn dump frm lost	id	编码通道号。
	frm_lost_open	编码丢帧开关，0关闭； 1：开启。
	frm_lost_bps_thr	编码丢帧阈值百分比。
	frm_lost_mode	编码丢帧模式，0：瞬时码率超过阈值时正常丢帧； 1：瞬时码率超过阈值时编码 pskip 帧。
	enc_frm_gaps	编码丢帧最大允许连续丢帧帧数。0表示全丢。
venc chn dump intra refresh	id	编码通道号。
	refresh_en	编码是否使能刷Islice功能，0：不使能； 1：使能。
	refresh_mode	I宏块刷新模式，分为按行刷新和按列刷新，0：按行刷新； 1：按列刷新。
	refresh_num	每次I宏块刷新行数或者列数。
	req_i_qp	I帧QP值。暂未使用。
venc chn dump hierarchical qp	id	编码通道号。
	hier_qp_en	QP分层是否使能，0：不使能； 1：使能。

参数		描述
	hier_qp_delta	第0层P帧的相对于每一层的Qp差值QpDelta。
	hier_frame_num	每一层对应P帧的数目。
venc chn dump de breath effect	id	编码通道号。
	de_breath_en	去除呼吸效应是否使能，0：不使能；1：使能。
	strength0	去除呼吸效应强度调节参数0。暂未使用。
	strength1	去除呼吸效应强度调节参数1。
venc chn dump ref param	id	编码通道号。
	base	高级跳帧base层的周期。
	enhance	高级跳帧enhance层的周期。
	en_preb	base层的帧是否被base层其他帧用作参考，0：等同于base设置为无限大；1：base层的所有帧都参考IDR帧
venc chn dump slice split	id	编码通道号。
	enable	是否开启slice分割功能。 Y：是；N：否。
	mode	slice分割模式。 0：按字节分割；1：按MB或LCU分割。
	size	slice分割大小。 split_mode=0，表示每个slice的byte数； split_mode=1，表示每个slice的MB或LCU数。

16. AVS

【调试信息】

```
[root@RK3588:/]# dumpsys avs
# dumpsys avs
-----
DUMP OF SERVICE avs:
----- avs mod param -----
work_set_size mb_source
68608 PRIVATE
----- avs group attr -----
grp_id mode enable pipe_num is_sync src_rate dst_rate
0 BLEND Y 4 Y -1 -1
----- avs input attr -----
grp_id param_source
0 CALIB

calib_path mesh_alpha_path
0 /data/dahua-MPA_Pano/tmp/rkipc_avs_mesh/
.pto
----- avs output attr -----
grp_id proj_mode center_x center_y fov_x fov_y
0 EQUIRECTANGULAR 2779 544 19834 10106

ori_yaw ori_pitch ori_roll yaw pitch roll
0 0 0 0 0 0 0
```

```

----- avs channel attr -----
grp_id chn_id enable width height is_compress dym_range frm_cnt depth src_rate dst_rate
0 0 Y 5520 2700 Y SDR8 8 1 -1 -1
----- avs group work status -----
grp_id cst_time max_cst_time prc_into rty_align rty_psh prc_suc start_false stitch_into stitch_succ
0 12005 13846 546500 408085 65 138415 0 138415 138415
----- avs pipe work status -----
grp_id pipe_id_0 pipe_id_1 pipe_id_2 pipe_id_3 pipe_id_4 pipe_id_5 pipe_id_6 pipe_id_7
0 152055 138481 152121 152120
13662 42 13705 13706

grp_id pipe_id send_frm_cnt lost_frm_cnt send_frm_rate
0 0 0 30.00
1 0 0 30.00
2 0 0 30.00
3 0 0 30.00
----- avs pipe queue -----
grp_id pipe_id_0 pipe_id_1 pipe_id_2 pipe_id_3 pipe_id_4 pipe_id_5 pipe_id_6 pipe_id_7
0
Y(152146) Y(152146)

```

```

----- avs chn work status -----
grp_id chn_id get_frm_cnt rel_frm_cnt get_frm_rate lft_in_cnt lft_out_cnt
0 0 0 0.00 0 1
-----
```

[root@RK3588:/]# dumpsys avs

DUMP OF SERVICE avs:

```

----- avs mod param -----
work_set_size mb_source
68608 PRIVATE
----- avs group attr -----
grp_id mode enable pipe_num is_sync src_rate dst_rate
0 BLEND Y 4 Y -1 -1
----- avs input attr -----
grp_id param_source
0 LUT

lut_data_acc lut_step_x lut_step_y fuse_width
HIGH LOW (64) LOW (64) LOW (512)
vir_addr[0] vir_addr[1] vir_addr[2] vir_addr[3]
0x7f3af26000 0x7f39335000 0x7f38f74000 0x7f383a3000
----- avs output attr -----
grp_id proj_mode center_x center_y fov_x fov_y
0 EQUIRECTANGULAR 2779 544 19834 10106

ori_yaw ori_pitch ori_roll yaw pitch roll
0 0 0 0 0 0 0

```

```

----- avs channel attr -----
grp_id chn_id enable width height is_compress dym_range frm_cnt depth src_rate dst_rate
0 0 Y 5520 2700 Y SDR8 8 1 -1 -1
----- avs group work status -----
grp_id cst_time max_cst_time prc_into rty_align rty_psh prc_suc start_false stitch_into stitch_succ
0 8376 9627 1336 1001 14 335 0 335 335
----- avs pipe work status -----
grp_id pipe_id_0 pipe_id_1 pipe_id_2 pipe_id_3 pipe_id_4 pipe_id_5 pipe_id_6 pipe_id_7
0 336 335 335 335
1 0 0 0

```

```

grp_id pipe_id send_frm_cnt lost_frm_cnt send_frm_rate
0 0 0 30.00
1 0 0 30.00
2 0 0 30.00
3 0 0 30.00

```

```
----- avs pipe queue -----
grp_id pipe_id_0 pipe_id_1 pipe_id_2 pipe_id_3 pipe_id_4 pipe_id_5 pipe_id_6 pipe_id_7
0
```

```
----- avs chn work status -----
grp_id chn_id get_frm_cnt rel_frm_cnt get_frm_rate lft_in_cnt lft_out_cnt
0      0       0       0.00      0       1
```

```
-----  
END DUMP OF SERVICE avs:
```

【调试信息分析】

记录当前全景拼接属性配置以及状态信息。

【参数说明】

参数		描述
avs mod param	work_set_size	AVS 的工作区的大小，单位为KB。
	mb_source	AVS 通道内存缓冲来源。 COMMON: 公共内存缓冲池； PRIVATE: 私有内存缓冲池； USER: 用户内存缓冲池。
avs group attr	grp_id	Grp ID 号。
	mode	AVS 工作模式。
	enable	Grp 是否启动。 N: 关闭； Y: 启动。
	pipe_num	Pipe 数目。
	is_sync	是否同步各 Pipe 图像。 N: 关闭； Y: 启动。
	src_rate	Grp 源帧率。
	dst_rate	Grp 目标帧率。
avs input attr	grp_id	Grp ID 号。
	param_source	拼接输入参数来源。 CALIB: 参数来自查找表； LUT: 参数来自查找表。
	calib_path	标定文件路径。
	mesh_alpha_path	Final Lut文件路径。
	lut_data_acc	查找表精度。 HIGH: 高精度； MEDIUM: 中精度； LOW: 低精度。
	lut_step_x	查找表在 X 方向上的采样步长。
	lut_step_y	查找表在 Y 方向上的采样步长。
	fuse_width	查找表的低频层融合带宽度。
	vir_addr[0] vir_addr[1] vir_addr[] ...	Final Lut文件路径。
avs out attr	grp_id	Grp ID 号。
	proj_mode	输出的投影模式。
	center_x	投影中心在输出图的位置的 X 坐标。
	center_y	投影中心在输出图的位置的 y 坐标。
	fov_x	水平方向上的视场角。
	fov_y	垂直方向上的视场角。
	ori_yaw	原始偏航角。
	ori_pitch	原始俯仰角。
	ori_roll	原始翻滚角。

参数		描述
	yaw	偏航角。
	pitch	俯仰角。
	roll	翻滚角。
avs channel attr	grp_id	Grp ID 号。
	chn_id	Chn ID 号。
	enable	Chn 是否启动。 N: 关闭； Y: 启动。
	Width	通道宽度。
	height	通道高度。
	is_compress	是否使能压缩。 N: 关闭； Y: 启动。
	dym_range	输出图像的动态范围。
	frm_cnt	最大通道图像缓冲个数。
	depth	用户获取通道图像的队列深度。
	src_rate	通道帧率控制的源帧率。
	dst_rate	通道帧率控制的目标帧率。
avs group work status attr	grp_id	Grp ID 号。
	now_cst_us	当前完成任务的耗时 (us)。
	avg_cst_us	最近10次任务的平均耗时 (ms)。
	max_cst_us	历史上耗时最长任务的执行时间 (us)。
	now_itv_us	当前任务执行与上次任务执行的间隔时间 (us)。
	run_cnt	历史上已执行的任务次数。
	suc_run_cnt	历史上执行成功的任务次数。
	fail_run_cnt	历史上执行失败的任务次数。
avs pipe work status	grp_id	Grp ID 号
	Pipe0 Pipe1 Pipe2 ...	对应输入Pipe 上的接收的图像数/对应输入Pipe 上的丢弃的图像数
avs pipe queue	grp_id	Grp ID 号
	Pipe0 Pipe1 Pipe2 ...	对应输入Pipe 上的缓冲队列状态
avs chn work status	grp_id	Grp ID 号
	chn_id	Chn ID 号。
	get_frm_cnt	用户通过RK_MPI_AVG_GetChnFrame取走的帧数。

参数		描述
	rel_frm_cnt	用户通过RK_MPI_AVN_ReleaseChnFrame还回的帧数。
	get_frm_rate	用户通过RK_MPI_AVN_GetChnFrame取走的帧数。
	lft_in_cnt	待处理的通道输入缓存帧数。
	lft_out_cnt	待取走的通道输出缓存帧数。

【调试命令】

- **数据录制**

进行数据录制时，保存全部的输入数据和经过拼接处理的输出数据，支持保存非压缩 / AFBC压缩数据。例如：进行 6路数据拼接并有 1路输出时，保存数据到同一个目录下的 7个文件中。以下指令，只能在程序正常运行时使用。

指令举例

使能对AVS Grp 0的数据录制 3次，并采集到的数据保存到 /userdata/avs_record_[grp]_[chn]_[n]/（n：指令序号）目录下，

```
./dumpsys avs record 0 3 2 /userdata/
```

参数解析：

“0”：AVS Grp ID 号

“3”：3帧数据

“2”：输入数据和经过拼接处理的输出数据；可用参数：0 输入数据；1 经过拼接处理的输出数据；2：输入数据和经过拼接处理的输出数据。

“/userdata/”：输出文件目录；会在该输出文件目录下创建一个新的目录“avs_record_[grp]_[chn]_[n]”

文件命名规则

pipe_input_[grp]_[pipe]_[w]x[h]_[syncID].bin

chn_output_[grp]_[chn]_[w]x[h]_[syncID].bin

- **帧同步检查**

检查拼接合成时，来自 VI 的输入图像是否具有相同帧序号。以下指令，需要在程序正常运行时使用。

◦ 帧同步检查开启

指令举例

打开AVS Grp 0的帧同步检查

```
./dumpsys avs open_sync_debug 0
```

◦ 帧同步检查关闭

指令举例

关闭AVS Grp 0的帧同步检查

```
./dumpsys avs close_sync_debug 0
```

串口打印输出举例

```
AVS Input Frame Info:  
PipeId:0, Seq:189, Y0-Y5: 00 c0 03 a0 12  
PipeId:1, Seq:189, Y0-Y5: 00 c0 03 b0 98  
PipeId:2, Seq:189, Y0-Y5: 00 c0 03 70 d4  
PipeId:3, Seq:189, Y0-Y5: 00 c0 03 b0 58  
PipeId:4, Seq:189, Y0-Y5: 00 c0 03 90 52  
PipeId:5, Seq:189, Y0-Y5: 00 c0 03 b0 58
```

打印内容包括：Pipe 号码、帧序号以及帧前 5字节的数据。

17. GDC

【调试信息】

```
dumpsys gcdc

DUMP OF SERVICE gcdc:
----- gcd module params -----
g_max_job_num  g_max_task_num
128          200

----- recent job info -----
seq_no job_hdl correct correct_ex q_src q_src_a q_pano q_pano_a state  task_num in_size out_size cost_time_us
hw_time_us
0   0   N   N   N   Y   N   N   proced  1   0   0
4791  0
1   0   N   N   N   Y   N   N   proced  1   0   0
5082  0
2   0   N   N   N   Y   N   N   proced  1   0   0
4744  0
3   0   Y   N   N   N   N   N   proced  1   2073600 2073600 3061   0
4   0   N   N   N   Y   N   N   proced  1   0   0
4783  0
5   0   N   N   N   Y   N   N   proced  1   0   0
5096  0
6   0   N   N   N   Y   N   N   proced  1   0   0
4755  0
7   0   Y   N   N   N   N   N   proced  1   2073600 2073600 3057   0

----- max waste time recent job info -----
seq_no job_hdl correct correct_ex q_src q_src_a q_pano q_pano_a state  task_num in_size out_size
0   0   N   N   N   Y   N   N   proced  1   0   0
cost_time_us hw_time_us
5559  0

----- gcd latest task stat params -----
job task type      stat auto_alloc step_x step_y
0 100 query_src_a busy Y     32   32

----- gcd history task common params -----
job task type      img_w img_h max_cost_us new_cost_us new_process_cnt all_process_cnt
0 100 correct    1920 1080 3624   3546   405   405
0 100 query_src_a 1920 1080 5658   5009   1215   1215

----- gcd history correct common params -----
job task all_region enable lmf en_bg_color bg_color hor_offset ver_offset trap_coef fan_strength
0 100 4   Y   N   Y   0   33   5   0   0
mount_mode
wall

----- gcd history correct region params -----
job task region view_mode in_rad out_rad pan tilt h_zoom v_zoom out_x out_y out_w out_h
0 100 0   none  0   442 180 180 4095 4095 0   0   960 540
0 100 1   normal 0   442 223 180 4095 4095 960 0   960 540
0 100 2   normal 0   442 180 250 4095 4095 0   540 960 540
0 100 3   normal 0   442 180 153 404 404 960 540 960 540

----- gcd history query region params -----
job task query_type point_num pano_region
0 100 query_src_a 3000  0

----- gcd job status -----
success fail  cancel all_job_num free_num begin_num busy_num procing_num
1620  0   0   128   128  0   0   0

----- gcd task status -----
success fail  cancel all_task_num free_num busy_num
1620  0   0   200   199  1
```

END DUMP OF SERVICE gcd:

【调试信息分析】

记录GDC模块最近完成的若干任务、最近耗时最大的认证、历史累计信息等。

【参数说明】

参数		描述
module para	g_max_job_num	最大的job数
	g_max_task_num	最大的task数
recent job info 最近完成的job的信息	seq_no	打印序号 取值范围:[0,7]
	job_hdl	该job的handle号
	correct	该job是否有鱼眼矫正任务 (N: 否, Y: 是)
	correct_ex	该job是否有鱼眼矫正 (XY模式) 任务 (N: 否, Y: 是)
	q_src	该job是否有矫正图像获取原图位置单点坐标任务 (N: 否, Y: 是)
	q_src_a	该job是否有矫正图像获取原图位置多点坐标任务 (N: 否, Y: 是)
	q_pano	该job是否有矫正图像获取全景图像位置单点坐标任务 (N: 否, Y: 是)
	q_pano_a	该job是否有矫正图像获取全景图像位置多点坐标任务 (N: 否, Y: 是)
	task_num	该job包含的task数目
	state	该job的处理状态
	in_size	该job下各task的输入图像面积之和, 单位像素
	out_size	该job下各task的输出图像面积之和, 单位像素
	cost_time	该job从提交 (end_job) 开始到成功完成的耗时时长。单位us
	hw_time	该job在硬件中处理的耗时时长。单位us
max waste time job info 最近耗时最大的job的信息	各个成员同recent job info的成员	最近200个任务中耗时最长的job信息, 当任务数超过200的时候, 会重置最大值。
gdc latest task stat params 最新任务信息	job	job id
	task	task id
	type	任务类型; none: 无; correct: 鱼眼矫正; correct_ex: 鱼眼矫正 (XY模式) ; query_src: 纠正图像获取原图位置单点坐标; query_src_a: 纠正图像获取原图位置多点坐标; query_pano: 纠正图像获取全景图位置单点坐标; query_pano_a: 纠正图像获取全景图位置多点坐标。
	stat	状态; idle: 空闲状态; busy: 忙状态。
	auto_alloc	task id是否自动分配 (N: 否, Y: 是)
	step_x	鱼眼矫正横线步进值X
	step_y	鱼眼矫正纵向步进值Y
gdc history task common params 历史任务公共信息	job	job id

参数		描述
	task	task id
	type	任务类型; none: 无; correct: 鱼眼矫正; correct_ex: 鱼眼矫正（XY模式）; query_src: 纠正图像获取原图位置单点坐标; query_src_a: 纠正图像获取原图位置多点坐标; query_pano: 纠正图像获取全景图位置单点坐标; query_pano_a: 纠正图像获取全景图位置多点坐标。
	img_w	输入图像宽度
	img_h	输入图像高度
	max_cost_us	任务最大耗时，单位us
	new_cost_us	最后一次任务耗时，单位us
	new_process_cnt	任务修改参数次数
	all_process_cnt	任务运行次数
gdc history correct common params历史任务鱼眼矫正公共信息	job	job id
	task	task id
	all_region	一幅图像的矫正区域数目
	enable	是否使能（N: 否，Y: 是）
	lmf	是否使用用户设置的鱼眼镜头 LMF 参数（N: 否，Y: 是）
	en_bg_color	是否在输出图像打上背景色（N: 否，Y: 是）
	bg_color	背景色的颜色 RGB888 格式
	hor_offset	镜头中心点相对于 SENSOR 中心点的水平偏移
	ver_offset	镜头中心点相对于 SENSOR 中心点的垂直偏移
	trap_coeff	梯形矫正强度系数。用于壁装时的俯仰角矫正
	fan_strength	扇矫正强度系数，仅在 180 模式时有效
	mount_mode	鱼眼矫正安装模式
gdc history correct region params历史任务鱼眼矫正区域信息	job	job id
	task	task id
	region	矫正区域id
	view_mode	该校正区域的校正模式; 360_pano: 360全景模式; 180_pano: 180全景模式; normal: normal矫正模式; none: 不做矫正模式，对原始图缩放输出。
	in_rad	360 全景模式表示该矫正区域所对应原图的内半径，其他模式无效

参数		描述
	out_rad	360 全景模式表示该矫正区域所对应原图的外半径， 其他模式为矫正区域的可视半径
	pan	该矫正区域 PTZ 参数的 Pan 值
	tilt	该矫正区域 PTZ 参数的 Tilt 值
	h_zoom	该矫正区域 PTZ 参数的水平 Zoom 值
	v_zoom	该矫正区域 PTZ 参数的垂直 Zoom 值
	out_x	该矫正区域的输出位置X
	out_y	该矫正区域的输出位置Y
	out_w	该矫正区域的输出宽度
	out_h	该矫正区域的输出高度
gdc history query region params 历史任务查询位置信息	job	job id
	task	task id
	query_type	查询类型； query_src: 纠正图像获取原图位置单点坐标； query_src_a: 纠正图像获取原图位置多点坐标； query_pano: 纠正图像获取全景图位置单点坐标； query_pano_a: 纠正图像获取全景图位置多点坐标。
	point_num	查询点数
	pano_region	全景区域id
gdc job status	success	累计成功处理的job数
	fail	累计处理失败的job数
	cancel	累计主动取消的job数
	all_job_num	GDC所有可用的任务数
	free_num	空闲的job数
	begin_num	用户已经创建但是未提交的job数量
	busy_num	用户已经提交但是未提交给硬件处理的任务数
	procing_num	正在进行硬件处理的任务数
gdc task status VGS task状态	success	累计成功处理的task数量，如果一个job处理成功，那么job中包含的task也全部处理成功。Task处理成功的数量累加上成功job中的task数。
	fail	累计处理失败的task数量，如果一个job处理失败，那么job中包含的task也全部处理失败，Task处理失败的数量累加上失败job的task数。
	cancel	累计处理cancel的task数量，如果一个jobcancel，那么job中包含的task也全部cancel，Task处理cancel的数量累加上canceljob的task数。
	all_task_num	GDC Task的总数，一般为200。
	free_num	空闲的task数
	busy_num	已添加到job下的task数

18. ALL

【调试信息】

```
dumpsys all
-----
DUMP OF SERVICE all:
rgn:
----- region status of overlay -----
hdl type used pixel_format width height mb virt

----- region chn status of overlay -----
hdl type mod dev chn is_show x y fg_alpha bg_alpha layer

----- region status of cover -----
hdl type used

----- region chn status of cover -----
hdl type mod dev chn is_show x y width height color layer coord_type

----- region status of mosaic -----
hdl type used

----- region chn status of mosaic -----
hdl type mod dev chn is_show x y width height blk_size layer

sys:
----- bind relation table -----
src_mod src_dev src_chn dst_mod dst_dev dst_chn send_cnt
vdec 0 0 vo 0 0 182

venc:
----- venc module_param -----
venc_max_chn_num
16

----- venc chn attr -----
id width height vir_w vir_h codec_typepix_format stream_buf_count

----- venc chn query stat -----
id left_pics left_strm_bytes left_strm_frms cur_packs left_recv_pics

----- venc chn roi info -----
id index is_intra abs_qp qp x y width height

vo:
-----DEV CONFIG-----
DevId DevEn InfType InfSync
0 Y HDMI 1920x1080p50
1 Y EDP 1024x768p60
2 N Unkown Unkown
-----LAYER BIND CONFIG-----
DevId Video Gfx Cursor
0 clu0 esm0 sm0
1 clu1 esm1 sm1
2 unkown unkown unkown
-----LAYER STATUS-----
LayId LayEn PixFmt ImgW ImgH DispW DispH FrmRt BufLens
0 Y RGBA32 1920 1080 1920 1080 25 3
1 N NV12 0 0 0 0 0 0
2 Y RGBA32 1024 768 1024 768 25 3
3 N NV12 0 0 0 0 0 0
4 N NV12 0 0 0 0 0 0
```

```

5 N NV12 0 0 0 0 0 0
6 N NV12 0 0 0 0 0 0
7 N NV12 0 0 0 0 0 0
LAYER clu0 CHANNEL STATUS:
ChnlId Prio X Y W H ChnFrft Show Pause Step Cache
0 1 0 0 480 270 25 Y N N 8
LAYER clu1 CHANNEL STATUS:
ChnlId Prio X Y W H ChnFrft Show Pause Step Cache
WBC STATUS:
WbcId En Src W H Fmt Frt Depth
0 N Dev0 0 0 NV12 0 0

vpss:
----- vpss group attr -----
grp_id max_w max_h pixel_format dym_range src_rate dst_rate is_compress

----- vpss channel attr -----
grp_id chn_id mode width height dym_range src_rate dst_rate depth align mirror flip

----- vpss group crop info -----
grp_id crop_en coor_type x y width height

----- vpss chn crop info -----
grp_id chn_id crop_en coor_type x y width height

----- vpss group pic queue -----
grp_id delay backup

----- vpss group pic info -----
grp_id width height vir_w vir_h pix_format dyn_range compress

----- vpss chn output resolution -----
grp_id chn_id width height vir_w vir_h pix_format dyn_range compress send_ok frm_rate

----- vpss chn rotation attr -----
grp_id chn_id rotate

-----
END DUMP OF SERVICE all:

```

【调试信息分析】

记录当前所有注册模块的配置和状态信息，使用命令dumpsys或者dumpsys all。

【参数说明】

具体参数说明参考各个注册模块的说明。